University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Collaboratory for Epidemiological Research (CEpiR)

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Acuzar, Carmel Francesca, Aguila
2007-09504

April, 2011

ACCEPTANCE SHEET

The special problem entitled "*Collaboratory for Epidemiological Research*" prepared and submitted by *Carmel Francesca Aguila Acuzar* in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Richard Bryann L. Chua, M.Sc.
Adviser

EXAMINERS	APPROVED	DISAPPROVED
1. Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Aldrich Colin K. Co., M.Sc. (candidate)	_____	_____
4. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____
6. Geoffrey A. Solano, M.Sc.	_____	_____
7. Bernie B. Terrado, M.Sc. (candidate)	_____	_____

_____
Date

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science

_____
Geoffrey A. Solano, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____
Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____
Reynaldo H. Imperial, Ph.D.
Dean
College of Arts and Sciences

ABSTRACT

Even though there is a need for collaboration in the field of epidemiology, there is no existing system that allows Philippine-based researchers and epidemiologists to collaborate and share data and information among themselves. This need is addressed by the Collaboratory for Epidemiological Research (CEpiR) by providing an interactive system where researchers and epidemiologists can join projects and have a common storage for data and information.

CEpiR enables researchers and epidemiologists to search for projects listed in the system thus avoiding research repetition. Also, the system allows researchers and epidemiologists to request for membership in any project. Once approved, the system will enable these users to manage and share epidemiological data, references and other relevant materials for the project with their research partners.

Keywords: Collaboratory, Epidemiological Data, Epidemiological Research

TABLE OF CONTENTS

TABLE OF FIGURES

I. INTRODUCTION

A. Background of the Study

Collaboration is working together to achieve a common goal. [1] This has been evident in the scientific world ever since 40 years ago or more [2] and continues to be popular nowadays. But, unlike before that collaboration takes the form of co-authorships only [2], it now also takes the form of collaboratories where collaboration is no longer bounded by time and space.

Collaboration has many benefits thus making it popular: first, it ensures effective use of individual talents. Second, it allows the transfer of knowledge or skills, especially tacit ones. Third, brain-storming activities can create new insights or perspectives that might not have been thought of if done individually. Fourth, it provides companionship. Fifth, it extends a researcher's network since he/she can have contacts from other disciplines or institutions. Finally, it enhances the dissemination of findings making it easier for others to locate it. [3]

But even though collaboration poses many benefits, there is a lack of software tools that will allow researchers to collaborate especially in the field of epidemiology. Collaboratories would be the solution but this technology is still in exploratory stages therefore, there is still no established recipe for success. Moreover, collaboratories are usually created specifically for a certain discipline making them difficult to generalize. There is also no single set of tools that can be proven relevant to collaboratories since issues on implementation are considered. [4]

B. Statement of the Problem

Even though there is a need for collaboration in the field of epidemiology, there is no existing system that integrates the country's epidemiological data for analysis and simulation and

lists the ongoing researches regarding epidemiology. Researchers have to manually distribute the data through email, mailing lists, etc. and as a consequence, the data will only be accessible to a small group of people. Also, researchers are unaware of the ongoing researches on epidemiology, thus, research repetition can occur.

Epidemiological data are already available in databases owned by the government or by different organizations such as the Department of Clinical Epidemiology of the University of the Philippines Manila. Unfortunately, these databases are heterogeneous and most are not accessible online. They are heterogeneous in the sense that they have different data models, meaning their data elements and relationships are defined differently and the same data may be represented differently in these databases. Both result from the different perspectives of the organizations regarding the data, the existence of different ways to construct data models, the incompatible design specifications [5] and the fact that there is no uniform format for defining epidemiological data here in the Philippines. Also, since some databases are not accessible online, there is no way to automatically fetch the data. A solution would be to manually import and export the data from these databases but it is difficult to create a parser for each file type that is not supported by the database management system (DBMS) of the integrating system.

C. Objectives

To develop a Collaboratory for Epidemiological Research (CEpiR) that has the following functionalities:

1. Allows the non-registered user to request for an account
2. Allows the registered user to
    a. Browse the list of projects in the system

b. Request for project creation
c. Request for membership in a project
d. Download software tools

3. Allows the tool administrator to manage and download software tools
4. Allows the project member to perform the following using the collaboratory
    a. Manage epidemiological records in the project's Epidemiological Data component
    b. Export epidemiological records data to Microsoft Excel (xls) format.
    c. Manage references and their information in the project's References component
    d. Upload and download miscellaneous files in the project's miscellaneous files component
5. Allows the project administrator to
    a. Perform all the functions of a project member
    b. Edit project information
    c. Manage project members
    d. Manage the data entry form/s in the project's Epidemiological Data component
6. Allows the system administrator to
    a. Manage user accounts
    b. Manage the list of projects

D. Significance of the Problem

With the implementation of this system, it is now possible to set up a common storage for the epidemiological data coming from heterogeneous databases. This allows more data to be available to epidemiologists so they could better understand the behavior of diseases and produce better epidemiological models.

Also, epidemiologists will now have better collaboration even if they are from different geographical locations by setting up a common interface for them to use. They will be able to share information effectively by letting them join projects and providing these with a system where they can share their databases and tools. Also, research repetition no longer has to occur since epidemiologists will now be aware of the ongoing researches performed by other Philippine-based epidemiologists. Epidemiologists would also be able to access data remotely since the system provides an option for data download.

E. Scope and Limitations

- The architecture of the collaboratory is designed in a way that could be used by different scientific disciplines but the specific design will be limited to epidemiological research.
- A project will have three components: Epidemiological Data, References and Miscellaneous Files. These components cannot be edited and deleted by any user.
- The References component contains any type of related literature, like for example, journals, e-books and other relevant references. The fields of the reference entry form are defined by the system and cannot be changed.
- The Miscellaneous component contains any other files related to the project, like for example, documents on the minutes of a particular meeting or schedule of events.

- The user who uploaded the reference and miscellaneous file is the one responsible for its copyright restriction
- The user who uploaded the reference with a link is not responsible for giving access to the link.
- The user is the person responsible for the availability and legality of the software use.
- It is assumed that the files being uploaded in the any of the components in the project are all relevant and that the given information on the files is correct.
- A user must choose a data entry form before entering data. It is assumed that the data being entered is correct and appropriate. Also, the data entry form is assumed to have been created prior to data entry.
- Epidemiological forms created by the system are in xls format only.
- Only .xls files are supported by the system in exporting data.
- The project administrator is responsible in approving or disapproving project membership requests.
- The evaluation of the credentials of a person who is requesting for an account is not part of the system and is based on the evaluation process set up or used by the administrator.
- The system administrator is responsible in ensuring that all approved projects are legitimate. Also, it is his/her responsibility to check the authenticity of the credentials of an individual or group requesting for an account and to assign access privileges to each user.

II. REVIEW OF RELATED LITERATURE

Collaboratories allowed the collaboration among researchers from different locations by setting up a common interface for them to interact and share information. This potential was already recognized by certain government agencies since the 1980's and appropriately developing them can contribute to enhancing research capacity, increasing scientific productivity, and expediting the translation of major scientific advances. [6]

Since then, numerous collaboratories were created. To separate one from another, a seven category taxonomy was described by Bos, et al. and the groupings were mainly based on the resource and activity (Table 1). In general, the level of management and sustainability of collaboratories get more difficult from the top left to the bottom right and that over time, the collaboratories move along the dimensions in both directions. [7]

	Instrument (Tools)	Information (Data)	Knowledge (New Findings)
Aggregating • Across distance • Loose coupling • Often asynchronously	Shared Instrument	Community Data System	Virtual Learning Community, Virtual Community of Practice
Co-creating • Across distance • Tighter coupling • Often synchronously	Infrastructure	Open Community Contribution System	Distributed Research Center

Table 1.Collaboratory types

The Biomedical Informatics Research Network (BIRN) is an example of a Community Data System and it seeks to advance biomedical research through data and instrumentation sharing and online collaboration. It has developed a federated and distributed infrastructure for the storage, retrieval, analysis, and documentation of biomedical imaging data and uses the Grid

architecture since it allows the running of jobs across the Grid without having to know how it happens. Also, the Grid allows users to seamlessly access and manage disparate data sets. [8]

On top of the infrastructure, tools having different data models and access patterns for data acquisition and management have been created. Examples of which are the Human Imaging Database (HID), the eXtensible Neuroimaging Archive Toolkit (XNAT) and XML-Based Clinical Experiment Data Exchange Schema (XCEDE). [8]

XNAT is a software platform designed to facilitate common management and productivity tasks for neuroimaging and associated data. It is heavily dependent on XML and XML Schema for its data representation, security system and generation of user interface content and it follows a three-tier design pattern composed of a relational database backend, a Java-based middleware and a web-based user interface. XSDs (XML Schema Document) provided by the system and those of the sites deploying the system provide the data types to be handled by the deployed system. A relational database is then created from the XSDs, as well as middleware classes that can be used by developers to implement custom functionality in the database. Also, the user interface content is generated upon creation of the relational database. [9]

To enter data in the archive, XML files or data entry forms are used. They allow flexible data entry, importing and exporting of data since the data entry forms are generated depending on a site's XSD and the XML files can easily be transformed to spreadsheets, PDF, HTML or alternative XML models using XSLT (eXtensibleStylesheet Language (XSL) Transformations). [9]

According to Saltz, et al., synthesizing information has become a need in biomedical research. Unfortunately, this still cannot be addressed because the same data sets are stored in disparate data sources and have different representations. Also, analysis tools can be difficult to

use since these tools may have different input and output formats and interfaces. In short, there is lack of interoperability. [10]

To address this need, the National Cancer Institute (NCI) launched the Cancer Biomedical Informatics Grid (caBIG) in 2004. It enables research groups in the cancer community to easily share, integrate and analyze data by developing data collection, management and analysis tools for the cancer community and by using the network architecture caGrid, or simply known as the Grid. [10]

caGrid is distinct from ordinary Grid technologies since it extends and customizes the current technologies to support the need of the cancer community. It focuses on the modelling of data and metadata information thus there is now syntactic and semantic interoperability across heterogeneous databases in the Grid. Also, it enabled unified and programmatic access to remote, autonomously controlled data and analytical resources. [10]

Aside from addressing the need for interoperability, collaboratories have also been used to access instrumentation and generate processing power that is not available to most researchers. NEESgrid provides earthquake researchers across the US with leading-edge computing resources and research equipment to help them perform increasingly complex, comprehensive and accurate earthquake simulations. They provide services like the NTCP which allows remote applications to perform experiments using instrumentation at far places, data repository services, e-notebook services that serve as virtual laboratory books, telepresence services to view the instrumentation and ongoing experiments using cameras at the location, and the CHEF service that allows researchers to access computing facilities, communicate via discussion forums and text chat, and collaborate using scheduling and data sharing. [11]

The European Bioinformatics Institute (EMBL-EBI) is an institute that provides access to databases and tools in bioinformatics by managing different databases of biological data. But instead of providing access via the traditional web-based interfaces, it now uses web services. These web services free the user from web page constraints and require no prior knowledge of the technology. But more importantly, scientists no longer have to invest resources in the installation, maintenance and execution of software and data with web services. [12]

Web services are also used by the RCSB Protein Data Bank for data distribution. The PCB is the central worldwide repository for 3D structure data of biological macromolecules and is based on the three-tier architecture composed of an underlying relational database, a presentation tier and an object-relational J2EE middle tier based on Hibernate. Data are loaded using XML or mmCIF data files which are integrated and collected every week. [13]

One application that uses the web service of PSB is soaPDB [14]. soaPDB allows authenticated users of PDB to save their searches and organize the search results in a relational database since queries are only saved for the duration of the browser session. It also allows for the running of the soaPSB server privately within one's institution to enable collaboration among local users without public exposure.

Aside from the collaboratories, some systems are being developed for integrating epidemiological data for analysis and simulation. An example of which is IntegraEPI [15,16]. Its data integration service, called Integra-GISE, is composed of different layers, the most important of which is the mediator. Queries coming from the client layer are split by the mediator into subqueries according to the schemas stored in the MCS (also known as fragmentation) therefore generating a set of local queries. These local queries are then passed to the Data Access Services Layer where wrappers are instantiated so that local operations can be performed on the needed

data sources. The wrappers send their results back to the Data Access Services Layer and then to the mediator where it performs a defragmentation process to integrate all results.

III. THEORETICAL FRAMEWORK

A. Collaboratory

According to the SURFfoundation, a collaboratory is a virtual research environment that enables researchers from distant places to work together and share information, resources and knowledge thus speeding up research. [17] William Wulf coined the word in 1989 by merging the words “collaboration” and “laboratory” and defined it as a “center without walls” [18].

Basically, a collaboratory is all about connecting people to people, information and facilities, as illustrated in the Collaboratory Triangle in Figure 1. To connect people and information, it makes use of cyber workspaces which include groupware technologies like audio/videoconferences, chat, file sharing and discussion forums that mimic face to face interactions and other tools helpful in their project like computation, simulation and analysis services. It also provides data sharing facilities and repositories that integrates data from heterogeneous databases and/or provides a location for researchers to store their data. Reference materials and links to them usually can also be found in collaboratories.

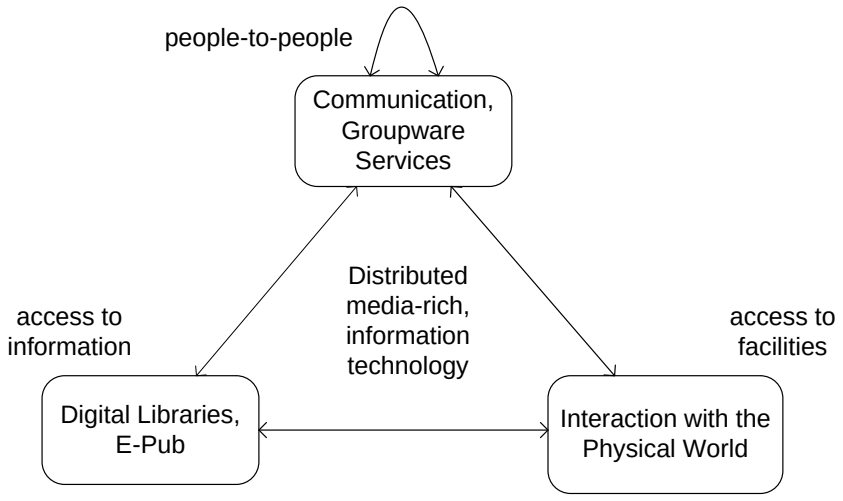


Figure 1 The Collaboratory Triangle

And since not all researchers have enough resources, it is important to be able to have access to facilities and instrumentation they need. To address this, collaboratories usually use Grid computing. According to EDUCAUSE, Grid computing provides users with access to the resources they need, when they need them. It also provides users with extraordinary computational needs significant processing power by aggregating the processing powers of those idle computers that are part of the Grid. [19]

Even though collaboratories are not bounded by time and space, there are other barriers that must be addressed. First of all, scientific knowledge is difficult to transmit. The knowledge often is complex and requires thorough explanation (which is difficult to do if at a distance). Second, researchers like to work independently. Finally, cross-institutional collaboration is difficult since problems like legal and privacy issues are difficult to resolve. [7]

Since collaboratory technology is still in its exploratory stages, there are no definite set of tools that are relevant to collaboratories. [4] But, according to Wulf et al., tools for audio/videoconferencing, chat, shared computer display and electronic notebook, file sharing,

online instrumentation, computation and visualization should be included since they allow researchers to interact with remote colleagues in a rich, in-process style. [18] An example of which is shown below.

The Biological Collaborative Research Environment (BioCoRE) is a collaborative environment for biomedical research, research management and training. This system is project-oriented, meaning invited members are the only persons that can view project files and messages. Also, there is a BioCoRE Control Panel that shows the notifications of all projects (each project has its own tab) the user is involved in and serves as the chat facility of the collaboratory. Simply open a new tab to conduct private chat with contacts or project chat using the tab of the project. [20]

Each BioCoRE project has three components: Documents, Workbench and Notebook. The Documents component consists of the shared file system of each project. The Workbench component consists of the Job Manager which monitors and logs the computational jobs done in the supercomputers, the VMD launcher which launches the molecular visualization program used in the system and Configuration File Generator which helps users easily create input files for NAMD, the molecular dynamics simulation tool used in the system. Finally, the Notebook component consists of the Message Board which serves as the discussion area of the project, the Lab Book which serves as a journal for a member's research progress and the Website Library which manages web links related to the project. [20]

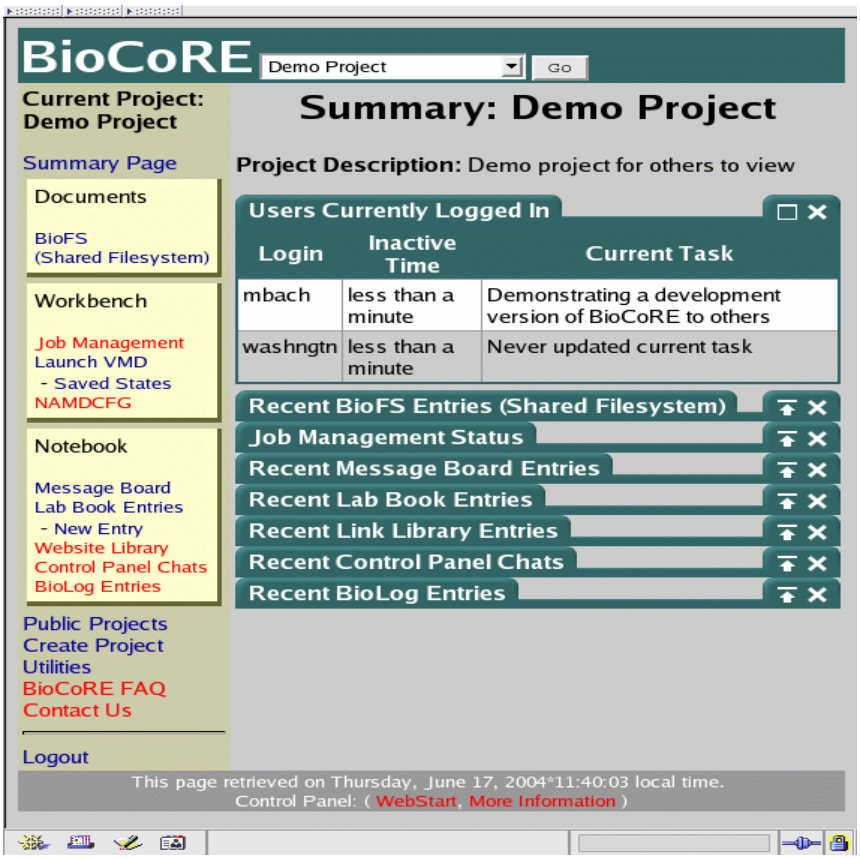


Figure 2 Summary/Main page for a BioCoRE project

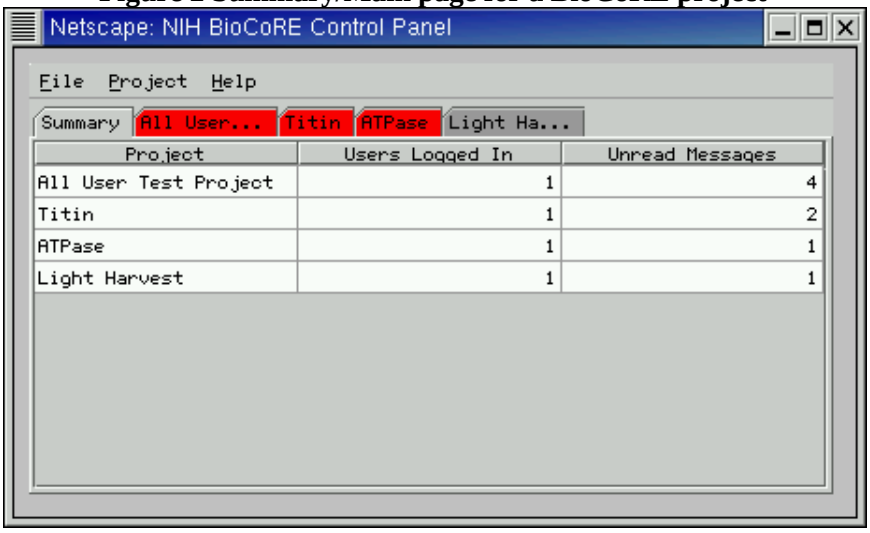


Figure 3BioCoRE Control Panel

B. Epidemiology

Epidemiology is the "study of the distribution and determinants of health-related states or events (including disease), and the application of this study to the control of diseases and other

health problems" [21]. Epidemiologists try to determine the factors associated with the disease (like risk and preventive factors) by checking the correlation with the disease but they can never prove causation. Nevertheless, epidemiology helps not just public health officials make informed decisions, but ordinary people as well. Epidemiologic studies can also serve as a stepping stone in finding the causes of a particular disease. [22]

According to the CDC, epidemiology has six core functions in public health practice: public health surveillance, field investigation, analytic studies, evaluation, linkages and policy development. Also, there are two approaches to epidemiology: descriptive and analytic.[23]

Descriptive epidemiology is involved with the study of disease incidence and distribution by time, place, andperson and includes calculation of rates and identification of parts of the population at higher risk than others. It tries to describe the who, what, where, when and how of an epidemic while the analytic epidemiology tries to test the results of descriptive epidemiology by using comparison groups.[23]

Public health surveillance and field investigations fall under descriptive epidemiology. Public health surveillance tries to portray the patterns of disease occurrence and potential by gathering and analyzing data related to morbidity (measured by the incidence and prevalence of the disease), mortality and other health related information while field investigations try to search for the source and mode of transmission of a disease and gather data related to it. [23]

Analytic studies, as the name implies, falls under analytic epidemiology and uses statistics to establish the relationship/association between the disease and the factor being tested. Risk, rate, odds and proportionate mortality ratios are examples of measures of association. [23]

Evaluation is the determination of the significance and worth of the object being evaluated, usually by careful appraisal and study. [24] In epidemiology, it is involved with the efficiency and effectiveness of plans, operations, impacts or outcomes which can be helpful in deciding the usefulness of the said objects. [23]

Gathering of epidemiological data is not done by one person alone. It involves multidisciplinary collaboration since data are coming from different locations and health care providers like clinicians, laboratories and hospitals. Linkages to these people should be established by epidemiologists to promote current and future collaborations. [23]

Finally, epidemiologists can provide input and recommendations regarding disease control measures, notifiable disease regulations and healthcare policies since they understand the problem and the population where it occurs. And even though epidemiology can never prove causation, it usually can provide sufficient evidence to prove causation. [23]

C. Web Service

According to the World Wide Web Consortium, a web service is a software system designed to support interoperable machine-to-machine interaction over a network. [25] It provides easy access to remote content and application functionality using industry-standard mechanisms, without dependency on the provider's location, implementation, or the data format. [26]

Remote Procedure Call (RPC) is the first well known and widely used Web service. It is a set of tools that allows one to write a program whose parts are on different computers, without having to know how it happens. [25] Unfortunately, this style produces tightly coupled, language specific systems [27]. To address this, Web services now interact with other systems using XML and HTTP. XML provides a language which can be used in different platforms and programming

languages and still express complex messages and functions whereas HTTP is the most used Internet protocol. [28]

Also, Web services make use of the Service-Oriented Architecture (SOA), as illustrated in Figure 4. This enables sharing and aggregating of services in a flexible way thus can be used to satisfy more complex functions. Also, this provides loose coupling since the clients who seek/request services can access the service providers without having to know much about them. The directory service is also available for fast searching of services since it stores some information about the services. [29]

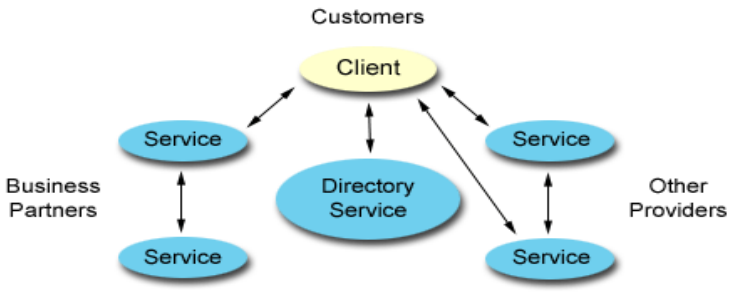


Figure 4 Service-Oriented Architecture

D. Database management system (DBMS)

A DBMS, as the name implies, is a system that manages databases. It determines how the data are to be stored and retrieved thus enabling quick search and retrieval of that data. It also provides data security by only allowing authorized users to access a database, data integrity by ensuring that no two users are updating the same data at the same time and data independence by asking for data by field name and not specifying the structure. Examples of DBMSs are Oracle Database, MySQL and Microsoft Access. [32]

DBMSs can be classified as homogeneous or heterogeneous. A homogeneous DBMS uses the same database technology on all data sources and the data are compatible. Also, all sources use the same hardware and software for the system. A heterogeneous DBMS, on the

other hand, can use different hardware, software and database technologies on the data sources and the data from various sources are incompatible. [33]

E. Definition of Terms

a. Data entry form – a dynamic form that defines the structure and definition of an epidemiological record. Details like default values, maximum and minimum values, etc. of attributes can be defined using this form.

b. Epidemiological record – corresponds to a set of variable values defined by a project's data entry form. An epidemiological record corresponds to a row in the database.

c. Tool – an application that is independent from the collaboratory but performs a certain function for the latter like for example, data management.

d. Double encoding – a quality assurance measure that works by randomly selecting records from a data group and double encoding those by inputting those records again and comparing them with the stored data

e. Field identifier – a unique number or label used to identify a field in a record

IV. DESIGN AND IMPLEMENTATION

A. Context Diagram

The system supports two types of users: registered user and system administrator where registered users can still be subdivided into project administrator, tool administrator and project member. The system also interacts with epidemiological tools that can be downloaded through the system for remote data access. The Context Diagram is shown in Figure 5.

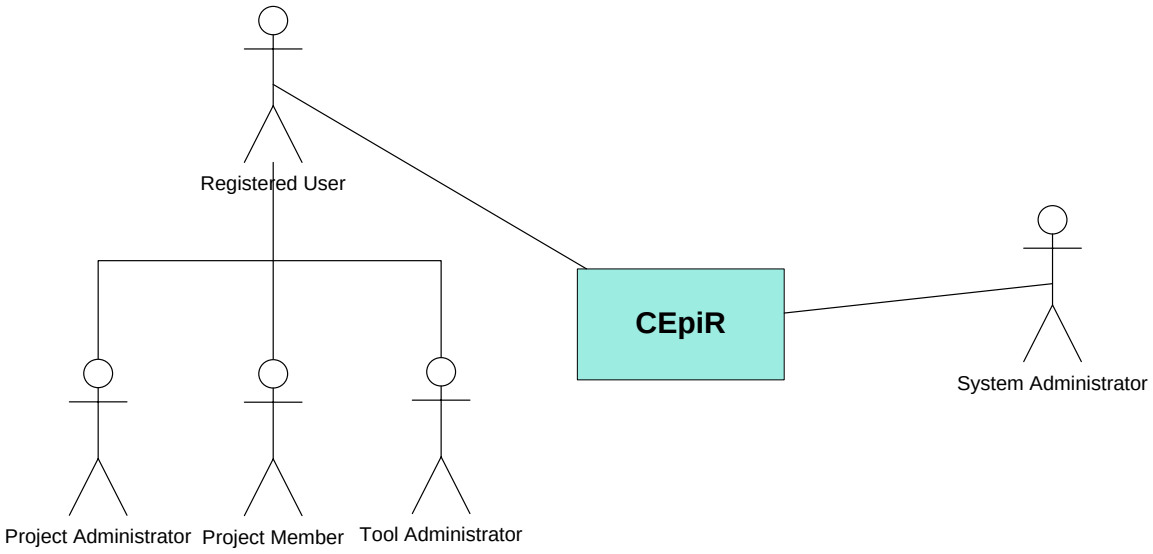


Figure 5 Context Diagram of CEpiR

B. Use Case Diagram of CEpiR

A non-registered user can request for an account upon visiting the collaboratory. A registered user can view projects in the system and then request for membership. He/she can also request for project creation and upon approval, he/she will be selected as project

administrator for that project. The project administrator will be responsible in managing the project's information, its data entry form/s and its members. Lastly, the system administrator is responsible in creating user accounts, assigning user roles and project approval or disapproval.

In addition, a registered user can also view and download the tools uploaded in the system. The tool administrator will be responsible in managing all the tools in the system. Figure 6 shows the top level use case diagram of the system.

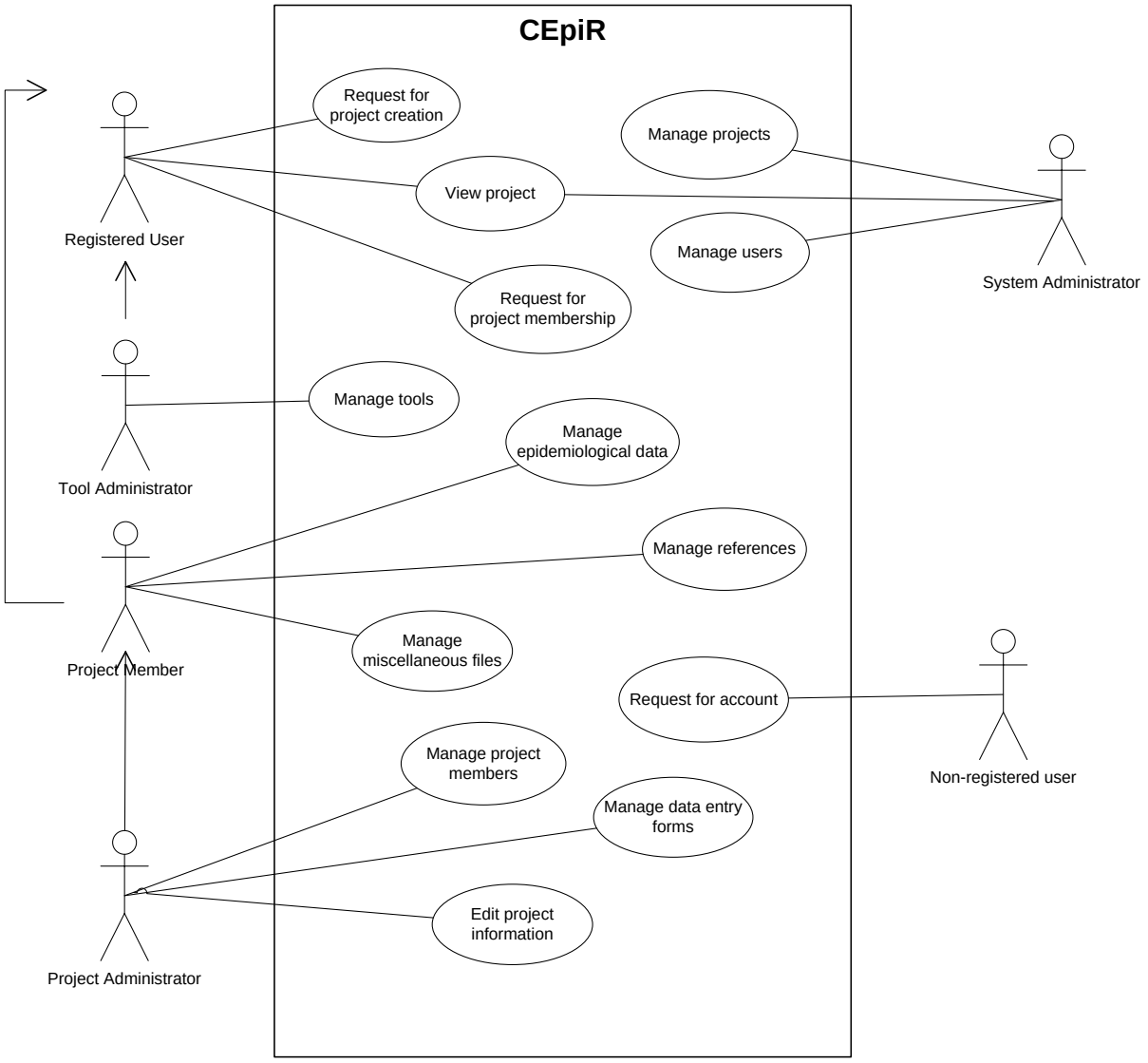


Figure 6 Top level use case diagram of CEpiR

1. **Request for account**

Any non-registered user can request for a user account in the collaboratory. Figure 7 shows the Request for account activity diagram.

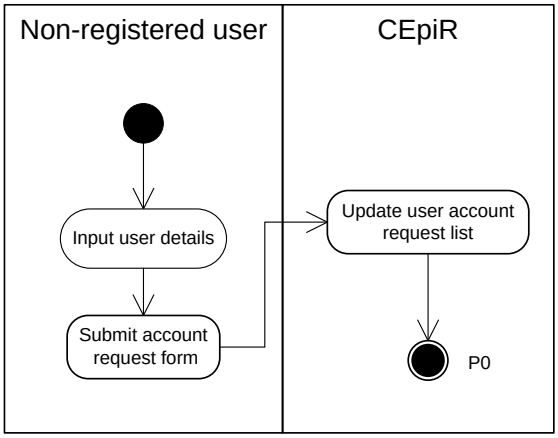


Figure 7 Activity diagram for Request for account use case

2. **Request for project creation**

Any registered user can request for project creation upon login. Figure 8 shows the request for project creation activity diagram.

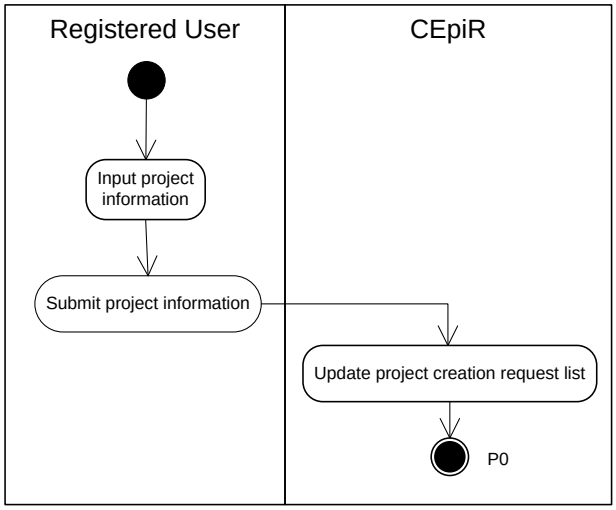


Figure 8 Activity diagram for Request for project creation use case

3. View project

Any registered user and the system administrator can search and view the list of projects. Upon selecting a project in the list, public information on that project will be displayed. Figure 9 shows the view project activity diagram.

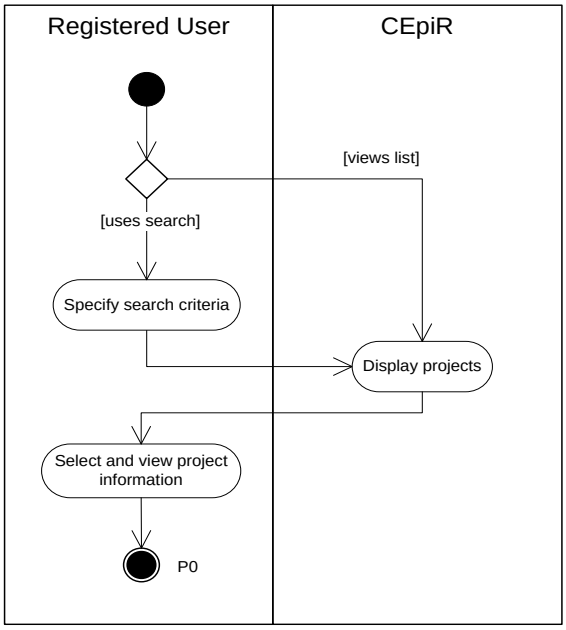


Figure 9 Activity diagram for View project use case

4. Manage Tools

The tool administrator can add, edit and delete tools. Any registered user or a tool administrator view and download any tool. Figure 10 shows the Add tool activity diagram, Figure 11 shows the Edit tool activity diagram, Figure 12 shows the Download tool activity diagram, Figure 13 shows the View tool activity diagram and Figure 14 shows the Delete tool activity diagram.

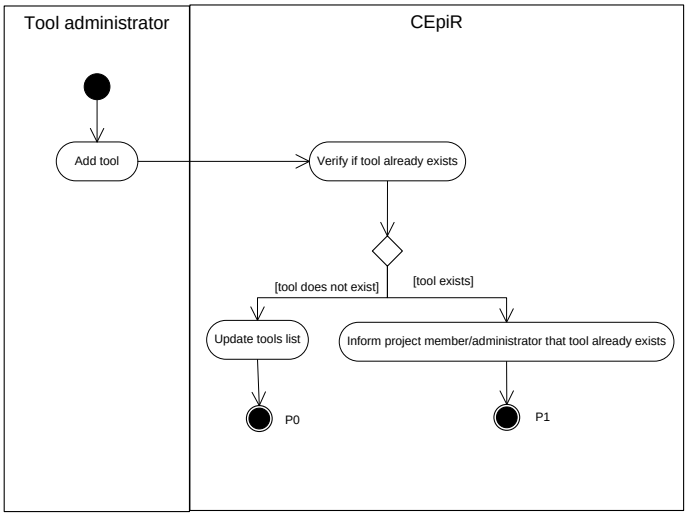


Figure 10 Activity diagram for Add tool use case

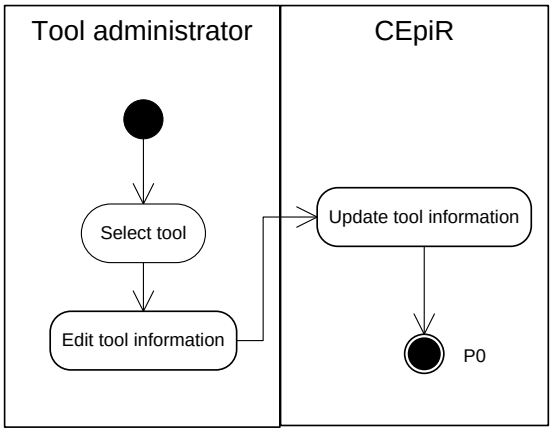


Figure 11 Activity diagram for Edit tool use case

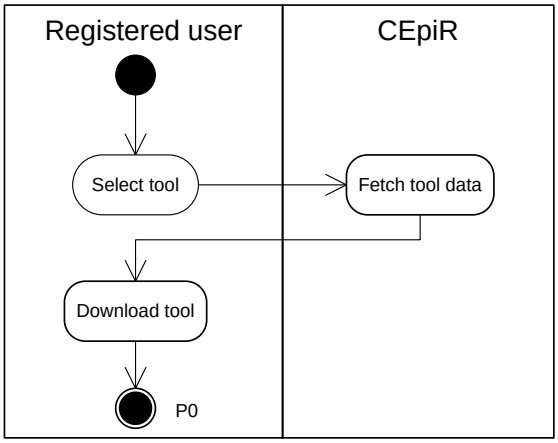


Figure 12 Activity diagram for Download tool use case

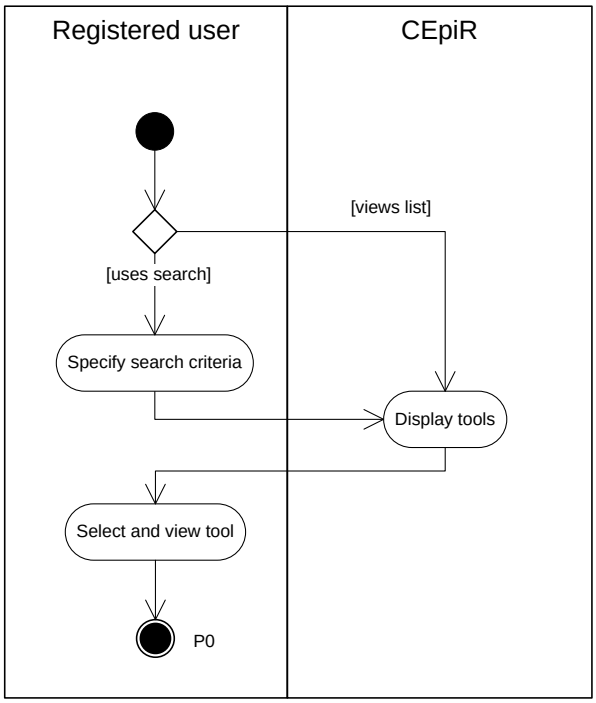


Figure 13 Activity diagram for View tool information use case

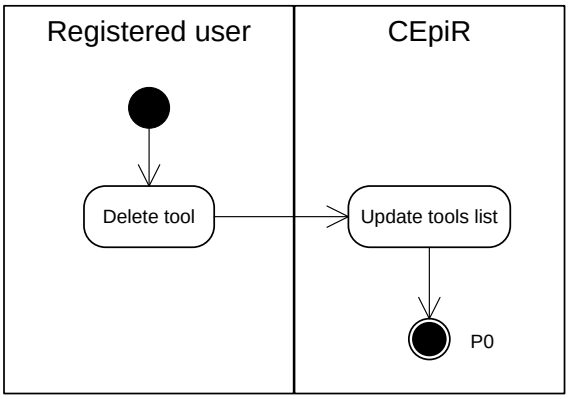


Figure 14 Activity diagram for Delete tool use case

5. Request for project membership

Any registered user can request for project membership in any of the projects listed in the system. Figure 15 shows the Request for project membership activity diagram.

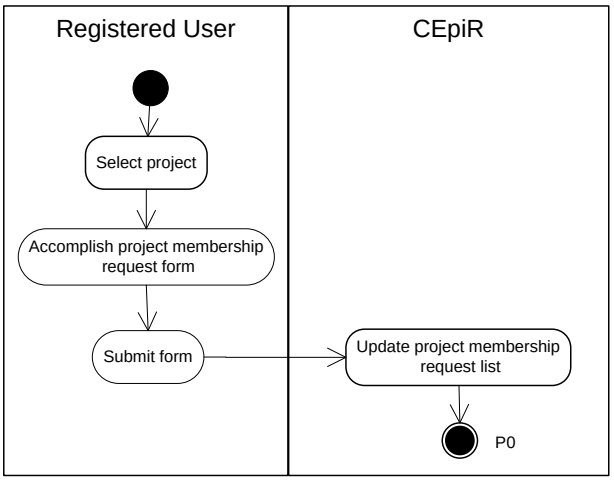


Figure 15 diagram for Request for project membership use case

6. Manage epidemiological data

Project members and the project administrator can store epidemiological data by manually adding records through the data entry forms. Both project members and the project administrator can add, edit, view, delete and export epidemiological data. Figure 16 shows the Add epidemiologic data activity diagram, Figure 17 shows the Edit epidemiologic data activity diagram, Figure 18 shows the Delete epidemiologic data activity diagram and Figure 19 shows the Export epidemiologic data activity diagram.

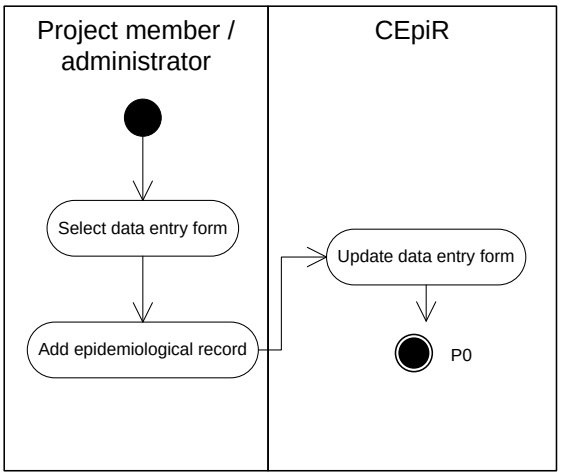


Figure 16 Activity diagram for Add epidemiological record use case

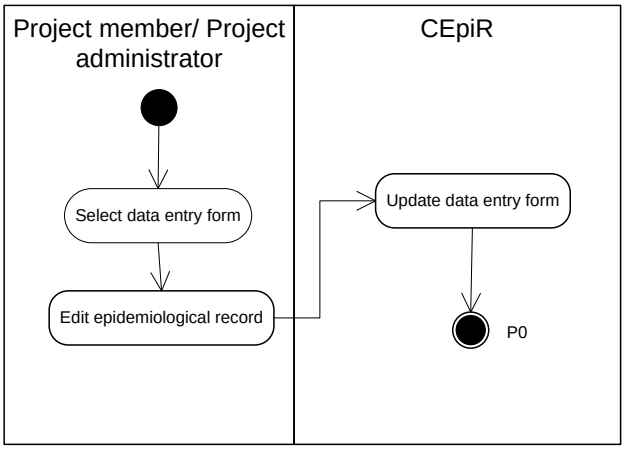


Figure 17 Activity diagram for Edit epidemiological record use case

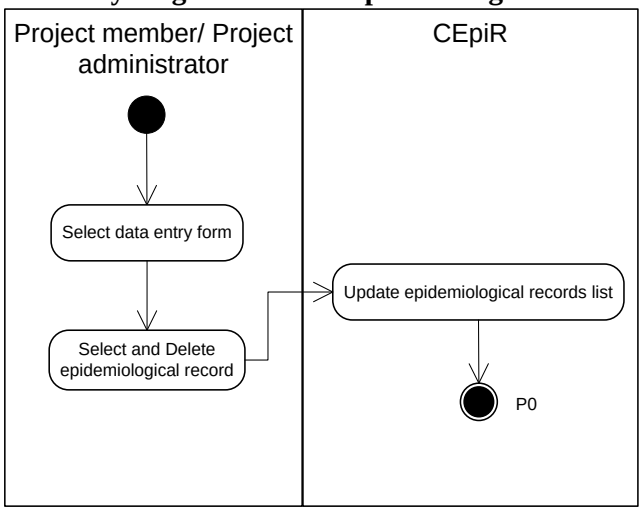


Figure 18 Activity diagram for Delete epidemiological record use case

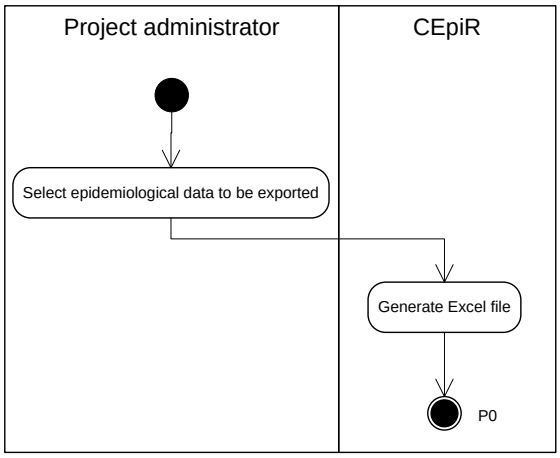
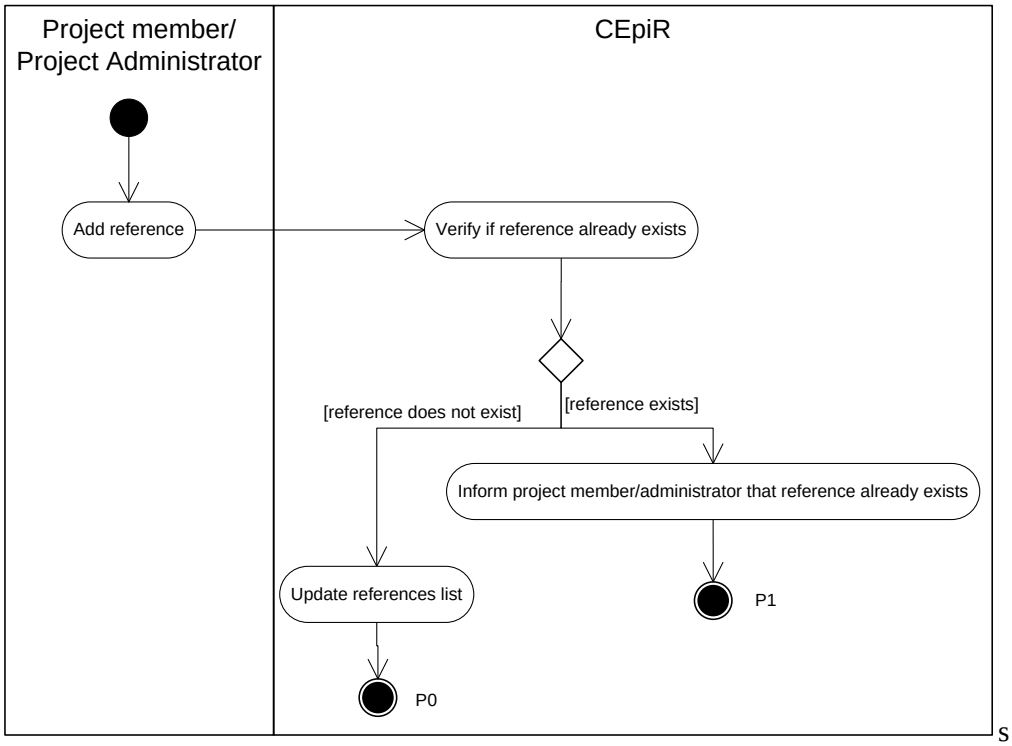


Figure 19 Activity diagram for Export epidemiological data use case

7. Manage references

The project member and project administrator can add, edit, view and delete any reference. Figure 20 shows the Add reference activity diagram, Figure 21 shows the Edit reference activity diagram, Figure 22 shows the View reference activity diagram, Figure 23 shows the Delete reference activity diagram and Figure 24 shows the Download reference activity diagram.



S

Figure 20 Activity diagram for Add reference use case

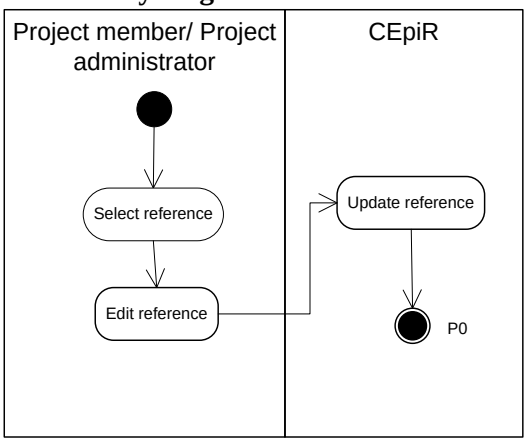


Figure 21 Activity diagram for Edit reference use case

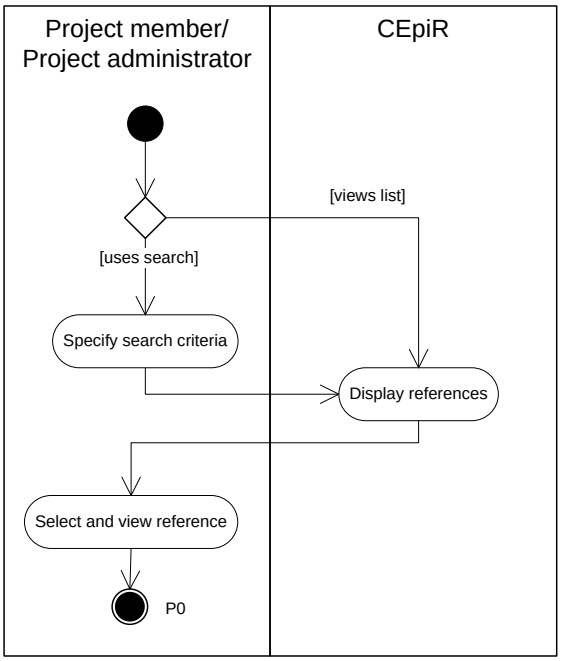


Figure 22 Activity diagram for View reference use case

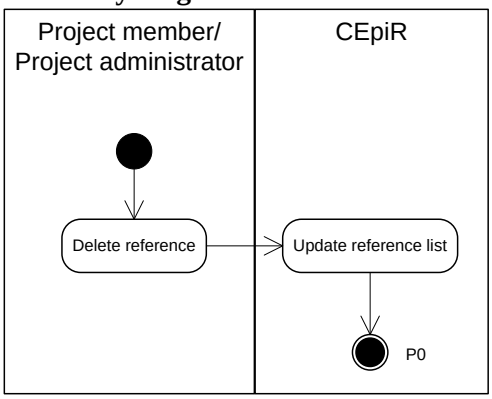


Figure 23 Activity diagram for Delete reference use case

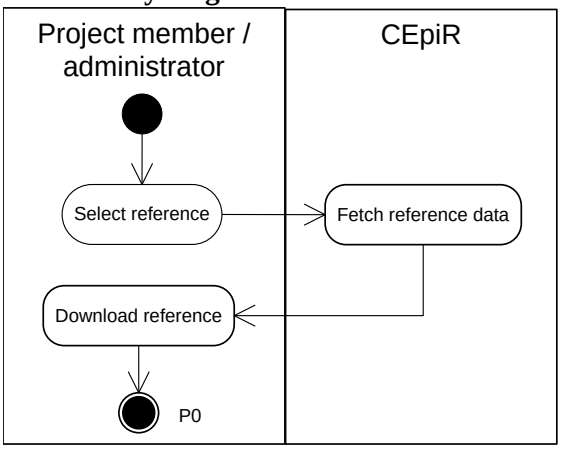


Figure 24 Activity diagram for Download reference use case

8. Manage miscellaneous files

The project member and project administrator can add, edit, view and delete any miscellaneous file. Figure 25 shows the Add miscellaneous file activity diagram, Figure 26 shows the Edit miscellaneous file activity diagram, Figure 27 shows the View miscellaneous file activity diagram, Figure 28 shows the Delete miscellaneous file activity diagram and Figure 29 shows the Download miscellaneous file activity diagram.

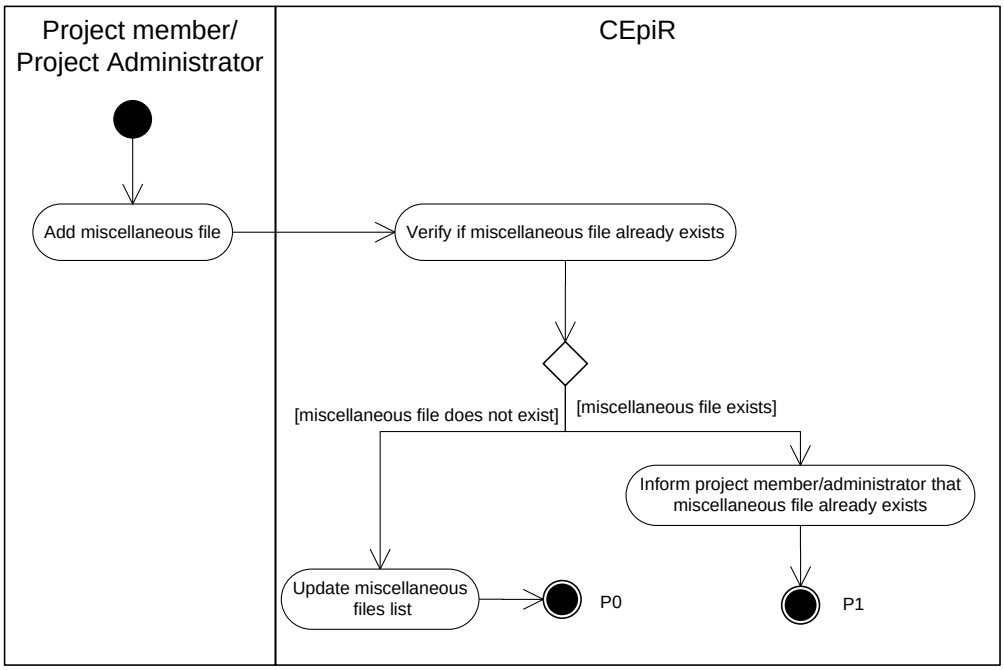


Figure 25 Activity diagram for Add miscellaneous file use case

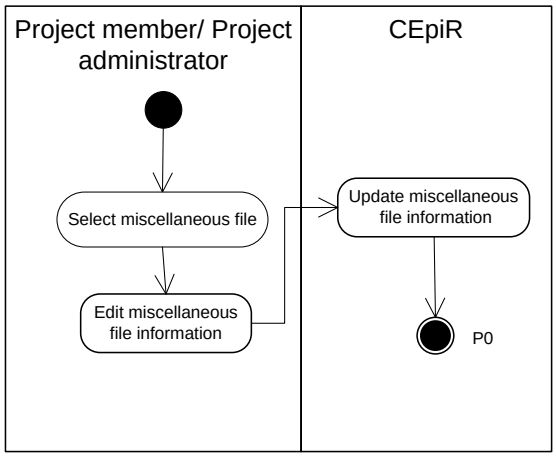


Figure 26 Activity diagram for Edit miscellaneous file use case

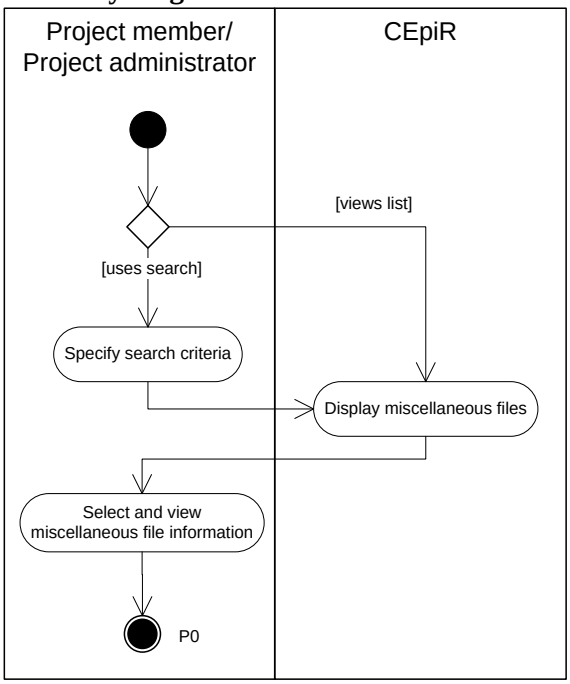


Figure 27 Activity diagram for View miscellaneous file information

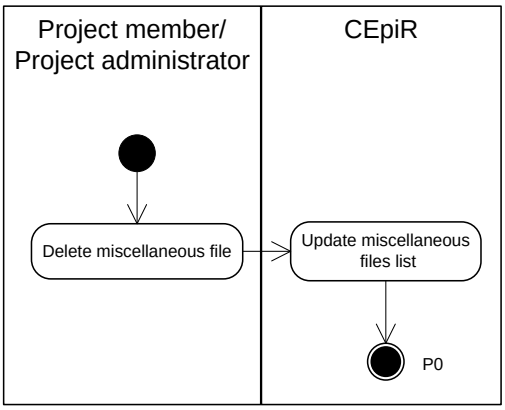


Figure 28 Activity diagram for Delete miscellaneous file

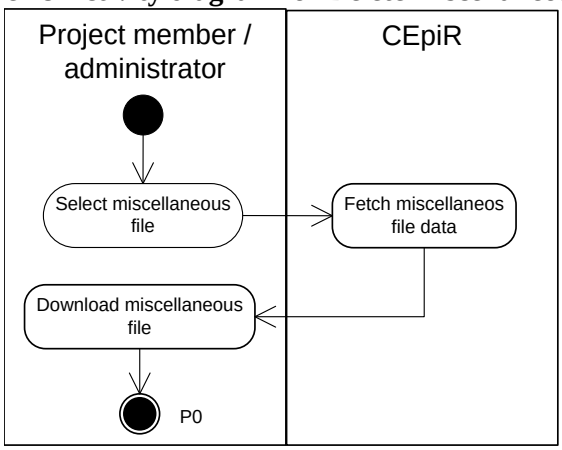


Figure 29 Activity diagram for Download reference use case

9. Manage project members

The project administrator can approve project membership requests, add and delete any member of the project and edit a member's membership. The project members or the project administrator can view project members. Figure 30 shows the Add project member activity diagram, Figure 31 shows the Edit project member's membership activity diagram, Figure 32 shows the View project members activity diagram, Figure 33 shows the Delete project member activity diagram and Figure 34 shows the Accept request for project membership activity diagram.

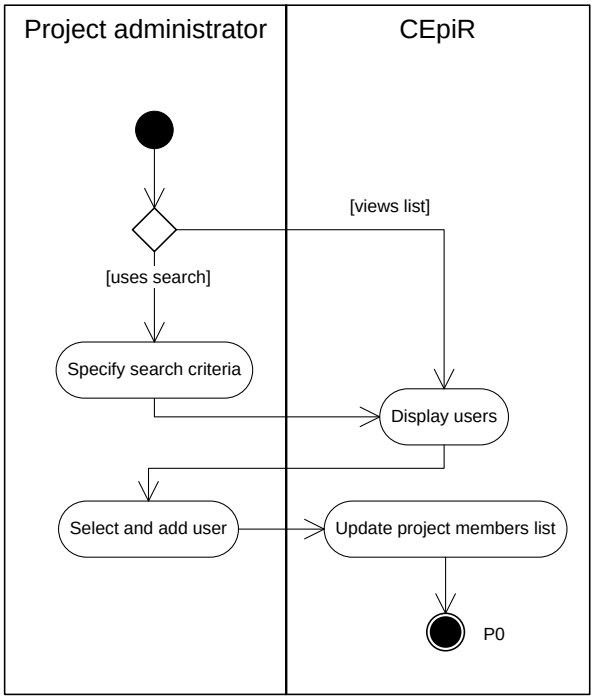


Figure 30 Activity diagram for Add member use case

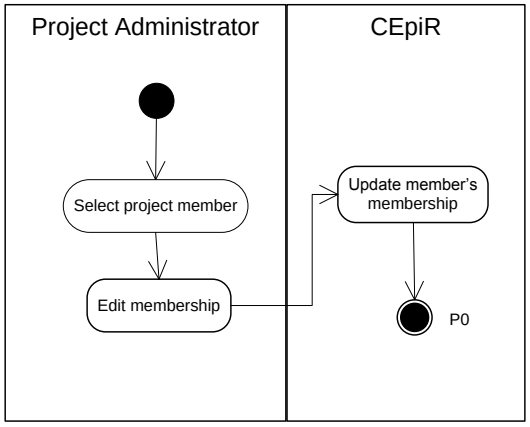


Figure 31 Activity diagram for Edit project member's membership use case

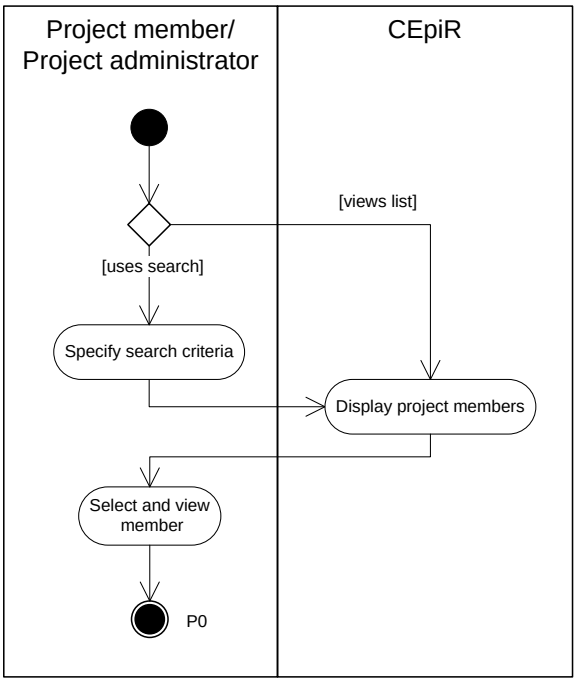


Figure 32 Activity diagram for View member use case

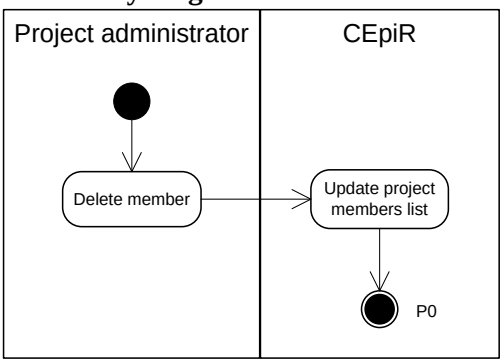


Figure 33 Activity diagram for Delete member use case

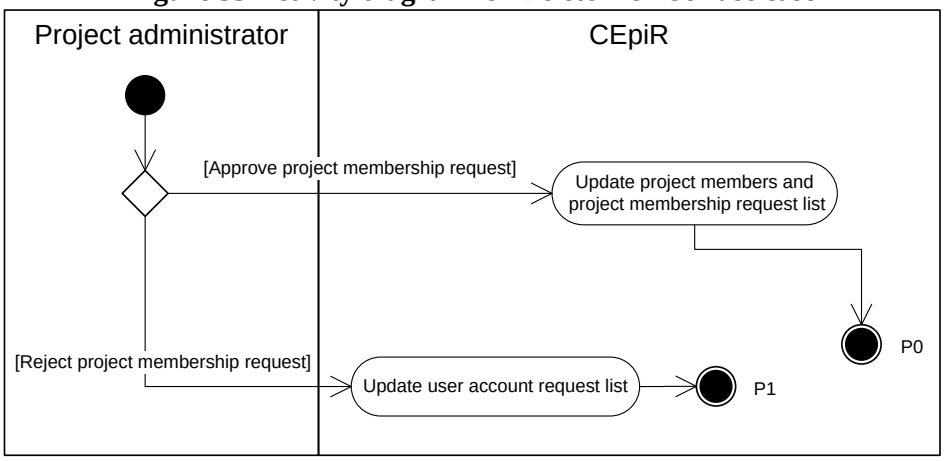


Figure 34 Activity diagram for Accept project membership request use case

10. Manage data entry forms

The project administrator can add and delete any data entry form. The project member and administrator can view and edit any data entry form. Figure 35 shows the Add data entry form activity diagram, Figure 36 shows the View data entry form activity diagram, Figure 37 shows the Edit data entry form activity diagram and Figure 38 shows the Delete data entry form activity diagram.

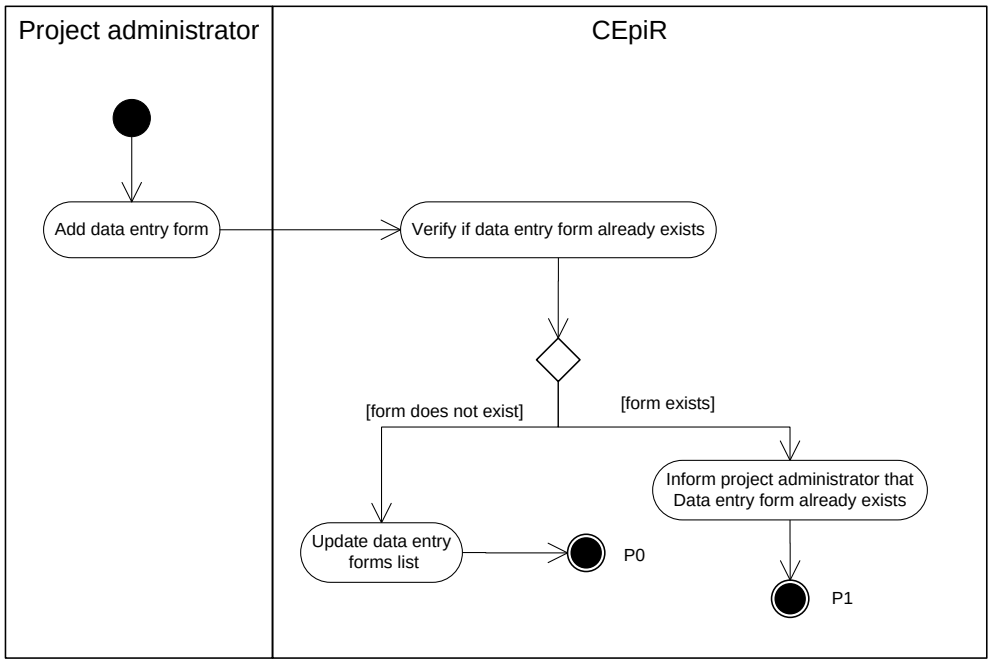


Figure 35 Activity diagram for Add data entry form use case

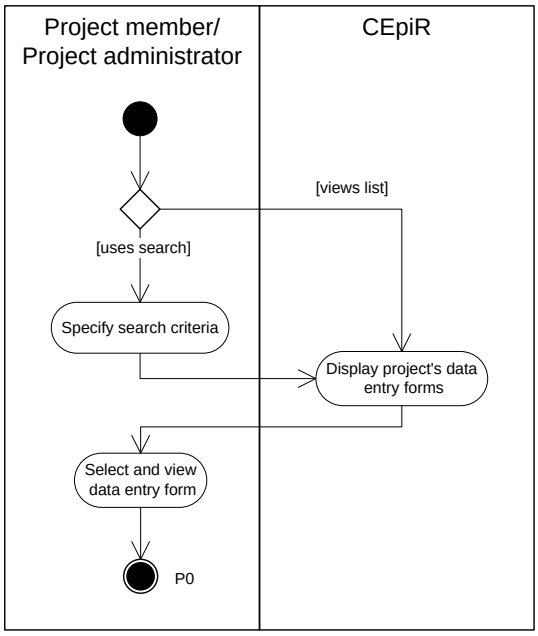


Figure 36 Activity diagram for View data entry form use case

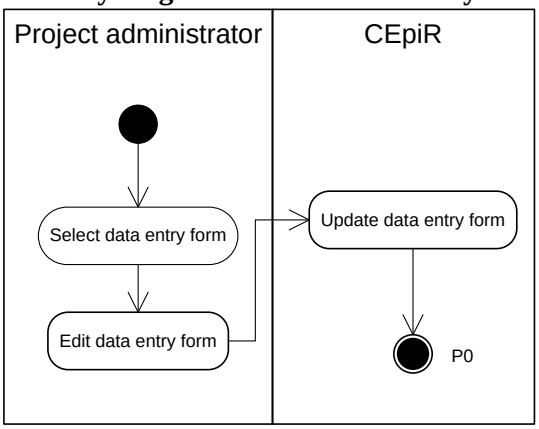


Figure 37 Activity diagram for Edit data entry form use case

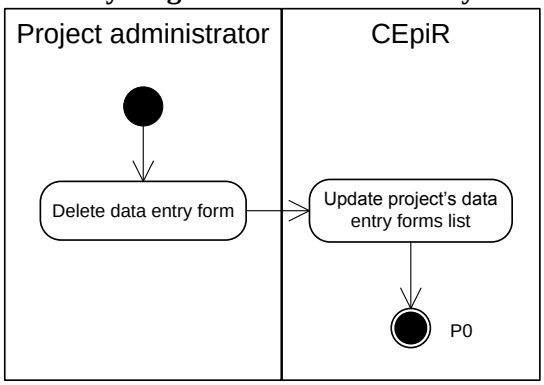


Figure 38 Activity diagram for Delete data entry form use case

11. Edit project information

The project administrator can edit any project information. Figure 39 shows the Edit project information activity diagram.

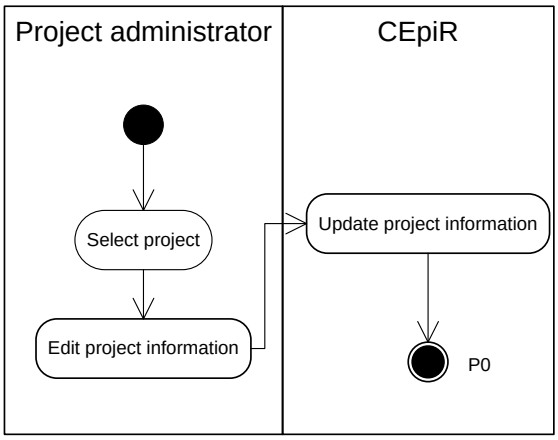


Figure 39 Activity diagram for Edit project information use case

12. Manage projects

The system administrator is the only user who can add, edit, delete and approve projects in the system. Figure 40 shows the Add project activity diagram, Figure 41 shows the Edit project activity diagram, Figure 42 shows the Delete project activity diagram and Figure 43 shows the Approve project creation request activity diagram.

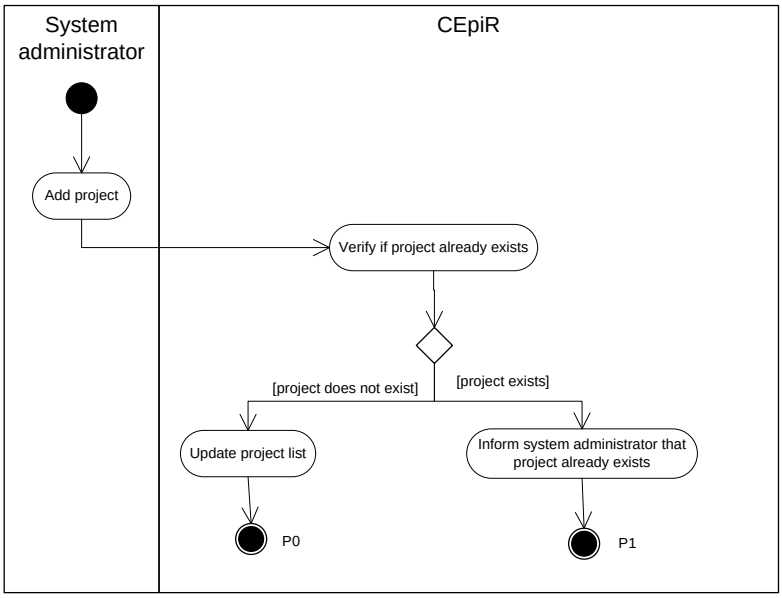


Figure 40 Activity diagram for Add project use case

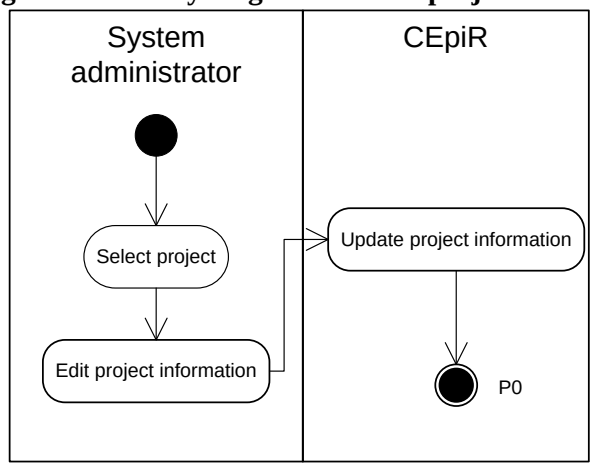


Figure 41 Activity diagram for Edit project use case

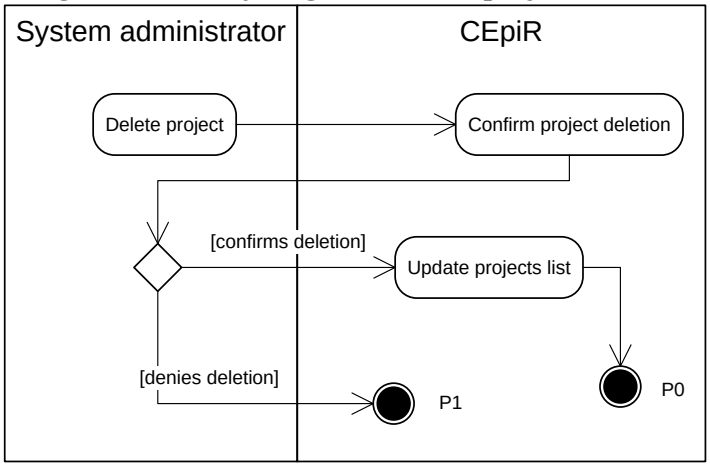


Figure 42 Activity diagram for Delete project use case

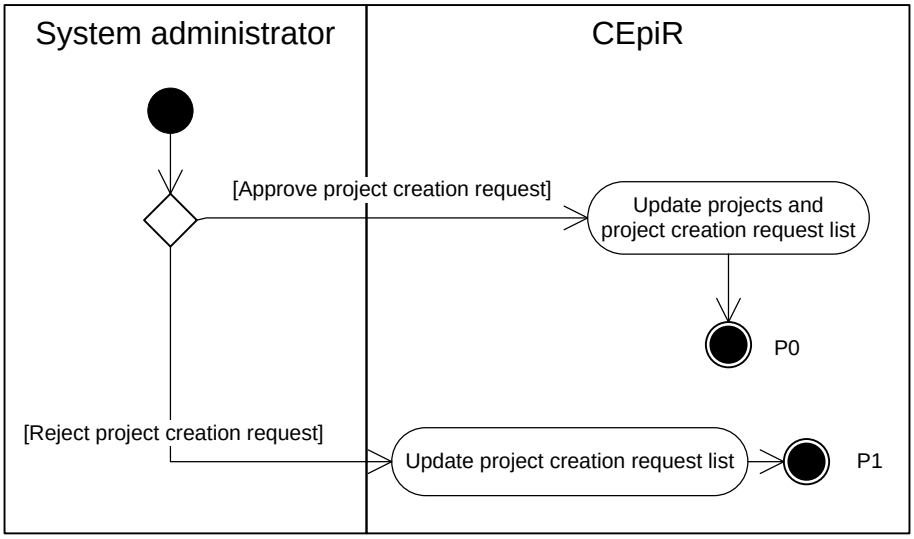


Figure 43 Activity diagram for Approve project creation request

13. Manage users

Only the system administrator can add, edit, view and delete any user account, approve user account requests and assign user roles. Figure 44 shows the Add user activity diagram, Figure 45 shows the Edit user activity diagram, Figure 46 shows the View user activity diagram, Figure 47 shows the Delete user activity diagram and Figure 48 shows the Approve user account request activity diagram.

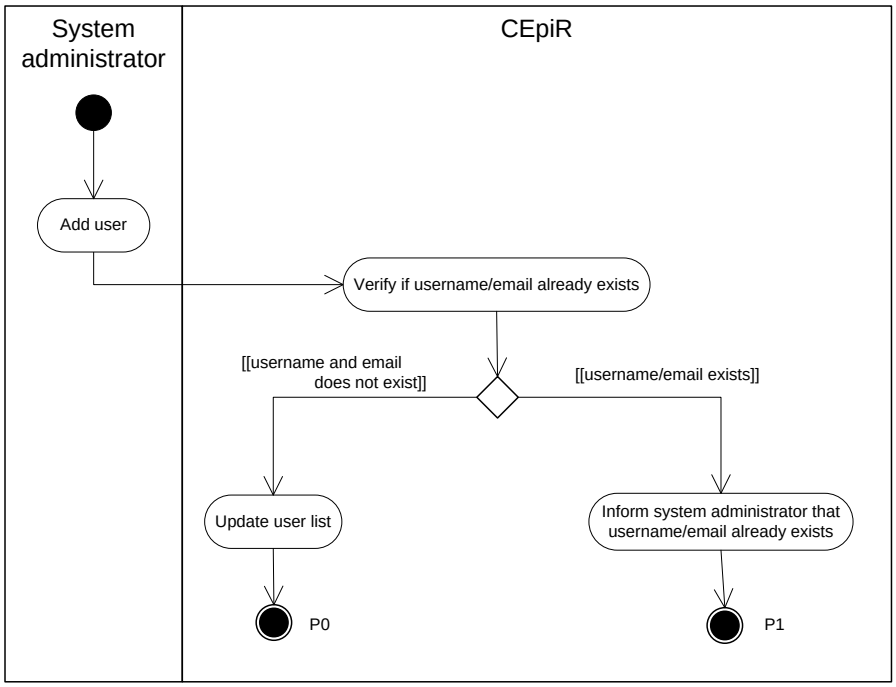


Figure 44 Activity diagram for Add user use case

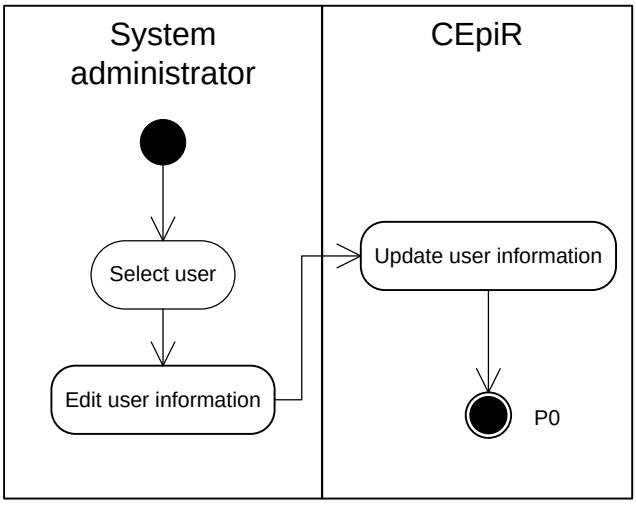


Figure 45 Activity diagram for assign user role use case

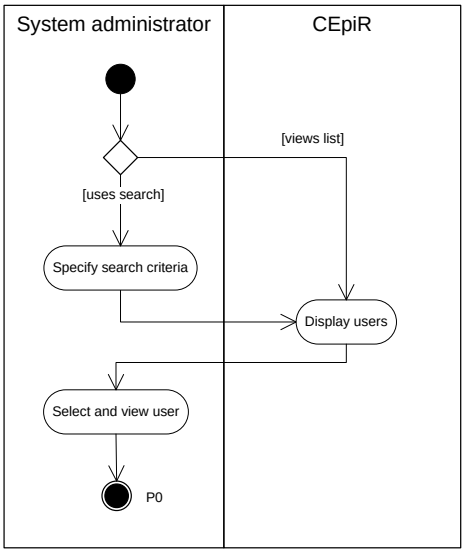


Figure 46 Activity diagram for View user use case

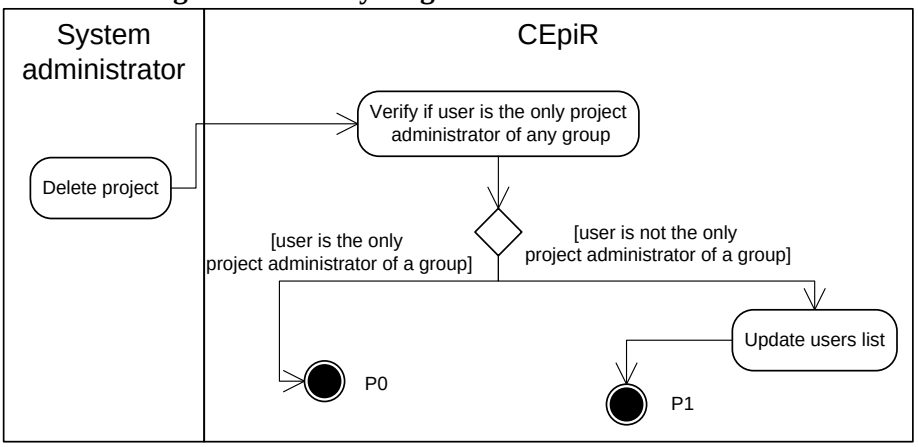


Figure 47 Activity diagram for Delete user use case

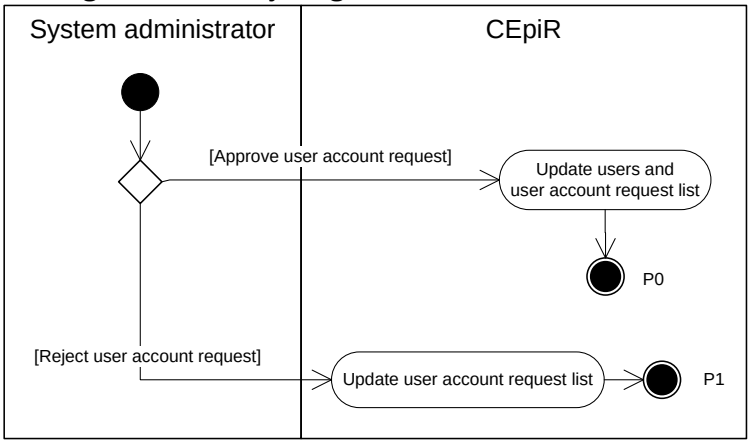


Figure 48 Activity diagram for Approve user account request use case

C. Entity Relationship Diagram

Figure 49 shows the Entity Relationship Diagram of CEpiR.

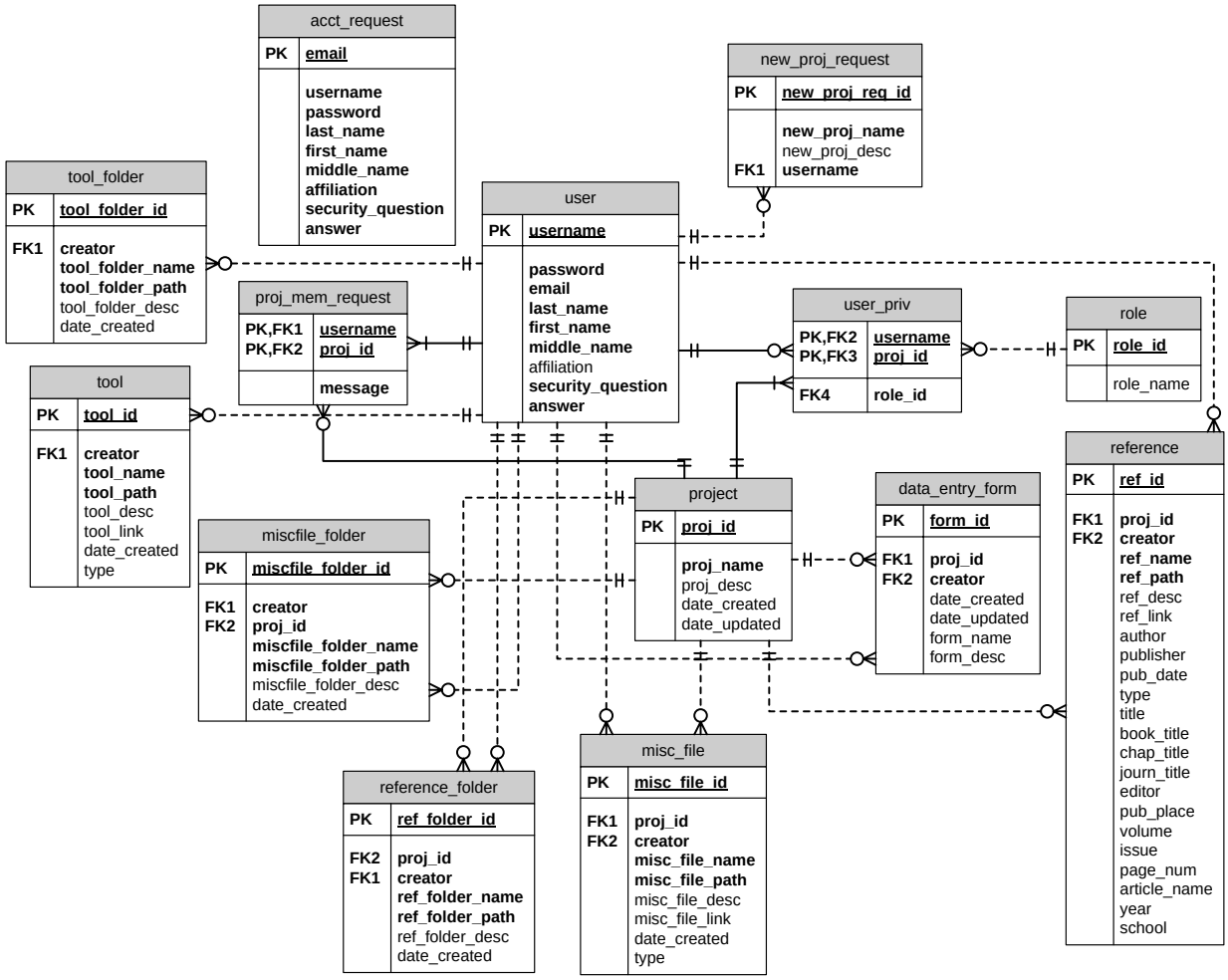


Figure 49 Entity Relationship Diagram for CEpiR

D. Data Dictionary

CEpiR consists of 14 tables, namely: user, user_priv, role, new_proj_request, proj_mem_request, acct_request, project, data_entry_form, reference, tool, misc_file, reference_folder, tool_folder and miscfile_folder. Listed below are the list of tables and their respective data fields. Primary keys are underlined.

user – contains the account details and some basic information of the user

Field	Type	Description
username	VARCHAR(24)	Unique identifier of user; used for login
password	VARCHAR(100)	Password of user
email	VARCHAR(50)	E-mail address of user
last_name	VARCHAR(30)	Last name of user
first_name	VARCHAR(40)	First name of user
middle_name	VARCHAR(30)	Middle name of user
affiliation	VARCHAR(30)	Institution/organization where the user is affiliated
security_question	VARCHAR(100)	Security question
answer	VARCHAR(30)	Answer to the security question

user_priv – contains the roles of a certain user in a project

Field	Type	Description
username	VARCHAR(24)	User with the role
proj_id	INT(10)	Project where the user's role is applied
role_id	INT(10)	Code assigned to a role

role – list of all user roles in the system (1 – project member, 2 – project administrator, 3 – system administrator)

Field	Type	Description
role_id	INT(10)	Code assigned to a role
role_name	VARCHAR(25)	Name of role

new_proj _request – list of all project creation requests

Field	Type	Description
new_proj_req_id	INT(10)	Unique identifier of a project creation request
new_proj_name	VARCHAR(200)	Name of project to be created
new_proj _desc	VARCHAR(500)	Description of project
username	VARCHAR(24)	User who requested

proj_mem _request – list of all project membership requests

Field	Type	Description

username	VARCHAR(24)	User with the role
proj_id	INT(10)	Project where the user is requesting membership
message	VARCHAR(200)	Message containing reasons to approve membership request

acct _request – list of all user account requests

Field	Type	Description
email	VARCHAR(50)	E-mail address of user
username	VARCHAR(24)	Unique identifier of user; used for login
password	VARCHAR(100)	Password of user
last_name	VARCHAR(30)	Last name of user
first_name	VARCHAR(40)	First name of user
middle_name	VARCHAR(30)	Middle name of user
affiliation	VARCHAR(50)	Institution/organization where the user is affiliated
security_question	VARCHAR(100)	Security question
answer	VARCHAR(30)	Answer to the security question

project – contains the list of all projects in the system

Field	Type	Description
proj_id	INT(10)	Unique identifier of project
proj_name	VARCHAR(200)	Project name
proj_desc	VARCHAR(500)	Project description
date_created	Timestamp	Date when project was created
date_updated	Timestamp	Date when project was last modified

data_entry_form – contains the list of all the data entry form/s in the system

Field	Type	Description
form_id	INT(10)	Unique identifier of the data entry form
form_name	VARCHAR(200)	Name of the data entry form
form_desc	VARCHAR(200)	Short description of the form
creator	VARCHAR(24)	User who added the form

proj_id	INT(10)	Project where data entry form was created
date_created	Timestamp	Date when project was created
date_updated	Timestamp	Date when project was last modified

reference – contains the list of all the references of all the projects in the system

Field	Type	Description
ref_id	INT(10)	Unique identifier of reference
ref_name	VARCHAR(200)	Reference title
ref_desc	VARCHAR(300)	Reference description
ref_link	VARCHAR(200)	Link where more information about the reference can be found or where it can be accessed
ref_path	VARCHAR(200)	Path where uploaded reference can be found
creator	VARCHAR(24)	User who added the reference
proj_id	INT(10)	Project owning the reference
type	ENUM	Reference Type
title	VARCHAR(300)	Reference Title
book_title	VARCHAR(300)	Book Title
chap_title	VARCHAR(300)	Chapter Title
journ_title	VARCHAR(300)	Journal Title
author	VARCHAR(100)	Author of the reference
publisher	VARCHAR(100)	Publisher of the reference
pub_date	TEXT	Date when reference was published
pub_place	VARCHAR(200)	Place of publication
editor	VARCHAR(200)	Book chapter editor
volume	VARCHAR(20)	Volume where journal was published
issue	VARCHAR(20)	Issue where journal was published
page_num	VARCHAR(20)	Page where journal can be found
article_name	VARCHAR(200)	Article name
year	VARCHAR(20)	Year when thesis was published
school	VARCHAR(20)	School where thesis was submitted

tool – contains the list of all the tools in the system

Field	Type	Description

tool_id	INT(10)	Unique identifier of the tool
tool_name	VARCHAR(200)	Tool name
tool_desc	VARCHAR(300)	Tool description
tool_link	VARCHAR(200)	Link where more information about the tool can be found or where it can be accessed
tool_path	VARCHAR(200)	Path where uploaded tool can be found
creator	VARCHAR(24)	User who added the reference
date_created	Timestamp	Date when tool was added
type	INT(10)	Indicates whether tool was uploaded or only a link was provided (1- file, 2 - link)

misc_file– contains the list of all the miscellaneous files of all the projects in the system

Field	Type	Description
misc_file _id	INT(10)	Unique identifier of the miscellaneous file
misc_file_name	VARCHAR(200)	File name
misc_file_desc	VARCHAR(300)	File description
misc_file_link	VARCHAR(200)	Link where more information about the miscellaneous file can be found or where it can be accessed
misc_file_path	VARCHAR(200)	Path where uploaded miscellaneous file can be found
creator	VARCHAR(24)	User who added the miscellaneous file
proj_id	INT(10)	Project owning the miscellaneous file
date_created	Timestamp	Date when miscellaneous file was added
type	INT(10)	Indicates whether miscellaneous file was uploaded or only a link was provided (1- file, 2 - link)

tool_folder – contains the list of all tool folders in the system

Field	Type	Description
tool_folder_id	INT(10)	Unique identifier of the tool folder
tool_folder_name	VARCHAR(200)	Tool folder name
tool_folder_desc	VARCHAR(300)	Tool folder description
tool_folder_path	VARCHAR(200)	Path where tool folder can be found
creator	VARCHAR(24)	User who added the tool folder
date_created	Timestamp	Date when tool folder was added

miscfile_folder – contains the list of all miscellaneous file folders in the system

Field	Type	Description
miscfile_folder_id	INT(10)	Unique identifier of the miscellaneous file folder
miscfile_folder_name	VARCHAR(200)	Miscellaneous file folder name
miscfile_folder_desc	VARCHAR(300)	Miscellaneous file folder description
miscfile_folder_path	VARCHAR(200)	Path where miscellaneous file folder can be found
creator	VARCHAR(24)	User who added the miscellaneous file folder
proj_id	INT(10)	Project owning the miscellaneous file folder
date_created	Timestamp	Date when miscellaneous file folder was added

reference_folder – contains the list of reference folders in the system

Field	Type	Description
ref_folder_id	INT(10)	Unique identifier of the reference folder
ref_folder_name	VARCHAR(200)	Reference folder name
ref_folder_desc	VARCHAR(300)	Reference folder description
ref_folder_path	VARCHAR(200)	Path where reference folder can be found
creator	VARCHAR(24)	User who added the reference folder
proj_id	INT(10)	Project owning the reference folder
date_created	Timestamp	Date when reference folder was added

E. System Architecture

1. GWT

Google Web Toolkit (GWT) is a development toolkit developed by Google that allows developers to quickly build and maintain complex yet highly performant JavaScript front-end applications in Java. Developers write the AJAX front-end in Java which GWT then cross-compiles into optimized JavaScript that automatically works across all major browsers. [34]

2. MVP

The Model View Presenter (MVP) framework is the framework highly recommended by Google to use with GWT. In MVP, the Model represents the business objects of the system. The View represents all UI components while the Presenter contains all the logic for the system. A fourth component, known as the AppController, controls all the logic outside the Presenter like for example, history management and view transition logic. So whenever a user navigates to another page, the Presenter initializes a new History token and passes the event handling to AppController. It then loads the corresponding Presenter and View for that History token. [34]

The View and Presenter interact through the Display interface. This interface is defined in the Presenter, implemented in the View and it allows data to pass from the Presenter to the View. Also, this allows the Presenter to handle multiple views without having to hardcode the implementation of the View to the Presenter. [35]

And whenever the client side wants to access the server side, a RPC service will be invoked. This service will then fetch the necessary data and send it back to the client.

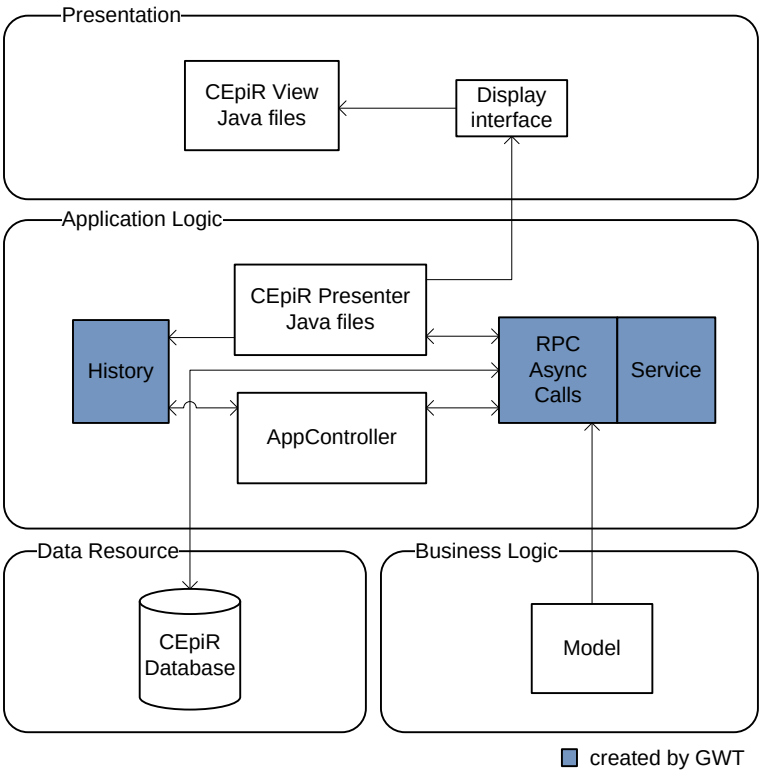


Figure 50 System Architecture of CEpiR

V. TECHNICAL ARCHITECTURE

A. Server Requirements

1. Operating System: Ubuntu Linux10.04 or Windows 7
2. Apache Tomcat 5.0 or higher
3. Sun Java 2 Runtime Environment 1.5
4. Sun Java Development Kit 1.6
5. MySQL 5.0 or higher
6. At least 200MB in hard disk space

B. Client Requirements

1. Browser: Google Chrome 10.0, Mozilla Firefox 3.0 or higher, Internet Explorer 8.0 or Safari 3.2; JavaScript enabled

VI. RESULTS

The homepage of the Collaboratory of the Epidemiological Research (CEpiR) is shown on Figure 51. The system allows registered users to login to the system and guests to request for an account. Also, registered users can reset their password by clicking the Forgot Password link.

Figure 51 Homepage of CEpiR

Upon logging in to the system, the user will be redirected to his/her dashboard. If the user is a system administrator, the system will display the number of all pending account and project requests of the system (shown in Figure 52).

Figure 52 Homepage for System Administrators

If the user is not a system administrator, the system will display all the user's pending project creation requests, pending project membership requests and if applicable, all the project membership requests of the projects where he/she is a project administrator of (Figure 53).

Figure 53 Homepage for registered users

The system also provides an interface wherein the users can edit their account information and change their passwords.

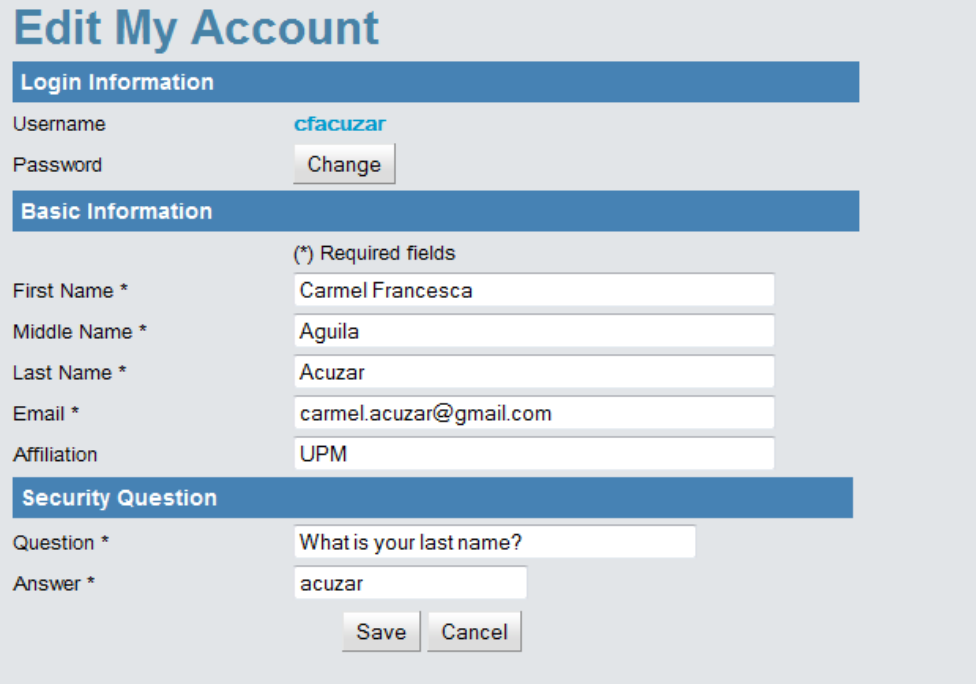


Figure 54 Edit Account Information

All registered users can access and download tools from the system's Tools page but only the tool administrator can add, edit and delete tools and its folders. Figure 55 shows the Add tool form of the project.

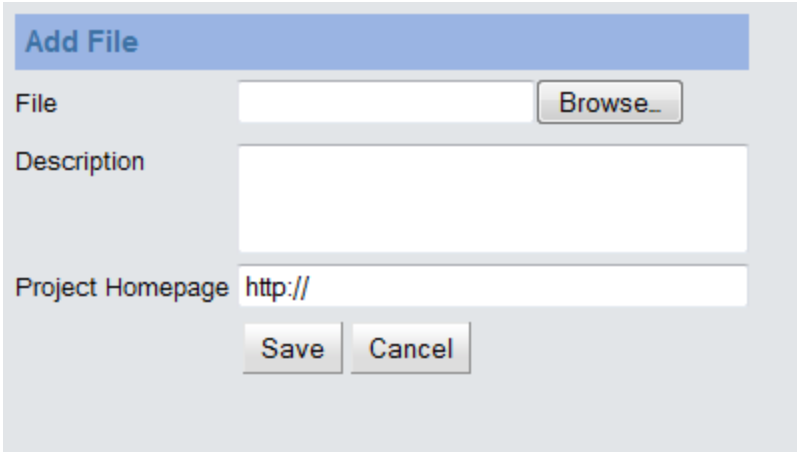


Figure 55 Add tool form

In Figure 56, the system lists all the projects found in the system. It also shows if the user has a pending membership request to that project or if he/she is a project member or administrator.

The system is able to produce a homepage for every project found in the system. Non-members can request for membership using this page (Figure 57) while members can use this page to leave the project. Only the project administrators have the option to edit or delete the project (Figure 58).

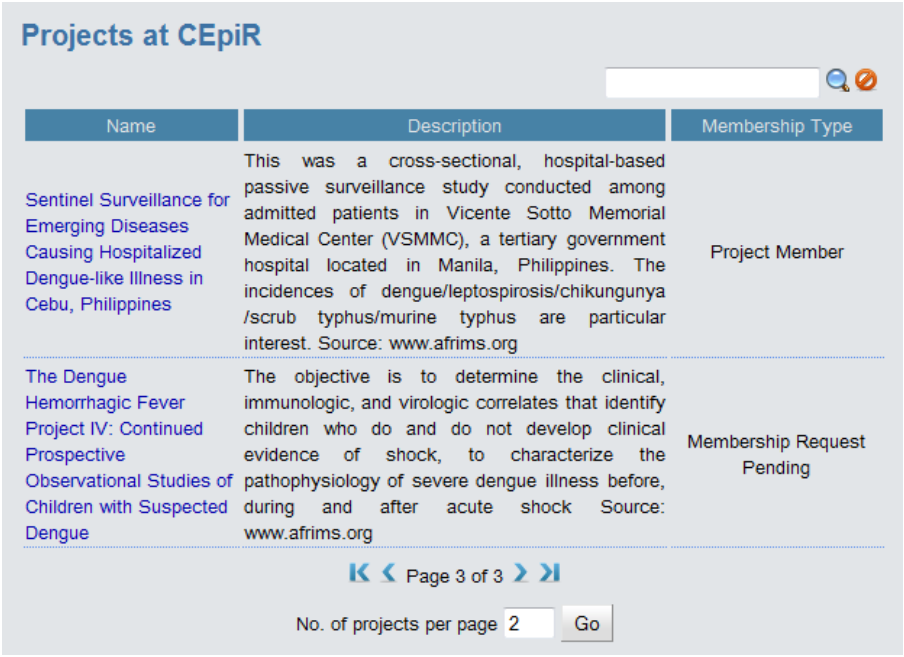


Figure 56 List of all project at CEpiR

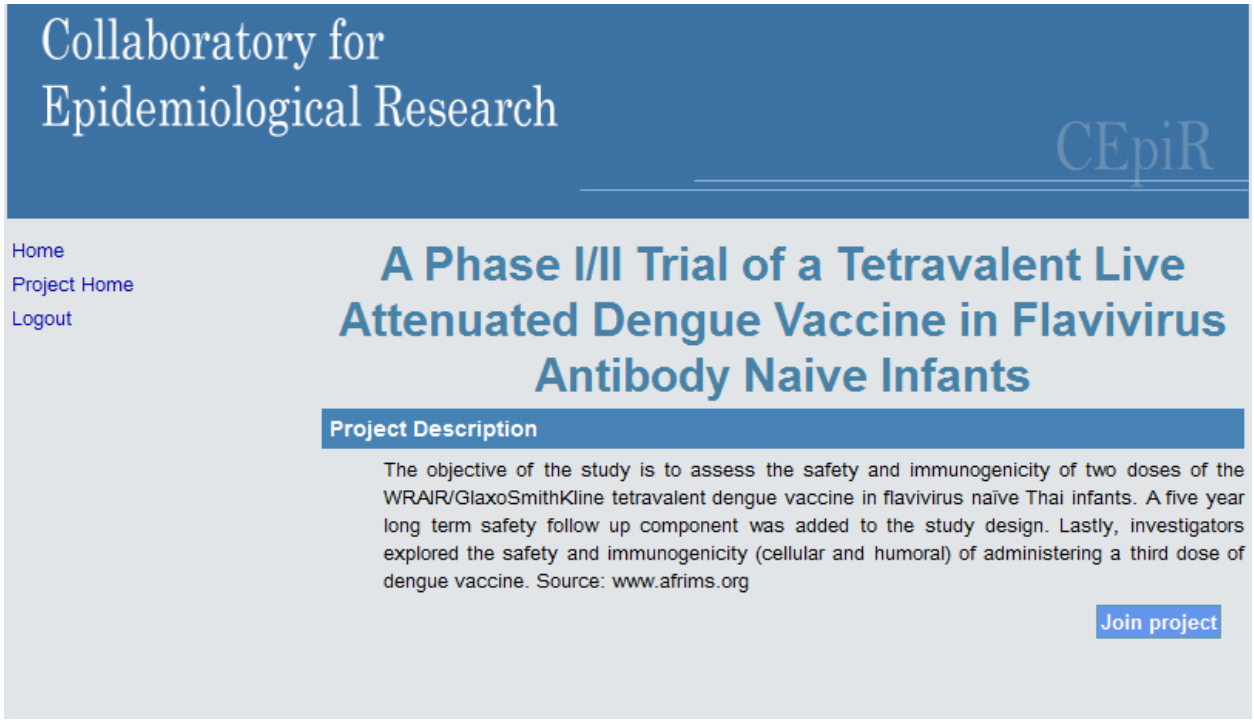


Figure 57 Project Homepage for non-members

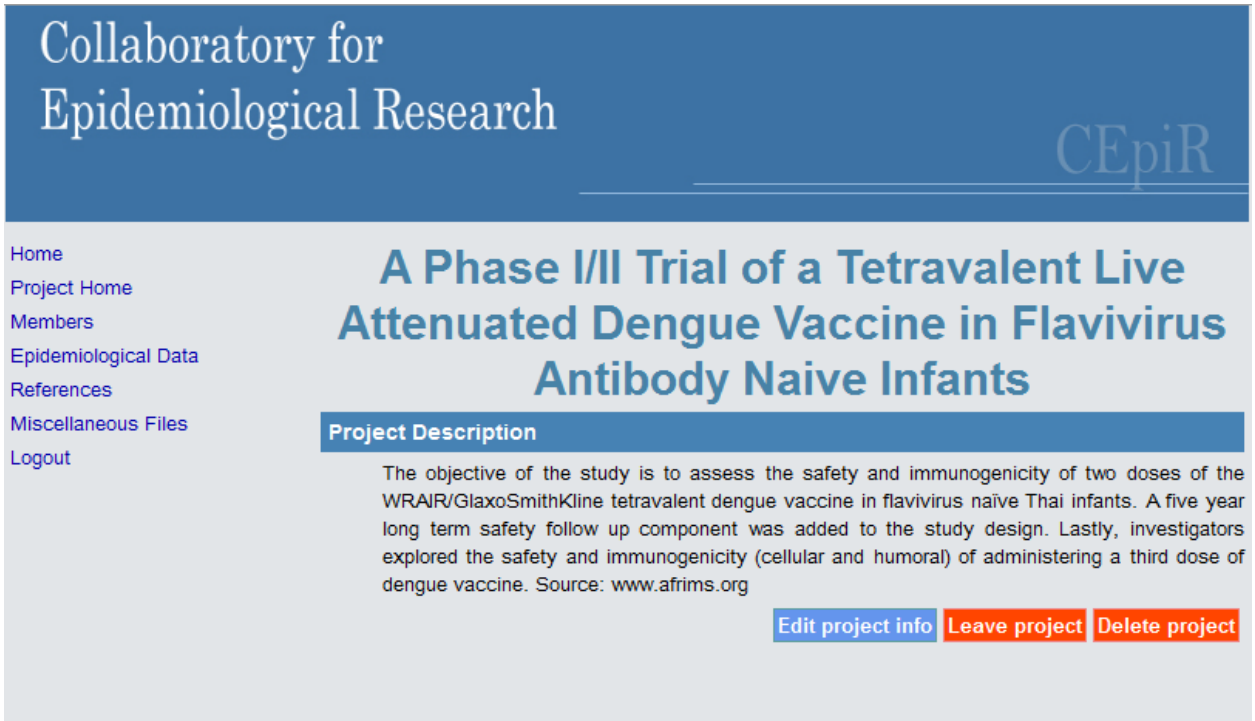


Figure 58 Project Homepage for project administrator/s

Each project in CEpiR has three modules: Epidemiological Data, Miscellaneous Files and References. Any member can add, edit and delete miscellaneous files and references while only the project administrator/s can manage the data entry forms in the Epidemiological Data component. Figure 59 shows the main page of the Miscellaneous files component, Figure 60 shows the Add miscellaneous file page, Figure 61 shows the Add Reference page and Figure 62 shows the Add data entry form page.

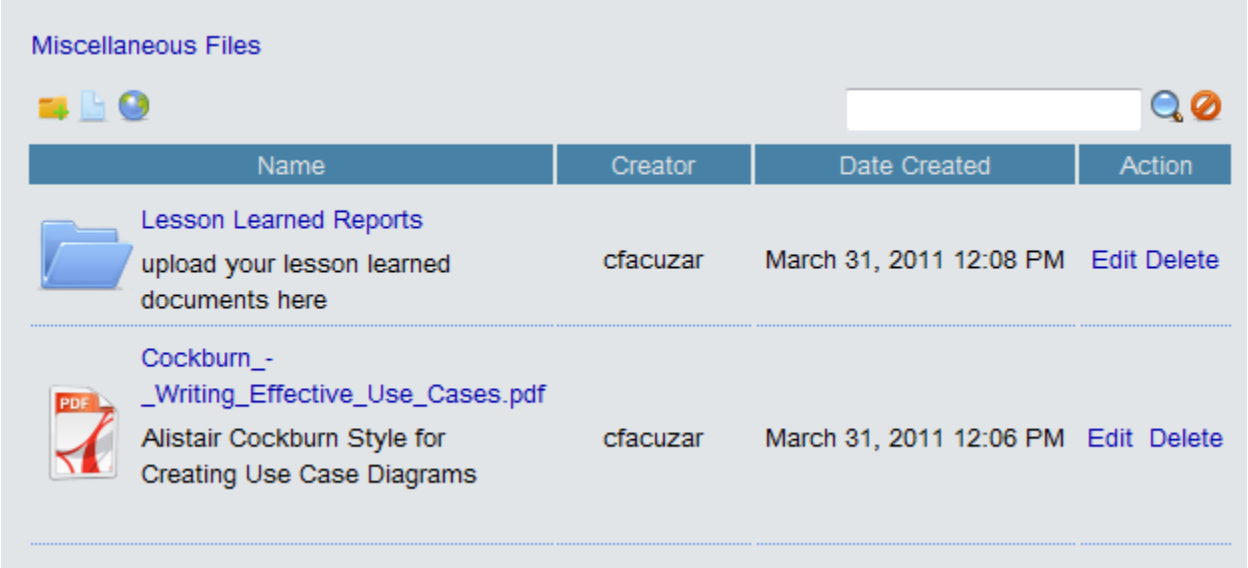


Figure 59 Homepage of Miscellaneous file module

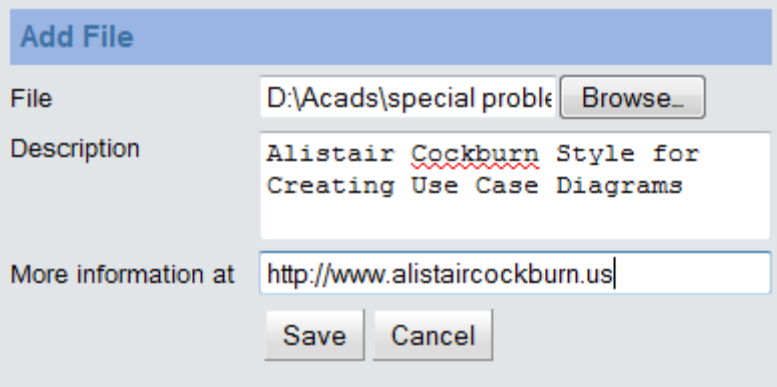


Figure 60 Add Miscellaneous File

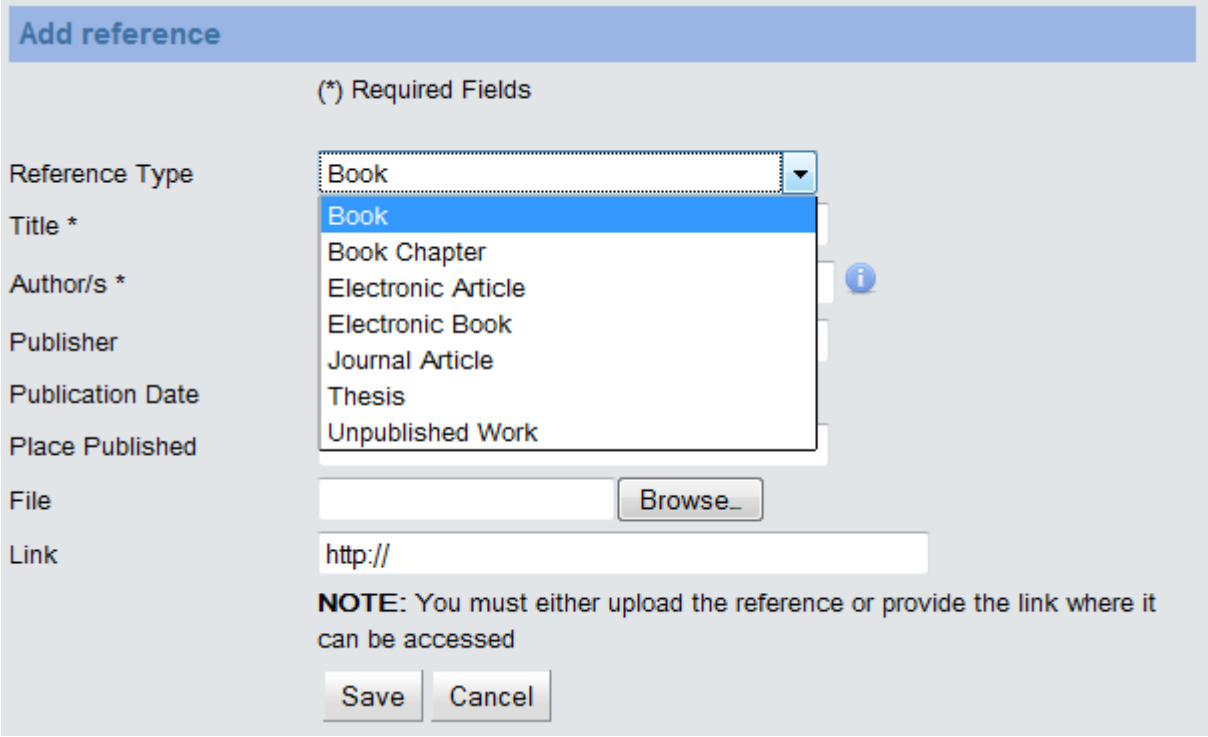


Figure 61 Add reference page

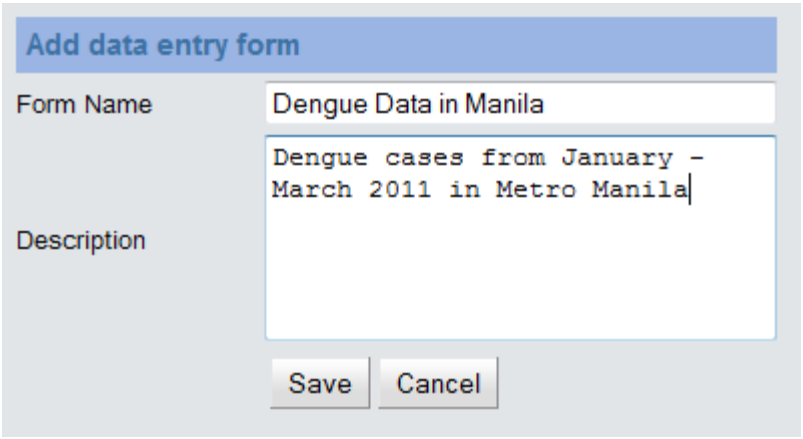
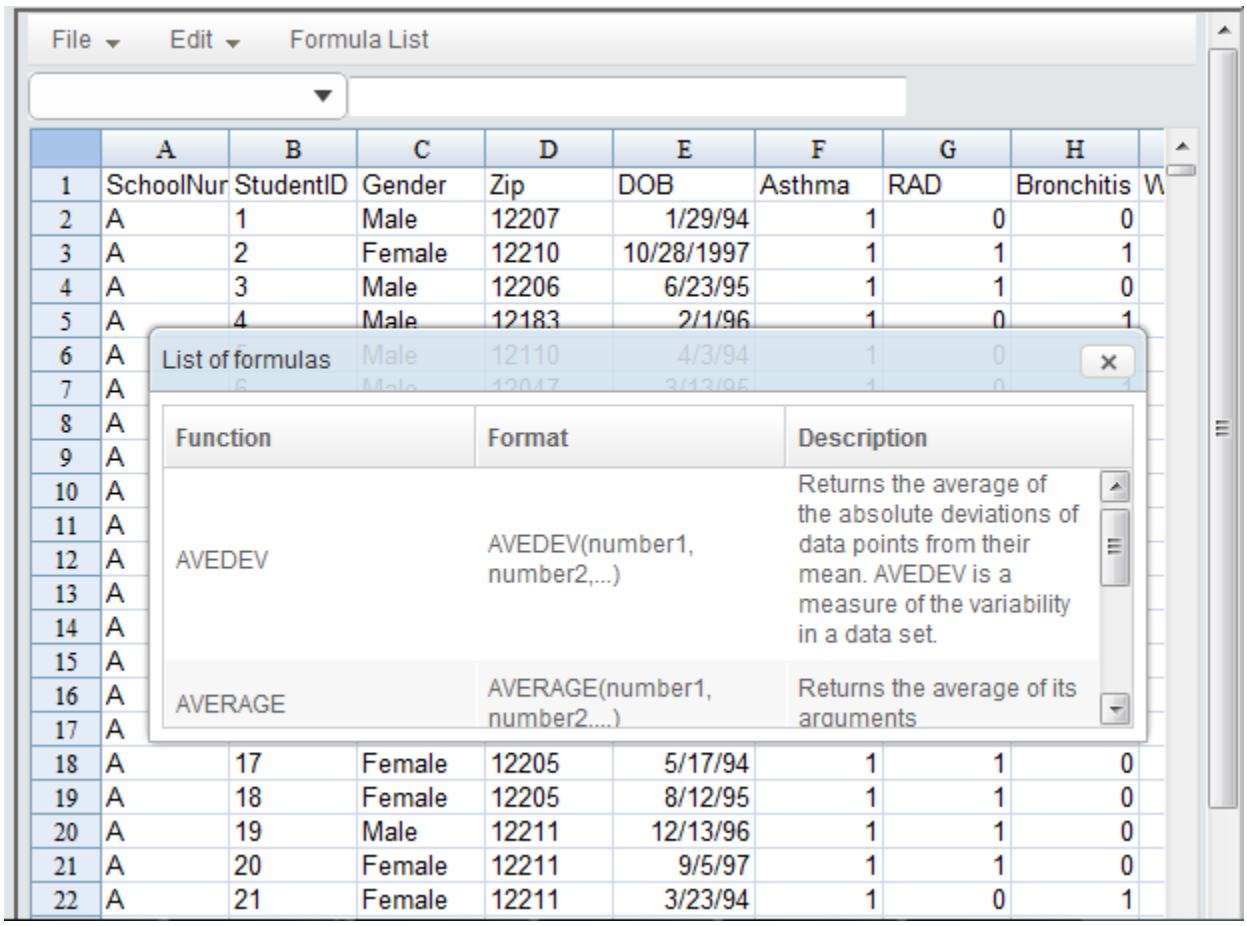


Figure 62 Add data entry form page

Figure 63 shows a data entry page for a project in the system. Project members can add, edit and delete data using this Excel-like editor. A formula bar is also available for entering data and the supported Excel formulas are displayed if the Formula List button is clicked. Finally, project members can save and export the data using the Save and Export option, respectively, in the File menu.



	A	B	C	D	E	F	G	H
1	SchoolNur	StudentID	Gender	Zip	DOB	Asthma	RAD	Bronchitis
2	A	1	Male	12207	1/29/94	1	0	0
3	A	2	Female	12210	10/28/1997	1	1	1
4	A	3	Male	12206	6/23/95	1	1	0
5	A	4	Male	12183	2/1/96	1	0	1
6	A	5	Male	12110	4/3/94	1	0	1
7	A	6	Male	12047	3/13/95	1	0	1
8	A							
9	A							
10	A							
11	A							
12	A							
13	A							
14	A							
15	A							
16	A							
17	A							
18	A	17	Female	12205	5/17/94	1	1	0
19	A	18	Female	12205	8/12/95	1	1	0
20	A	19	Male	12211	12/13/96	1	1	0
21	A	20	Female	12211	9/5/97	1	1	0
22	A	21	Female	12211	3/23/94	1	0	1

Figure 63 Data entry page for a form

If the user is a project administrator, he/she can manage members of the project. Figure 64 shows the members list of the project and the available options for the project administrator.

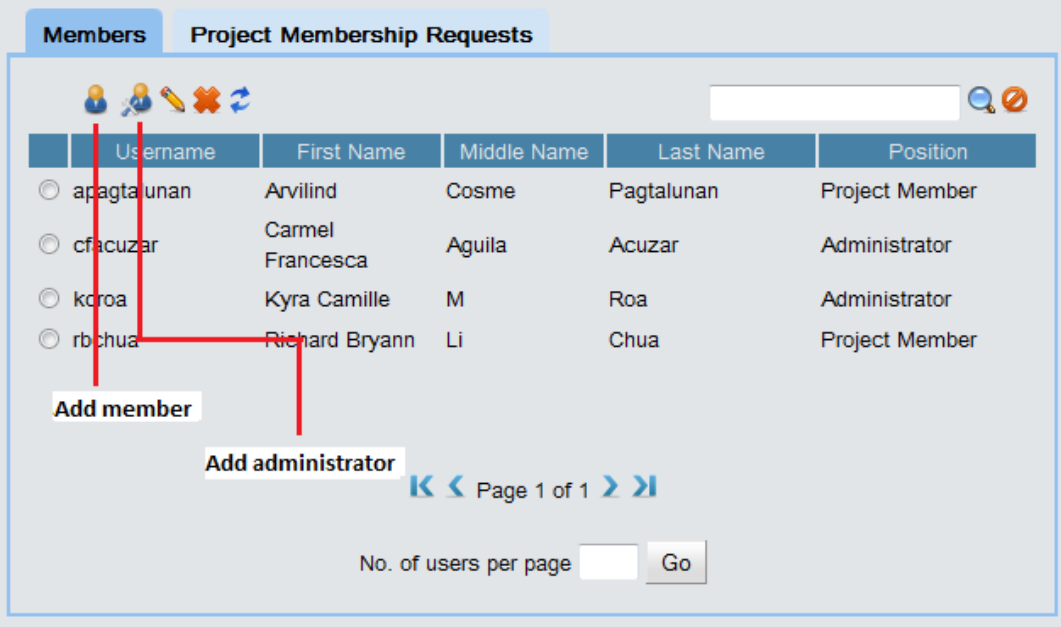


Figure 64 Project members list

Project administrators can also approve or reject project membership requests. Users will be notified via email once their request has been approved / rejected.

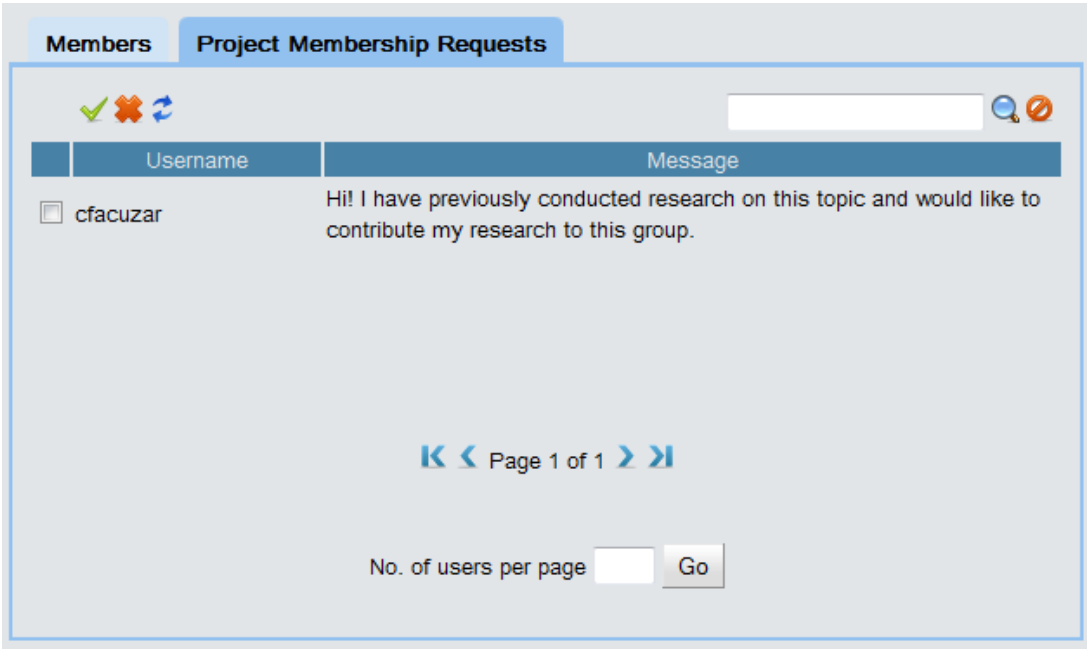


Figure 65 Approve or reject project membership request

System administrators can add, edit and delete any user in the system, except for him/herself. Figure 66 shows the users list of CEpiR and Figure 67 shows the Add user form.

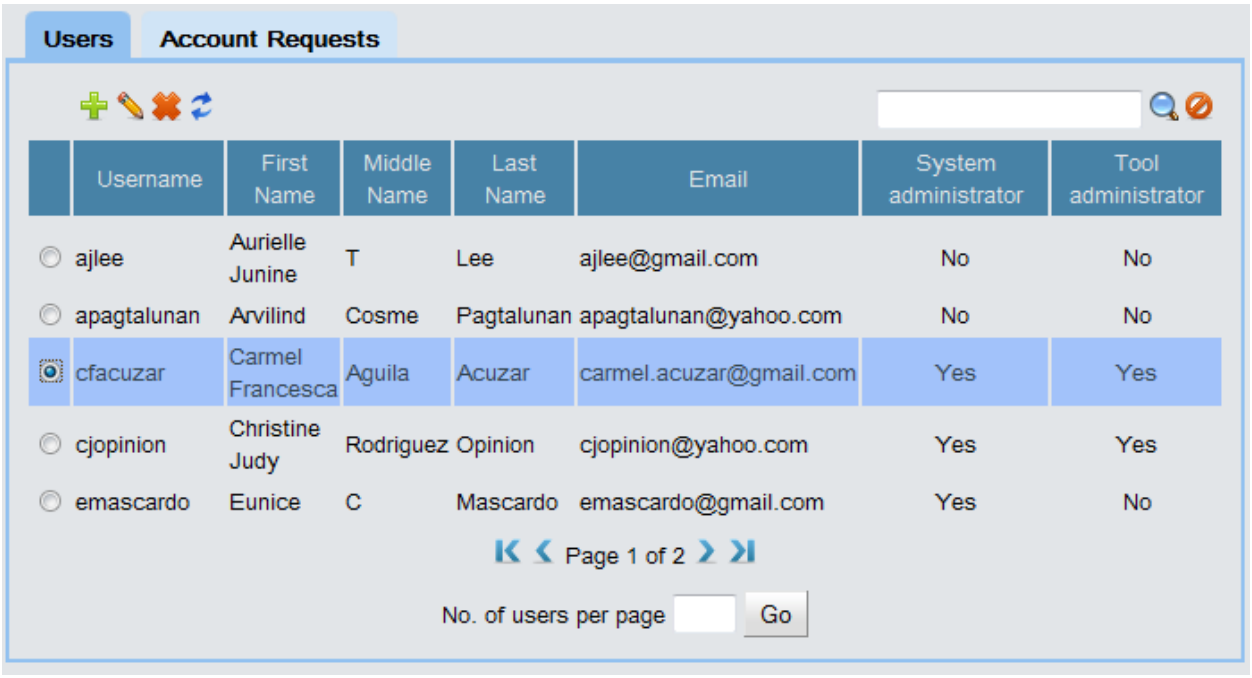


Figure 66 Users list of CEpiR

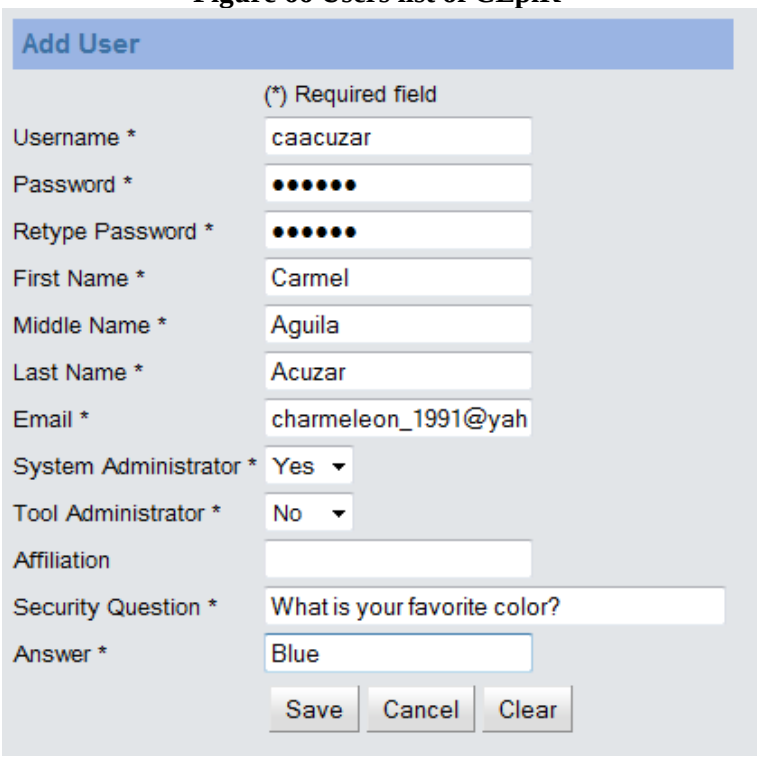


Figure 67 Add user form

System administrators can approve or reject account requests. Users will also be notified if their request has been approved / rejected.

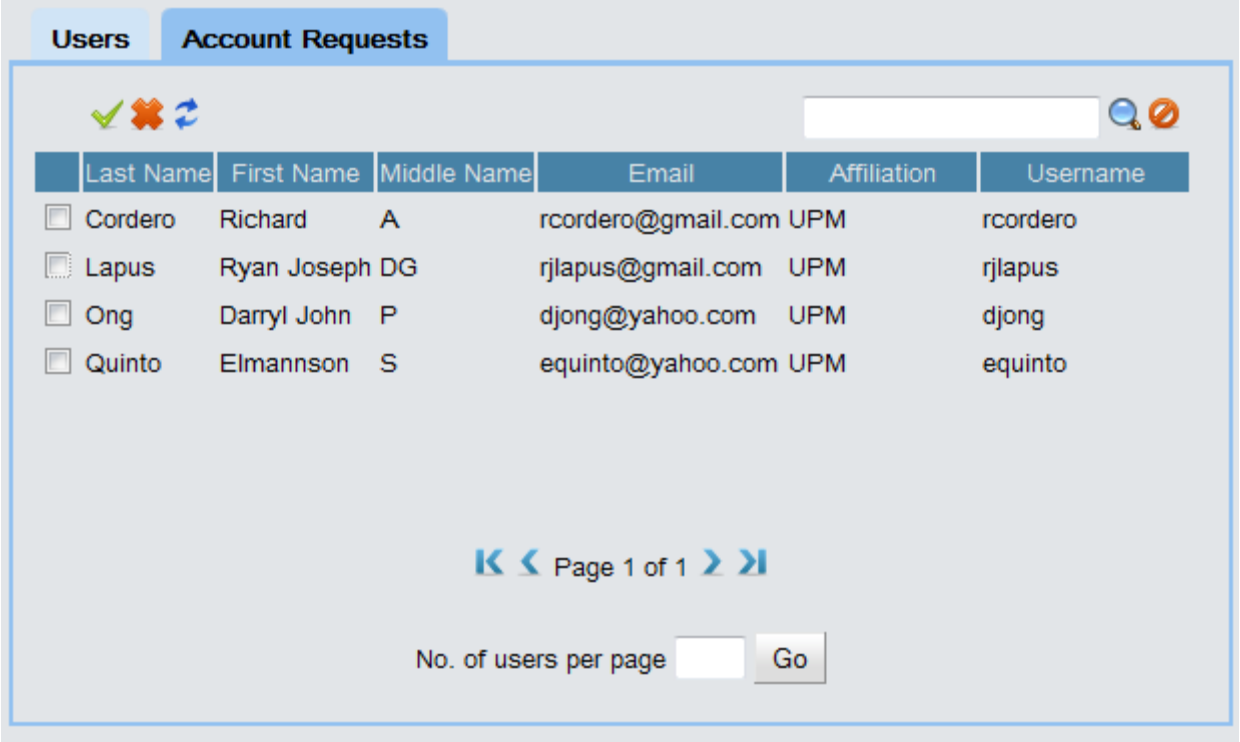


Figure 68 Approve or reject user account request

VII.DISCUSSION

As a web based system, CEpiR was able to provide service to its different users. The system was able to let unregistered users request for account. As for registered users, the system was able to provide them the list of all projects in the system to which they could request for project membership and a service to request for project creation. In addition, registered users can edit their account information, reset their password and view and download the uploaded tools in the system. Only the tool administrators can add, edit and delete tools.

A project member is a registered user that is a member of at least one project. The system was able to provide this type of user access to all modules of the project where he/she belongs to. Project members can add, edit, view and delete epidemiological data and upload and download references and miscellaneous files. Also, the system allowed project members an option to leave the project if they wanted to.

The highest type of user in a project is the project administrator. The system allowed this type of user to manage the memberships of all members in the project and approve / reject project membership requests. He/she was also given rights to edit the project information and also delete the project.

For CEpiR, there is a global user named the system administrator. The system provided this type of user the privilege to add, edit, view and delete user accounts and projects. Also, system administrators are responsible in approving / rejecting user account and project creation requests.

VIII. CONCLUSION

CEpiR is a web-based system that can be useful to Philippine-based researchers and epidemiologists since the system can help these people to collaborate even if they are physically apart. The system provides people in the same project a common place to store data, documents, references and other relevant information and materials for the project.

The system can also prevent research repetition since the system could allow researchers and epidemiologists to become aware of the ongoing researches here in the Philippines. Registered users could just view or search the list of all projects in the system and check their public information.

IX. RECOMMENDATION

CEpiR is a web based system that enables registered users to join projects and share data and information with them. What is lacking in the project component of CEpiR is the option for project members to conduct conversations with co-members. This functionality could help project members, especially those who are physically apart, since information or knowledge can be difficult to transfer at times.

At present, there is no locking mechanism for editing data entry forms. If two users are editing the same form, then race conditions would be present. Therefore, it is highly recommended that a mechanism to resolve race conditions be implemented.

Also, it is advisable that alternative ways on how to enter data be considered since the current method is limited and tends to be inconvenient. Creating a more flexible and convenient method on data entry would allow faster sharing and better representation of data.

X. BIBLIOGRAPHY

1. MSN Encarta. Website, 2010. http://encarta.msn.com
2. Katz, J. S., and Martin, B. R. (1997). What is Research Collaboration? *Research Policy*, 26, 1-18
3. http://www.rcr.emich.edu/module9/i4_benefits.html
4. Science of Collaboratories. Website, 2010. http://www.scienceofcollaboratories.org
5. Zisman, A., and Kramer, J. (1995). Towards Interoperability in Heterogeneous Database Systems.
6. Fleming, E. S., Perkins J., Easa, D., Conde J. G., Baker, R. S., Southerland, W. M., Dottin R., ... Norris, K. C. (2008). Addressing Health Disparities through Multi-institutional, Multidisciplinary Collaboratories. *Ethn Dis*. 18(2 Suppl 2), S2–161-7
7. Bos, N., Zimmerman, A., Olson, J., Yew, J. Yerkie, J., Dahl, E., and Olson, G. (2007). From Shared Databases to Communities of Practice: A Taxonomy of Collaboratories. *Journal of Computer-Mediated Communication*, 12, 652–672
8. Keator, D.B., Grethe, J.S., Marcus, D., Ozyurt, B., Gadde, S., Murphy, S., Pieper, S.; Greve, D., Notestine, R., Bockholt, H.J., and Papadopoulos, P. (2008). A National Human Neuroimaging Collaboratory Enabled by the Biomedical Informatics Research Network (BIRN). *IEEE Transactions on Information Technology in Biomedicine*, 12(2), 162-172
9. Marcus D.S., Olsen T.R., Ramaratnam M., and Buckner R.L. (2007). The Extensible Neuroimaging Archive Toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data. *Neuroinformatics*, 5(1), 11-34.

10. Saltz, J., Oster, S., Hastings, S., Langella, S., Kurc, T., Sanchez, W., Kher, M., Manisundaram, A., Shanbhag, K., and Covitz, P. (2006). caGrid: design and implementation of the core architectureof the cancer biomedical informatics grid. *Bioinformatics*, 22(15), 1910-1916.
11. caBIG Community Site. Website, 2010. https://cabig.nci.nih.gov/
12. Mcwilliam, H., Valentin, F., Goujon, M., Li, W., Narayanasamy, M., Martin, J., Miyar, T., and Lopez, R. (2009). Web services at the European Bioinformatics Institute-2009. *Nucleic Acids Research*, 37.
13. Deshpande, N., Addess, K. J., Bluhm W. F., Merino-Ott, J. C., Townsend-Merino, W., Zhang, Q., Knezevich, C., Xie, L., Chen, L., Feng, Z., Green, R. K., Flippen-Anderson, J. L., Westbrook, J., Berman, H. M., and Bourne, P. E. (2005). The RCSB Protein Data Bank: a redesigned query system and relational database based on the mmCIF schema. *Nucleic Acids Research*, 33, D233-D237.
14. Lesburg, C. A., and Duca, J. S. (2008). soaPDB: a web application for searching the Protein Data Bank, organizing results, and receiving automatic email alerts. *Nucleic Acids Research*, 36, W252-4.
15. da Silva, F. A. B., Gagliardi, H. F., Gallo, E., Madope, M. A., Neto, V. C., Pisa, I. T., and Alves, D. (2007). IntegraEPI: a Grid-based Epidemic Surveillance System. *Stud Health Technol Inform.*, 126, 197-206
16. da Silva, F. A. B., Gagliardi, H. F., Gallo, E., Neto, V. C., and Alves, D. (2007). GISE: A Data Access and Integration Service of Epidemiological Data for a Grid-Based Monitoring and Simulation System.*40th Annual Simulation Symposium ANSS 2007*, 267-274
17. http://www.surffoundation.nl/en/themas/openonderzoek/collaboratories/Pages/Default.aspx

18. Wulf, W. A., Myers, J.D., and Kouzes, R. T. (1996). Collaboratories: Doing Science on the Internet. *Computer*, 29(8), 40-46
19. EDUCAUSE. Website, 2010. http://www.educause.edu/eli
20. BioCoRE. Website, 2010. http://www.ks.uiuc.edu/Research/biocore/
21. WHO – World Health Organization. Website, 2010. http://www.who.int/
22. CDC – Centers for Disease Control and Prevention. Website, 2010. http://cdc.gov/
23. Centers for Disease Control and Prevention.(2006). *Principles of Epidemiology in Public Health Practice: An Introduction to Applied Epidemiology and Biostatistics*. (3rd ed.). Atlanta, Georgia: U.S. Department Of Health And Human Services.
24. Merriam-Webster Online. Website, 2010. http://www.merriam-webster.com
25. World Wide Web Consortium (W3C). Website, 2010. http://www.w3.org
26. http://java.sys-con.com/node/39726
27. http://xml.indelv.com/rpc-soa-rest-the-most-well-known-types-of-web-services.html
28. http://www.w3schools.com
29. Oracle. Website, 2010. http://java.sun.com
30. xFront. Website, 2010. http://www.xfront.com
31. http://www.infoq.com/articles/rest-soap-when-to-use-each
32. http://encyclopedia2.thefreedictionary.com/DBMS
33. http://www.scribd.com/doc/9074928/Homogeneous-Databases
34. Google Web Toolkit. Website, 2011. http://code.google.com/webtoolkit/
35. http://www.buggyprogrammer.co.uk/2010/04/10/model-view-presenter-mvp-architecture/

XI. APPENDIX

CEpiR.gwt.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='cepir'>
  <!-- Inherit the core Web Toolkit stuff.                  -->
  <inherits name='com.google.gwt.user.User'/>

  <!-- Inherit the default GWT style sheet.  You can change       -->
  <!-- the theme of your GWT application by uncommenting          -->
  <!-- any one of the following lines.                            -->
  <inherits name='com.google.gwt.user.theme.standard.Standard'/>

  <!-- <inherits name='com.google.gwt.user.theme.chrome.Chrome'/> -->
  <!-- <inherits name='com.google.gwt.user.theme.dark.Dark'/>     -->

  <!-- Other module inherits                                      -->

  <!-- Specify the app entry point class.                         -->
  <entry-point class='cepir.client.CEpiR'/>

  <!-- Specify the paths for translatable code                    -->
  <source path='client'/>
  <source path='shared'/>
  <public path='images'/>

</module>
```

AppController.java

```
package cepir.client;

import cepir.client.event.*;
import cepir.client.presenter.*;
import cepir.client.view.*;
import cepir.shared.Base64Coder;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.Frame;
import com.google.gwt.user.client.ui.HasWidgets;

public class AppController implements Presenter,
ValueChangeHandler<String>{
	private final HandlerManager eventBus;
	private HasWidgets container;
	private String token;
	private MainView main = new MainView();

	public AppController(HandlerManager eventBus) {
		this.eventBus = eventBus;
		bind();
	}

	private void bind() {
		History.addValueChangeHandler(this);

		eventBus.addHandler(UserLoggedInEvent.TYPE,
				new
UserLoggedInEventHandler() {
			public void
onUserLogIn(UserLoggedInEvent event) {
				doLoginUser();
			}
		});

		eventBus.addHandler(AddUserEvent.TYPE, new
AddUserEventHandler() {

			public void
onAddUser(AddUserEvent event) {
				doAddUser();
			}
		});

		eventBus.addHandler(AddProjectEvent.TYPE,
new AddProjectEventHandler() {

				public void
onAddProject(AddProjectEvent event) {
					doAddProject();
				}
			});

	}

	private void doAddProject() {
		History.newItem("addProject");
	}

	private void doLoginUser(){
		History.newItem("memberIndex");
	}

	private void doAddUser(){
		History.newItem("addUser");
	}

	public void go(final HasWidgets container) {
		this.container = container;

		if ("".equals(History.getToken())) {
			History.newItem("index");
		}
		else {
			History.fireCurrentHistoryState();
		}
	}

	public void onValueChange(ValueChangeEvent<String> event)
{
		token = event.getValue();
		UserServiceAsync rpcService =
GWT.create(UserService.class);
		rpcService.getSessionKeyData("isSysAd", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				if (token != null) {
					Presenter
presenter = null;


	if(token.equals("index")){

	IntroMsgView introMsg = new IntroMsgView();

	presenter = new MainPresenter(eventBus, new MainView(),
new IntroMsgPresenter(introMsg), introMsg);
					}else
if(token.equals("#")){

	History.back();
					}else
if(token.equals("##")){

	History.back();

	History.back();

	History.back();
					}else
if(token.equals("locationError")){

	UnauthorizedNoticeView unauth = new
UnauthorizedNoticeView();

	presenter = new MainPresenter(eventBus, new MainView(),
new UnauthorizedNoticePresenter(unauth,"Location does not exist"),
unauth);
					}else
if(token.equals("unauthorized")){
```

```
            UnauthorizedNoticeView unauth = new
UnauthorizedNoticeView();

            presenter = new MainPresenter(eventBus, new MainView(),
new UnauthorizedNoticePresenter(unauth), unauth);
                                                                    }else if
(token.equals("requestForAcct")){

            RequestForAcctView requestForAcctView = new
RequestForAcctView();

            presenter = new MainPresenter(eventBus, new MainView(),
new RequestForAcctPresenter(eventBus, requestForAcctView),
requestForAcctView);
                                                                    }else
if(token.equals("forgotPass")){

            ForgotPasswordView forgot = new ForgotPasswordView();

            presenter = new MainPresenter(eventBus, new MainView(),
new ForgotPasswordPresenter(eventBus, forgot), forgot);
                                                                    }else
if(result == null){

            History.newItem("index");
                                                                    /*
everything below this part are for members only */
                                                                    }else
if(token.equals("memberIndex")){

            if(result.equals("No")){

            HomeForMemberView home = new HomeForMemberView();

            presenter = new MainPresenter(eventBus, new MainView(),
new HomeForMemberPresenter(eventBus, home), home);

            }else{

            HomeForSysAdView home = new HomeForSysAdView();

            presenter = new MainPresenter(eventBus, new MainView(),
new HomeForSysAdPresenter(eventBus, home), home);

            }
                                                                    }else
if(token.equals("manageAcct")){

            EditMyInfoView edit = new EditMyInfoView();

            presenter = new MainPresenter(eventBus, main, new
EditMyInfoPresenter(eventBus, edit), edit);
                                                                    }else
if(token.startsWith("editProject")){

            if(result.equals("No")){

            ViewProjView proj = new ViewProjView();

            presenter = new MainPresenter(eventBus, main, new
ViewProjPresenter(eventBus, proj, parseForIndex(token), true), proj);

            }else{

            ViewProjView proj = new ViewProjView();

            presenter = new MainPresenter(eventBus, main, new
ViewProjPresenter(eventBus, proj, parseForIndex(token)), proj);

            }
                                                                    }else
if(token.equals("requestForProj")){

            RequestForProjView req = new RequestForProjView();

            presenter = new MainPresenter(eventBus, main, new
RequestForProjPresenter(eventBus, req), req);
                                                                    }else
if(token.equals("myProjects")){

            ViewMyProjsView myProjs = new ViewMyProjsView();

            presenter = new MainPresenter(eventBus, main, new
ViewMyProjsPresenter(eventBus, myProjs),myProjs);
                                                                    }else
if(token.startsWith("pendingProjReq")){

            MyPendingRequestsView pending = new
MyPendingRequestsView();

            presenter = new MainPresenter(eventBus, main, new
MyPendingRequestsPresenter(eventBus, pending, parseForIndex(token)),
pending);
                                                                    }else
if(token.equals("allPendingReq")){

            MemRequestsToApproveView mem = new
MemRequestsToApproveView();

            presenter = new MainPresenter(eventBus, main, new
MemRequestsToApprovePresenter(eventBus, mem), mem);
                                                                    }else
if(token.startsWith("viewMemReq")){

            ViewProjMemRequestView mem = new
ViewProjMemRequestView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new MainPresenter(eventBus, main, new
ViewProjMemRequestPresenter(eventBus, mem, parse[0],
Integer.valueOf(parse[1])), mem);

                                                                    }else
if(token.startsWith("projHome")){

            ProjHomeView projHome = new ProjHomeView();

            presenter = new ProjMainPresenter(eventBus, new
MainView(), new ProjHomePresenter(eventBus, projHome,
parseForIndex(token)), projHome, parseForIndex(token));
                                                                    }else
if(token.equals("allProjs")){

            ViewAllProjsView viewAll = new ViewAllProjsView();

            presenter = new MainPresenter(eventBus, main, new
ViewAllProjsPresenter(eventBus, viewAll), viewAll);
                                                                    }else
if(token.startsWith("members")){

            MemberMngtView member = new MemberMngtView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
MemberMngtPresenter(eventBus, member, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1])), member, Integer.valueOf(parse[0]));
                                                                    }else
if(token.startsWith("editMem")){

            EditMembershipView member = new EditMembershipView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
EditMembershipPresenter(eventBus, member, parse[1],
Integer.valueOf(parse[0])), member, Integer.valueOf(parse[0]));
                                                                    }else
if(token.startsWith("tools")){

            ViewToolsView tool = new ViewToolsView();

            String[] parse = token.split("/",2);

            presenter = new MainPresenter(eventBus, main, new
ViewToolsPresenter(eventBus, tool, parse[1]), tool);
                                                                    }else
if(token.startsWith("addTool")){
```

```
            AddToolView add = new AddToolView();

            presenter = new MainPresenter(eventBus, main, new
AddToolPresenter(eventBus, add, null, parseForLocation(token), 1), add);
                                                                }else
if(token.startsWith("addLink")){

            AddToolView add = new AddToolView();

            presenter = new MainPresenter(eventBus, main, new
AddToolPresenter(eventBus, add, null, parseForLocation(token), 2), add);
                                                                }else
if(token.startsWith("editTool")){

            AddToolView add = new AddToolView();

            presenter = new MainPresenter(eventBus, main, new
AddToolPresenter(eventBus, add, parseForIndex(token), null, 1), add);
                                                                }else
if(token.startsWith("editLink")){

            AddToolView add = new AddToolView();

            presenter = new MainPresenter(eventBus, main, new
AddToolPresenter(eventBus, add, parseForIndex(token), null, 2), add);
                                                                }else
if(token.startsWith("addTFolder")){

            ViewToolFolderView folder = new ViewToolFolderView();

            presenter = new MainPresenter(eventBus, main, new
ViewToolFolderPresenter(eventBus, folder, parseForLocation(token)),
folder);
                                                                }else
if(token.startsWith("editTFolder")){

            ViewToolFolderView folder = new ViewToolFolderView();

            presenter = new MainPresenter(eventBus, main, new
ViewToolFolderPresenter(eventBus, folder, parseForIndex(token)), folder);
                                                                }else
if(token.startsWith("delTFolder")){

            ViewToolsView tool = new ViewToolsView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new MainPresenter(eventBus, main, new
ViewToolsPresenter(eventBus, tool, Integer.valueOf(parse[0]), parse[1], 1),
tool);
                                                                }else
if(token.startsWith("delTool")){

            ViewToolsView tool = new ViewToolsView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new MainPresenter(eventBus, main, new
ViewToolsPresenter(eventBus, tool, Integer.valueOf(parse[0]), parse[1], 2),
tool);
                                                                }else
if(token.startsWith("miscFiles")){

            ViewMiscFilesView file = new ViewMiscFilesView();

            String[] parse = token.split("/",3);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewMiscFilesPresenter(eventBus, file, parse[2], Integer.valueOf(parse[1])),
file, Integer.valueOf(parse[1]));
                                                                }else
if(token.startsWith("addMFile")){

            AddMiscFileView file = new AddMiscFileView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddMiscFilePresenter(eventBus, file, null, Integer.valueOf(parse[0]),
parse[1], 1), file, Integer.valueOf(parse[0]));
                                                                }else
if(token.startsWith("addMLink")){

            AddMiscFileView file = new AddMiscFileView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddMiscFilePresenter(eventBus, file, null, Integer.valueOf(parse[0]),
parse[1], 2), file, Integer.valueOf(parse[0]));
                                                                }else
if(token.startsWith("addMFolder")){

            AddMiscFileFolderView folder = new
AddMiscFileFolderView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddMiscFileFolderPresenter(eventBus, folder, null,
Integer.valueOf(parse[0]), parse[1]), folder, Integer.valueOf(parse[0]));
                                                                }else
if(token.startsWith("editMFile")){

            AddMiscFileView file = new AddMiscFileView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddMiscFilePresenter(eventBus, file, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1]), null, 1), file, Integer.valueOf(parse[1]));
                                                                }else
if(token.startsWith("editMLink")){

            AddMiscFileView file = new AddMiscFileView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddMiscFilePresenter(eventBus, file, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1]), null, 2), file, Integer.valueOf(parse[1]));
                                                                }else
if(token.startsWith("editMFolder")){

            AddMiscFileFolderView folder = new
AddMiscFileFolderView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddMiscFileFolderPresenter(eventBus, folder, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1]), null), folder, Integer.valueOf(parse[1]));
                                                                }else
if(token.startsWith("delMFile")){

            ViewMiscFilesView file = new ViewMiscFilesView();

            String[] parse = parseIntoThreeTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewMiscFilesPresenter(eventBus, file, parse[1], 2,
Integer.valueOf(parse[0]),Integer.valueOf(parse[2])), file,
Integer.valueOf(parse[2]));
                                                                }else
if(token.startsWith("delMFolder")){

            ViewMiscFilesView file = new ViewMiscFilesView();

            String[] parse = parseIntoThreeTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewMiscFilesPresenter(eventBus, file, parse[1], 1,
Integer.valueOf(parse[0]),Integer.valueOf(parse[2])), file,
Integer.valueOf(parse[2]));
                                                                }else
if(token.startsWith("refs")){

            ViewReferencesView file = new ViewReferencesView();
```

```
            String[] parse = token.split("/",3);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewReferencesPresenter(eventBus, file, parse[2],
Integer.valueOf(parse[1])), file, Integer.valueOf(parse[1]));
                                                                    }else
if(token.startsWith("addRFile")){

            AddReferenceView file = new AddReferenceView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddReferencePresenter(eventBus, file, null, Integer.valueOf(parse[0]),
parse[1]), file, Integer.valueOf(parse[0]));
                                                                    }else
if(token.startsWith("addRFolder")){

            AddReferenceFolderView folder = new
AddReferenceFolderView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddReferenceFolderPresenter(eventBus, folder, null,
Integer.valueOf(parse[0]), parse[1]), folder, Integer.valueOf(parse[0]));
                                                                    }else
if(token.startsWith("editRFile")){

            AddReferenceView file = new AddReferenceView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddReferencePresenter(eventBus, file, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1]), null), file, Integer.valueOf(parse[1]));
                                                                    }else
if(token.startsWith("editRFolder")){

            AddReferenceFolderView folder = new
AddReferenceFolderView();

            String[] parse = parseIntoTwoTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddReferenceFolderPresenter(eventBus, folder, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1]), null), folder, Integer.valueOf(parse[0]));
                                                                    }else
if(token.startsWith("delRFile")){

            ViewReferencesView file = new ViewReferencesView();

            String[] parse = parseIntoThreeTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewReferencesPresenter(eventBus, file, parse[1], 2,
Integer.valueOf(parse[0]),Integer.valueOf(parse[2])), file,
Integer.valueOf(parse[2]));
                                                                    }else
if(token.startsWith("delRFolder")){

            ViewReferencesView file = new ViewReferencesView();

            String[] parse = parseIntoThreeTokensWithLoc(token);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewReferencesPresenter(eventBus, file, parse[1], 1,
Integer.valueOf(parse[0]),Integer.valueOf(parse[2])), file,
Integer.valueOf(parse[2]));
                                                                    }else
if(token.startsWith("viewRef")){

            ViewReferenceView file = new ViewReferenceView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
ViewReferencePresenter(eventBus, file, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1])),          file, Integer.valueOf(parse[1]));
                                                                    }else
if(token.startsWith("dataLoad")){

            String[] parse = parseIntoThreeTokensWithLoc(token);

            Frame frame = new
Frame(GWT.getHostPageBaseURL()+"index.zul?p="+parse[0]+"&f="+pars
e[2]+"&u="+Base64Coder.decodeString(parse[1]));

            frame.setHeight("100%");

            frame.setWidth("100%");

            presenter = new ProjMainPresenter(eventBus, main, null,
frame, Integer.valueOf(parse[0]));
                                                                    }else
if(token.startsWith("data")){

            ViewDataEntryFormsView view = new
ViewDataEntryFormsView();

            presenter = new ProjMainPresenter(eventBus, main, new
ViewDataEntryFormsPresenter(eventBus, view, parseForIndex(token)),
view, parseForIndex(token));
                                                                    }else
if(token.startsWith("addForm")){

            AddDataEntryFormView add = new
AddDataEntryFormView();

            presenter = new ProjMainPresenter(eventBus, main, new
AddDataEntryFormPresenter(eventBus, add, parseForIndex(token), null),
add, parseForIndex(token));
                                                                    }else
if(token.startsWith("editForm")){

            AddDataEntryFormView add = new
AddDataEntryFormView();

            String[] parse = parseIntoTwoTokens(token);

            presenter = new ProjMainPresenter(eventBus, main, new
AddDataEntryFormPresenter(eventBus, add, Integer.valueOf(parse[0]),
Integer.valueOf(parse[1])), add, Integer.valueOf(parse[0]));
                                                                    }else
if(result.equals("No")){
                                                                    /*
everything below this part are for sys ads only */
                                                                    }else
if(token.startsWith("userMngt")){

            UserMngtPageView userView = new UserMngtPageView();

            presenter = new MainPresenter(eventBus, main, new
UserMngtPagePresenter(eventBus, userView, parseForIndex(token)),
userView);
                                                                    }else
if(token.startsWith("projMngt")){

            ProjMngtPageView projView = new ProjMngtPageView();

            presenter = new MainPresenter(eventBus, main, new
ProjMngtPagePresenter(eventBus, projView, parseForIndex(token)),
projView);
                                                                    }else
if(token.equals("addUser")){

            AddUserView add = new AddUserView();

            presenter = new MainPresenter(eventBus, main, new
AddUserPresenter(eventBus, add), add);
                                                                    }else
if(token.startsWith("editUser")){

            String username = parseForUsername(token);

            EditUserView edit = new EditUserView();

            presenter = new MainPresenter(eventBus, main, new
EditUserPresenter(eventBus, edit, username), edit);
```

```
                                                }else
if(token.equals("addProject")){

                ViewProjView proj = new ViewProjView();

                presenter = new MainPresenter(eventBus, main, new
ViewProjPresenter(eventBus, proj, null), proj);
                                                }else
if(token.equals("editProject")){

                ViewProjView proj = new ViewProjView();

                presenter = new MainPresenter(eventBus, main, new
ViewProjPresenter(eventBus, proj, parseForIndex(token)), proj);
                                                }

                                                if (presenter
!= null) {

                presenter.go(container);
                                                }

                                                        }
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                        });
                }

                private String parseForUsername(String token) {
                        String[] parsed = token.split("=");
                        return parsed[parsed.length-1];
                }

                private String parseForLocation(String token){
                        String[] parsed = token.split("=",2);
                        return parsed[parsed.length-1];
                }

                private String[] parseIntoTwoTokensWithLoc(String token){
                        String[] parse = token.split("\\?");
                        parse = parse[1].split("&");
                        String[] first = parse[0].split("=");
                        String[] second = parse[1].split("=",2);
                        return new String[]{first[1],second[1]};
                }

                private Integer parseForIndex(String token) {
                        String[] parsed = token.split("=");
                        return Integer.parseInt(parsed[parsed.length-1]);
                }

                private String[] parseIntoTwoTokens(String token){
                        String[] parse = token.split("\\?");
                        parse = parse[1].split("&");
                        String[] first = parse[0].split("=");
                        String[] second = parse[1].split("=");
                        return new String[]{first[1],second[1]};
                }

                private String[] parseIntoThreeTokensWithLoc(String token){
                        String[] parse = token.split("\\?");
                        parse = parse[1].split("&");
                        String[] first = parse[0].split("=");
                        String[] second = parse[1].split("=",2);
                        String[] third = parse[2].split("=");
                        return new String[]{first[1],second[1],third[1]};
                }
}
```

CEpiR.java

```
package cepir.client;


import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.ui.RootPanel;

public class CEpiR implements EntryPoint {

        public void onModuleLoad() {
                HandlerManager eventBus = new
HandlerManager(null);
          AppController appViewer = new AppController(eventBus);
          appViewer.go(RootPanel.get());
        }
}
```

ExcelService.java

```
package cepir.client;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("excelService")
public interface ExcelService extends RemoteService {

}
```

ExcelServiceAsync.java

```
package cepir.client;

public interface ExcelServiceAsync {

}
```

FormService.java

```
package cepir.client;

import java.util.ArrayList;

import cepir.shared.Data;
import cepir.shared.DataEntryForm;
import cepir.shared.Field;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("formService")
public interface FormService extends RemoteService {
        String getSessionKeyData(String key);
        public ArrayList<DataEntryForm> getFormsSearchList(Integer
projID, String query);
        ArrayList<Field> getFormFields(Integer formID);
        void deleteField(Integer formID, Integer fieldID);
        DataEntryForm getForm(Integer formID);
        String saveForm(DataEntryForm form, Boolean isUpdate);
        void deleteForm(Integer formID, Integer projID);
        Field getField(Integer fieldID);
        String saveField(Field field, Boolean isUpdate);
        ArrayList<Data> getRecords(Integer formID);
        Integer[] getFormDimensions(Integer formID);
}
```

FormServiceAsync.java

```
package cepir.client;

import java.util.ArrayList;

import cepir.shared.Data;
import cepir.shared.DataEntryForm;
import cepir.shared.Field;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface FormServiceAsync {

        void getSessionKeyData(String key, AsyncCallback<String>
callback);

        void getFormsSearchList(Integer projID, String query,

        AsyncCallback<ArrayList<DataEntryForm>> callback);

        void getFormFields(Integer formID,
AsyncCallback<ArrayList<Field>> callback);
```

```
        void deleteField(Integer formID, Integer fieldID,
                        AsyncCallback<Void> callback);

        void getForm(Integer formID,
AsyncCallback<DataEntryForm> callback);

        void saveForm(DataEntryForm form, Boolean isUpdate,
                        AsyncCallback<String> callback);

        void deleteForm(Integer formID, Integer projID,
AsyncCallback<Void> callback);

        void getField(Integer fieldID, AsyncCallback<Field> callback);

        void saveField(Field field, Boolean isUpdate,
AsyncCallback<String> callback);

        void getRecords(Integer formID,
AsyncCallback<ArrayList<Data>> callback);

        void getFormDimensions(Integer formID,
AsyncCallback<Integer[]> callback);

}
```

ImageResources.java

```
package cepir.client;

import com.google.gwt.resources.client.ClientBundle;
import com.google.gwt.resources.client.DataResource;

public interface ImageResources extends ClientBundle {
        @Source("images/accept.png")
        DataResource accept();

        @Source("images/add.png")
        DataResource add();

        @Source("images/add_folder.png")
        DataResource addFolder();

        @Source("images/admin.png")
        DataResource admin();

        @Source("images/ascending.png")
        DataResource ascending();

        @Source("images/avi.png")
        DataResource avi();

        @Source("images/back_to_start.png")
        DataResource backToStart();

        @Source("images/blank.png")
        DataResource blank();

        @Source("images/bmp.png")
        DataResource bmp();

        @Source("images/cancel.png")
        DataResource cancel();

        @Source("images/clipboard.png")
        DataResource clipboard();

        @Source("images/delete.png")
        DataResource delete();

        @Source("images/doc.png")
        DataResource doc();

        @Source("images/delete_folder.png")
        DataResource deleteFolder();

        @Source("images/descending.png")
        DataResource descending();

        @Source("images/edit.png")
        DataResource edit();

        @Source("images/erase.png")
        DataResource erase();

        @Source("images/file.png")
        DataResource file();

        @Source("images/flag.png")
        DataResource flag();

        @Source("images/folder.png")
        DataResource folder();

        @Source("images/gear.png")
        DataResource gear();

        @Source("images/gif.png")
        DataResource gif();

        @Source("images/globe.png")
        DataResource globe();

        @Source("images/go_to_last.png")
        DataResource goToLast();

        @Source("images/html.png")
        DataResource html();

        @Source("images/info.png")
        DataResource info();

        @Source("images/jpg.png")
        DataResource jpg();

        @Source("images/link.png")
        DataResource link();

        @Source("images/member.png")
        DataResource member();

        @Source("images/mp3.png")
        DataResource mp3();

        @Source("images/mpeg.png")
        DataResource mpeg();

        @Source("images/next.png")
        DataResource next();

        @Source("images/options.png")
        DataResource options();

        @Source("images/pdf.png")
        DataResource pdf();

        @Source("images/png.png")
        DataResource png();

        @Source("images/ppt.png")
        DataResource ppt();

        @Source("images/previous.png")
        DataResource previous();

        @Source("images/rar.png")
        DataResource rar();

        @Source("images/refresh.png")
        DataResource refresh();

        @Source("images/save.png")
        DataResource save();

        @Source("images/search.png")
        DataResource search();

        @Source("images/small_warning.png")
        DataResource smallWarn();

        @Source("images/site_banner.jpg")
        DataResource siteBanner();
```

```
		@Source("images/table.jpg")
		DataResource table();

		@Source("images/tar.png")
		DataResource tar();

		@Source("images/trash.png")
		DataResource trash();

		@Source("images/txt.png")
		DataResource txt();

		@Source("images/up_folder.png")
		DataResource upFolder();

		@Source("images/user.png")
		DataResource user();

		@Source("images/warning.png")
		DataResource warning();

		@Source("images/wav.png")
		DataResource wav();

		@Source("images/wma.png")
		DataResource wma();

		@Source("images/xls.png")
		DataResource xls();

		@Source("images/zip.png")
		DataResource zip();
}
```

InputChecker.java

```
package cepir.client;

public class InputChecker {

		public boolean isValidUsername(String username){
			return (username.matches("[a-zA-Z][a-zA-Z0-
9_.]{5,23}"));
		}

		public boolean isValidEmail(String email){
			return email.matches("[a-zA-Z][a-zA-Z0-9_.\\-
]*[@][a-zA-Z0-9][a-zA-Z0-9_.\\-]*");
		}

		public boolean isValidFileOrFolderName(String name){
			return name.matches("[^\\?\\:\\*\\\\/\\|\\<\\>]*");
		}

		public boolean isValidYear(String year){
			return year.matches("[0-9]{4}");
		}

		public boolean isValidNumber(String number){
			return number.matches("[0-9][0-9]*");
		}
}
```

MailService.java

```
package cepir.client;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("mailService")
public interface MailService extends RemoteService{
		Boolean postMail( String recipients[ ], String subject, String
message);
}
```

MailServiceAsync.java

```
package cepir.client;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface MailServiceAsync {

		void postMail(String recipients[], String subject, String
message,
				AsyncCallback<Boolean> callback);
}
```

MiscFileService.java

```
package cepir.client;

import java.util.ArrayList;
import java.util.List;

import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("miscFileService")
public interface MiscFileService extends RemoteService {
		String getSessionKeyData(String key);
		public MiscFile getMiscFile(Integer id);
		String saveMiscFile(MiscFile file, Boolean isUpdate);
		MiscFileFolder getFolder(Integer folderID);
		String saveDirectory(MiscFileFolder folder, Boolean isUpdate);
		Boolean checkIfDirExists(String currLoc);
		ArrayList<MiscFileFolder> getFolders(Integer projID,String
currLoc, String query);
		ArrayList<MiscFile> getFiles(Integer projID,String currLoc,
String query);
		List<String> getFolderNamesFromIds(List<String> ids);
		void deleteMiscFileFolder(Integer miscFileFolderID, Integer
projID, String currLoc);
		void deleteMiscFile(Integer fileID, Integer projID, String
currLoc);
}
```

MiscFileServiceAsync.java

```
package cepir.client;

import java.util.ArrayList;
import java.util.List;

import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface MiscFileServiceAsync {

		void getSessionKeyData(String key, AsyncCallback<String>
callback);

		void getMiscFile(Integer id, AsyncCallback<MiscFile>
callback);

		void saveMiscFile(MiscFile file, Boolean isUpdate,
				AsyncCallback<String> callback);

		void getFolder(Integer folderID,
AsyncCallback<MiscFileFolder> callback);

		void saveDirectory(MiscFileFolder folder, Boolean isUpdate,
				AsyncCallback<String> callback);

		void checkIfDirExists(String currLoc,
AsyncCallback<Boolean> callback);

		void getFiles(Integer projID,String currLoc, String query,

		AsyncCallback<ArrayList<MiscFile>> callback);

		void getFolders(Integer projID,String currLoc, String query,

		AsyncCallback<ArrayList<MiscFileFolder>> callback);

		void getFolderNamesFromIds(List<String> ids,
				AsyncCallback<List<String>>
callback);
```

```
	void deleteMiscFileFolder(Integer miscFileFolderID, Integer
projID,
					String currLoc,
AsyncCallback<Void> callback);

	void deleteMiscFile(Integer fileID, Integer projID, String
currLoc,
					AsyncCallback<Void> callback);

}
```

ProjectService.java

```
package cepir.client;

import java.util.ArrayList;

import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("projectService")
public interface ProjectService extends RemoteService{
	String getSessionKeyData(String key);
	Project getProject(Integer projID);
	ArrayList<Project> getProjects();
	ArrayList<Project> getProjectSearchList(String query);
	void deleteProject(Integer projID);
	ArrayList<User> getProjAdmins(Integer projID);
	String addProject(Project proj, ArrayList<String> projAdmin,
Boolean isUpdate);
	Boolean projNameNotUsed(String projName);
	ArrayList<NewProjRequest> getProjRequests();
	ArrayList<NewProjRequest> getProjRequests(String
username);
	ArrayList<NewProjRequest> getProjRequestsSearchList(String
query);
	void deleteProjRequest(String projName);
	String addProjRequest(NewProjRequest newProj);
	Boolean checkIfSoleProjAdmin(String username);
	ArrayList<Project> getProjects(String username);
	ArrayList<ProjMemRequest> getProjMemRequests(String
username);
	ArrayList<ProjMemRequest>
getMemRequestsWhereAdmin(String username);
	void deleteProjCreationRequest(Integer newProjID);
	ArrayList<ProjMemRequest>
getMemRequestsWhereAdminSearchList(String username, String query);
	void approveMemRequest(String username, Integer projID,
Integer memType, Boolean isUpdate);
	void deleteMemRequest(String username, Integer projID);
	ProjMemRequest getProjMemRequest(String username, Integer
projID);
	ArrayList<User> getProjMembers(Integer projID);
	void deleteUserPriv(String username, Integer projID);
	ArrayList<Project> getProjectsWhereAdmin(String username);
	void saveMemRequest(ProjMemRequest request);
	ArrayList<String> getMemberships(String username,
ArrayList<Project> projects);
	ArrayList<User> getProjMembersSearchList(Integer projID,
String query);
	Role getRoleInProj(String username, Integer projID);
	ArrayList<ProjMemRequest> getProjMemRequests(Integer
projID, String query);
	String addProjMembers(Integer projID, ArrayList<String>
projAdmins, Boolean isUpdate, String roleName);
	Boolean projNameNotUsed(String projName, Integer projID);
}
```

ProjectServiceAsync.java

```
package cepir.client;

import java.util.ArrayList;

import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface ProjectServiceAsync {

	void getProjects(AsyncCallback<ArrayList<Project>>
callback);

	void getProjectSearchList(String query,
					AsyncCallback<ArrayList<Project>>
callback);

	void deleteProject(Integer projID, AsyncCallback<Void>
callback);

	void getProject(Integer projID, AsyncCallback<Project>
callback);

	void addProject(Project proj, ArrayList<String> projAdmin,
Boolean isUpdate,
					AsyncCallback<String> callback);

	void projNameNotUsed(String projName,
AsyncCallback<Boolean> callback);

	void
getProjRequests(AsyncCallback<ArrayList<NewProjRequest>> callback);

	void getProjRequestsSearchList(String query,

	AsyncCallback<ArrayList<NewProjRequest>> callback);

	void deleteProjRequest(String projName,
AsyncCallback<Void> callback);

	void getSessionKeyData(String key, AsyncCallback<String>
callback);

	void addProjRequest(NewProjRequest newProj,
AsyncCallback<String> callback);

	void getProjAdmins(Integer projID,
					AsyncCallback<ArrayList<User>>
callback);

	void checkIfSoleProjAdmin(String username,
AsyncCallback<Boolean> callback);

	void getProjects(String username,
AsyncCallback<ArrayList<Project>> callback);

	void getProjRequests(String username,

	AsyncCallback<ArrayList<NewProjRequest>> callback);

	void getProjMemRequests(String username,

	AsyncCallback<ArrayList<ProjMemRequest>> callback);

	void getMemRequestsWhereAdmin(String username,

	AsyncCallback<ArrayList<ProjMemRequest>> callback);

	void deleteProjCreationRequest(Integer newProjID,
					AsyncCallback<Void> callback);

	void getMemRequestsWhereAdminSearchList(String
username, String query,

	AsyncCallback<ArrayList<ProjMemRequest>> callback);

	void approveMemRequest(String username, Integer projID,
Integer memType, Boolean isUpdate,
					AsyncCallback<Void> callback);
```

```
	void deleteMemRequest(String username, Integer projID,
AsyncCallback<Void> callback);

	void getProjMemRequest(String username, Integer projID,
				AsyncCallback<ProjMemRequest>
callback);

	void getProjMembers(Integer projID,
AsyncCallback<ArrayList<User>> callback);

	void deleteUserPriv(String username, Integer projID,
AsyncCallback<Void> callback);

	void getProjectsWhereAdmin(String username,
				AsyncCallback<ArrayList<Project>>
callback);

	void saveMemRequest(ProjMemRequest request,
AsyncCallback<Void> callback);

	void getMemberships(String username, ArrayList<Project>
projects,
				AsyncCallback<ArrayList<String>>
callback);

	void getProjMembersSearchList(Integer projID, String query,
				AsyncCallback<ArrayList<User>>
callback);

	void getRoleInProj(String username, Integer projID,
				AsyncCallback<Role> callback);

	void getProjMemRequests(Integer projID, String query,

	AsyncCallback<ArrayList<ProjMemRequest>> callback);

	void addProjMembers(Integer projID, ArrayList<String>
projAdmins, Boolean isUpdate, String roleName,
				AsyncCallback<String> callback);

	void projNameNotUsed(String projName, Integer projID,
				AsyncCallback<Boolean> callback);

}
```

ReferenceService.java

```
package cepir.client;

import java.util.ArrayList;
import java.util.List;

import cepir.shared.Reference;
import cepir.shared.ReferenceFolder;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("refService")
public interface ReferenceService extends RemoteService {
	String getSessionKeyData(String key);
	public Reference getReference(Integer id);
	String saveReference(Reference file, Boolean isUpdate);
	ReferenceFolder getFolder(Integer folderID);
	String saveDirectory(ReferenceFolder folder, Boolean
isUpdate);
	Boolean checkIfDirExists(String currLoc);
	ArrayList<ReferenceFolder> getFolders(Integer projID,String
currLoc, String query);
	ArrayList<Reference> getFiles(Integer projID,String currLoc,
String query);
	List<String> getFolderNamesFromIds(List<String> ids);
	void deleteReferenceFolder(Integer miscFileFolderID, Integer
projID, String currLoc);
	void deleteReference(Integer fileID, Integer projID, String
currLoc);
}
```

ReferenceServiceAsync.java

```
package cepir.client;

import java.util.ArrayList;
import java.util.List;

import cepir.shared.Reference;
import cepir.shared.ReferenceFolder;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface ReferenceServiceAsync {

	void getSessionKeyData(String key, AsyncCallback<String>
callback);

	void getReference(Integer id, AsyncCallback<Reference>
callback);

	void checkIfDirExists(String currLoc,
AsyncCallback<Boolean> callback);

	void deleteReference(Integer fileID, Integer projID, String
currLoc,
				AsyncCallback<Void> callback);

	void deleteReferenceFolder(Integer miscFileFolderID, Integer
projID,
				String currLoc,
AsyncCallback<Void> callback);

	void getFolder(Integer folderID,
AsyncCallback<ReferenceFolder> callback);

	void getFiles(Integer projID, String currLoc, String query,

	AsyncCallback<ArrayList<Reference>> callback);

	void getFolderNamesFromIds(List<String> ids,
				AsyncCallback<List<String>>
callback);

	void getFolders(Integer projID, String currLoc, String query,

	AsyncCallback<ArrayList<ReferenceFolder>> callback);

	void saveDirectory(ReferenceFolder folder, Boolean isUpdate,
				AsyncCallback<String> callback);

	void saveReference(Reference file, Boolean isUpdate,
				AsyncCallback<String> callback);

}
```

ToolService.java

```
package cepir.client;

import java.util.ArrayList;
import java.util.List;

import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;
import cepir.shared.Tool;
import cepir.shared.ToolFolder;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("toolService")
public interface ToolService extends RemoteService {
	String getSessionKeyData(String key);
	ArrayList<ToolFolder> getFolders(String currLoc, String
query);
	ArrayList<Tool> getFiles(String currLoc, String query);
	ToolFolder getFolder(Integer toolFolderID);
	String saveDirectory(ToolFolder folder, Boolean isUpdate);
	void deleteToolFolder(Integer toolFolderID, String currLoc);
	List<String> getFolderNamesFromIds(List<String> ids);
	Boolean checkIfDirExists(String currLoc);
	Tool getFile(Integer toolID);
	String saveTool(Tool tool, Boolean isUpdate);
	void deleteTool(Integer toolID, String currLoc);
}
```

ToolServiceAsync.java

```
package cepir.client;

import java.util.ArrayList;
import java.util.List;

import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;
import cepir.shared.Tool;
import cepir.shared.ToolFolder;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface ToolServiceAsync {

	void getFolders(String currLoc, String query,

	AsyncCallback<ArrayList<ToolFolder>> callback);

	void getFiles(String currLoc, String query,
				AsyncCallback<ArrayList<Tool>>
callback);

	void getSessionKeyData(String key, AsyncCallback<String>
callback);

	void getFolder(Integer toolFolderID,
AsyncCallback<ToolFolder> callback);

	void saveDirectory(ToolFolder folder, Boolean
isUpdate,AsyncCallback<String> callback);

	void deleteToolFolder(Integer toolFolderID, String currLoc,
AsyncCallback<Void> callback);

	void getFolderNamesFromIds(List<String> ids,
				AsyncCallback<List<String>>
callback);

	void checkIfDirExists(String currLoc,
AsyncCallback<Boolean> callback);

	void getFile(Integer toolID, AsyncCallback<Tool> callback);

	void saveTool(Tool tool, Boolean isUpdate,
AsyncCallback<String> callback);

	void deleteTool(Integer toolID, String currLoc,
AsyncCallback<Void> callback);


}
```

UserService.java

```
package cepir.client;

import java.util.ArrayList;

import cepir.shared.AcctRequest;
import cepir.shared.User;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("userService")
public interface UserService extends RemoteService{
	Boolean checkIfUser(String username, String password);
	String getSessionKeyData(String key);
	void storeSessionData(String key, String value);
	void closeSession();
	User getUser(String username);
	ArrayList<User> getUsersList();
	ArrayList<User> getUsersList(String colToSort, String order);
	String saveRequestForAccount(AcctRequest acct);
	void deleteUser(String username);
	Boolean usernameIsNotUsed(String username);
	Boolean emailIsNotUsed(String email);
	Boolean addUser(User user, Boolean isEncrypted);
	String addUserWithChecking(User user, Boolean isEncrypted);
	String updateUser(User user, Boolean changePass);
	ArrayList<AcctRequest> getAcctRequests();
	void deleteRequest(String username);
	ArrayList<User> getUserSearchList(String query);
	ArrayList<AcctRequest> getAcctRequestSearchList(String
query);
	Boolean checkPasswordMatch(String pass, String hash);
	User getSecurityQuestion(String email);
	void resetPassword(String username, String password);
}
```

UserServiceAsync.java

```
package cepir.client;

import java.util.ArrayList;

import cepir.shared.AcctRequest;
import cepir.shared.User;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface UserServiceAsync {

	void checkIfUser(String username, String password,
				AsyncCallback<Boolean> callback);

	void closeSession(AsyncCallback<Void> callback);

	void getSessionKeyData(String key, AsyncCallback<String>
callback);

	void storeSessionData(String key, String value,
AsyncCallback<Void> callback);

	void getUsersList(AsyncCallback<ArrayList<User>>
callback);

	void saveRequestForAccount(AcctRequest acct,
AsyncCallback<String> callback);

	void deleteUser(String username, AsyncCallback<Void>
callback);

	void usernameIsNotUsed(String username,
AsyncCallback<Boolean> callback);

	void emailIsNotUsed(String email, AsyncCallback<Boolean>
callback);

	void getUsersList(String colToSort, String order,
				AsyncCallback<ArrayList<User>>
callback);

	void addUserWithChecking(User user, Boolean isEncrypted,
AsyncCallback<String> callback);

	void
getAcctRequests(AsyncCallback<ArrayList<AcctRequest>> callback);

	void deleteRequest(String username, AsyncCallback<Void>
callback);

	void addUser(User user, Boolean isEncrypted,
AsyncCallback<Boolean> callback);

	void getUserSearchList(String query,
AsyncCallback<ArrayList<User>> callback);

	void getAcctRequestSearchList(String query,

	AsyncCallback<ArrayList<AcctRequest>> callback);

	void updateUser(User user, Boolean changePass,
				AsyncCallback<String> callback);

	void getUser(String username, AsyncCallback<User>
callback);

	void checkPasswordMatch(String pass, String hash,
				AsyncCallback<Boolean> callback);
```

```
	void getSecurityQuestion(String email, AsyncCallback<User>
callback);

	void resetPassword(String username, String password,
				AsyncCallback<Void> callback);

}
```

AddProjectEvent.java

```
package cepir.client.event;

import com.google.gwt.event.shared.GwtEvent;

public class AddProjectEvent extends
GwtEvent<AddProjectEventHandler>{
	public static Type<AddProjectEventHandler> TYPE = new
Type<AddProjectEventHandler>();

	protected void dispatch(AddProjectEventHandler handler) {
		handler.onAddProject(this);
	}

	public Type<AddProjectEventHandler> getAssociatedType() {
		return TYPE;
	}

}
```

AddProjectEventHandler.java

```
package cepir.client.event;

import com.google.gwt.event.shared.EventHandler;

public interface AddProjectEventHandler extends EventHandler {
	void onAddProject(AddProjectEvent event);
}
```

AddUserEvent.java

```
package cepir.client.event;

import com.google.gwt.event.shared.GwtEvent;
import com.google.gwt.event.shared.GwtEvent.Type;

public class AddUserEvent extends GwtEvent<AddUserEventHandler>{
	public static Type<AddUserEventHandler> TYPE = new
Type<AddUserEventHandler>();

	protected void dispatch(AddUserEventHandler handler) {
		handler.onAddUser(this);
	}

	public Type<AddUserEventHandler> getAssociatedType() {
		return TYPE;
	}

}
```

AddUserEventHandler.java

```
package cepir.client.event;

import com.google.gwt.event.shared.EventHandler;

public interface AddUserEventHandler extends EventHandler {
	void onAddUser(AddUserEvent event);

}
```

UserLoggedInEvent.java

```
package cepir.client.event;

import com.google.gwt.event.shared.GwtEvent;
import com.google.gwt.event.shared.GwtEvent.Type;

public class UserLoggedInEvent extends
GwtEvent<UserLoggedInEventHandler> {
	public static Type<UserLoggedInEventHandler> TYPE = new
Type<UserLoggedInEventHandler>();

	@Override
	public Type<UserLoggedInEventHandler>
getAssociatedType() {
	 return TYPE;
	}

	@Override
	protected void dispatch(UserLoggedInEventHandler handler)
{
	 handler.onUserLogIn(this);
	}
}
```

UserLoggedInEventHandler.java

```
package cepir.client.event;

import com.google.gwt.event.shared.EventHandler;


public interface UserLoggedInEventHandler extends EventHandler {
	void onUserLogIn(UserLoggedInEvent event);
}
```

AddDataEntryFormPresenter.java

```
package cepir.client.presenter;

import cepir.client.FormService;
import cepir.client.FormServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.DataEntryForm;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class AddDataEntryFormPresenter implements Presenter{
	public interface Display{
		void setHeader(String text);
		void setName(String text);
		void setDesc(String text);
		String getName();
		String getDesc();
		HasClickHandlers getSaveButton();
		HasClickHandlers getCancelButton();
		void setErrorTxt(String text);
		Widget asWidget();
	}
	private final FormServiceAsync formService =
GWT.create(FormService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private Integer projID;
	private Integer formID;
	private String user;

	//addForm?p=
	//editForm?p=&id=
	public AddDataEntryFormPresenter(HandlerManager
eventBus, Display display, Integer projID, Integer formID) {
		this.eventBus = eventBus;
		this.display = display;
		this.projID = projID;
		this.formID = formID;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
```

```
		}
		getUsername();
		bind();
	}

	private void bind() {
		display.getSaveButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	if(display.getName().isEmpty()||display.getDesc().isEmpty()){

	display.setErrorTxt("Name / Description is missing");
				}else{

	saveDataEntryForm();
				}
			}
		});

		display.getCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("data?p="+projID);
			}
		});
	}

	protected void saveDataEntryForm() {
		DataEntryForm form = new DataEntryForm();
		if(formID!=null){
			form.formID = formID;
		}
		form.formName = display.getName();
		form.formDesc = display.getDesc();
		User u = new User();
		u.username = user;
		form.creator = u;
		Project proj = new Project();
		proj.projID = projID;
		form.projID = proj;
		formService.saveForm(form,
(formID==null?false:true), new AsyncCallback<String>() {

			public void onSuccess(String result) {
				if(result.equals("ok")){

	History.newItem("data?p="+projID);
				}else{

	display.setErrorTxt(result);
				}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void getUsername() {
		formService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				user = result;
				checkPriv();
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void checkPriv(){
		projService.getRoleInProj(user, projID, new
AsyncCallback<Role>() {

			public void onSuccess(Role result) {

	if(result.roleID==null||result.roleName.isEmpty()){

	History.newItem("unauthorized");
				}else{

	if(formID==null){

	display.setHeader("Add data entry form");
					}else{

	display.setHeader("Edit data entry form");

	loadForm();
					}
				}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void loadForm(){
		formService.getForm(formID, new
AsyncCallback<DataEntryForm>() {

			public void
onSuccess(DataEntryForm result) {

	display.setName(result.formName);

	display.setDesc(result.formDesc);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}
}
```

AddMiscFileFolderPresenter.java

```
package cepir.client.presenter;

import cepir.client.MiscFileService;
import cepir.client.MiscFileServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.MiscFileFolder;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.ToolFolder;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class AddMiscFileFolderPresenter implements Presenter {
	public interface Display{
		Widget asWidget();
		HasClickHandlers getSaveButton();
```

```
                HasClickHandlers getCancelButton();
                void setHeader(String text);
                void setName(String text);
                void setDesc(String text);
                void setErrorTxt(String text);
                String getName();
                String getDesc();
        }
        private final MiscFileServiceAsync miscFileService =
GWT.create(MiscFileService.class);
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final Integer folderID;
        private Integer projID;
        private String folderPath;
        private String user;
        private MiscFileFolder folder;

        //add -> addMFolder?p=<projID>&loc=<path>
        //edit -> editMFolder?id=<folderID>&p=<projId>
        public AddMiscFileFolderPresenter(HandlerManager eventBus,
Display display,
                        Integer folderID, Integer projID,
String folderPath) {
                this.eventBus = eventBus;
                this.display = display;
                this.folderID = folderID;
                this.projID = projID;
                this.folderPath = folderPath;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                getUsername();
                bind();
        }

        private void bind() {
                display.getSaveButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                saveFolder();
                        }
                });

                display.getCancelButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("miscFiles/"+projID+"/"+folderPath);
                        }
                });
        }

        private void getUsername() {
                miscFileService.getSessionKeyData("username",
new AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                user = result;
                                if(folderID!=null){
                                        getFolder();
                                }else{
                                        checkPriv();
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void getFolder(){
                miscFileService.getFolder(folderID, new
AsyncCallback<MiscFileFolder>() {

                        public void
onSuccess(MiscFileFolder result) {
                                if(result!=null){
                                        folder =
result;
                                        projID =
result.project.projID;
                                        folderPath =
result.miscfileFolderPath;
                                        checkPriv();
                                }else{

        History.newItem("locationError");
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void checkPriv(){
                projService.getRoleInProj(user, projID, new
AsyncCallback<Role>() {

                        public void onSuccess(Role result) {

        if(result.roleID==null||result.roleName.isEmpty()){

        History.newItem("unauthorized");
                                }else{

        if(folderID==null){

        display.setHeader("Add folder");
                                        }else{

        display.setHeader("Edit folder");

        display.setDesc(folder.miscfileFolderDesc);

        display.setName(folder.miscfileFolderName);
                                        }
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void saveFolder(){

        if(display.getName().isEmpty()||display.getDesc().isEmpty()){
                        display.setErrorTxt("Required field/s
missing");
                }else{
                        MiscFileFolder folder = new
MiscFileFolder();
                        folder.miscfileFolderID = folderID;
                        folder.miscfileFolderName =
display.getName();
                        folder.miscfileFolderDesc =
display.getDesc();
                        folder.miscfileFolderPath =
folderPath;
                        User u = new User();
                        u.username = user;
                        folder.creator = u;
```

```
                                        Project proj = new Project();
                                        proj.projID = projID;
                                        folder.project = proj;

                                        miscFileService.saveDirectory(folder,
(folderID==null?false:true), new AsyncCallback<String>() {

                                                public void
onSuccess(String result) {

        if(result.equals("ok")){

        History.newItem("miscFiles/"+projID+"/"+folderPath);
                                                }else{

        display.setErrorTxt(result);
                                                }
                                        }

                                                public void
onFailure(Throwable caught) {

        caught.printStackTrace();
                                                }
                                        });
                        }
                }
}
```

AddMiscFilePresenter.java

```
package cepir.client.presenter;

import cepir.client.MiscFileService;
import cepir.client.MiscFileServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.MiscFile;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteEvent;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteHandler;
import com.google.gwt.user.client.ui.FormPanel.SubmitEvent;
import com.google.gwt.user.client.ui.FormPanel.SubmitHandler;

public class AddMiscFilePresenter implements Presenter{
        public interface Display{
                void setHeader(String text);
                void setHidden(String currLoc, String filename);
                void setDescTxt(String text);
                void setTitleTxt(String text);
                void setLink(String text);
                void setErrorTxt(String text);
                String getDescTxt();
                String getTitleTxt();
                String getLink();
                String getFile();
                HasClickHandlers getSaveButton();
                HasClickHandlers getCancelButton();
                void setFileUploadVisibility(Boolean isVisible);
                FormPanel getForm();
                Widget asWidget();
        }
        private final MiscFileServiceAsync miscFileService =
GWT.create(MiscFileService.class);
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final int type; //1 = file, 2 = link
        private final Integer miscFileID;
        private Integer projID;
        private String folderPath;
        private String user;
        private MiscFile miscFile;

        //add -> #addMFile?p=<projID>&loc=<path>
        //edit -> #editMFile?id=<miscfileID>&p=<projID>
        public AddMiscFilePresenter(HandlerManager eventBus,
Display display, Integer miscFileID, Integer projID, String folderPath, int
type) {
                this.eventBus = eventBus;
                this.display = display;
                this.miscFileID = miscFileID;
                this.folderPath = folderPath;
                this.type = type;
                this.projID = projID;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                getUsername();
                bind();
        }

        private void bind() {
                display.getCancelButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("miscFiles/"+projID+"/"+folderPath);
                        }
                });

                display.getSaveButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                if(type==1){

        if(miscFileID==null){

        String[] parse = display.getFile().split("\\\\");

        display.setHidden("uploads/misc_file/"+projID+"/"+folderPath,
parse[parse.length-1]);

        display.getForm().submit();
                                }else{

        saveMiscFile();
                                }
                        }else{
                                saveLink();
                        }
                }
                });

                display.getForm().addSubmitHandler(new
SubmitHandler() {

                        public void onSubmit(SubmitEvent
event) {

        if(display.getFile().isEmpty()){

        display.setErrorTxt("File is missing");

        event.cancel();
                        }else
if(display.getDescTxt().isEmpty()){

        display.setErrorTxt("Description is missing");
```

```
event.cancel();
                                        }
                                }
                        });


        display.getForm().addSubmitCompleteHandler(new
SubmitCompleteHandler() {

                                public void
onSubmitComplete(SubmitCompleteEvent event) {
                                        saveMiscFile();

        History.newItem("miscFiles/"+projID+"/"+folderPath);
                                }
                        });
        }

        private void saveMiscFile(){
                MiscFile file = new MiscFile();
                file.setMiscFileID(miscFileID);
                file.setMiscFileDesc(display.getDescTxt());

        file.setMiscFileLink((display.getLink().equals("http://")?"":disp
lay.getLink()));
                file.setType(type);
                if(type==2){
                        file.miscFileName =
display.getTitleTxt();
                }else if(type==1){
                        String[] parse =
display.getFile().split("\\\\");
                        file.miscFileName =
(miscFileID==null?parse[parse.length-1]:display.getTitleTxt());
                }
                file.miscFilePath = folderPath;
                User u = new User();
                u.username = user;
                file.creator = u;
                Project proj = new Project();
                proj.projID = projID;
                file.project = proj;

                miscFileService.saveMiscFile(file,
(miscFileID==null?false:true), new AsyncCallback<String>() {

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }

                        public void onSuccess(String result) {
                                if(result.equals("ok")){

        History.newItem("miscFiles/"+projID+"/"+folderPath);
                                }else{

        display.setErrorTxt(result);
                                }
                        }
                });
        }

        private void saveLink(){
                if(display.getTitleTxt().isEmpty()){
                        display.setErrorTxt("Title is
missing");
                }else if(display.getDescTxt().isEmpty()){
                        display.setErrorTxt("Description is
missing");
                }else
if(display.getLink().isEmpty()||display.getLink().equalsIgnoreCase("http://")
){
                        display.setErrorTxt("Link is
missing");
                }else if(!display.getLink().startsWith("http://")){
                        display.setErrorTxt("Link should start
with 'http://'");
                }else{
                                saveMiscFile();
                        }
                }

                private void getUsername() {
                        miscFileService.getSessionKeyData("username",
new AsyncCallback<String>() {

                                public void onSuccess(String result) {
                                        user = result;
                                        if(miscFileID!=null){

                getMiscFile();
                                        }else{
                                                checkPriv();
                                        }
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                private void getMiscFile(){
                        miscFileService.getMiscFile(miscFileID, new
AsyncCallback<MiscFile>() {

                                public void onSuccess(MiscFile
result) {
                                        if(result!=null){
                                                projID =
result.project.projID;
                                                folderPath =
result.miscFilePath;

                                                miscFile =
result;
                                                checkPriv();
                                        }else{

                History.newItem("locationError");
                                        }
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                private void checkPriv(){
                        projService.getRoleInProj(user, projID, new
AsyncCallback<Role>() {

                                public void onSuccess(Role result) {

                if(result.roleID==null||result.roleName.isEmpty()){

                History.newItem("unauthorized");
                                        }else{

                if(miscFileID==null){

                display.setHeader("Add "+(type==1?"File":"Link"));

                display.setFileUploadVisibility((type==1?true:false));

                display.setLink("http://");
                                                }else{

                display.setHeader("Edit "+miscFile.miscFileName);

                display.setDescTxt(miscFile.miscFileDesc);

                display.setLink((miscFile.miscFileLink.isEmpty()||miscFile.mis
cFileLink==null?"http://":miscFile.miscFileLink));
```

```
		display.setTitleTxt(miscFile.miscFileName);

		display.setFileUploadVisibility((type==2?false:null));
                                                                }
                                                        }
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
		}

}
```

AddReferenceFolderPresenter.java

```
package cepir.client.presenter;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.ReferenceService;
import cepir.client.ReferenceServiceAsync;
import cepir.shared.MiscFileFolder;
import cepir.shared.Project;
import cepir.shared.Reference;
import cepir.shared.ReferenceFolder;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class AddReferenceFolderPresenter implements Presenter {
	public interface Display{
		Widget asWidget();
		HasClickHandlers getSaveButton();
		HasClickHandlers getCancelButton();
		void setHeader(String text);
		void setName(String text);
		void setDesc(String text);
		void setErrorTxt(String text);
		String getName();
		String getDesc();
	}
	private final ReferenceServiceAsync refService =
GWT.create(ReferenceService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final Integer folderID;
	private Integer projID;
	private String folderPath;
	private String user;

	public AddReferenceFolderPresenter(HandlerManager
eventBus,
			Display display, Integer folderID,
Integer projID, String folderPath) {
		this.eventBus = eventBus;
		this.display = display;
		this.folderID = folderID;
		this.projID = projID;
		this.folderPath = folderPath;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		getUsername();
		bind();
	}

	private void bind() {
		display.getSaveButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				saveFolder();
			}
		});

		display.getCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("miscFiles/"+projID+"/"+folderPath);
			}
		});
	}

	private void getUsername() {
		refService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				user = result;
				checkPriv();
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	protected void checkPriv() {
		projService.getRoleInProj(user, projID, new
AsyncCallback<Role>() {

			public void onSuccess(Role result) {

	if(result.roleID==null||result.roleName.isEmpty()){

	History.newItem("unauthorized");
					}else{

	if(folderID==null){

	display.setHeader("Add folder");
							}else{

	loadFolder();
							}
					}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void loadFolder(){
		refService.getFolder(folderID, new
AsyncCallback<ReferenceFolder>() {

			public void
onSuccess(ReferenceFolder result) {
				folderPath =
result.refFolderPath;
				display.setHeader("Edit
folder");
```

```
		display.setDesc(result.refFolderDesc);

		display.setName(result.refFolderName);
					}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
					}
				});
		}

		private void saveFolder(){

		if(display.getName().isEmpty()||display.getDesc().isEmpty()){
					display.setErrorTxt("Required field/s
missing");
				}else{
					ReferenceFolder folder = new
ReferenceFolder();
					folder.refFolderID = folderID;
					folder.refFolderName =
display.getName();
					folder.refFolderDesc =
display.getDesc();
					folder.refFolderPath = folderPath;
					User u = new User();
					u.username = user;
					folder.creator = u;
					Project proj = new Project();
					proj.projID = projID;
					folder.project = proj;

					refService.saveDirectory(folder,
(folderID==null?false:true), new AsyncCallback<String>() {

						public void
onSuccess(String result) {

		if(result.equals("ok")){

		History.newItem("refs/"+projID+"/"+folderPath);
							}else{

		display.setErrorTxt(result);
							}
						}

						public void
onFailure(Throwable caught) {

		caught.printStackTrace();
						}
					});
				}
		}

}
```

AddReferencePresenter.java

```
package cepir.client.presenter;

import java.util.Arrays;
import java.util.List;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.ReferenceService;
import cepir.client.ReferenceServiceAsync;
import cepir.shared.Project;
import cepir.shared.Reference;
import cepir.shared.ReferenceType;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ChangeEvent;
import com.google.gwt.event.dom.client.ChangeHandler;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasChangeHandlers;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteEvent;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteHandler;
import com.google.gwt.user.client.ui.FormPanel.SubmitEvent;
import com.google.gwt.user.client.ui.FormPanel.SubmitHandler;

public class AddReferencePresenter implements Presenter {
	public interface Display{
		void setHeader(String text);
		void setHidden(String currLoc, String filename);
		String getTitleTxt();
		void setTitleTxt(String text);
		String getChapTitleTxt();
		void setChapTitleTxt(String text);
		String getJournTitleTxt();
		void setJournTitleTxt(String text);
		String getAuthorTxt();
		void setAuthorTxt(String text);
		String getBookTitleTxt();
		void setBookTitleTxt(String text);
		String getEditorTxt();
		void setEditorTxt(String text);
		void setRefType(int selectedIndex);
		void setTypeBox(List<ReferenceType> types);
		Integer getRefType();
		String getPublisherTxt();
		void setPublisherTxt(String text);
		String getPubDate();
		void setPubDate(String date);
		String getPubPlace();
		void setPubPlace(String text);
		String getVolume();
		void setVolume(String text);
		String getIssue();
		void setIssue(String text);
		String getPageNo();
		void setPageNo(String text);
		String getArticleName();
		void setArticleName(String text);
		String getYear();
		void setYear(String text);
		String getSchool();
		void setSchool(String text);
		String getLink();
		void setLink(String text);
		String getFile();
		void setErrorTxt(String text);
		void showOverWriteFileWarn(Boolean visible);
		HasClickHandlers getSaveButton();
		HasClickHandlers getCancelButton();
		HasChangeHandlers getRefOption();
		FormPanel getForm();
		void setFileUploadVisibility(Boolean isVisible);
		void setNote(Boolean isVisible);
		void hideAllFields();
		void showBookForm();
		void showBookChapForm();
		void showJournalArticleForm();
		void showEArticleForm();
		void showUnpublishedWorkForm();
		void showThesisForm();
		void setRefTypeEnabled(Boolean enabled);
		Widget asWidget();
	}
	private final ReferenceServiceAsync refService =
GWT.create(ReferenceService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private Integer refID;
	private Integer projID;
```

```
	private String folderPath;
	private String user;
	private List<ReferenceType> types;
	private String refName;

	public AddReferencePresenter(HandlerManager eventBus,
Display display,
				Integer refID, Integer projID, String
folderPath) {
		this.eventBus = eventBus;
		this.display = display;
		this.refID = refID;
		this.projID = projID;
		this.folderPath = folderPath;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		getUsername();
		bind();
	}

	private void bind() {
		display.getCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("refs/"+projID+"/"+folderPath);
			}
		});

		display.getSaveButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	if(display.getFile().isEmpty()){
								saveFile();
						}else{
								String[]
parse = display.getFile().split("\\\\");

	display.setHidden("uploads/ref/"+projID+"/"+folderPath,parse[
parse.length-1]);

	display.getForm().submit();
						}
			}
		});

		display.getForm().addSubmitHandler(new
SubmitHandler() {

			public void onSubmit(SubmitEvent
event) {

	if(hasMissingFields()||((display.getLink().isEmpty()||display.get
Link().equals("http://"))&&display.getFile().isEmpty())){

	display.setErrorTxt("Required field/s missing");

	event.cancel();
						}
			}
		});


	display.getForm().addSubmitCompleteHandler(new
SubmitCompleteHandler() {

			public void
onSubmitComplete(SubmitCompleteEvent event) {
						saveFile();
			}
		});

		display.getRefOption().addChangeHandler(new
ChangeHandler() {

			public void onChange(ChangeEvent
event) {
				display.hideAllFields();
				clearAllFields();
				loadForm();
			}
		});
	}

	protected void clearAllFields() {
		display.setTitleTxt("");
		display.setAuthorTxt("");
		display.setChapTitleTxt("");
		display.setJournTitleTxt("");
		display.setBookTitleTxt("");
		display.setEditorTxt("");
		display.setPublisherTxt("");
		display.setPubDate("");
		display.setPubPlace("");
		display.setVolume("");
		display.setIssue("");
		display.setPageNo("");
		display.setArticleName("");
		display.setYear("");
		display.setSchool("");
		display.setErrorTxt("");
	}

	private void loadForm(){

	if(display.getRefType()==0||display.getRefType()==3){ //book,
e-book
			display.showBookForm();
		}else if(display.getRefType()==1){ //book chapter
			display.showBookChapForm();
		}else if(display.getRefType()==2){ //e-article
			display.showEArticleForm();
		}else if(display.getRefType()==4){ //journal
article
			display.showJournalArticleForm();
		}else if(display.getRefType()==5){ //thesis
			display.showThesisForm();
		}else if(display.getRefType()==6){ //unpublished
work

	display.showUnpublishedWorkForm();
		}
	}

	public Boolean hasMissingFields(){

	if(display.getRefType()==0||display.getRefType()==3){ //book,
e-book

	if(display.getTitleTxt().isEmpty()||display.getAuthorTxt().isEm
pty()||display.getPubDate().isEmpty()){
				return true;
			}
		}else if(display.getRefType()==1){ //book chapter

	if(display.getChapTitleTxt().isEmpty()||display.getAuthorTxt().
isEmpty()||display.getBookTitleTxt().isEmpty()||display.getPubDate().isEmp
ty()){
				return true;
			}
		}else if(display.getRefType()==2){ //e-article

	if(display.getTitleTxt().isEmpty()||display.getAuthorTxt().isEm
pty()){
				return true;
			}
		}else if(display.getRefType()==4){ //journal
article

	if(display.getJournTitleTxt().isEmpty()||display.getAuthorTxt().
```

```
isEmpty()||display.getArticleName().isEmpty()||display.getPubDate().isEmpty()){
                                                return true;
                                        }
                                }else if(display.getRefType()==5){ //thesis

                                if(display.getTitleTxt().isEmpty()||display.getAuthorTxt().isEmpty()||display.getYear().isEmpty()||display.getSchool().isEmpty()){
                                                return true;
                                        }
                                }else if(display.getRefType()==6){ //unpublished work

                                if(display.getTitleTxt().isEmpty()||display.getAuthorTxt().isEmpty()){
                                                return true;
                                        }
                                }
                                return false;
                        }

                        protected void saveFile() {

                        if(hasMissingFields()||((display.getLink().isEmpty()||display.getLink().equals("http://"))&&display.getFile().isEmpty()&&(refName==null||refName.isEmpty()))){
                                        display.setErrorTxt("Required field/s missing");
                                }else{
                                        Reference file = new Reference();
                                        if(!display.getFile().isEmpty()){
                                                String[] parse = display.getFile().split("\\\\");
                                                file.refName = parse[parse.length-1];
                                        }else{
                                                file.refName = "";
                                        }
                                        file.refLink = (display.getLink().isEmpty()||display.getLink().equals("http://")?"":display.getLink());
                                        file.refPath = folderPath;
                                        file.refID = refID;
                                        file.bookTitle = display.getBookTitleTxt();
                                        file.author = display.getAuthorTxt();
                                        file.publisher = display.getPublisherTxt();
                                        file.type = types.get(display.getRefType());
                                        file.pubDate = display.getPubDate();
                                        file.volume = display.getVolume();
                                        file.issue = display.getIssue();
                                        file.chapTitle = display.getChapTitleTxt();
                                        file.journTitle = display.getJournTitleTxt();
                                        file.title = display.getTitleTxt();
                                        file.editor = display.getEditorTxt();
                                        file.pubPlace = display.getPubPlace();
                                        file.pageNum = display.getPageNo();
                                        file.articleName = display.getArticleName();
                                        file.year = display.getYear();
                                        file.school = display.getSchool();
                                        User u = new User();
                                        u.username = user;
                                        file.creator = u;
                                        Project proj = new Project();
                                        proj.projID = projID;
                                        file.project = proj;

                                        refService.saveReference(file, (refID==null?false:true), new AsyncCallback<String>() {

                                                public void onSuccess(String result) {

                        if(result.equals("ok")){
                                History.newItem("refs/"+projID+"/"+folderPath);
                                                }else{
                        display.setErrorTxt(result);
                                                }
                                        }

                                        public void onFailure(Throwable caught) {

                        caught.printStackTrace();
                                        }
                                });
                        }
                }

                private void getUsername(){
                        refService.getSessionKeyData("username", new AsyncCallback<String>() {

                                public void onSuccess(String result) {
                                        user = result;
                                        checkPriv();
                                }

                                public void onFailure(Throwable caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                protected void checkPriv() {
                        projService.getRoleInProj(user, projID, new AsyncCallback<Role>() {

                                public void onSuccess(Role result) {

                        if(result.roleID==null||result.roleName.isEmpty()){

                        History.newItem("unauthorized");
                                                }else{
                                                        types = Arrays.asList(ReferenceType.values());

                        display.setTypeBox(types);

                        if(refID==null){

                        display.setHeader("Add reference");

                        display.showBookForm();
//
                        display.setFileUploadVisibility(true);
//
                        display.setNote(true);

                        display.showOverWriteFileWarn(false);

                        display.setLink("http://");

                                                }else{

                        loadReference();
                                                }
                                        }
                                }

                                public void onFailure(Throwable caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                protected void loadReference() {
                        refService.getReference(refID, new AsyncCallback<Reference>() {
```

```
                                        public void onSuccess(Reference
result) {
                                                refName =
result.refName;
                                                folderPath =
result.refPath;
                                                display.setHeader("Edit
reference");

                display.setTitleTxt(result.title);

                display.setAuthorTxt(result.author);

                display.setPublisherTxt(result.publisher);

                display.setPubDate(result.pubDate);

                display.setRefType(types.indexOf(result.type));

                display.setRefTypeEnabled(false);

                display.setVolume(result.volume);

                display.setIssue(result.issue);

                display.setBookTitleTxt(result.bookTitle);

                display.setChapTitleTxt(result.chapTitle);

                display.setJournTitleTxt(result.journTitle);

                display.setEditorTxt(result.editor);

                display.setPubPlace(result.pubPlace);

                display.setPageNo(result.pageNum);

                display.setArticleName(result.articleName);

                display.setYear(result.year);

                display.setSchool(result.school);

                display.setLink((result.refLink.isEmpty()?"http://":result.refLin
k));
                                                loadForm();

                if(result.refName==null||result.refName.isEmpty()){

                display.showOverWriteFileWarn(false);
                                                }
//
                display.setFileUploadVisibility((result.refName.isEmpty()?true:
false));
//                                              display.setNote(false);
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                        });
                }

}
```

AddToolPresenter.java

```
package cepir.client.presenter;

import cepir.client.ToolService;
import cepir.client.ToolServiceAsync;
import cepir.shared.Tool;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteEvent;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteHandler;
import com.google.gwt.user.client.ui.FormPanel.SubmitEvent;
import com.google.gwt.user.client.ui.FormPanel.SubmitHandler;

public class AddToolPresenter implements Presenter {
        public interface Display{
                String getFile();
                void setHidden(String currLoc, String filename);
                void setDescTxt(String text);
                String getDescTxt();
                void setTitleTxt(String text);
                String getTitleTxt();
                void setLink(String text);
                String getLink();
                void setHeaderTxt(String text);
                void setErrorTxt(String text);
                void setNameErrorTxt(String text);
                void setLinkOrFileErrorTxt(String text);
                HasClickHandlers getSaveButton();
                HasClickHandlers getCancelButton();
                void setVisibilityUploader(Boolean isVisible);
                void setVisibilityTitle(Boolean isVisible);
                FormPanel getForm();
                Widget asWidget();
        }
        private final ToolServiceAsync toolService =
GWT.create(ToolService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final int type; //1 = file, 2 = link
        private final Integer toolID;
        private String folderPath;
        private String user;
        private String fileName;

        public AddToolPresenter(HandlerManager eventBus, Display
display, Integer toolFolderID, String folderPath, int type) {
                this.eventBus = eventBus;
                this.display = display;
                this.toolID = toolFolderID;
                this.folderPath = folderPath;
                this.type = type;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                bind();
                checkIfToolAd();
        }

        private void checkIfToolAd(){
                toolService.getSessionKeyData("isToolAd", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                if(result.equals("Yes")){

        if(folderPath!=null){

        checkIfLocExists();
                                                }else{

        getUsername();
                                                }
                                        }else{

        History.newItem("unauthorized");
                                        }
                        }

                        public void onFailure(Throwable
caught) {
```

```
                                        caught.printStackTrace();
                                }
                        });
        }

        private void checkIfLocExists(){
                toolService.checkIfDirExists(folderPath, new
AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{
                                if(result){

        getUsername();
                                }else{

        History.newItem("locationError");
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void bind() {
                display.getCancelButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("tools/"+folderPath);
                        }
                });

                display.getSaveButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                if(type==1){

        if(toolID==null){

        String[] parse = display.getFile().split("\\\\");

        display.setHidden("uploads/tool/"+folderPath,parse[parse.lengt
h-1]);

        display.getForm().submit();
                                        }else{

        saveTool();
                                        }
                                }else{
                                        saveLink();
                                }
                        }
                });

                display.getForm().addSubmitHandler(new
SubmitHandler() {

                        public void onSubmit(SubmitEvent
event) {

        if(display.getFile().isEmpty()){

        display.setErrorTxt("File is missing");

        event.cancel();
                                }else
if(display.getDescTxt().isEmpty()){

        display.setErrorTxt("Description is missing");

        event.cancel();
                                }
                        }
                });


        display.getForm().addSubmitCompleteHandler(new
SubmitCompleteHandler() {

                        public void
onSubmitComplete(SubmitCompleteEvent event) {
                                saveTool();

        History.newItem("tools/"+folderPath);
                        }
                });
        }

        private void saveLink(){
                if(display.getTitleTxt().isEmpty()){
                        display.setErrorTxt("Title is
missing");
                }else if(display.getDescTxt().isEmpty()){
                        display.setErrorTxt("Description is
missing");
                }else
if(display.getLink().isEmpty()||display.getLink().equalsIgnoreCase("http://")
){
                        display.setErrorTxt("Link is
missing");
                }else if(!display.getLink().startsWith("http://")){
                        display.setErrorTxt("Link should start
with 'http://'");
                }else{
                        saveTool();
                }
        }

        private void saveTool(){
                Tool tool = new Tool();
                tool.toolID = toolID;
                tool.toolDesc = display.getDescTxt();
                tool.toolLink =
(display.getLink().equals("http://")?"":display.getLink());
                tool.type = type;
                if(type==2){
                        tool.toolName = display.getTitleTxt();
                }else if(type==1){
                        String[] parse =
display.getFile().split("\\\\");
                        tool.toolName =
(toolID==null?parse[parse.length-1]:fileName);
                }
                tool.toolPath = folderPath;
                User u = new User();
                u.username = user;
                tool.creator = u;

        toolService.saveTool(tool,(toolID==null?false:true), new
AsyncCallback<String>() {

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }

                        public void onSuccess(String result) {
                                if(result.equals("ok")){

        History.newItem("tools/"+folderPath);
                                }else{

        display.setErrorTxt(result);
                                }
                        }
                });
        }

        private void getUsername() {
                toolService.getSessionKeyData("username", new
AsyncCallback<String>() {
```

```
                                        public void onSuccess(String result) {
                                                user = result;
                                                if(toolID==null){

                display.setHeaderTxt("Add "+(type==1?"File":"Link"));

                display.setVisibilityTitle((type==1?false:true));

                display.setVisibilityUploader((type==1?true:false));

                display.setLink("http://");
                                                }else{

                loadToolDetails();
                                                }
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                        });
                }

                private void loadToolDetails(){
                        toolService.getFile(toolID, new
AsyncCallback<Tool>() {

                                        public void onSuccess(Tool result) {
                                                folderPath =
result.toolPath;

                display.setHeaderTxt("Edit
"+(type==1?result.toolName:"Link"));

                display.setDescTxt(result.toolDesc);

                display.setLink((result.toolLink==null||result.toolLink.isEmpty(
)?"http://":result.toolLink));

                display.setVisibilityUploader(false);
                                                if(type==2){


                display.setTitleTxt(result.toolName);
                                                }else if(type==1){
                                                        fileName =
result.toolName;

                display.setVisibilityTitle(false);
                                                }
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                        });
                }

}
```

AddUserPresenter.java

```
package cepir.client.presenter;

import cepir.client.InputChecker;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.User;
import cepir.shared.YesNo;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class AddUserPresenter implements Presenter{
        public interface Display{
                HasClickHandlers getOkButton();
                HasClickHandlers getCancelButton();
                HasClickHandlers getClearButton();
                void clearForm();
                void setErrorTxt(String errMsg,Boolean isError);
                void setUsernameMsgTxt(String usernameMsg);
                void setEmailMsgTxt(String emailMsg);
                void setRetypePasswdMsgTxt(String msg);
                String getFirstName();
                String getMiddleName();
                String getLastName();
                String getEmail();
                String getAffiliation();
                String getUsername();
                String getPassword();
                String getRetypedPassword();
                String getSysAdPriv();
                String getToolAdPriv();
                String getQuestion();
                String getAnswer();
                Widget asWidget();
        }

        private final UserServiceAsync rpcService =
GWT.create(UserService.class);
        private final HandlerManager eventBus;
        private final Display display;

        public AddUserPresenter(HandlerManager eventBus, Display
display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
        }

        public void bind(){
                display.getClearButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                display.clearForm();
                        }
                });

                display.getCancelButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                History.newItem("#");

        History.newItem("userMngt?tab=0");
                        }
                });

                display.getOkButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                addUser();
                        }
                });
        }

        private void checkUsernameAndEmailIfUsed(){
```

```
		rpcService.usernameIsNotUsed(display.getUsername(), new
AsyncCallback<Boolean>() {

					public void onSuccess(Boolean result)
{
						if(result==false){

		display.setUsernameMsgTxt("Username already in use");
						}else{

		display.setUsernameMsgTxt("");
						}
					}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
					}
				});

				rpcService.emailIsNotUsed(display.getEmail(),
new AsyncCallback<Boolean>() {

					public void onSuccess(Boolean result)
{
						if(result==false){

		display.setEmailMsgTxt("Email already in use");
						}else{

		display.setEmailMsgTxt("");
						}
					}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
					}
				});
		}

		private Boolean passwordMatches(){
			return
display.getPassword().equals(display.getRetypedPassword());
		}

		private void addUser(){
			InputChecker input = new InputChecker();

		if(!input.isValidUsername(display.getUsername())){
					display.setEmailMsgTxt("");
					display.setErrorTxt("",true);
					display.setRetypePasswdMsgTxt("");

		display.setUsernameMsgTxt("Username must start with a letter,
must be 6-24 characters long and should contain
letters,numbers,underscores(_) and dots(.) only");
				}else if(!input.isValidEmail(display.getEmail())){
					display.setUsernameMsgTxt("");
					display.setErrorTxt("",true);
					display.setRetypePasswdMsgTxt("");
					display.setEmailMsgTxt("Invalid
email address");
				}else
if(display.getEmail().isEmpty()||display.getFirstName().isEmpty()||display.g
etLastName().isEmpty()||

		display.getMiddleName().isEmpty()||display.getUsername().isE
mpty()||display.getPassword().isEmpty()||display.getQuestion().isEmpty()||di
splay.getAnswer().isEmpty()){
					display.setErrorTxt("Required field/s
missing",true);
					display.setUsernameMsgTxt("");
					display.setEmailMsgTxt("");
					display.setRetypePasswdMsgTxt("");
				}else{
					if(passwordMatches()){
```

```
						User user = new
User(display.getUsername(),display.getPassword(),display.getEmail(),displa
y.getLastName(),

		display.getFirstName(),display.getMiddleName(),
display.getAffiliation(), YesNo.valueOf(display.getSysAdPriv()),
YesNo.valueOf(display.getToolAdPriv()),

		display.getQuestion(),display.getAnswer());

		checkUsernameAndEmailIfUsed();

		rpcService.addUserWithChecking(user, false, new
AsyncCallback<String>() {

							public void
onSuccess(String result) {

		if(result.equals("ok")){

		History.newItem("#");

		History.newItem("userMngt?tab=0");

		}else{

		display.setErrorTxt(result,true);

		}
							}

							public void
onFailure(Throwable caught) {

		caught.printStackTrace();
							}
						});
					}else{

		display.setRetypePasswdMsgTxt("Password does not match");
					}
				}
		}

}
```

EditMembershipPresenter.java

```
package cepir.client.presenter;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class EditMembershipPresenter implements Presenter {
		public interface Display{
			void setUsernameTxt(String text);
			void setFirstnameTxt(String text);
			void setMidnameTxt(String text);
			void setLastnameTxt(String text);
			HasClickHandlers getSaveButton();
			HasClickHandlers getCancelButton();
			Integer getSelectedMemberType();
			void setMemberType(Integer roleID);
			Widget asWidget();
		}
```

```
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final UserServiceAsync userService =
GWT.create(UserService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final String username;
	private final Integer projID;
	private Role myRole;

	public EditMembershipPresenter(HandlerManager eventBus,
Display display, String username, Integer projID) {
		this.eventBus = eventBus;
		this.display = display;
		this.username = username;
		this.projID = projID;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		bind();
		setupMember();
	}

	private void bind() {
		display.getSaveButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				saveMembership();
			}
		});

		display.getCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				setupHistory();
			}
		});
	}

	private void setupHistory(){

	History.newItem("members?p="+projID+"&tab=0");
	}

	private void setupMember() {
		userService.getUser(username, new
AsyncCallback<User>() {

			public void onSuccess(User result) {

	display.setUsernameTxt(result.username);

	display.setFirstnameTxt(result.firstName);

	display.setMidnameTxt(result.middleName);

	display.setLastnameTxt(result.lastName);
				getUserPriv();
			}

			public void onFailure(Throwable
caught) {

			}
		});
	}

	private void getUserPriv(){
		projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

			public void onSuccess(Role result) {
				myRole = result;

	display.setMemberType(result.roleID);
			}

			public void onFailure(Throwable
caught) {

			}
		});
	}

	private void saveMembership(){
		projService.approveMemRequest(username,
projID, display.getSelectedMemberType()+1, true, new
AsyncCallback<Void>() {

			public void onSuccess(Void result) {
				setupHistory();
			}

			public void onFailure(Throwable
caught) {

			}
		});
	}

}
```

EditMyInfoPresenter.java

```
package cepir.client.presenter;

import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.server.BCrypt;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class EditMyInfoPresenter implements Presenter{
	public interface Display{
		Widget asWidget();
		void setFirstNameTxt(String text);
		void setMiddleNameTxt(String text);
		void setLastNameTxt(String text);
		void setEmailTxt(String text);
		void setAffiliationTxt(String text);
		void setQuestion(String text);
		void setAnswer(String text);
		void setUsernameLabel(String text);
		String getFirstnameTxt();
		String getMidnameTxt();
		String getLastnameTxt();
		String getEmailTxt();
		String getAffiliationTxt();
		void setErrorTxt(String text);
		void clearErrorTxt();
		void setEmailErrTxt(String text);
		void clearEmailErrTxt();
		void setPwdErrTxt(String text);
		void clearPwdErrTxt();
		void showPwdDialog();
		void hidePwdDialog();
		String getOldPwd();
		String getNewPwd();
		String getRetypePwd();
		String getQuestion();
		String getAnswer();
		void clearPwdDialog();
		HasClickHandlers getChangePwdButton();
		HasClickHandlers getSaveButton();
```

```
                HasClickHandlers getCancelButton();
                HasClickHandlers getOkPwdButton();
                HasClickHandlers getCancelPwdButton();
        }
        private final UserServiceAsync userService =
GWT.create(UserService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private User user;

        public EditMyInfoPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                checkWhoseAccount();
        }

        private void checkWhoseAccount() {
                userService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                loadDetails(result);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void loadDetails(String username){
                userService.getUser(username, new
AsyncCallback<User>() {

                        public void onSuccess(User result) {
                                user = result;

        display.setUsernameLabel(result.username);

        display.setFirstNameTxt(result.firstName);

        display.setMiddleNameTxt(result.middleName);

        display.setLastNameTxt(result.lastName);

        display.setEmailTxt(result.email);

        display.setAffiliationTxt(result.affiliation);

        display.setQuestion(result.question);

        display.setAnswer(result.answer);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void bind() {
                display.getCancelButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("memberIndex");
                        }
                });

                display.getSaveButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                saveUserDetails();
                        }
                });


        display.getChangePwdButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.showPwdDialog();
                        }
                });

                display.getOkPwdButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                checkOldPwd();
                        }
                });


        display.getCancelPwdButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.hidePwdDialog();

        display.clearPwdDialog();
                        }
                });
        }

        private void checkOldPwd(){

        userService.checkPasswordMatch(display.getOldPwd(),
user.password, new AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{
                                if(!result){

        display.setPwdErrTxt("Old password does not match");
                                }else{

        if(!display.getNewPwd().equals(display.getRetypePwd())){

        display.setPwdErrTxt("Re-typed and new password do not
match");
                                        }else{

        display.hidePwdDialog();

        savePassword();
                                        }
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void savePassword(){
                userService.resetPassword(user.username,
display.getNewPwd(), new AsyncCallback<Void>() {
```

```
                                        public void onSuccess(Void result) {

                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }

        public void saveUserDetails(){
                display.clearEmailErrTxt();
                display.clearErrorTxt();

        if(display.getFirstnameTxt().isEmpty()||display.getMidnameTxt
().isEmpty()||display.getLastnameTxt().isEmpty()

        ||display.getEmailTxt().isEmpty()||display.getQuestion().isEmpt
y()||display.getAnswer().isEmpty()){
                                        display.setErrorTxt("Required field/s
missing");
                }else{
                                        User updatedUser = new User();
                                        updatedUser.username =
user.username;
                                        updatedUser.sysAd = user.sysAd;
                                        updatedUser.toolAd = user.toolAd;
                                        updatedUser.firstName =
display.getFirstnameTxt();
                                        updatedUser.middleName =
display.getMidnameTxt();
                                        updatedUser.lastName =
display.getLastnameTxt();
                                        updatedUser.email =
display.getEmailTxt();
                                        updatedUser.affiliation =
display.getAffiliationTxt();
                                        updatedUser.question =
display.getQuestion();
                                        updatedUser.answer =
display.getAnswer();
                                        userService.updateUser(updatedUser,
false, new AsyncCallback<String>() {

                                                public void
onFailure(Throwable caught) {

        caught.printStackTrace();
                                                }

                                                public void
onSuccess(String result) {

        if(result.equals("updated")){

        History.newItem("memberIndex");
                                                        }else
if(result.equals("Email already in use")){

        display.setEmailErrTxt(result);
                                                        }else{

        display.setErrorTxt(result);
                                                        }
                                                }
                                        });
                }
        }

}
```

EditUserPresenter.java

```
package cepir.client.presenter;

import cepir.client.InputChecker;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.User;
import cepir.shared.YesNo;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class EditUserPresenter implements Presenter{
        public interface Display{
                Widget asWidget();
                void setEmailMsgTxt(String emailMsg);
                void setRetypePasswdMsgTxt(String msg);
                void setErrorTxt(String errMsg);
                HasClickHandlers getUpdateButton();
                HasClickHandlers getCancelButton();
                HasClickHandlers getChangeAction();
                String getUsername();
                void setUserName(String name);
                String getPassword();
                String getRetypedPassword();
                String getFirstName();
                void setFirstName(String name);
                String getMiddleName();
                void setMiddleName(String name);
                String getLastName();
                void setLastName(String name);
                String getEmail();
                void setEmail(String name);
                String getAffiliation();
                void setAffiliation(String name);
                String getSysAdPriv();
                void setSysAdPriv(int selectedIndex);
                String getToolAdPriv();
                void setQuestion(String text);
                String getQuestion();
                void setAnswer(String text);
                String getAnswer();
                void setToolAdPriv(int selectedIndex);
                Boolean willChangePasswd();
                void enablePassChange(Boolean willChange);

        }

        private final UserServiceAsync rpcService =
GWT.create(UserService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private String username;
        private String email;

        public EditUserPresenter(HandlerManager eventBus, Display
display, String username) {
                this.eventBus = eventBus;
                this.display = display;
                this.username = username;
        }

        public void go(HasWidgets container) {
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                loadUserInfo();
        }

        private void bind() {
                display.getCancelButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("userMngt?tab=0");
                        }
```

```
                });

                display.getUpdateButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                updateUser();
                        }
                });

                display.getChangeAction().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                editPassword();
                        }
                });
        }

        public void loadUserInfo(){
                rpcService.getUser(username, new
AsyncCallback<User>() {

                        public void onSuccess(User result) {

        display.setUserName(result.username);

        display.setFirstName(result.firstName);

        display.setMiddleName(result.middleName);

        display.setLastName(result.lastName);

        display.setEmail(result.email);
                                email = result.email;

        display.setAffiliation(result.affiliation);

        display.setSysAdPriv((result.sysAd.toString().equalsIgnoreCas
e("yes")?0:1));

        display.setToolAdPriv((result.toolAd.toString().equalsIgnoreCa
se("yes")?0:1));

        display.setQuestion(result.question);

        display.setAnswer(result.answer);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        // returns true if there are field errors
        private Boolean checkFields(){
                Boolean hasError = false;
                if(display.willChangePasswd()){

        if(display.getPassword().isEmpty()||display.getRetypedPasswor
d().isEmpty()){

        display.setErrorTxt("Required field/s missing");

        display.setRetypePasswdMsgTxt("");

        display.setEmailMsgTxt("");
                                hasError = true;
                        }

        if(!display.getPassword().equals(display.getRetypedPassword()
)){

        display.setRetypePasswdMsgTxt("Password does not match");

        display.setEmailMsgTxt("");
                                display.setErrorTxt("");
                                hasError = true;
                        }
                }else
if(display.getFirstName().isEmpty()||display.getMiddleName().isEmpty()||

        display.getLastName().isEmpty()||display.getEmail().isEmpty()|
|display.getQuestion().isEmpty()||display.getAnswer().isEmpty()){
                        display.setErrorTxt("Required field/s
missing");
                        display.setRetypePasswdMsgTxt("");
                        display.setEmailMsgTxt("");
                        hasError = true;
                }
                return hasError;
        }

        private void updateUser() {
                InputChecker input = new InputChecker();
                if(!input.isValidEmail(display.getEmail())){
                        display.setEmailMsgTxt("Invalid
email address");
                        display.setRetypePasswdMsgTxt("");
                        display.setErrorTxt("");
                }else if(!checkFields()){
                        User user = new User();
                        user.username = username;
                        user.password =
display.getPassword();
                        user.firstName =
display.getFirstName();
                        user.middleName =
display.getMiddleName();
                        user.lastName =
display.getLastName();
                        user.email = display.getEmail();
                        user.affiliation =
display.getAffiliation();
                        user.sysAd =
YesNo.valueOf(display.getSysAdPriv());
                        user.toolAd =
YesNo.valueOf(display.getToolAdPriv());
                        user.question = display.getQuestion();
                        user.answer = display.getAnswer();
                        rpcService.updateUser(user,
(display.willChangePasswd()?true:false), new AsyncCallback<String>() {

                                public void
onSuccess(String result) {

        if(result.equals("updated")){

        History.newItem("userMngt?tab=0");
                                }else
if(result.equals("Email already in use")){

        display.setEmailMsgTxt(result);
                                }else{

        display.setErrorTxt(result);
                                }
                        }


                                public void
onFailure(Throwable caught) {

        caught.printStackTrace();
                                }
                        });
                }
        }

        private void editPassword(){
                if(display.willChangePasswd()){
                        display.enablePassChange(true);
                }else{
                        display.enablePassChange(false);
                }
        }
```

```
}
```

ForgotPasswordPresenter.java

```
package cepir.client.presenter;

import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ForgotPasswordPresenter implements Presenter {
	public interface Display{
		Widget asWidget();
		String getEmailBox();
		String getAnswerBox();
		String getPasswd();
		String getRetypedPwd();
		HasClickHandlers getEmailSubmitButton();
		HasClickHandlers getAnswerSubmitButton();
		HasClickHandlers getSaveButton();
		HasClickHandlers getCancelButton();
		HasClickHandlers getOkButton();
		void setEmailError(String text);
		void setAnswerError(String text);
		void setSaveError(String text);
		void setQuestion(String text);
		void setSelectedDeck(int index);
		void showDialog();
		void hideDialog();
	}
	private final UserServiceAsync userService =
GWT.create(UserService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private User user;

	public ForgotPasswordPresenter(HandlerManager eventBus,
Display display) {
		this.eventBus = eventBus;
		this.display = display;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		bind();
	}

	private void bind() {

	display.getEmailSubmitButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {

	if(display.getEmailBox().isEmpty()){

	display.setEmailError("Username is missing");
					}else{

	getQuestion();
					}
				}
		});

	display.getAnswerSubmitButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {

	if(display.getAnswerBox().isEmpty()){

	display.setAnswerError("Answer is missing");
					}else{

	checkAnswer();
					}
				}
		});

		display.getSaveButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {

	if(display.getPasswd().isEmpty()||display.getRetypedPwd().isE
mpty()){

	display.setSaveError("Password is empty");
					}else
if(display.getPasswd().equals(display.getRetypedPwd())){

	savePassword();
					}else{

	display.setSaveError("Re-typed password does not match");
					}
				}
		});

		display.getOkButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					display.hideDialog();

	History.newItem("index");
				}
		});

		display.getCancelButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {

	History.newItem("index");
				}
		});
	}

	private void checkAnswer() {

	if(user.answer.equalsIgnoreCase(display.getAnswerBox())){
				display.setSelectedDeck(2);
		}else{
				display.setAnswerError("Invalid
answer");
		}
	}

	private void getQuestion(){

	userService.getSecurityQuestion(display.getEmailBox(), new
AsyncCallback<User>() {

				public void onSuccess(User result) {

	if(result.username!=null){
					user =
result;
```

```
                display.setQuestion(result.question);

                display.setSelectedDeck(1);
                                                }else{

                display.setEmailError("Username is not registered in the
system");
                                                }
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                }

                private void savePassword(){
                        userService.resetPassword(user.username,
display.getPasswd(), new AsyncCallback<Void>() {

                                        public void onSuccess(Void result) {
                                                display.showDialog();
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                }

}
```

HomeForMemberPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class HomeForMemberPresenter implements Presenter {
        public interface Display{
                void setPendingProjCreationReq(String count);
                void setPendingProjMemReq(String count);
                void setNeedsApproval(String count);
                void setWelcomeMsg(String user);
                void setVisibilityApprovalPanel(Boolean
isVisible);
                Widget asWidget();
        }
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private String user;

        public HomeForMemberPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        private void getUsername(){
                projService.getSessionKeyData("username", new
AsyncCallback<String>() {

                                        public void onSuccess(String result) {
                                                user = result;
                display.setWelcomeMsg(result);

                setPendingProjCreationReqsNo();
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                }

                public void go(HasWidgets container) {
                        if(container!=null){
                                container.clear();
                                container.add(display.asWidget());
                        }
                        getUsername();
                }

                private void setPendingProjCreationReqsNo(){
                        projService.getProjRequests(user, new
AsyncCallback<ArrayList<NewProjRequest>>() {

                                        public void
onSuccess(ArrayList<NewProjRequest> result) {

                display.setPendingProjCreationReq(Integer.toString(result.size(
)));

                setPendingApprovalNo();
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                }

                private void setPendingApprovalNo(){
                        projService.getMemRequestsWhereAdmin(user,
new AsyncCallback<ArrayList<ProjMemRequest>>() {

                                        public void
onSuccess(ArrayList<ProjMemRequest> result) {
                                                if(result.size()==0){

                display.setVisibilityApprovalPanel(false);
                                                }else{

                display.setVisibilityApprovalPanel(true);
                                                }

                display.setNeedsApproval(Integer.toString(result.size()));

                setPendingProjMemReqsNo();
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                }

                private void setPendingProjMemReqsNo(){
                        projService.getProjMemRequests(user, new
AsyncCallback<ArrayList<ProjMemRequest>>() {

                                        public void
onSuccess(ArrayList<ProjMemRequest> result) {

                display.setPendingProjMemReq(Integer.toString(result.size()));
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
```

```
                }
            });
        }

}
```

HomeForSysAdPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.presenter.PendingUsersListTabPresenter.Display;
import cepir.shared.AcctRequest;
import cepir.shared.NewProjRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class HomeForSysAdPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
                void setNoOfPendingUsers(String pendingUsers);
                void setNoOfPendingProjs(String pendingProjs);
                void setWelcomeMsg(String username);
        }

        private final UserServiceAsync userService =
GWT.create(UserService.class);
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;

        public HomeForSysAdPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                getUsername();
                getPendingUsers();
                getPendingProjs();
        }

        private void getPendingProjs() {
                projService.getProjRequests(new
AsyncCallback<ArrayList<NewProjRequest>>() {

                        public void
onSuccess(ArrayList<NewProjRequest> result) {

        display.setNoOfPendingProjs(Integer.toString(result.size()));
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });

        }

        private void getPendingUsers() {
                userService.getAcctRequests(new
AsyncCallback<ArrayList<AcctRequest>>() {

                        public void
onSuccess(ArrayList<AcctRequest> result) {

        display.setNoOfPendingUsers(Integer.toString(result.size()));
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });

        }

        private void getUsername() {
                userService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {

        display.setWelcomeMsg(result);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

}
```

IntroMsgPresenter.java

```
package cepir.client.presenter;

import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class IntroMsgPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
        }
        public final Display display;

        public IntroMsgPresenter(Display display){
                this.display = display;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
        }

}
```

LoginPagePresenter.java

```
package cepir.client.presenter;

import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.event.UserLoggedInEvent;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class LoginPagePresenter implements Presenter{

        public interface Display{
                HasClickHandlers getSignInButton();
                Widget asWidget();
                String getUsername();
                String getPassword();
                void setErrorTxt(String errMsg);
```

```
	}

	private final UserServiceAsync rpcService =
GWT.create(UserService.class);
	private final HandlerManager eventBus;
	private final Display display;

	public LoginPagePresenter(HandlerManager eventBus, Display
view) {
		this.eventBus = eventBus;
		this.display = view;
	}

	public void go(HasWidgets container) {
		bind();
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
	}

	public void bind() {
		display.getSignInButton().addClickHandler(new
ClickHandler() {
			public void onClick(ClickEvent
event) {
				checkIfUser();
			}
		});
	}

	public void checkIfUser(){
		if(display.getUsername().isEmpty() ||
display.getPassword().isEmpty()){
			Window.alert("Username or password
is missing");
		}else{

	rpcService.checkIfUser(display.getUsername(),
display.getPassword(), new AsyncCallback<Boolean>() {

				public void
onSuccess(Boolean result) {
					if(result !=
false){

	eventBus.fireEvent(new UserLoggedInEvent());
					}else{

	display.setErrorTxt("Login failed. Incorrect
username/password");
					}

				}

				public void
onFailure(Throwable caught) {

	display.setErrorTxt("Error "+caught.getMessage());
				}
			});
		}
	}
}
```

LogoutPresenter.java

```
package cepir.client.presenter;

import cepir.client.UserService;
import cepir.client.UserServiceAsync;

import com.google.gwt.core.client.GWT;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;

public class LogoutPresenter implements Presenter{
	private final UserServiceAsync rpcService =
GWT.create(UserService.class);

	public LogoutPresenter(){
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
		}
		rpcService.closeSession(new
AsyncCallback<Void>() {

			public void onSuccess(Void result) {
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

}
```

MainPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.view.LoginPageView;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Widget;

public class MainPresenter implements Presenter {
	public interface Display {
		void setPage(Widget widget);
		void setSideMenu(Widget widget);
		void setCenterHeader(ArrayList<Project>
projects);
		HasClickHandlers getGoButton();
		HasClickHandlers getOptionsButton();
		Integer getSelectedItem();
		void setSelectedItem(Integer index);
		void hideHeader();
		void clearList();
		Widget asWidget();
	}

	private final UserServiceAsync rpcService =
GWT.create(UserService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final Presenter newPagePresenter;
	private final Widget newPageView;
	private ArrayList<Project> projects;

	public MainPresenter(HandlerManager eventBus, Display
view, Presenter newPagePresenter, Widget newPageView){
		this.eventBus = eventBus;
		this.display = view;
		this.newPagePresenter = newPagePresenter;
		this.newPageView = newPageView;
	}
	public void go(HasWidgets container) {
		display.clearList();
		setupCenterHeader();
		if(container!=null){
```

```
                container.clear();
                container.add(display.asWidget());
            }
            setSideBar();
            setupNewPage();
        }

        private void bind() {
            display.getGoButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

        if(display.getSelectedItem()!=-1){

        History.newItem("projHome="+projects.get(display.getSelecte
dItem()).projID);
                    }
                }
            });

            display.getOptionsButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

        History.newItem("manageAcct");
                }
            });
        }

        private void setupNewPage(){
            display.setPage(newPageView);
            newPagePresenter.go(null);
        }

        private void setSideBar() {
            rpcService.getSessionKeyData("isSysAd", new
AsyncCallback<String>() {

                public void onSuccess(String result) {
                    if(result==null){ //guest

        if(History.getToken().equals("requestForAcct")||History.getTok
en().equals("forgotPass")){

        FlexTable menu = new FlexTable();


        Hyperlink home = new Hyperlink("Home", "index");

        menu.setWidget(0, 0, home);

        display.setSideMenu(menu);

                                                    }else{

        LoginPageView loginPage = new LoginPageView();

        LoginPagePresenter loginPresenter = new
LoginPagePresenter(eventBus, loginPage);

        display.setSideMenu(loginPage);

        loginPresenter.go(null);
                                                    }
                                                }else{ //member
                                                    FlexTable
menu = new FlexTable();
                                                    Hyperlink
home = new Hyperlink("Home", "memberIndex");

        menu.setWidget(0, 0, home);
                                                    Hyperlink
projects = new Hyperlink("Projects","myProjects");

        menu.setWidget(1, 0, projects);

        if(result.equalsIgnoreCase("yes")){

        Hyperlink userMngt = new Hyperlink("User Management",
"userMngt?tab=0");

        menu.setWidget(2,0,userMngt);

        Hyperlink projMngt = new Hyperlink("Project Management",
"projMngt?tab=0");

        menu.setWidget(3,0,projMngt);
                                                    }else{

        Hyperlink requestForProj = new Hyperlink("Request for
Project Creation", "requestForProj");

        menu.setWidget(2, 0, requestForProj);
                                                    }
                                                    Hyperlink
tools = new Hyperlink("Tools", "tools/");

        menu.setWidget((result.equalsIgnoreCase("yes")?4:3), 0, tools);
                                                    Hyperlink
logout = new Hyperlink("Logout", "index");

        logout.addClickHandler(new ClickHandler() {


        public void onClick(ClickEvent event) {

        LogoutPresenter logoutPresenter = new LogoutPresenter();

        logoutPresenter.go(null);

        History.newItem("index");

        }
                                                    });

        menu.setWidget((result.equalsIgnoreCase("yes")?5:4),0,logout)
;

        display.setSideMenu(menu);
                                                }
                                            }

                                            public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                            }
                                        });
        }

        private void setupCenterHeader(){
                rpcService.getSessionKeyData("username", new
AsyncCallback<String>() {

                                            public void onSuccess(String result) {
                                                if(result!=null){

        fetchProjects(result);
                                                }else{

        display.hideHeader();
                                                    projects =
new ArrayList<Project>();
                                                }
                                            }

                                            public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                            }
                                        });
        }

        private void fetchProjects(String username){
                ProjectServiceAsync projService =
GWT.create(ProjectService.class);
```

```
                projService.getProjects(username, new
AsyncCallback<ArrayList<Project>>() {

                        public void
onSuccess(ArrayList<Project> result) {
                                projects = result;

        display.setCenterHeader(result);
                                bind();
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }
}
```

MemberMngtPresenter.java

```
package cepir.client.presenter;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.presenter.UserMngtPagePresenter.Display;
import cepir.client.view.MemListTabView;
import cepir.client.view.PendingMembersView;
import cepir.shared.Role;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class MemberMngtPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
                void addTab(Widget widget,String tabName);
                HasSelectionHandlers<Integer> getTabPanel();
                void setSelectedTab(int index);
                void setVisibilityTabs(Boolean isVisible);
        }
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private Integer selectedIndex;
        private final Integer projID;

        public MemberMngtPresenter(HandlerManager eventBus,
Display display, Integer projID, Integer selectedIndex) {
                this.eventBus = eventBus;
                this.display = display;
                this.selectedIndex = selectedIndex;
                this.projID = projID;
        }

        public void go(HasWidgets container) {
                display.setVisibilityTabs(false);
                getUsername();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                bind();
        }

        private void getUsername(){
                projService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                checkRole(result);
                        }

                        public void onFailure(Throwable
caught) {

                        }
                });
        }

        private void checkRole(String username){
                projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

                        public void onSuccess(Role result) {

        if(result.roleID==null||result.roleName.isEmpty()){

        History.newItem("unauthorized");
                                }else{

        setupTabs(result);
                                }
                        }

                        public void onFailure(Throwable
caught) {

                        }
                });
        }

        private void setupTabs(Role role) {
                MemListTabView memlist = new
MemListTabView();
                display.addTab(memlist, "Members");
                MemListTabPresenter memPresenter = new
MemListTabPresenter(eventBus, memlist, projID);
                memPresenter.go(null);

                if(role.roleName.equalsIgnoreCase("project
administrator")){
                        PendingMembersView pending = new
PendingMembersView();
                        display.addTab(pending, "Project
Membership Requests");
                        PendingMembersPresenter
pendingPresenter = new PendingMembersPresenter(eventBus, pending,
projID);
                        pendingPresenter.go(null);
                }else{
                        selectedIndex = 0;
                }
                display.setSelectedTab(selectedIndex);
                display.setVisibilityTabs(true);
        }

        private void bind() {
                display.getTabPanel().addSelectionHandler(new
SelectionHandler<Integer>() {

                        public void
onSelection(SelectionEvent<Integer> event) {

        display.setVisibilityTabs(false);

        History.newItem("members?p="+projID+"&tab="+event.getSe
lectedItem());
                        }
                });
        }

}
```

MemListTabPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;
import java.util.Collections;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
```

```
import cepir.client.UserServiceAsync;
import cepir.client.event.AddUserEvent;
import cepir.client.presenter.UsersListTabPresenter.Display;
import cepir.client.view.MainView;
import cepir.client.view.UnauthorizedNoticeView;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class MemListTabPresenter implements Presenter {
	public interface Display{
		Widget asWidget();
		void loadUsersInDiagList(ArrayList<User> users);
		void loadProjMembersList(ArrayList<User>
users);
		void loadUsersList(ArrayList<User> users,
Boolean isProjAdmin);
		ArrayList<Integer> getSelectedAdmins();
		ArrayList<Integer> getSelectedUsers();
		void showAdd(String title);
		void hideAdd();
		void editUser_Warning(int left, int top);
		void delUser_Warning(int left, int top);
		void addMember_Warning(int left, int top);
		void usersPerPage_Warning(int left, int top);
		void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
		Integer getSelectedRow();
		String getSearchText();
		void clearSearchText();
		String getSearchInAddText();
		void clearSearchInAddText();
		void clearRowStyles();
		Integer getUsersPerPageText();
		void clearUsersPerPageText();
		FlexTable getUsersListTable();
		void setVisibilityButtons(Boolean isVisible);
		HasClickHandlers getUsersPerPageButton();
		HasClickHandlers getDeleteButton();
		HasClickHandlers getAddMemButton();
		HasClickHandlers getAddAdminButton();
		HasClickHandlers getEditButton();
		HasClickHandlers getRefreshButton();
		HasClickHandlers getSearchButton();
		HasClickHandlers getcancelSearchButton();
		HasClickHandlers getSearchInAddButton();
		HasClickHandlers getcancelSearchInAddButton();
		HasClickHandlers getGoToFirstPageButton();
		HasClickHandlers getGoToLastPageButton();
		HasClickHandlers getPrevButton();
		HasClickHandlers getNextButton();
		HasClickHandlers getCantDeleteOkButton();
		HasClickHandlers getSelectOkButton();
		HasClickHandlers getSelectCancelButton();
		HasClickHandlers getSelectMemberButton();
		HasClickHandlers getDeselectMemberButton();
		HasKeyDownHandlers getSearchKeyHandler();
		HasKeyDownHandlers
getSearchKeyHandlerInAdd();
		void showDeleteUserWarning();
		void hideDeleteUserWarning();
	}
	private final UserServiceAsync userService =
GWT.create(UserService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private Integer projID;
	private String currUser;
	private ArrayList<User> users;
	private ArrayList<User> unselected = new ArrayList<User>();
	private ArrayList<User> selected = new ArrayList<User>();
	private Boolean isNewMemberAnAdmin;
	private Boolean willDisplayDialog;
	private Boolean isProjAdmin;
	private int itemsPerPage = 5;
	private int totalPages;
	private int currPage = 1;

	public MemListTabPresenter(HandlerManager eventBus,
Display display, Integer projID) {
		this.eventBus = eventBus;
		this.display = display;
		this.projID = projID;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		getUsername();

	}

	private void getUsername(){
		projService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				currUser = result;
				checkPriv();
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void checkPriv(){
		projService.getRoleInProj(currUser, projID, new
AsyncCallback<Role>() {

			public void onSuccess(Role result) {

	if(result.roleID==null||result.roleName.isEmpty()){

	History.newItem("unauthorized");
					}else{

	if(result.roleName.equalsIgnoreCase("project member")){

	display.setVisibilityButtons(false);

	isProjAdmin = false;
							}else{

	display.setVisibilityButtons(true);

	isProjAdmin = true;
							}

	performSearch();
							bind();
					}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}
```

```
	private void bind() {
		display.getUsersListTable().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Cell cell =
display.getUsersListTable().getCellForEvent(event);
				Integer selected = null;

	if(!users.isEmpty()&&isProjAdmin){
					selected =
display.getSelectedRow();
				}

	if(cell!=null&&selected!=null){

	display.clearRowStyles();

	display.getUsersListTable().getRowFormatter().setStyleName(s
elected,"tableHighlight");
				}
			}
		});


	display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

			public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	performSearch();
				}
			}
		});


	display.getSearchKeyHandlerInAdd().addKeyDownHandler(ne
w KeyDownHandler() {

			public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	search(display.getSearchInAddText());
				}
			}
		});

		display.getDeleteButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	if(display.getSelectedRow()==null){
					Widget
source = (Widget) event.getSource();
			int left = source.getAbsoluteLeft() + 10;
			int top = source.getAbsoluteTop() + 10;
			display.delUser_Warning(left, top);
				}else{

	deleteMember();
				}
			}
		});


	display.getAddMemButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				selected.clear();
				isNewMemberAnAdmin
= false;
				willDisplayDialog = true;
				loadLists();
			}
		});


	display.getAddAdminButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				selected.clear();
				isNewMemberAnAdmin
= true;
				willDisplayDialog = true;
				loadLists();
			}
		});


	display.getSelectMemberButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				for(int row = 0; row <
display.getSelectedUsers().size(); row++){

	selected.add(unselected.get(display.getSelectedUsers().get(row)
-row));

	unselected.remove(unselected.get(display.getSelectedUsers().ge
t(row)-row));
				}

	Collections.sort(selected);

	Collections.sort(unselected);

	display.loadProjMembersList(selected);

	display.loadUsersInDiagList(unselected);
			}
		});


	display.getDeselectMemberButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				for(int row = 0; row <
display.getSelectedAdmins().size(); row++){

	unselected.add(selected.get(display.getSelectedAdmins().get(ro
w)-row));

	selected.remove(selected.get(display.getSelectedAdmins().get(r
ow)-row));
				}

	Collections.sort(selected);

	Collections.sort(unselected);

	display.loadProjMembersList(selected);

	display.loadUsersInDiagList(unselected);
			}
		});


	display.getSelectCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				display.hideAdd();
```

```
                }
            });

            display.getSelectOkButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    if(selected.isEmpty()){
                        Widget
source = (Widget) event.getSource();
                int left = source.getAbsoluteLeft() + 10;
                int top = source.getAbsoluteTop() + 10;

        display.addMember_Warning(left, top);
                    }else{

        saveMembers();
                    }
                }
            });


        display.getSearchInAddButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

        search(display.getSearchInAddText());
                }
            });


        display.getcancelSearchInAddButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

        display.clearSearchInAddText();
                    willDisplayDialog =
false;
                    loadLists();
                    if(!selected.isEmpty()){

        unselected.removeAll(selected);
                    }

        display.loadProjMembersList(selected);

        display.loadUsersInDiagList(unselected);
                }
            });

            display.getRefreshButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    History.newItem("#");

        History.newItem("members?p="+projID.toString()+"&tab=0");
                }
            });

            display.getEditButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    editUser(event);
                }

            });

            display.getSearchButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    performSearch();
                }
            });


        display.getcancelSearchButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

        display.clearSearchText();
                    performSearch();
                }
            });


        display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    currPage = 1;
                    paginate(1);
                }
            });


        display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    currPage = totalPages;
                    paginate(totalPages);
                }
            });

            display.getNextButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    currPage++;
                    paginate(currPage);
                }
            });

            display.getPrevButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    currPage--;
                    paginate(currPage);
                }
            });


        display.getUsersPerPageButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                    Integer items =
display.getUsersPerPageText();

        if(items!=null&&items>0){

        itemsPerPage = items;

        performSearch();
                    }else{
                        Widget
source = (Widget) event.getSource();
                int left = source.getAbsoluteLeft() + 10;
                int top = source.getAbsoluteTop() + 10;
```

```
		display.usersPerPage_Warning(left, top);
					}
				}
			});


		display.getCantDeleteOkButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {

		display.hideDeleteUserWarning();
					}
			});
		}

		private void saveMembers(){
			ArrayList<String> newAdmins = new
ArrayList<String>();
			for(int i = 0; i < selected.size(); i++){

		newAdmins.add(selected.get(i).username);
			}
			projService.addProjMembers(projID, newAdmins,
false, (isNewMemberAnAdmin?"Project Administrator":"Project Member"),
new AsyncCallback<String>() {

					public void onSuccess(String result) {
						if(result.equals("ok")){

		display.hideAdd();

		display.clearSearchText();

		performSearch();
					}
				}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
				}
			});
		}

		private void loadLists(){
			userService.getUsersList(new
AsyncCallback<ArrayList<User>>() {

					public void
onSuccess(ArrayList<User> result) {
						users = result;
						getNonMembers();
				}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
				}
			});
		}

		private void getNonMembers(){
			projService.getProjMembers(projID, new
AsyncCallback<ArrayList<User>>() {

					public void
onSuccess(ArrayList<User> result) {
						unselected.clear();
						unselected.addAll(users);
						if(!result.isEmpty()){

		unselected.removeAll(result);
					}
					if(willDisplayDialog){

		loadDialogBox();
					}
```

```
				}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
				}
			});
		}

		private void loadDialogBox(){
			display.loadUsersInDiagList(unselected);
			display.loadProjMembersList(selected);
			if(isNewMemberAnAdmin){
				display.showAdd("Project
Administrator/s");
			}else{
				display.showAdd("Project
Member/s");
			}
		}

		private void search(String query){
			willDisplayDialog = false;
			loadLists();
			unselected.removeAll(selected);
			for(int i = 0; i < unselected.size(); i++){
				User user = unselected.get(i);

		if(user.username.toLowerCase().contains(query.toLowerCase())
||user.firstName.toLowerCase().contains(query.toLowerCase())

		||user.lastName.toLowerCase().contains(query.toLowerCase()))
{
					}else{
						i--;
						unselected.remove(user);
					}
				}
				display.loadUsersInDiagList(unselected);
		}

		private void deleteMember(){
			if(!users.get((itemsPerPage*(currPage-
1))+(display.getSelectedRow()-2)).getUsername().equals(currUser)){

		projService.deleteUserPriv(users.get((itemsPerPage*(currPage-
1))+(display.getSelectedRow()-2)).getUsername(), projID, new
AsyncCallback<Void>() {

					public void
onSuccess(Void result) {

		display.clearSearchText();

		performSearch();
					}

					public void
onFailure(Throwable caught) {

		caught.printStackTrace();
					}
				});
			}else{
				display.showDeleteUserWarning();
			}
		}

		private void paginate(int page) {
			totalPages =
(int)Math.ceil((double)users.size()/itemsPerPage);
			ArrayList<User> concat = new
ArrayList<User>();
			for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(users.size()-
1)?users.size():itemsPerPage*page); ctr++){
				concat.add(users.get(ctr));
			}
			display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
```

```
                display.loadUsersList(concat,isProjAdmin);
        }

        private void editUser(ClickEvent event) {
                if(display.getSelectedRow()==null){
                        Widget source = (Widget)
event.getSource();
    int left = source.getAbsoluteLeft() + 10;
    int top = source.getAbsoluteTop() + 10;
                        display.editUser_Warning(left, top);
                }else{

        History.newItem("editMem?p="+projID.toString()+"&uname=
"+users.get((itemsPerPage*(currPage-1))+(display.getSelectedRow()-
2)).getUsername());
                }
        }

        private void performSearch() {
                projService.getProjMembersSearchList(projID,
display.getSearchText(), new AsyncCallback<ArrayList<User>>() {

                        public void
onSuccess(ArrayList<User> result) {
                                users = result;
                                currPage = 1;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }


}
```

MemRequestsToApprovePresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.MailService;
import cepir.client.MailServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.presenter.UsersListTabPresenter.Display;
import cepir.client.view.ViewProjMemRequestView;
import cepir.shared.ProjMemRequest;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class MemRequestsToApprovePresenter implements Presenter {
        public interface Display{
                void
loadRequestsList(ArrayList<ProjMemRequest> requests);
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                Integer getSelectedRow();
                String getSearchText();
                void clearSearchText();
                void clearRowStyles();
                Integer getMemReqsPerPageText();
                void clearMemReqsPerPageText();
                void memReqsPerPage_Warning(int left, int top);
                void memReqs_Warning(int left, int top);
                HasClickHandlers getSearchButton();
                HasClickHandlers getcancelSearchButton();
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasClickHandlers getAcceptButton();
                HasClickHandlers getRejectButton();
                HasClickHandlers getRefreshButton();
                HasClickHandlers getMemReqsPerPageButton();
                HasKeyDownHandlers getSearchKeyHandler();
                FlexTable getMemReqsTable();
                Widget asWidget();
        }

        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private ArrayList<ProjMemRequest> requests;
        private String user;
        private int itemsPerPage = 5;
        private int totalPages;
        private int currPage = 1;

        public MemRequestsToApprovePresenter(HandlerManager
eventBus, Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                bind();
                getUsername();
        }

        private void getUsername() {
                projService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                user = result;
                                performSearch();
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void reload(){
                History.newItem("#");
                History.newItem("allPendingReq");
        }

        private void bind() {
                display.getMemReqsTable().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Cell cell =
display.getMemReqsTable().getCellForEvent(event);
                                Integer selected = null;
                                if(!requests.isEmpty()){
                                        selected =
display.getSelectedRow();
                                }

        if(cell!=null&&selected!=null){
```

```
		display.clearRowStyles();

		display.getMemReqsTable().getRowFormatter().setStyleName(
selected,"tableHighlight");
					}
				}
			});


		display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

				public void
onKeyDown(KeyDownEvent event) {

		if(event.getNativeKeyCode()==13){

		performSearch();
					}
				}
			});

			display.getRefreshButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					reload();
				}
			});

			display.getAcceptButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					approveReq();
				}
			});

			display.getRejectButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					rejectReq();
				}
			});

			display.getSearchButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					performSearch();
				}
			});


		display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					currPage = 1;
					paginate(1);
				}
			});


		display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					currPage = totalPages;
					paginate(totalPages);
				}
			});

			display.getNextButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					currPage++;
					paginate(currPage);
				}
			});

			display.getPrevButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					currPage--;
					paginate(currPage);
				}
			});


		display.getMemReqsPerPageButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					Integer items =
display.getMemReqsPerPageText();

		if(items!=null&&items>0){

		itemsPerPage = items;

		performSearch();
					}else{
						Widget
source = (Widget) event.getSource();
				int left = source.getAbsoluteLeft() + 10;
				int top = source.getAbsoluteTop() + 10;

		display.memReqsPerPage_Warning(left, top);
					}
				}
			});


		display.getcancelSearchButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {

		display.clearSearchText();
					performSearch();
				}
			});
		}

		private void performSearch(){

		projService.getMemRequestsWhereAdminSearchList(user,
display.getSearchText(), new
AsyncCallback<ArrayList<ProjMemRequest>>() {

				public void
onSuccess(ArrayList<ProjMemRequest> result) {
					requests = result;
					currPage = 1;
					paginate(1);
				}

				public void onFailure(Throwable
caught) {

				}
			});
		}

		private void paginate(int page){
```

```
                totalPages =
(int)Math.ceil((double)requests.size()/itemsPerPage);
                ArrayList<ProjMemRequest> concat = new
ArrayList<ProjMemRequest>();
                for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(requests.size()-
1)?requests.size():itemsPerPage*page); ctr++){
                        concat.add(requests.get(ctr));
                }
                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                display.loadRequestsList(concat);

        }

        private void approveReq(){

        projService.approveMemRequest(requests.get((display.getSelec
tedRow()-2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.username.username,

        requests.get((display.getSelectedRow()-2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.projID.projID, 1, false, new
AsyncCallback<Void>() {

                        public void onSuccess(Void result) {
                                sendApprovedMail();
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void rejectReq(){

        projService.deleteMemRequest(requests.get((display.getSelecte
dRow()-2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.username.username,

        requests.get((display.getSelectedRow()-2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.projID.projID, new
AsyncCallback<Void>() {

                        public void onSuccess(Void result) {
                                sendRejectedMail();

                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void sendApprovedMail(){
                MailServiceAsync mail =
GWT.create(MailService.class);
                String APPROVED_SUBJ = "Project Membership
Request Approved - Collaboratory for Epidemiological Research (CEpiR)";
                String APPROVED_MSG = "Your request for
membership in project "+requests.get((display.getSelectedRow()-
2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.projID.projName.toUpperCase()
                        +" at the Collaboratory for Epidemiological
Research (CEpiR) has been approved. \n\n" + "NOTE: This message has
been generated automatically. Please do not reply to this message.";
                mail.postMail(new
String[]{requests.get((display.getSelectedRow()-2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.username.email},
APPROVED_SUBJ, APPROVED_MSG, new AsyncCallback<Boolean>()
{

                        public void onSuccess(Boolean result)
{
                                performSearch();
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void sendRejectedMail(){
                MailServiceAsync mail =
GWT.create(MailService.class);
                String UNAPPROVED_SUBJ = "Project
Membership Request Unapproved - Collaboratory for Epidemiological
Research (CEpiR)";
                String UNAPPROVED_MSG = "Your request for
membership in project "+requests.get((display.getSelectedRow()-
2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.projID.projName.toUpperCase().to
UpperCase()
                        +" at the Collaboratory for Epidemiological
Research (CEpiR) has been rejected. \n\n" + "NOTE: This message has been
generated automatically. Please do not reply to this message.";
                mail.postMail(new
String[]{requests.get((display.getSelectedRow()-2)+((currPage-
1)*itemsPerPage)).projMemRequestPK.username.email},
UNAPPROVED_SUBJ, UNAPPROVED_MSG, new
AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{
                                performSearch();
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

}
```

MyPendingRequestsPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class MyPendingRequestsPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
                void loadPendingList(ArrayList<Project> projects,
Boolean isHyperlink);
                void clearBoxes();
                ArrayList<Integer> getSelectedRows();
                void setHeader(String header);
                HasClickHandlers getCancelButton();
                HasClickHandlers getClearButton();
        }

        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final Integer selectedIndex;
        private ArrayList<NewProjRequest> creationRequests;
```

```
	private ArrayList<ProjMemRequest> memRequests;
	private String user;

	public MyPendingRequestsPresenter(HandlerManager
eventBus, Display display, Integer selectedIndex) {
		this.eventBus = eventBus;
		this.display = display;
		this.selectedIndex = selectedIndex;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		bind();
		getUsername();
	}

	private void bind(){
		display.getCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				if(selectedIndex == 0){

	deleteCreationReqs();

	fetchProjCreationReqs();
				}else{

	deleteMemReqs();

	fetchProjMemReqs();
				}
			}
		});

		display.getClearButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				display.clearBoxes();
			}
		});
	}

	private void getUsername(){
		projService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				user = result;
				if(selectedIndex==0){

	display.setHeader("My Pending Project Creation Request/s");

	fetchProjCreationReqs();
				}else{

	display.setHeader("My Pending Project Membership
Request/s");

	fetchProjMemReqs();
				}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void fetchProjCreationReqs() {
		projService.getProjRequests(user, new
AsyncCallback<ArrayList<NewProjRequest>>() {

			public void
onSuccess(ArrayList<NewProjRequest> result) {
				creationRequests =
result;
				ArrayList<Project>
projects = new ArrayList<Project>();
				for(int i = 0; i <
result.size(); i++){
					Project proj
= new Project();

	proj.projName = result.get(i).newProjName;

	proj.projDesc = result.get(i).newProjDesc;

	projects.add(proj);
				}

	display.loadPendingList(projects,false);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void fetchProjMemReqs() {
		projService.getProjMemRequests(user, new
AsyncCallback<ArrayList<ProjMemRequest>>() {

			public void
onSuccess(ArrayList<ProjMemRequest> result) {
				memRequests = result;
				ArrayList<Project>
projects = new ArrayList<Project>();
				for(int i = 0; i <
result.size(); i++){
					Project proj
= new Project();
					proj.projID
= result.get(i).projMemRequestPK.projID.projID;

	proj.projName =
result.get(i).projMemRequestPK.projID.projName;

	proj.projDesc =
result.get(i).projMemRequestPK.projID.projDesc;

	projects.add(proj);
				}

	display.loadPendingList(projects,true);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void deleteCreationReqs(){
		ArrayList<Integer> selectedRows =
display.getSelectedRows();
		for(int i = 0; i < selectedRows.size(); i++){

	projService.deleteProjCreationRequest(creationRequests.get(sel
ectedRows.get(i)-1).newProjRequestID, new AsyncCallback<Void>() {

				public void
onSuccess(Void result) {

				}

				public void
onFailure(Throwable caught) {

	caught.printStackTrace();
```

```
                                        }
                                });
                        }
                }

                private void deleteMemReqs(){
                        ArrayList<Integer> selectedRows =
display.getSelectedRows();
                        for(int i = 0; i < selectedRows.size(); i++){

                projService.deleteMemRequest(memRequests.get(selectedRow
s.get(i)-1).getProjMemRequestPK().getUsername().getUsername(),

                memRequests.get(selectedRows.get(i)-
1).getProjMemRequestPK().getProjID().getProjID(), new
AsyncCallback<Void>() {

                                        public void
onSuccess(Void result) {

                                        }

                                        public void
onFailure(Throwable caught) {

                caught.printStackTrace();
                                        }
                                });
                        }
                }

}
```

PendingMembersPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.MailService;
import cepir.client.MailServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class PendingMembersPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
                String getSearchText();
                void clearSearchText();
                Integer getUsersPerPageText();
                HasClickHandlers getAcceptButton();
                HasClickHandlers getRejectButton();
                HasClickHandlers getRefreshButton();
                HasClickHandlers getSearchButton();
                HasClickHandlers getcancelSearchButton();
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasClickHandlers getUsersPerPageButton();
                HasKeyDownHandlers getSearchKeyHandler();
                void loadUsersList(ArrayList<ProjMemRequest>
users);
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                ArrayList<Integer> getSelectedRows();
                void usersPerPage_Warning(int left, int top);
        }
        private final ProjectServiceAsync rpcService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final Integer projID;
        private ArrayList<ProjMemRequest> requests;
        private Boolean isApproved;
        private int itemsPerPage = 5;
        private int totalPages;
        private int currPage = 1;

        private final String UNAPPROVED_SUBJ = "Project
Membership Request Unapproved - Collaboratory for Epidemiological
Research (CEpiR)";
        private final String APPROVED_SUBJ = "Project Membership
Request Approved - Collaboratory for Epidemiological Research (CEpiR)";

        public PendingMembersPresenter(HandlerManager
eventBus,Display display, Integer projID) {
                this.eventBus = eventBus;
                this.display = display;
                this.projID = projID;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                bind();
                performSearch();
        }

        private void bind() {
                display.getAcceptButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                isApproved = true;
                                ArrayList<Integer>
selectedRows = display.getSelectedRows();
                                for(int i = 0; i <
selectedRows.size(); i++){

                addMember(requests.get((itemsPerPage*(currPage-
1))+(selectedRows.get(i)-2)));
                                }
                        }

                });

                display.getRejectButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                isApproved = false;
                                ArrayList<Integer>
selectedRows = display.getSelectedRows();
                                for(int i = 0; i <
selectedRows.size(); i++){

                deleteRequest(requests.get((itemsPerPage*(currPage-
1))+(selectedRows.get(i)-2)));
                                }
                        }
                });

                display.getRefreshButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                History.newItem("#");

                History.newItem("members?p="+projID+"&tab=1");
                        }
```

```
                });


        display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        performSearch();
                                }
                        }
                });

                display.getSearchButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                performSearch();
                        }
                });


        display.getcancelSearchButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.clearSearchText();
                                performSearch();
                        }
                });


        display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage = 1;
                                paginate(1);
                        }
                });


        display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage = totalPages;
                                paginate(totalPages);
                        }
                });

                display.getNextButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage++;
                                paginate(currPage);
                        }
                });

                display.getPrevButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage--;
                                paginate(currPage);
                        }
                });

        display.getUsersPerPageButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Integer items =
display.getUsersPerPageText();

        if(items!=null&&items>0){

        itemsPerPage = items;

        performSearch();
                                }else{
                                        Widget
source = (Widget) event.getSource();
                                int left = source.getAbsoluteLeft() + 10;
                                int top = source.getAbsoluteTop() + 10;

        display.usersPerPage_Warning(left, top);
                                }
                        }
                });
        }

        private void paginate(int page){
                totalPages =
(int)Math.ceil((double)requests.size()/itemsPerPage);
                ArrayList<ProjMemRequest> concat = new
ArrayList<ProjMemRequest>();
                for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(requests.size()-
1)?requests.size():itemsPerPage*page); ctr++){
                        concat.add(requests.get(ctr));
                }
                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                display.loadUsersList(concat);
        }

        private void performSearch() {
                rpcService.getProjMemRequests(projID,
display.getSearchText(), new
AsyncCallback<ArrayList<ProjMemRequest>>() {

                        public void
onSuccess(ArrayList<ProjMemRequest> result) {
                                requests = result;
                                currPage = 1;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {

                        }
                });
        }

        private void addMember(final ProjMemRequest
projMemRequest) {

        rpcService.approveMemRequest(projMemRequest.projMemRe
questPK.username.username, projID,
                                1, false, new
AsyncCallback<Void>() { //1 (project member) is default membership type

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }

                        public void onSuccess(Void result) {

        deleteRequest(projMemRequest);
                        }
                });
        }
```

```
	private void deleteRequest(final ProjMemRequest
projMemRequest) {

	rpcService.deleteMemRequest(projMemRequest.projMemRequ
estPK.username.username, projID, new AsyncCallback<Void>() {

				public void onSuccess(Void result) {

	sendMail(projMemRequest);
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
	}

	protected void sendMail(ProjMemRequest projMemRequest) {
			if(!isApproved){
					MailServiceAsync mail =
GWT.create(MailService.class);
					String
UNAPPROVED_MSG = "We regret to inform you that your request for
membership in project
"+projMemRequest.projMemRequestPK.projID.projName.toUpperCase()
					+" at the Collaboratory
for Epidemiological Research (CEpiR) has been rejected. \n\n" + "NOTE:
This message has been generated automatically. Please do not reply to this
message.";
					mail.postMail(new
String[]{projMemRequest.projMemRequestPK.username.email},
UNAPPROVED_SUBJ, UNAPPROVED_MSG, new
AsyncCallback<Boolean>() {

						public void
onSuccess(Boolean result) {

	performSearch();
						}

						public void
onFailure(Throwable caught){

	caught.printStackTrace();
						}
					});
			}else{
					MailServiceAsync mail =
GWT.create(MailService.class);
					String
APPROVED_MSG = "Your request for membership in project
"+projMemRequest.projMemRequestPK.projID.projName.toUpperCase()
					+" at the Collaboratory
for Epidemiological Research (CEpiR) has been approved. \n\n" + "NOTE:
This message has been generated automatically. Please do not reply to this
message.";
					mail.postMail(new
String[]{projMemRequest.projMemRequestPK.username.email},
APPROVED_SUBJ, APPROVED_MSG, new AsyncCallback<Boolean>()
{

						public void
onSuccess(Boolean result) {

	performSearch();
						}

						public void
onFailure(Throwable caught) {

	caught.printStackTrace();
						}
					});
			}
	}

}
```

PendingProjListPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.MailService;
import cepir.client.MailServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class PendingMembersPresenter implements Presenter {
	public interface Display{
		Widget asWidget();
		String getSearchText();
		void clearSearchText();
		Integer getUsersPerPageText();
		HasClickHandlers getAcceptButton();
		HasClickHandlers getRejectButton();
		HasClickHandlers getRefreshButton();
		HasClickHandlers getSearchButton();
		HasClickHandlers getcancelSearchButton();
		HasClickHandlers getGoToFirstPageButton();
		HasClickHandlers getGoToLastPageButton();
		HasClickHandlers getPrevButton();
		HasClickHandlers getNextButton();
		HasClickHandlers getUsersPerPageButton();
		HasKeyDownHandlers getSearchKeyHandler();
		void loadUsersList(ArrayList<ProjMemRequest>
users);
		void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
		ArrayList<Integer> getSelectedRows();
		void usersPerPage_Warning(int left, int top);
	}
	private final ProjectServiceAsync rpcService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final Integer projID;
	private ArrayList<ProjMemRequest> requests;
	private Boolean isApproved;
	private int itemsPerPage = 5;
	private int totalPages;
	private int currPage = 1;

	private final String UNAPPROVED_SUBJ = "Project
Membership Request Unapproved - Collaboratory for Epidemiological
Research (CEpiR)";
	private final String APPROVED_SUBJ = "Project Membership
Request Approved - Collaboratory for Epidemiological Research (CEpiR)";

	public PendingMembersPresenter(HandlerManager
eventBus,Display display, Integer projID) {
		this.eventBus = eventBus;
		this.display = display;
		this.projID = projID;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		bind();
		performSearch();
	}
```

```
	private void bind() {
		display.getAcceptButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				isApproved = true;
				ArrayList<Integer>
selectedRows = display.getSelectedRows();
				for(int i = 0; i <
selectedRows.size(); i++){

	addMember(requests.get((itemsPerPage*(currPage-
1))+(selectedRows.get(i)-2)));
				}
			}

		});

		display.getRejectButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				isApproved = false;
				ArrayList<Integer>
selectedRows = display.getSelectedRows();
				for(int i = 0; i <
selectedRows.size(); i++){

	deleteRequest(requests.get((itemsPerPage*(currPage-
1))+(selectedRows.get(i)-2)));
				}
			}
		});

		display.getRefreshButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				History.newItem("#");

	History.newItem("members?p="+projID+"&tab=1");
			}
		});


	display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

			public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	performSearch();
				}
			}
		});

		display.getSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				performSearch();
			}
		});


	display.getcancelSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	display.clearSearchText();
				performSearch();
			}
		});

	display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage = 1;
				paginate(1);
			}
		});


	display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage = totalPages;
				paginate(totalPages);
			}
		});

		display.getNextButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage++;
				paginate(currPage);
			}
		});

		display.getPrevButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage--;
				paginate(currPage);
			}
		});


	display.getUsersPerPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Integer items =
display.getUsersPerPageText();

	if(items!=null&&items>0){

	itemsPerPage = items;

	performSearch();
				}else{
					Widget
source = (Widget) event.getSource();
				int left = source.getAbsoluteLeft() + 10;
				int top = source.getAbsoluteTop() + 10;

	display.usersPerPage_Warning(left, top);
					}
			}
		});
	}

	private void paginate(int page){
		totalPages =
(int)Math.ceil((double)requests.size()/itemsPerPage);
		ArrayList<ProjMemRequest> concat = new
ArrayList<ProjMemRequest>();
		for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(requests.size()-
1)?requests.size():itemsPerPage*page); ctr++){
			concat.add(requests.get(ctr));
```

```
                }
                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                display.loadUsersList(concat);
        }

        private void performSearch() {
                rpcService.getProjMemRequests(projID,
display.getSearchText(), new
AsyncCallback<ArrayList<ProjMemRequest>>() {

                        public void
onSuccess(ArrayList<ProjMemRequest> result) {
                                requests = result;
                                currPage = 1;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {

                        }
                });
        }

        private void addMember(final ProjMemRequest
projMemRequest) {

        rpcService.approveMemRequest(projMemRequest.projMemRe
questPK.username.username, projID,
                                1, false, new
AsyncCallback<Void>() { //1 (project member) is default membership type

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }

                        public void onSuccess(Void result) {

        deleteRequest(projMemRequest);
                        }
                });
        }

        private void deleteRequest(final ProjMemRequest
projMemRequest) {

        rpcService.deleteMemRequest(projMemRequest.projMemRequ
estPK.username.username, projID, new AsyncCallback<Void>() {

                        public void onSuccess(Void result) {

        sendMail(projMemRequest);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        protected void sendMail(ProjMemRequest projMemRequest) {
                if(!isApproved){
                                MailServiceAsync mail =
GWT.create(MailService.class);
                                String
UNAPPROVED_MSG = "We regret to inform you that your request for
membership in project
"+projMemRequest.projMemRequestPK.projID.projName.toUpperCase()
                                +" at the Collaboratory
for Epidemiological Research (CEpiR) has been rejected. \n\n" + "NOTE:
This message has been generated automatically. Please do not reply to this
message.";
                                mail.postMail(new
String[]{projMemRequest.projMemRequestPK.username.email},
UNAPPROVED_SUBJ, UNAPPROVED_MSG, new
AsyncCallback<Boolean>() {
                                        public void
onSuccess(Boolean result) {

                performSearch();
                                        }

                                        public void
onFailure(Throwable caught){

                caught.printStackTrace();
                                        }
                                });
                        }else{
                                MailServiceAsync mail =
GWT.create(MailService.class);
                                String
APPROVED_MSG = "Your request for membership in project
"+projMemRequest.projMemRequestPK.projID.projName.toUpperCase()
                                +" at the Collaboratory
for Epidemiological Research (CEpiR) has been approved. \n\n" + "NOTE:
This message has been generated automatically. Please do not reply to this
message.";
                                mail.postMail(new
String[]{projMemRequest.projMemRequestPK.username.email},
APPROVED_SUBJ, APPROVED_MSG, new AsyncCallback<Boolean>()
{

                                        public void
onSuccess(Boolean result) {

                performSearch();
                                        }

                                        public void
onFailure(Throwable caught) {

                caught.printStackTrace();
                                        }
                                });
                        }
        }

}
```

PendingUsersListTabPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.MailService;
import cepir.client.MailServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.AcctRequest;
import cepir.shared.User;
import cepir.shared.YesNo;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class PendingUsersListTabPresenter implements Presenter{
        public interface Display{
                Widget asWidget();
                String getSearchText();
                void clearSearchText();
                Integer getUsersPerPageText();
                void clearUsersPerPageText();
                HasClickHandlers getUsersPerPageButton();
                HasClickHandlers getAcceptButton();
                HasClickHandlers getRejectButton();
```

```
                HasClickHandlers getRefreshButton();
                HasClickHandlers getSearchButton();
                HasClickHandlers getcancelSearchButton();
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasKeyDownHandlers getSearchKeyHandler();
                void loadUsersList(ArrayList<AcctRequest>
users);
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                ArrayList<Integer> getSelectedRows();
                void sendingMail_Warning();
                void usersPerPage_Warning(int left, int top);
        }

        private final UserServiceAsync rpcService =
GWT.create(UserService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private ArrayList<AcctRequest> requests;
        private Boolean hasNotAddedUser;
        private int itemsPerPage = 5;
        private int totalPages;
        private int currPage = 1;

        private final String UNAPPROVED_SUBJ = "Account Request
Unapproved - Collaboratory for Epidemiological Research (CEpiR)";
        private final String UNAPPROVED_MSG = "We regret to
inform you that your account application at the Collaboratory for
Epidemiological Research (CEpiR) has been denied. \n\n" +
        "NOTE: This message has been generated automatically. Please
do not reply to this message.";
        private final String APPROVED_SUBJ = "Account Request
Approved - Collaboratory for Epidemiological Research (CEpiR)";
        private final String APPROVED_MSG = "Your account
application at the Collaboratory for Epidemiological Research (CEpiR) has
been approved. " +
        "You can now login using the information you have provided
during account registration.\n\n" +
        "NOTE: This message has been generated automatically. Please
do not reply to this message.";


        public PendingUsersListTabPresenter(HandlerManager
eventBus, Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }


        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                fetchRequests();
                bind();
        }

        public void fetchRequests(){
                rpcService.getAcctRequests(new
AsyncCallback<ArrayList<AcctRequest>>() {

                        public void
onSuccess(ArrayList<AcctRequest> result) {
                                requests = result;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void paginate(int page){
                totalPages =
(int)Math.ceil((double)requests.size()/itemsPerPage);
                ArrayList<AcctRequest> concat = new
ArrayList<AcctRequest>();
                for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(requests.size()-
1)?requests.size():itemsPerPage*page); ctr++){
                        concat.add(requests.get(ctr));
                }
                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                display.loadUsersList(concat);

        }

        public void bind(){
                display.getAcceptButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                hasNotAddedUser =
false;
                                ArrayList<Integer>
selectedRows = display.getSelectedRows();
                                for(int i = 0; i <
selectedRows.size(); i++){

        addUser(requests.get((itemsPerPage*(currPage-
1))+(selectedRows.get(i)-2)));
                                }
                        }
                });

                display.getRejectButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                hasNotAddedUser = true;
                                ArrayList<Integer>
selectedRows = display.getSelectedRows();
                                for(int i = 0; i <
selectedRows.size(); i++){

        rejectRequest(requests.get((itemsPerPage*(currPage-
1))+(selectedRows.get(i)-2)));
                                }
//                              rejectRequests();
                        }
                });

                display.getRefreshButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                History.newItem("#");

        History.newItem("userMngt?tab=1");
                        }
                });


        display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        performSearch();
                                }
                        }
                });

                display.getSearchButton().addClickHandler(new
ClickHandler() {
```

```
                public void onClick(ClickEvent
event) {
                        performSearch();
                }
        });


        display.getcancelSearchButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
        display.clearSearchText();
                        fetchRequests();
                }
        });


        display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        currPage = 1;
                        paginate(1);
                }
        });


        display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        currPage = totalPages;
                        paginate(totalPages);
                }
        });

        display.getNextButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        currPage++;
                        paginate(currPage);
                }
        });

        display.getPrevButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        currPage--;
                        paginate(currPage);
                }
        });


        display.getUsersPerPageButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        Integer items =
display.getUsersPerPageText();
        if(items!=null&&items>0){

        itemsPerPage = items;

        performSearch();
                        }else{
                                Widget
source = (Widget) event.getSource();
                int left = source.getAbsoluteLeft() + 10;
                int top = source.getAbsoluteTop() + 10;
        display.usersPerPage_Warning(left, top);
                        }
                }
        });
}

        private void performSearch(){

        rpcService.getAcctRequestSearchList(display.getSearchText(),
new AsyncCallback<ArrayList<AcctRequest>>() {

                public void
onSuccess(ArrayList<AcctRequest> result) {
                        requests = result;
                        currPage = 1;
                        paginate(1);
                }

                public void onFailure(Throwable
caught) {

                }
        });
}

        private void addUser(final AcctRequest acctRequest) {
                User user = new User();
                user.setFirstName(acctRequest.getFirstName());

        user.setMiddleName(acctRequest.getMiddleName());
                user.setLastName(acctRequest.getLastName());
                user.setEmail(acctRequest.getEmail());
                user.setUsername(acctRequest.getUsername());
                user.setAffiliation(acctRequest.getAffiliation());
                user.setPassword(acctRequest.getPassword());
                user.setQuestion(acctRequest.getQuestion());
                user.setAnswer(acctRequest.getAnswer());
                user.setSysAd(YesNo.No);
                user.setToolAd(YesNo.No);
                rpcService.addUser(user, true, new
AsyncCallback<Boolean>() {

                public void onSuccess(Boolean result)
{
        rejectRequest(acctRequest);
                }

                public void onFailure(Throwable
caught) {
                        caught.printStackTrace();
                }
        });
        }

        private void rejectRequest(final AcctRequest acctRequest) {
                rpcService.deleteRequest(acctRequest.username,
new AsyncCallback<Void>() {

                public void onSuccess(Void result) {
                        sendMail(acctRequest);
                }

                public void onFailure(Throwable
caught) {
                        caught.printStackTrace();
                }
        });
        }

        private void sendMail(AcctRequest acctRequest) {
                MailServiceAsync mail =
GWT.create(MailService.class);
                mail.postMail(new
String[]{acctRequest.getEmail()},
(hasNotAddedUser?UNAPPROVED_SUBJ:APPROVED_SUBJ),
(hasNotAddedUser?UNAPPROVED_MSG:APPROVED_MSG), new
AsyncCallback<Boolean>() {
```

```
                                        public void onSuccess(Boolean result)
{
                                                currPage = 1;
                                                fetchRequests();
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }
}
```

Presenter.java

```
package cepir.client.presenter;

import com.google.gwt.user.client.ui.HasWidgets;

public abstract interface Presenter {
  public abstract void go(final HasWidgets container);
}
```

ProjHomePresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.ProjMemRequest;
import cepir.shared.ProjMemRequestPK;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ProjHomePresenter implements Presenter{
        public interface Display{
                Widget asWidget();
                void setProjName(String projName);
                void setProjDesc(String projDesc);
                void showDeleteWarning();
                void hideDeleteWarning();
                void showLeaveWarning();
                void hideLeaveWarning();
                void showJoinMessage();
                void hideJoinMessage();
                void showSendNotice();
                void hideSendNotice();
                void clearMsg();
                String getMessage();
                void setErrorTxt(String msg);
                HasClickHandlers getDeleteOKButton();
                HasClickHandlers getDeleteNotOKButton();
                HasClickHandlers getLeaveOKButton();
                HasClickHandlers getLeaveNotOKButton();
                HasClickHandlers getJoinOkButton();
                HasClickHandlers getJoinNotOkButton();
                HasClickHandlers getSendOkButton();
                HasClickHandlers getJoinButton();
                HasClickHandlers getLeaveButton();
                HasClickHandlers getEditButton();
                HasClickHandlers getDeleteButton();
                void setVisibilityJoin(Boolean isVisible);
                void setVisibilityLeave(Boolean isVisible);
                void setVisibilityEdit(Boolean isVisible);
                void setVisibilityDelete(Boolean isVisible);
                void joinProject_Warning(int left, int top);
        }
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final Integer projID;
        private String username; //person who is logged in

        public ProjHomePresenter(HandlerManager eventBus, Display
display, Integer projID) {
                this.eventBus = eventBus;
                this.display = display;
                this.projID = projID;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                bind();
                loadProjInfo();
                getUsername();
        }

        private void bind() {
                display.getJoinButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Widget source =
(Widget) event.getSource();
                int left = source.getAbsoluteLeft() + 10;
                int top = source.getAbsoluteTop() + 10;
                                checkIfPending(left,
top);
                        }
                });

                display.getEditButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("editProject="+projID.toString());
                        }
                });

                display.getLeaveButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.showLeaveWarning();
                        }
                });

                display.getDeleteButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.showDeleteWarning();
                        }
                });


        display.getDeleteOKButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                deleteProject();
                        }
                });
```

```
		display.getDeleteNotOKButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {

		display.hideDeleteWarning();
					}
			});


		display.getLeaveOKButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {
						leaveProject();
					}
			});


		display.getLeaveNotOKButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {

		display.hideLeaveWarning();
					}
			});

			display.getJoinOkButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {

		if(display.getMessage().isEmpty()){

		display.setErrorTxt("Message is required");
						}else{

		saveRequest();
						}
					}
			});


		display.getJoinNotOkButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {

		display.hideJoinMessage();
					}
			});

			display.getSendOkButton().addClickHandler(new
ClickHandler() {

					public void onClick(ClickEvent
event) {

		display.hideSendNotice();
					}
			});
		}

		private void refresh(){
			History.newItem("#");
			History.newItem("projHome="+projID);
		}

		private void saveRequest(){
			ProjMemRequest  request =  new
ProjMemRequest();
			request.message = display.getMessage();
			User user = new User();
			user.username = username;
			Project proj = new Project();
			proj.projID = projID;
			request.projMemRequestPK = new
ProjMemRequestPK(user, proj);
			projService.saveMemRequest(request, new
AsyncCallback<Void>() {

					public void onSuccess(Void result) {

		display.hideJoinMessage();

		display.showSendNotice();
					}

					public void onFailure(Throwable
caught) {

					}
			});
		}

		private void loadProjInfo() {
			projService.getProject(projID, new
AsyncCallback<Project>() {

					public void onSuccess(Project result)
{

		display.setProjName(result.projName);

		display.setProjDesc(result.projDesc);

					}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
					}
			});
		}

		private void getUsername(){
			UserServiceAsync userService =
GWT.create(UserService.class);
			userService.getSessionKeyData("username", new
AsyncCallback<String>() {

					public void onSuccess(String result) {
						username = result;
						setupVisibility();
					}

					public void onFailure(Throwable
caught) {
						caught.printStackTrace();
					}
			});
		}

		private void setupVisibility(){
			projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

					public void onSuccess(Role result) {

		if(result.roleID==null||result.roleName.isEmpty()){

		display.setVisibilityJoin(true);
						}else
if(result.roleName.equalsIgnoreCase("project member")){

		display.setVisibilityLeave(true);
						}else{

		display.setVisibilityEdit(true);

		display.setVisibilityDelete(true);

		display.setVisibilityLeave(true);
```

```
                                        }
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                private void checkIfPending(final int left, final int top){
                        projService.getProjMemRequest(username,
projID, new AsyncCallback<ProjMemRequest>() {

                                public void
onSuccess(ProjMemRequest result) {
                                        if(result==null){

                display.showJoinMessage();

                display.setErrorTxt("");

                display.clearMsg();
                                        }else{


                display.joinProject_Warning(left, top);
                                        }
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                private void leaveProject(){
                        projService.deleteUserPriv(username, projID, new
AsyncCallback<Void>() {

                                public void onSuccess(Void result) {

                display.hideLeaveWarning();
                                        refresh();
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

                private void deleteProject(){
                        projService.deleteProject(projID, new
AsyncCallback<Void>() {

                                public void onSuccess(Void result) {

                display.hideDeleteWarning();

                History.newItem("myProjects");
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
                }

}
```

ProjListTabPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.event.AddProjectEvent;
import cepir.client.event.AddUserEvent;
import cepir.client.view.ViewProjView;
import cepir.shared.AcctRequest;
import cepir.shared.Project;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HTMLTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class ProjListTabPresenter implements Presenter{
        public interface Display{
                Widget asWidget();
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                void loadProjectList(ArrayList<Project> projects);
                Integer getUsersPerPageText();
                void clearUsersPerPageText();
                String getSearchText();
                void clearSearchText();
                Integer getSelectedRow();
                void clearRowStyles();
                HasClickHandlers getDeleteButton();
                HasClickHandlers getAddButton();
                HasClickHandlers getEditButton();
                HasClickHandlers getRefreshButton();
                HasClickHandlers getSearchButton();
                HasClickHandlers getcancelSearchButton();
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasClickHandlers getUsersPerPageButton();
                HasClickHandlers getDeleteOkButton();
                HasClickHandlers getDeleteNotOkButton();
                HasKeyDownHandlers getSearchKeyHandler();
                FlexTable getProjListTable();
                void delete_Warning(int left, int top);
                void editProj_Warning(int left, int top);
                void usersPerPage_Warning(int left, int top);
                void showDeleteProjWarning();
                void hideDeleteProjWarning();
        }

        private final ProjectServiceAsync rpcService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private ArrayList<Project> projects;
        private int itemsPerPage = 5;
        private int totalPages;
        private int currPage = 1;

        public ProjListTabPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
```

```
                fetchProjects();
                bind();
        }

        private void fetchProjects() {
                rpcService.getProjects(new
AsyncCallback<ArrayList<Project>>() {

                        public void
onSuccess(ArrayList<Project> result) {
                                projects = result;
                                currPage = 1;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void bind() {
//                display.getProjListTable().addClickHandler(new
ClickHandler() {
//
//                        public void onClick(ClickEvent
event) {
//                                Cell cell =
display.getProjListTable().getCellForEvent(event);
//                                if(cell!=null){
//                                        selectedRow
= cell.getRowIndex();
//
        display.clearRowStyles();
//
        if(selectedRow>=2){
//
        display.getProjListTable().getRowFormatter().setStyleName(se
lectedRow,"tableHighlight");
//
        ViewProjView proj = new ViewProjView();
//
        ViewProjPresenter projPresenter = new
ViewProjPresenter(eventBus, proj, projects.get((selectedRow-
2)+((currPage-1)*itemsPerPage)).getProjName());
//
        display.setProjLoader(proj);
//
        projPresenter.go(null);
//                                        }
//                                }
//                        }
//                });

                display.getProjListTable().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Cell cell =
display.getProjListTable().getCellForEvent(event);
                                Integer selected = null;
                                if(!projects.isEmpty()){
                                        selected =
display.getSelectedRow();
                                }

        if(cell!=null&&selected!=null){

        display.clearRowStyles();

        display.getProjListTable().getRowFormatter().setStyleName(se
lected,"tableHighlight");
                                }
                        }
                });

        display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        performSearch();
                                }
                        }
                });

                display.getAddButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                eventBus.fireEvent(new
AddProjectEvent());
                        }
                });

                display.getEditButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        if(display.getSelectedRow()==null){
                                Widget
source = (Widget) event.getSource();
                        int left = source.getAbsoluteLeft() + 10;
                        int top = source.getAbsoluteTop() + 10;

        display.editProj_Warning(left, top);
                                }else{

        History.newItem("editProject="+projects.get((display.getSelect
edRow()-2)+((currPage-1)*itemsPerPage)).projID);
                                }
                        }
                });

                display.getDeleteButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Integer selected =
display.getSelectedRow();
                                if(selected==null){
                                Widget
source = (Widget) event.getSource();
                        int left = source.getAbsoluteLeft() + 15;
                        int top = source.getAbsoluteTop() + 15;

        display.delete_Warning(left, top);
                                }else{

        display.showDeleteProjWarning();
                                }
                        }
                });

                display.getSearchButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                performSearch();
                        }
                });


        display.getcancelSearchButton().addClickHandler(new
ClickHandler() {
```

```
                                        public void onClick(ClickEvent
event) {

                display.clearSearchText();
                                                fetchProjects();
                                        }
                                });

                                display.getRefreshButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                History.newItem("#");

                History.newItem("projMngt?tab=0");
                                        }
                                });


                display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                currPage = 1;
                                                paginate(1);
                                        }
                                });


                display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                currPage = totalPages;
                                                paginate(totalPages);
                                        }
                                });

                                display.getNextButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                currPage++;
                                                paginate(currPage);
                                        }
                                });

                                display.getPrevButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                currPage--;
                                                paginate(currPage);
                                        }
                                });


                display.getUsersPerPageButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                Integer items =
display.getUsersPerPageText();

                if(items!=null&&items>0){

                itemsPerPage = items;

                performSearch();
                                                }else{
                                                        Widget
source = (Widget) event.getSource();
                                int left = source.getAbsoluteLeft() + 10;
                                int top = source.getAbsoluteTop() + 10;
                display.usersPerPage_Warning(left, top);
                                                }
                                        }
                                });


                display.getDeleteOkButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {

                display.hideDeleteProjWarning();
                                                deleteProject();
                                        }
                                });


                display.getDeleteNotOkButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {

                display.hideDeleteProjWarning();
                                        }
                                });
                }

                private void performSearch(){

                rpcService.getProjectSearchList(display.getSearchText(), new
AsyncCallback<ArrayList<Project>>() {

                                        public void
onSuccess(ArrayList<Project> result) {
                                                projects = result;
                                                currPage = 1;
                                                paginate(1);
                                        }

                                        public void onFailure(Throwable
caught) {

                                        }
                                });
                }

                private void paginate(int page){
                                totalPages =
(int)Math.ceil((double)projects.size()/itemsPerPage);
                                ArrayList<Project> concat = new
ArrayList<Project>();
                                for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(projects.size()-
1)?projects.size():itemsPerPage*page); ctr++){
                                        concat.add(projects.get(ctr));
                                }
                                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                                display.loadProjectList(concat);

                }

                private void deleteProject() {

                rpcService.deleteProject(projects.get((display.getSelectedRow()
-2)+((currPage-1)*itemsPerPage)).getProjID(), new
AsyncCallback<Void>() {

                                        public void onSuccess(Void result) {
                                                currPage = 1;

                display.clearSearchText();
                                                fetchProjects();
                                        }

                                        public void onFailure(Throwable
caught) {
```

```
                                        caught.printStackTrace();
                                }
                        });
        }

}
```

ProjMainPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.presenter.MainPresenter.Display;
import cepir.shared.Project;
import cepir.shared.Role;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Widget;

public class ProjMainPresenter implements Presenter {

        private final UserServiceAsync rpcService =
GWT.create(UserService.class);
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final MainPresenter.Display display;
        private final Presenter newPagePresenter;
        private final Widget newPageView;
        private final Integer projID;
        private ArrayList<Project> projects;

        public ProjMainPresenter(HandlerManager eventBus,
MainPresenter.Display display, Presenter newPagePresenter, Widget
newPageView, Integer projID) {
                this.eventBus = eventBus;
                this.display = display;
                this.newPagePresenter = newPagePresenter;
                this.newPageView = newPageView;
                this.projID = projID;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                display.clearList();
                setupCenterHeader();
                setSideBar();
                setupNewPage();
                bind();
        }

        private void bind() {
                display.getGoButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Integer x =
display.getSelectedItem();

        if(display.getSelectedItem()!=-1){

        History.newItem("projHome="+projects.get(display.getSelecte
dItem()).projID);
                                }
                        });

                display.getOptionsButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("manageAcct");
                        }
                });
        }

        private void setupNewPage() {
                display.setPage(newPageView);
                if(newPagePresenter!=null){
                        newPagePresenter.go(null);
                }
        }

        private void setSideBar() {
                rpcService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                checkRole(result);
                        }

                        public void onFailure(Throwable
caught) {

                        }
                });
        }

        private void checkRole(String username){
                ProjectServiceAsync projService =
GWT.create(ProjectService.class);
                projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

                        public void onSuccess(Role result) {
                                FlexTable menu = new
FlexTable();

                                Hyperlink home = new
Hyperlink("Home", "memberIndex");
                                menu.setWidget(0, 0,
home);
                                Hyperlink projHome =
new Hyperlink("Project Home", "projHome="+projID.toString());
                                menu.setWidget(1, 0,
projHome);

        if(result.roleID==null||result.roleName.isEmpty()){
                                }else{
                                        Hyperlink
members = new Hyperlink("Members",
"members?p="+projID.toString()+"&tab=0");

        menu.setWidget(2, 0, members);
                                Hyperlink
epiData = new Hyperlink("Epidemiological Data",
"data?p="+projID.toString());

        menu.setWidget(3, 0, epiData);
                                Hyperlink
refs = new Hyperlink("References", "refs/"+projID.toString()+"/");

        menu.setWidget(4, 0, refs);
                                Hyperlink
miscs = new Hyperlink("Miscellaneous Files",
"miscFiles/"+projID.toString()+"/");

        menu.setWidget(5, 0, miscs);
                                }
                                Hyperlink logout = new
Hyperlink("Logout", "index");
```

```
		logout.addClickHandler(new ClickHandler() {

                                                                public void
onClick(ClickEvent event) {

		LogoutPresenter logoutPresenter = new LogoutPresenter();

		logoutPresenter.go(null);

		History.newItem("index");
                                                                }
                                                        });

		menu.setWidget((result.roleID==null||result.roleName.isEmpty(
)?2:6),0,logout);

		display.setSideMenu(menu);
                                        }

                                        public void onFailure(Throwable
caught) {

                                        }
                        });
		}

		private void setupCenterHeader(){
                        rpcService.getSessionKeyData("username", new
AsyncCallback<String>() {

                                        public void onSuccess(String result) {
                                                fetchProjects(result);
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                        });
		}

		private void fetchProjects(String username){
                        ProjectServiceAsync projService =
GWT.create(ProjectService.class);
                        projService.getProjects(username, new
AsyncCallback<ArrayList<Project>>() {

                                        public void
onSuccess(ArrayList<Project> result) {
                                                projects = result;
                                                if(!result.isEmpty()){

		display.setCenterHeader(result);
                                                                for(int i = 0;
i < projects.size(); i++){

		if(projects.get(i).projID==projID){

		display.setSelectedItem(i);

		}
                                                                }
                                                }else{

		display.hideHeader();
                                                }

                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                        });
		}
}
```

ProjMngtPagePresenter.java

```
package cepir.client.presenter;

import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.presenter.UserMngtPagePresenter.Display;
import cepir.client.view.PendingProjListTabView;
import cepir.client.view.PendingUsersListTabView;
import cepir.client.view.ProjListTabView;
import cepir.client.view.UsersListTabView;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ProjMngtPagePresenter implements Presenter{

		public interface Display{
                        Widget asWidget();
                        void addTab(Widget widget,String tabName);
                        HasSelectionHandlers<Integer> getTabPanel();
                        void setSelectedTab(int index);
                        void replaceTab(int index, Widget widget, String
tabName);
		}

		private final HandlerManager eventBus;
		private final Display display;
		private final Integer selectedIndex;


		public ProjMngtPagePresenter(HandlerManager eventBus,
Display display, Integer selectedIndex) {
                        this.eventBus = eventBus;
                        this.display = display;
                        this.selectedIndex = selectedIndex;
		}


		public void go(HasWidgets container) {
                        setupTabs();
                        bind();
                        if(container!=null){
                                container.clear();
                                container.add(display.asWidget());
                        }
		}

		private void bind() {
                        display.getTabPanel().addSelectionHandler(new
SelectionHandler<Integer>() {

                                public void
onSelection(SelectionEvent<Integer> event) {

		History.newItem("projMngt?tab="+event.getSelectedItem());
                                }
                        });
		}


		private void setupTabs() {
                        //tab 1
                        ProjListTabView projTab = new
ProjListTabView();
                        display.addTab(projTab, "Projects");

                        //tab 2
                        PendingProjListTabView pendingProjTab = new
PendingProjListTabView();
                        display.addTab(pendingProjTab, "Project
Requests");

                        display.setSelectedTab(selectedIndex);

                        if(selectedIndex==0){
```

```
                                        ProjListTabPresenter
projTabPresenter = new ProjListTabPresenter(eventBus, projTab);
                                        projTabPresenter.go(null);
                                }else{
                                        PendingProjListPresenter
pendingProjTabPresenter = new PendingProjListPresenter(eventBus,
pendingProjTab);
                                        pendingProjTabPresenter.go(null);
                                }
                        }

}
```

RequestForAcctPresenter.java

```
package cepir.client.presenter;

import cepir.client.InputChecker;
import cepir.client.MailService;
import cepir.client.MailServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.shared.AcctRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class RequestForAcctPresenter implements Presenter{
        public interface Display{
                HasClickHandlers getSubmitButton();
                HasClickHandlers getClearButton();
                HasClickHandlers getOkDialog();
                HasClickHandlers getInfoButton();
                void setErrorTxt(String errMsg);
                void setUsernameMsg(String usernameMsg);
                void setEmailMsg(String emailMsg);
                void setRetypePasswdMsg(String passwdMsg);
                void clearForm();
                void showInfo();
                String getFirstName();
                String getMiddleName();
                String getLastName();
                String getEmail();
                String getAffiliation();
                String getUsername();
                String getPassword();
                String getRetypedPassword();
                String getQuestion();
                String getAnswer();
                void requestSent();
                Widget asWidget();
        }

        private final UserServiceAsync rpcService =
GWT.create(UserService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final String REQSENT_SUBJ = "Request for account at
CEpiR was sent";
        private final String REQSENT_MSG = "Your request for an
account at the Collaboratory for Epidemiological Research"+
        " was successfully sent. A notification will be sent to you via
email once your request has been processed. Thank you.\n\n"+
        "NOTE: This message has been generated automatically. Please
do not reply to this message.";

        public RequestForAcctPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
        }

        public void bind(){
                display.getSubmitButton().addClickHandler(new
ClickHandler() {
                        public void onClick(ClickEvent
event) {
                                saveRequest();
                        }
                });

                display.getClearButton().addClickHandler(new
ClickHandler() {
                        public void onClick(ClickEvent
event) {
                                display.clearForm();
                        }
                });
                display.getOkDialog().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        History.newItem("index");
                        }
                });
                display.getInfoButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                display.showInfo();
                        }
                });
        }

        public void checkUsernameAndEmailIfUsed(){

        rpcService.usernameIsNotUsed(display.getUsername(), new
AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{
                                if(result==true){

        display.setUsernameMsg("");
                                }else{

        display.setUsernameMsg("Username already in use");
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });

                rpcService.emailIsNotUsed(display.getEmail(),
new AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{
                                if(result==true){

        display.setEmailMsg("");
                                }else{

        display.setEmailMsg("Email already in use");
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
```

```
                        }
                });
        }

        private Boolean passwdMatches(){
                return
display.getPassword().equals(display.getRetypedPassword());
        }

        public void saveRequest(){
                InputChecker input = new InputChecker();

        if(!input.isValidUsername(display.getUsername())){
                        display.setErrorTxt("");
                        display.setEmailMsg("");
                        display.setRetypePasswdMsg("");
                        display.setUsernameMsg("Username
must start with a letter, must be 6-24 characters long and should contain
letters, numbers, underscores (_) and dots (.) only");
                }else if(!input.isValidEmail(display.getEmail())){
                        display.setEmailMsg("Invalid email
address");
                        display.setUsernameMsg("");
                        display.setErrorTxt("");
                        display.setRetypePasswdMsg("");
                }else
if(display.getEmail().isEmpty()||display.getUsername().isEmpty()||display.ge
tPassword().isEmpty()||display.getFirstName().isEmpty()

        ||display.getMiddleName().isEmpty()||display.getLastName().is
Empty()||display.getQuestion().isEmpty()||display.getAnswer().isEmpty()){
                        display.setErrorTxt("Required field/s
missing");
                        display.setUsernameMsg("");
                        display.setEmailMsg("");
                        display.setRetypePasswdMsg("");
                }else{
                        AcctRequest acct = new
AcctRequest(display.getEmail(), display.getUsername(),
display.getPassword(),display.getLastName(), display.getFirstName(),

        display.getMiddleName(),
display.getAffiliation(),display.getQuestion(),display.getAnswer());
                        checkUsernameAndEmailIfUsed();
                        if(passwdMatches()){

        rpcService.saveRequestForAccount(acct, new
AsyncCallback<String>() {

                                                        public void
onSuccess(String result) {

        if(result.equalsIgnoreCase("Request sent.")){

        sendMail();

        display.requestSent();

        }else{

        display.setErrorTxt(result);

        }
                                                        }

                                                        public void
onFailure(Throwable caught) {

        caught.printStackTrace();
                                                        }
                                                });
                        }else{

        display.setRetypePasswdMsg("Password does not match");
                        }
                }
        }

        private void sendMail(){
                String[] recipient = {display.getEmail()};
                MailServiceAsync mail =
GWT.create(MailService.class);
                mail.postMail(recipient, REQSENT_SUBJ,
REQSENT_MSG, new AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{

                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }
}
```

RequestForProjPresenter.java

```
package cepir.client.presenter;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.presenter.PendingProjListPresenter.Display;
import cepir.shared.NewProjRequest;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class RequestForProjPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
                String getProjName();
                String getProjDesc();
                void setProjNameErrTxt(String text);
                void setErrTxt(String text);
                void clearFields();
                void requestSent();
                HasClickHandlers getOkDialog();
                HasClickHandlers getSendButton();
                HasClickHandlers getCancelButton();
                HasClickHandlers getClearButton();
        }

        private final ProjectServiceAsync rpcService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private String requestedBy;

        public RequestForProjPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
                getPersonWhoRequested();
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                bind();
        }

        private void getPersonWhoRequested(){
                rpcService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                if(result==null){
```

```
		History.newItem("index");
					}else{
						requestedBy
= result;
					}
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void setupHistory(){
			History.newItem("memberIndex");
		}

		private void bind(){
			display.getSendButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					sendRequest();
				}
			});
			display.getCancelButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					setupHistory();
				}
			});
			display.getClearButton().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					display.clearFields();
				}
			});

			display.getOkDialog().addClickHandler(new
ClickHandler() {

				public void onClick(ClickEvent
event) {
					setupHistory();
				}
			});
		}

		private void sendRequest(){

		if(display.getProjDesc().isEmpty()||display.getProjName().isEm
pty()){
				display.setErrTxt("Project
name/description missing");
			}else{
				NewProjRequest newProj = new
NewProjRequest();
				newProj.newProjName =
display.getProjName();
				newProj.newProjDesc =
display.getProjDesc();
				User user = new User();
				user.username = requestedBy;
				newProj.user = user;
				rpcService.addProjRequest(newProj,
new AsyncCallback<String>() {

					public void
onSuccess(String result) {

		if(result.equals("ok")){

		display.requestSent();
			}else
if(result.isEmpty()){

		display.setProjNameErrTxt("Project name already in use");
			}else{

		display.setErrTxt(result);
			}
		}

		public void
onFailure(Throwable caught) {

		caught.printStackTrace();
		}
		});
	}
	}

}
```

UnauthorizedNoticePresenter.java

```
package cepir.client.presenter;

import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class UnauthorizedNoticePresenter implements Presenter{
	public interface Display{
		void setErrorMsg(String text);
		Widget asWidget();
	}

	private final Display display;
	private String message;

	public UnauthorizedNoticePresenter(Display display) {
		this.display = display;
		this.message = "You are not authorized to view
this page";
	}

	public UnauthorizedNoticePresenter(Display display, String
message) {
		this.display = display;
		this.message = message;
	}


	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		display.setErrorMsg(message);
	}

}
```

UserMngtPagePresenter.java

```
package cepir.client.presenter;

import cepir.client.view.PendingUsersListTabView;
import cepir.client.view.UsersListTabView;

import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;


public class UserMngtPagePresenter implements Presenter{

	public interface Display{
		Widget asWidget();
		void addTab(Widget widget,String tabName);
		HasSelectionHandlers<Integer> getTabPanel();
```

```
                        void setSelectedTab(int index);
                        void replaceTab(int index, Widget widget, String
tabName);
        }

        private final HandlerManager eventBus;
        private final Display display;
        private final Integer selectedIndex;

        public UserMngtPagePresenter(HandlerManager
eventBus,Display view, Integer selectedIndex){
                this.eventBus = eventBus;
                this.display = view;
                this.selectedIndex = selectedIndex;
        }

        public void go(HasWidgets container) {
                setupTabs();
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
        }

        private void bind(){
                display.getTabPanel().addSelectionHandler(new
SelectionHandler<Integer>() {

                        public void
onSelection(SelectionEvent<Integer> event) {

        History.newItem("userMngt?tab="+event.getSelectedItem());
                        }
                });
        }

        public void setupTabs(){
                //tab 1
                UsersListTabView userTab = new
UsersListTabView();
                display.addTab(userTab, "Users");

                //tab 2
                PendingUsersListTabView pendingUserTab = new
PendingUsersListTabView();
                display.addTab(pendingUserTab, "Account
Requests");

                display.setSelectedTab(selectedIndex);
                if(selectedIndex==0){
                        UsersListTabPresenter
userTabPresenter = new UsersListTabPresenter(eventBus, userTab);
                        userTabPresenter.go(null);
                }else{
                        PendingUsersListTabPresenter
pendingUserTabPresenter = new PendingUsersListTabPresenter(eventBus,
pendingUserTab);
                        pendingUserTabPresenter.go(null);
                }

        }

}
```

UsersListTabPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.event.AddUserEvent;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class UsersListTabPresenter implements Presenter{
        public interface Display{
                Widget asWidget();
                void loadUsersList(ArrayList<User> users);
                void editUser_Warning(int left, int top);
                void delUser_Warning(int left, int top);
                void usersPerPage_Warning(int left, int top);
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                Integer getSelectedRow();
                String getSearchText();
                void clearSearchText();
                void clearRowStyles();
                Integer getUsersPerPageText();
                void clearUsersPerPageText();
                void showDeleteUserWarning();
                void hideDeleteUserWarning();
                void showConfirmDeleteWarning();
                void hideConfirmDeleteWarning();
                FlexTable getUsersListTable();
                HasClickHandlers getUsersPerPageButton();
                HasClickHandlers getDeleteButton();
                HasClickHandlers getAddButton();
                HasClickHandlers getEditButton();
                HasClickHandlers getRefreshButton();
                HasClickHandlers getSearchButton();
                HasClickHandlers getcancelSearchButton();
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasClickHandlers getDeleteOkButton();
                HasClickHandlers getConfirmDeleteButton();
                HasClickHandlers getRejectDeleteButton();
                HasKeyDownHandlers getSearchKeyHandler();
                HasClickHandlers getUsername();
                HasClickHandlers getFirstName();
                HasClickHandlers getMiddleName();
                HasClickHandlers getLastName();
                HasClickHandlers getEmail();
                HasClickHandlers getSysAd();
        }

        private final UserServiceAsync userService =
GWT.create(UserService.class);
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private ArrayList<User> users;
        ArrayList<String> soleProjAdmins;
        private Integer selectedRow;
        private int itemsPerPage = 5;
        private int totalPages;
        private int currPage = 1;

        public UsersListTabPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                fetchUsersList();
```

```
		bind();
	}

	public void fetchUsersList() {
		userService.getUsersList(new
AsyncCallback<ArrayList<User>>() {

			public void
onSuccess(ArrayList<User> result) {
				users = result;
				paginate(1);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	public void bind(){
		display.getUsersListTable().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Cell cell =
display.getUsersListTable().getCellForEvent(event);
				Integer selected = null;
				if(!users.isEmpty()){
					selected =
display.getSelectedRow();
				}

	if(cell!=null&&selected!=null){

	display.clearRowStyles();

	display.getUsersListTable().getRowFormatter().setStyleName(s
elected,"tableHighlight");
				}
			}
		});


	display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

			public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	performSearch();
				}
			}
		});

		display.getDeleteButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				selectedRow =
display.getSelectedRow();
				if(selectedRow==null){
					Widget
source = (Widget) event.getSource();
				int left = source.getAbsoluteLeft() + 10;
				int top = source.getAbsoluteTop() + 10;

	display.editUser_Warning(left, top);
				}else{

	checkDelete();
				}
			}
		});

		display.getAddButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				eventBus.fireEvent(new
AddUserEvent());
			}
		});

		display.getRefreshButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				History.newItem("#");

	History.newItem("userMngt?tab=0");
			}
		});

		display.getEditButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				editUser(event);
			}
		});

		display.getSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				performSearch();
			}
		});


	display.getcancelSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	display.clearSearchText();
				fetchUsersList();
			}
		});


	display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage = 1;
				paginate(1);
			}
		});


	display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage = totalPages;
				paginate(totalPages);
			}
		});

		display.getNextButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage++;
				paginate(currPage);
```

```
                        }
                });

                display.getPrevButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage--;
                                paginate(currPage);
                        }
                });


        display.getUsersPerPageButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                Integer items =
display.getUsersPerPageText();

        if(items!=null&&items>0){

        itemsPerPage = items;

        performSearch();
                                }else{
                                        Widget
source = (Widget) event.getSource();
                        int left = source.getAbsoluteLeft() + 10;
                        int top = source.getAbsoluteTop() + 10;

        display.usersPerPage_Warning(left, top);
                                }
                        }
                });


        display.getDeleteOkButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.hideDeleteUserWarning();
                                currPage = 1;
                                fetchUsersList();
                        }
                });


        display.getRejectDeleteButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.hideConfirmDeleteWarning();
                        }
                });


        display.getConfirmDeleteButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {

        display.hideConfirmDeleteWarning();
                                deleteUser();
                        }
                });
        }

        private void paginate(int page){
                totalPages =
(int)Math.ceil((double)users.size()/itemsPerPage);
                ArrayList<User> concat = new
ArrayList<User>();
                for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(users.size()-
1)?users.size():itemsPerPage*page); ctr++){
                        concat.add(users.get(ctr));
                }
                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                display.loadUsersList(concat);
        }

        private void performSearch(){

        userService.getUserSearchList(display.getSearchText(), new
AsyncCallback<ArrayList<User>>() {

                        public void
onSuccess(ArrayList<User> result) {
                                users = result;
                                currPage = 1;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {

                        }
                });
        }

        private void editUser(ClickEvent event){
                selectedRow = display.getSelectedRow();
                if(selectedRow==null){
                        Widget source = (Widget)
event.getSource();
        int left = source.getAbsoluteLeft() + 10;
        int top = source.getAbsoluteTop() + 10;
                        display.editUser_Warning(left, top);
                }else{

        History.newItem("editUser?uname="+users.get((itemsPerPage*
(currPage-1))+(selectedRow-3)).getUsername());
                }
        }

        private void checkIfSoleProjAdmin(){
                selectedRow = display.getSelectedRow();

        projService.checkIfSoleProjAdmin(users.get((itemsPerPage*(c
urrPage-1))+(selectedRow-3)).getUsername(), new
AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{
                                if(result){

        display.showConfirmDeleteWarning();
                                }else{

        deleteUser();
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void checkDelete(){
                selectedRow = display.getSelectedRow();
                userService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {

        if(result.equalsIgnoreCase(users.get((itemsPerPage*(currPage-
1))+(selectedRow-3)).getUsername())){

        display.showDeleteUserWarning();
```

```
                                        }else{

            checkIfSoleProjAdmin();
                                        }
                                    }

                                    public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                    }
                                });
            }

            private void deleteUser(){

            userService.deleteUser(users.get((itemsPerPage*(currPage-
1))+(selectedRow-3)).getUsername(), new AsyncCallback<Void>() {

                                    public void onSuccess(Void result) {
                                        currPage = 1;

            display.clearSearchText();
                                        fetchUsersList();
                                    }

                                    public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                    }
                                });
            }

}
```

ViewAllProjsPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.presenter.ViewMyProjsPresenter.Display;
import cepir.shared.NewProjRequest;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ViewAllProjsPresenter implements Presenter {
        public interface Display{
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                void loadProjectList(ArrayList<Project> projects,
ArrayList<String> isMember);
                String getSearchText();
                void clearSearchText();
                Integer getProjsPerPageText();
                void clearProjsPerPageText();
                HasClickHandlers getSearchButton();
                HasClickHandlers getCancelSearchButton();
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasClickHandlers getProjsPerPageButton();
                HasKeyDownHandlers getSearchKeyHandler();
                void projsPerPage_Warning(int left, int top);
                Widget asWidget();
        }
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private ArrayList<Project> projects;
        private String user;
        private int itemsPerPage = 5;
        private int totalPages;
        private int currPage = 1;

        public ViewAllProjsPresenter(HandlerManager eventBus,
Display display) {
                this.eventBus = eventBus;
                this.display = display;
        }

        public void go(HasWidgets container) {
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                getUsername();
        }

        private void bind() {

        display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage = 1;
                                paginate(1);
                        }
                });


        display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage = totalPages;
                                paginate(totalPages);
                        }
                });

                display.getNextButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage++;
                                paginate(currPage);
                        }
                });

                display.getPrevButton().addClickHandler(new
ClickHandler() {

                        public void onClick(ClickEvent
event) {
                                currPage--;
                                paginate(currPage);
                        }
                });


        display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        performSearch();
                                }
                        }
                });
```

```
		display.getSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				performSearch();
			}
		});


	display.getCancelSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	display.clearSearchText();
				fetchAllProjects();
			}
		});


	display.getProjsPerPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Integer items =
display.getProjsPerPageText();

	if(items!=null&&items>0){

	itemsPerPage = items;
						currPage =
1;

	performSearch();
				}else{
						Widget
source = (Widget) event.getSource();
			int left = source.getAbsoluteLeft() + 10;
			int top = source.getAbsoluteTop() + 10;

	display.projsPerPage_Warning(left, top);
				}
			}
		});
	}

	private void getUsername(){
		projService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				user = result;
				performSearch();

			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void fetchAllProjects(){
		projService.getProjects(new
AsyncCallback<ArrayList<Project>>() {

			public void
onSuccess(ArrayList<Project> result) {
				projects = result;
				currPage = 1;
				paginate(1);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void paginate(int page){
		totalPages =
(int)Math.ceil((double)projects.size()/itemsPerPage);
		ArrayList<Project> concat = new
ArrayList<Project>();
		for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(projects.size()-
1)?projects.size():itemsPerPage*page); ctr++){
			concat.add(projects.get(ctr));
		}
		display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
		getMembership(concat);
	}

	private void getMembership(final ArrayList<Project> concat) {
		projService.getMemberships(user, concat, new
AsyncCallback<ArrayList<String>>() {

			public void
onSuccess(ArrayList<String> result) {

	display.loadProjectList(concat, result);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void performSearch(){

	projService.getProjectSearchList(display.getSearchText(), new
AsyncCallback<ArrayList<Project>>() {

			public void
onSuccess(ArrayList<Project> result) {
				projects = result;
				currPage = 1;
				paginate(1);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

}
```

ViewDataEntryFormsPresenter.java

```
package cepir.client.presenter;
import java.util.ArrayList;

import cepir.client.FormService;
import cepir.client.FormServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.DataEntryForm;
import cepir.shared.Project;
import cepir.shared.Role;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
```

```
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class ViewDataEntryFormsPresenter implements Presenter {
	public interface Display{
		void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
		void loadFormsList(ArrayList<DataEntryForm>
forms, String username, Boolean isProjAdmin);
		void del_Warning(int left, int top);
		void edit_Warning(int left, int top);
		void formsPerPage_Warning(int left, int top);
		String getSearchText();
		void clearSearchText();
		Integer getFormsPerPageText();
		void clearFormsPerPageText();
		HasClickHandlers getGoToFirstPageButton();
		HasClickHandlers getGoToLastPageButton();
		HasClickHandlers getPrevButton();
		HasClickHandlers getNextButton();
		HasClickHandlers getFormsPerPageButton();
		HasClickHandlers getSearchButton();
		HasClickHandlers getCancelSearchButton();
		HasClickHandlers getAddButton();
		HasClickHandlers getEditButton();
		HasClickHandlers getDelButton();
		HasClickHandlers
getConfirmFormDeleteButton();
		HasClickHandlers getRejectFormDeleteButton();
		void showConfirmFormDeleteWarning();
		void hideConfirmFormDeleteWarning();
		HasKeyDownHandlers getSearchKeyHandler();
		HasKeyDownHandlers
getFormsPerPageKeyHandler();
		void showButtonVisibility(Boolean visible);
		Integer getSelectedRow();
		void clearRowStyles();
		FlexTable getTable();
		Widget asWidget();
	}
	private final FormServiceAsync formService =
GWT.create(FormService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private ArrayList<DataEntryForm> forms;
	private Integer projID;
	private String user;
	private Boolean isProjAdmin;
	private int itemsPerPage = 5;
	private int totalPages;
	private int currPage = 1;

	//data?p=<projID>
	public ViewDataEntryFormsPresenter(HandlerManager
eventBus, Display display, Integer projID) {
		this.eventBus = eventBus;
		this.display = display;
		this.projID = projID;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		getUsername();
		bind();
	}

	private void bind() {
		display.getTable().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Cell cell =
display.getTable().getCellForEvent(event);
				Integer selected = null;

	if(!forms.isEmpty()&&isProjAdmin){
					selected =
display.getSelectedRow();
				}

	if(cell!=null&&selected!=null){

	display.clearRowStyles();

	display.getTable().getRowFormatter().setStyleName(selected,"t
ableHighlight");
				}
			}
		});

		display.getAddButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addForm?p="+projID);
			}
		});

		display.getEditButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Integer items =
display.getSelectedRow();

	if(items!=null&&items>0){

	History.newItem("editForm?p="+projID+"&id="+forms.get(ite
ms-2).formID);
				}else{
					Widget
source = (Widget) event.getSource();
				int left = source.getAbsoluteLeft() + 10;
				int top = source.getAbsoluteTop() + 10;

	display.edit_Warning(left, top);
				}
			}
		});

		display.getDelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Integer items =
display.getSelectedRow();

	if(items!=null&&items>1){

	display.showConfirmFormDeleteWarning();
				}else{
					Widget
source = (Widget) event.getSource();
				int left = source.getAbsoluteLeft() + 10;
				int top = source.getAbsoluteTop() + 10;

	display.del_Warning(left, top);
				}
			}
		});


	display.getConfirmFormDeleteButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	display.hideConfirmFormDeleteWarning();
```

```
                                        deleteForm();
                                }
                        });


                display.getRejectFormDeleteButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {

                display.hideConfirmFormDeleteWarning();
                                }
                        });


                display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

                                public void
onKeyDown(KeyDownEvent event) {

                if(event.getNativeKeyCode()==13){

                performSearch();
                                        }
                                }
                        });

                        display.getSearchButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {
                                        performSearch();
                                }
                        });


                display.getCancelSearchButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {

                display.clearSearchText();
                                        performSearch();
                                }
                        });


                display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {
                                        currPage = 1;
                                        paginate(1);
                                }
                        });


                display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {
                                        currPage = totalPages;
                                        paginate(totalPages);
                                }
                        });

                        display.getNextButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {
                                        currPage++;
                                        paginate(currPage);
                                }
                        });

                        display.getPrevButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {
                                        currPage--;
                                        paginate(currPage);
                                }
                        });


                display.getFormsPerPageKeyHandler().addKeyDownHandler(n
ew KeyDownHandler() {

                                public void
onKeyDown(KeyDownEvent event) {

                if(event.getNativeKeyCode()==13){
                                                        Integer
items = display.getFormsPerPageText();

                if(items!=null&&items>0){

                itemsPerPage = items;

                performSearch();
                                                        }else{

                Widget source = (Widget) event.getSource();
                                                int left =
source.getAbsoluteLeft() + 10;
                                                int top =
source.getAbsoluteTop() + 10;

                display.formsPerPage_Warning(left, top);
                                                        }
                                                }
                                        }
                                });


                display.getFormsPerPageButton().addClickHandler(new
ClickHandler() {

                                public void onClick(ClickEvent
event) {
                                        Integer items =
display.getFormsPerPageText();

                if(items!=null&&items>0){

                itemsPerPage = items;

                performSearch();
                                                }else{
                                                        Widget
source = (Widget) event.getSource();
                                        int left = source.getAbsoluteLeft() + 10;
                                        int top = source.getAbsoluteTop() + 10;

                display.formsPerPage_Warning(left, top);
                                                }
                                        }
                                });
                }

                private void deleteForm(){

                formService.deleteForm(forms.get(display.getSelectedRow()-
2).formID, projID, new AsyncCallback<Void>() {

                                public void onSuccess(Void result) {
                                        performSearch();
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
```

```
                        }
                });
        }

        private void getUsername() {
                formService.getSessionKeyData("username", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {
                                user = result;
                                checkPriv();
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void checkPriv(){
                projService.getRoleInProj(user, projID, new
AsyncCallback<Role>() {

                        public void onSuccess(Role result) {

        if(result.roleID==null||result.roleName.isEmpty()){

        History.newItem("unauthorized");
                                }else{

        if(result.roleName.equalsIgnoreCase("project member")){

        isProjAdmin = false;

        display.showButtonVisibility(false);
                                                }else{

        isProjAdmin = true;

        display.showButtonVisibility(true);
                                                }

        performSearch();
                                }
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        protected void performSearch() {
                formService.getFormsSearchList(projID,
display.getSearchText(), new
AsyncCallback<ArrayList<DataEntryForm>>() {

                        public void
onSuccess(ArrayList<DataEntryForm> result) {
                                forms = result;
                                currPage = 1;
                                paginate(1);
                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void paginate(int page){
                totalPages =
(int)Math.ceil((double)forms.size()/itemsPerPage);
                ArrayList<DataEntryForm> concat = new
ArrayList<DataEntryForm>();
                for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(forms.size()-
1)?forms.size():itemsPerPage*page); ctr++){
                        concat.add(forms.get(ctr));
                }
                display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
                display.loadFormsList(concat,user,isProjAdmin);
        }

}
```

ViewMiscFilesPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import cepir.client.MiscFileService;
import cepir.client.MiscFileServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.*;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class ViewMiscFilesPresenter implements Presenter {
        public interface Display{
                void loadEverything(ArrayList<MiscFile> files,
ArrayList<MiscFileFolder> folders, String path, Integer projID);
                String getSearchTxt();
                void clearSearchTxt();
                HasClickHandlers getAddFolderButton();
                HasClickHandlers getAddFileButton();
                HasClickHandlers getAddLinkButton();
                HasClickHandlers getSearchButton();
                HasClickHandlers getCancelSearchButton();
                HasKeyDownHandlers getSearchKeyHandler();
                FlexTable getMiscFilesList();
                void setupTracker(String currLoc, List<String>
names, Integer projID);
                Widget asWidget();
        }
        private final MiscFileServiceAsync miscFileService =
GWT.create(MiscFileService.class);
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final String currLoc;
        private int type;
        private Integer fileID;
        private Integer projID;
        private ArrayList<MiscFile> files;
        private ArrayList<MiscFileFolder> folders;

        public ViewMiscFilesPresenter(HandlerManager eventBus,
Display display, String currLoc, Integer projID) {
                this.eventBus = eventBus;
                this.display = display;
                this.currLoc = currLoc;
                this.projID = projID;
        }

        public ViewMiscFilesPresenter(HandlerManager eventBus,
Display display, String currLoc, int type, Integer fileID, Integer projID) {
```

```
		// type: 1 = folder, 2 = misc file
		this.eventBus = eventBus;
		this.display = display;
		this.currLoc = currLoc;
		this.type = type;
		this.fileID = fileID;
		this.projID = projID;
		checkIfLocExists();
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		if(fileID==null){
			checkIfLocExists();
			bind();
		}
	}

	private void bind() {
		display.getMiscFilesList().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Cell cell =
display.getMiscFilesList().getCellForEvent(event);

	if(cell.getRowIndex()==2){

	History.newItem("miscFiles/"+projID+"/"+getParentPath(currL
oc));
				}
			}
		});

		display.getAddFileButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addMFile?p="+projID+"&loc="+currLoc);
			}
		});


	display.getAddFolderButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addMFolder?p="+projID+"&loc="+currLoc)
;
			}
		});

		display.getAddLinkButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addMLink?p="+projID+"&loc="+currLoc);
			}
		});

		display.getSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				performSearch();
			}
		});

	display.getCancelSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				display.clearSearchTxt();
				performSearch();
			}
		});


	display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

			public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	performSearch();
				}
			}
		});
	}

	public String getParentPath(String path){
		String[] parse = path.split("/");
		String prevPath="";
		for(int i = 0; i < parse.length-1; i++){
			prevPath += parse[i]+"/";
		}
		return prevPath;
	}

	private void checkIfLocExists() {
		miscFileService.checkIfDirExists(currLoc, new
AsyncCallback<Boolean>() {

			public void onSuccess(Boolean result)
{
				if(!result){

	History.newItem("locationError");
				}else{

	getUsername();

				}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void getUsername(){
		miscFileService.getSessionKeyData("username",
new AsyncCallback<String>() {

			public void onSuccess(String result) {

	checkIfProjMember(result);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void checkIfProjMember(String username){
		projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

			public void onFailure(Throwable
caught) {
```

```
                                        caught.printStackTrace();
                                }

                                public void onSuccess(Role result) {

        if(result.roleID==null||result.roleName.isEmpty()){

        History.newItem("unauthorized");
                                        }else{

        if(fileID!=null){

        if(type==1){

        deleteFolder();

        }else if(type==2){

        deleteFile();

        }
                                                }else{

        performSearch();
                                                }
                                        }
                                }
                        });
        }

        private void performSearch(){
                        miscFileService.getFolders(projID, currLoc,
display.getSearchTxt(), new AsyncCallback<ArrayList<MiscFileFolder>>()
{

                                public void
onSuccess(ArrayList<MiscFileFolder> result) {
                                        folders = result;
                                        getFiles();
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void getFiles(){
                        miscFileService.getFiles(projID, currLoc,
display.getSearchTxt(), new AsyncCallback<ArrayList<MiscFile>>() {

                                public void
onSuccess(ArrayList<MiscFile> result) {
                                        files = result;

        getFolderNames(Arrays.asList(currLoc.split("/")));
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void getFolderNames(List<String> ids){
                        miscFileService.getFolderNamesFromIds(ids, new
AsyncCallback<List<String>>() {

                                public void onSuccess(List<String>
result) {

        display.setupTracker(currLoc, result, projID);

        display.loadEverything(files, folders, currLoc, projID);
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void deleteFolder(){
                        miscFileService.deleteMiscFileFolder(fileID,
projID, currLoc, new AsyncCallback<Void>() {

                                public void onSuccess(Void result) {

        History.newItem("miscFiles/"+projID+"/"+currLoc);
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void deleteFile(){
                        miscFileService.deleteMiscFile(fileID, projID,
currLoc, new AsyncCallback<Void>() {

                                public void onSuccess(Void result) {

        History.newItem("miscFiles/"+projID+"/"+currLoc);
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

}
```

ViewMyProjsPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.presenter.HomeForMemberPresenter.Display;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ViewMyProjsPresenter implements Presenter {
        public interface Display{
                void setupPaginator(int currPage,int totalPage,
boolean isFirstPage, boolean isLastPage);
                void loadProjsList(ArrayList<Project> projects);
                Integer getProjsPerPageText();
                void clearProjsPerPageText();
                void usersPerPage_Warning(int left, int top);
                HasClickHandlers getGoToFirstPageButton();
                HasClickHandlers getGoToLastPageButton();
                HasClickHandlers getPrevButton();
                HasClickHandlers getNextButton();
                HasClickHandlers getProjsPerPageButton();
                Widget asWidget();
        }
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private ArrayList<Project> projects;
        private String user;
        private int itemsPerPage = 5;
```

```
	private int totalPages;
	private int currPage = 1;

	public ViewMyProjsPresenter(HandlerManager
eventBus,Display display) {
		this.eventBus = eventBus;
		this.display = display;
	}

	public void go(HasWidgets container) {
		bind();
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		getUsername();
	}

	private void getUsername(){
		projService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				user = result;
				fetchMyProjects();
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void bind() {

	display.getGoToFirstPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage = 1;
				paginate(1);
			}
		});


	display.getGoToLastPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage = totalPages;
				paginate(totalPages);
			}
		});

		display.getNextButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage++;
				paginate(currPage);
			}
		});

		display.getPrevButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				currPage--;
				paginate(currPage);
			}
		});


	display.getProjsPerPageButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Integer items =
display.getProjsPerPageText();

	if(items!=null&&items>0){

	itemsPerPage = items;
					currPage =
1;

	fetchMyProjects();
				}else{
					Widget
source = (Widget) event.getSource();
				int left = source.getAbsoluteLeft() + 10;
				int top = source.getAbsoluteTop() + 10;

	display.usersPerPage_Warning(left, top);
				}
			}
		});
	}

	private void fetchMyProjects() {
		projService.getProjects(user, new
AsyncCallback<ArrayList<Project>>() {

			public void
onSuccess(ArrayList<Project> result) {
				projects = result;
				paginate(1);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void paginate(int page){
		totalPages =
(int)Math.ceil((double)projects.size()/itemsPerPage);
		ArrayList<Project> concat = new
ArrayList<Project>();
		for(int ctr = itemsPerPage*(page-1);
ctr<((itemsPerPage*page)-1>(projects.size()-
1)?projects.size():itemsPerPage*page); ctr++){
			concat.add(projects.get(ctr));
		}
		display.setupPaginator(page, totalPages,
(page==1?true:false), (page==totalPages?true:false));
		display.loadProjsList(concat);
	}
}
```

ViewProjMemRequestPresenter.java

```
package cepir.client.presenter;

import cepir.client.MailService;
import cepir.client.MailServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.ProjMemRequest;
import cepir.shared.Role;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ViewProjMemRequestPresenter implements Presenter {
	public interface Display{
```

```
		void setUsernameTxt(String text);
		void setProjTxt(String text);
		void setMessageTxt(String text);
		Integer setSelectedMemberType();
		HasClickHandlers getApproveButton();
		HasClickHandlers getRejectButton();
		Widget asWidget();
	}
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final String username;
	private final Integer projID;
	private String email;
	private String projName;

	public ViewProjMemRequestPresenter(HandlerManager
eventBus, Display display, String username, Integer projID) {
		this.eventBus = eventBus;
		this.display = display;
		this.username = username;
		this.projID = projID;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		bind();
		getUsername();
	}

	private void getUsername(){
		projService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				checkPriv(result);
			}

			public void onFailure(Throwable
caught) {

			}
		});
	}

	private void checkPriv(String currUser){
		projService.getRoleInProj(currUser, projID, new
AsyncCallback<Role>() {

			public void onSuccess(Role result) {

	if(result.roleID==null||result.roleName.isEmpty()||result.roleNa
me.equalsIgnoreCase("project member")){

	History.newItem("unauthorized");
					}else{

	setupProjMemRequest();
					}
			}

			public void onFailure(Throwable
caught) {

			}
		});
	}

	private void setupProjMemRequest() {
		projService.getProjMemRequest(username,
projID, new AsyncCallback<ProjMemRequest>() {

			public void
onSuccess(ProjMemRequest result) {

	display.setUsernameTxt(result.projMemRequestPK.username.u
sername);
					String[] parse =
result.projMemRequestPK.projID.projName.split("=");

	display.setProjTxt(parse[0]);

	display.setMessageTxt(result.message);
					email =
result.projMemRequestPK.username.email;
					projName = parse[0];
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
	}

	private void reloadTab(){
		History.newItem("allPendingReq");
	}

	private void bind() {
		display.getApproveButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				approveReq();
			}
		});

		display.getRejectButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				rejectReq();
			}
		});
	}

	private void approveReq(){
		projService.approveMemRequest(username,
projID, display.setSelectedMemberType()+1, false, new
AsyncCallback<Void>() {

			public void onSuccess(Void result) {
				sendApprovedMail();
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void rejectReq(){
		projService.deleteMemRequest(username, projID,
new AsyncCallback<Void>() {

			public void onSuccess(Void result) {
				sendRejectedMail();

			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void sendApprovedMail(){
		MailServiceAsync mail =
GWT.create(MailService.class);
```

```
                String APPROVED_SUBJ = "Project Membership
Request Approved - Collaboratory for Epidemiological Research (CEpiR)";
                String APPROVED_MSG = "Your request for
membership in project "+projName.toUpperCase()
                +" at the Collaboratory for Epidemiological
Research (CEpiR) has been approved. \n\n" + "NOTE: This message has
been generated automatically. Please do not reply to this message.";
                mail.postMail(new String[]{email},
APPROVED_SUBJ, APPROVED_MSG, new AsyncCallback<Boolean>()
{

                        public void onSuccess(Boolean result)
{

                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

        private void sendRejectedMail(){
                MailServiceAsync mail =
GWT.create(MailService.class);
                String UNAPPROVED_SUBJ = "Project
Membership Request Unapproved - Collaboratory for Epidemiological
Research (CEpiR)";
                String UNAPPROVED_MSG = "Your request for
membership in project "+projName.toUpperCase()
                +" at the Collaboratory for Epidemiological
Research (CEpiR) has been rejected. \n\n" + "NOTE: This message has been
generated automatically. Please do not reply to this message.";
                mail.postMail(new String[]{email},
UNAPPROVED_SUBJ, UNAPPROVED_MSG, new
AsyncCallback<Boolean>() {

                        public void onSuccess(Boolean result)
{

                        }

                        public void onFailure(Throwable
caught) {
                                caught.printStackTrace();
                        }
                });
        }

}
```

ViewProjPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;
import java.util.Collections;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.client.UserServiceAsync;
import cepir.client.view.MainView;
import cepir.client.view.UnauthorizedNoticeView;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ViewProjPresenter implements Presenter {
        public interface Display{
                Widget asWidget();
                void setHeader(String header);
                void setProjName(String projName);
                void setProjDesc(String desc);
                void setDateCreated(String date);
                void setDateUpdated(String date);
                void setErrorText(String err);
                void setProjNameErrorText(String err);
                void setSearchTxt(String text);
                void setProjAdminErrorText(String err);
                String getProjName();
                String getProjDesc();
                String getSearchTxt();
                void loadUsersList(ArrayList<User> users);
                void loadProjAdminsList(ArrayList<User> users);
                ArrayList<Integer> getSelectedAdmins();
                ArrayList<Integer> getSelectedUsers();
                HasClickHandlers getSaveButton();
                HasClickHandlers getCancelButton();
                HasClickHandlers getSearchButton();
                HasClickHandlers getCancelSearchButton();
                HasClickHandlers getChooseAdmins();
                HasClickHandlers getSelectOkButton();
                HasClickHandlers getSelectAdminButton();
                HasClickHandlers getDeselectAdminButton();
                HasKeyDownHandlers getSearchKeyHandler();
                void showProjAdminSelect();
                void hideProjAdminSelect();
        }
        private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
        private final UserServiceAsync userService =
GWT.create(UserService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private final Integer projID;
        private ArrayList<String> projAdmins;
        private ArrayList<User> users;
        private ArrayList<User> selected;
        private ArrayList<User> unselected;

        public ViewProjPresenter(HandlerManager eventBus, Display
view, Integer projID) {
                this.eventBus = eventBus;
                this.display = view;
                this.projID = projID;
        }

        public ViewProjPresenter(HandlerManager eventBus, Display
view, Integer projID, Boolean isMember) {
                this.eventBus = eventBus;
                this.display = view;
                this.projID = projID;
        }

        public void go(HasWidgets container) {
                bind();
                if(container!=null){
                        container.clear();
                        container.add(display.asWidget());
                }
                checkIfSysAd();
        }

        private void checkIfSysAd(){
                projService.getSessionKeyData("isSysAd", new
AsyncCallback<String>() {

                        public void onSuccess(String result) {

        if(result.equalsIgnoreCase("yes")){

        loadEverything();
                                        }else{

        checkIfMemberCanEdit();
                                        }
```

```
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }

        private void checkIfMemberCanEdit(){
                projService.getSessionKeyData("username", new
AsyncCallback<String>() {

                                        public void onSuccess(String result) {

        checkWhereProjAdmin(result);
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }

        private void checkWhereProjAdmin(String username){
                projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

                                        public void onSuccess(Role result) {

        if(result.roleID==null||result.roleName.isEmpty()){

        History.newItem("unauthorized");
                                                }else{

        loadEverything();
                                                }
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }

        private void loadEverything(){
                lookupProjAdminList();
                if(projID!=null){
                        display.setHeader("Edit project");
                        loadProj();
                }else{
                        display.setHeader("Add project");
                        display.setProjName("");
                }
        }

        private void loadProj(){
                projService.getProject(projID, new
AsyncCallback<Project>() {

                                        public void onSuccess(Project result)
{

        display.setProjName(result.projName);

        display.setProjDesc(result.projDesc);

        display.setDateCreated(result.dateCreated.toString());

        display.setDateUpdated(result.dateUpdated.toString());
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }

        private void lookupProjAdminList() {
                userService.getUsersList(new
AsyncCallback<ArrayList<User>>() {

                                        public void
onSuccess(ArrayList<User> result) {
                                                users = result;
                                                selectProjAdmins();
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
        }

        private void bind() {
                display.getSaveButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                saveProject();
                                        }
                                });

                display.getCancelButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                reloadProjTab();
                                        }
                                });

                display.getChooseAdmins().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {

        display.showProjAdminSelect();
                                                loadLists();
                                                display.setSearchTxt("");
                                                unselected.clear();
                                                unselected.addAll(users);
                                                if(!selected.isEmpty()){

        unselected.removeAll(selected);
                                                }

        display.loadProjAdminsList(selected);

        display.loadUsersList(unselected);
                                        }
                                });


        display.getSelectAdminButton().addClickHandler(new
ClickHandler() {

                                        public void onClick(ClickEvent
event) {
                                                for(int row = 0; row <
display.getSelectedUsers().size(); row++){

        selected.add(unselected.get(display.getSelectedUsers().get(row)
-row));

        unselected.remove(unselected.get(display.getSelectedUsers().ge
t(row)-row));
                                                }

        Collections.sort(selected);

        Collections.sort(unselected);
```

```
display.loadProjAdminsList(selected);

display.loadUsersList(unselected);
                }
        });


display.getDeselectAdminButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        for(int row = 0; row <
display.getSelectedAdmins().size(); row++){

unselected.add(selected.get(display.getSelectedAdmins().get(ro
w)-row));

selected.remove(selected.get(display.getSelectedAdmins().get(r
ow)-row));
                        }

Collections.sort(selected);

Collections.sort(unselected);

display.loadProjAdminsList(selected);

display.loadUsersList(unselected);
                }
        });

                display.getSelectOkButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

display.hideProjAdminSelect();
                        if(projID!=null){

saveProjAdmins();
                        }
                }
        });


display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

                public void
onKeyDown(KeyDownEvent event) {

if(event.getNativeKeyCode()==13){

search(display.getSearchTxt());
                        }
                }
        });

                display.getSearchButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {

search(display.getSearchTxt());
                }
        });


display.getCancelSearchButton().addClickHandler(new
ClickHandler() {

                public void onClick(ClickEvent
event) {
                        display.setSearchTxt("");
                        unselected.clear();
                        unselected.addAll(users);
                        if(!selected.isEmpty()){

unselected.removeAll(selected);
                        }

display.loadProjAdminsList(selected);

display.loadUsersList(unselected);
                }
        });
}

private void search(String query){
        unselected.clear();
        unselected.addAll(users);
        unselected.removeAll(selected);
        for(int i = 0; i < unselected.size(); i++){
                User user = unselected.get(i);

if(user.username.toLowerCase().contains(query.toLowerCase())
||user.firstName.toLowerCase().contains(query.toLowerCase())

||user.lastName.toLowerCase().contains(query.toLowerCase()))
{
                }else{
                        i--;
                        unselected.remove(user);
                }
        }
        display.loadUsersList(unselected);
}

private void loadLists(){
        if(selected==null&&unselected==null){
                unselected = new ArrayList<User>();
                selected = new ArrayList<User>();
                for(int ctr = 0; ctr < users.size();
ctr++){
                        for(int ctr2 = 0; ctr2 <
projAdmins.size(); ctr2++){

if(searchRowNo(projAdmins.get(ctr2))==ctr){

selected.add(users.get(ctr));
                                }
                        }
                }
                unselected.addAll(users);
                if(!selected.isEmpty()){

unselected.removeAll(selected);
                }
        }
}

private void selectProjAdmins(){
        projService.getProjAdmins(projID, new
AsyncCallback<ArrayList<User>>() {

                public void
onSuccess(ArrayList<User> result) {
                        projAdmins = new
ArrayList<String>();
                        for(int i = 0; i <
result.size(); i++){

projAdmins.add(result.get(i).username);
                        }
                        loadLists();

display.loadProjAdminsList(selected);

display.loadUsersList(unselected);
                }

                public void onFailure(Throwable
caught) {
                        caught.printStackTrace();
                }
        });
```

```
	}

	private Integer searchRowNo(String username){
		for(int ctr = 0;ctr<users.size(); ctr++){

	if(users.get(ctr).getUsername().equalsIgnoreCase(username)){
					return ctr;
				}
			}
			return null;
	}

	private void reloadProjTab(){
			History.back();
	}

	private void checkIfProjNameNotUsed(){

	projService.projNameNotUsed(display.getProjName(), projID,
new AsyncCallback<Boolean>() {

				public void onSuccess(Boolean result)
{
					if(result==false){

	display.setProjNameErrorText("Project name already in use");
					}
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
	}

	private void saveProjAdmins(){
			if (selected.isEmpty()){
				display.setErrorText("Please select
project administrator/s");
			}else{
				ArrayList<String> newAdmins = new
ArrayList<String>();
				for(int i = 0; i < selected.size(); i++){

	newAdmins.add(selected.get(i).username);
				}
				projService.addProjMembers(projID,
newAdmins, true, "Project Administrator", new AsyncCallback<String>() {

					public void
onSuccess(String result) {

	if(result.equals("ok")){

	display.hideProjAdminSelect();
						}else{

	display.setProjAdminErrorText(result);
						}
					}

					public void
onFailure(Throwable caught) {

	caught.printStackTrace();
					}
				});
			}
	}

	private void saveProject(){

	if(display.getProjName().isEmpty()||display.getProjDesc().isEm
pty()){
				display.setErrorText("Required field/s
missing");
			}else if(selected.isEmpty()){
				display.setErrorText("Please select
project administrator/s");
			}else{
				ArrayList<String> newAdmins = new
ArrayList<String>();
				checkIfProjNameNotUsed();
				if(projID==null){
					for(int i = 0 ; i <
selected.size(); i++){

	newAdmins.add(selected.get(i).username);
					}
				}
				Project proj = new Project();
				proj.projID = projID;
				proj.projName =
display.getProjName();
				proj.projDesc =
display.getProjDesc().trim();
				projService.addProject(proj,
newAdmins, (projID==null?false:true),new AsyncCallback<String>() {

					public void
onSuccess(String result) {

	if(result.equals("ok")){

	reloadProjTab();
					}else{

	display.setErrorText(result);
					}
				}

				public void
onFailure(Throwable caught) {

	caught.printStackTrace();
				}
			});
		}
	}

}
```

ViewReferencePresenter.java

```
package cepir.client.presenter;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.ReferenceService;
import cepir.client.ReferenceServiceAsync;
import cepir.shared.Reference;
import cepir.shared.ReferenceType;
import cepir.shared.Role;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ViewReferencePresenter implements Presenter {
	public interface Display{
		void hideAllFields();
		void showBook();
		void showBookChap();
		void showJournal();
		void showEArticleOrUnpublishedWork();
		void showThesis();
		void loadReference(Reference ref);
		Widget asWidget();
	}
	private final ReferenceServiceAsync refService =
GWT.create(ReferenceService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final Integer refID;
	private final Integer projID;
```

```
	public ViewReferencePresenter(HandlerManager eventBus,
Display display,
				Integer refID, Integer projID) {
			this.eventBus = eventBus;
			this.display = display;
			this.refID = refID;
			this.projID = projID;
		}

		public void go(HasWidgets container) {
			if(container!=null){
				container.clear();
				container.add(display.asWidget());
			}
			getUsername();
		}

		private void getUsername(){
			refService.getSessionKeyData("username", new
AsyncCallback<String>() {

				public void onSuccess(String result) {
					checkPriv(result);
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void loadFields(ReferenceType type) {

		if(type.equals(ReferenceType.Book)||type.equals(ReferenceTyp
e.Electronic_Book)){
				display.showBook();
			}else
if(type.equals(ReferenceType.Book_Chapter)){
				display.showBookChap();
			}else
if(type.equals(ReferenceType.Journal_Article)){
				display.showJournal();
			}else
if(type.equals(ReferenceType.Electronic_Article)||type.equals(ReferenceTyp
e.Unpublished_Work)){

		display.showEArticleOrUnpublishedWork();
			}else if(type.equals(ReferenceType.Thesis)){
				display.showThesis();
			}
		}

		protected void checkPriv(String username) {
			projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}

				public void onSuccess(Role result) {

		if(result.roleID==null||result.roleName.isEmpty()){

		History.newItem("unauthorized");
					}else{

		refService.getReference(refID, new
AsyncCallback<Reference>() {


		public void onSuccess(Reference result) {

		display.loadReference(result);

		loadFields(result.type);
		}


		public void onFailure(Throwable caught) {

		caught.printStackTrace();

		}
					});
				}
			});
		}
	}
```

ViewReferencesPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import cepir.client.ReferenceService;
import cepir.client.ReferenceServiceAsync;
import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.shared.Reference;
import cepir.shared.ReferenceFolder;
import cepir.shared.Role;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;

public class ViewReferencesPresenter implements Presenter {
	public interface Display{
		void loadEverything(ArrayList<Reference> files,
ArrayList<ReferenceFolder> folders, String path, Integer projID);
		String getSearchTxt();
		void clearSearchTxt();
		HasClickHandlers getAddFolderButton();
		HasClickHandlers getAddFileButton();
		HasClickHandlers getSearchButton();
		HasClickHandlers getCancelSearchButton();
		HasKeyDownHandlers getSearchKeyHandler();
		FlexTable getReferencesList();
		void setupTracker(String currLoc, List<String>
names, Integer projID);
		Widget asWidget();
	}
	private final ReferenceServiceAsync refService =
GWT.create(ReferenceService.class);
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private final String currLoc;
	private int type;
	private Integer fileID;
	private Integer projID;
	private ArrayList<Reference> files;
	private ArrayList<ReferenceFolder> folders;

	public ViewReferencesPresenter(HandlerManager eventBus,
Display display, String currLoc, Integer projID) {
		this.eventBus = eventBus;
		this.display = display;
```

```
		this.currLoc = currLoc;
		this.projID = projID;
	}

	public ViewReferencesPresenter(HandlerManager eventBus,
Display display, String currLoc, int type, Integer fileID, Integer projID) {
		// type: 1 = folder, 2 = reference
		this.eventBus = eventBus;
		this.display = display;
		this.currLoc = currLoc;
		this.type = type;
		this.fileID = fileID;
		this.projID = projID;
		checkIfLocExists();
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		if(fileID==null){
			checkIfLocExists();
			bind();
		}
	}

	private void bind() {
		display.getReferencesList().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Cell cell =
display.getReferencesList().getCellForEvent(event);

	if(cell.getRowIndex()==2){

	History.newItem("refs/"+projID+"/"+getParentPath(currLoc));
				}
			}
		});

		display.getAddFileButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addRFile?p="+projID+"&loc="+currLoc);
			}
		});


	display.getAddFolderButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addRFolder?p="+projID+"&loc="+currLoc);
			}
		});


		display.getSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				performSearch();
			}
		});


	display.getCancelSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				display.clearSearchTxt();
				performSearch();
			}
		});


	display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {

			public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	performSearch();
				}
			}
		});
	}

	public String getParentPath(String path){
		String[] parse = path.split("/");
		String prevPath="";
		for(int i = 0; i < parse.length-1; i++){
			prevPath += parse[i]+"/";
		}
		return prevPath;
	}

	private void checkIfLocExists() {
		refService.checkIfDirExists(currLoc, new
AsyncCallback<Boolean>() {

			public void onSuccess(Boolean result)
{
				if(!result){

	History.newItem("locationError");
				}else{

	getUsername();

				}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void getUsername(){
		refService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {

	checkIfProjMember(result);
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void checkIfProjMember(String username){
		projService.getRoleInProj(username, projID, new
AsyncCallback<Role>() {

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}

			public void onSuccess(Role result) {

	if(result.roleID==null||result.roleName.isEmpty()){
```

```
History.newItem("unauthorized");
                                        }else{

        if(fileID!=null){

        if(type==1){

        deleteFolder();

        }else if(type==2){

        deleteFile();

        }
                                                }else{

        performSearch();
                                                }
                                        }
                                }
                        });
        }

        private void performSearch(){
                refService.getFolders(projID, currLoc,
display.getSearchTxt(), new
AsyncCallback<ArrayList<ReferenceFolder>>() {

                                public void
onSuccess(ArrayList<ReferenceFolder> result) {
                                        folders = result;
                                        getFiles();
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void getFiles(){
                refService.getFiles(projID, currLoc,
display.getSearchTxt(), new AsyncCallback<ArrayList<Reference>>() {

                                public void
onSuccess(ArrayList<Reference> result) {
                                        files = result;

        getFolderNames(Arrays.asList(currLoc.split("/")));
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void getFolderNames(List<String> ids){
                refService.getFolderNamesFromIds(ids, new
AsyncCallback<List<String>>() {

                                public void onSuccess(List<String>
result) {

        display.setupTracker(currLoc, result, projID);

        display.loadEverything(files, folders, currLoc, projID);
                                }

                                public void onFailure(Throwable
caught) {
                                        caught.printStackTrace();
                                }
                        });
        }

        private void deleteFolder(){
                        refService.deleteReferenceFolder(fileID, projID,
currLoc, new AsyncCallback<Void>() {

                                        public void onSuccess(Void result) {

        History.newItem("refs/"+projID+"/"+currLoc);
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                        }

                        private void deleteFile(){
                                refService.deleteReference(fileID, projID,
currLoc, new AsyncCallback<Void>() {

                                        public void onSuccess(Void result) {

                        History.newItem("refs/"+projID+"/"+currLoc);
                                        }

                                        public void onFailure(Throwable
caught) {
                                                caught.printStackTrace();
                                        }
                                });
                        }

}
```

ViewToolFolderPresenter.java

```
package cepir.client.presenter;

import cepir.client.InputChecker;
import cepir.client.ToolService;
import cepir.client.ToolServiceAsync;
import cepir.shared.ToolFolder;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;

public class ViewToolFolderPresenter implements Presenter {
        public interface Display{
                void setHeader(String text);
                void setName(String text);
                String getName();
                void setDesc(String text);
                String getDesc();
                void setErrorTxt(String text);
                HasClickHandlers getSaveButton();
                HasClickHandlers getCancelButton();
                Widget asWidget();
        }
        private final ToolServiceAsync toolService =
GWT.create(ToolService.class);
        private final HandlerManager eventBus;
        private final Display display;
        private Integer toolFolderID;
        private String folderPath;
        private String user;

        public ViewToolFolderPresenter(HandlerManager eventBus,
Display display, Integer toolFolderID) {
                this.eventBus = eventBus;
                this.display = display;
                this.toolFolderID = toolFolderID;
        }
```

```
	public ViewToolFolderPresenter(HandlerManager eventBus,
Display display, String folderPath) {
		this.eventBus = eventBus;
		this.display = display;
		this.folderPath = folderPath;
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		getUsername();
		bind();
	}

	private void bind() {
		display.getSaveButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				saveFolder();
			}
		});

		display.getCancelButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				History.back();
			}
		});
	}

	private void getUsername() {
		toolService.getSessionKeyData("username", new
AsyncCallback<String>() {

			public void onSuccess(String result) {
				user = result;
				if(toolFolderID!=null){

	loadFolderDetails();
				}else{

	display.setHeader("Add folder");
				}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	private void loadFolderDetails(){
		toolService.getFolder(toolFolderID, new
AsyncCallback<ToolFolder>() {

			public void onSuccess(ToolFolder
result) {
				if(result!=null){
					folderPath =
result.toolFolderPath;

	display.setHeader("Edit folder");

	display.setName(result.toolFolderName);

	display.setDesc(result.toolFolderDesc);
				}else{

	History.newItem("locationError");
				}
			}

			public void onFailure(Throwable
caught) {
				caught.printStackTrace();
			}
		});
	}

	public void saveFolder(){

	if(display.getName().isEmpty()||display.getDesc().isEmpty()){
		display.setErrorTxt("Required field/s
missing");
		}else{
			ToolFolder folder = new
ToolFolder();
			folder.toolFolderID = toolFolderID;
			folder.toolFolderName =
display.getName();
			folder.toolFolderDesc =
display.getDesc();
			folder.toolFolderPath = folderPath;
			User u = new User();
			u.username = user;
			folder.creator = u;
			toolService.saveDirectory(folder,
(toolFolderID==null?false:true),new AsyncCallback<String>() {

				public void
onSuccess(String result) {

	if(result.equals("ok")){

	History.newItem("tools/"+folderPath);
					}else{

	display.setErrorTxt(result);
					}
				}

				public void
onFailure(Throwable caught) {

	caught.printStackTrace();
				}
			});
		}
	}

}
```

ViewToolsPresenter.java

```
package cepir.client.presenter;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.ToolService;
import cepir.client.ToolServiceAsync;
import cepir.shared.Role;
import cepir.shared.Tool;
import cepir.shared.ToolFolder;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasWidgets;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HTMLTable.Cell;
```

```
public class ViewToolsPresenter implements Presenter{
	public interface Display{
		void loadEverything(ArrayList<ToolFolder>
folders, ArrayList<Tool> tools, String path, String username);
		String getSearchTxt();
		void clearSearchTxt();
		HasClickHandlers getAddFolderButton();
		HasClickHandlers getAddFileButton();
		HasClickHandlers getAddLinkButton();
		HasClickHandlers getSearchButton();
		HasClickHandlers getCancelSearchButton();
		HasKeyDownHandlers getSearchKeyHandler();
		void setButtonsVisibility(Boolean isVisible);
		FlexTable getToolsList();
		void setupTracker(String currLoc, List<String>
names);
		Widget asWidget();
	}
	private final ProjectServiceAsync projService =
GWT.create(ProjectService.class);
	private final ToolServiceAsync toolService =
GWT.create(ToolService.class);
	private final HandlerManager eventBus;
	private final Display display;
	private String isToolAd;
	private final String currLoc;
	private Integer toolFolderID;
	private Integer type;
	private ArrayList<ToolFolder> folders;
	private ArrayList<Tool> tools;

	public ViewToolsPresenter(HandlerManager eventBus, Display
display, String currLoc) {
		this.eventBus = eventBus;
		this.display = display;
		this.currLoc = currLoc;
	}

	//for delete purposes
	public ViewToolsPresenter(HandlerManager eventBus, Display
display, Integer toolFolderID, String currLoc, Integer type) {
		// type: 1 = folder, 2 = tool
		this.eventBus = eventBus;
		this.display = display;
		this.toolFolderID = toolFolderID;
		this.currLoc = currLoc;
		this.type = type;
		checkIfToolAd();
	}

	public void go(HasWidgets container) {
		if(container!=null){
			container.clear();
			container.add(display.asWidget());
		}
		if(toolFolderID==null){
			checkIfLocExists();
			bind();
		}
	}

	private void bind() {
		display.getToolsList().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				Cell cell =
display.getToolsList().getCellForEvent(event);

	if(cell.getRowIndex()==2){

	History.newItem("tools/"+getParentPath(currLoc));
				}
			}
		});


	display.getSearchKeyHandler().addKeyDownHandler(new
KeyDownHandler() {
		public void
onKeyDown(KeyDownEvent event) {

	if(event.getNativeKeyCode()==13){

	performSearch();
				}
			}
		});


	display.getAddFolderButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addTFolder?loc="+currLoc);
			}
		});

		display.getAddLinkButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addLink?loc="+currLoc);
			}
		});

		display.getAddFileButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {

	History.newItem("addTool?loc="+currLoc);
			}
		});

		display.getSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				performSearch();
			}
		});


	display.getCancelSearchButton().addClickHandler(new
ClickHandler() {

			public void onClick(ClickEvent
event) {
				display.clearSearchTxt();
				performSearch();
			}
		});
	}

	public String getParentPath(String path){
		String[] parse = path.split("/");
		String prevPath="";
		for(int i = 0; i < parse.length-1; i++){
			prevPath += parse[i]+"/";
		}
		return prevPath;
	}

	private void checkIfLocExists(){
		toolService.checkIfDirExists(currLoc, new
AsyncCallback<Boolean>() {

			public void onSuccess(Boolean result)
{
				if(result){
```

```
		checkIfToolAd();
					}else{

		History.newItem("locationError");
					}
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void checkIfToolAd(){
			projService.getSessionKeyData("isToolAd", new
AsyncCallback<String>() {

				public void onSuccess(String result) {
					isToolAd = result;
					if(toolFolderID==null){

		if(result.equals("Yes")){

		display.setButtonsVisibility(true);
						}

		performSearch();
					}else{

		if(result.equals("Yes")){

		if(type==1){

		deleteFolder();

		}else if(type==2){

		deleteLink();

		}
						}else{

		History.newItem("unauthorized");
						}
					}
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void performSearch(){
			toolService.getFolders(currLoc,
display.getSearchTxt(), new AsyncCallback<ArrayList<ToolFolder>>() {

				public void
onSuccess(ArrayList<ToolFolder> result) {
					folders = result;
					getTools();
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void getTools(){
			toolService.getFiles(currLoc,
display.getSearchTxt(), new AsyncCallback<ArrayList<Tool>>() {

				public void
onSuccess(ArrayList<Tool> result) {
					tools = result;
		getFolderNames(Arrays.asList(currLoc.split("/")));
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void getFolderNames(List<String> ids){
			toolService.getFolderNamesFromIds(ids, new
AsyncCallback<List<String>>() {

				public void onSuccess(List<String>
result) {

		display.setupTracker(currLoc,result);

		display.loadEverything(folders, tools, currLoc, isToolAd);
				}

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}
			});
		}

		private void deleteFolder(){
			toolService.deleteToolFolder(toolFolderID,
currLoc, new AsyncCallback<Void>() {

				public void onSuccess(Void result) {

		History.newItem("tools/"+currLoc);
				}

				public void onFailure(Throwable
caught) {

				}
			});
		}

		private void deleteLink(){
			toolService.deleteTool(toolFolderID, currLoc, new
AsyncCallback<Void>() {

				public void onFailure(Throwable
caught) {
					caught.printStackTrace();
				}

				public void onSuccess(Void result) {

		History.newItem("tools/"+currLoc);
				}
			});
		}
}
```

AddDataEntryFormView.java

```
package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.*;

public class AddDataEntryFormView extends Composite implements
cepir.client.presenter.AddDataEntryFormPresenter.Display{
		private Label lblHeader;
		private TextBox nameTxt;
		private TextArea descTxt;
		private Button saveButton;
		private Button cancelButton;
		private Label errorTxt;

		public AddDataEntryFormView() {
			FlexTable mainFlexTable = new FlexTable();
```

```
		initWidget(mainFlexTable);

		lblHeader = new Label();
		lblHeader.setStyleName("header2");
		mainFlexTable.setWidget(0, 0, lblHeader);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

		mainFlexTable.setText(1, 0, "Form Name");

	mainFlexTable.getFlexCellFormatter().setWidth(1,0,"120px");

		nameTxt = new TextBox();
		nameTxt.setWidth("250px");
		mainFlexTable.setWidget(1, 1, nameTxt);

		mainFlexTable.setText(2, 0, "Description");

	mainFlexTable.getFlexCellFormatter().setWidth(2,0,"120px");

		descTxt = new TextArea();
		descTxt.setWidth("250px");
		descTxt.setVisibleLines(5);
		mainFlexTable.setWidget(2, 1, descTxt);

		FlexTable buttons = new FlexTable();
		mainFlexTable.setWidget(3, 1, buttons);

		saveButton = new Button("Save");
		buttons.setWidget(0, 0, saveButton);

		cancelButton = new Button("Cancel");
		buttons.setWidget(0, 1, cancelButton);

		errorTxt = new Label();
		errorTxt.setStyleName("errorMsg");
		mainFlexTable.setWidget(4, 1, errorTxt);
	}

	public void setHeader(String text){
		lblHeader.setText(text);
	}

	public void setName(String text){
		nameTxt.setText(text);
	}

	public String getName(){
		return nameTxt.getText();
	}

	public void setDesc(String text){
		descTxt.setText(text);
	}

	public String getDesc(){
		return descTxt.getText();
	}

	public HasClickHandlers getSaveButton(){
		return saveButton;
	}

	public HasClickHandlers getCancelButton(){
		return cancelButton;
	}

	public void setErrorTxt(String text){
		errorTxt.setText(text);
	}

	public Widget asWidget(){
		return this;
	}

}
```

AddMiscFileFolderView.java

```
package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.Button;

public class AddMiscFileFolderView extends Composite implements
cepir.client.presenter.AddMiscFileFolderPresenter.Display{
	private TextBox nameTxt;
	private TextArea descTxt;
	private Button saveButton;
	private Button cancelButton;
	private Label lblErrorTxt;
	private Label lblHeader;

	public AddMiscFileFolderView() {

		FlexTable flexTable = new FlexTable();
		initWidget(flexTable);

		lblHeader = new Label("");
		lblHeader.setStyleName("header2");
		flexTable.setWidget(0, 0, lblHeader);

		Label lblFolderName = new Label("Folder
Name");
		lblFolderName.setWidth("120px");
		flexTable.setWidget(1, 0, lblFolderName);

		nameTxt = new TextBox();
		nameTxt.setWidth("250px");
		flexTable.setWidget(1, 1, nameTxt);

		Label lblDescription = new Label("Description");
		flexTable.setWidget(2, 0, lblDescription);
		flexTable.getFlexCellFormatter().setColSpan(0, 0,
2);

		descTxt = new TextArea();
		descTxt.setWidth("250px");
		flexTable.setWidget(2, 1, descTxt);

		FlexTable buttonTable = new FlexTable();
		flexTable.setWidget(3, 1, buttonTable);

		saveButton = new Button("New button");
		saveButton.setText("Save");
		buttonTable.setWidget(0, 0, saveButton);

		cancelButton = new Button("New button");
		cancelButton.setText("Cancel");
		buttonTable.setWidget(0, 1, cancelButton);

		lblErrorTxt = new Label("");
		lblErrorTxt.setStyleName("errorMsg");
		flexTable.setWidget(4, 1, lblErrorTxt);
	}

	public void setHeader(String text){
		lblHeader.setText(text);
	}

	public void setDesc(String text){
		descTxt.setText(text);
	}

	public void setName(String text){
		nameTxt.setText(text);
	}

	public String getName(){
		return nameTxt.getText();
	}

	public String getDesc(){
		return descTxt.getText();
	}
```

```
	public void setErrorTxt(String text){
		lblErrorTxt.setText(text);
	}

	public HasClickHandlers getSaveButton(){
		return saveButton;
	}
	public HasClickHandlers getCancelButton(){
		return cancelButton;
	}


}
```

AddMiscFileView.java

```
package cepir.client.view;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Hidden;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.DeckPanel;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.FileUpload;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Widget;

public class AddMiscFileView extends Composite implements
cepir.client.presenter.AddMiscFilePresenter.Display{
	private final String UPLOAD_ACTION_URL =
GWT.getModuleBaseURL() + "upload";

	private FlexTable mainFlexTable;
	private FormPanel form;
	private Label lblHeader;
	private TextArea descTxt;
	private TextBox linkTxt;
	private Button saveButton;
	private Button cancelButton;
	private Label lblErrorTxt;
	private FileUpload fileUpload;
	private TextBox titleTxt;
	private Hidden currLoc;
	private Hidden fileName;

	public AddMiscFileView() {

		form = new FormPanel();

	form.setEncoding(FormPanel.ENCODING_MULTIPART);
		form.setMethod(FormPanel.METHOD_POST);
		form.setAction(UPLOAD_ACTION_URL);
		initWidget(form);

		mainFlexTable = new FlexTable();
		form.setWidget(mainFlexTable);

		lblHeader = new Label("");
		lblHeader.setStyleName("header2");
		mainFlexTable.setWidget(0, 0, lblHeader);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

		currLoc = new Hidden("currLoc");
		mainFlexTable.setWidget(0, 2, currLoc);

		fileName = new Hidden("name");
		mainFlexTable.setWidget(0, 3, fileName);

		Label lblFile = new Label("File");
		mainFlexTable.setWidget(1, 0, lblFile);

		fileUpload = new FileUpload();
		fileUpload.setName("file");
		mainFlexTable.setWidget(1, 1, fileUpload);

		mainFlexTable.getRowFormatter().setVisible(1,
false);

		Label lblTitle = new Label("Title");
		mainFlexTable.setWidget(2, 0, lblTitle);

		titleTxt = new TextBox();
		titleTxt.setWidth("250px");
		mainFlexTable.setWidget(2, 1, titleTxt);

		mainFlexTable.getRowFormatter().setVisible(2,
false);

		Label lblDescription = new Label("Description");
		lblDescription.setWidth("120px");
		mainFlexTable.setWidget(3, 0, lblDescription);

	mainFlexTable.getFlexCellFormatter().setVerticalAlignment(3,
0, HasVerticalAlignment.ALIGN_TOP);

		descTxt = new TextArea();
		descTxt.setWidth("250px");
		mainFlexTable.setWidget(3, 1, descTxt);

		Label lblMoreInformationAt = new Label("More
information at");
		mainFlexTable.setWidget(4, 0,
lblMoreInformationAt);

		linkTxt = new TextBox();
		linkTxt.setWidth("250px");
		mainFlexTable.setWidget(4, 1, linkTxt);

		FlexTable button = new FlexTable();
		mainFlexTable.setWidget(5, 1, button);

		saveButton = new Button("New button");
		saveButton.setText("Save");
		button.setWidget(0, 0, saveButton);

		cancelButton = new Button("New button");
		cancelButton.setText("Cancel");
		button.setWidget(0, 1, cancelButton);

		lblErrorTxt = new Label("");
		lblErrorTxt.setStyleName("errorMsg");
		mainFlexTable.setWidget(6, 1, lblErrorTxt);
	}

	public void setHeader(String text){
		lblHeader.setText(text);
	}

	public void setHidden(String currLoc, String filename){
		this.currLoc.setValue(currLoc);
		this.fileName.setValue(filename);
	}

	public void setDescTxt(String text){
		descTxt.setText(text);
	}

	public String getDescTxt(){
		return descTxt.getText();
	}

	public String getTitleTxt(){
		return titleTxt.getText();
	}

	public void setTitleTxt(String text){
		titleTxt.setText(text);
	}

	public String getLink(){
		return linkTxt.getText();
	}

	public void setLink(String text){
```

```
		linkTxt.setText(text);
	}

	public String getFile(){
		return fileUpload.getFilename();
	}

	public void setErrorTxt(String text){
		lblErrorTxt.setText(text);
	}

	public HasClickHandlers getSaveButton(){
		return saveButton;
	}

	public HasClickHandlers getCancelButton(){
		return cancelButton;
	}

	public FormPanel getForm(){
		return form;
	}

	public void setFileUploadVisibility(Boolean isVisible){
		if(isVisible==null){

	mainFlexTable.getRowFormatter().setVisible(1, false);

	mainFlexTable.getRowFormatter().setVisible(2, false);
		}else{

	mainFlexTable.getRowFormatter().setVisible(1, isVisible);

	mainFlexTable.getRowFormatter().setVisible(2, !isVisible);
		}
	}

	public Widget asWidget(){
		return this;
	}

}
```

AddReferenceFolderView.java

```
package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.*;

public class AddReferenceFolderView extends Composite implements
cepir.client.presenter.AddReferenceFolderPresenter.Display{
	private TextBox nameTxt;
	private TextArea descTxt;
	private Button saveButton;
	private Button cancelButton;
	private Label lblErrorTxt;
	private Label lblHeader;

	public AddReferenceFolderView() {
		FlexTable flexTable = new FlexTable();
		initWidget(flexTable);

		lblHeader = new Label("");
		lblHeader.setStyleName("header2");
		flexTable.setWidget(0, 0, lblHeader);

		Label lblFolderName = new Label("Folder
Name");
		lblFolderName.setWidth("120px");
		flexTable.setWidget(1, 0, lblFolderName);

		nameTxt = new TextBox();
		nameTxt.setWidth("250px");
		flexTable.setWidget(1, 1, nameTxt);

		Label lblDescription = new Label("Description");
		flexTable.setWidget(2, 0, lblDescription);
		flexTable.getFlexCellFormatter().setColSpan(0, 0,
2);
		descTxt = new TextArea();
		descTxt.setWidth("250px");
		flexTable.setWidget(2, 1, descTxt);

		FlexTable buttonTable = new FlexTable();
		flexTable.setWidget(3, 1, buttonTable);

		saveButton = new Button("Save");
		buttonTable.setWidget(0, 0, saveButton);

		cancelButton = new Button("Cancel");
		buttonTable.setWidget(0, 1, cancelButton);

		lblErrorTxt = new Label("");
		lblErrorTxt.setStyleName("errorMsg");
		flexTable.setWidget(4, 1, lblErrorTxt);
	}

	public void setHeader(String text){
		lblHeader.setText(text);
	}

	public void setDesc(String text){
		descTxt.setText(text);
	}

	public void setName(String text){
		nameTxt.setText(text);
	}

	public String getName(){
		return nameTxt.getText();
	}

	public String getDesc(){
		return descTxt.getText();
	}

	public void setErrorTxt(String text){
		lblErrorTxt.setText(text);
	}

	public HasClickHandlers getSaveButton(){
		return saveButton;
	}
	public HasClickHandlers getCancelButton(){
		return cancelButton;
	}


}
```

AddReferenceView.java

```
package cepir.client.view;

import java.util.Date;

public class AddReferenceView extends Composite implements
cepir.client.presenter.AddReferencePresenter.Display{
	private final String UPLOAD_ACTION_URL =
GWT.getModuleBaseURL() + "upload";

	private FormPanel form;
	private FlexTable mainFlexTable;
	private Label lblHeader;
	private TextBox titleBox;
	private TextBox authorBox;
	private TextBox publisherBox;
	private TextBox linkBox;
	private FileUpload fileUpload;
	private Button saveButton;
	private Button cancelButton;
	private Label lblError;
	private Hidden currLoc;
	private Hidden fileName;
	private TextBox pubDateBox;
	private ListBox refTypeBox;
	private TextBox volTxt;
	private TextBox issueTxt;
	private TextBox pagesBox;
```

```
	private TextBox bookTitleTxt;
	private TextBox editorTxt;
	private TextBox placePublishedTxt;
	private TextBox articleNameTxt;
	private TextBox yearTxt;
	private TextBox schoolTxt;
	private TextBox chapTitleBox;
	private TextBox journTitleBox;
	private Image warnButton;

	public AddReferenceView() {
		form = new FormPanel();

	form.setEncoding(FormPanel.ENCODING_MULTIPART);
		form.setMethod(FormPanel.METHOD_POST);
		form.setAction(UPLOAD_ACTION_URL);
		initWidget(form);

		mainFlexTable = new FlexTable();
		form.setWidget(mainFlexTable);

		lblHeader = new Label("");
		lblHeader.setStyleName("header2");
		mainFlexTable.setWidget(0, 0, lblHeader);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 3);

		currLoc = new Hidden("currLoc");
		mainFlexTable.setWidget(0, 2, currLoc);

		fileName = new Hidden("name");
		mainFlexTable.setWidget(0, 3, fileName);

		HTML htmlAllFieldsAre = new HTML("(*)
Required Fields<br/><br/>", true);
		mainFlexTable.setWidget(1, 1, htmlAllFieldsAre);

		Label lblReferenceType = new Label("Reference
Type");
		mainFlexTable.setWidget(2, 0, lblReferenceType);

		refTypeBox = new ListBox();
		refTypeBox.setSelectedIndex(-1);
		refTypeBox.setWidth("250px");
		mainFlexTable.setWidget(2, 1, refTypeBox);

		Label lblTitle = new Label("Title *");
		mainFlexTable.setWidget(3, 0, lblTitle);

		titleBox = new TextBox();
		titleBox.setWidth("250px");
		mainFlexTable.setWidget(3, 1, titleBox);

		Label lblChapTitle = new Label("Chapter  Title
*");
		mainFlexTable.setWidget(4, 0, lblChapTitle);

		chapTitleBox = new TextBox();
		mainFlexTable.setWidget(4, 1, chapTitleBox);
		chapTitleBox.setWidth("250px");

		Label lblArticleName = new Label("Journal
Article Title *");
		mainFlexTable.setWidget(5, 0, lblArticleName);

		articleNameTxt = new TextBox();
		articleNameTxt.setWidth("250px");
		mainFlexTable.setWidget(5, 1, articleNameTxt);

		Label lblJournalName = new Label("Journal Name
*");
		mainFlexTable.setWidget(6, 0, lblJournalName);

		journTitleBox = new TextBox();
		mainFlexTable.setWidget(6, 1, journTitleBox);
		journTitleBox.setWidth("250px");

		Label lblAuthor = new Label("Author/s *");
		lblAuthor.setWidth("150px");
		mainFlexTable.setWidget(7, 0, lblAuthor);

		FlexTable auth = new FlexTable();
		mainFlexTable.setWidget(7, 1, auth);
		authorBox = new TextBox();
		authorBox.setWidth("250px");
		auth.setWidget(0, 0, authorBox);

		final DecoratedPopupPanel info = new
DecoratedPopupPanel(true);
		info.setTitle("Note");
		info.setWidth("175px");
		info.setAutoHideEnabled(false);
		info.setWidget(new Label("e.g. Acuzar, C.A.,
Chua, R.L., and Lee, A.T."));

		ImageResources images =
GWT.create(ImageResources.class);
		Image infoButton = new
Image(images.info().getUrl());
		infoButton.addMouseOverHandler(new
MouseOverHandler() {

			public void
onMouseOver(MouseOverEvent event) {
				Widget source =
(Widget) event.getSource();
		int left = source.getAbsoluteLeft() + 10;
		int top = source.getAbsoluteTop() + 10;

	info.setPopupPosition(left, top);
				info.show();
			}
		});
		infoButton.addMouseOutHandler(new
MouseOutHandler() {

			public void
onMouseOut(MouseOutEvent event) {
				info.hide();
			}
		});

		auth.setWidget(0, 1, infoButton);

		Label lblBookTitle = new Label("Book Title *");
		mainFlexTable.setWidget(8, 0, lblBookTitle);

		bookTitleTxt = new TextBox();
		bookTitleTxt.setWidth("250px");
		mainFlexTable.setWidget(8, 1, bookTitleTxt);

		Label lblEditor = new Label("Editor");
		mainFlexTable.setWidget(9, 0, lblEditor);

		editorTxt = new TextBox();
		editorTxt.setWidth("250px");
		mainFlexTable.setWidget(9, 1, editorTxt);

		Label lblPublisher = new Label("Publisher");
		mainFlexTable.setWidget(10, 0, lblPublisher);

		publisherBox = new TextBox();
		publisherBox.setWidth("250px");
		mainFlexTable.setWidget(10, 1, publisherBox);

		Label lblPublicationDate = new
Label("Publication Date *");
		mainFlexTable.setWidget(11, 0,
lblPublicationDate);

		pubDateBox = new TextBox();
		pubDateBox.setWidth("150px");
		mainFlexTable.setWidget(11, 1, pubDateBox);

		Label lblPlacePublished = new Label("Place
Published");
		mainFlexTable.setWidget(12, 0,
lblPlacePublished);

		placePublishedTxt = new TextBox();
```

```
                placePublishedTxt.setWidth("250px");
                mainFlexTable.setWidget(12, 1,
placePublishedTxt);

                Label lblVolume = new Label("Volume");
                mainFlexTable.setWidget(13, 0, lblVolume);

                volTxt = new TextBox();
                volTxt.setWidth("250px");
                mainFlexTable.setWidget(13, 1, volTxt);

                Label lblIssue = new Label("Issue");
                mainFlexTable.setWidget(14, 0, lblIssue);

                issueTxt = new TextBox();
                issueTxt.setWidth("250px");
                mainFlexTable.setWidget(14, 1, issueTxt);

                Label lblPageNo = new Label("Page Number");
                mainFlexTable.setWidget(15, 0, lblPageNo);

                pagesBox = new TextBox();
                pagesBox.setWidth("250px");
                mainFlexTable.setWidget(15, 1, pagesBox);

                Label lblYear = new Label("Year *");
                mainFlexTable.setWidget(16, 0, lblYear);

                yearTxt = new TextBox();
                yearTxt.setWidth("150px");
                mainFlexTable.setWidget(16, 1, yearTxt);

                Label lblSchool = new Label("School *");
                mainFlexTable.setWidget(17, 0, lblSchool);

                schoolTxt = new TextBox();
                schoolTxt.setWidth("250px");
                mainFlexTable.setWidget(17, 1, schoolTxt);

                Label lblFile = new Label("File");
                mainFlexTable.setWidget(18, 0, lblFile);

                FlexTable file = new FlexTable();
                mainFlexTable.setWidget(18, 1, file);
                fileUpload = new FileUpload();
                fileUpload.setName("file");
                file.setWidget(0, 0, fileUpload);

                final DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidth("160px");
                warning.setWidget(new Label("Uploading a new
file will overwrite existing file"));

                warnButton = new
Image(images.smallWarn().getUrl());
                warnButton.addMouseOverHandler(new
MouseOverHandler() {

                        public void
onMouseOver(MouseOverEvent event) {
                                Widget source =
(Widget) event.getSource();
                int left = source.getAbsoluteLeft() + 10;
                int top = source.getAbsoluteTop() + 10;

        warning.setPopupPosition(left, top);
                                warning.show();
                        }
                });

                warnButton.addMouseOutHandler(new
MouseOutHandler() {

                        public void
onMouseOut(MouseOutEvent event) {
                                warning.hide();
                        }
                });
                file.setWidget(0, 1, warnButton);

                Label lblLink = new Label("Link");
                mainFlexTable.setWidget(19, 0, lblLink);

                linkBox = new TextBox();
                linkBox.setWidth("300px");
                mainFlexTable.setWidget(19, 1, linkBox);

                HTML htmlnoteYoumust = new
HTML("<b>NOTE:</b> You must either upload the reference or provide "
+
                                "the link where it can be
accessed", true);
                mainFlexTable.setWidget(20, 1,
htmlnoteYoumust);

                FlexTable buttonTable = new FlexTable();
                mainFlexTable.setWidget(21, 1, buttonTable);

                saveButton = new Button("Save");
                buttonTable.setWidget(0, 0, saveButton);

                cancelButton = new Button("Cancel");
                buttonTable.setWidget(0, 1, cancelButton);

                lblError = new Label("");
                lblError.setStyleName("errorMsg");
                mainFlexTable.setWidget(22, 1, lblError);

        mainFlexTable.getFlexCellFormatter().setColSpan(22, 1, 2);

                hideAllFields();
        }

        public void showOverWriteFileWarn(Boolean visible){
                warnButton.setVisible(visible);
        }

        public void setHeader(String text){
                lblHeader.setText(text);
        }

        public void setHidden(String currLoc, String filename){
                this.currLoc.setValue(currLoc);
                this.fileName.setValue(filename);
        }

        public String getTitleTxt(){
                return titleBox.getText();
        }

        public void setTitleTxt(String text){
                titleBox.setText(text);
        }

        public String getChapTitleTxt(){
                return chapTitleBox.getText();
        }

        public void setChapTitleTxt(String text){
                chapTitleBox.setText(text);
        }

        public String getJournTitleTxt(){
                return journTitleBox.getText();
        }

        public void setJournTitleTxt(String text){
                journTitleBox.setText(text);
        }

        public String getAuthorTxt(){
                return authorBox.getText();
        }

        public void setAuthorTxt(String text){
                authorBox.setText(text);
        }
```

```
public String getBookTitleTxt(){
		return bookTitleTxt.getText();
}

public void setBookTitleTxt(String text){
		bookTitleTxt.setText(text);
}

public String getEditorTxt(){
		return editorTxt.getText();
}

public void setEditorTxt(String text){
		editorTxt.setText(text);
}

public void setRefType(int selectedIndex){
		refTypeBox.setSelectedIndex(selectedIndex);
}

public void setTypeBox(List<ReferenceType> types){
		refTypeBox.clear();
		for(int i = 0; i < types.size(); i++){

refTypeBox.insertItem(types.get(i).toString(), i);
		}
		refTypeBox.setSelectedIndex(0);
}

public Integer getRefType(){
		return refTypeBox.getSelectedIndex();
}

public String getPublisherTxt(){
		return publisherBox.getText();
}

public void setPublisherTxt(String text){
		publisherBox.setText(text);
}

public void setPubDate(String date){
		pubDateBox.setText(date);
}

public String getPubDate(){
		return pubDateBox.getText();
}

public void setPubPlace(String text){
		placePublishedTxt.setText(text);
}

public String getPubPlace(){
		return placePublishedTxt.getText();
}

public void setVolume(String text){
		volTxt.setText(text);
}

public String getVolume(){
		return volTxt.getText();
}

public void setIssue(String text){
		issueTxt.setText(text);
}

public String getIssue(){
		return issueTxt.getText();
}

public void setPageNo(String text){
		pagesBox.setText(text);
}

public String getPageNo(){
		return pagesBox.getText();
}

public void setArticleName(String text){
		articleNameTxt.setText(text);
}

public String getArticleName(){
		return articleNameTxt.getText();
}

public void setYear(String text){
		yearTxt.setText(text);
}

public String getYear(){
		return yearTxt.getText();
}

public void setSchool(String text){
		schoolTxt.setText(text);
}

public String getSchool(){
		return schoolTxt.getText();
}

public String getLink(){
		return linkBox.getText();
}

public void setLink(String text){
		linkBox.setText(text);
}

public String getFile(){
		return fileUpload.getFilename();
}

public void setErrorTxt(String text){
		lblError.setText(text);
}

public HasClickHandlers getSaveButton(){
		return saveButton;
}

public HasClickHandlers getCancelButton(){
		return cancelButton;
}

public HasChangeHandlers getRefOption(){
		return refTypeBox;
}

public void hideAllFields(){
		for(int i = 3; i < mainFlexTable.getRowCount();
i++){

mainFlexTable.getRowFormatter().setVisible(i, false);
		}
}

public void showBookForm(){
		mainFlexTable.getRowFormatter().setVisible(3,
true);
		mainFlexTable.getRowFormatter().setVisible(7,
true);
		mainFlexTable.getRowFormatter().setVisible(10,
true);
		mainFlexTable.getRowFormatter().setVisible(11,
true);
		mainFlexTable.getRowFormatter().setVisible(12,
true);

		mainFlexTable.getRowFormatter().setVisible(18,
true);
		mainFlexTable.getRowFormatter().setVisible(19,
true);
		mainFlexTable.getRowFormatter().setVisible(20,
true);
```

```
                mainFlexTable.getRowFormatter().setVisible(21,
true);
                mainFlexTable.getRowFormatter().setVisible(22,
true);
        }

        public void showBookChapForm(){
                mainFlexTable.getRowFormatter().setVisible(4,
true);
                mainFlexTable.getRowFormatter().setVisible(7,
true);
                mainFlexTable.getRowFormatter().setVisible(8,
true);
                mainFlexTable.getRowFormatter().setVisible(9,
true);
                mainFlexTable.getRowFormatter().setVisible(10,
true);
                mainFlexTable.getRowFormatter().setVisible(11,
true);

                mainFlexTable.getRowFormatter().setVisible(18,
true);
                mainFlexTable.getRowFormatter().setVisible(19,
true);
                mainFlexTable.getRowFormatter().setVisible(20,
true);
                mainFlexTable.getRowFormatter().setVisible(21,
true);
                mainFlexTable.getRowFormatter().setVisible(22,
true);
        }

        public void showJournalArticleForm(){
                mainFlexTable.getRowFormatter().setVisible(5,
true);
                mainFlexTable.getRowFormatter().setVisible(6,
true);
                mainFlexTable.getRowFormatter().setVisible(7,
true);
                mainFlexTable.getRowFormatter().setVisible(11,
true);
                mainFlexTable.getRowFormatter().setVisible(13,
true);
                mainFlexTable.getRowFormatter().setVisible(14,
true);
                mainFlexTable.getRowFormatter().setVisible(15,
true);

                mainFlexTable.getRowFormatter().setVisible(18,
true);
                mainFlexTable.getRowFormatter().setVisible(19,
true);
                mainFlexTable.getRowFormatter().setVisible(20,
true);
                mainFlexTable.getRowFormatter().setVisible(21,
true);
                mainFlexTable.getRowFormatter().setVisible(22,
true);
        }

        public void showEArticleForm(){
                mainFlexTable.getRowFormatter().setVisible(3,
true);
                mainFlexTable.getRowFormatter().setVisible(7,
true);

                mainFlexTable.getRowFormatter().setVisible(18,
true);
                mainFlexTable.getRowFormatter().setVisible(19,
true);
                mainFlexTable.getRowFormatter().setVisible(20,
true);
                mainFlexTable.getRowFormatter().setVisible(21,
true);
                mainFlexTable.getRowFormatter().setVisible(22,
true);
        }

        public void showUnpublishedWorkForm(){
                mainFlexTable.getRowFormatter().setVisible(3,
true);
                mainFlexTable.getRowFormatter().setVisible(7,
true);

                mainFlexTable.getRowFormatter().setVisible(18,
true);
                mainFlexTable.getRowFormatter().setVisible(19,
true);
                mainFlexTable.getRowFormatter().setVisible(20,
true);
                mainFlexTable.getRowFormatter().setVisible(21,
true);
                mainFlexTable.getRowFormatter().setVisible(22,
true);
        }

        public void showThesisForm(){
                mainFlexTable.getRowFormatter().setVisible(3,
true);
                mainFlexTable.getRowFormatter().setVisible(7,
true);
                mainFlexTable.getRowFormatter().setVisible(16,
true);
                mainFlexTable.getRowFormatter().setVisible(17,
true);

                mainFlexTable.getRowFormatter().setVisible(18,
true);
                mainFlexTable.getRowFormatter().setVisible(19,
true);
                mainFlexTable.getRowFormatter().setVisible(20,
true);
                mainFlexTable.getRowFormatter().setVisible(21,
true);
                mainFlexTable.getRowFormatter().setVisible(22,
true);
        }

        public FormPanel getForm(){
                return form;
        }

        public void setRefTypeEnabled(Boolean enabled){
                refTypeBox.setEnabled(enabled);
        }

        public void setFileUploadVisibility(Boolean isVisible){

        mainFlexTable.getRowFormatter().setVisible(mainFlexTable.g
etRowCount()-5, isVisible);
        }

        public void setNote(Boolean isVisible){

        mainFlexTable.getRowFormatter().setVisible(mainFlexTable.g
etRowCount()-3, isVisible);
        }

        public Widget asWidget(){
                return this;
        }

}
```

AddToolView.java

```
 package cepir.client.view;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Hidden;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.FileUpload;
import com.google.gwt.user.client.ui.Button;
```

```
import com.google.gwt.user.client.ui.Widget;

public class AddToolView extends Composite implements
cepir.client.presenter.AddToolPresenter.Display{
	private final String UPLOAD_ACTION_URL =
GWT.getModuleBaseURL() + "upload";

	private FormPanel formPanel;
	private FlexTable mainFlexTable;
	private Label lblHeader;
	private Label lblFile;
	private TextArea descTxt;
	private TextBox linkTxt;
	private FileUpload fileUpload;
	private Button saveButton;
	private Button cancelButton;
	private Label lblError;
	private Label lblNameError;
	private Label lblLinkOrFileError;
	private Hidden currLoc;
	private Hidden fileName;
	private Label lblTitle;
	private TextBox titleTxt;

	public AddToolView() {

		formPanel = new FormPanel();
		initWidget(formPanel);

	formPanel.setEncoding(FormPanel.ENCODING_MULTIPAR
T);

	formPanel.setMethod(FormPanel.METHOD_POST);
		formPanel.setAction(UPLOAD_ACTION_URL);

		mainFlexTable = new FlexTable();
		formPanel.setWidget(mainFlexTable);

		lblHeader = new Label("");
		lblHeader.setStyleName("header2");
		mainFlexTable.setWidget(0, 0, lblHeader);

		currLoc = new Hidden("currLoc");
		mainFlexTable.setWidget(0, 2, currLoc);

		fileName = new Hidden("name");
		mainFlexTable.setWidget(0, 3, fileName);

		lblFile = new Label("File");
		lblFile.setVisible(false);
		mainFlexTable.setWidget(1, 0, lblFile);

		fileUpload = new FileUpload();
		fileUpload.setName("file");
		fileUpload.setVisible(false);
		mainFlexTable.setWidget(1, 1, fileUpload);

		lblTitle = new Label("Title");
		lblTitle.setVisible(false);
		mainFlexTable.setWidget(2, 0, lblTitle);

		titleTxt = new TextBox();
		titleTxt.setWidth("250px");
		titleTxt.setVisible(false);
		mainFlexTable.setWidget(2, 1, titleTxt);

		Label lblDescription = new Label("Description");
		lblDescription.setWidth("100px");
		mainFlexTable.setWidget(3, 0, lblDescription);

	mainFlexTable.getFlexCellFormatter().setVerticalAlignment(3,
0, HasVerticalAlignment.ALIGN_TOP);

		descTxt = new TextArea();
		descTxt.setWidth("250px");
		mainFlexTable.setWidget(3, 1, descTxt);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

		Label lblLink = new Label("Project Homepage");
		mainFlexTable.setWidget(4, 0, lblLink);

		linkTxt = new TextBox();
		linkTxt.setWidth("250px");
		mainFlexTable.setWidget(4, 1, linkTxt);

		FlexTable buttonTable = new FlexTable();
		mainFlexTable.setWidget(5, 1, buttonTable);

		saveButton = new Button("Save");
		buttonTable.setWidget(0, 0, saveButton);

		cancelButton = new Button("Cancel");
		buttonTable.setWidget(0, 1, cancelButton);

		lblError = new Label("");
		lblError.setStyleName("errorMsg");
		mainFlexTable.setWidget(6, 1, lblError);

		lblNameError = new Label("");
		lblNameError.setStyleName("errorMsg");
		mainFlexTable.setWidget(7, 1, lblNameError);

		lblLinkOrFileError = new Label("");
		lblLinkOrFileError.setStyleName("errorMsg");
		mainFlexTable.setWidget(8, 1,
lblLinkOrFileError);
	}

	public void setVisibilityUploader(Boolean isVisible){
		fileUpload.setVisible(isVisible);
		lblFile.setVisible(isVisible);
	}

	public void setVisibilityTitle(Boolean isVisible){
		titleTxt.setVisible(isVisible);
		lblTitle.setVisible(isVisible);
	}

	public void setHidden(String currLoc, String filename){
		this.currLoc.setValue(currLoc);
		fileName.setValue(filename);
	}

	public void setDescTxt(String text){
		descTxt.setText(text);
	}

	public String getDescTxt(){
		return descTxt.getText();
	}

	public void setHeaderTxt(String text){
		lblHeader.setText(text);
	}

	public String getTitleTxt(){
		return titleTxt.getText();
	}

	public void setTitleTxt(String text){
		titleTxt.setText(text);
	}
	public String getLink(){
		return linkTxt.getText();
	}

	public void setLink(String text){
		linkTxt.setText(text);
	}

	public String getFile(){
		return fileUpload.getFilename();
	}

	public void setErrorTxt(String text){
		lblError.setText(text);
	}

	public void setNameErrorTxt(String text){
```

```
                lblNameError.setText(text);
        }

        public void setLinkOrFileErrorTxt(String text){
                lblLinkOrFileError.setText(text);
        }

        public HasClickHandlers getSaveButton(){
                return saveButton;
        }

        public HasClickHandlers getCancelButton(){
                return cancelButton;
        }

        public FormPanel getForm(){
                return formPanel;
        }

        public Widget asWidget(){
                return this;
        }
}
```

AddUserView.java

```
package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.PasswordTextBox;
import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Widget;

import cepir.client.presenter.AddUserPresenter;
import com.google.gwt.user.client.ui.ListBox;

public class AddUserView extends Composite implements
AddUserPresenter.Display{
        private FlexTable addUserTable;
        private TextBox textBoxUsername;
        private PasswordTextBox textBoxPassword;
        private TextBox textBoxFirstName;
        private TextBox textBoxMiddleName;
        private Button okButton;
        private Button cancelButton;
        private Button clearButton;
        private TextBox textBoxLastName;
        private TextBox textBoxEmail;
        private TextBox textBoxAffiliation;
        private ListBox sysAdBox;
        private Label lblError;
        private FlexTable buttonPanel;
        private Label lblAddUser;
        private Label lblUsername;
        private Label lblPassword;
        private Label lblFirstName;
        private Label lblMiddleName;
        private Label lblLastName;
        private Label lblEmail;
        private Label lblAffiliation;
        private Label lblSystemAdministrator;
        private Label lblRequiredField;
        private Label lblUsernameMsg;
        private Label lblRetypePassword;
        private PasswordTextBox textBoxRetypePassword;
        private Label lblRetypePasswordMsg;
        private Label lblEmailMsg;
        private Label lblToolAdministrator;
        private ListBox toolAdBox;
        private TextBox questionBox;
        private TextBox answerBox;

        public AddUserView() {
                addUserTable = new FlexTable();
                initWidget(addUserTable);
                addUserTable.setSize("320px", "40px");

                lblAddUser = new Label("Add User");
                lblAddUser.setStyleName("header2");
                addUserTable.setWidget(0, 0, lblAddUser);

                lblRequiredField = new Label("(*) Required
field");
                addUserTable.setWidget(1, 1, lblRequiredField);

                lblUsername = new Label("Username *");
                addUserTable.setWidget(2, 0, lblUsername);

                textBoxUsername = new TextBox();
                addUserTable.setWidget(2, 1, textBoxUsername);

                lblUsernameMsg = new Label("");
                addUserTable.setWidget(2, 2, lblUsernameMsg);

                lblPassword = new Label("Password *");
                addUserTable.setWidget(3, 0, lblPassword);

                textBoxPassword = new PasswordTextBox();
                addUserTable.setWidget(3, 1, textBoxPassword);

                lblRetypePassword = new Label("Retype
Password *");
                addUserTable.setWidget(4, 0,
lblRetypePassword);

                textBoxRetypePassword = new
PasswordTextBox();
                addUserTable.setWidget(4, 1,
textBoxRetypePassword);

                lblRetypePasswordMsg = new Label("");
                lblRetypePasswordMsg.setWordWrap(false);
                addUserTable.setWidget(4, 2,
lblRetypePasswordMsg);

                lblFirstName = new Label("First Name *");
                addUserTable.setWidget(5, 0, lblFirstName);

                textBoxFirstName = new TextBox();
                addUserTable.setWidget(5, 1, textBoxFirstName);

                lblMiddleName = new Label("Middle Name *");
                addUserTable.setWidget(6, 0, lblMiddleName);

                textBoxMiddleName = new TextBox();
                addUserTable.setWidget(6, 1,
textBoxMiddleName);

                lblLastName = new Label("Last Name *");
                addUserTable.setWidget(7, 0, lblLastName);

                textBoxLastName = new TextBox();
                addUserTable.setWidget(7, 1, textBoxLastName);

                lblEmail = new Label("Email *");
                addUserTable.setWidget(8, 0, lblEmail);

                textBoxEmail = new TextBox();
                addUserTable.setWidget(8, 1, textBoxEmail);

                lblEmailMsg = new Label("");
                lblEmailMsg.setWordWrap(false);
                addUserTable.setWidget(8, 2, lblEmailMsg);

                lblSystemAdministrator = new Label("System
Administrator *");
                lblSystemAdministrator.setWordWrap(false);
                addUserTable.setWidget(9, 0,
lblSystemAdministrator);

                sysAdBox = new ListBox();
                sysAdBox.addItem("Yes");
                sysAdBox.addItem("No");
                addUserTable.setWidget(9, 1, sysAdBox);
```

```
                lblToolAdministrator = new Label("Tool
Administrator *");
                addUserTable.setWidget(10, 0,
lblToolAdministrator);

                toolAdBox = new ListBox();
                toolAdBox.addItem("Yes");
                toolAdBox.addItem("No");
                addUserTable.setWidget(10, 1, toolAdBox);

                lblAffiliation = new Label("Affiliation");
                addUserTable.setWidget(11, 0, lblAffiliation);

                textBoxAffiliation = new TextBox();
                addUserTable.setWidget(11, 1,
textBoxAffiliation);

                Label lblSecurityQuestion = new Label("Security
Question *");
                addUserTable.setWidget(12, 0,
lblSecurityQuestion);

                questionBox = new TextBox();
                questionBox.setWidth("250px");
                addUserTable.setWidget(12, 1, questionBox);

                Label lblAnswer = new Label("Answer *");
                addUserTable.setWidget(13, 0, lblAnswer);

                answerBox = new TextBox();
                addUserTable.setWidget(13, 1, answerBox);

                buttonPanel = new FlexTable();
                addUserTable.setWidget(14, 1, buttonPanel);

                okButton = new Button("Save");
                buttonPanel.setWidget(0,0,okButton);

                cancelButton = new Button("Cancel");
                buttonPanel.setWidget(0,1,cancelButton);

                clearButton = new Button("Clear");
                buttonPanel.setWidget(0,2,clearButton);

                lblError = new Label("");
                addUserTable.setWidget(15, 1, lblError);

        addUserTable.getFlexCellFormatter().setColSpan(0, 0, 3);

        }

        public void setErrorTxt(String errMsg,Boolean isError){

        this.lblError.setStyleName((isError?"errorMsg":"okMsg"));
                this.lblError.setText(errMsg);
        }

        public void setUsernameMsgTxt(String usernameMsg){
                this.lblUsernameMsg.setStyleName("errorMsg");
                this.lblUsernameMsg.setWidth("350px");
                this.lblUsernameMsg.setText(usernameMsg);
        }

        public void setEmailMsgTxt(String emailMsg){
                this.lblEmailMsg.setStyleName("errorMsg");
                this.lblEmailMsg.setText(emailMsg);
        }

        public void setRetypePasswdMsgTxt(String msg){

        this.lblRetypePasswordMsg.setStyleName("errorMsg");
                this.lblRetypePasswordMsg.setText(msg);
        }


        public String getFirstName(){
                return textBoxFirstName.getText();
        }

        public String getMiddleName(){
                return textBoxMiddleName.getText();
        }

        public String getLastName(){
                return textBoxLastName.getText();
        }

        public String getEmail(){
                return textBoxEmail.getText();
        }

        public String getAffiliation(){
                return textBoxAffiliation.getText();
        }

        public String getUsername(){
                return textBoxUsername.getText();
        }

        public String getPassword(){
                return textBoxPassword.getText();
        }

        public String getRetypedPassword(){
                return textBoxRetypePassword.getText();
        }

        public String getSysAdPriv(){
                return
sysAdBox.getItemText(sysAdBox.getSelectedIndex());
        }

        public String getToolAdPriv(){
                return
toolAdBox.getItemText(toolAdBox.getSelectedIndex());
        }

        public String getQuestion(){
                return questionBox.getText();
        }

        public String getAnswer(){
                return answerBox.getText();
        }

        public void clearForm(){
                textBoxAffiliation.setText("");
                textBoxFirstName.setText("");
                textBoxLastName.setText("");
                textBoxMiddleName.setText("");
                textBoxEmail.setText("");
                textBoxUsername.setText("");
                textBoxPassword.setText("");
                textBoxRetypePassword.setText("");

                lblError.setText("");
                lblUsernameMsg.setText("");
                lblEmailMsg.setText("");
                lblRetypePasswordMsg.setText("");
                sysAdBox.setSelectedIndex(0);
        }

        public HasClickHandlers getOkButton(){
                return okButton;
        }

        public HasClickHandlers getCancelButton(){
                return cancelButton;
        }

        public HasClickHandlers getClearButton(){
                return clearButton;
        }

        public Widget asWidget(){
                return this;
        }
}
```

EditMembershipView.java

```
package cepir.client.view;

import cepir.client.presenter.EditMembershipPresenter;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Widget;

public class EditMembershipView extends Composite implements
EditMembershipPresenter.Display{
	private Label usernameTxt;
	private Label fnameTxt;
	private Label mnameTxt;
	private Label lnameTxt;
	private ListBox comboBox;
	private Button saveButton;
	private Button cancelButton;

	public EditMembershipView() {

		FlexTable mainTable = new FlexTable();
		initWidget(mainTable);

	mainTable.setWidth(Integer.toString((int)(Window.getClientW
idth()*0.8*0.4))+"px");

		Label lblEditMembership = new Label("Edit
Membership");
		lblEditMembership.setStyleName("header2");
		mainTable.setWidget(0, 0, lblEditMembership);

		Label lblUsername = new Label("Username");
		mainTable.setWidget(1, 0, lblUsername);

		usernameTxt = new Label("");
		usernameTxt.setStyleName("idLabel");
		mainTable.setWidget(1, 1, usernameTxt);

		Label lblFirstName = new Label("First Name");
		mainTable.setWidget(2, 0, lblFirstName);

		fnameTxt = new Label("");
		mainTable.setWidget(2, 1, fnameTxt);

		Label lblMiddleName = new Label("Middle
Name");
		mainTable.setWidget(3, 0, lblMiddleName);

		mnameTxt = new Label("");
		mainTable.setWidget(3, 1, mnameTxt);

		Label lblLastName = new Label("Last Name");
		mainTable.setWidget(4, 0, lblLastName);

		lnameTxt = new Label("");
		mainTable.setWidget(4, 1, lnameTxt);

		Label lblMembership = new Label("Membership
Type");
		mainTable.setWidget(5, 0, lblMembership);
		mainTable.getFlexCellFormatter().setColSpan(0,
0, 2);

		comboBox = new ListBox();
		comboBox.addItem("Member");
		comboBox.addItem("Administrator");
		comboBox.setSelectedIndex(-1);
		mainTable.setWidget(5, 1, comboBox);

		FlexTable buttonPanel = new FlexTable();
		mainTable.setWidget(6, 1, buttonPanel);

		saveButton = new Button("New button");
		saveButton.setText("Save");
		buttonPanel.setWidget(0, 0, saveButton);

		cancelButton = new Button("New button");
		cancelButton.setText("Cancel");
		buttonPanel.setWidget(0, 1, cancelButton);
	}

	public void setUsernameTxt(String text){
		usernameTxt.setText(text);
	}

	public void setFirstnameTxt(String text){
		fnameTxt.setText(text);
	}

	public void setMidnameTxt(String text){
		mnameTxt.setText(text);
	}

	public void setLastnameTxt(String text){
		lnameTxt.setText(text);
	}

	public Integer getSelectedMemberType(){
		return comboBox.getSelectedIndex();
	}

	public void setMemberType(Integer roleID){
		comboBox.setSelectedIndex(roleID-1);
	}

	public HasClickHandlers getSaveButton(){
		return saveButton;
	}

	public HasClickHandlers getCancelButton(){
		return cancelButton;
	}

	public Widget asWidget(){
		return this;
	}

}
```

EditMyInfoView.java

```
package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.PasswordTextBox;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Widget;

public class EditMyInfoView extends Composite implements
cepir.client.presenter.EditMyInfoPresenter.Display{
	private Button saveButton;
	private Button changeButton;
	private Button cancelButton;
	private TextBox firstnameTxt;
	private TextBox midnameTxt;
	private TextBox lastnameTxt;
	private TextBox emailTxt;
	private TextBox affiliationTxt;
	private Label errorTxt;
	private Label emailErrorTxt;
	private Label idlabel;
	private DialogBox passwdDialog;
	private PasswordTextBox oldPwdBox;
	private PasswordTextBox newPwdBox;
	private PasswordTextBox retypeNewPwdBox;
	private Button okPwdButton;
	private Button cancelPwdButton;
	private Label pwdError;
```

```
	private TextBox questionBox;
	private TextBox answerBox;

	public EditMyInfoView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

	mainFlexTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75*0.9))+"px");

		Label lblEditMyAccount = new Label("Edit My
Account");
		lblEditMyAccount.setStyleName("header4");
		mainFlexTable.setWidget(0, 0,
lblEditMyAccount);

		Label lblLoginInformation = new Label("Login
Information");
		lblLoginInformation.setStyleName("header5");
		mainFlexTable.setWidget(1, 0,
lblLoginInformation);

		Label lblUsername = new Label("Username");
		lblUsername.setWidth("150px");
		mainFlexTable.setWidget(2, 0, lblUsername);

		idlabel = new Label("");
		idlabel.setStyleName("idLabel");
		mainFlexTable.setWidget(2, 1, idlabel);

		Label lblPassword = new Label("Password");
		mainFlexTable.setWidget(3, 0, lblPassword);

		changeButton = new Button("New button");
		changeButton.setText("Change");
		mainFlexTable.setWidget(3, 1, changeButton);

		Label lblBasicInformation = new Label("Basic
Information");
		lblBasicInformation.setStyleName("header5");
		mainFlexTable.setWidget(4, 0,
lblBasicInformation);

		Label lblRequiredFields = new Label("(*)
Required fields");
		mainFlexTable.setWidget(5, 1, lblRequiredFields);

		Label lblFirstName = new Label("First Name *");
		mainFlexTable.setWidget(6, 0, lblFirstName);

		firstnameTxt = new TextBox();
		firstnameTxt.setWidth("300px");
		mainFlexTable.setWidget(6, 1, firstnameTxt);

		Label lblMiddleName = new Label("Middle Name
*");
		mainFlexTable.setWidget(7, 0, lblMiddleName);

		midnameTxt = new TextBox();
		midnameTxt.setWidth("300px");
		mainFlexTable.setWidget(7, 1, midnameTxt);

		Label lblLastName = new Label("Last Name *");
		mainFlexTable.setWidget(8, 0, lblLastName);

		lastnameTxt = new TextBox();
		lastnameTxt.setWidth("300px");
		mainFlexTable.setWidget(8, 1, lastnameTxt);

		Label lblEmail = new Label("Email *");
		mainFlexTable.setWidget(9, 0, lblEmail);

		emailTxt = new TextBox();
		emailTxt.setWidth("300px");
		mainFlexTable.setWidget(9, 1, emailTxt);

		emailErrorTxt = new Label("");
		emailErrorTxt.setStyleName("errorMsg");
		mainFlexTable.setWidget(9, 2, emailErrorTxt);

		Label lblAffiliation = new Label("Affiliation");
		mainFlexTable.setWidget(10, 0, lblAffiliation);

	mainFlexTable.getFlexCellFormatter().setColSpan(1, 0, 3);

	mainFlexTable.getFlexCellFormatter().setColSpan(4, 0, 3);

		affiliationTxt = new TextBox();
		affiliationTxt.setWidth("300px");
		mainFlexTable.setWidget(10, 1, affiliationTxt);

		Label lblSecurit = new Label("Security
Question");
		lblSecurit.setStyleName("header5");
		mainFlexTable.setWidget(11, 0, lblSecurit);

		Label lblQuestion = new Label("Question *");
		mainFlexTable.setWidget(12, 0, lblQuestion);

		questionBox = new TextBox();
		questionBox.setWidth("250px");
		mainFlexTable.setWidget(12, 1, questionBox);

		Label lblAnswer = new Label("Answer *");
		mainFlexTable.setWidget(13, 0, lblAnswer);

		answerBox = new TextBox();
		mainFlexTable.setWidget(13, 1, answerBox);

		FlexTable buttonTable = new FlexTable();
		mainFlexTable.setWidget(14, 0, buttonTable);

		saveButton = new Button("Save");
		buttonTable.setWidget(0, 0, saveButton);

		cancelButton = new Button("Cancel");
		buttonTable.setWidget(0, 1, cancelButton);

	mainFlexTable.getFlexCellFormatter().setColSpan(14, 0, 2);

	mainFlexTable.getCellFormatter().setHorizontalAlignment(14,
0, HasHorizontalAlignment.ALIGN_CENTER);

		errorTxt = new Label("");
		errorTxt.setStyleName("errorMsg");
		mainFlexTable.setWidget(15, 0, errorTxt);

	mainFlexTable.getFlexCellFormatter().setColSpan(15, 0, 2);

	mainFlexTable.getCellFormatter().setHorizontalAlignment(15,
0, HasHorizontalAlignment.ALIGN_CENTER);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

	mainFlexTable.getFlexCellFormatter().setColSpan(11, 0, 2);

		//password dialog box
		passwdDialog = new DialogBox();
		passwdDialog.setAnimationEnabled(true);
		passwdDialog.setText("Change password");
		passwdDialog.setWidth("350px");

		FlexTable pwdForm = new FlexTable();
		passwdDialog.setWidget(pwdForm);

		Label oldPwd = new Label("Type current
password");
		oldPwd.setWidth("150px");
		pwdForm.setWidget(0, 0, oldPwd);

		Label newPwd = new Label("Type new
password");
		pwdForm.setWidget(1, 0, newPwd);

		Label retypeNewPwd = new Label("Re-type new
password");
		pwdForm.setWidget(2, 0, retypeNewPwd);

		oldPwdBox = new PasswordTextBox();
```

```
                pwdForm.setWidget(0, 1, oldPwdBox);

                newPwdBox = new PasswordTextBox();
                pwdForm.setWidget(1, 1, newPwdBox);

                retypeNewPwdBox = new PasswordTextBox();
                pwdForm.setWidget(2, 1, retypeNewPwdBox);

                FlexTable buttons = new FlexTable();
                pwdForm.setWidget(3, 0, buttons);
                pwdForm.getFlexCellFormatter().setColSpan(3, 0,
2);

        pwdForm.getFlexCellFormatter().setHorizontalAlignment(3, 0,
HasHorizontalAlignment.ALIGN_CENTER);

                okPwdButton = new Button("Ok");
                buttons.setWidget(0, 0, okPwdButton);

                cancelPwdButton = new Button("Cancel");
                buttons.setWidget(0, 1, cancelPwdButton);

                pwdError = new Label("");
                pwdError.setStyleName("errorMsg");
                pwdForm.setWidget(4, 0, pwdError);
                pwdForm.getFlexCellFormatter().setColSpan(4, 0,
2);

        pwdForm.getFlexCellFormatter().setHorizontalAlignment(4, 0,
HasHorizontalAlignment.ALIGN_CENTER);
        }

        public void setFirstNameTxt(String text){
                firstnameTxt.setText(text);
        }

        public void setMiddleNameTxt(String text){
                midnameTxt.setText(text);
        }

        public void setLastNameTxt(String text){
                lastnameTxt.setText(text);
        }

        public void setEmailTxt(String text){
                emailTxt.setText(text);
        }

        public void setAffiliationTxt(String text){
                affiliationTxt.setText(text);
        }

        public String getFirstnameTxt(){
                return firstnameTxt.getText();
        }

        public String getMidnameTxt(){
                return midnameTxt.getText();
        }

        public String getLastnameTxt(){
                return lastnameTxt.getText();
        }

        public String getEmailTxt(){
                return emailTxt.getText();
        }

        public String getAffiliationTxt(){
                return affiliationTxt.getText();
        }

        public void setErrorTxt(String text){
                errorTxt.setText(text);
        }

        public void setEmailErrTxt(String text){
                emailErrorTxt.setText(text);
        }

public void clearErrorTxt(){
        errorTxt.setText("");
}

public void clearEmailErrTxt(){
        emailErrorTxt.setText("");
}

public void setPwdErrTxt(String text){
        pwdError.setText(text);
}

public void clearPwdErrTxt(){
        pwdError.setText("");
}

public void clearPwdDialog(){
        pwdError.setText("");
        oldPwdBox.setText("");
        newPwdBox.setText("");
        retypeNewPwdBox.setText("");
}

public HasClickHandlers getChangePwdButton(){
        return changeButton;
}

public HasClickHandlers getSaveButton(){
        return saveButton;
}

public HasClickHandlers getCancelButton(){
        return cancelButton;
}

public HasClickHandlers getCancelPwdButton(){
        return cancelPwdButton;
}

public HasClickHandlers getOkPwdButton(){
        return okPwdButton;
}

public void setUsernameLabel(String text){
        idlabel.setText(text);
}

public void showPwdDialog(){
        passwdDialog.center();
        passwdDialog.show();
}

public void hidePwdDialog(){
        passwdDialog.hide();
}

public String getOldPwd(){
        return oldPwdBox.getText();
}

public String getNewPwd(){
        return newPwdBox.getText();
}

public String getRetypePwd(){
        return retypeNewPwdBox.getText();
}

public String getQuestion(){
        return questionBox.getText();
}

public void setQuestion(String text){
        questionBox.setText(text);
}

public void setAnswer(String text){
        answerBox.setText(text);
}
```

```
	public String getAnswer(){
		return answerBox.getText();
	}
	public Widget asWidget(){
		return this;
	}

}
```

EditUserView.java

```
package cepir.client.view;

import cepir.client.presenter.EditUserPresenter;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.PasswordTextBox;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.CheckBox;

public class EditUserView extends Composite implements
EditUserPresenter.Display{
	private Label lblUsernameTxt;
	private TextBox textBoxFirstName;
	private PasswordTextBox textBoxRetypePassword;
	private PasswordTextBox textBoxPassword;
	private TextBox textBoxMiddleName;
	private TextBox textBoxLastName;
	private TextBox textBoxEmail;
	private TextBox textBoxAffiliation;
	private Button updateButton;
	private Button cancelButton;
	private ListBox sysAdBox;
	private CheckBox chckbxChange;
	private Label lblErrorMsg;
	private Label lblEmailMsg;
	private Label lblRetypePasswdMsg;
	private Label lblUsernameMsg;
	private ListBox toolAdBox;
	private Label lblSecurityQuestion;
	private Label lblAnswer;
	private TextBox questionBox;
	private TextBox answerBox;

	public EditUserView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		Label lblEditUser = new Label("Edit User");
		lblEditUser.setStyleName("header2");
		mainFlexTable.setWidget(0, 0, lblEditUser);

		Label lblRequiredField = new Label("(*) Required
field");
		mainFlexTable.setWidget(1, 1, lblRequiredField);

		Label lblUsername = new Label("Username");
		mainFlexTable.setWidget(2, 0, lblUsername);

		lblUsernameTxt = new Label("");
		lblUsernameTxt.setStyleName("idLabel");
		mainFlexTable.setWidget(2, 1, lblUsernameTxt);

		lblUsernameMsg = new Label("");
		mainFlexTable.setWidget(2, 2, lblUsernameMsg);

		Label lblPassword = new Label("Password *");
		mainFlexTable.setWidget(3, 0, lblPassword);

		textBoxPassword = new PasswordTextBox();
		textBoxPassword.setEnabled(false);
		mainFlexTable.setWidget(3, 1, textBoxPassword);

		chckbxChange = new CheckBox("Change");
		mainFlexTable.setWidget(3, 2, chckbxChange);

		Label lblRetypePassword = new Label("Re-type
password *");
		mainFlexTable.setWidget(4, 0,
lblRetypePassword);

		textBoxRetypePassword = new
PasswordTextBox();
		textBoxRetypePassword.setEnabled(false);
		mainFlexTable.setWidget(4, 1,
textBoxRetypePassword);

		lblRetypePasswdMsg = new Label("");
		mainFlexTable.setWidget(4, 2,
lblRetypePasswdMsg);

		Label lblFirstName = new Label("First Name *");
		mainFlexTable.setWidget(5, 0, lblFirstName);

		textBoxFirstName = new TextBox();
		mainFlexTable.setWidget(5, 1,
textBoxFirstName);

		Label lblMiddleName = new Label("Middle Name
*");
		mainFlexTable.setWidget(6, 0, lblMiddleName);

		textBoxMiddleName = new TextBox();
		mainFlexTable.setWidget(6, 1,
textBoxMiddleName);

		Label lblLastName = new Label("Last Name *");
		mainFlexTable.setWidget(7, 0, lblLastName);

		textBoxLastName = new TextBox();
		mainFlexTable.setWidget(7, 1,
textBoxLastName);

		Label lblEmail = new Label("Email *");
		mainFlexTable.setWidget(8, 0, lblEmail);

		textBoxEmail = new TextBox();
		mainFlexTable.setWidget(8, 1, textBoxEmail);

		lblEmailMsg = new Label("");
		mainFlexTable.setWidget(8, 2, lblEmailMsg);

		Label lblAffiliation = new Label("Affiliation");
		mainFlexTable.setWidget(9, 0, lblAffiliation);

		textBoxAffiliation = new TextBox();
		mainFlexTable.setWidget(9, 1,
textBoxAffiliation);

		Label lblSystemAdministrator = new
Label("System Administrator *");
		lblSystemAdministrator.setWordWrap(false);
		mainFlexTable.setWidget(10, 0,
lblSystemAdministrator);

		sysAdBox = new ListBox();
		sysAdBox.addItem("Yes");
		sysAdBox.addItem("No");
		sysAdBox.setSelectedIndex(0);
		mainFlexTable.setWidget(10, 1, sysAdBox);

		Label lblToolAdministrator = new Label("Tool
Administrator *");
		mainFlexTable.setWidget(11, 0,
lblToolAdministrator);

		toolAdBox = new ListBox();
		toolAdBox.addItem("Yes");
		toolAdBox.addItem("No");
		mainFlexTable.setWidget(11, 1, toolAdBox);
```

```
                lblSecurityQuestion = new Label("Security
Question *");
                mainFlexTable.setWidget(12, 0,
lblSecurityQuestion);

                questionBox = new TextBox();
                questionBox.setWidth("250px");
                mainFlexTable.setWidget(12, 1, questionBox);

                lblAnswer = new Label("Answer *");
                mainFlexTable.setWidget(13, 0, lblAnswer);

                answerBox = new TextBox();
                mainFlexTable.setWidget(13, 1, answerBox);

                FlexTable buttonTable = new FlexTable();
                mainFlexTable.setWidget(14, 1, buttonTable);

                updateButton = new Button("New button");
                updateButton.setText("Update");
                buttonTable.setWidget(0, 0, updateButton);

                cancelButton = new Button("New button");
                cancelButton.setText("Cancel");
                buttonTable.setWidget(0, 1, cancelButton);

                lblErrorMsg = new Label("");
                mainFlexTable.setWidget(15, 1, lblErrorMsg);

        mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 3);
        }
        public void setErrorTxt(String errMsg){
                this.lblErrorMsg.setStyleName("errorMsg");
                this.lblErrorMsg.setText(errMsg);
        }

        public void setEmailMsgTxt(String emailMsg){
                this.lblEmailMsg.setStyleName("errorMsg");
                this.lblEmailMsg.setText(emailMsg);
        }

        public void setRetypePasswdMsgTxt(String msg){

        this.lblRetypePasswdMsg.setStyleName("errorMsg");
                this.lblRetypePasswdMsg.setText(msg);
        }

        public HasClickHandlers getUpdateButton(){
                return updateButton;
        }

        public HasClickHandlers getCancelButton(){
                return cancelButton;
        }

        public HasClickHandlers getChangeAction(){
                return chckbxChange;
        }

        public String getUsername(){
                return this.lblUsernameTxt.getText();
        }

        public void setUserName(String name){
                this.lblUsernameTxt.setText(name);
        }

        public String getPassword(){
                return this.textBoxPassword.getText();
        }

        public String getRetypedPassword(){
                return this.textBoxRetypePassword.getText();
        }

        public String getFirstName(){
                return this.textBoxFirstName.getText();
        }

        public void setFirstName(String name){
                this.textBoxFirstName.setText(name);
        }

        public String getMiddleName(){
                return this.textBoxMiddleName.getText();
        }

        public void setMiddleName(String name){
                this.textBoxMiddleName.setText(name);
        }

        public String getLastName(){
                return this.textBoxLastName.getText();
        }

        public void setLastName(String name){
                this.textBoxLastName.setText(name);
        }

        public String getEmail(){
                return this.textBoxEmail.getText();
        }

        public void setEmail(String name){
                this.textBoxEmail.setText(name);
        }

        public String getAffiliation(){
                return this.textBoxAffiliation.getText();
        }

        public void setAffiliation(String name){
                this.textBoxAffiliation.setText(name);
        }

        public void setSysAdPriv(int selectedIndex){
                this.sysAdBox.setSelectedIndex(selectedIndex);
        }

        public String getSysAdPriv(){
                return
sysAdBox.getItemText(sysAdBox.getSelectedIndex());
        }

        public void setToolAdPriv(int selectedIndex){
                this.toolAdBox.setSelectedIndex(selectedIndex);
        }

        public String getToolAdPriv(){
                return
toolAdBox.getItemText(toolAdBox.getSelectedIndex());
        }

        public void setQuestion(String text){
                questionBox.setText(text);
        }

        public void setAnswer(String text){
                answerBox.setText(text);
        }

        public String getQuestion(){
                return questionBox.getText();
        }

        public String getAnswer(){
                return answerBox.getText();
        }

        public Boolean willChangePasswd(){
                return this.chckbxChange.getValue();
        }

        public void enablePassChange(Boolean willChange){
                if(willChange){
                        textBoxPassword.setEnabled(true);

        textBoxRetypePassword.setEnabled(true);
                        textBoxPassword.setText("");
                        textBoxRetypePassword.setText("");
```

```
                }else{
                        textBoxPassword.setEnabled(false);

        textBoxRetypePassword.setEnabled(false);
                }
        }

        public Widget asWidget(){
                return this;
        }
}
```

ForgotPasswordView.java

```
package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.DeckPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.PasswordTextBox;
import com.google.gwt.user.client.ui.Widget;

public class ForgotPasswordView extends Composite implements
cepir.client.presenter.ForgotPasswordPresenter.Display{
        private DeckPanel deckPanel;
        private TextBox usernameBox;
        private Button submitButton;
        private Label lblEmailError;
        private TextBox answerBox;
        private Label lblQuestion;
        private Button submitButton2;
        private Label lblAnswerError;
        private PasswordTextBox passwordBox;
        private PasswordTextBox retypePassBox;
        private Button saveButton;
        private Button cancelButton;
        private Label lblSaveError;
        private DialogBox dialogBox;
        private Button okButton;

        public ForgotPasswordView() {

                FlexTable mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);

        mainFlexTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75))+"px");

                Label lblForgotPassword = new Label("Forgot
Password?");
                lblForgotPassword.setStyleName("header4");
                mainFlexTable.setWidget(0, 0,
lblForgotPassword);

                deckPanel = new DeckPanel();
                mainFlexTable.setWidget(1, 0, deckPanel);

                FlexTable getEmailTable = new FlexTable();
                deckPanel.add(getEmailTable);

                Label lblPleaseProvideYour = new Label("Please
provide your username to be able to reset your password");
                getEmailTable.setWidget(0, 0,
lblPleaseProvideYour);

                HTML html = new HTML("<br/>", true);
                getEmailTable.setWidget(1, 0, html);

                Label lblEnterEmailAddress = new Label("Enter
username");
                getEmailTable.setWidget(2, 0,
lblEnterEmailAddress);
                getEmailTable.getCellFormatter().setWidth(3, 0,
"250px");

        getEmailTable.getFlexCellFormatter().setColSpan(0, 0, 1);

                usernameBox = new TextBox();
                usernameBox.setWidth("250px");
                usernameBox.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        submitButton.click();
                                }
                        }
                });
                getEmailTable.setWidget(3, 0, usernameBox);

                submitButton = new Button("Submit");
                getEmailTable.setWidget(4, 0, submitButton);

                lblEmailError = new Label();
                lblEmailError.setStyleName("errorMsg");
                getEmailTable.setWidget(5, 0, lblEmailError);

                FlexTable answerQuestionTable = new
FlexTable();
                deckPanel.add(answerQuestionTable);

                Label lblPleaseAnswerYour = new Label("Please
answer your security question");
                answerQuestionTable.setWidget(0, 0,
lblPleaseAnswerYour);

        answerQuestionTable.getFlexCellFormatter().setColSpan(0, 0,
2);

                HTML html_1 = new HTML("<br/>", true);
                answerQuestionTable.setWidget(1, 0, html_1);

                lblQuestion = new Label("");
                lblQuestion.setStyleName("header3");
                answerQuestionTable.setWidget(2, 0,
lblQuestion);

        answerQuestionTable.getFlexCellFormatter().setColSpan(2, 0,
2);

                HTML html_3 = new HTML("<br/>", true);
                answerQuestionTable.setWidget(3, 0, html_3);

                Label lblAnswer = new Label("Answer");
                answerQuestionTable.setWidget(4, 0, lblAnswer);

        answerQuestionTable.getColumnFormatter().setWidth(0,
"80px");

                answerBox = new TextBox();
                answerBox.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        submitButton2.click();
                                }
                        }
                });
                answerQuestionTable.setWidget(4, 1, answerBox);

                submitButton2 = new Button("Submit");
                answerQuestionTable.setWidget(5, 1,
submitButton2);
```

```
                lblAnswerError = new Label();
                lblAnswerError.setStyleName("errorMsg");
                answerQuestionTable.setWidget(6,1,
lblAnswerError);

                FlexTable resetPwdTable = new FlexTable();
                deckPanel.add(resetPwdTable);

                Label lblCongratulationsYouHave = new
Label("Congratulations! You have correctly answered your security
question. " +
                                "You can now enter your
new password.");
                resetPwdTable.setWidget(0, 0,
lblCongratulationsYouHave);

                HTML html_2 = new HTML("<br/>", true);
                resetPwdTable.setWidget(1, 0, html_2);

                Label lblYourNewPassword = new Label("Your
new password");
                resetPwdTable.setWidget(2, 0,
lblYourNewPassword);

                passwordBox = new PasswordTextBox();
                passwordBox.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        saveButton.click();
                                }
                        }
                });
                resetPwdTable.setWidget(2, 1, passwordBox);

                Label lblRetypeNewPassword = new Label("Re-
type new password");
                resetPwdTable.setWidget(3, 0,
lblRetypeNewPassword);

        resetPwdTable.getFlexCellFormatter().setColSpan(0, 0, 2);
                resetPwdTable.getColumnFormatter().setWidth(0,
"150px");

                retypePassBox = new PasswordTextBox();
                retypePassBox.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        saveButton.click();
                                }
                        }
                });
                resetPwdTable.setWidget(3, 1, retypePassBox);

                FlexTable button = new FlexTable();
                resetPwdTable.setWidget(4, 1, button);

                saveButton = new Button("Save");
                button.setWidget(0, 0, saveButton);

                cancelButton = new Button("Cancel");
                button.setWidget(0, 1, cancelButton);

                lblSaveError = new Label();
                lblSaveError.setStyleName("errorMsg");
                resetPwdTable.setWidget(5, 1, lblSaveError);

                deckPanel.showWidget(0);

                //ok dialog box
                dialogBox = new DialogBox();
                dialogBox.setAnimationEnabled(true);
                dialogBox.setGlassEnabled(true);
                dialogBox.setWidth("250px");
                dialogBox.setText("Password saved");

                FlexTable dialog = new FlexTable();
                dialogBox.setWidget(dialog);
                dialog.setWidget(0, 0, new Label("Your password
was successfully updated"));

                okButton = new Button("OK");
                dialog.setWidget(1, 0, okButton);
        }

        public String getEmailBox(){
                return usernameBox.getText();
        }

        public String getAnswerBox(){
                return answerBox.getText();
        }

        public String getPasswd(){
                return passwordBox.getText();
        }

        public String getRetypedPwd(){
                return retypePassBox.getText();
        }

        public void setEmailError(String text){
                lblEmailError.setText(text);
        }

        public void setAnswerError(String text){
                lblAnswerError.setText(text);
        }

        public void setSaveError(String text){
                lblSaveError.setText(text);
        }

        public void setQuestion(String text){
                lblQuestion.setText(text);
        }

        public HasClickHandlers getEmailSubmitButton(){
                return submitButton;
        }

        public HasClickHandlers getAnswerSubmitButton(){
                return submitButton2;
        }

        public HasClickHandlers getSaveButton(){
                return saveButton;
        }

        public HasClickHandlers getCancelButton(){
                return cancelButton;
        }

        public HasClickHandlers getOkButton(){
                return okButton;
        }

        public void showDialog(){
                dialogBox.center();
                dialogBox.show();
        }

        public void hideDialog(){
                dialogBox.hide();
        }

        public void setSelectedDeck(int index){
                deckPanel.showWidget(index);
        }
```

```
	public Widget asWidget(){
		return this;
	}

}
```

HomeForMemberView.java

```
package cepir.client.view;

import cepir.client.ImageResources;
import cepir.client.presenter.HomeForMemberPresenter;

import com.google.gwt.core.client.GWT;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratorPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Widget;

public class HomeForMemberView extends Composite implements
HomeForMemberPresenter.Display{
	private Label lblPendingProjCreation;
	private Label lblPendingProjMem;
	private Label lblNeedsApproval;
	private Label lblWelcome;
	private DecoratorPanel approvalDecoratorPanel;
	private Hyperlink creationCheckLink;
	private Hyperlink memCheckLink;
	private Hyperlink hprlnkManage;


	public HomeForMemberView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		lblWelcome = new Label();
		lblWelcome.setStyleName("header");
		mainFlexTable.setWidget(0, 0, lblWelcome);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

	mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(0, 0, HasHorizontalAlignment.ALIGN_CENTER);

		ImageResources images =
GWT.create(ImageResources.class);

		DecoratorPanel pendingDecoratorPanel = new
DecoratorPanel();
		mainFlexTable.setWidget(1, 0,
pendingDecoratorPanel);

		FlexTable pendingTable = new FlexTable();
		pendingTable.setHeight("100px");

	pendingTable.setWidth(Integer.toString((int)(Window.getClien
tWidth()*0.8*0.75*0.5))+"px");
		pendingDecoratorPanel.setWidget(pendingTable);

		Image flag = new Image(images.flag().getUrl());
		pendingTable.setWidget(0, 0, flag);

	pendingTable.getFlexCellFormatter().setRowSpan(0, 0, 2);

		Label lblPendingProjectCreation = new
Label("Pending project creation requests:");
		pendingTable.setWidget(0, 1,
lblPendingProjectCreation);

		lblPendingProjCreation = new Label("");
		pendingTable.setWidget(0, 2,
lblPendingProjCreation);
		lblPendingProjCreation.setStyleName("header4");

		creationCheckLink = new Hyperlink("Manage",
false, "pendingProjReq?list=0");
		creationCheckLink.setVisible(false);
		pendingTable.setWidget(0, 3, creationCheckLink);

		Label lblPendingProjectMembership = new
Label("Pending project membership requests:");
		pendingTable.setWidget(1, 0,
lblPendingProjectMembership);

		lblPendingProjMem = new Label("");
		pendingTable.setWidget(1, 1,
lblPendingProjMem);
		lblPendingProjMem.setStyleName("header4");

		memCheckLink = new Hyperlink("Manage", false,
"pendingProjReq?list=1");
		memCheckLink.setVisible(false);
		pendingTable.setWidget(1, 2, memCheckLink);

		approvalDecoratorPanel = new DecoratorPanel();
		approvalDecoratorPanel.setVisible(false);
		mainFlexTable.setWidget(1, 1,
approvalDecoratorPanel);

		FlexTable approvalTable = new FlexTable();
		approvalTable.setHeight("100px");

	approvalTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75*0.5))+"px");

	approvalDecoratorPanel.setWidget(approvalTable);

		Label lblMembershipRequestsThat = new
Label("Membership requests that need approval:");
		approvalTable.setWidget(0, 1,
lblMembershipRequestsThat);

		Image gear = new Image(images.gear().getUrl());
		approvalTable.setWidget(0, 0, gear);

		lblNeedsApproval = new Label("");
		approvalTable.setWidget(0, 2, lblNeedsApproval);
		lblNeedsApproval.setStyleName("header4");

		hprlnkManage = new Hyperlink("Manage", false,
"allPendingReq");
		hprlnkManage.setVisible(false);
		approvalTable.setWidget(0, 3, hprlnkManage);
	}

	public void setPendingProjCreationReq(String count){
		lblPendingProjCreation.setText(count);

	creationCheckLink.setVisible((Integer.valueOf(count)==0?fals
e:true));
	}

	public void setPendingProjMemReq(String count){
		lblPendingProjMem.setText(count);

	memCheckLink.setVisible((Integer.valueOf(count)==0?false:tr
ue));
	}

	public void setNeedsApproval(String count){
		lblNeedsApproval.setText(count);

	hprlnkManage.setVisible((Integer.valueOf(count)==0?false:tru
e));
	}

	public void setVisibilityApprovalPanel(Boolean isVisible){
		approvalDecoratorPanel.setVisible(isVisible);
	}

	public void setWelcomeMsg(String user){
		lblWelcome.setText("Welcome "+user+"!");
	}
```

```
	public Widget asWidget(){
		return this;
	}

}
```

HomeForSysAdView.java

```
package cepir.client.view;

import cepir.client.ImageResources;
import cepir.client.presenter.HomeForSysAdPresenter;

import com.google.gwt.core.client.GWT;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.DecoratorPanel;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Widget;

public class HomeForSysAdView extends Composite implements
HomeForSysAdPresenter.Display{
	private Label lblWelcome;
	private Label lblPendingUsers;
	private Label lblPendingProjs;

	public HomeForSysAdView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		lblWelcome = new Label("");

	lblWelcome.setHorizontalAlignment(HasHorizontalAlignment.
ALIGN_CENTER);
		lblWelcome.setStyleName("header");
		mainFlexTable.setWidget(0, 0, lblWelcome);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 1);

		DecoratorPanel userDecoratorPanel = new
DecoratorPanel();
		mainFlexTable.setWidget(1, 0,
userDecoratorPanel);

		FlexTable userTable = new FlexTable();
		userDecoratorPanel.setWidget(userTable);

		ImageResources images =
GWT.create(ImageResources.class);
		Image user = new Image(images.user().getUrl());
		userTable.setWidget(0, 0, user);
		userTable.getFlexCellFormatter().setRowSpan(0,
0, 2);

		FlexTable userNotifier = new FlexTable();
		userTable.setWidget(0, 1, userNotifier);
		lblPendingUsers = new Label();
		lblPendingUsers.setStyleName("header4");
		userNotifier.setWidget(0, 0, lblPendingUsers);
		userNotifier.setWidget(0, 1, new Label(" user
account request/s need approval"));

		Hyperlink pendingUsers = new Hyperlink("Click
here to manage pending user/s", "userMngt?tab=1");
		userTable.setWidget(1, 0, pendingUsers);

	userTable.getFlexCellFormatter().setHorizontalAlignment(1, 0,
HasHorizontalAlignment.ALIGN_CENTER);

	userTable.getCellFormatter().setHorizontalAlignment(0, 1,
HasHorizontalAlignment.ALIGN_CENTER);

		DecoratorPanel projdecoratorPanel = new
DecoratorPanel();
		mainFlexTable.setWidget(1, 1,
projdecoratorPanel);

		FlexTable projTable = new FlexTable();
		projdecoratorPanel.setWidget(projTable);

		Image proj = new
Image(images.clipboard().getUrl());
		projTable.setWidget(0, 0, proj);
		projTable.getFlexCellFormatter().setRowSpan(0,
0, 2);

		FlexTable projNotifier = new FlexTable();
		projTable.setWidget(0, 1, projNotifier);
		lblPendingProjs = new Label();
		lblPendingProjs.setStyleName("header4");
		projNotifier.setWidget(0, 0, lblPendingProjs);
		projNotifier.setWidget(0, 1, new Label(" project
creation request/s need approval"));

		Hyperlink pendingProjs = new Hyperlink("Click
here to manage pending project/s", "projMngt?tab=1");
		projTable.setWidget(1, 0, pendingProjs);

	projTable.getFlexCellFormatter().setHorizontalAlignment(1, 0,
HasHorizontalAlignment.ALIGN_CENTER);

	projTable.getCellFormatter().setHorizontalAlignment(0, 1,
HasHorizontalAlignment.ALIGN_CENTER);


	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

	mainFlexTable.getCellFormatter().setHorizontalAlignment(1,
0, HasHorizontalAlignment.ALIGN_RIGHT);

	mainFlexTable.getCellFormatter().setVerticalAlignment(1, 0,
HasVerticalAlignment.ALIGN_TOP);

	mainFlexTable.getCellFormatter().setVerticalAlignment(1, 1,
HasVerticalAlignment.ALIGN_TOP);
	}

	public void setWelcomeMsg(String username){
		lblWelcome.setText("Welcome "+username+"!");
	}

	public void setNoOfPendingUsers(String pendingUsers){
		lblPendingUsers.setText(pendingUsers);
	}

	public void setNoOfPendingProjs(String pendingProjs){
		lblPendingProjs.setText(pendingProjs);
	}

	public Widget asWidget(){
		return this;
	}

}
```

IntroMsgView.java

```
package cepir.client.view;

import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Widget;

public class IntroMsgView extends Composite implements
cepir.client.presenter.IntroMsgPresenter.Display{

	public IntroMsgView() {
		Label introMsg = new Label("Welcome to the
Collaboratory for Epidemiological Research (CEpiR)!");
		initWidget(introMsg);
		introMsg.addStyleName("header");
	}

	public Widget asWidget(){
```

```
                return this;
        }

}
```

LoginPageView.java

```
package cepir.client.view;

import cepir.client.presenter.LoginPagePresenter;

import com.google.gwt.user.client.ui.*;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;

public class LoginPageView extends Composite implements
LoginPagePresenter.Display{
        private TextBox txtUsername;
        private Label lblError;
        private PasswordTextBox txtPassword;
        private Button btnSignIn;
        private Hyperlink hprlnkNotAMember;
        private Hyperlink hprlnkForgotPassword;

        public LoginPageView() {

                FlexTable flexTable = new FlexTable();
                initWidget(flexTable);

                Label lblLogin = new Label("Login");
                lblLogin.setStyleName("header2");
                flexTable.setWidget(0, 0, lblLogin);
                flexTable.getFlexCellFormatter().setColSpan(0, 0,
2);

                Label lblUsername = new Label("Username");
                flexTable.setWidget(1, 0, lblUsername);

                txtUsername = new TextBox();
                txtUsername.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        btnSignIn.click();
                                }
                        }
                });
                flexTable.setWidget(1, 1, txtUsername);

                Label lblPassword = new Label("Password");
                flexTable.setWidget(2, 0, lblPassword);

                btnSignIn = new Button("Sign In");
                txtPassword = new PasswordTextBox();
                txtPassword.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        btnSignIn.click();
                                }
                        }
                });
                flexTable.setWidget(2, 1, txtPassword);
                flexTable.setWidget(3, 1, btnSignIn);

                lblError = new Label("");
                lblError.setStyleName("errorMsg");
                flexTable.setWidget(4, 1, lblError);

                hprlnkNotAMember = new Hyperlink("Not a
member yet?", false, "requestForAcct");
                hprlnkNotAMember.addStyleName("link");
                flexTable.setWidget(5, 1, hprlnkNotAMember);

                hprlnkForgotPassword = new Hyperlink("Forgot
password?", false, "forgotPass");
                flexTable.setWidget(6, 1, hprlnkForgotPassword);
        }

        public HasClickHandlers getSignInButton(){
                return btnSignIn;
        }

        public Widget asWidget() {
                return this;
        }

        public String getUsername(){
                return txtUsername.getText();
        }

        public String getPassword(){
                return txtPassword.getText();
        }

        public void setErrorTxt(String errMsg){
                this.lblError.setText(errMsg);
        }
}
```

MainView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.FlexTable;

public class MainView extends Composite implements
cepir.client.presenter.MainPresenter.Display{
        private FlexTable center;
        private FlexTable header;
        private Button goButton;
        private ListBox list;
        private PushButton options;

        public MainView() {

                FlexTable mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);

        mainFlexTable.setSize(Integer.toString(Window.getClientWidt
h()), Integer.toString(Window.getClientHeight()));
                mainFlexTable.setStyleName("flexTable");

                FlexTable leftside = new FlexTable();
                mainFlexTable.setWidget(0, 0, leftside);

        mainFlexTable.getFlexCellFormatter().setHeight(0, 0,
Integer.toString(Window.getClientHeight())+"px");
                mainFlexTable.getFlexCellFormatter().setWidth(0,
0, "10%");
                leftside.setSize("100%", "100%");

                center = new FlexTable();
                center.setSize("100%", "100%");
```

```
		mainFlexTable.getFlexCellFormatter().setHeight(0, 1,
Integer.toString(Window.getClientHeight())+"px");
			mainFlexTable.getFlexCellFormatter().setWidth(0,
1, "80%");
			mainFlexTable.setWidget(0, 1, center);

			ImageResources images =
GWT.create(ImageResources.class);
			Image banner = new
Image(images.siteBanner().getUrl());
			banner.setAltText("Collaboratory for
Epidemiological Research");

		banner.setPixelSize((int)(Window.getClientWidth()*0.8), 150);
			center.setWidget(1, 0, banner);
			center.getFlexCellFormatter().setColSpan(1, 0, 2);
			center.addStyleName("center");

			FlexTable rightside = new FlexTable();

		mainFlexTable.getFlexCellFormatter().setHeight(0, 2,
Integer.toString(Window.getClientHeight())+"px");
			mainFlexTable.getFlexCellFormatter().setWidth(0,
2, "10%");
			mainFlexTable.setWidget(0, 2, rightside);
			rightside.setSize("100%", "100%");

			center.getFlexCellFormatter().setWidth(2, 0,
Integer.toString((int)(Window.getClientWidth()*0.8*0.25))+"px");
			center.getFlexCellFormatter().setHeight(2, 1,
Integer.toString(Window.getClientHeight()-banner.getHeight()));

			header = new FlexTable();
			header.setWidget(0, 0, new Label("Choose a
project:"));
			list = new ListBox(false);
			list.setWidth("250px");
			header.setWidget(0, 1, list);

			goButton = new Button("Go");
			header.setWidget(0, 2, goButton);

			options = new PushButton(new
Image(images.options().getUrl()));
			options.setStyleName("icon");
			options.setTitle("Edit account");
			header.setWidget(0, 3, options);

			center.setWidget(0, 0, header);
			center.getFlexCellFormatter().setColSpan(0, 0, 2);

		center.getFlexCellFormatter().setHorizontalAlignment(0, 0,
HasHorizontalAlignment.ALIGN_RIGHT);
			header.setVisible(false);
		}

		public void setPage(Widget widget){;
			center.setWidget(2, 1, widget);

		center.getFlexCellFormatter().setHorizontalAlignment(2, 1,
HasHorizontalAlignment.ALIGN_LEFT);

		center.getFlexCellFormatter().setVerticalAlignment(2, 1,
HasVerticalAlignment.ALIGN_TOP);
		}

		public void setSideMenu(Widget widget){
			center.setWidget(2, 0, widget);

		center.getFlexCellFormatter().setHorizontalAlignment(2, 0,
HasHorizontalAlignment.ALIGN_LEFT);

		center.getFlexCellFormatter().setVerticalAlignment(2, 0,
HasVerticalAlignment.ALIGN_TOP);
		}

		public void setCenterHeader(ArrayList<Project> projects){
			list.clear();
			for(int i = 0; i < projects.size(); i++){
				String[] parse =
projects.get(i).projName.split("=");
				list.addItem(parse[0]);
			}
			list.setSelectedIndex(-1);
			header.setVisible(true);
		}

		public void hideHeader(){
			header.setVisible(false);
		}

		public HasClickHandlers getGoButton(){
			return goButton;
		}

		public HasClickHandlers getOptionsButton(){
			return options;
		}

		public Integer getSelectedItem(){
			return list.getSelectedIndex();
		}

		public void setSelectedItem(Integer index){
			list.setSelectedIndex(index);
		}

		public void clearList(){
			list.clear();
		}

		public Widget asWidget(){
			return this;
		}
}
```

MemberMngtView.java

```
package cepir.client.view;

import cepir.client.presenter.MemberMngtPresenter;

import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedTabPanel;
import com.google.gwt.user.client.ui.Widget;

public class MemberMngtView extends Composite implements
MemberMngtPresenter.Display{

		public DecoratedTabPanel decoratedTabPanel;

		public MemberMngtView(){
			decoratedTabPanel = new DecoratedTabPanel();
			initWidget(decoratedTabPanel);
		}

		public void addTab(Widget widget, String tabName){
			decoratedTabPanel.add(widget, tabName, false);
		}

		public void setSelectedTab(int index){
			decoratedTabPanel.selectTab(index);
		}

		public HasSelectionHandlers<Integer> getTabPanel(){
			return decoratedTabPanel;
		}

		public Widget asWidget() {
			return this;
		}

		public void setVisibilityTabs(Boolean isVisible){
			decoratedTabPanel.setVisible(isVisible);
		}

}
```

MemListTabView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.MemListTabPresenter;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Widget;

public class MemListTabView extends Composite implements
MemListTabPresenter.Display{
	private FlexTable memListTable;
	private PushButton addMemButton;
	private PushButton addAdminButton;
	private PushButton editButton;
	private PushButton deleteButton;
	private PushButton refreshButton;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private PushButton nextButton;
	private PushButton prevButton;
	private PushButton firstPageButton;
	private PushButton lastPageButton;
	private TextBox search;
	private IntegerBox intBoxUsersPerPage;
	private Label pageNumber;
	private Button usersPerPageButton;
	private FlexTable buttonTable;
	private DialogBox deleteWarning;
	private Button cantDeleteOkButton;
	private DialogBox selectMembers;
	private FlexTable selectMembersTable;
	private Button selectOkButton;
	private Button selectCancelButton;
	private Button selectProjMemberButton;
	private Button deselectProjMemberButton;
	private ListBox membersList;
	private ListBox usersList;
	private PushButton searchInAddButton;
	private PushButton cancelSearchInAddButton;
	private TextBox searchInAdd;

	public MemListTabView() {

		FlexTable mainTable = new FlexTable();
		initWidget(mainTable);

	mainTable.setSize(Integer.toString((int)(Window.getClientWid
th()*0.8*0.75))+"px",

	Integer.toString((int)(Window.getClientHeight()-150))+"px");
//150->banner size

		ImageResources images =
GWT.create(ImageResources.class);
		memListTable = new FlexTable();
		mainTable.setWidget(0, 0, memListTable);

	mainTable.getFlexCellFormatter().setVerticalAlignment(0, 0,
HasVerticalAlignment.ALIGN_TOP);

		buttonTable = new FlexTable();
		buttonTable.setVisible(false);
		memListTable.setWidget(0, 1, buttonTable);

		addMemButton = new PushButton(new
Image(images.member().getUrl()));
		addMemButton.setTitle("Add Project Member");
		addMemButton.setStyleName("icon");
		buttonTable.setWidget(0, 0, addMemButton);

		addAdminButton = new PushButton(new
Image(images.admin().getUrl()));
		addAdminButton.setTitle("Add Project Admin");
		addAdminButton.setStyleName("icon");
		buttonTable.setWidget(0, 1, addAdminButton);

		editButton = new PushButton(new
Image(images.edit().getUrl()));
		editButton.setTitle("Edit membership");
		editButton.setStyleName("icon");
		buttonTable.setWidget(0, 2, editButton);

		deleteButton = new PushButton(new
Image(images.delete().getUrl()));
		deleteButton.setStyleName("icon");
		deleteButton.setTitle("Delete as project member");
		buttonTable.setWidget(0, 3, deleteButton);

		refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
		refreshButton.setTitle("Refresh");
		refreshButton.setStyleName("icon");
		buttonTable.setWidget(0, 4, refreshButton);

		FlexTable searchTable = new FlexTable();
		memListTable.setWidget(0, 3, searchTable);

	memListTable.getFlexCellFormatter().setColSpan(0, 3, 2);

	memListTable.getCellFormatter().setHorizontalAlignment(0, 3,
HasHorizontalAlignment.ALIGN_RIGHT);

		search = new TextBox();
		searchTable.setWidget(0, 0, search);

		searchButton = new PushButton(new
Image(images.search().getUrl()));
		searchButton.setTitle("Search");
		searchButton.setStyleName("icon");
		searchTable.setWidget(0, 1, searchButton);

		cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
		cancelSearchButton.setStyleName("icon");
		cancelSearchButton.setTitle("Cancel");
		searchTable.setWidget(0, 2, cancelSearchButton);

		Label lblUsername = new Label("Username");
		memListTable.setWidget(1, 1, lblUsername);
		memListTable.getFlexCellFormatter().setWidth(1,
1, Integer.toString((int)(Window.getClientWidth()*0.8*0.2))+"px");

		Label lblFirstName = new Label("First Name");
		memListTable.setWidget(1, 2, lblFirstName);
		memListTable.getFlexCellFormatter().setWidth(1,
2, Integer.toString((int)(Window.getClientWidth()*0.8*0.2))+"px");

		Label lblMiddleName = new Label("Middle
Name");
		memListTable.setWidget(1, 3, lblMiddleName);

	memListTable.getFlexCellFormatter().setColSpan(0, 1, 2);
		memListTable.getFlexCellFormatter().setWidth(1,
3, Integer.toString((int)(Window.getClientWidth()*0.8*0.2))+"px");

		Label lblLastName = new Label("Last Name");
```

```
                memListTable.setWidget(1, 4, lblLastName);
                memListTable.getFlexCellFormatter().setWidth(1,
4, Integer.toString((int)(Window.getClientWidth()*0.8*0.2))+"px");

                Label lblPosition = new Label("Position");
                memListTable.setWidget(1, 5, lblPosition);
                memListTable.getFlexCellFormatter().setWidth(1,
5, Integer.toString((int)(Window.getClientWidth()*0.8*0.2))+"px");


        memListTable.getRowFormatter().setStyleName(1,
"tableHeader");

                //paginator
                FlexTable paginator = new FlexTable();
                mainTable.setWidget(1, 0, paginator);

        mainTable.getFlexCellFormatter().setHorizontalAlignment(1,
0, HasHorizontalAlignment.ALIGN_CENTER);

                firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
                firstPageButton.setStyleName("icon");
                firstPageButton.setTitle("Go to first page");
                paginator.setWidget(0,0,firstPageButton);

                prevButton = new PushButton(new
Image(images.previous().getUrl()));
                prevButton.setStyleName("icon");
                prevButton.setTitle("Previous");
                paginator.setWidget(0,1,prevButton);

                pageNumber = new Label("");
                paginator.setWidget(0,2,pageNumber);

                nextButton = new PushButton(new
Image(images.next().getUrl()));
                nextButton.setStyleName("icon");
                nextButton.setTitle("Next");
                paginator.setWidget(0,3,nextButton);

                lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                lastPageButton.setStyleName("icon");
                lastPageButton.setTitle("Go to last page");
                paginator.setWidget(0, 4, lastPageButton);

                //users per page
                FlexTable usersPerPage = new FlexTable();
                mainTable.setWidget(2, 0, usersPerPage);

        mainTable.getFlexCellFormatter().setHorizontalAlignment(2,
0, HasHorizontalAlignment.ALIGN_CENTER);

                Label lblUsersPerPage = new Label("No. of users
per page");
                usersPerPage.setWidget(0, 0, lblUsersPerPage);

                intBoxUsersPerPage = new IntegerBox();
                intBoxUsersPerPage.setWidth("30px");
                intBoxUsersPerPage.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        usersPerPageButton.click();
                                }
                        }
                });
                usersPerPage.setWidget(0, 1,
intBoxUsersPerPage);

                usersPerPageButton = new Button("Go");
                usersPerPage.setWidget(0, 2,
usersPerPageButton);

                //cannot delete own account dialogBox
                deleteWarning = new DialogBox();
                deleteWarning.setGlassEnabled(true);
                deleteWarning.setAnimationEnabled(true);
                deleteWarning.setText("Delete user error");

                FlexTable warningContents = new FlexTable();
                deleteWarning.setWidget(warningContents);

                Image warningIcon = new
Image(images.warning().getUrl());
                warningContents.setWidget(0, 0, warningIcon);

        warningContents.getFlexCellFormatter().setRowSpan(0, 0, 2);

                Label warningTxt = new Label("You cannot delete
your own account");
                warningContents.setWidget(0, 1, warningTxt);

        warningContents.getFlexCellFormatter().setHorizontalAlignme
nt(0, 1, HasHorizontalAlignment.ALIGN_CENTER);

        warningContents.getFlexCellFormatter().setColSpan(0, 1, 3);

                cantDeleteOkButton = new Button("Ok");
                warningContents.setWidget(1, 1,
cantDeleteOkButton);

        warningContents.getFlexCellFormatter().setHorizontalAlignme
nt(1, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                //add admin/member dialog box
                selectMembers = new DialogBox();
                selectMembers.setAnimationEnabled(true);
                selectMembers.setGlassEnabled(true);

                selectMembersTable = new FlexTable();
                selectMembers.setWidget(selectMembersTable);

                membersList = new ListBox(true);
                membersList.setWidth("250px");
                membersList.setHeight("150px");
                selectMembersTable.setWidget(1, 0,
membersList);

                FlexTable center = new FlexTable();
                selectMembersTable.setWidget(1, 1, center);
                selectProjMemberButton = new Button("<");
                center.setWidget(0, 0, selectProjMemberButton);

                deselectProjMemberButton = new Button(">");
                center.setWidget(1, 0,
deselectProjMemberButton);

                selectMembersTable.setWidget(0, 2, new
Label("Users"));

                FlexTable searchFlexTable = new FlexTable();
                selectMembersTable.setWidget(0, 3,
searchFlexTable);

        selectMembersTable.getFlexCellFormatter().setHorizontalAlig
nment(0, 3, HasHorizontalAlignment.ALIGN_RIGHT);

                searchInAdd = new TextBox();
                searchFlexTable.setWidget(0, 0, searchInAdd);

                searchInAddButton = new PushButton(new
Image(images.search().getUrl()));
                searchInAddButton.setStyleName("icon");
                searchInAddButton.setTitle("Search");
                searchFlexTable.setWidget(0, 1,
searchInAddButton);

                cancelSearchInAddButton = new PushButton(new
Image(images.cancel().getUrl()));
                cancelSearchInAddButton.setStyleName("icon");
                cancelSearchInAddButton.setTitle("Cancel");
                searchFlexTable.setWidget(0, 2,
cancelSearchInAddButton);
```

```
                usersList = new ListBox(true);
                usersList.setWidth("250px");
                usersList.setHeight("150px");
                selectMembersTable.setWidget(1, 2, usersList);

        selectMembersTable.getFlexCellFormatter().setColSpan(1, 2,
2);

                FlexTable buttonTable = new FlexTable();
                selectMembersTable.setWidget(2, 0, buttonTable);

        selectMembersTable.getFlexCellFormatter().setColSpan(2, 0,
4);

        selectMembersTable.getFlexCellFormatter().setHorizontalAlig
nment(2, 0, HasHorizontalAlignment.ALIGN_CENTER);
                selectOkButton = new Button("Ok");
                buttonTable.setWidget(0, 0, selectOkButton);
                selectCancelButton = new Button("Cancel");
                buttonTable.setWidget(0, 1, selectCancelButton);
        }

        public void loadUsersInDiagList(ArrayList<User> users){
                usersList.clear();
                for(int row = 0;row<users.size();row++){

        usersList.addItem(users.get(row).username+"
("+users.get(row).firstName+" "+users.get(row).lastName+")");
                }
        }
        public void loadProjMembersList(ArrayList<User> users){
                membersList.clear();
                for(int row = 0;row<users.size();row++){

        membersList.addItem(users.get(row).username+"
("+users.get(row).firstName+" "+users.get(row).lastName+")");
                }
        }

        public ArrayList<Integer> getSelectedAdmins(){
                ArrayList<Integer> selectedRows = new
ArrayList<Integer>();
                for(int ctr = 0; ctr<membersList.getItemCount();
ctr++){
                        if(membersList.isItemSelected(ctr)){
                                selectedRows.add(ctr);
                        }
                }
                return selectedRows;
        }

        public ArrayList<Integer> getSelectedUsers(){
                ArrayList<Integer> selectedRows = new
ArrayList<Integer>();
                for(int ctr = 0; ctr<usersList.getItemCount();
ctr++){
                        if(usersList.isItemSelected(ctr)){
                                selectedRows.add(ctr);
                        }
                }
                return selectedRows;
        }

        public void showAdd(String title){
                selectMembers.center();
                selectMembers.show();
                selectMembers.setText("Select "+title);
                selectMembersTable.setWidget(0, 0, new
Label(title));
        }

        public void hideAdd(){
                selectMembers.hide();
        }

        public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
                firstPageButton.setEnabled(!isFirstPage);
                prevButton.setEnabled(!isFirstPage);
                pageNumber.setText("Page "+currPage+" of
"+totalPage);
                nextButton.setEnabled(!isLastPage);
                lastPageButton.setEnabled(!isLastPage);
        }

        public void loadUsersList(ArrayList<User> users, Boolean
isProjAdmin){
                clearRows();
                if(users.isEmpty()){
                        memListTable.setText(2, 0, "No
user/s to display");

        memListTable.getFlexCellFormatter().setColSpan(2, 0, 6);

        memListTable.getFlexCellFormatter().setHorizontalAlignment(
2, 0, HasHorizontalAlignment.ALIGN_CENTER);
                }else{
                        for(int row =
0;row<users.size();row++){
                                if(isProjAdmin){

        memListTable.setWidget(row+2, 0, new RadioButton(""));
                                }else{

        memListTable.setText(row+2, 0, " ");
                                }

        memListTable.setText(row+2, 1, users.get(row).username);

        memListTable.setText(row+2, 2, users.get(row).firstName);

        memListTable.setText(row+2, 3, users.get(row).middleName);

        memListTable.setText(row+2, 4, users.get(row).lastName);

        memListTable.setText(row+2, 5,
(users.get(row).password.equals("Project
Administrator")?"Administrator":users.get(row).password));
                        }
                }
        }

        private void clearRows(){
                int tableCount =memListTable.getRowCount();
                for(int row = tableCount-1; row>=2; row--){
                        memListTable.removeRow(row);
                }
        }

        public Integer getSelectedRow() {
                for (int i = 2; i < memListTable.getRowCount();
++i) {
                        RadioButton radioButton =
(RadioButton)memListTable.getWidget(i, 0);
                        if(radioButton.getValue()){
                                return i;
                        }
                }
                return null;
        }

        public void editUser_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Choose a
member"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public void delUser_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Choose a member
to delete"));
                warning.setPopupPosition(left, top);
                warning.show();
```

```
	}

	public void addMember_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Please select at
least 1 user"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public void usersPerPage_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Minimum no. of
users per page is 1"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public void clearRowStyles(){
		for(int row = 2; row <
memListTable.getRowCount(); row++){

	memListTable.getRowFormatter().removeStyleName(row,
"tableHighlight");
		}
	}

	public void showDeleteUserWarning(){
		deleteWarning.center();
		deleteWarning.show();
	}

	public void hideDeleteUserWarning(){
		deleteWarning.hide();
	}

	public String getSearchText(){
		return search.getText();
	}

	public void clearSearchText(){
		this.search.setText("");
	}

	public String getSearchInAddText(){
		return searchInAdd.getText();
	}

	public void clearSearchInAddText(){
		this.searchInAdd.setText("");
	}

	public Integer getUsersPerPageText(){
		return intBoxUsersPerPage.getValue();
	}

	public void clearUsersPerPageText(){
		this.intBoxUsersPerPage.setText("");
	}

	public HasClickHandlers getAddMemButton(){
		return addMemButton;
	}

	public HasClickHandlers getAddAdminButton(){
		return addAdminButton;
	}

	public HasClickHandlers getSelectOkButton(){
		return selectOkButton;
	}

	public HasClickHandlers getSelectCancelButton(){
		return selectCancelButton;
	}

public HasClickHandlers getEditButton(){
		return editButton;
}

public HasClickHandlers getDeleteButton(){
		return deleteButton;
}

public HasClickHandlers getRefreshButton(){
		return refreshButton;
}

public HasClickHandlers getSearchButton(){
		return searchButton;
}

public HasClickHandlers getSearchInAddButton(){
		return searchInAddButton;
}

public HasClickHandlers getcancelSearchButton(){
		return cancelSearchButton;
}

public HasClickHandlers getcancelSearchInAddButton(){
		return cancelSearchInAddButton;
}

public HasClickHandlers getGoToFirstPageButton(){
		return firstPageButton;
}

public HasClickHandlers getGoToLastPageButton(){
		return lastPageButton;
}

public HasClickHandlers getPrevButton(){
		return prevButton;
}

public HasClickHandlers getNextButton(){
		return nextButton;
}

public HasClickHandlers getUsersPerPageButton(){
		return usersPerPageButton;
}

public FlexTable getUsersListTable(){
		return memListTable;
}

public void setVisibilityButtons(Boolean isVisible){
		buttonTable.setVisible(isVisible);
}

public HasClickHandlers getCantDeleteOkButton(){
		return cantDeleteOkButton;
}

public HasClickHandlers getSelectMemberButton(){
		return selectProjMemberButton;
}

public HasClickHandlers getDeselectMemberButton(){
		return deselectProjMemberButton;
}

public HasKeyDownHandlers getSearchKeyHandler(){
		return search;
}

public HasKeyDownHandlers getSearchKeyHandlerInAdd(){
		return searchInAdd;
}

public Widget asWidget() {
		return this;
}
```

```
}
```

MemRequestsToApprove.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.MemRequestsToApprovePresenter;
import cepir.shared.ProjMemRequest;
import cepir.shared.User;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Widget;

public class MemRequestsToApproveView extends Composite implements
MemRequestsToApprovePresenter.Display{
	private FlexTable memRequestsTable;
	private FlexTable paginator;
	private FlexTable memReqsPerPage;
	private TextBox searchBox;
	private IntegerBox intBoxMemReqPerPage;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private PushButton nextButton;
	private PushButton prevButton;
	private PushButton firstPageButton;
	private PushButton lastPageButton;
	private Button memReqPerPageButton;
	private Label pageNumber;
	private PushButton approveButton;
	private PushButton rejectButton;
	private FlexTable buttonTable;
	private PushButton refreshButton;

	public MemRequestsToApproveView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

	mainFlexTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75))+"px");

		Label lblProjectMembershipRequests = new
Label("Project Membership Request/s");

	lblProjectMembershipRequests.setStyleName("header");
		mainFlexTable.setWidget(0, 0,
lblProjectMembershipRequests);

		memRequestsTable = new FlexTable();
		mainFlexTable.setWidget(1, 0,
memRequestsTable);

		ImageResources images =
GWT.create(ImageResources.class);

		buttonTable = new FlexTable();
		memRequestsTable.setWidget(0, 1, buttonTable);
		approveButton = new PushButton(new
Image(images.accept().getUrl()));
		buttonTable.setWidget(0, 0, approveButton);
		approveButton.setStyleName("icon");
		approveButton.setTitle("Approve Membership
Request");

		rejectButton = new PushButton(new
Image(images.delete().getUrl()));
		buttonTable.setWidget(0, 1, rejectButton);
		rejectButton.setStyleName("icon");
		rejectButton.setTitle("Reject Membership
Request");

		refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
		refreshButton.setStyleName("icon");
		refreshButton.setTitle("Refresh");
		buttonTable.setWidget(0, 2, refreshButton);

		//search Box
		FlexTable searchTable = new FlexTable();
		memRequestsTable.setWidget(0, 2, searchTable);

	memRequestsTable.getFlexCellFormatter().setColSpan(0, 2, 3);

	memRequestsTable.getFlexCellFormatter().setHorizontalAlign
ment(0, 2, HasHorizontalAlignment.ALIGN_RIGHT);

		searchBox = new TextBox();
		searchTable.setWidget(0, 0, searchBox);

		searchButton = new PushButton(new
Image(images.search().getUrl()));
		searchButton.setStyleName("icon");
		searchButton.setTitle("Search");
		searchTable.setWidget(0, 1, searchButton);

		cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
		cancelSearchButton.setStyleName("icon");
		cancelSearchButton.setTitle("Cancel Search");
		searchTable.setWidget(0, 2, cancelSearchButton);

		//table labels
		Label lblName = new Label("Name");
		memRequestsTable.setWidget(1, 1, lblName);

	memRequestsTable.getFlexCellFormatter().setWidth(1, 1,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.15))+"px");

		Label lblUsername = new Label("Username");
		memRequestsTable.setWidget(1, 2, lblUsername);

	memRequestsTable.getFlexCellFormatter().setWidth(1, 2,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.15))+"px");

		Label lblMessage = new Label("Message");
		memRequestsTable.setWidget(1, 3, lblMessage);

	memRequestsTable.getFlexCellFormatter().setWidth(1, 3,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.4))+"px");

		Label lblProject = new Label("Project");
		memRequestsTable.setWidget(1, 4, lblProject);

	memRequestsTable.getFlexCellFormatter().setWidth(1, 4,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.3))+"px");

	memRequestsTable.getRowFormatter().setStyleName(1,
"tableHeader");

	mainFlexTable.getCellFormatter().setHorizontalAlignment(1,
0, HasHorizontalAlignment.ALIGN_LEFT);

	mainFlexTable.getCellFormatter().setVerticalAlignment(1, 0,
HasVerticalAlignment.ALIGN_TOP);

		//paginator
		paginator = new FlexTable();
		mainFlexTable.setWidget(2, 0, paginator);

	mainFlexTable.getCellFormatter().setHorizontalAlignment(2,
0, HasHorizontalAlignment.ALIGN_CENTER);
```

```
                        firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
                        firstPageButton.setStyleName("icon");
                        firstPageButton.setTitle("Go to first page");
                        paginator.setWidget(0,0,firstPageButton);

                        prevButton = new PushButton(new
Image(images.previous().getUrl()));
                        prevButton.setStyleName("icon");
                        prevButton.setTitle("Previous");
                        paginator.setWidget(0,1,prevButton);

                        pageNumber = new Label("");
                        paginator.setWidget(0,2,pageNumber);

                        nextButton = new PushButton(new
Image(images.next().getUrl()));
                        nextButton.setStyleName("icon");
                        nextButton.setTitle("Next");
                        paginator.setWidget(0,3,nextButton);

                        lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                        lastPageButton.setStyleName("icon");
                        lastPageButton.setTitle("Go to last page");
                        paginator.setWidget(0, 4, lastPageButton);

                        //membership requests per page
                        memReqsPerPage = new FlexTable();
                        mainFlexTable.setWidget(3, 0,
memReqsPerPage);

                mainFlexTable.getCellFormatter().setHorizontalAlignment(3,
0, HasHorizontalAlignment.ALIGN_CENTER);

                mainFlexTable.getFlexCellFormatter().setRowSpan(1, 1, 3);

                        Label lblUsersPerPage = new Label("No. of
membership requests per page");
                        memReqsPerPage.setWidget(0, 0,
lblUsersPerPage);

                        intBoxMemReqPerPage = new IntegerBox();
                        intBoxMemReqPerPage.setWidth("30px");

                intBoxMemReqPerPage.addKeyDownHandler(new
KeyDownHandler() {

                                public void
onKeyDown(KeyDownEvent event) {

                if(event.getNativeKeyCode()==13){

                memReqPerPageButton.click();
                                                }
                                }
                        });
                        memReqsPerPage.setWidget(0, 1,
intBoxMemReqPerPage);

                        memReqPerPageButton = new Button("Go");
                        memReqsPerPage.setWidget(0, 2,
memReqPerPageButton);
                }

                public void loadRequestsList(ArrayList<ProjMemRequest>
requests){
                        clearRows();
                        if(requests.isEmpty()){
                                memRequestsTable.setText(2, 0, "No
request/s to display");

                memRequestsTable.getFlexCellFormatter().setColSpan(2, 0, 5);

                memRequestsTable.getFlexCellFormatter().setHorizontalAlign
ment(2, 0, HasHorizontalAlignment.ALIGN_CENTER);
                        }else{
                                for(int row =
0;row<requests.size();row++){

                memRequestsTable.setWidget(row+2, 0, new
RadioButton(""));

                memRequestsTable.getFlexCellFormatter().setStyleName(row+
2, 0, "recordBorder");
                                                User user =
requests.get(row).projMemRequestPK.username;

                memRequestsTable.setText(row+2, 1, user.firstName+"
"+user.lastName);

                memRequestsTable.getFlexCellFormatter().setStyleName(row+
2, 1, "recordBorder");

                memRequestsTable.setText(row+2, 2, user.username);

                memRequestsTable.getFlexCellFormatter().setStyleName(row+
2, 2, "recordBorder");

                memRequestsTable.setText(row+2, 3,
requests.get(row).message);

                memRequestsTable.getFlexCellFormatter().setStyleName(row+
2, 3, "recordBorder");
                                                String[] parse =
requests.get(row).projMemRequestPK.projID.projName.split("=");

                memRequestsTable.setText(row+2, 4, parse[0]);

                memRequestsTable.getFlexCellFormatter().setStyleName(row+
2, 4, "recordBorder");
                                        }
                                }
                }

                private void clearRows(){
                        int tableCount =
memRequestsTable.getRowCount();
                        for(int row = tableCount-1; row>=2; row--){

                memRequestsTable.removeRow(row);
                        }
                }

                public void clearRowStyles(){
                        for(int row = 2; row <
memRequestsTable.getRowCount(); row++){

                memRequestsTable.getRowFormatter().removeStyleName(row,
"tableHighlight");
                        }
                }

                public Integer getSelectedRow() {
                        for (int i = 2; i <
memRequestsTable.getRowCount(); ++i) {
                                RadioButton radioButton =
(RadioButton)memRequestsTable.getWidget(i, 0);
                                if(radioButton.getValue()){
                                        return i;
                                }
                        }
                        return null;
                }

                public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
                        firstPageButton.setEnabled(!isFirstPage);
                        prevButton.setEnabled(!isFirstPage);
                        pageNumber.setText("Page "+currPage+" of
"+totalPage);
                        nextButton.setEnabled(!isLastPage);
                        lastPageButton.setEnabled(!isLastPage);
                }

                public String getSearchText(){
                        return searchBox.getText();
                }
```

```
	public void clearSearchText(){
		this.searchBox.setText("");
	}

	public Integer getMemReqsPerPageText(){
		return intBoxMemReqPerPage.getValue();
	}

	public void clearMemReqsPerPageText(){
		this.intBoxMemReqPerPage.setText("");
	}

	public HasClickHandlers getSearchButton(){
		return searchButton;
	}

	public void memReqs_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Choose a request
to view"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public void memReqsPerPage_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Minimum no. of
requests per page is 1"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public HasClickHandlers getcancelSearchButton(){
		return cancelSearchButton;
	}

	public HasClickHandlers getGoToFirstPageButton(){
		return firstPageButton;
	}

	public HasClickHandlers getGoToLastPageButton(){
		return lastPageButton;
	}

	public HasClickHandlers getPrevButton(){
		return prevButton;
	}

	public HasClickHandlers getNextButton(){
		return nextButton;
	}

	public HasClickHandlers getAcceptButton(){
		return approveButton;
	}

	public HasClickHandlers getRejectButton(){
		return rejectButton;
	}

	public HasClickHandlers getRefreshButton(){
		return refreshButton;
	}

	public HasClickHandlers getMemReqsPerPageButton(){
		return memReqPerPageButton;
	}

	public HasKeyDownHandlers getSearchKeyHandler(){
		return searchBox;
	}

	public FlexTable getMemReqsTable(){
		return memRequestsTable;
	}

	public Widget asWidget() {
		return this;
	}

}
```

MyPendingRequestsView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.presenter.MyPendingRequestsPresenter;
import cepir.shared.Project;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;

public class MyPendingRequestsView extends Composite implements
MyPendingRequestsPresenter.Display{
	private Label lblHeader;
	private FlexTable pendingListTable;
	private Button cancelButton;
	private Button clearButton;

	public MyPendingRequestsView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

	mainFlexTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75))+"px");

		lblHeader = new Label("");
		lblHeader.setStyleName("header");
		mainFlexTable.setWidget(0, 0, lblHeader);

		pendingListTable = new FlexTable();
		mainFlexTable.setWidget(1, 0, pendingListTable);

		Label lblProjectName = new Label("Project
Name");
		lblProjectName.setStyleName("tableHeader");
		pendingListTable.setWidget(0, 0,
lblProjectName);

	pendingListTable.getFlexCellFormatter().setWidth(0, 0,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

	pendingListTable.getCellFormatter().setHorizontalAlignment(0
, 0, HasHorizontalAlignment.ALIGN_CENTER);

		Label lblDescription = new Label("Description");
		lblDescription.setStyleName("tableHeader");
		pendingListTable.setWidget(0, 1, lblDescription);

	pendingListTable.getFlexCellFormatter().setWidth(0, 1,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.6))+"px");

	pendingListTable.getCellFormatter().setHorizontalAlignment(0
, 1, HasHorizontalAlignment.ALIGN_CENTER);

		Label lblCancelRequest = new Label("Cancel
Request?");
		lblCancelRequest.setStyleName("tableHeader");
		pendingListTable.setWidget(0, 2,
lblCancelRequest);

	pendingListTable.getFlexCellFormatter().setWidth(0, 2,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

	pendingListTable.getCellFormatter().setHorizontalAlignment(0
, 2, HasHorizontalAlignment.ALIGN_CENTER);
```

```
                        FlexTable buttonTable = new FlexTable();
                        mainFlexTable.setWidget(2, 0, buttonTable);

                        cancelButton = new Button("New button");
                        cancelButton.setText("Cancel request/s");
                        buttonTable.setWidget(0, 0, cancelButton);

                        clearButton = new Button("New button");
                        clearButton.setText("Clear");
                        buttonTable.setWidget(0, 1, clearButton);
                }

                public void loadPendingList(ArrayList<Project> projects,
Boolean isHyperlink){
                        clearRows();
                        if(projects.isEmpty()){
                                pendingListTable.setText(1, 0, "No
pending request/s to display");

                pendingListTable.getFlexCellFormatter().setColSpan(1, 0, 3);

                pendingListTable.getFlexCellFormatter().setHorizontalAlignm
ent(1, 0, HasHorizontalAlignment.ALIGN_CENTER);
                        }else{
                                for(int row =
0;row<projects.size();row++){
                                        if(isHyperlink){

                pendingListTable.setWidget(row+1, 0, new
Hyperlink(projects.get(row).projName,
"projHome="+projects.get(row).projID));
                                        }else{

                pendingListTable.setText(row+1, 0,
projects.get(row).projName);
                                        }

                pendingListTable.getFlexCellFormatter().setStyleName(row+1,
0, "recordBorder");

                pendingListTable.setText(row+1, 1,
projects.get(row).projDesc);

                pendingListTable.getFlexCellFormatter().setStyleName(row+1,
1, "recordBorder");

                pendingListTable.setWidget(row+1, 2, new CheckBox());

                pendingListTable.getFlexCellFormatter().setStyleName(row+1,
2, "recordBorder");

                pendingListTable.getFlexCellFormatter().setHorizontalAlignm
ent(row+1, 2, HasHorizontalAlignment.ALIGN_CENTER);
                                }
                        }
                }

                private void clearRows(){
                        int tableCount =pendingListTable.getRowCount();
                        for(int row = tableCount-1; row>=1; row--){
                                pendingListTable.removeRow(row);
                        }
                }

                public void clearBoxes(){
                        for (int i = 1; i < pendingListTable.getRowCount();
++i) {
                                CheckBox checkBox =
(CheckBox)pendingListTable.getWidget(i, 2);
                                checkBox.setValue(false);
                        }
                }

                public ArrayList<Integer> getSelectedRows() {
                        ArrayList<Integer> selectedRows = new
ArrayList<Integer>();
                        for (int i = 1; i < pendingListTable.getRowCount();
++i) {
                                CheckBox checkBox =
(CheckBox)pendingListTable.getWidget(i, 2);
                                if (checkBox.getValue()) {
                                        selectedRows.add(i);
                                }
                        }
                        return selectedRows;
                }

                public void setHeader(String header){
                        lblHeader.setText(header);
                }

                public HasClickHandlers getCancelButton(){
                        return cancelButton;
                }

                public HasClickHandlers getClearButton(){
                        return clearButton;
                }

}
```

PendingMembersView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.PendingMembersPresenter;
import cepir.shared.ProjMemRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Widget;

public class PendingMembersView extends Composite implements
PendingMembersPresenter.Display{
        private FlexTable pendingUsersTable;
        private PushButton acceptButton;
        private PushButton rejectButton;
        private PushButton refreshButton;
        private TextBox search;
        private PushButton searchButton;
        private PushButton cancelSearchButton;
        private FlexTable paginator;
        private FlexTable usersPerPage;
        private PushButton nextButton;
        private PushButton prevButton;
        private PushButton firstPageButton;
        private PushButton lastPageButton;
        private Label pageNumber;
        private IntegerBox intBoxUsersPerPage;
        private Button usersPerPageButton;

        public PendingMembersView() {

                FlexTable mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);

        mainFlexTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75))+"px");

        mainFlexTable.setHeight(Integer.toString(Window.getClientHe
ight()-150)+"px"); //150 - banner size
```

```
                pendingUsersTable = new FlexTable();
                mainFlexTable.setWidget(0, 0,
pendingUsersTable);

        mainFlexTable.getFlexCellFormatter().setVerticalAlignment(0,
0, HasVerticalAlignment.ALIGN_TOP);

                FlexTable buttonTable = new FlexTable();
                pendingUsersTable.setWidget(0, 1, buttonTable);

                ImageResources images =
GWT.create(ImageResources.class);
                acceptButton = new PushButton(new
Image(images.accept().getUrl()));
                acceptButton.setStyleName("icon");
                acceptButton.setTitle("Approve request");
                buttonTable.setWidget(0, 0, acceptButton);

                rejectButton = new PushButton(new
Image(images.delete().getUrl()));
                rejectButton.setStyleName("icon");
                rejectButton.setTitle("Reject request");
                buttonTable.setWidget(0, 1, rejectButton);

                refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
                refreshButton.setStyleName("icon");
                buttonTable.setWidget(0, 2, refreshButton);

                FlexTable searchTable = new FlexTable();
                pendingUsersTable.setWidget(0, 2, searchTable);

        pendingUsersTable.getFlexCellFormatter().setHorizontalAlign
ment(0, 2, HasHorizontalAlignment.ALIGN_RIGHT);

                search = new TextBox();
                searchTable.setWidget(0, 0, search);

                searchButton = new PushButton(new
Image(images.search().getUrl()));
                searchButton.setStyleName("icon");
                searchTable.setWidget(0, 1, searchButton);

                cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
                cancelSearchButton.setStyleName("icon");
                searchTable.setWidget(0, 2, cancelSearchButton);

                Label lblUsername = new Label("Username");
                pendingUsersTable.setWidget(1, 1, lblUsername);

        pendingUsersTable.getFlexCellFormatter().setWidth(1, 1,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.25))+"px");

                Label lblFirstName = new Label("Message");
                pendingUsersTable.setWidget(1, 2, lblFirstName);

        pendingUsersTable.getFlexCellFormatter().setWidth(1, 2,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.75))+"px");

        pendingUsersTable.getRowFormatter().setStyleName(1,
"tableHeader");

                //paginator
                paginator = new FlexTable();
                mainFlexTable.setWidget(1, 0, paginator);

        mainFlexTable.getCellFormatter().setHorizontalAlignment(1,
0, HasHorizontalAlignment.ALIGN_CENTER);

                firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
                firstPageButton.setStyleName("icon");
                firstPageButton.setTitle("Go to first page");
                paginator.setWidget(0,0,firstPageButton);

                prevButton = new PushButton(new
Image(images.previous().getUrl()));
                prevButton.setStyleName("icon");
                prevButton.setTitle("Previous");
                paginator.setWidget(0,1,prevButton);

                pageNumber = new Label("");
                paginator.setWidget(0,2,pageNumber);

                nextButton = new PushButton(new
Image(images.next().getUrl()));
                nextButton.setStyleName("icon");
                nextButton.setTitle("Next");
                paginator.setWidget(0,3,nextButton);

                lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                lastPageButton.setStyleName("icon");
                lastPageButton.setTitle("Go to last page");
                paginator.setWidget(0, 4, lastPageButton);

                //users per page
                usersPerPage = new FlexTable();
                mainFlexTable.setWidget(2, 0, usersPerPage);

        mainFlexTable.getCellFormatter().setHorizontalAlignment(2,
0, HasHorizontalAlignment.ALIGN_CENTER);

                Label lblUsersPerPage = new Label("No. of users
per page");
                usersPerPage.setWidget(0, 0, lblUsersPerPage);

                intBoxUsersPerPage = new IntegerBox();
                intBoxUsersPerPage.setWidth("30px");
                intBoxUsersPerPage.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        usersPerPageButton.click();
                                }
                        }
                });
                usersPerPage.setWidget(0, 1,
intBoxUsersPerPage);

                usersPerPageButton = new Button("Go");
                usersPerPage.setWidget(0, 2,
usersPerPageButton);
        }
        public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
                firstPageButton.setEnabled(!isFirstPage);
                prevButton.setEnabled(!isFirstPage);
                pageNumber.setText("Page "+currPage+" of
"+totalPage);
                nextButton.setEnabled(!isLastPage);
                lastPageButton.setEnabled(!isLastPage);
        }

        public void loadUsersList(ArrayList<ProjMemRequest>
users){
                clearRows();
                if(users.isEmpty()){
                        pendingUsersTable.setText(2, 1, "No
membership requests to display");

        pendingUsersTable.getFlexCellFormatter().setColSpan(2, 1, 3);

        pendingUsersTable.getFlexCellFormatter().setHorizontalAlign
ment(2, 1, HasHorizontalAlignment.ALIGN_CENTER);
                }else{
                        for(int row =
0;row<users.size();row++){

        pendingUsersTable.setWidget(row+2, 0, new CheckBox());

        pendingUsersTable.setText(row+2, 1,
users.get(row).projMemRequestPK.username.username);

        pendingUsersTable.setText(row+2, 2, users.get(row).message);
```

```
                        }
                }
        }
        private void clearRows(){
                int tableCount =
pendingUsersTable.getRowCount();
                for(int row = tableCount-1; row>=2; row--){
                        pendingUsersTable.removeRow(row);
                }
        }

        public ArrayList<Integer> getSelectedRows() {
                ArrayList<Integer> selectedRows = new
ArrayList<Integer>();

                for (int i = 2; i <
pendingUsersTable.getRowCount(); ++i) {
                        CheckBox checkBox =
(CheckBox)pendingUsersTable.getWidget(i, 0);
                        if (checkBox.getValue()) {
                                selectedRows.add(i);
                        }
                }

                return selectedRows;
        }

        public void usersPerPage_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Minimum no. of
requests per page is 1"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public String getSearchText(){
                return search.getText();
        }

        public void clearSearchText(){
                this.search.setText("");
        }

        public Integer getUsersPerPageText(){
                return intBoxUsersPerPage.getValue();
        }

        public void clearUsersPerPageText(){
                this.intBoxUsersPerPage.setText("");
        }

        public HasClickHandlers getAcceptButton(){
                return acceptButton;
        }

        public HasClickHandlers getRejectButton(){
                return rejectButton;
        }

        public HasClickHandlers getRefreshButton(){
                return refreshButton;
        }

        public HasClickHandlers getSearchButton(){
                return searchButton;
        }

        public HasClickHandlers getcancelSearchButton(){
                return cancelSearchButton;
        }

        public HasClickHandlers getGoToFirstPageButton(){
                return firstPageButton;
        }

        public HasClickHandlers getGoToLastPageButton(){
                return lastPageButton;
        }

        public HasClickHandlers getPrevButton(){
                return prevButton;
        }

        public HasClickHandlers getNextButton(){
                return nextButton;
        }

        public HasClickHandlers getUsersPerPageButton(){
                return usersPerPageButton;
        }

        public HasKeyDownHandlers getSearchKeyHandler(){
                return search;
        }

        public Widget asWidget() {
                return this;
        }
}
```

PendingProjListTabView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.PendingProjListPresenter;
import cepir.shared.NewProjRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Widget;

public class PendingProjListTabView extends Composite implements
PendingProjListPresenter.Display{
        private FlexTable pendingProjTable;
        private TextBox textBoxSearch;
        private Label pageNumber;
        private PushButton acceptButton;
        private PushButton rejectButton;
        private PushButton refreshButton;
        private PushButton searchButton;
        private PushButton cancelSearchButton;
        private PushButton nextButton;
        private PushButton prevButton;
        private PushButton firstPageButton;
        private PushButton lastPageButton;
        private IntegerBox intBoxUsersPerPage;
        private Button usersPerPageButton;

        public PendingProjListTabView() {

                FlexTable mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);

        mainFlexTable.setSize(Integer.toString((int)(Window.getClient
Width()*0.8*0.75))+"px",

        Integer.toString((int)(Window.getClientHeight()-150))+"px");
//150->banner size

                pendingProjTable = new FlexTable();
```

```
                        mainFlexTable.setWidget(0, 0, pendingProjTable);

                        FlexTable buttonPanel = new FlexTable();
                        pendingProjTable.setWidget(0, 1, buttonPanel);

                        ImageResources images =
GWT.create(ImageResources.class);
                        acceptButton = new PushButton(new
Image(images.accept().getUrl()));
                        acceptButton.setStyleName("icon");
                        acceptButton.setTitle("Accept");
                        buttonPanel.setWidget(0, 0, acceptButton);

                        rejectButton = new PushButton(new
Image(images.delete().getUrl()));
                        rejectButton.setStyleName("icon");
                        rejectButton.setTitle("Reject");
                        buttonPanel.setWidget(0, 1, rejectButton);

                        refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
                        refreshButton.setStyleName("icon");
                        refreshButton.setTitle("Refresh");
                        buttonPanel.setWidget(0, 2, refreshButton);

                        FlexTable searchTable = new FlexTable();
                        pendingProjTable.setWidget(0, 2, searchTable);

                pendingProjTable.getFlexCellFormatter().setColSpan(0, 2, 2);

                pendingProjTable.getFlexCellFormatter().setHorizontalAlignm
ent(0, 2, HasHorizontalAlignment.ALIGN_RIGHT);

                        searchButton = new PushButton(new
Image(images.search().getUrl()));
                        searchButton.setStyleName("icon");
                        searchTable.setTitle("Search");

                        textBoxSearch = new TextBox();
                        searchTable.setWidget(0, 0, textBoxSearch);
                        searchTable.setWidget(0, 1, searchButton);

                        cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
                        cancelSearchButton.setStyleName("icon");
                        cancelSearchButton.setTitle("Cancel Search");
                        searchTable.setWidget(0, 2, cancelSearchButton);

                pendingProjTable.getCellFormatter().setHorizontalAlignment(1
, 1, HasHorizontalAlignment.ALIGN_CENTER);

                pendingProjTable.getCellFormatter().setVerticalAlignment(1,
1, HasVerticalAlignment.ALIGN_MIDDLE);

                pendingProjTable.getCellFormatter().setHorizontalAlignment(1
, 2, HasHorizontalAlignment.ALIGN_CENTER);

                pendingProjTable.getCellFormatter().setVerticalAlignment(1,
2, HasVerticalAlignment.ALIGN_MIDDLE);

                pendingProjTable.getCellFormatter().setHorizontalAlignment(1
, 3, HasHorizontalAlignment.ALIGN_CENTER);

                pendingProjTable.getCellFormatter().setVerticalAlignment(1,
3, HasVerticalAlignment.ALIGN_MIDDLE);

                        Label lblProjectName = new Label("Project
Name");
                        pendingProjTable.setWidget(1, 1,
lblProjectName);

                pendingProjTable.getFlexCellFormatter().setWidth(1, 1,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

                        Label lblProjectDescription = new Label("Project
Description");
                        pendingProjTable.setWidget(1, 2,
lblProjectDescription);

                pendingProjTable.getFlexCellFormatter().setWidth(1, 2,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.6))+"px");

                        Label lblRequestdBy = new Label("Requested
By");
                        pendingProjTable.setWidget(1, 3, lblRequestdBy);

                pendingProjTable.getFlexCellFormatter().setWidth(1, 3,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

                pendingProjTable.getRowFormatter().setStyleName(1,"tableHe
ader");

                        FlexTable paginator = new FlexTable();
                        mainFlexTable.setWidget(1, 0, paginator);

                        firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
                        firstPageButton.setStyleName("icon");
                        firstPageButton.setTitle("Go to first page");
                        paginator.setWidget(0,0,firstPageButton);

                        prevButton = new PushButton(new
Image(images.previous().getUrl()));
                        prevButton.setStyleName("icon");
                        prevButton.setTitle("Previous");
                        paginator.setWidget(0,1,prevButton);

                        pageNumber = new Label("");
                        paginator.setWidget(0,2,pageNumber);

                        nextButton = new PushButton(new
Image(images.next().getUrl()));
                        nextButton.setStyleName("icon");
                        nextButton.setTitle("Next");
                        paginator.setWidget(0,3,nextButton);

                        lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                        lastPageButton.setStyleName("icon");
                        lastPageButton.setTitle("Go to last page");
                        paginator.setWidget(0, 4, lastPageButton);


                mainFlexTable.getCellFormatter().setHorizontalAlignment(0,
0, HasHorizontalAlignment.ALIGN_LEFT);

                mainFlexTable.getCellFormatter().setVerticalAlignment(0, 0,
HasVerticalAlignment.ALIGN_TOP);

                mainFlexTable.getCellFormatter().setHorizontalAlignment(1,
0, HasHorizontalAlignment.ALIGN_CENTER);

                        FlexTable usersPerPage = new FlexTable();
                        mainFlexTable.setWidget(2, 0, usersPerPage);

                mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(2, 0, HasHorizontalAlignment.ALIGN_CENTER);

                        Label lblUsersPerPage = new Label("No. of
projects per page");
                        usersPerPage.setWidget(0, 0, lblUsersPerPage);

                        intBoxUsersPerPage = new IntegerBox();
                        intBoxUsersPerPage.setWidth("30px");
                        intBoxUsersPerPage.addKeyDownHandler(new
KeyDownHandler() {

                                public void
onKeyDown(KeyDownEvent event) {

                if(event.getNativeKeyCode()==13){

                usersPerPageButton.click();
                                        }
                                }
                        });
                        usersPerPage.setWidget(0, 1,
intBoxUsersPerPage);
```

```
                usersPerPageButton = new Button("Go");
                usersPerPage.setWidget(0, 2,
usersPerPageButton);
        }

        public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
                firstPageButton.setEnabled(!isFirstPage);
                prevButton.setEnabled(!isFirstPage);
                pageNumber.setText("Page
"+(totalPage==0?0:currPage)+" of "+totalPage);
                nextButton.setEnabled(!isLastPage);
                lastPageButton.setEnabled(!isLastPage);
        }

        public void loadProjectList(ArrayList<NewProjRequest>
projects){
                clearRows();
                if(projects.isEmpty()){
                        pendingProjTable.setText(2, 0, "No
project/s to display");

        pendingProjTable.getFlexCellFormatter().setColSpan(2, 0, 6);

        pendingProjTable.getFlexCellFormatter().setHorizontalAlignm
ent(2, 0, HasHorizontalAlignment.ALIGN_CENTER);
                }else{
                        for(int row =
0;row<projects.size();row++){

        pendingProjTable.setWidget(row+2, 0, new CheckBox());

        pendingProjTable.getFlexCellFormatter().setStyleName(row+2,
0, "recordBorder");

        pendingProjTable.setText(row+2, 1,
projects.get(row).newProjName);

        pendingProjTable.getFlexCellFormatter().setStyleName(row+2,
1, "recordBorder");

        pendingProjTable.setText(row+2, 2,
projects.get(row).newProjDesc);

        pendingProjTable.getFlexCellFormatter().setStyleName(row+2,
2, "recordBorder");

        pendingProjTable.setText(row+2, 3,
projects.get(row).user.username);

        pendingProjTable.getFlexCellFormatter().setStyleName(row+2,
3, "recordBorder");
                        }
                }
        }

        private void clearRows(){
                int tableCount =pendingProjTable.getRowCount();
                for(int row = tableCount-1; row>=2; row--){
                        pendingProjTable.removeRow(row);
                }
        }

        public ArrayList<Integer> getSelectedRows() {
                ArrayList<Integer> selectedRows = new
ArrayList<Integer>();

                for (int i = 2; i <
pendingProjTable.getRowCount(); ++i) {
                        CheckBox checkBox =
(CheckBox)pendingProjTable.getWidget(i, 0);
                        if (checkBox.getValue()) {
                                selectedRows.add(i);
                        }
                }

                return selectedRows;
        }

        public void usersPerPage_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Minimum no. of
requests per page is 1"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public String getSearchText(){
                return textBoxSearch.getText();
        }

        public void clearSearchText(){
                this.textBoxSearch.setText("");
        }

        public Integer getUsersPerPageText(){
                return intBoxUsersPerPage.getValue();
        }

        public void clearUsersPerPageText(){
                this.intBoxUsersPerPage.setText("");
        }

        public HasClickHandlers getAcceptButton(){
                return acceptButton;
        }

        public HasClickHandlers getRejectButton(){
                return rejectButton;
        }

        public HasClickHandlers getRefreshButton(){
                return refreshButton;
        }

        public HasClickHandlers getSearchButton(){
                return searchButton;
        }

        public HasClickHandlers getCancelSearchButton(){
                return cancelSearchButton;
        }

        public HasClickHandlers getGoToFirstPageButton(){
                return firstPageButton;
        }

        public HasClickHandlers getGoToLastPageButton(){
                return lastPageButton;
        }

        public HasClickHandlers getPrevButton(){
                return prevButton;
        }

        public HasClickHandlers getNextButton(){
                return nextButton;
        }

        public HasClickHandlers getUsersPerPageButton(){
                return usersPerPageButton;
        }

        public HasKeyDownHandlers getSearchKeyHandler(){
                return textBoxSearch;
        }

        public Widget asWidget(){
                return this;
        }
}
```

PendingUsersListTabView.java

```
package cepir.client.view;

import java.util.ArrayList;
```

```
import cepir.client.ImageResources;
import cepir.client.presenter.PendingUsersListTabPresenter;
import cepir.shared.AcctRequest;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Widget;

public class PendingUsersListTabView extends Composite implements
PendingUsersListTabPresenter.Display{
	private FlexTable requestListFlexTable;
	private Label lblFirstName;
	private Label lblLastName;
	private Label lblMiddleName;
	private Label lblAffiliation;
	private Label lblUsername;
	private Label lblEmail;
	private PushButton acceptButton;
	private PushButton rejectButton;
	private PushButton refreshButton;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private PushButton nextButton;
	private PushButton prevButton;
	private PushButton firstPageButton;
	private PushButton lastPageButton;
	private Label pageNumber;
	private TextBox search;
	private IntegerBox intBoxUsersPerPage;
	private Button usersPerPageButton;

	public PendingUsersListTabView() {
		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

	mainFlexTable.getFlexCellFormatter().setWidth(0,0,Integer.toS
tring((int)(Window.getClientWidth()*0.8*0.75))+"px");

	mainFlexTable.setHeight(Integer.toString(Window.getClientHe
ight()-150)+"px"); //150 - banner size

	mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(0, 0, HasHorizontalAlignment.ALIGN_LEFT);

	mainFlexTable.getFlexCellFormatter().setVerticalAlignment(0,
0, HasVerticalAlignment.ALIGN_TOP);

		requestListFlexTable = new FlexTable();
		mainFlexTable.setWidget(0, 0,
requestListFlexTable);

		FlexTable buttonFlexTable = new FlexTable();
		requestListFlexTable.setWidget(0, 1,
buttonFlexTable);

		ImageResources images =
GWT.create(ImageResources.class);
		acceptButton = new PushButton(new
Image(images.accept().getUrl()));
		acceptButton.setTitle("Accept");
		acceptButton.setStyleName("icon");
		buttonFlexTable.setWidget(0, 0, acceptButton);

		rejectButton = new PushButton(new
Image(images.delete().getUrl()));
		rejectButton.setTitle("Reject");
		rejectButton.setStyleName("icon");
		buttonFlexTable.setWidget(0, 1, rejectButton);

		refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
		refreshButton.setStyleName("icon");
		refreshButton.setTitle("Refresh");
		buttonFlexTable.setWidget(0, 2, refreshButton);

		FlexTable searchFlexTable = new FlexTable();
		requestListFlexTable.setWidget(0, 4,
searchFlexTable);

		search = new TextBox();
		searchFlexTable.setWidget(0, 1, search);

		searchButton = new PushButton(new
Image(images.search().getUrl()));
		searchButton.setStyleName("icon");
		searchButton.setTitle("Search");
		searchFlexTable.setWidget(0, 2, searchButton);

		cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
		cancelSearchButton.setStyleName("icon");
		cancelSearchButton.setTitle("Cancel");
		searchFlexTable.setWidget(0, 3,
cancelSearchButton);

	requestListFlexTable.getFlexCellFormatter().setColSpan(0, 4,
2);

		lblLastName = new Label("Last Name");

	lblLastName.setHorizontalAlignment(HasHorizontalAlignment
.ALIGN_CENTER);
		requestListFlexTable.setWidget(1, 1,
lblLastName);

		lblFirstName = new Label("First Name");

	lblFirstName.setHorizontalAlignment(HasHorizontalAlignment
.ALIGN_CENTER);
		requestListFlexTable.setWidget(1, 2,
lblFirstName);

		lblMiddleName = new Label("Middle Name");

	lblMiddleName.setHorizontalAlignment(HasHorizontalAlignm
ent.ALIGN_CENTER);
		requestListFlexTable.setWidget(1, 3,
lblMiddleName);

		lblEmail = new Label("Email");

	lblEmail.setHorizontalAlignment(HasHorizontalAlignment.AL
IGN_CENTER);
		requestListFlexTable.setWidget(1, 4, lblEmail);

		lblAffiliation = new Label("Affiliation");

	lblAffiliation.setHorizontalAlignment(HasHorizontalAlignment
.ALIGN_CENTER);
		requestListFlexTable.setWidget(1, 5,
lblAffiliation);

	requestListFlexTable.getFlexCellFormatter().setColSpan(0, 1,
2);

		lblUsername = new Label("Username");

	lblUsername.setHorizontalAlignment(HasHorizontalAlignment
.ALIGN_CENTER);
		requestListFlexTable.setWidget(1, 6,
lblUsername);
```

```
		requestListFlexTable.getRowFormatter().addStyleName(1,
"tableHeader");

			FlexTable paginator = new FlexTable();
			mainFlexTable.setWidget(1, 0, paginator);

		mainFlexTable.getFlexCellFormatter().setColSpan(1, 0, 8);

		mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(1, 0, HasHorizontalAlignment.ALIGN_CENTER);

			firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
			firstPageButton.setStyleName("icon");
			firstPageButton.setTitle("Go to first page");
			paginator.setWidget(0,0,firstPageButton);

			prevButton = new PushButton(new
Image(images.previous().getUrl()));
			prevButton.setStyleName("icon");
			prevButton.setTitle("Previous");
			paginator.setWidget(0,1,prevButton);

			pageNumber = new Label("");
			paginator.setWidget(0,2,pageNumber);

			nextButton = new PushButton(new
Image(images.next().getUrl()));
			nextButton.setStyleName("icon");
			nextButton.setTitle("Next");
			paginator.setWidget(0,3,nextButton);

			lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
			lastPageButton.setStyleName("icon");
			lastPageButton.setTitle("Go to last page");
			paginator.setWidget(0, 4, lastPageButton);

			FlexTable usersPerPage = new FlexTable();
			mainFlexTable.setWidget(2, 0, usersPerPage);

		mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(2, 0, HasHorizontalAlignment.ALIGN_CENTER);

			Label lblUsersPerPage = new Label("No. of users
per page");
			usersPerPage.setWidget(0, 0, lblUsersPerPage);

			intBoxUsersPerPage = new IntegerBox();
			intBoxUsersPerPage.setWidth("30px");
			intBoxUsersPerPage.addKeyDownHandler(new
KeyDownHandler() {

				public void
onKeyDown(KeyDownEvent event) {

		if(event.getNativeKeyCode()==13){

		usersPerPageButton.click();
					}
				}
			});
			usersPerPage.setWidget(0, 1,
intBoxUsersPerPage);

			usersPerPageButton = new Button("Go");
			usersPerPage.setWidget(0, 2,
usersPerPageButton);
		}

		public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
			firstPageButton.setEnabled(!isFirstPage);
			prevButton.setEnabled(!isFirstPage);
			pageNumber.setText("Page "+currPage+" of
"+totalPage);
			nextButton.setEnabled(!isLastPage);
			lastPageButton.setEnabled(!isLastPage);
		}

		public void loadUsersList(ArrayList<AcctRequest> users){
			clearRows();
			if(users.isEmpty()){
				requestListFlexTable.setText(2, 1,
"No account requests to display");

		requestListFlexTable.getFlexCellFormatter().setColSpan(2, 1,
6);

		requestListFlexTable.getFlexCellFormatter().setHorizontalAlig
nment(2, 1, HasHorizontalAlignment.ALIGN_CENTER);
			}else{
				for(int row =
0;row<users.size();row++){

		requestListFlexTable.setWidget(row+2, 0, new CheckBox());

		requestListFlexTable.setText(row+2, 1,
users.get(row).lastName);

		requestListFlexTable.setText(row+2, 2,
users.get(row).firstName);

		requestListFlexTable.setText(row+2, 3,
users.get(row).middleName);

		requestListFlexTable.setText(row+2, 4, users.get(row).email);

		requestListFlexTable.setText(row+2, 5,
users.get(row).affiliation);

		requestListFlexTable.setText(row+2, 6,
users.get(row).username);
					}
			}
		}
		private void clearRows(){
			int tableCount
=requestListFlexTable.getRowCount();
			for(int row = tableCount-1; row>=2; row--){

		requestListFlexTable.removeRow(row);
			}
		}

		public ArrayList<Integer> getSelectedRows() {
			ArrayList<Integer> selectedRows = new
ArrayList<Integer>();

			for (int i = 2; i <
requestListFlexTable.getRowCount(); ++i) {
				CheckBox checkBox =
(CheckBox)requestListFlexTable.getWidget(i, 0);
				if (checkBox.getValue()) {
					selectedRows.add(i);
				}
			}

			return selectedRows;
		}

		public void sendingMail_Warning(){
			DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
			warning.setTitle("Error");
			warning.setWidget(new Label("Sending mail eror.
Please connect to the internet."));

		warning.setPopupPosition((Window.getClientWidth() -
getOffsetWidth()) >> 1, 0);
			warning.show();
		}

		public void usersPerPage_Warning(int left, int top){
			DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
			warning.setTitle("Warning");
			warning.setWidget(new Label("Minimum no. of
requests per page is 1"));
```

```
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public String getSearchText(){
                return search.getText();
        }

        public void clearSearchText(){
                this.search.setText("");
        }

        public Integer getUsersPerPageText(){
                return intBoxUsersPerPage.getValue();
        }

        public void clearUsersPerPageText(){
                this.intBoxUsersPerPage.setText("");
        }

        public HasClickHandlers getAcceptButton(){
                return acceptButton;
        }

        public HasClickHandlers getRejectButton(){
                return rejectButton;
        }

        public HasClickHandlers getRefreshButton(){
                return refreshButton;
        }

        public HasClickHandlers getSearchButton(){
                return searchButton;
        }

        public HasClickHandlers getcancelSearchButton(){
                return cancelSearchButton;
        }

        public HasClickHandlers getGoToFirstPageButton(){
                return firstPageButton;
        }

        public HasClickHandlers getGoToLastPageButton(){
                return lastPageButton;
        }

        public HasClickHandlers getPrevButton(){
                return prevButton;
        }

        public HasClickHandlers getNextButton(){
                return nextButton;
        }

        public HasClickHandlers getUsersPerPageButton(){
                return usersPerPageButton;
        }

        public HasKeyDownHandlers getSearchKeyHandler(){
                return search;
        }

        public Widget asWidget(){
                return this;
        }

}
```

ProjHomeView.java

```
package cepir.client.view;

import cepir.client.ImageResources;
import cepir.client.presenter.ProjHomePresenter;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.Widget;

public class ProjHomeView extends Composite implements
ProjHomePresenter.Display{
        private Label lblProjName;
        private Label lblProjDescTxt;
        private PushButton joinButton;
        private PushButton leaveButton;
        private PushButton editButton;
        private PushButton deleteButton;
        private DialogBox deleteWarning;
        private Button deleteOkButton;
        private Button deleteNotOkButton;
        private DialogBox leaveWarning;
        private Button leaveOkButton;
        private Button leaveNotOkButton;
        private DialogBox joinDialogBox;
        private TextArea joinMessage;
        private Label joinError;
        private Button joinOkButton;
        private Button joinNotOkButton;
        private DialogBox sendNotice;
        private Button sendOK;

        public ProjHomeView() {

                FlexTable mainTable = new FlexTable();
                initWidget(mainTable);

        mainTable.setWidth(Integer.toString((int)(Window.getClientW
idth()*0.8*0.75))+"px");

                lblProjName = new Label();
                lblProjName.setStyleName("header4");
                mainTable.setWidget(0, 0, lblProjName);

        mainTable.getCellFormatter().setHorizontalAlignment(0, 0,
HasHorizontalAlignment.ALIGN_CENTER);

                Label lblProjectDescription = new Label("Project
Description");
                lblProjectDescription.setStyleName("header5");
                mainTable.setWidget(1, 0, lblProjectDescription);

                lblProjDescTxt = new Label();
                lblProjDescTxt.setStyleName("labelPadding");
                mainTable.setWidget(2, 0, lblProjDescTxt);

                FlexTable flexTable = new FlexTable();
                mainTable.setWidget(3, 0, flexTable);

        mainTable.getCellFormatter().setHorizontalAlignment(3, 0,
HasHorizontalAlignment.ALIGN_RIGHT);

                joinButton = new PushButton("Join project");
                joinButton.setStyleName("icon2");
                joinButton.setTitle("Join project");
                joinButton.setVisible(false);
                flexTable.setWidget(0, 0, joinButton);

                editButton = new PushButton("Edit project info");
                editButton.setStyleName("icon2");
                editButton.setTitle("Edit project information");
                editButton.setVisible(false);
                flexTable.setWidget(0, 1, editButton);

                leaveButton = new PushButton("Leave project");
                leaveButton.setStyleName("icon3");
                leaveButton.setTitle("Leave project");
                leaveButton.setVisible(false);
                flexTable.setWidget(0, 2, leaveButton);
```

```
                deleteButton = new PushButton("Delete project");
                deleteButton.setStyleName("icon3");
                deleteButton.setTitle("Delete project");
                deleteButton.setVisible(false);
                flexTable.setWidget(0, 3, deleteButton);

                ImageResources images =
GWT.create(ImageResources.class);
                //delete project warning
                deleteWarning = new DialogBox();
                deleteWarning.setGlassEnabled(true);
                deleteWarning.setAnimationEnabled(true);
                deleteWarning.setText("Delete project");

                FlexTable warningContents = new FlexTable();
                deleteWarning.setWidget(warningContents);

                Image warningIcon = new
Image(images.warning().getUrl());
                warningContents.setWidget(0, 0, warningIcon);

        warningContents.getFlexCellFormatter().setRowSpan(0, 0, 2);

                Label warningTxt = new Label("Are you sure you
want to delete the project?");
                warningContents.setWidget(0, 1, warningTxt);

        warningContents.getFlexCellFormatter().setColSpan(0, 1, 3);

                deleteOkButton = new Button("Yes");
                warningContents.setWidget(1, 1, deleteOkButton);

        warningContents.getFlexCellFormatter().setHorizontalAlignme
nt(1, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                deleteNotOkButton = new Button("No");
                warningContents.setWidget(1, 2,
deleteNotOkButton);

                //leave project warning
                leaveWarning = new DialogBox();
                leaveWarning.setGlassEnabled(true);
                leaveWarning.setAnimationEnabled(true);
                leaveWarning.setText("Leave project");

                FlexTable leaveContents = new FlexTable();
                leaveWarning.setWidget(leaveContents);

                Image leaveWarningIcon = new
Image(images.warning().getUrl());
                leaveContents.setWidget(0, 0, leaveWarningIcon);

        leaveContents.getFlexCellFormatter().setRowSpan(0, 0, 2);

                Label leaveWarningTxt = new Label("Are you
sure you want to leave the project?");
                leaveContents.setWidget(0, 1, leaveWarningTxt);

        leaveContents.getFlexCellFormatter().setColSpan(0, 1, 3);

                leaveOkButton = new Button("Yes");
                leaveContents.setWidget(1, 1, leaveOkButton);

        leaveContents.getFlexCellFormatter().setHorizontalAlignment(
1, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                leaveNotOkButton = new Button("No");
                leaveContents.setWidget(1, 2, leaveNotOkButton);

                //join project dialog box
                joinDialogBox = new DialogBox();
                joinDialogBox.setAnimationEnabled(true);
                joinDialogBox.setGlassEnabled(true);
                joinDialogBox.setSize("400px", "200px");
                joinDialogBox.setText("Message to project
administrator/s");

                FlexTable joinTable = new FlexTable();
                joinDialogBox.setWidget(joinTable);

                joinTable.setText(0, 0, "Please tell the project
administrator/s more about yourself and why you would like to " +
                                "join the project (200
characters maximum)");

                joinMessage = new TextArea();
                joinMessage.setSize("400px", "150px");
                joinTable.setWidget(1, 0, joinMessage);

                FlexTable buttons = new FlexTable();
                joinTable.setWidget(2, 0, buttons);

        joinTable.getFlexCellFormatter().setHorizontalAlignment(2, 0,
HasHorizontalAlignment.ALIGN_CENTER);

                joinOkButton = new Button("Send request");
                buttons.setWidget(0, 0, joinOkButton);
                joinNotOkButton = new Button("Cancel");
                buttons.setWidget(0, 1, joinNotOkButton);

                joinError = new Label();
                joinError.setStyleName("errorMsg");
                joinTable.setWidget(3, 0, joinError);

        joinTable.getFlexCellFormatter().setHorizontalAlignment(3, 0,
HasHorizontalAlignment.ALIGN_CENTER);

                //request sent Notice
                sendNotice = new DialogBox();
                sendNotice.setAnimationEnabled(true);
                sendNotice.setGlassEnabled(true);
                sendNotice.setWidth("250px");
                sendNotice.setText("Notice");

                FlexTable notice = new FlexTable();
                sendNotice.setWidget(notice);
                notice.setText(0, 0, "Your request for project
membership has been sent. An email will be sent to you once it has been
processed.");

                sendOK = new Button("OK");
                notice.setWidget(1, 0, sendOK);
        }

        public Widget asWidget(){
                return this;
        }

        public void setProjName(String projName){
                lblProjName.setText(projName);
        }

        public void setProjDesc(String projDesc){
                lblProjDescTxt.setText(projDesc);
        }

        public void showDeleteWarning(){
                deleteWarning.center();
                deleteWarning.show();
        }

        public void hideDeleteWarning(){
                deleteWarning.hide();
        }

        public void showLeaveWarning(){
                leaveWarning.center();
                leaveWarning.show();
        }

        public void hideLeaveWarning(){
                leaveWarning.hide();
        }

        public void showJoinMessage(){
                joinDialogBox.center();
                joinDialogBox.show();
        }

        public void hideJoinMessage(){
```

```
		joinDialogBox.hide();
	}

	public void showSendNotice(){
		sendNotice.center();
		sendNotice.show();
	}

	public void hideSendNotice(){
		sendNotice.hide();
	}

	public String getMessage(){
		return joinMessage.getText();
	}

	public void clearMsg(){
		joinMessage.setText("");
	}

	public void setErrorTxt(String msg){
		joinError.setText(msg);
	}

	public HasClickHandlers getDeleteOKButton(){
		return deleteOkButton;
	}

	public HasClickHandlers getDeleteNotOKButton(){
		return deleteNotOkButton;
	}

	public HasClickHandlers getLeaveOKButton(){
		return leaveOkButton;
	}

	public HasClickHandlers getLeaveNotOKButton(){
		return leaveNotOkButton;
	}

	public HasClickHandlers getJoinOkButton(){
		return joinOkButton;
	}

	public HasClickHandlers getJoinNotOkButton(){
		return joinNotOkButton;
	}

	public HasClickHandlers getSendOkButton(){
		return sendOK;
	}

	public HasClickHandlers getJoinButton(){
		return joinButton;
	}

	public void setVisibilityJoin(Boolean isVisible){
		joinButton.setVisible(isVisible);
	}

	public HasClickHandlers getLeaveButton(){
		return leaveButton;
	}

	public void setVisibilityLeave(Boolean isVisible){
		leaveButton.setVisible(isVisible);
	}

	public HasClickHandlers getEditButton(){
		return editButton;
	}

	public void setVisibilityEdit(Boolean isVisible){
		editButton.setVisible(isVisible);
	}

	public HasClickHandlers getDeleteButton(){
		return deleteButton;
	}

	public void setVisibilityDelete(Boolean isVisible){
		deleteButton.setVisible(isVisible);
	}

	public void joinProject_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Request to join
project has been sent already"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

}
```

ProjListTabView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.ProjListTabPresenter;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;

public class ProjListTabView extends Composite implements
ProjListTabPresenter.Display{
	private FlexTable mainFlexTable;
	private FlexTable projListTable;
	private PushButton addButton;
	private PushButton delButton;
	private PushButton refreshButton;
	private Label projName;
	private Label dateCreated;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private PushButton nextButton;
	private PushButton prevButton;
	private PushButton firstPageButton;
	private PushButton lastPageButton;
	private Label pageNumber;
	private TextBox search;
	private IntegerBox intBoxUsersPerPage;
	private Button usersPerPageButton;
	private DialogBox deleteWarning;
	private Button deleteOkButton;
	private Button deleteNotOkButton;
	private PushButton editButton;

	public ProjListTabView() {

		mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		projListTable = new FlexTable();
		mainFlexTable.setWidget(0, 0, projListTable);
```

```
		mainFlexTable.getFlexCellFormatter().setVerticalAlignment(0,
0, HasVerticalAlignment.ALIGN_TOP);

		mainFlexTable.setSize(Integer.toString((int)(Window.getClient
Width()*0.8*0.75))+"px",

		Integer.toString((int)(Window.getClientHeight()-150))+"px");
//150->banner size

				FlexTable buttonPanel = new FlexTable();
				projListTable.setWidget(0, 1, buttonPanel);

				ImageResources images =
GWT.create(ImageResources.class);
				addButton = new PushButton(new
Image(images.add().getUrl()));
				addButton.setStyleName("icon");
				addButton.setTitle("Add");
				buttonPanel.setWidget(0, 0, addButton);

				editButton = new PushButton(new
Image(images.edit().getUrl()));
				editButton.setStyleName("icon");
				editButton.setTitle("Edit");
				buttonPanel.setWidget(0, 1, editButton);

				delButton = new PushButton(new
Image(images.delete().getUrl()));
				delButton.setStyleName("icon");
				delButton.setTitle("Delete");
				buttonPanel.setWidget(0, 2, delButton);

				refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
				refreshButton.setStyleName("icon");
				refreshButton.setTitle("Refresh");
				buttonPanel.setWidget(0, 4, refreshButton);

				FlexTable searchFlexTable = new FlexTable();
				projListTable.setWidget(0, 2, searchFlexTable);

		projListTable.getFlexCellFormatter().setHorizontalAlignment(
0, 2, HasHorizontalAlignment.ALIGN_RIGHT);

				search = new TextBox();
				searchFlexTable.setWidget(0, 1, search);

				searchButton = new PushButton(new
Image(images.search().getUrl()));
				searchButton.setStyleName("icon");
				searchButton.setTitle("Search");
				searchFlexTable.setWidget(0, 2, searchButton);

				cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
				cancelSearchButton.setStyleName("icon");
				cancelSearchButton.setTitle("Cancel");
				searchFlexTable.setWidget(0, 3,
cancelSearchButton);

				projName = new Label("Project Name");
				projListTable.setWidget(1, 1, projName);
				projListTable.getFlexCellFormatter().setWidth(1,
1, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.6))+"px");

				projListTable.getRowFormatter().setStyleName(1,
"tableHeader");

				dateCreated = new Label("Date Created");
				projListTable.setWidget(1, 2, dateCreated);
				projListTable.getFlexCellFormatter().setWidth(1,
2, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.3))+"px");

				//paginator
				FlexTable paginator = new FlexTable();
				mainFlexTable.setWidget(1, 0, paginator);

		mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(1, 0, HasHorizontalAlignment.ALIGN_CENTER);

				firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
				firstPageButton.setStyleName("icon");
				firstPageButton.setTitle("Go to first page");
				paginator.setWidget(0,0,firstPageButton);

				prevButton = new PushButton(new
Image(images.previous().getUrl()));
				prevButton.setStyleName("icon");
				prevButton.setTitle("Previous");
				paginator.setWidget(0,1,prevButton);

				pageNumber = new Label("");
				paginator.setWidget(0,2,pageNumber);

				nextButton = new PushButton(new
Image(images.next().getUrl()));
				nextButton.setStyleName("icon");
				nextButton.setTitle("Next");
				paginator.setWidget(0,3,nextButton);

				lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
				lastPageButton.setStyleName("icon");
				lastPageButton.setTitle("Go to last page");
				paginator.setWidget(0, 4, lastPageButton);

				//projects per page
				FlexTable usersPerPage = new FlexTable();
				mainFlexTable.setWidget(2, 0, usersPerPage);

		mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(2, 0, HasHorizontalAlignment.ALIGN_CENTER);

				Label lblUsersPerPage = new Label("No. of
projects per page");
				usersPerPage.setWidget(0, 0, lblUsersPerPage);

				intBoxUsersPerPage = new IntegerBox();
				intBoxUsersPerPage.setWidth("30px");
				intBoxUsersPerPage.addKeyDownHandler(new
KeyDownHandler() {

					public void
onKeyDown(KeyDownEvent event) {

		if(event.getNativeKeyCode()==13){

		usersPerPageButton.click();
							}
						}
				});
				usersPerPage.setWidget(0, 1,
intBoxUsersPerPage);

				usersPerPageButton = new Button("Go");
				usersPerPage.setWidget(0, 2,
usersPerPageButton);

				//delete project warning
				deleteWarning = new DialogBox();
				deleteWarning.setGlassEnabled(true);
				deleteWarning.setAnimationEnabled(true);
				deleteWarning.setText("Delete project");

				FlexTable warningContents = new FlexTable();
				deleteWarning.setWidget(warningContents);

				Image warningIcon = new
Image(images.warning().getUrl());
				warningContents.setWidget(0, 0, warningIcon);

		warningContents.getFlexCellFormatter().setRowSpan(0, 0, 2);

				Label warningTxt = new Label("Are you sure you
want to delete the project?");
				warningContents.setWidget(0, 1, warningTxt);

		warningContents.getFlexCellFormatter().setColSpan(0, 1, 3);
```

```
                deleteOkButton = new Button("Yes");
                warningContents.setWidget(1, 1, deleteOkButton);

        warningContents.getFlexCellFormatter().setHorizontalAlignme
nt(1, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                deleteNotOkButton = new Button("No");
                warningContents.setWidget(1, 2,
deleteNotOkButton);

        }

        public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
                firstPageButton.setEnabled(!isFirstPage);
                prevButton.setEnabled(!isFirstPage);
                pageNumber.setText("Page
"+(totalPage==0?0:currPage)+" of "+totalPage);
                nextButton.setEnabled(!isLastPage);
                lastPageButton.setEnabled(!isLastPage);
        }

        public void loadProjectList(ArrayList<Project> projects){
                clearRows();
                if(projects.isEmpty()){
                        projListTable.setText(2, 0, "No
project/s to display");

        projListTable.getFlexCellFormatter().setColSpan(2, 0, 3);

        projListTable.getFlexCellFormatter().setHorizontalAlignment(
2, 0, HasHorizontalAlignment.ALIGN_CENTER);
                }else{
                        for(int row =
0;row<projects.size();row++){

        projListTable.setWidget(row+2, 0, new RadioButton(""));

        projListTable.setText(row+2, 1, projects.get(row).projName);
                                        DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy EEEE");

        projListTable.setText(row+2, 2,
formatted.format(projects.get(row).dateCreated));
                        }
                }
        }

        public Integer getSelectedRow() {
                for (int i = 2; i <projListTable.getRowCount();
++i) {
                        RadioButton radioButton =
(RadioButton)projListTable.getWidget(i, 0);

                        if(radioButton.getValue()){
                                return i;
                        }
                }
                return null;
        }

        public void clearRowStyles(){
                for(int row = 2; row <
projListTable.getRowCount(); row++){

        projListTable.getRowFormatter().removeStyleName(row,
"tableHighlight");
                }
        }

        private void clearRows(){
                int tableCount =projListTable.getRowCount();
                for(int row = tableCount-1; row>=2; row--){
                        projListTable.removeRow(row);
                }
        }

        public void delete_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Error");
                warning.setWidget(new Label("Choose a project
to delete"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public void editProj_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Choose a
project"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public void usersPerPage_Warning(int left, int top){
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Minimum no. of
projects per page is 1"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public void showDeleteProjWarning(){
                deleteWarning.center();
                deleteWarning.show();
        }

        public void hideDeleteProjWarning(){
                deleteWarning.hide();
        }

        public String getSearchText(){
                return search.getText();
        }

        public void clearSearchText(){
                this.search.setText("");
        }

        public Integer getUsersPerPageText(){
                return intBoxUsersPerPage.getValue();
        }

        public void clearUsersPerPageText(){
                this.intBoxUsersPerPage.setText("");
        }

        public Widget asWidget(){
                return this;
        }

        public HasClickHandlers getAddButton(){
                return addButton;
        }

        public HasClickHandlers getEditButton(){
                return editButton;
        }

        public HasClickHandlers getDeleteButton(){
                return delButton;
        }

        public HasClickHandlers getRefreshButton(){
                return refreshButton;
        }

        public HasClickHandlers getSearchButton(){
                return searchButton;
        }

        public HasClickHandlers getcancelSearchButton(){
                return cancelSearchButton;
```

```
	}

	public HasClickHandlers getGoToFirstPageButton(){
		return firstPageButton;
	}

	public HasClickHandlers getGoToLastPageButton(){
		return lastPageButton;
	}

	public HasClickHandlers getPrevButton(){
		return prevButton;
	}

	public HasClickHandlers getNextButton(){
		return nextButton;
	}

	public HasClickHandlers getUsersPerPageButton(){
		return usersPerPageButton;
	}

	public HasClickHandlers getDeleteOkButton(){
		return deleteOkButton;
	}

	public HasClickHandlers getDeleteNotOkButton(){
		return deleteNotOkButton;
	}

	public HasKeyDownHandlers getSearchKeyHandler(){
		return search;
	}

	public FlexTable getProjListTable(){
		return projListTable;
	}

}
```

ProjMngtPageView.java

```
package cepir.client.view;

import cepir.client.presenter.ProjMngtPagePresenter;

import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedTabPanel;
import com.google.gwt.user.client.ui.Widget;

public class ProjMngtPageView extends Composite implements
ProjMngtPagePresenter.Display{
	public DecoratedTabPanel decoratedTabPanel;

	public ProjMngtPageView() {
		decoratedTabPanel = new DecoratedTabPanel();
		initWidget(decoratedTabPanel);
	}

	public void addTab(Widget widget, String tabName){
		decoratedTabPanel.add(widget, tabName);
	}

	public void replaceTab(int index, Widget widget, String
tabName){
		decoratedTabPanel.remove(index);
		decoratedTabPanel.insert(widget, tabName, index
- 1);
	}

	public void setSelectedTab(int index){
		decoratedTabPanel.selectTab(index);
	}

	public HasSelectionHandlers<Integer> getTabPanel(){
		return decoratedTabPanel;
	}

	public Widget asWidget() {
		return this;
	}
}
```

RequestForAcctView.java

```
package cepir.client.view;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Widget;

import cepir.client.ImageResources;
import cepir.client.presenter.RequestForAcctPresenter;
import com.google.gwt.user.client.ui.PasswordTextBox;
import com.google.gwt.user.client.ui.PushButton;

public class RequestForAcctView extends Composite implements
RequestForAcctPresenter.Display{
	private TextBox txtBoxAffiliation;
	private TextBox txtBoxLastName;
	private TextBox txtBoxMiddleName;
	private TextBox txtBoxEmail;
	private Label lblError;
	private Button submitButton;
	private Button clearButton;
	private TextBox txtBoxUsername;
	private PasswordTextBox txtBoxPassword;
	private Label lblEmailMsg;
	private Label label;
	private Label lblUsername;
	private TextBox txtBoxFirstName;
	private Label lblUsernameMsg;
	private Label lblRetypePassword;
	private PasswordTextBox txtBoxRetypePassword;
	private Label lblRetypePasswordMsg;
	private Button okDialogBox;
	private DialogBox dialogBox;
	private TextBox securityTxt;
	private TextBox answerTxt;
	private PushButton infoButton;

	public RequestForAcctView(){
		FlexTable formTable = new FlexTable();
		initWidget(formTable);

		Label lblRequestForAcct = new Label("Request
for an account");
		lblRequestForAcct.setStyleName("header2");
		formTable.setWidget(0, 0, lblRequestForAcct);
		formTable.getFlexCellFormatter().setColSpan(0,
0, 3);

		Label lblRequiredField = new Label("All fields are
required");
		formTable.setWidget(1, 1, lblRequiredField);

		lblUsername = new Label("Username");
		formTable.setWidget(2, 0, lblUsername);

		txtBoxUsername = new TextBox();
		formTable.setWidget(2, 1, txtBoxUsername);

		lblUsernameMsg = new Label("");
		formTable.setWidget(2, 2, lblUsernameMsg);

		Label lblPassword = new Label("Password");
		formTable.setWidget(3, 0, lblPassword);

		txtBoxPassword = new PasswordTextBox();
		formTable.setWidget(3, 1, txtBoxPassword);
```

```
                lblRetypePassword = new Label("Re-type
password");
                formTable.setWidget(4, 0, lblRetypePassword);

                txtBoxRetypePassword = new
PasswordTextBox();
                formTable.setWidget(4, 1,
txtBoxRetypePassword);

                lblRetypePasswordMsg = new Label("");
                formTable.setWidget(4, 2,
lblRetypePasswordMsg);

                label = new Label("First Name");
                formTable.setWidget(5, 0, label);

                txtBoxFirstName = new TextBox();
                formTable.setWidget(5, 1, txtBoxFirstName);

                Label lblMiddleName = new Label("Middle
Name");
                formTable.setWidget(6, 0, lblMiddleName);

                txtBoxMiddleName = new TextBox();
                formTable.setWidget(6, 1, txtBoxMiddleName);

                Label lblLastName = new Label("Last Name");
                formTable.setWidget(7, 0, lblLastName);

                txtBoxLastName = new TextBox();
                formTable.setWidget(7, 1, txtBoxLastName);

                Label lblEmail = new Label("E-mail");
                formTable.setWidget(8, 0, lblEmail);

                txtBoxEmail = new TextBox();
                txtBoxEmail.setVisibleLength(30);
                formTable.setWidget(8, 1, txtBoxEmail);

                lblEmailMsg = new Label("");
                formTable.setWidget(8, 2, lblEmailMsg);

                Label lblAffiliation = new Label("Affiliation");
                formTable.setWidget(9, 0, lblAffiliation);

                txtBoxAffiliation = new TextBox();
                txtBoxAffiliation.setVisibleLength(30);
                formTable.setWidget(9, 1, txtBoxAffiliation);
                formTable.getFlexCellFormatter().setColSpan(0,
0, 2);
                formTable.getFlexCellFormatter().setColSpan(1,
1, 1);
                formTable.getFlexCellFormatter().setColSpan(9,
1, 1);
                formTable.getFlexCellFormatter().setColSpan(7,
1, 1);
                formTable.getFlexCellFormatter().setColSpan(6,
1, 1);

                Label lblSecurityQuestion = new Label("Security
Question");
                lblSecurityQuestion.setWidth("150px");
                formTable.setWidget(10, 0, lblSecurityQuestion);

                securityTxt = new TextBox();
                securityTxt.setWidth("250px");
                formTable.setWidget(10, 1, securityTxt);

                FlexTable info = new FlexTable();
                formTable.setWidget(10, 2, info);

                ImageResources images =
GWT.create(ImageResources.class);
                infoButton = new PushButton(new
Image(images.info().getUrl()));
                infoButton.setTitle("Why do I need to provide a
security question?");
                infoButton.setStyleName("icon");
                info.setWidget(0, 0, infoButton);

                Label lblAnswer = new Label("Answer");
                formTable.setWidget(11, 0, lblAnswer);

                answerTxt = new TextBox();
                formTable.setWidget(11, 1, answerTxt);

                AbsolutePanel absolutePanel = new
AbsolutePanel();
                formTable.setWidget(12, 1, absolutePanel);

                submitButton = new Button("Submit");
                absolutePanel.add(submitButton);

                clearButton = new Button("Clear");
                absolutePanel.add(clearButton, 64, 0);

                lblError = new Label("");
                formTable.setWidget(13, 1, lblError);

                dialogBox = new DialogBox();
            dialogBox.setGlassEnabled(true);
            dialogBox.setAnimationEnabled(true);
            dialogBox.setText("Message");
            FlexTable contents = new FlexTable();
            contents.setWidget(0, 0, new Label("Request for an account
sent."));

            okDialogBox = new Button("OK");
            contents.setWidget(1, 0, okDialogBox);
            dialogBox.setWidget(contents);
        }

        public HasClickHandlers getSubmitButton(){
                return submitButton;
        }

        public HasClickHandlers getClearButton(){
                return clearButton;
        }

        public void setErrorTxt(String errMsg){
                this.lblError.setText(errMsg);
                this.lblError.setStyleName("errorMsg");
        }

        public void setUsernameMsg(String usernameMsg){
                this.lblUsernameMsg.setText(usernameMsg);
                this.lblUsernameMsg.setStyleName("errorMsg");
        }

        public void setEmailMsg(String emailMsg){
                this.lblEmailMsg.setText(emailMsg);
                this.lblEmailMsg.setStyleName("errorMsg");
        }

        public void setRetypePasswdMsg(String passwdMsg){
                this.lblRetypePasswordMsg.setText(passwdMsg);

        this.lblRetypePasswordMsg.setStyleName("errorMsg");
        }

        public Widget asWidget(){
                return this;
        }

        public String getFirstName(){
                return txtBoxFirstName.getText();
        }

        public String getMiddleName(){
                return txtBoxMiddleName.getText();
        }

        public String getLastName(){
                return txtBoxLastName.getText();
        }

        public String getEmail(){
                return txtBoxEmail.getText();
```

```
	}

	public String getAffiliation(){
		return txtBoxAffiliation.getText();
	}

	public String getUsername(){
		return txtBoxUsername.getText();
	}

	public String getPassword(){
		return txtBoxPassword.getText();
	}

	public String getRetypedPassword(){
		return txtBoxRetypePassword.getText();
	}

	public String getQuestion(){
		return securityTxt.getText();
	}

	public String getAnswer(){
		return answerTxt.getText();
	}

	public void clearForm(){
		txtBoxAffiliation.setText("");
		txtBoxFirstName.setText("");
		txtBoxLastName.setText("");
		txtBoxMiddleName.setText("");
		txtBoxEmail.setText("");
		txtBoxUsername.setText("");
		txtBoxPassword.setText("");
		txtBoxRetypePassword.setText("");
		lblError.setText("");
		lblEmailMsg.setText("");
		lblUsernameMsg.setText("");
	}

	public void requestSent(){
	   dialogBox.center();
	   dialogBox.show();
	}

	public void showInfo(){
		DialogBox dialog = new DialogBox();
		dialog.setAnimationEnabled(true);
		dialog.setAutoHideEnabled(true);
		dialog.setText("Why do I need a security
question?");
		dialog.setWidth("350px");
		dialog.setWidget(new HTML("<p
align='justify'>In case you forget your password, answering the security
question will enable you to reset it. " +
					"Just enter your email on
the 'Forgot Password?' link found on the homepage.\n\nExamples:
</p><ul><li>What is the name of your favorite book?</li>" +
					"<li>What is your child's
nickname?</li><li>What is the street name on which you grew
up?</li></ul>"));
		dialog.center();
		dialog.show();
	}
	public HasClickHandlers getOkDialog(){
		return okDialogBox;
	}

	public HasClickHandlers getInfoButton(){
		return infoButton;
	}
}
```

RequestForProjView.java

```
package cepir.client.view;

import cepir.client.presenter.RequestForProjPresenter;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Widget;

public class RequestForProjView extends Composite implements
RequestForProjPresenter.Display{
	private TextBox textBoxProjName;
	private TextArea textBoxProjDesc;
	private Label lblProjNameError;
	private Label lblErrorMsg;
	private Button sendButton;
	private Button cancelButton;
	private Button clearButton;
	private Button okDialogBox;
	private DialogBox dialogBox;

	public RequestForProjView() {
		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		Label lblRequestForProject = new Label("Request
for Project Creation");
		lblRequestForProject.setStyleName("header2");
		mainFlexTable.setWidget(0, 0,
lblRequestForProject);

		Label lblAllFieldsAre = new Label("All fields are
required");
		mainFlexTable.setWidget(1, 1, lblAllFieldsAre);

		Label lblProjectName = new Label("Project
Name");
		mainFlexTable.setWidget(2, 0, lblProjectName);

		textBoxProjName = new TextBox();
		textBoxProjName.setWidth("300px");
		mainFlexTable.setWidget(2, 1, textBoxProjName);

		lblProjNameError = new Label("");
		mainFlexTable.setWidget(2, 2, lblProjNameError);

		Label lblProjectDescription = new Label("Project
Description (500 characters max)");
		lblProjectDescription.setWidth("130px");
		mainFlexTable.setWidget(3, 0,
lblProjectDescription);

	mainFlexTable.getFlexCellFormatter().setVerticalAlignment(3,
0, HasVerticalAlignment.ALIGN_TOP);

		textBoxProjDesc = new TextArea();
		textBoxProjDesc.setWidth("300px");
		textBoxProjDesc.setVisibleLines(6);
		mainFlexTable.setWidget(3, 1, textBoxProjDesc);

	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 3);

		FlexTable flexTable = new FlexTable();
		mainFlexTable.setWidget(4, 1, flexTable);

		sendButton = new Button("New button");
		sendButton.setText("Send");
		flexTable.setWidget(0, 0, sendButton);

		cancelButton = new Button("New button");
		cancelButton.setText("Cancel");
		flexTable.setWidget(0, 1, cancelButton);

		clearButton = new Button("New button");
		clearButton.setText("Clear");
		flexTable.setWidget(0, 2, clearButton);
```

```
		mainFlexTable.getFlexCellFormatter().setColSpan(4, 1, 2);

			lblErrorMsg = new Label("");
			mainFlexTable.setWidget(5, 1, lblErrorMsg);

		mainFlexTable.getFlexCellFormatter().setColSpan(5, 1, 2);

			dialogBox = new DialogBox();
		   dialogBox.setGlassEnabled(true);
		   dialogBox.setAnimationEnabled(true);
		   dialogBox.setText("Message");
		   FlexTable contents = new FlexTable();
		   contents.setWidget(0, 0, new Label("Request for project
creation sent."));

		   okDialogBox = new Button("OK");
		   contents.setWidget(1, 0, okDialogBox);
		   dialogBox.setWidget(contents);
		}

		public String getProjName(){
			return this.textBoxProjName.getText();
		}

		public String getProjDesc(){
			return this.textBoxProjDesc.getText();
		}

		public void setProjNameErrTxt(String text){
			this.lblProjNameError.setText(text);
			this.lblProjNameError.setStyleName("errorMsg");
		}

		public void setErrTxt(String text){
			this.lblErrorMsg.setText(text);
			this.lblErrorMsg.setStyleName("errorMsg");
		}

		public void clearFields(){
			this.textBoxProjName.setText("");
			this.textBoxProjDesc.setText("");
			this.lblErrorMsg.setText("");
			this.lblProjNameError.setText("");
		}

		public void requestSent(){
		   dialogBox.center();
		   dialogBox.show();
		}
		public HasClickHandlers getOkDialog(){
			return okDialogBox;
		}
		public HasClickHandlers getSendButton(){
			return sendButton;
		}

		public HasClickHandlers getCancelButton(){
			return cancelButton;
		}

		public HasClickHandlers getClearButton(){
			return clearButton;
		}

		public Widget asWidget(){
			return this;
		}

}
```

UnauthorizedNoticeView.java

```
package cepir.client.view;

import cepir.client.ImageResources;
import cepir.client.presenter.UnauthorizedNoticePresenter;

import com.google.gwt.core.client.GWT;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.Widget;

public class UnauthorizedNoticeView extends Composite implements
UnauthorizedNoticePresenter.Display{
	private Label errorMsg;

	public UnauthorizedNoticeView() {

		FlexTable table = new FlexTable();
		initWidget(table);

		ImageResources images =
GWT.create(ImageResources.class);
		table.setWidget(0, 0, new
Image(images.warning().getUrl()));

		errorMsg = new Label();
		errorMsg.setStyleName("errorMsg");
		table.setWidget(0,1,errorMsg);

		Hyperlink back = new Hyperlink("Back", "##");
		table.setWidget(0, 2, back);
	}

	public void setErrorMsg(String text){
		errorMsg.setText(text);
	}

	public Widget asWidget(){
		return this;
	}

}
```

UserMngtPageView.java

```
package cepir.client.view;

import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedTabPanel;
import com.google.gwt.user.client.ui.Widget;

import cepir.client.presenter.UserMngtPagePresenter;

public class UserMngtPageView extends Composite implements
UserMngtPagePresenter.Display{
	public DecoratedTabPanel decoratedTabPanel;

	public UserMngtPageView(){
		decoratedTabPanel = new DecoratedTabPanel();
		initWidget(decoratedTabPanel);
	}

	public void addTab(Widget widget, String tabName){
		decoratedTabPanel.add(widget, tabName, false);
	}

	public void replaceTab(int index, Widget widget, String
tabName){
		decoratedTabPanel.remove(index);
		decoratedTabPanel.insert(widget, tabName, index
- 1);
	}

	public void setSelectedTab(int index){
		decoratedTabPanel.selectTab(index);
	}

	public HasSelectionHandlers<Integer> getTabPanel(){
		return decoratedTabPanel;
	}

	public Widget asWidget() {
```

```
                return this;
        }
}
```

UsersListTabView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.UsersListTabPresenter;
import cepir.shared.User;
import cepir.shared.YesNo;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.Widget;

public class UsersListTabView extends Composite implements
UsersListTabPresenter.Display{
        private FlexTable userListFlexTable;
        private PushButton addButton;
        private PushButton editButton;
        private PushButton delButton;
        private PushButton refreshButton;
        private PushButton searchButton;
        private PushButton cancelSearchButton;
        private PushButton nextButton;
        private PushButton prevButton;
        private PushButton firstPageButton;
        private PushButton lastPageButton;
        private Label lblUsername;
        private Label lblFirstName;
        private Label lblMiddleName;
        private Label lblLastName;
        private Label lblEmail;
        private Label lblSystemAdministrator;
        private Label pageNumber;
        private TextBox search;
        private IntegerBox intBoxUsersPerPage;
        private Button usersPerPageButton;
        private DialogBox deleteWarning;
        private Button deleteOkButton;
        private DialogBox confirmDelete;
        private Button confirmsDelete;
        private Button rejectsDelete;

        public UsersListTabView() {
                FlexTable mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);
                userListFlexTable = new FlexTable();
                mainFlexTable.setWidget(0, 0,
userListFlexTable);

        mainFlexTable.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75))+"px");

        mainFlexTable.setHeight(Integer.toString(Window.getClientHe
ight()-150)+"px"); //150 - banner size

        mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(0, 0, HasHorizontalAlignment.ALIGN_LEFT);

        mainFlexTable.getFlexCellFormatter().setVerticalAlignment(0,
0, HasVerticalAlignment.ALIGN_TOP);

                FlexTable flexTable_1 = new FlexTable();
                userListFlexTable.setWidget(0, 1, flexTable_1);

                ImageResources images =
GWT.create(ImageResources.class);
                addButton = new PushButton(new
Image(images.add().getUrl()));
                addButton.setStyleName("icon");
                addButton.setTitle("Add");
                flexTable_1.setWidget(0, 0, addButton);

                editButton = new PushButton(new
Image(images.edit().getUrl()));
                editButton.setStyleName("icon");
                editButton.setTitle("Edit");
                flexTable_1.setWidget(0, 1, editButton);

                delButton = new PushButton(new
Image(images.delete().getUrl()));
                delButton.setStyleName("icon");
                delButton.setTitle("Delete");
                flexTable_1.setWidget(0, 2, delButton);

                refreshButton = new PushButton(new
Image(images.refresh().getUrl()));
                refreshButton.setStyleName("icon");
                refreshButton.setTitle("Refresh");
                flexTable_1.setWidget(0, 3, refreshButton);

                //search
                FlexTable searchFlexTable = new FlexTable();
                userListFlexTable.setWidget(0, 6,
searchFlexTable);

        userListFlexTable.getFlexCellFormatter().setColSpan(0, 6, 2);

        userListFlexTable.getFlexCellFormatter().setHorizontalAlignm
ent(0, 6, HasHorizontalAlignment.ALIGN_RIGHT);

                search = new TextBox();
                searchFlexTable.setWidget(0, 1, search);

                searchButton = new PushButton(new
Image(images.search().getUrl()));
                searchButton.setStyleName("icon");
                searchButton.setTitle("Search");
                searchFlexTable.setWidget(0, 2, searchButton);

                cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
                cancelSearchButton.setStyleName("icon");
                cancelSearchButton.setTitle("Cancel");
                searchFlexTable.setWidget(0, 3,
cancelSearchButton);

                lblUsername = new Label("Username");
                lblUsername.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 1, lblUsername);


                lblFirstName = new Label("First Name");
                lblFirstName.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 2, lblFirstName);

                lblMiddleName = new Label("Middle Name");
                lblMiddleName.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 3,
lblMiddleName);

                lblLastName = new Label("Last Name");
                lblLastName.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 4, lblLastName);

                lblEmail = new Label("Email");
                lblEmail.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 5, lblEmail);
```

```
                lblSystemAdministrator = new Label("System
administrator");

        lblSystemAdministrator.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 6,
lblSystemAdministrator);

                Label lblToolAdministrator = new Label("Tool
administrator");

        lblToolAdministrator.setStyleName("tableHeader");
                userListFlexTable.setWidget(1, 7,
lblToolAdministrator);


        userListFlexTable.getRowFormatter().addStyleName(1,
"tableHeader");

                //paginator
                FlexTable paginator = new FlexTable();
                mainFlexTable.setWidget(1, 0, paginator);

        mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(1, 0, HasHorizontalAlignment.ALIGN_CENTER);

                firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
                firstPageButton.setStyleName("icon");
                firstPageButton.setTitle("Go to first page");
                paginator.setWidget(0,0,firstPageButton);

                prevButton = new PushButton(new
Image(images.previous().getUrl()));
                prevButton.setStyleName("icon");
                prevButton.setTitle("Previous");
                paginator.setWidget(0,1,prevButton);

                pageNumber = new Label("");
                paginator.setWidget(0,2,pageNumber);

                nextButton = new PushButton(new
Image(images.next().getUrl()));
                nextButton.setStyleName("icon");
                nextButton.setTitle("Next");
                paginator.setWidget(0,3,nextButton);

                lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                lastPageButton.setStyleName("icon");
                lastPageButton.setTitle("Go to last page");
                paginator.setWidget(0, 4, lastPageButton);

                //users per page
                FlexTable usersPerPage = new FlexTable();
                mainFlexTable.setWidget(2, 0, usersPerPage);

        mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(2, 0, HasHorizontalAlignment.ALIGN_CENTER);

                Label lblUsersPerPage = new Label("No. of users
per page");
                usersPerPage.setWidget(0, 0, lblUsersPerPage);

                intBoxUsersPerPage = new IntegerBox();
                intBoxUsersPerPage.setWidth("30px");
                intBoxUsersPerPage.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        usersPerPageButton.click();
                                }
                        }
                });
                usersPerPage.setWidget(0, 1,
intBoxUsersPerPage);
                usersPerPageButton = new Button("Go");
                usersPerPage.setWidget(0, 2,
usersPerPageButton);

                //cannot delete own account dialogBox
                deleteWarning = new DialogBox();
                deleteWarning.setGlassEnabled(true);
                deleteWarning.setAnimationEnabled(true);
                deleteWarning.setText("Delete user error");

                FlexTable warningContents = new FlexTable();
                deleteWarning.setWidget(warningContents);

                Image warningIcon = new
Image(images.warning().getUrl());
                warningContents.setWidget(0, 0, warningIcon);

        warningContents.getFlexCellFormatter().setRowSpan(0, 0, 2);

                Label warningTxt = new Label("You cannot delete
your own account");
                warningContents.setWidget(0, 1, warningTxt);

        warningContents.getFlexCellFormatter().setHorizontalAlignme
nt(0, 1, HasHorizontalAlignment.ALIGN_CENTER);

        warningContents.getFlexCellFormatter().setColSpan(0, 1, 3);

                deleteOkButton = new Button("Ok");
                warningContents.setWidget(1, 1, deleteOkButton);

        warningContents.getFlexCellFormatter().setHorizontalAlignme
nt(1, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                //confirm deletion of sole project admin of a
project
                confirmDelete = new DialogBox();
                confirmDelete.setGlassEnabled(true);
                confirmDelete.setAnimationEnabled(true);
                confirmDelete.setText("Confirm delete");

                FlexTable confirmTable = new FlexTable();
                confirmDelete.setWidget(confirmTable);

                confirmTable.setWidget(0, 0, warningIcon);

        confirmTable.getFlexCellFormatter().setRowSpan(0, 0, 3);

                Label confirmWarning = new Label("Are you sure
you want to delete user account?");
                confirmTable.setWidget(0, 1, confirmWarning);

        confirmTable.getFlexCellFormatter().setHorizontalAlignment(
0, 1, HasHorizontalAlignment.ALIGN_CENTER);

        confirmTable.getFlexCellFormatter().setColSpan(0, 1, 3);

                Label note = new Label("NOTE: Selected user is
the only project administrator of a project");
                confirmTable.setWidget(1, 1, note);

        confirmTable.getFlexCellFormatter().setHorizontalAlignment(
1, 1, HasHorizontalAlignment.ALIGN_CENTER);

        confirmTable.getFlexCellFormatter().setColSpan(1, 1, 3);

                confirmsDelete = new Button("Yes");
                confirmTable.setWidget(2, 1, confirmsDelete);

        confirmTable.getFlexCellFormatter().setHorizontalAlignment(
2, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                rejectsDelete = new Button("No");
                confirmTable.setWidget(2, 2, rejectsDelete);

        }

        public Widget asWidget() {
                return this;
```

```
	}

	public void showDeleteUserWarning(){
		deleteWarning.center();
		deleteWarning.show();
	}

	public void hideDeleteUserWarning(){
		deleteWarning.hide();
	}

	public void showConfirmDeleteWarning(){
		confirmDelete.center();
		confirmDelete.setWidth("400px");
		confirmDelete.show();
	}

	public void hideConfirmDeleteWarning(){
		confirmDelete.hide();
	}

	public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
		firstPageButton.setEnabled(!isFirstPage);
		prevButton.setEnabled(!isFirstPage);
		pageNumber.setText("Page "+currPage+" of
"+totalPage);
		nextButton.setEnabled(!isLastPage);
		lastPageButton.setEnabled(!isLastPage);
	}

	public void loadUsersList(ArrayList<User> users){
		clearRows();
		if(users.isEmpty()){
			userListFlexTable.setText(3, 0, "No
user/s to display");

	userListFlexTable.getFlexCellFormatter().setColSpan(3, 0, 7);

	userListFlexTable.getFlexCellFormatter().setHorizontalAlignm
ent(3, 0, HasHorizontalAlignment.ALIGN_CENTER);
		}else{
			for(int row =
0;row<users.size();row++){

	userListFlexTable.setWidget(row+3, 0, new RadioButton(""));

	userListFlexTable.setText(row+3, 1, users.get(row).username);

	userListFlexTable.setText(row+3, 2, users.get(row).firstName);

	userListFlexTable.setText(row+3, 3,
users.get(row).middleName);

	userListFlexTable.setText(row+3, 4, users.get(row).lastName);

	userListFlexTable.setText(row+3, 5, users.get(row).email);

	userListFlexTable.getFlexCellFormatter().setHorizontalAlignm
ent(row+3, 6, HasHorizontalAlignment.ALIGN_CENTER);

	if(users.get(row).sysAd.equals(YesNo.Yes)){

	userListFlexTable.setText(row+3, 6, "Yes");
					}else{

	userListFlexTable.setText(row+3, 6, "No");
					}

	userListFlexTable.getFlexCellFormatter().setHorizontalAlignm
ent(row+3, 7, HasHorizontalAlignment.ALIGN_CENTER);

	if(users.get(row).toolAd.equals(YesNo.Yes)){

	userListFlexTable.setText(row+3, 7, "Yes");
					}else{

	userListFlexTable.setText(row+3, 7, "No");
					}
			}
		}
	}

	private void clearRows(){
		int tableCount =userListFlexTable.getRowCount();
		for(int row = tableCount-1; row>=3; row--){
			userListFlexTable.removeRow(row);
		}
	}

	public Integer getSelectedRow() {
		for (int i = 3; i <
userListFlexTable.getRowCount(); ++i) {
			RadioButton radioButton =
(RadioButton)userListFlexTable.getWidget(i, 0);
			if(radioButton.getValue()){
				return i;
			}
		}
		return null;
	}

	public void editUser_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Choose a user"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public void delUser_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Choose a user to
delete"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public void usersPerPage_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Minimum no. of
users per page is 1"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public void clearRowStyles(){
		for(int row = 3; row <
userListFlexTable.getRowCount(); row++){

	userListFlexTable.getRowFormatter().removeStyleName(row,
"tableHighlight");
		}
	}

	public String getSearchText(){
		return search.getText();
	}

	public void clearSearchText(){
		this.search.setText("");
	}

	public Integer getUsersPerPageText(){
		return intBoxUsersPerPage.getValue();
	}

	public void clearUsersPerPageText(){
		this.intBoxUsersPerPage.setText("");
	}

	public HasClickHandlers getAddButton(){
		return addButton;
	}
```

```
public HasClickHandlers getEditButton(){
		return editButton;
}

public HasClickHandlers getDeleteButton(){
		return delButton;
}

public HasClickHandlers getRefreshButton(){
		return refreshButton;
}

public HasClickHandlers getSearchButton(){
		return searchButton;
}

public HasClickHandlers getcancelSearchButton(){
		return cancelSearchButton;
}

public HasClickHandlers getGoToFirstPageButton(){
		return firstPageButton;
}

public HasClickHandlers getGoToLastPageButton(){
		return lastPageButton;
}

public HasClickHandlers getPrevButton(){
		return prevButton;
}

public HasClickHandlers getNextButton(){
		return nextButton;
}

public HasClickHandlers getUsersPerPageButton(){
		return usersPerPageButton;
}

public HasClickHandlers getDeleteOkButton(){
		return deleteOkButton;
}

public HasClickHandlers getConfirmDeleteButton(){
		return confirmsDelete;
}

public HasClickHandlers getRejectDeleteButton(){
		return rejectsDelete;
}

public HasClickHandlers getUsername(){
		return lblUsername;
}

public HasClickHandlers getFirstName(){
		return lblFirstName;
}

public HasClickHandlers getMiddleName(){
		return lblMiddleName;
}

public HasClickHandlers getLastName(){
		return lblLastName;
}

public HasClickHandlers getEmail(){
		return lblEmail;
}

public HasClickHandlers getSysAd(){
		return lblSystemAdministrator;
}

public HasKeyDownHandlers getSearchKeyHandler(){
		return search;
}

		public FlexTable getUsersListTable(){
				return userListFlexTable;
		}

}
```

ViewAllProjsView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.ViewAllProjsPresenter;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Widget;

public class ViewAllProjsView extends Composite implements
ViewAllProjsPresenter.Display{
		private TextBox search;
		private PushButton searchButton;
		private PushButton cancelSearchButton;
		private PushButton nextButton;
		private PushButton prevButton;
		private PushButton firstPageButton;
		private PushButton lastPageButton;
		private Label pageNumber;
		private FlexTable projListTable;
		private IntegerBox intBoxProjsPerPage;
		private Button projsPerPageButton;

		public ViewAllProjsView() {

				FlexTable mainTable = new FlexTable();
				initWidget(mainTable);

		mainTable.setWidth(Integer.toString((int)(Window.getClientW
idth()*0.8*0.75))+"px");

				Label lblProjectsAtCepir = new Label("Projects at
CEpiR");
				lblProjectsAtCepir.setStyleName("header");
				mainTable.setWidget(0, 0, lblProjectsAtCepir);

				projListTable = new FlexTable();
				mainTable.setWidget(1, 0, projListTable);

				FlexTable buttonTable = new FlexTable();
				projListTable.setWidget(0, 1, buttonTable);

		projListTable.getFlexCellFormatter().setColSpan(0, 1, 2);

		projListTable.getCellFormatter().setHorizontalAlignment(0, 1,
HasHorizontalAlignment.ALIGN_RIGHT);

				search = new TextBox();
				buttonTable.setWidget(0, 0, search);

				ImageResources images =
GWT.create(ImageResources.class);

				searchButton = new PushButton(new
Image(images.search().getUrl()));
```

```
searchButton.setStyleName("icon");
buttonTable.setWidget(0, 1, searchButton);

cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
cancelSearchButton.setStyleName("icon");
buttonTable.setWidget(0, 2, cancelSearchButton);

Label lblName = new Label("Name");
lblName.setStyleName("tableHeader");
projListTable.setWidget(1, 0, lblName);
projListTable.getFlexCellFormatter().setWidth(1,
0, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.25))+"px");

Label lblDescription = new Label("Description");
lblDescription.setStyleName("tableHeader");
projListTable.setWidget(1, 1, lblDescription);
projListTable.getFlexCellFormatter().setWidth(1,
1, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.5))+"px");

Label lblMember = new Label("Membership
Type");
lblMember.setStyleName("tableHeader");
projListTable.setWidget(1, 2, lblMember);
projListTable.getFlexCellFormatter().setWidth(1,
2, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.25))+"px");

//paginator
FlexTable paginator = new FlexTable();
mainTable.setWidget(2, 0, paginator);

mainTable.getCellFormatter().setHorizontalAlignment(2, 0,
HasHorizontalAlignment.ALIGN_CENTER);

firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
firstPageButton.setStyleName("icon");
firstPageButton.setTitle("Go to first page");
paginator.setWidget(0,0,firstPageButton);

prevButton = new PushButton(new
Image(images.previous().getUrl()));
prevButton.setStyleName("icon");
prevButton.setTitle("Previous");
paginator.setWidget(0,1,prevButton);

pageNumber = new Label("");
paginator.setWidget(0,2,pageNumber);

nextButton = new PushButton(new
Image(images.next().getUrl()));
nextButton.setStyleName("icon");
nextButton.setTitle("Next");
paginator.setWidget(0,3,nextButton);

lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
lastPageButton.setStyleName("icon");
lastPageButton.setTitle("Go to last page");
paginator.setWidget(0, 4, lastPageButton);

//projects per page
FlexTable projsPerPage = new FlexTable();
mainTable.setWidget(3, 0, projsPerPage);

mainTable.getCellFormatter().setHorizontalAlignment(3, 0,
HasHorizontalAlignment.ALIGN_CENTER);

Label lblProjsPerPage = new Label("No. of
projects per page");
projsPerPage.setWidget(0, 0, lblProjsPerPage);

intBoxProjsPerPage = new IntegerBox();
intBoxProjsPerPage.setWidth("30px");
intBoxProjsPerPage.addKeyDownHandler(new
KeyDownHandler() {

public void
onKeyDown(KeyDownEvent event) {

if(event.getNativeKeyCode()==13){

projsPerPageButton.click();
}
}
});
projsPerPage.setWidget(0, 1,
intBoxProjsPerPage);

projsPerPageButton = new Button("Go");
projsPerPage.setWidget(0, 2, projsPerPageButton);
}

public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
firstPageButton.setEnabled(!isFirstPage);
prevButton.setEnabled(!isFirstPage);
pageNumber.setText("Page
"+(totalPage==0?0:currPage)+" of "+totalPage);
nextButton.setEnabled(!isLastPage);
lastPageButton.setEnabled(!isLastPage);
}

public void loadProjectList(ArrayList<Project> projects,
ArrayList<String> isMember){
clearRows();
if(projects.isEmpty()){
projListTable.setText(2, 0, "No
project/s to display");

projListTable.getFlexCellFormatter().setColSpan(2, 0, 3);

projListTable.getFlexCellFormatter().setHorizontalAlignment(
2, 0, HasHorizontalAlignment.ALIGN_CENTER);
}else{
for(int row =
0;row<projects.size();row++){

projListTable.setWidget(row+2, 0, new
Hyperlink(projects.get(row).projName,
"projHome="+projects.get(row).projID));

projListTable.getFlexCellFormatter().setStyleName(row+2,0,
"recordBorder");

projListTable.setText(row+2, 1, projects.get(row).projDesc);

projListTable.getFlexCellFormatter().setStyleName(row+2,1,
"recordBorder");

projListTable.getFlexCellFormatter().setHorizontalAlignment(r
ow+2, 1, HasHorizontalAlignment.ALIGN_JUSTIFY);

projListTable.setText(row+2, 2, isMember.get(row));

projListTable.getFlexCellFormatter().setStyleName(row+2,2,
"recordBorder");

projListTable.getFlexCellFormatter().setHorizontalAlignment(r
ow+2, 2, HasHorizontalAlignment.ALIGN_CENTER);
}
}
}

private void clearRows(){
int tableCount =projListTable.getRowCount();
for(int row = tableCount-1; row>=2; row--){
projListTable.removeRow(row);
}
}

public String getSearchText(){
return search.getText();
}

public void clearSearchText(){
this.search.setText("");
}
```

```
	public Integer getProjsPerPageText(){
		return intBoxProjsPerPage.getValue();
	}

	public void clearProjsPerPageText(){
		this.intBoxProjsPerPage.setText("");
	}

	public void projsPerPage_Warning(int left, int top){
		DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
		warning.setTitle("Warning");
		warning.setWidget(new Label("Minimum no. of
projects per page is 1"));
		warning.setPopupPosition(left, top);
		warning.show();
	}

	public HasClickHandlers getSearchButton(){
		return searchButton;
	}

	public HasClickHandlers getCancelSearchButton(){
		return cancelSearchButton;
	}

	public HasClickHandlers getGoToFirstPageButton(){
		return firstPageButton;
	}

	public HasClickHandlers getGoToLastPageButton(){
		return lastPageButton;
	}

	public HasClickHandlers getPrevButton(){
		return prevButton;
	}

	public HasClickHandlers getNextButton(){
		return nextButton;
	}

	public HasClickHandlers getProjsPerPageButton(){
		return projsPerPageButton;
	}

	public HasKeyDownHandlers getSearchKeyHandler(){
		return search;
	}

	public Widget asWidget(){
		return this;
	}

}
```

ViewDataEntryFormsView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.shared.Base64Coder;
import cepir.shared.DataEntryForm;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.*;

public class ViewDataEntryFormsView extends Composite implements
cepir.client.presenter.ViewDataEntryFormsPresenter.Display{
	private FlexTable formsListTable;
	private FlexTable buttons;
	private PushButton addButton;
	private PushButton editButton;
	private PushButton delButton;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private TextBox search;
	private PushButton nextButton;
	private PushButton prevButton;
	private PushButton firstPageButton;
	private PushButton lastPageButton;
	private Label pageNumber;
	private IntegerBox intBoxFormsPerPage;
	private Button formsPerPageButton;
	private DialogBox confirmFormDelete;
	private Button confirmsFormDelete;
	private Button rejectsFormDelete;

	public ViewDataEntryFormsView() {

		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		Label lblDataEntryForms = new Label("Data
Entry Forms");
		lblDataEntryForms.setStyleName("header4");
		mainFlexTable.setWidget(0, 0,
lblDataEntryForms);

		formsListTable = new FlexTable();
		mainFlexTable.setWidget(1, 0, formsListTable);

		ImageResources images =
GWT.create(ImageResources.class);

		buttons = new FlexTable();
		buttons.setVisible(false);
		formsListTable.setWidget(0, 1, buttons);
		addButton = new PushButton(new
Image(images.add().getUrl()));
		addButton.setStyleName("icon");
		addButton.setTitle("Add Data Entry Form");
		buttons.setWidget(0,0,addButton);

		editButton = new PushButton(new
Image(images.edit().getUrl()));
		editButton.setStyleName("icon");
		editButton.setTitle("Edit Data Entry Form");
		buttons.setWidget(0,1,editButton);

		delButton = new PushButton(new
Image(images.delete().getUrl()));
		delButton.setStyleName("icon");
		delButton.setTitle("Delete Data Entry Form");
		buttons.setWidget(0,2,delButton);

		FlexTable searchTable = new FlexTable();
		formsListTable.setWidget(0, 2, searchTable);

	formsListTable.getCellFormatter().setHorizontalAlignment(0,
2, HasHorizontalAlignment.ALIGN_RIGHT);

	formsListTable.getFlexCellFormatter().setColSpan(0, 2, 2);

		search = new TextBox();
		searchTable.setWidget(0, 1, search);

		searchButton = new PushButton(new
Image(images.search().getUrl()));
		searchButton.setStyleName("icon");
		searchButton.setTitle("Search");
		searchTable.setWidget(0, 2, searchButton);

		cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
		cancelSearchButton.setStyleName("icon");
		cancelSearchButton.setTitle("Cancel");
		searchTable.setWidget(0, 3, cancelSearchButton);

		Label lblFormName = new Label("Form Name");

	lblFormName.setWidth(Integer.toString((int)(Window.getClien
tWidth()*0.8*0.75*0.4))+"px");
```

```
                formsListTable.setWidget(1, 1, lblFormName);

                Label lblCreator = new Label("Creator");

        lblCreator.setWidth(Integer.toString((int)(Window.getClientWi
dth()*0.8*0.75*0.25))+"px");
                formsListTable.setWidget(1, 2, lblCreator);

                Label lblDateCreated = new Label("Date
Created");

        lblDateCreated.setWidth(Integer.toString((int)(Window.getClie
ntWidth()*0.8*0.75*0.3))+"px");
                formsListTable.setWidget(1, 3, lblDateCreated);

        formsListTable.getRowFormatter().setStyleName(1,
"tableHeader");

                FlexTable paginator = new FlexTable();
                mainFlexTable.setWidget(2, 0, paginator);

        mainFlexTable.getCellFormatter().setHorizontalAlignment(2,
0, HasHorizontalAlignment.ALIGN_CENTER);

                firstPageButton = new PushButton(new
Image(images.backToStart().getUrl()));
                firstPageButton.setStyleName("icon");
                firstPageButton.setTitle("Go to first page");
                paginator.setWidget(0,0,firstPageButton);

                prevButton = new PushButton(new
Image(images.previous().getUrl()));
                prevButton.setStyleName("icon");
                prevButton.setTitle("Previous");
                paginator.setWidget(0,1,prevButton);

                pageNumber = new Label("");
                paginator.setWidget(0,2,pageNumber);

                nextButton = new PushButton(new
Image(images.next().getUrl()));
                nextButton.setStyleName("icon");
                nextButton.setTitle("Next");
                paginator.setWidget(0,3,nextButton);

                lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                lastPageButton.setStyleName("icon");
                lastPageButton.setTitle("Go to last page");
                paginator.setWidget(0, 4, lastPageButton);

                FlexTable formsPerPageTable = new FlexTable();
                mainFlexTable.setWidget(3, 0,
formsPerPageTable);

        mainFlexTable.getCellFormatter().setHorizontalAlignment(3,
0, HasHorizontalAlignment.ALIGN_CENTER);

                Label lblUsersPerPage = new Label("No. of forms
per page");
                formsPerPageTable.setWidget(0, 0,
lblUsersPerPage);

                intBoxFormsPerPage = new IntegerBox();
                intBoxFormsPerPage.setWidth("30px");
                intBoxFormsPerPage.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        formsPerPageButton.click();
                                }
                        }
                });
                formsPerPageTable.setWidget(0, 1,
intBoxFormsPerPage);
                formsPerPageButton = new Button("Go");
                formsPerPageTable.setWidget(0, 2,
formsPerPageButton);

                //confirm delete dialog box
                confirmFormDelete = new DialogBox();
                confirmFormDelete.setGlassEnabled(true);
                confirmFormDelete.setAnimationEnabled(true);
                confirmFormDelete.setText("Confirm delete");

                FlexTable confirmFormTable = new FlexTable();

        confirmFormDelete.setWidget(confirmFormTable);

                confirmFormTable.setWidget(0, 0, new
Image(images.warning().getUrl()));

        confirmFormTable.getFlexCellFormatter().setRowSpan(0, 0,
3);

                Label warning = new Label("Are you sure you
want to delete this form?");
                confirmFormTable.setWidget(0, 1, warning);

        confirmFormTable.getFlexCellFormatter().setHorizontalAlign
ment(0, 1, HasHorizontalAlignment.ALIGN_CENTER);

        confirmFormTable.getFlexCellFormatter().setColSpan(0, 1, 3);

                confirmsFormDelete = new Button("Yes");
                confirmFormTable.setWidget(1, 1,
confirmsFormDelete);

        confirmFormTable.getFlexCellFormatter().setHorizontalAlign
ment(1, 1, HasHorizontalAlignment.ALIGN_RIGHT);

                rejectsFormDelete = new Button("No");
                confirmFormTable.setWidget(1, 2,
rejectsFormDelete);
        }

        public void setupPaginator(int currPage,int totalPage, boolean
isFirstPage, boolean isLastPage){
                firstPageButton.setEnabled(!isFirstPage);
                prevButton.setEnabled(!isFirstPage);
                pageNumber.setText("Page "+currPage+" of
"+totalPage);
                nextButton.setEnabled(!isLastPage);
                lastPageButton.setEnabled(!isLastPage);
        }

        public void loadFormsList(ArrayList<DataEntryForm> forms,
String username, Boolean isProjAdmin){
                clearRows();
                if(forms.isEmpty()){
                        formsListTable.setText(2, 0, "No data
entry form/s to display");

        formsListTable.getFlexCellFormatter().setColSpan(2, 0, 4);

        formsListTable.getFlexCellFormatter().setHorizontalAlignment
(2, 0, HasHorizontalAlignment.ALIGN_CENTER);
                }else{
                        for(int row = 0; row < forms.size();
row++){
                                if(isProjAdmin){

        formsListTable.setWidget(row+2, 0, new RadioButton(""));
                                }else{

        formsListTable.setText(row+2, 0, " ");
                                }

        formsListTable.getFlexCellFormatter().setWidth(row+2, 0,
"20px");
//
        formsListTable.setWidget(row+2, 1, new HTML("<a
href='"+GWT.getHostPageBaseURL()+"dataLoad/index.zul?u="+Base64Co
der.encodeString(forms.get(row).creator.username)+
```

```
//
		"&p="+Base64Coder.encodeString(forms.get(row).projID.projI
D.toString())+
//
		"&f="+Base64Coder.encodeString(forms.get(row).formID.toSt
ring())+
//
		"&l="+Base64Coder.encodeString(GWT.getModuleBaseURL(
))+
//
		"' target='_blank'>"+forms.get(row).formName+"</a>"));
//
		formsListTable.setWidget(row+2, 1, new HTML("<a
href='http://localhost:8080/dataLoad/index.zul?u="+Base64Coder.encodeStr
ing(forms.get(row).creator.username)+
//
		"&p="+Base64Coder.encodeString(forms.get(row).projID.projI
D.toString())+
//
		"&f="+Base64Coder.encodeString(forms.get(row).formID.toSt
ring())+
//
		"&l="+Base64Coder.encodeString(GWT.getModuleBaseURL(
))+
//
		"' target='_blank'>"+forms.get(row).formName+"</a>"));

		formsListTable.setWidget(row+2, 1, new
Hyperlink(forms.get(row).formName,"dataLoad?p="+forms.get(row).projID
.projID

		+"&u="+Base64Coder.encodeString(username)+"&id="+forms
.get(row).formID));

		formsListTable.setText(row+2, 2,
forms.get(row).creator.username);

		formsListTable.getFlexCellFormatter().setHorizontalAlignment
(row+2, 2, HasHorizontalAlignment.ALIGN_CENTER);
                                                DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy EEEE");

		formsListTable.setText(row+2, 3,
formatted.format(forms.get(row).dateCreated));

		formsListTable.getFlexCellFormatter().setHorizontalAlignment
(row+2, 3, HasHorizontalAlignment.ALIGN_CENTER);
                                        }
                        }
                }

                private void clearRows(){
                        int tableCount =formsListTable.getRowCount();
                        for(int row = tableCount-1; row>=2; row--){
                                formsListTable.removeRow(row);
                        }
                }

                public Integer getSelectedRow() {
                        for (int i = 2; i < formsListTable.getRowCount();
++i) {
                                        RadioButton radioButton =
(RadioButton)formsListTable.getWidget(i, 0);
                                        if(radioButton.getValue()){
                                                return i;
                                        }



                        }
                        return null;
                }

                public void clearRowStyles(){
                        for(int row = 2; row <
formsListTable.getRowCount(); row++){

		formsListTable.getRowFormatter().removeStyleName(row,
"tableHighlight");
                        }
                }

                public void edit_Warning(int left, int top){
                        DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                        warning.setTitle("Warning");
                        warning.setWidget(new Label("Choose 1 form"));
                        warning.setPopupPosition(left, top);
                        warning.show();
                }

                public void del_Warning(int left, int top){
                        DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                        warning.setTitle("Warning");
                        warning.setWidget(new Label("Choose 1 form"));
                        warning.setPopupPosition(left, top);
                        warning.show();
                }

                public void formsPerPage_Warning(int left, int top){
                        DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                        warning.setTitle("Warning");
                        warning.setWidget(new Label("Minimum no. of
forms per page is 1"));
                        warning.setPopupPosition(left, top);
                        warning.show();
                }

                public HasClickHandlers getGoToFirstPageButton(){
                        return firstPageButton;
                }

                public HasClickHandlers getGoToLastPageButton(){
                        return lastPageButton;
                }

                public HasClickHandlers getPrevButton(){
                        return prevButton;
                }

                public HasClickHandlers getNextButton(){
                        return nextButton;
                }

                public HasClickHandlers getSearchButton(){
                        return searchButton;
                }

                public HasClickHandlers getCancelSearchButton(){
                        return cancelSearchButton;
                }

                public HasKeyDownHandlers getSearchKeyHandler(){
                        return search;
                }

                public HasClickHandlers getFormsPerPageButton(){
                        return formsPerPageButton;
                }

                public HasKeyDownHandlers getFormsPerPageKeyHandler(){
                        return intBoxFormsPerPage;
                }

                public String getSearchText(){
                        return search.getText();
                }

                public void clearSearchText(){
                        this.search.setText("");
                }

                public Integer getFormsPerPageText(){
                        return intBoxFormsPerPage.getValue();
                }

                public void clearFormsPerPageText(){
                        this.intBoxFormsPerPage.setText("");
                }
```

```
	public HasClickHandlers getAddButton(){
		return addButton;
	}

	public HasClickHandlers getEditButton(){
		return editButton;
	}

	public HasClickHandlers getDelButton(){
		return delButton;
	}

	public void showConfirmFormDeleteWarning(){
		confirmFormDelete.center();
		confirmFormDelete.setWidth("400px");
		confirmFormDelete.show();
	}

	public void hideConfirmFormDeleteWarning(){
		confirmFormDelete.hide();
	}

	public HasClickHandlers getConfirmFormDeleteButton(){
		return confirmsFormDelete;
	}

	public HasClickHandlers getRejectFormDeleteButton(){
		return rejectsFormDelete;
	}

	public void showButtonVisibility(Boolean visible){
		buttons.setVisible(visible);
	}

	public FlexTable getTable(){
		return formsListTable;
	}

	public Widget asWidget() {
		return this;
	}

}
```

ViewMiscFilesView.java

```
package cepir.client.view;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import cepir.client.ImageResources;
import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.*;

public class ViewMiscFilesView extends Composite implements
cepir.client.presenter.ViewMiscFilesPresenter.Display{
	private FlexTable mainFlexTable;
	private FlexTable miscFilesListTable;
	private TextBox search;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private PushButton addFolderButton;
	private PushButton addFileButton;
	private PushButton addLinkButton;

	public ViewMiscFilesView() {
		mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		miscFilesListTable = new FlexTable();
		mainFlexTable.setWidget(1, 0,
miscFilesListTable);

		FlexTable buttonTable = new FlexTable();
		miscFilesListTable.setWidget(0, 0, buttonTable);

		ImageResources images =
GWT.create(ImageResources.class);
		addFolderButton = new PushButton(new
Image(images.addFolder().getUrl()));
		addFolderButton.setTitle("Add folder");
		addFolderButton.setStyleName("icon");
		buttonTable.setWidget(0, 0, addFolderButton);

		addFileButton = new PushButton(new
Image(images.file().getUrl()));
		addFileButton.setTitle("Add file");
		addFileButton.setStyleName("icon");
		buttonTable.setWidget(0, 1, addFileButton);

		addLinkButton = new PushButton(new
Image(images.globe().getUrl()));
		addLinkButton.setTitle("Add link");
		addLinkButton.setStyleName("icon");
		buttonTable.setWidget(0, 2, addLinkButton);

		FlexTable searchTable = new FlexTable();
		miscFilesListTable.setWidget(0, 2, searchTable);

	miscFilesListTable.getFlexCellFormatter().setHorizontalAlign
ment(0, 2, HasHorizontalAlignment.ALIGN_RIGHT);

	miscFilesListTable.getFlexCellFormatter().setColSpan(0, 2, 2);

		search = new TextBox();
		searchTable.setWidget(0, 0, search);

		searchButton = new PushButton(new
Image(images.search().getUrl()));
		searchButton.setTitle("Search");
		searchButton.setStyleName("icon");
		searchTable.setWidget(0, 1, searchButton);

		cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
		cancelSearchButton.setTitle("Cancel");
		cancelSearchButton.setStyleName("icon");
		searchTable.setWidget(0, 2, cancelSearchButton);

		Label lblName = new Label("Name");
		miscFilesListTable.setWidget(1, 0, lblName);
		miscFilesListTable.getCellFormatter().setWidth(1,
0, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.4))+"px");

		Label lblCreator = new Label("Creator");
		miscFilesListTable.setWidget(1, 1, lblCreator);
		miscFilesListTable.getCellFormatter().setWidth(1,
1, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

		Label lblDateCreated = new Label("Date
Created");
		miscFilesListTable.setWidget(1, 2,
lblDateCreated);
		miscFilesListTable.getCellFormatter().setWidth(1,
2, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.3))+"px");

	miscFilesListTable.getRowFormatter().setStyleName(1,
"tableHeader");

		Label lblAction = new Label("Action");
		miscFilesListTable.setWidget(1, 3, lblAction);
		miscFilesListTable.getCellFormatter().setWidth(1,
3, Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.1))+"px");
	}

	public void loadEverything(ArrayList<MiscFile> files,
ArrayList<MiscFileFolder> folders, String path, Integer projID){
		clearRows();
		ImageResources images =
GWT.create(ImageResources.class);
```

```
                        if(!path.isEmpty()){
                                FlexTable upFolder = new
FlexTable();
                                miscFilesListTable.setWidget(2, 0,
upFolder);
                                upFolder.addStyleName("cursor");

        miscFilesListTable.getFlexCellFormatter().setColSpan(2, 0, 4);

        miscFilesListTable.getFlexCellFormatter().setHorizontalAlign
ment(2, 0, HasHorizontalAlignment.ALIGN_LEFT);
                                Image up = new
Image(images.upFolder().getUrl());
                                upFolder.setWidget(0, 0, up);
                                upFolder.setText(0, 1, "..");

        miscFilesListTable.getFlexCellFormatter().addStyleName(2, 0,
"recordBorder");

        miscFilesListTable.getFlexCellFormatter().addStyleName(2, 1,
"recordBorder");

        miscFilesListTable.getFlexCellFormatter().addStyleName(2, 2,
"recordBorder");

        miscFilesListTable.getFlexCellFormatter().addStyleName(2, 3,
"recordBorder");
                        }
                        if(!folders.isEmpty()){
                                for(int row = 0; row < folders.size();
row++){
                                        FlexTable folder = new
FlexTable();

        miscFilesListTable.setWidget(row+3, 0, folder);

        miscFilesListTable.getCellFormatter().setStyleName(row+3,0,"
recordBorder");
                                        folder.setWidget(0, 0,
new Image(images.folder().getUrl()));

        folder.getFlexCellFormatter().setRowSpan(0, 0, 2);
                                        folder.setWidget(0, 1,
new Hyperlink(folders.get(row).miscfileFolderName,
"miscFiles/"+projID+"/"+path+folders.get(row).miscfileFolderID+"/"));
                                        folder.setText(1, 0,
folders.get(row).miscfileFolderDesc);


        miscFilesListTable.setText(row+3, 1,
folders.get(row).creator.username);

        miscFilesListTable.getCellFormatter().setStyleName(row+3,1,"
recordBorder");

        miscFilesListTable.getCellFormatter().setHorizontalAlignment(
row+3,1,HasHorizontalAlignment.ALIGN_CENTER);
                                        DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy hh:mm a");

        miscFilesListTable.setText(row+3, 2,
formatted.format(folders.get(row).dateCreated));

        miscFilesListTable.getCellFormatter().setStyleName(row+3,2,"
recordBorder");

        miscFilesListTable.getCellFormatter().setHorizontalAlignment(
row+3,2,HasHorizontalAlignment.ALIGN_CENTER);

                                        FlexTable buttons = new
FlexTable();

        miscFilesListTable.setWidget(row+3, 3, buttons);

        miscFilesListTable.getCellFormatter().setStyleName(row+3,3,"
recordBorder");

        miscFilesListTable.getCellFormatter().setHorizontalAlignment(
row+3,3,HasHorizontalAlignment.ALIGN_CENTER);
                                        Hyperlink editFolder =
new Hyperlink("Edit",
"editMFolder?id="+folders.get(row).miscfileFolderID+"&p="+projID);
                                        buttons.setWidget(0, 0,
editFolder);

                                        Hyperlink deleteFolder =
new Hyperlink("Delete",
"delMFolder?id="+folders.get(row).miscfileFolderID+"&path="+path+"&p
="+projID);
                                        buttons.setWidget(0, 1,
deleteFolder);
                                }
                        }
                        if(!files.isEmpty()){
                                for(int row = 0; row < files.size();
row++){
                                        FlexTable miscFile =
new FlexTable();

        miscFilesListTable.setWidget(row+folders.size()+3, 0,
miscFile);

        miscFilesListTable.getFlexCellFormatter().setStyleName(row+
folders.size()+3, 0, "recordBorder");
                                        miscFile.setWidget(0, 0,
(files.get(row).type==1?getIcon(files.get(row).miscFileName):new
Image(images.link().getUrl())));

        miscFile.getFlexCellFormatter().setRowSpan(0, 0, 3);

        if(files.get(row).type==1){

        miscFile.setHTML(0, 1, "<a
href='"+GWT.getModuleBaseURL()+"download?f="+files.get(row).miscFil
eID+"&t="+"misc_file"+"'
target='_blank'>"+files.get(row).miscFileName+"</a>");

        miscFile.setHTML(2, 0,
(files.get(row).miscFileLink.isEmpty()||files.get(row).equals("http://")?"<br/
>":"More information at <a href='"+files.get(row).miscFileLink+"'
target='_blank'>"+files.get(row).miscFileLink+"</a>"));
                                        }else{

        miscFile.setHTML(0, 1, "<a
href='"+files.get(row).miscFileLink+"'
target='_blank'>"+files.get(row).miscFileName+"</a>");
                                        }
                                        miscFile.setText(1, 0,
files.get(row).miscFileDesc);


        miscFilesListTable.setText(row+folders.size()+3, 1,
files.get(row).creator.username);

        miscFilesListTable.getCellFormatter().setStyleName(row+fold
ers.size()+3,1,"recordBorder");

        miscFilesListTable.getCellFormatter().setHorizontalAlignment(
row+folders.size()+3,1,HasHorizontalAlignment.ALIGN_CENTER);
                                        DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy hh:mm a");

        miscFilesListTable.setText(row+folders.size()+3, 2,
formatted.format(files.get(row).dateCreated));

        miscFilesListTable.getCellFormatter().setStyleName(row+fold
ers.size()+3,2,"recordBorder");

        miscFilesListTable.getCellFormatter().setHorizontalAlignment(
row+folders.size()+3,2,HasHorizontalAlignment.ALIGN_CENTER);

                                        FlexTable buttons = new
FlexTable();

        miscFilesListTable.setWidget(row+folders.size()+3, 3,
buttons);

        miscFilesListTable.getCellFormatter().setStyleName(row+fold
ers.size()+3,3,"recordBorder");
```

```
                miscFilesListTable.getCellFormatter().setHorizontalAlignment(
row+folders.size()+3,3,HasHorizontalAlignment.ALIGN_CENTER);

                                                Hyperlink editFolder =
new Hyperlink("Edit",
"editM"+(files.get(row).type==1?"File":"Link")+"?id="+files.get(row).misc
FileID+"&p="+projID);
                                                buttons.setWidget(0, 0,
editFolder);

                                                Hyperlink deleteFolder =
new Hyperlink("Delete",
"delMFile?id="+files.get(row).miscFileID+"&path="+path+"&p="+projID);
                                                buttons.setWidget(0, 2,
deleteFolder);
                                        }
                                }

                if(files.isEmpty()&&folders.isEmpty()&&path.isEmpty()){
                                        miscFilesListTable.setText(3, 0, "No
files or folders to display");

                miscFilesListTable.getFlexCellFormatter().setColSpan(3, 0, 4);

                miscFilesListTable.getFlexCellFormatter().setHorizontalAlign
ment(3, 0, HasHorizontalAlignment.ALIGN_CENTER);
                                }
                }

                private Image getIcon(String toolName){
                                ImageResources images =
GWT.create(ImageResources.class);
                                if(toolName.endsWith(".avi")){
                                        return new
Image(images.avi().getUrl());
                                }else if(toolName.endsWith(".bmp")){
                                        return new
Image(images.bmp().getUrl());
                                }else
if(toolName.endsWith(".doc")||toolName.endsWith(".docx")){
                                        return new
Image(images.doc().getUrl());
                                }else if(toolName.endsWith(".gif")){
                                        return new
Image(images.gif().getUrl());
                                }else
if(toolName.endsWith(".htm")||toolName.endsWith(".html")){
                                        return new
Image(images.html().getUrl());
                                }else
if(toolName.endsWith(".jpg")||toolName.endsWith(".jpeg")){
                                        return new
Image(images.jpg().getUrl());
                                }else if(toolName.endsWith(".mp3")){
                                        return new
Image(images.mp3().getUrl());
                                }else if(toolName.endsWith(".mpeg")){
                                        return new
Image(images.mpeg().getUrl());
                                }else if(toolName.endsWith(".pdf")){
                                        return new
Image(images.pdf().getUrl());
                                }else if(toolName.endsWith(".png")){
                                        return new
Image(images.png().getUrl());
                                }else
if(toolName.endsWith(".ppt")||toolName.endsWith(".pptx")){
                                        return new
Image(images.ppt().getUrl());
                                }else if(toolName.endsWith(".rar")){
                                        return new
Image(images.rar().getUrl());
                                }else if(toolName.endsWith(".tar.gz")){
                                        return new
Image(images.tar().getUrl());
                                }else if(toolName.endsWith(".txt")){
                                        return new
Image(images.txt().getUrl());
                                }else if(toolName.endsWith(".wav")){
                                        return new
Image(images.wav().getUrl());
                                }else if(toolName.endsWith(".wma")){
                                        return new
Image(images.wma().getUrl());
                                }else
if(toolName.endsWith(".xls")||toolName.endsWith(".xlsx")){
                                        return new
Image(images.xls().getUrl());
                                }else if(toolName.endsWith(".zip")){
                                        return new
Image(images.zip().getUrl());
                                }else{
                                        return new
Image(images.blank().getUrl());
                                }
                }

                public void setupTracker(String currLoc, List<String> names,
Integer projID){
                                FlexTable tracker = new FlexTable();
                                mainFlexTable.setWidget(0, 0, tracker);
                                tracker.setWidget(0, 0, new
Hyperlink("Miscellaneous Files", "miscFiles/"+projID+"/"));
                                List<String> parse =
Arrays.asList(currLoc.split("/"));
                                for(int i = 0; i < parse.size(); i++){
                                        String prevPath ="";
                                        for(int j = 0; j <= i; j++){
                                                prevPath +=
parse.get(j)+"/";
                                        }
                                        if(parse.get(i).length()!=0){
                                                tracker.setHTML(0,
(i*2)+1, "&raquo;");
                                                tracker.setWidget(0,
(i*2)+2, new Hyperlink(names.get(i), "miscFiles/"+projID+"/"+prevPath));
                                        }
                                }
                }

                private void clearRows(){
                                int tableCount
=miscFilesListTable.getRowCount();
                                for(int row = tableCount-1; row>=2; row--){
                                        miscFilesListTable.removeRow(row);
                                }
                }

                public String getSearchTxt(){
                                return search.getText();
                }

                public void clearSearchTxt(){
                                search.setText("");
                }

                public HasClickHandlers getAddFolderButton(){
                                return addFolderButton;
                }

                public HasClickHandlers getAddFileButton(){
                                return addFileButton;
                }

                public HasClickHandlers getAddLinkButton(){
                                return addLinkButton;
                }

                public HasClickHandlers getSearchButton(){
                                return searchButton;
                }

                public HasClickHandlers getCancelSearchButton(){
                                return cancelSearchButton;
                }

                public HasKeyDownHandlers getSearchKeyHandler(){
                                return search;
                }
```

```
	public FlexTable getMiscFilesList(){
		return miscFilesListTable;
	}

	public Widget asWidget(){
		return this;
	}

}
```

ViewMyProjsView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.client.ImageResources;
import cepir.client.presenter.ViewMyProjsPresenter;
import cepir.shared.Project;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.KeyDownEvent;
import com.google.gwt.event.dom.client.KeyDownHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DecoratedPopupPanel;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.IntegerBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.Widget;

public class ViewMyProjsView extends Composite implements
		ViewMyProjsPresenter.Display {
	private Label pageNumber;
	private PushButton nextButton;
	private PushButton prevButton;
	private PushButton firstPageButton;
	private PushButton lastPageButton;
	private IntegerBox intBoxProjsPerPage;
	private Button projsPerPageButton;
	private FlexTable projListTable;
	private FlexTable paginator;
	private FlexTable projsPerPage;

	public ViewMyProjsView() {
		FlexTable mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);
		mainFlexTable.setWidth(Integer
				.toString((int)
(Window.getClientWidth() * 0.8 * 0.75))
				+ "px");

		Label lblMyProjects = new Label("My Projects");
		lblMyProjects.setStyleName("header");
		mainFlexTable.setWidget(0, 0, lblMyProjects);

	mainFlexTable.getCellFormatter().setVerticalAlignment(0, 0,

	HasVerticalAlignment.ALIGN_TOP);

	mainFlexTable.getCellFormatter().setHorizontalAlignment(0,
0,

	HasHorizontalAlignment.ALIGN_LEFT);

		// list of projects
		projListTable = new FlexTable();
		mainFlexTable.setWidget(1, 0, projListTable);

		Label lblName = new Label("Name");
		lblName.setStyleName("tableHeader");
		projListTable.setWidget(0, 0, lblName);
		projListTable
				.getFlexCellFormatter()
				.setWidth(

	0,

	0,

	Integer

			.toString((int) (Window.getClientWidth() * 0.8 *
0.75 * 0.2))

			+ "px");

	projListTable.getCellFormatter().setHorizontalAlignment(0, 0,

	HasHorizontalAlignment.ALIGN_CENTER);

		Label lblDescription = new Label("Description");
		lblDescription.setStyleName("tableHeader");
		projListTable.setWidget(0, 1, lblDescription);
		projListTable
				.getFlexCellFormatter()
				.setWidth(

	0,

	1,

	Integer

			.toString((int) (Window.getClientWidth() * 0.8 *
0.75 * 0.6))

			+ "px");

	projListTable.getCellFormatter().setHorizontalAlignment(0, 1,

	HasHorizontalAlignment.ALIGN_CENTER);

		Label lblPosition = new Label("Position");
		lblPosition.setStyleName("tableHeader");
		projListTable.setWidget(0, 2, lblPosition);
		projListTable
				.getFlexCellFormatter()
				.setWidth(

	0,

	2,

	Integer

			.toString((int) (Window.getClientWidth() * 0.8 *
0.75 * 0.2))

			+ "px");

	projListTable.getCellFormatter().setHorizontalAlignment(0, 2,

	HasHorizontalAlignment.ALIGN_CENTER);

		ImageResources images =
GWT.create(ImageResources.class);

		// paginator
		paginator = new FlexTable();
		mainFlexTable.setWidget(2, 0, paginator);

	mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(2, 0,

	HasHorizontalAlignment.ALIGN_CENTER);
		firstPageButton = new PushButton(new
Image(images.backToStart()
				.getUrl()));
		firstPageButton.setStyleName("icon");
		firstPageButton.setTitle("Go to first page");
		paginator.setWidget(0, 0, firstPageButton);
```

```
                prevButton = new PushButton(new
Image(images.previous().getUrl()));
                prevButton.setStyleName("icon");
                prevButton.setTitle("Previous");
                paginator.setWidget(0, 1, prevButton);

                pageNumber = new Label("");
                paginator.setWidget(0, 2, pageNumber);

                nextButton = new PushButton(new
Image(images.next().getUrl()));
                nextButton.setStyleName("icon");
                nextButton.setTitle("Next");
                paginator.setWidget(0, 3, nextButton);

                lastPageButton = new PushButton(new
Image(images.goToLast().getUrl()));
                lastPageButton.setStyleName("icon");
                lastPageButton.setTitle("Go to last page");
                paginator.setWidget(0, 4, lastPageButton);

                // projects per page
                projsPerPage = new FlexTable();
                mainFlexTable.setWidget(3, 0, projsPerPage);

        mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(3, 0,

        HasHorizontalAlignment.ALIGN_CENTER);
                Label lblUsersPerPage = new Label("No. of users
per page");
                projsPerPage.setWidget(0, 0, lblUsersPerPage);

                intBoxProjsPerPage = new IntegerBox();
                intBoxProjsPerPage.setWidth("30px");
                intBoxProjsPerPage.addKeyDownHandler(new
KeyDownHandler() {

                        public void
onKeyDown(KeyDownEvent event) {

        if(event.getNativeKeyCode()==13){

        projsPerPageButton.click();
                                }
                        }
                });
                projsPerPage.setWidget(0, 1,
intBoxProjsPerPage);

                projsPerPageButton = new Button("Go");
                projsPerPage.setWidget(0, 2, projsPerPageButton);

                // view all projects link
                Hyperlink hprlnkViewAllProjects = new
Hyperlink(
                                "View all projects at
CEpiR", false, "allProjs");
                mainFlexTable.setWidget(4, 0,
hprlnkViewAllProjects);

        mainFlexTable.getFlexCellFormatter().setHorizontalAlignment
(4, 0,

        HasHorizontalAlignment.ALIGN_CENTER);
        }

        public void setupPaginator(int currPage, int totalPage,
                        boolean isFirstPage, boolean
isLastPage) {
                firstPageButton.setEnabled(!isFirstPage);
                prevButton.setEnabled(!isFirstPage);
                pageNumber.setText("Page " + currPage + " of " +
totalPage);
                nextButton.setEnabled(!isLastPage);
                lastPageButton.setEnabled(!isLastPage);
        }

        public void loadProjsList(ArrayList<Project> projects) {
                clearRows();
                if (projects.isEmpty()) {
                        projListTable.setWidget(1, 0, new
Label("No project/s to display"));

        projListTable.getFlexCellFormatter().setColSpan(1, 0, 3);

        projListTable.getFlexCellFormatter().setHorizontalAlignment(
1, 0,

        HasHorizontalAlignment.ALIGN_CENTER);
                } else {
                        for (int row = 0; row < projects.size();
row++) {
                                String[] parse =
projects.get(row).projName.split("=");

        projListTable.setWidget(row + 1, 0, new Hyperlink(parse[0],

        "projHome=" + projects.get(row).projID));

        projListTable.getFlexCellFormatter().setStyleName(row + 1, 0,

        "recordBorder");

        projListTable.setText(row + 1, 1, projects.get(row).projDesc);

        projListTable.getFlexCellFormatter().setStyleName(row + 1, 1,

        "recordBorder");

        projListTable.getFlexCellFormatter().setHorizontalAlignment(

        row + 1, 1, HasHorizontalAlignment.ALIGN_JUSTIFY);
                                if (parse[1].equals("1"))
{

        projListTable.setText(row + 1, 2, "Member");
                                } else {

        projListTable.setText(row + 1, 2, "Administrator");
                                }

        projListTable.getFlexCellFormatter().setHorizontalAlignment(

        row + 1, 2, HasHorizontalAlignment.ALIGN_CENTER);

        projListTable.getFlexCellFormatter().setStyleName(row + 1, 2,

        "recordBorder");
                        }
                }
        }

        public void clearRows() {
                int tableCount = projListTable.getRowCount();
                for (int row = tableCount - 1; row >= 1; row--) {
                        projListTable.removeRow(row);
                }
        }

        public Integer getProjsPerPageText() {
                return intBoxProjsPerPage.getValue();
        }

        public void clearProjsPerPageText() {
                this.intBoxProjsPerPage.setText("");
        }

        public void usersPerPage_Warning(int left, int top) {
                DecoratedPopupPanel warning = new
DecoratedPopupPanel(true);
                warning.setTitle("Warning");
                warning.setWidget(new Label("Minimum no. of
projects per page is 1"));
                warning.setPopupPosition(left, top);
                warning.show();
        }

        public HasClickHandlers getGoToFirstPageButton() {
                return firstPageButton;
```

```
	}

	public HasClickHandlers getGoToLastPageButton() {
		return lastPageButton;
	}

	public HasClickHandlers getPrevButton() {
		return prevButton;
	}

	public HasClickHandlers getNextButton() {
		return nextButton;
	}

	public HasClickHandlers getProjsPerPageButton() {
		return projsPerPageButton;
	}

	public Widget asWidget() {
		return this;
	}

}
```

ViewProjMemRequestView.java

```
package cepir.client.view;

import cepir.client.presenter.ViewProjMemRequestPresenter;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;

public class ViewProjMemRequestView extends Composite implements
ViewProjMemRequestPresenter.Display{
	private Label lblUsernameTxt;
	private Button approveButton;
	private Button rejectButton;
	private Label lblMessageTxt;
	private ListBox comboBox;
	private Label lblProjTxt;

	public ViewProjMemRequestView() {

		FlexTable mainTable = new FlexTable();
		initWidget(mainTable);

	mainTable.setWidth(Integer.toString((int)(Window.getClientW
idth()*0.8*0.75*0.9))+"px");
		//0.8(center width), 0.75 (remaining space in the
center table)

		Label lblViewProjectMembership = new
Label("View Project Membership Request");

	lblViewProjectMembership.setStyleName("header2");
		mainTable.setWidget(0, 0,
lblViewProjectMembership);

		Label lblUsername = new Label("Username");
		mainTable.setWidget(1, 0, lblUsername);

		lblUsernameTxt = new Label("");
		mainTable.setWidget(1, 1, lblUsernameTxt);

		Label lblMessage = new Label("Message");
		mainTable.setWidget(2, 0, lblMessage);

	mainTable.getFlexCellFormatter().setVerticalAlignment(2, 0,
HasVerticalAlignment.ALIGN_TOP);

		lblMessageTxt = new Label("");
		mainTable.setWidget(2, 1, lblMessageTxt);

		Label lblProject = new Label("Project");
		mainTable.setWidget(3, 0, lblProject);

	mainTable.getFlexCellFormatter().setVerticalAlignment(3, 0,
HasVerticalAlignment.ALIGN_TOP);

		lblProjTxt = new Label("");
		mainTable.setWidget(3, 1, lblProjTxt);

		Label lblMembershipType = new
Label("Membership Type");
		lblMembershipType.setWidth("150px");
		mainTable.setWidget(4, 0, lblMembershipType);

		comboBox = new ListBox();
		comboBox.addItem("Member");
		comboBox.addItem("Administrator");
		mainTable.setWidget(4, 1, comboBox);
		mainTable.getFlexCellFormatter().setColSpan(0,
0, 2);

		FlexTable flexTable = new FlexTable();
		mainTable.setWidget(5, 1, flexTable);

		approveButton = new Button("New button");
		approveButton.setText("Approve");
		flexTable.setWidget(0, 0, approveButton);

		rejectButton = new Button("New button");
		rejectButton.setText("Reject");
		flexTable.setWidget(0, 1, rejectButton);

		Hyperlink hprlnkBack = new Hyperlink("<<
Back", false, "allPendingReq");
		mainTable.setWidget(6, 1, hprlnkBack);

	mainTable.getCellFormatter().setHorizontalAlignment(6, 1,
HasHorizontalAlignment.ALIGN_RIGHT);
	}

	public void setUsernameTxt(String text){
		lblUsernameTxt.setText(text);
	}

	public void setProjTxt(String text){
		lblProjTxt.setText(text);
	}

	public void setMessageTxt(String text){
		lblMessageTxt.setText(text);
	}

	public Integer setSelectedMemberType(){
		return comboBox.getSelectedIndex();
	}

	public HasClickHandlers getApproveButton(){
		return approveButton;
	}

	public HasClickHandlers getRejectButton(){
		return rejectButton;
	}


	public Widget asWidget(){
		return this;
	}
}
```

ViewProjView.java

```
package cepir.client.view;

import java.util.ArrayList;

import cepir.shared.User;
import cepir.client.ImageResources;
```

```
import cepir.client.presenter.ViewProjPresenter;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.event.logical.shared.HasValueChangeHandlers;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.MultiWordSuggestOracle;
import com.google.gwt.user.client.ui.ScrollPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.Widget;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.SuggestBox;

public class ViewProjView extends Composite implements
ViewProjPresenter.Display{
	private TextArea textBoxProjDesc;
	private TextBox textBoxProjName;
	private Label dateCreated;
	private Label dateUpdated;
	private Label lblDateCreated;
	private Label lblLastUpdated;
	private Button saveButton;
	private Button cancelButton;
	private Button chooseAdmins;
	private FlexTable mainFlexTable;
	private Label lblError;
	private Label lblProjNameError;
	private Label lblProjAdminsError;
	private Label lblHeader;
	private DialogBox selectAdmin;
	private FlexTable selectAdminsTable;
	private Button selectOkButton;
	private Button selectProjAdminButton;
	private Button deselectProjAdminButton;
	private ListBox projAdminsList;
	private ListBox usersList;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private TextBox search;

	public ViewProjView() {

		mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		lblHeader = new Label("");
		lblHeader.setStyleName("header2");
		mainFlexTable.setWidget(0, 0, lblHeader);

		Label lblRequiredField = new Label("All fields are
required");
		mainFlexTable.setWidget(1, 1, lblRequiredField);

		Label lblProjectName = new Label("Project
Name");
		mainFlexTable.setWidget(2, 0, lblProjectName);

		Label lblProjectDescription = new Label("Project
Description (500 characters max)");
		lblProjectDescription.setWidth("130px");
		mainFlexTable.setWidget(3, 0,
lblProjectDescription);

	mainFlexTable.getFlexCellFormatter().setVerticalAlignment(3,
0, HasVerticalAlignment.ALIGN_TOP);

		textBoxProjDesc = new TextArea();
		textBoxProjDesc.setVisibleLines(3);
		textBoxProjDesc.setWidth("300px");
		mainFlexTable.setWidget(3, 1, textBoxProjDesc);

		Label lblProjectAdministrator = new
Label("Project Administrator");
		mainFlexTable.setWidget(4, 0,
lblProjectAdministrator);

		chooseAdmins = new Button("Select");
		mainFlexTable.setWidget(4, 1, chooseAdmins);

		dateCreated = new Label("");
		mainFlexTable.setWidget(5, 1, dateCreated);


	mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

		dateUpdated = new Label("");
		mainFlexTable.setWidget(6, 1, dateUpdated);

		FlexTable buttonPanel = new FlexTable();
		mainFlexTable.setWidget(7, 1, buttonPanel);

		saveButton = new Button("Save");
		buttonPanel.setWidget(0, 0, saveButton);

		cancelButton = new Button("Cancel");
		buttonPanel.setWidget(0, 1, cancelButton);

		lblError = new Label("");
		mainFlexTable.setWidget(8, 1, lblError);

		lblProjNameError = new Label("");
		mainFlexTable.setWidget(9, 1, lblProjNameError);

		//project admin/s dialog box
		selectAdmin = new DialogBox();
		selectAdmin.setText("Choose project
administrator/s");
		selectAdmin.setAnimationEnabled(true);
		selectAdmin.setGlassEnabled(true);

		selectAdminsTable = new FlexTable();
		selectAdmin.setWidget(selectAdminsTable);

		selectAdminsTable.setWidget(0, 0, new
Label("Project Administrator/s"));
		projAdminsList = new ListBox(true);
		projAdminsList.setWidth("250px");
		projAdminsList.setHeight("150px");
		selectAdminsTable.setWidget(1, 0,
projAdminsList);

		FlexTable center = new FlexTable();
		selectAdminsTable.setWidget(1, 1, center);
		selectProjAdminButton = new Button("<");
		center.setWidget(0, 0, selectProjAdminButton);

		deselectProjAdminButton = new Button(">");
		center.setWidget(1, 0, deselectProjAdminButton);

		selectAdminsTable.setWidget(0, 2, new
Label("Users"));

		FlexTable searchFlexTable = new FlexTable();
		selectAdminsTable.setWidget(0, 3,
searchFlexTable);

	selectAdminsTable.getFlexCellFormatter().setHorizontalAlign
ment(0, 3, HasHorizontalAlignment.ALIGN_RIGHT);

		search = new TextBox();
		searchFlexTable.setWidget(0, 0, search);

		ImageResources images =
GWT.create(ImageResources.class);
		searchButton = new PushButton(new
Image(images.search().getUrl()));
		searchButton.setStyleName("icon");
		searchButton.setTitle("Search");
		searchFlexTable.setWidget(0, 1, searchButton);
```

```
                cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
                cancelSearchButton.setStyleName("icon");
                cancelSearchButton.setTitle("Cancel");
                searchFlexTable.setWidget(0, 2,
cancelSearchButton);

                usersList = new ListBox(true);
                usersList.setWidth("250px");
                usersList.setHeight("150px");
                selectAdminsTable.setWidget(1, 2, usersList);

        selectAdminsTable.getFlexCellFormatter().setColSpan(1, 2, 2);

                selectOkButton = new Button("Ok");
                selectAdminsTable.setWidget(2, 1,
selectOkButton);

                lblProjAdminsError = new Label();
                lblProjAdminsError.setStyleName("errorMsg");
                selectAdminsTable.setWidget(3, 0,
lblProjAdminsError);

        selectAdminsTable.getFlexCellFormatter().setColSpan(3, 0, 3);

        selectAdminsTable.getFlexCellFormatter().setHorizontalAlign
ment(3, 0, HasHorizontalAlignment.ALIGN_CENTER);
        }

        public void setHeader(String header){
                lblHeader.setText(header);
        }

        public void setProjName(String projName){
                textBoxProjName = new TextBox();
                textBoxProjName.setText(projName);
                textBoxProjName.setWidth("300px");
                mainFlexTable.setWidget(2, 1, textBoxProjName);
        }

        public String getProjName(){
                return textBoxProjName.getText();
        }

        public void setProjDesc(String desc){
                textBoxProjDesc.setText(desc);
        }

        public String getProjDesc(){
                return textBoxProjDesc.getText();
        }

        public void setSearchTxt(String text){
                search.setText(text);
        }

        public String getSearchTxt(){
                return search.getText();
        }

        public void setErrorText(String err){
                lblError.setText(err);
                lblError.setStyleName("errorMsg");
        }

        public void setProjAdminErrorText(String err){
                lblProjAdminsError.setText(err);
        }

        public void setProjNameErrorText(String err){
                lblProjNameError.setText(err);
                lblProjNameError.setStyleName("errorMsg");
        }

        public void setDateCreated(String date){
                lblDateCreated = new Label("Date Created");
                mainFlexTable.setWidget(5, 0, lblDateCreated);
                dateCreated.setText(date);
        }

        public void setDateUpdated(String date){
                lblLastUpdated = new Label("Last Updated");
                mainFlexTable.setWidget(6, 0, lblLastUpdated);
                dateUpdated.setText(date);
        }

        public void loadUsersList(ArrayList<User> users){
                usersList.clear();
                for(int row = 0;row<users.size();row++){

        usersList.addItem(users.get(row).username+"
("+users.get(row).firstName+" "+users.get(row).lastName+")");
                }
        }

        public void loadProjAdminsList(ArrayList<User> users){
                projAdminsList.clear();
                for(int row = 0;row<users.size();row++){

        projAdminsList.addItem(users.get(row).username+"
("+users.get(row).firstName+" "+users.get(row).lastName+")");
                }
        }

        public ArrayList<Integer> getSelectedAdmins(){
                ArrayList<Integer> selectedRows = new
ArrayList<Integer>();
                for(int ctr = 0;
ctr<projAdminsList.getItemCount(); ctr++){

        if(projAdminsList.isItemSelected(ctr)){
                                        selectedRows.add(ctr);
                                }
                        }
                        return selectedRows;
        }

        public ArrayList<Integer> getSelectedUsers(){
                ArrayList<Integer> selectedRows = new
ArrayList<Integer>();
                for(int ctr = 0; ctr<usersList.getItemCount();
ctr++){
                        if(usersList.isItemSelected(ctr)){
                                selectedRows.add(ctr);
                        }
                }
                return selectedRows;
        }

        public void showProjAdminSelect(){
                selectAdmin.center();
                selectAdmin.show();
        }

        public void hideProjAdminSelect(){
                selectAdmin.hide();
        }

        public HasClickHandlers getSaveButton(){
                return saveButton;
        }

        public HasClickHandlers getCancelButton(){
                return cancelButton;
        }

        public HasClickHandlers getSearchButton(){
                return searchButton;
        }

        public HasClickHandlers getCancelSearchButton(){
                return cancelSearchButton;
        }

        public HasClickHandlers getChooseAdmins(){
                return chooseAdmins;
        }

        public HasClickHandlers getSelectOkButton(){
```

```
		return selectOkButton;
	}

	public HasClickHandlers getSelectAdminButton(){
		return selectProjAdminButton;
	}

	public HasClickHandlers getDeselectAdminButton(){
		return deselectProjAdminButton;
	}

	public HasKeyDownHandlers getSearchKeyHandler(){
		return search;
	}

	public Widget asWidget(){
		return this;
	}

}
```

ViewReferencesView.java

```
package cepir.client.view;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import cepir.client.ImageResources;
import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;
import cepir.shared.Reference;
import cepir.shared.ReferenceFolder;
import cepir.shared.ReferenceType;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.safehtml.shared.SafeHtml;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.*;

public class ViewReferencesView extends Composite implements
cepir.client.presenter.ViewReferencesPresenter.Display{
	private FlexTable mainFlexTable;
	private FlexTable refsListTable;
	private TextBox search;
	private PushButton searchButton;
	private PushButton cancelSearchButton;
	private PushButton addFolderButton;
	private PushButton addFileButton;

	public ViewReferencesView() {
		mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);

		refsListTable = new FlexTable();
		mainFlexTable.setWidget(1, 0, refsListTable);

		FlexTable buttonTable = new FlexTable();
		refsListTable.setWidget(0, 0, buttonTable);

		ImageResources images =
GWT.create(ImageResources.class);
		addFolderButton = new PushButton(new
Image(images.addFolder().getUrl()));
		addFolderButton.setTitle("Add folder");
		addFolderButton.setStyleName("icon");
		buttonTable.setWidget(0, 0, addFolderButton);

		addFileButton = new PushButton(new
Image(images.file().getUrl()));
		addFileButton.setTitle("Add reference");
		addFileButton.setStyleName("icon");
		buttonTable.setWidget(0, 1, addFileButton);

		FlexTable searchTable = new FlexTable();
		refsListTable.setWidget(0, 2, searchTable);
		refsListTable.getFlexCellFormatter().setHorizontalAlignment(0
, 2, HasHorizontalAlignment.ALIGN_RIGHT);

		refsListTable.getFlexCellFormatter().setColSpan(0, 2, 2);

			search = new TextBox();
			searchTable.setWidget(0, 0, search);

			searchButton = new PushButton(new
Image(images.search().getUrl()));
			searchButton.setTitle("Search");
			searchButton.setStyleName("icon");
			searchTable.setWidget(0, 1, searchButton);

			cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
			cancelSearchButton.setTitle("Cancel");
			cancelSearchButton.setStyleName("icon");
			searchTable.setWidget(0, 2, cancelSearchButton);

			Label lblName = new Label("Name");
			refsListTable.setWidget(1, 0, lblName);
			refsListTable.getCellFormatter().setWidth(1, 0,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.4))+"px");

			Label lblCreator = new Label("Author");
			refsListTable.setWidget(1, 1, lblCreator);
			refsListTable.getCellFormatter().setWidth(1, 1,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

			Label lblDateCreated = new Label("Upload
Details");
			refsListTable.setWidget(1, 2, lblDateCreated);
			refsListTable.getCellFormatter().setWidth(1, 2,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.3))+"px");
			refsListTable.getRowFormatter().setStyleName(1,
"tableHeader");

			Label lblAction = new Label("Action");
			refsListTable.setWidget(1, 3, lblAction);
			refsListTable.getCellFormatter().setWidth(1, 3,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.1))+"px");
		}

	public void loadEverything(ArrayList<Reference> files,
ArrayList<ReferenceFolder> folders, String path, Integer projID){
			clearRows();
			ImageResources images =
GWT.create(ImageResources.class);
			if(!path.isEmpty()){
				FlexTable upFolder = new
FlexTable();
				refsListTable.setWidget(2, 0,
upFolder);
				upFolder.addStyleName("cursor");

		refsListTable.getFlexCellFormatter().setColSpan(2, 0, 4);

		refsListTable.getFlexCellFormatter().setHorizontalAlignment(2
, 0, HasHorizontalAlignment.ALIGN_LEFT);
				Image up = new
Image(images.upFolder().getUrl());
				upFolder.setWidget(0, 0, up);
				upFolder.setText(0, 1, "..");

		refsListTable.getFlexCellFormatter().addStyleName(2, 0,
"recordBorder");

		refsListTable.getFlexCellFormatter().addStyleName(2, 1,
"recordBorder");

		refsListTable.getFlexCellFormatter().addStyleName(2, 2,
"recordBorder");

		refsListTable.getFlexCellFormatter().addStyleName(2, 3,
"recordBorder");
			}
			if(!folders.isEmpty()){
```

```
                        for(int row = 0; row < folders.size();
row++){
                                FlexTable folder = new
FlexTable();

            refsListTable.setWidget(row+3, 0, folder);

            refsListTable.getCellFormatter().setStyleName(row+3,0,"recor
dBorder");
                                folder.setWidget(0, 0,
new Image(images.folder().getUrl()));

            folder.getFlexCellFormatter().setRowSpan(0, 0, 2);
                                folder.setWidget(0, 1,
new Hyperlink(folders.get(row).refFolderName,
"refs/"+projID+"/"+path+folders.get(row).refFolderID+"/"));
                                folder.setText(1, 0,
folders.get(row).refFolderDesc);


            refsListTable.setText(row+3, 1,
folders.get(row).creator.username);

            refsListTable.getCellFormatter().setStyleName(row+3,1,"recor
dBorder");

            refsListTable.getCellFormatter().setHorizontalAlignment(row+
3,1,HasHorizontalAlignment.ALIGN_CENTER);
                                DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy hh:mm a");

            refsListTable.setText(row+3, 2,
formatted.format(folders.get(row).dateCreated));

            refsListTable.getCellFormatter().setStyleName(row+3,2,"recor
dBorder");

            refsListTable.getCellFormatter().setHorizontalAlignment(row+
3,2,HasHorizontalAlignment.ALIGN_CENTER);

                                FlexTable buttons = new
FlexTable();

            refsListTable.setWidget(row+3, 3, buttons);

            refsListTable.getCellFormatter().setStyleName(row+3,3,"recor
dBorder");

            refsListTable.getCellFormatter().setHorizontalAlignment(row+
3,3,HasHorizontalAlignment.ALIGN_CENTER);

                                Hyperlink editFolder =
new Hyperlink("Edit",
"editRFolder?id="+folders.get(row).refFolderID+"&p="+projID);
                                buttons.setWidget(0, 0,
editFolder);

                                Hyperlink deleteFolder =
new Hyperlink("Delete",
"delRFolder?id="+folders.get(row).refFolderID+"&path="+path+"&p="+pr
ojID);
                                buttons.setWidget(0, 1,
deleteFolder);
                        }
                }
                if(!files.isEmpty()){
                        for(int row = 0; row < files.size();
row++){
                                FlexTable ref = new
FlexTable();

            refsListTable.setWidget(row+folders.size()+3, 0, ref);

            refsListTable.getFlexCellFormatter().setStyleName(row+folder
s.size()+3, 0, "recordBorder");
                                ref.setWidget(0, 0,
(!files.get(row).refName.isEmpty()?getIcon(files.get(row).refName):new
Image(images.link().getUrl())));

            ref.getFlexCellFormatter().setRowSpan(0, 0, 2);
                                ref.setWidget(0, 1, new
Hyperlink(getTitle(files.get(row)),
"viewRef?id="+files.get(row).refID+"&p="+projID));


            refsListTable.setText(row+folders.size()+3, 1,
files.get(row).author);

            refsListTable.getCellFormatter().setStyleName(row+folders.siz
e()+3,1,"recordBorder");

            refsListTable.getCellFormatter().setHorizontalAlignment(row+
folders.size()+3,1,HasHorizontalAlignment.ALIGN_CENTER);
                                DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy hh:mm a");

            refsListTable.setHTML(row+folders.size()+3, 2, "Uploaded by
"+files.get(row).creator.username+" on
"+formatted.format(files.get(row).dateCreated));

            refsListTable.getCellFormatter().setStyleName(row+folders.siz
e()+3,2,"recordBorder");

            refsListTable.getCellFormatter().setHorizontalAlignment(row+
folders.size()+3,2,HasHorizontalAlignment.ALIGN_CENTER);

                                FlexTable buttons = new
FlexTable();

            refsListTable.setWidget(row+folders.size()+3, 3, buttons);

            refsListTable.getCellFormatter().setStyleName(row+folders.siz
e()+3,3,"recordBorder");

            refsListTable.getCellFormatter().setHorizontalAlignment(row+
folders.size()+3,3,HasHorizontalAlignment.ALIGN_CENTER);

                                Hyperlink editFolder =
new Hyperlink("Edit",
"editRFile?id="+files.get(row).refID+"&p="+projID);
                                buttons.setWidget(0, 0,
editFolder);

                                Hyperlink deleteFolder =
new Hyperlink("Delete",
"delRFile?id="+files.get(row).refID+"&path="+path+"&p="+projID);
                                buttons.setWidget(0, 2,
deleteFolder);
                        }
                }

            if(files.isEmpty()&&folders.isEmpty()&&path.isEmpty()){
                        refsListTable.setText(3, 0, "No files
or folders to display");

            refsListTable.getFlexCellFormatter().setColSpan(3, 0, 4);

            refsListTable.getFlexCellFormatter().setHorizontalAlignment(3
, 0, HasHorizontalAlignment.ALIGN_CENTER);
                }
            }

            private String getTitle(Reference ref) {

            if(ref.type.equals(ReferenceType.Journal_Article)){
                        return ref.articleName;
                }else
if(ref.type.equals(ReferenceType.Book_Chapter)){
                        return ref.chapTitle;
                }else{
                        return ref.title;
                }
            }

            private Image getIcon(String toolName){
                    ImageResources images =
GWT.create(ImageResources.class);
                if(toolName.endsWith(".avi")){
                        return new
Image(images.avi().getUrl());
```

```
                        }else if(toolName.endsWith(".bmp")){
                                return new
Image(images.bmp().getUrl());
                        }else
if(toolName.endsWith(".doc")||toolName.endsWith(".docx")){
                                return new
Image(images.doc().getUrl());
                        }else if(toolName.endsWith(".gif")){
                                return new
Image(images.gif().getUrl());
                        }else
if(toolName.endsWith(".htm")||toolName.endsWith(".html")){
                                return new
Image(images.html().getUrl());
                        }else
if(toolName.endsWith(".jpg")||toolName.endsWith(".jpeg")){
                                return new
Image(images.jpg().getUrl());
                        }else if(toolName.endsWith(".mp3")){
                                return new
Image(images.mp3().getUrl());
                        }else if(toolName.endsWith(".mpeg")){
                                return new
Image(images.mpeg().getUrl());
                        }else if(toolName.endsWith(".pdf")){
                                return new
Image(images.pdf().getUrl());
                        }else if(toolName.endsWith(".png")){
                                return new
Image(images.png().getUrl());
                        }else
if(toolName.endsWith(".ppt")||toolName.endsWith(".pptx")){
                                return new
Image(images.ppt().getUrl());
                        }else if(toolName.endsWith(".rar")){
                                return new
Image(images.rar().getUrl());
                        }else if(toolName.endsWith(".tar.gz")){
                                return new
Image(images.tar().getUrl());
                        }else if(toolName.endsWith(".txt")){
                                return new
Image(images.txt().getUrl());
                        }else if(toolName.endsWith(".wav")){
                                return new
Image(images.wav().getUrl());
                        }else if(toolName.endsWith(".wma")){
                                return new
Image(images.wma().getUrl());
                        }else
if(toolName.endsWith(".xls")||toolName.endsWith(".xlsx")){
                                return new
Image(images.xls().getUrl());
                        }else if(toolName.endsWith(".zip")){
                                return new
Image(images.zip().getUrl());
                        }else{
                                return new
Image(images.blank().getUrl());
                        }
                }

                public void setupTracker(String currLoc, List<String> names,
Integer projID){
                        FlexTable tracker = new FlexTable();
                        mainFlexTable.setWidget(0, 0, tracker);
                        tracker.setWidget(0, 0, new
Hyperlink("References", "refs/"+projID+"/"));
                        List<String> parse =
Arrays.asList(currLoc.split("/"));
                        for(int i = 0; i < parse.size(); i++){
                                String prevPath ="";
                                for(int j = 0; j <= i; j++){
                                        prevPath +=
parse.get(j)+"/";
                                }
                                if(parse.get(i).length()!=0){
                                        tracker.setHTML(0,
(i*2)+1, "&raquo;");
                                        tracker.setWidget(0,
(i*2)+2, new Hyperlink(names.get(i), "refs/"+projID+"/"+prevPath));
                                }
                        }
                }

                private void clearRows(){
                        int tableCount =refsListTable.getRowCount();
                        for(int row = tableCount-1; row>=2; row--){
                                refsListTable.removeRow(row);
                        }
                }

                public String getSearchTxt(){
                        return search.getText();
                }

                public void clearSearchTxt(){
                        search.setText("");
                }

                public HasClickHandlers getAddFolderButton(){
                        return addFolderButton;
                }

                public HasClickHandlers getAddFileButton(){
                        return addFileButton;
                }

                public HasClickHandlers getSearchButton(){
                        return searchButton;
                }

                public HasClickHandlers getCancelSearchButton(){
                        return cancelSearchButton;
                }

                public HasKeyDownHandlers getSearchKeyHandler(){
                        return search;
                }

                public FlexTable getReferencesList(){
                        return refsListTable;
                }

                public Widget asWidget(){
                        return this;
                }

}
```

ViewReferenceView.java

```
package cepir.client.view;

import cepir.shared.Reference;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.HasMouseOverHandlers;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.ui.*;

public class ViewReferenceView extends Composite implements
cepir.client.presenter.ViewReferencePresenter.Display{
        private FlexTable mainFlexTable;
        private Label titleTxt;
        private Label chapTitleTxt;
        private Label journTitleTxt;
        private Label authorTxt;
        private Label refTypeTxt;
        private Label bookTitleTxt;
        private Label editorTxt;
        private Label publisherTxt;
        private Label pubDateTxt;
        private Label pubPlaceTxt;
        private Label volIssuePgTxt;
        private Label articleNameTxt;
        private Label yearTxt;
        private Label schoolTxt;
        private Label uploadDateTxt;
        private Label lblAccessReferenceHere;
```

```
	private HTML linkTxt;
	private HTML downloadTxt;

	public ViewReferenceView() {
		mainFlexTable = new FlexTable();
		initWidget(mainFlexTable);
		mainFlexTable.setVisible(false);
		mainFlexTable.getColumnFormatter().setWidth(0,
"250px");

		HTML html = new HTML("<br/>", true);
		mainFlexTable.setWidget(1, 0, html);

		Label lblTitle = new Label("Title");
		mainFlexTable.setWidget(2, 0, lblTitle);

		titleTxt = new Label();
		titleTxt.setStyleName("idLabel");
		mainFlexTable.setWidget(2, 1, titleTxt);

		Label lblChapTitle = new Label("Chapter Title");
		mainFlexTable.setWidget(3, 0, lblChapTitle);

		chapTitleTxt = new Label();
		chapTitleTxt.setStyleName("idLabel");
		mainFlexTable.setWidget(3, 1, chapTitleTxt);

		Label lblArticleName = new Label("Journal
Article Title");
		mainFlexTable.setWidget(4, 0, lblArticleName);

		articleNameTxt = new Label();
		articleNameTxt.setStyleName("idLabel");
		mainFlexTable.setWidget(4, 1, articleNameTxt);

		Label lblJournalTitle = new Label("Journal
Name");
		mainFlexTable.setWidget(5, 0, lblJournalTitle);

		journTitleTxt = new Label();
		mainFlexTable.setWidget(5, 1, journTitleTxt);

		Label lblAuthor = new Label("Author/s");
		mainFlexTable.setWidget(6, 0, lblAuthor);

		authorTxt = new Label();
		mainFlexTable.setWidget(6, 1, authorTxt);

		Label lblReferenceType = new Label("Reference
Type");
		mainFlexTable.setWidget(7, 0, lblReferenceType);

		refTypeTxt = new Label();
		mainFlexTable.setWidget(7, 1, refTypeTxt);

		Label lblBookTitle = new Label("Book Title");
		mainFlexTable.setWidget(8, 0, lblBookTitle);

		bookTitleTxt = new Label();
		mainFlexTable.setWidget(8, 1, bookTitleTxt);

		Label lblEditor = new Label("Editor");
		mainFlexTable.setWidget(9, 0, lblEditor);

		editorTxt = new Label();
		mainFlexTable.setWidget(9, 1, editorTxt);

		Label lblPublisher = new Label("Publisher");
		mainFlexTable.setWidget(10, 0, lblPublisher);

		publisherTxt = new Label();
		mainFlexTable.setWidget(10, 1, publisherTxt);

		Label lblYearOfPublication = new
Label("Publication Date");
		lblYearOfPublication.setWidth("150px");
		mainFlexTable.setWidget(11, 0,
lblYearOfPublication);

		pubDateTxt = new Label();
		mainFlexTable.setWidget(11, 1, pubDateTxt);

		Label lblPubPlace = new Label("Place of
Publication");
		mainFlexTable.setWidget(12, 0, lblPubPlace);

		pubPlaceTxt = new Label();
		mainFlexTable.setWidget(12, 1, pubPlaceTxt);

		Label lblVolume = new Label("Volume, Issue,
Page Number");
		mainFlexTable.setWidget(13, 0, lblVolume);

		volIssuePgTxt = new Label();
		mainFlexTable.setWidget(13, 1, volIssuePgTxt);

		Label lblYear = new Label("Year");
		mainFlexTable.setWidget(14, 0, lblYear);

		yearTxt = new Label();
		mainFlexTable.setWidget(14, 1, yearTxt);

		Label lblSchool = new Label("School");
		mainFlexTable.setWidget(15, 0, lblSchool);

		schoolTxt = new Label();
		mainFlexTable.setWidget(15, 1, schoolTxt);

		HTML html_1 = new HTML("<br/>", true);
		mainFlexTable.setWidget(16, 0, html_1);

		Label lblDateUploaded = new Label("Date
Uploaded");
		mainFlexTable.setWidget(17, 0,
lblDateUploaded);

		uploadDateTxt = new Label();
		mainFlexTable.setWidget(17, 1, uploadDateTxt);

		lblAccessReferenceHere = new Label("Access
Reference Here");
		mainFlexTable.setWidget(18, 0,
lblAccessReferenceHere);

		linkTxt = new HTML();
		mainFlexTable.setWidget(18, 1, linkTxt);

		downloadTxt = new HTML();
		mainFlexTable.setWidget(19, 1, downloadTxt);
	}

	public void hideAllFields(){
		for(int i = mainFlexTable.getRowCount(); i >=2;
i++){

	mainFlexTable.getRowFormatter().setVisible(i, false);
		}
	}

	public void showBook(){
		mainFlexTable.setVisible(true);
		mainFlexTable.getRowFormatter().setVisible(2,
true);
		mainFlexTable.getRowFormatter().setVisible(6,
true);
		mainFlexTable.getRowFormatter().setVisible(10,
true);
		mainFlexTable.getRowFormatter().setVisible(11,
true);
		mainFlexTable.getRowFormatter().setVisible(12,
true);

		mainFlexTable.getRowFormatter().setVisible(7,
true);
		mainFlexTable.getRowFormatter().setVisible(17,
true);
		mainFlexTable.getRowFormatter().setVisible(18,
true);
		mainFlexTable.getRowFormatter().setVisible(19,
true);
```

```
	}

	public void showBookChap(){
		mainFlexTable.setVisible(true);
		mainFlexTable.getRowFormatter().setVisible(3,
true);
		mainFlexTable.getRowFormatter().setVisible(6,
true);
		mainFlexTable.getRowFormatter().setVisible(8,
true);
		mainFlexTable.getRowFormatter().setVisible(9,
true);
		mainFlexTable.getRowFormatter().setVisible(10,
true);
		mainFlexTable.getRowFormatter().setVisible(11,
true);

		mainFlexTable.getRowFormatter().setVisible(7,
true);
		mainFlexTable.getRowFormatter().setVisible(17,
true);
		mainFlexTable.getRowFormatter().setVisible(18,
true);
		mainFlexTable.getRowFormatter().setVisible(19,
true);
	}

	public void showJournal(){
		mainFlexTable.setVisible(true);
		mainFlexTable.getRowFormatter().setVisible(4,
true);
		mainFlexTable.getRowFormatter().setVisible(5,
true);
		mainFlexTable.getRowFormatter().setVisible(6,
true);
		mainFlexTable.getRowFormatter().setVisible(11,
true);
		mainFlexTable.getRowFormatter().setVisible(13,
true);

		mainFlexTable.getRowFormatter().setVisible(7,
true);
		mainFlexTable.getRowFormatter().setVisible(17,
true);
		mainFlexTable.getRowFormatter().setVisible(18,
true);
		mainFlexTable.getRowFormatter().setVisible(19,
true);
	}

	public void showEArticleOrUnpublishedWork(){
		mainFlexTable.setVisible(true);
		mainFlexTable.getRowFormatter().setVisible(2,
true);
		mainFlexTable.getRowFormatter().setVisible(6,
true);

		mainFlexTable.getRowFormatter().setVisible(7,
true);
		mainFlexTable.getRowFormatter().setVisible(17,
true);
		mainFlexTable.getRowFormatter().setVisible(18,
true);
		mainFlexTable.getRowFormatter().setVisible(19,
true);
	}

	public void showThesis(){
		mainFlexTable.setVisible(true);
		mainFlexTable.getRowFormatter().setVisible(2,
true);
		mainFlexTable.getRowFormatter().setVisible(6,
true);
		mainFlexTable.getRowFormatter().setVisible(14,
true);
		mainFlexTable.getRowFormatter().setVisible(15,
true);

		mainFlexTable.getRowFormatter().setVisible(7,
true);
		mainFlexTable.getRowFormatter().setVisible(17,
true);
		mainFlexTable.getRowFormatter().setVisible(18,
true);
		mainFlexTable.getRowFormatter().setVisible(19,
true);
	}

	public void loadReference(Reference ref){
		Hyperlink back = new Hyperlink();
		back.setHTML("&laquo; Back");

	back.setTargetHistoryToken("refs/"+ref.project.projID+"/"+ref.
refPath);
		mainFlexTable.setWidget(0, 0, back);
		titleTxt.setText(ref.title);
		chapTitleTxt.setText(ref.chapTitle);
		journTitleTxt.setText(ref.journTitle);
		authorTxt.setText(ref.author);
		refTypeTxt.setText(ref.type.toString());
		bookTitleTxt.setText(ref.bookTitle);
		editorTxt.setText(ref.editor);
		publisherTxt.setText( ref.publisher);
		pubDateTxt.setText(ref.pubDate);
		pubPlaceTxt.setText(ref.pubPlace);
		volIssuePgTxt.setText((ref.volume.isEmpty()?"-
":ref.volume)+ " , " + (ref.issue.isEmpty()?"-":ref.issue) + " , " +
(ref.pageNum.isEmpty()?"-":ref.pageNum));
		articleNameTxt.setText(ref.articleName);
		yearTxt.setText(ref.year);
		schoolTxt.setText(ref.school);
		DateTimeFormat formatted =
DateTimeFormat.getFormat("MMMM dd, yyyy EEEE");

	uploadDateTxt.setText(formatted.format(ref.dateCreated));
		linkTxt.setHTML("<a
href='"+ref.refLink+"'>"+ref.refLink+"</a>");
		if(ref.refLink.trim().isEmpty()){
			lblAccessReferenceHere.setText("");
		}
		if(!ref.refName.isEmpty()){
			downloadTxt.setHTML("<a
href='"+GWT.getModuleBaseURL()+"download?f="+ref.refID+"&t="+"ref
erence"+"' target='_blank'>Download</a>");
		}
		for(int i = 2;
i<mainFlexTable.getRowCount();i++){

	mainFlexTable.getRowFormatter().setVisible(i, false);
		}
	}

	public Widget asWidget(){
		return this;
	}

}
```

ViewToolFolderView.java

```
 package cepir.client.view;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Widget;

public class ViewToolFolderView extends Composite implements
cepir.client.presenter.ViewToolFolderPresenter.Display{
	private Button saveButton;
	private Button cancelButton;
	private Label errorTxt;
	private Label lblHeader;
	private TextBox nameTxt;
	private TextArea descTxt;

	public ViewToolFolderView() {
```

```
                FlexTable mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);

                lblHeader = new Label("");
                lblHeader.setStyleName("header2");
                mainFlexTable.setWidget(0, 0, lblHeader);

                Label lblAllFieldsAre = new Label("All fields are
required");
                mainFlexTable.setWidget(1, 1, lblAllFieldsAre);

                Label lblFolderName = new Label("Folder
Name");
                mainFlexTable.setWidget(2, 0, lblFolderName);

                nameTxt = new TextBox();
                nameTxt.setWidth("250px");
                mainFlexTable.setWidget(2, 1, nameTxt);

                Label lblDescription = new Label("Description");
                mainFlexTable.setWidget(3, 0, lblDescription);

        mainFlexTable.getFlexCellFormatter().setColSpan(0, 0, 2);

                descTxt = new TextArea();
                descTxt.setWidth("250px");
                mainFlexTable.setWidget(3, 1, descTxt);

                FlexTable buttonTable = new FlexTable();
                mainFlexTable.setWidget(4, 1, buttonTable);

                saveButton = new Button("Save");
                buttonTable.setWidget(0, 0, saveButton);

                cancelButton = new Button("Cancel");
                buttonTable.setWidget(0, 1, cancelButton);

                errorTxt = new Label("");
                errorTxt.setStyleName("errorMsg");
                mainFlexTable.setWidget(5, 1, errorTxt);
        }

        public void setHeader(String text){
                lblHeader.setText(text);
        }

        public void setName(String text){
                nameTxt.setText(text);
        }

        public String getName(){
                return nameTxt.getText();
        }

        public String getDesc(){
                return descTxt.getText();
        }

        public void setDesc(String text){
                descTxt.setText(text);
        }

        public void setErrorTxt(String text){
                errorTxt.setText(text);
        }

        public HasClickHandlers getSaveButton(){
                return saveButton;
        }

        public HasClickHandlers getCancelButton(){
                return cancelButton;
        }

        public Widget asWidget(){
                return this;
        }

}
```

ViewToolsView.java

```
package cepir.client.view;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import cepir.client.ImageResources;
import cepir.shared.Tool;
import cepir.shared.ToolFolder;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.DoubleClickEvent;
import com.google.gwt.event.dom.client.DoubleClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.dom.client.HasKeyDownHandlers;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.Hyperlink;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.PushButton;
import com.google.gwt.user.client.ui.Widget;

public class ViewToolsView extends Composite implements
cepir.client.presenter.ViewToolsPresenter.Display{
        private FlexTable mainFlexTable;
        private FlexTable toolsListTable;
        private FlexTable buttonTable;
        private TextBox search;
        private PushButton searchButton;
        private PushButton cancelSearchButton;
        private PushButton addFolderButton;
        private PushButton addFileButton;
        private PushButton addLinkButton;

        public ViewToolsView() {

                mainFlexTable = new FlexTable();
                initWidget(mainFlexTable);

                toolsListTable = new FlexTable();
                mainFlexTable.setWidget(1, 0, toolsListTable);

                buttonTable = new FlexTable();
                buttonTable.setVisible(false);
                toolsListTable.setWidget(0, 0, buttonTable);

                ImageResources images =
GWT.create(ImageResources.class);
                addFolderButton = new PushButton(new
Image(images.addFolder().getUrl()));
                addFolderButton.setTitle("Add folder");
                addFolderButton.setStyleName("icon");
                buttonTable.setWidget(0, 0, addFolderButton);

                addFileButton = new PushButton(new
Image(images.file().getUrl()));
                addFileButton.setTitle("Add file");
                addFileButton.setStyleName("icon");
                buttonTable.setWidget(0, 1, addFileButton);

                addLinkButton = new PushButton(new
Image(images.globe().getUrl()));
                addLinkButton.setTitle("Add link");
                addLinkButton.setStyleName("icon");
                buttonTable.setWidget(0, 2, addLinkButton);
```

```
                        FlexTable searchTable = new FlexTable();
                        toolsListTable.setWidget(0, 2, searchTable);

        toolsListTable.getFlexCellFormatter().setHorizontalAlignment(
0, 2, HasHorizontalAlignment.ALIGN_RIGHT);

        toolsListTable.getFlexCellFormatter().setColSpan(0, 2, 2);

                        search = new TextBox();
                        searchTable.setWidget(0, 0, search);

                        searchButton = new PushButton(new
Image(images.search().getUrl()));
                        searchButton.setTitle("Search");
                        searchButton.setStyleName("icon");
                        searchTable.setWidget(0, 1, searchButton);

                        cancelSearchButton = new PushButton(new
Image(images.cancel().getUrl()));
                        cancelSearchButton.setTitle("Cancel");
                        cancelSearchButton.setStyleName("icon");
                        searchTable.setWidget(0, 2, cancelSearchButton);

                        Label lblName = new Label("Name");
                        toolsListTable.setWidget(1, 0, lblName);
                        toolsListTable.getCellFormatter().setWidth(1, 0,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.4))+"px");

                        Label lblCreator = new Label("Creator");
                        toolsListTable.setWidget(1, 1, lblCreator);
                        toolsListTable.getCellFormatter().setWidth(1, 1,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.2))+"px");

                        Label lblDateCreated = new Label("Date
Created");
                        toolsListTable.setWidget(1, 2, lblDateCreated);
                        toolsListTable.getCellFormatter().setWidth(1, 2,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.3))+"px");

        toolsListTable.getRowFormatter().setStyleName(1,
"tableHeader");

                        Label lblAction = new Label("Action");
                        toolsListTable.setWidget(1, 3, lblAction);
                        toolsListTable.getCellFormatter().setWidth(1, 3,
Integer.toString((int)(Window.getClientWidth()*0.8*0.75*0.1))+"px");

        }

        public void loadEverything(ArrayList<ToolFolder> folders,
ArrayList<Tool> tools, String path, String isToolAd){
                        clearRows();
                        ImageResources images =
GWT.create(ImageResources.class);
                        if(!path.isEmpty()){
                                FlexTable upFolder = new
FlexTable();
                                toolsListTable.setWidget(2, 0,
upFolder);
                                upFolder.addStyleName("cursor");

        toolsListTable.getFlexCellFormatter().setColSpan(2, 0, 4);

        toolsListTable.getFlexCellFormatter().setHorizontalAlignment(
2, 0, HasHorizontalAlignment.ALIGN_LEFT);
                                Image up = new
Image(images.upFolder().getUrl());
                                upFolder.setWidget(0, 0, up);
                                upFolder.setText(0, 1, "..");

        toolsListTable.getFlexCellFormatter().addStyleName(2, 0,
"recordBorder");

        toolsListTable.getFlexCellFormatter().addStyleName(2, 1,
"recordBorder");

        toolsListTable.getFlexCellFormatter().addStyleName(2, 2,
"recordBorder");

        toolsListTable.getFlexCellFormatter().addStyleName(2, 3,
"recordBorder");
                        }
                        if(!folders.isEmpty()){
                                for(int row = 0; row < folders.size();
row++){
                                        FlexTable folder = new
FlexTable();

        toolsListTable.setWidget(row+3, 0, folder);

        toolsListTable.getCellFormatter().setStyleName(row+3,0,"reco
rdBorder");
                                        folder.setWidget(0, 0,
new Image(images.folder().getUrl()));

        folder.getFlexCellFormatter().setRowSpan(0, 0, 2);
                                        folder.setWidget(0, 1,
new Hyperlink(folders.get(row).toolFolderName,
"tools/"+path+folders.get(row).toolFolderID+"/"));
                                        folder.setText(1, 0,
folders.get(row).toolFolderDesc);


        toolsListTable.setText(row+3, 1,
folders.get(row).creator.username);

        toolsListTable.getCellFormatter().setStyleName(row+3,1,"reco
rdBorder");

        toolsListTable.getCellFormatter().setHorizontalAlignment(row
+3,1,HasHorizontalAlignment.ALIGN_CENTER);
                                        DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy hh:mm a");

        toolsListTable.setText(row+3, 2,
formatted.format(folders.get(row).dateCreated));

        toolsListTable.getCellFormatter().setStyleName(row+3,2,"reco
rdBorder");

        toolsListTable.getCellFormatter().setHorizontalAlignment(row
+3,2,HasHorizontalAlignment.ALIGN_CENTER);


        toolsListTable.getCellFormatter().setStyleName(row+3,3,"reco
rdBorder");

        if(isToolAd.equalsIgnoreCase("yes")){
                                                FlexTable
buttons = new FlexTable();

        toolsListTable.setWidget(row+3, 3, buttons);

        toolsListTable.getCellFormatter().setHorizontalAlignment(row
+3,3,HasHorizontalAlignment.ALIGN_CENTER);

                                                Hyperlink
editFolder = new Hyperlink("Edit",
"editTFolder?id="+folders.get(row).toolFolderID);

        buttons.setWidget(0, 0, editFolder);

                                                Hyperlink
deleteFolder = new Hyperlink("Delete",
"delTFolder?id="+folders.get(row).toolFolderID+"&path="+path);

        buttons.setWidget(0, 1, deleteFolder);
                                        }
                                }
                        }
                        if(!tools.isEmpty()){
                                for(int row = 0; row < tools.size();
row++){
                                        FlexTable tool = new
FlexTable();

        toolsListTable.setWidget(row+folders.size()+3, 0, tool);
```

```
toolsListTable.getFlexCellFormatter().setStyleName(row+folde
rs.size()+3, 0, "recordBorder");
                                        tool.setWidget(0, 0,
(tools.get(row).type==1?getIcon(tools.get(row).toolName):new
Image(images.link().getUrl())));

        tool.getFlexCellFormatter().setRowSpan(0, 0, 3);

        if(tools.get(row).type==1){

        tool.setHTML(0, 1, "<a
href='"+GWT.getModuleBaseURL()+"download?f="+tools.get(row).toolID
+"&t="+"tool"+"' target='_blank'>"+tools.get(row).toolName+"</a>");

        tool.setHTML(2, 0,
(tools.get(row).toolLink.isEmpty()||tools.get(row).equals("http://")?"<br/>":"
More information at <a href='"+tools.get(row).toolLink+"'
target='_blank'>"+tools.get(row).toolLink+"</a>"));
                                        }else{

        tool.setHTML(0, 1, "<a href='"+tools.get(row).toolLink+"'
target='_blank'>"+tools.get(row).toolName+"</a>");
                                        }
                                        tool.setText(1, 0,
tools.get(row).toolDesc);


        toolsListTable.setText(row+folders.size()+3, 1,
tools.get(row).creator.username);

        toolsListTable.getCellFormatter().setStyleName(row+folders.si
ze()+3,1,"recordBorder");

        toolsListTable.getCellFormatter().setHorizontalAlignment(row
+folders.size()+3,1,HasHorizontalAlignment.ALIGN_CENTER);
                                        DateTimeFormat
formatted = DateTimeFormat.getFormat("MMMM dd, yyyy hh:mm a");

        toolsListTable.setText(row+folders.size()+3, 2,
formatted.format(tools.get(row).dateCreated));

        toolsListTable.getCellFormatter().setStyleName(row+folders.si
ze()+3,2,"recordBorder");

        toolsListTable.getCellFormatter().setHorizontalAlignment(row
+folders.size()+3,2,HasHorizontalAlignment.ALIGN_CENTER);


        toolsListTable.getCellFormatter().setStyleName(row+folders.si
ze()+3,3,"recordBorder");

        if(isToolAd.equalsIgnoreCase("yes")){
                                                FlexTable
buttons = new FlexTable();

        toolsListTable.setWidget(row+folders.size()+3, 3, buttons);

        toolsListTable.getCellFormatter().setHorizontalAlignment(row
+folders.size()+3,3,HasHorizontalAlignment.ALIGN_CENTER);

                                                Hyperlink
editFolder = new Hyperlink("Edit",
"edit"+(tools.get(row).type==1?"Tool":"Link")+"?id="+tools.get(row).toolI
D);

        buttons.setWidget(0, 0, editFolder);

                                                Hyperlink
deleteFolder = new Hyperlink("Delete",
"delTool?id="+tools.get(row).toolID+"&path="+path);

        buttons.setWidget(0, 2, deleteFolder);
                                                }
                                }
                        }

        if(tools.isEmpty()&&folders.isEmpty()&&path.isEmpty()){
                                toolsListTable.setText(3, 0, "No files
or folders to display");

        toolsListTable.getFlexCellFormatter().setColSpan(3, 0, 4);

        toolsListTable.getFlexCellFormatter().setHorizontalAlignment(
3, 0, HasHorizontalAlignment.ALIGN_CENTER);
                        }
        }

        private Image getIcon(String toolName){
                        ImageResources images =
GWT.create(ImageResources.class);
                        if(toolName.endsWith(".avi")){
                                return new
Image(images.avi().getUrl());
                        }else if(toolName.endsWith(".bmp")){
                                return new
Image(images.bmp().getUrl());
                        }else
if(toolName.endsWith(".doc")||toolName.endsWith(".docx")){
                                return new
Image(images.doc().getUrl());
                        }else if(toolName.endsWith(".gif")){
                                return new
Image(images.gif().getUrl());
                        }else
if(toolName.endsWith(".htm")||toolName.endsWith(".html")){
                                return new
Image(images.html().getUrl());
                        }else
if(toolName.endsWith(".jpg")||toolName.endsWith(".jpeg")){
                                return new
Image(images.jpg().getUrl());
                        }else if(toolName.endsWith(".mp3")){
                                return new
Image(images.mp3().getUrl());
                        }else if(toolName.endsWith(".mpeg")){
                                return new
Image(images.mpeg().getUrl());
                        }else if(toolName.endsWith(".pdf")){
                                return new
Image(images.pdf().getUrl());
                        }else if(toolName.endsWith(".png")){
                                return new
Image(images.png().getUrl());
                        }else
if(toolName.endsWith(".ppt")||toolName.endsWith(".pptx")){
                                return new
Image(images.ppt().getUrl());
                        }else if(toolName.endsWith(".rar")){
                                return new
Image(images.rar().getUrl());
                        }else if(toolName.endsWith(".tar.gz")){
                                return new
Image(images.tar().getUrl());
                        }else if(toolName.endsWith(".txt")){
                                return new
Image(images.txt().getUrl());
                        }else if(toolName.endsWith(".wav")){
                                return new
Image(images.wav().getUrl());
                        }else if(toolName.endsWith(".wma")){
                                return new
Image(images.wma().getUrl());
                        }else
if(toolName.endsWith(".xls")||toolName.endsWith(".xlsx")){
                                return new
Image(images.xls().getUrl());
                        }else if(toolName.endsWith(".zip")){
                                return new
Image(images.zip().getUrl());
                        }else{
                                return new
Image(images.blank().getUrl());
                        }
        }

        public void setupTracker(String currLoc, List<String> names){
                        FlexTable tracker = new FlexTable();
                        mainFlexTable.setWidget(0, 0, tracker);
```

```
                        tracker.setWidget(0, 0, new Hyperlink("Tools",
"tools/"));
                        List<String> parse =
Arrays.asList(currLoc.split("/"));
                        for(int i = 0; i < parse.size(); i++){
                                String prevPath ="";
                                for(int j = 0; j <= i; j++){
                                        prevPath +=
parse.get(j)+"/";
                                }
                                if(parse.get(i).length()!=0){
                                        tracker.setHTML(0,
(i*2)+1, "&raquo;");
                                        tracker.setWidget(0,
(i*2)+2, new Hyperlink(names.get(i), "tools/"+prevPath));
                                }
                        }
                }

                private void clearRows(){
                        int tableCount =toolsListTable.getRowCount();
                        for(int row = tableCount-1; row>=2; row--){
                                toolsListTable.removeRow(row);
                        }
                }

                public String getSearchTxt(){
                        return search.getText();
                }

                public void clearSearchTxt(){
                        search.setText("");
                }

                public HasClickHandlers getAddFolderButton(){
                        return addFolderButton;
                }

                public HasClickHandlers getAddFileButton(){
                        return addFileButton;
                }

                public HasClickHandlers getAddLinkButton(){
                        return addLinkButton;
                }

                public HasClickHandlers getSearchButton(){
                        return searchButton;
                }

                public HasClickHandlers getCancelSearchButton(){
                        return cancelSearchButton;
                }

                public HasKeyDownHandlers getSearchKeyHandler(){
                        return search;
                }

                public FlexTable getToolsList(){
                        return toolsListTable;
                }

                public void setButtonsVisibility(Boolean isVisible){
                        buttonTable.setVisible(isVisible);
                }

                public Widget asWidget(){
                        return this;
                }

}
```

BCrypt.java

```
package cepir.server;

// Copyright (c) 2006 Damien Miller <djm@mindrot.org>
//
// Permission to use, copy, modify, and distribute this software for any
// purpose with or without fee is hereby granted, provided that the above
// copyright notice and this permission notice appear in all copies.
//
// THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR
DISCLAIMS ALL WARRANTIES
// WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF
// MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
AUTHOR BE LIABLE FOR
// ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES
// WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
PROFITS, WHETHER IN AN
// ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF
// OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
SOFTWARE.

import java.io.UnsupportedEncodingException;

import java.security.SecureRandom;

/**
 * BCrypt implements OpenBSD-style Blowfish password hashing using
 * the scheme described in "A Future-Adaptable Password Scheme" by
 * Niels Provos and David Mazieres.
 * <p>
 * This password hashing system tries to thwart off-line password
 * cracking using a computationally-intensive hashing algorithm,
 * based on Bruce Schneier's Blowfish cipher. The work factor of
 * the algorithm is parameterised, so it can be increased as
 * computers get faster.
 * <p>
 * Usage is really simple. To hash a password for the first time,
 * call the hashpw method with a random salt, like this:
 * <p>
 * <code>
 * String pw_hash = BCrypt.hashpw(plain_password, BCrypt.gensalt()); <br
/>
 * </code>
 * <p>
 * To check whether a plaintext password matches one that has been
 * hashed previously, use the checkpw method:
 * <p>
 * <code>
 * if (BCrypt.checkpw(candidate_password, stored_hash))<br />
 *     System.out.println("It matches");<br />
 * else<br />
 *     System.out.println("It does not match");<br
/>
 * </code>
 * <p>
 * The gensalt() method takes an optional parameter (log_rounds)
 * that determines the computational complexity of the hashing:
 * <p>
 * <code>
 * String strong_salt = BCrypt.gensalt(10)<br />
 * String stronger_salt = BCrypt.gensalt(12)<br />
 * </code>
 * <p>
 * The amount of work increases exponentially (2**log_rounds), so
 * each increment is twice as much work. The default log_rounds is
 * 10, and the valid range is 4 to 31.
 *
 * @author Damien Miller
 * @version 0.2
 */
public class BCrypt {
        // BCrypt parameters
        private static final int
GENSALT_DEFAULT_LOG2_ROUNDS = 10;
        private static final int BCRYPT_SALT_LEN = 16;

        // Blowfish parameters
        private static final int BLOWFISH_NUM_ROUNDS = 16;

        // Initial contents of key schedule
        private static final int P_orig[] = {
                0x243f6a88, 0x85a308d3, 0x13198a2e,
0x03707344,
                0xa4093822, 0x299f31d0, 0x082efa98,
0xec4e6c89,
```

```
                0x452821e6, 0x38d01377, 0xbe5466cf,
0x34e90c6c,
                0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5,
0xb5470917,
                0x9216d5d9, 0x8979fb1b
        };
        private static final int S_orig[] = {
                0xd1310ba6, 0x98dfb5ac, 0x2ffd72db,
0xd01adfb7,
                0xb8e1afed, 0x6a267e96, 0xba7c9045,
0xf12c7f99,
                0x24a19947, 0xb3916cf7, 0x0801f2e2,
0x858efc16,
                0x636920d8, 0x71574e69, 0xa458fea3,
0xf4933d7e,
                0x0d95748f, 0x728eb658, 0x718bcd58,
0x82154aee,
                0x7b54a41d, 0xc25a59b5, 0x9c30d539,
0x2af26013,
                0xc5d1b023, 0x286085f0, 0xca417918,
0xb8db38ef,
                0x8e79dcb0, 0x603a180e, 0x6c9e0e8b,
0xb01e8a3e,
                0xd71577c1, 0xbd314b27, 0x78af2fda,
0x55605c60,
                0xe65525f3, 0xaa55ab94, 0x57489862,
0x63e81440,
                0x55ca396a, 0x2aab10b6, 0xb4cc5c34,
0x1141e8ce,
                0xa15486af, 0x7c72e993, 0xb3ee1411,
0x636fbc2a,
                0x2ba9c55d, 0x741831f6, 0xce5c3e16,
0x9b87931e,
                0xafd6ba33, 0x6c24cf5c, 0x7a325381,
0x28958677,
                0x3b8f4898, 0x6b4bb9af, 0xc4bfe81b,
0x66282193,
                0x61d809cc, 0xfb21a991, 0x487cac60,
0x5dec8032,
                0xef845d5d, 0xe98575b1, 0xdc262302,
0xeb651b88,
                0x23893e81, 0xd396acc5, 0x0f6d6ff3,
0x83f44239,
                0x2e0b4482, 0xa4842004, 0x69c8f04a,
0x9e1f9b5e,
                0x21c66842, 0xf6e96c9a, 0x670c9c61,
0xabd388f0,
                0x6a51a0d2, 0xd8542f68, 0x960fa728,
0xab5133a3,
                0x6eef0b6c, 0x137a3be4, 0xba3bf050,
0x7efb2a98,
                0xa1f1651d, 0x39af0176, 0x66ca593e,
0x82430e88,
                0x8cee8619, 0x456f9fb4, 0x7d84a5c3,
0x3b8b5ebe,
                0xe06f75d8, 0x85c12073, 0x401a449f,
0x56c16aa6,
                0x4ed3aa62, 0x363f7706, 0x1bfedf72,
0x429b023d,
                0x37d0d724, 0xd00a1248, 0xdb0fead3,
0x49f1c09b,
                0x075372c9, 0x80991b7b, 0x25d479d8,
0xf6e8def7,
                0xe3fe501a, 0xb6794c3b, 0x976ce0bd,
0x04c006ba,
                0xc1a94fb6, 0x409f60c4, 0x5e5c9ec2,
0x196a2463,
                0x68fb6faf, 0x3e6c53b5, 0x1339b2eb,
0x3b52ec6f,
                0x6dfc511f, 0x9b30952c, 0xcc814544,
0xaf5ebd09,
                0xbee3d004, 0xde334afd, 0x660f2807,
0x192e4bb3,
                0xc0cba857, 0x45c8740f, 0xd20b5f39,
0xb9d3fbdb,
                0x5579c0bd, 0x1a60320a, 0xd6a100c6,
0x402c7279,
                0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8, 0xdb3222f8,
                0x3c7516df, 0xfd616b15, 0x2f501ec8,
0xad0552ab,
                0x323db5fa, 0xfd238760, 0x53317b48,
0x3e00df82,
                0x9e5c57bb, 0xca6f8ca0, 0x1a87562e,
0xdf1769db,
                0xd542a8f6, 0x287effc3, 0xac6732c6,
0x8c4f5573,
                0x695b27b0, 0xbbca58c8, 0xe1ffa35d,
0xb8f011a0,
                0x10fa3d98, 0xfd2183b8, 0x4afcb56c,
0x2dd1d35b,
                0x9a53e479, 0xb6f84565, 0xd28e49bc,
0x4bfb9790,
                0xe1ddf2da, 0xa4cb7e33, 0x62fb1341,
0xcee4c6e8,
                0xef20cada, 0x36774c01, 0xd07e9efe,
0x2bf11fb4,
                0x95dbda4d, 0xae909198, 0xeaad8e71,
0x6b93d5a0,
                0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f,
0x8e7594b7,
                0x8ff6e2fb, 0xf2122b64, 0x8888b812,
0x900df01c,
                0x4fad5ea0, 0x688fc31c, 0xd1cff191,
0xb3a8c1ad,
                0x2f2f2218, 0xbe0e1777, 0xea752dfe,
0x8b021fa1,
                0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6,
0xce89e299,
                0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81,
0xd2ada8d9,
                0x165fa266, 0x80957705, 0x93cc7314,
0x211a1477,
                0xe6ad2065, 0x77b5fa86, 0xc75442f5,
0xfb9d35cf,
                0xebcdaf0c, 0x7b3e89a0, 0xd6411bd3,
0xae1e7e49,
                0x00250e2d, 0x2071b35e, 0x226800bb,
0x57b8e0af,
                0x2464369b, 0xf009b91e, 0x5563911d,
0x59dfa6aa,
                0x78c14389, 0xd95a537f, 0x207d5ba2,
0x02e5b9c5,
                0x83260376, 0x6295cfa9, 0x11c81968,
0x4e734a41,
                0xb3472dca, 0x7b14a94a, 0x1b510052,
0x9a532915,
                0xd60f573f, 0xbc9bc6e4, 0x2b60a476,
0x81e67400,
                0x08ba6fb5, 0x571be91f, 0xf296ec6b,
0x2a0dd915,
                0xb6636521, 0xe7b9f9b6, 0xff34052e,
0xc5855664,
                0x53b02d5d, 0xa99f8fa1, 0x08ba4799,
0x6e85076a,
                0x4b7a70e9, 0xb5b32944, 0xdb75092e,
0xc4192623,
                0xad6ea6b0, 0x49a7df7d, 0x9cee60b8,
0x8fedb266,
                0xecaa8c71, 0x699a17ff, 0x5664526c,
0xc2b19ee1,
                0x193602a5, 0x75094c29, 0xa0591340,
0xe4183a3e,
                0x3f54989a, 0x5b429d65, 0x6b8fe4d6,
0x99f73fd6,
                0xa1d29c07, 0xefe830f5, 0x4d2d38e6,
0xf0255dc1,
                0x4cdd2086, 0x8470eb26, 0x6382e9c6,
0x021ecc5e,
                0x09686b3f, 0x3ebaefc9, 0x3c971814,
0x6b6a70a1,
                0x687f3584, 0x52a0e286, 0xb79c5305,
0xaa500737,
                0x3e07841c, 0x7fdeae5c, 0x8e7d44ec,
0x5716f2b8,
                0xb03ada37, 0xf0500c0d, 0xf01c1f04,
0x0200b3ff,
                0xae0cf51a, 0x3cb574b2, 0x25837a58,
0xdc0921bd,
                0xd19113f9, 0x7ca92ff6, 0x94324773,
0x22f54701,
```

0x9af3dda7,
0xa4751e41,
0x183eb331,
0xf60a04bf,
0xbcaf89af,
0xdccf3f2e,
0x9f84cd87,
0xe94b7d8c,
0xc464c3d2,
0x24c2ba16,
0x133ae4dd,
0x5f11199b,
0x5924a509,

0x5a3e2ab3,
0x99e71d0f,
0x9c10b36a,
0x1e0a2df4,
0x19c27960,
0xeac31f66,
0x018cff28,
0x68ab9802,
0x5b6e2f84,
0x619f1510,
0xaa0363cf,
0xcbaade14,
0xb2f3846e,
0x655abb50,
0xc021b8f7,
0x623d7da8,
0x16681281,
0x7858ba99,
0x1ac24696,
0x6dbc3128,

0x203e13e0,
0xfacb4fd0,
0x41cd2105,
0x3d816250,
0xc1c7b6a3,
0x5692b285,
0x3ae5e581, 0x37c2dadc, 0xc8b57634,
0xa9446146, 0x0fd0030e, 0xecc8c73e,
0xe238cd99, 0x3bea0e2f, 0x3280bba1,
0x4e548b38, 0x4f6db908, 0x6f420d03,
0x2cb81290, 0x24977c79, 0x5679b072,
0xde9a771f, 0xd9930810, 0xb38bae12,
0x5512721f, 0x2e6b7124, 0x501adde6,
0x7a584718, 0x7408da17, 0xbc9f9abc,
0xec7aec3a, 0xdb851dfa, 0x63094366,
0xef1c1847, 0x3215d908, 0xdd433b37,
0x12a14d43, 0x2a65c451, 0x50940002,
0x71dff89e, 0x10314e55, 0x81ac77d6,
0x043556f1, 0xd7a3c76b, 0x3c11183b,
0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,
0x86e34570, 0xeae96fb1, 0x860e5e0a,
0x771fe71c, 0x4e3d06fa, 0x2965dcb9,
0x803e89d6, 0x5266c825, 0x2e4cc978,
0xc6150eba, 0x94e2ea78, 0xa5fc3c53,
0xf2f74ea7, 0x361d2b3d, 0x1939260f,
0x5223a708, 0xf71312b6, 0xebadfe6e,
0xe3bc4595, 0xa67bc883, 0xb17f37d1,
0xc332ddef, 0xbe6c5aa5, 0x65582185,
0xeecea50f, 0xdb2f953b, 0x2aef7dad,
0x1521b628, 0x29076170, 0xecdd4775,
0x13cca830, 0xeb61bd96, 0x0334fe1e,
0xb5735c90, 0x4c70a239, 0xd59e9e0b,
0xeecc86bc, 0x60622ca7, 0x9cab5cab,
0x648b1eaf, 0x19bdf0ca, 0xa02369b9,
0x40685a32, 0x3c2ab4b3, 0x319ee9d5,
0x9b540b19, 0x875fa099, 0x95f7997e,
0xf837889a, 0x97e32d77, 0x11ed935f,
0x0e358829, 0xc7e61fd6, 0x96dedfa1,
0x57f584a5, 0x1b227263, 0x9b83c3ff,
0xcdb30aeb, 0x532e3054, 0x8fd948e4,
0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73,
0x5d4a14d9, 0xe864b7e3, 0x42105d14,
0x45eee2b6, 0xa3aaabea, 0xdb6c4f15,
0xc742f442, 0xef6abbb5, 0x654f3b1d,
0xd81e799e, 0x86854dc7, 0xe44b476a,
0xcf62a1f2, 0x5b8d2646, 0xfc8883a0,
0x7f1524c3, 0x69cb7492, 0x47848a0b,
0x23820e00,
0x233f7061,
0x6c223bdb,
0xce77326e,
0x61d99735,
0x800bcadc,
0x0e1e9ec9,
0xe3674340,
0xf16dff20,
0xdb83adf7,
0x94692934,
0xd4a20068,
0x500061af,
0xbf8b8840,
0x66a02f45,
0x31cb8504,
0xabca0a9a,
0xe9b66dfb,
0x27a18dee,
0x7af4d6b6,
0x406b2a42,
0x3b124e8b,
0xeae397b2,
0xca7820fb,
0xba489527,
0xd096954b,
0x99e1db33,
0x9029317c,
0x05282ce3,
0xc70f86dc,
0x5d886e17,
0x41113564,
0x1f636c1b,
0xcad18115,
0xeebeb922,
0x2da2f728,
0xe7ccf5f0,
0xf33e8d1e,

0xdb6e6b0d,
0x095bbf00, 0xad19489d, 0x1462b174,
0x58428d2a, 0x0c55f5ea, 0x1dadf43e,
0x3372f092, 0x8d937e41, 0xd65fecf1,
0x7cde3759, 0xcbee7460, 0x4085f2a7,
0xa6078084, 0x19f8509e, 0xe8efd855,
0xa969a7aa, 0xc50c06c2, 0x5a04abfc,
0x9e447a2e, 0xc3453484, 0xfdd56705,
0xdb73dbd3, 0x105588cd, 0x675fda79,
0xc5c43465, 0x713e38d8, 0x3d28f89e,
0x153e21e7, 0x8fb03d4a, 0xe6e39f2b,
0xe93d5a68, 0x948140f7, 0xf64c261c,
0x411520f7, 0x7602d4f7, 0xbcf46b2e,
0xd4082471, 0x3320f46a, 0x43b7d4b7,
0x1e39f62e, 0x97244546, 0x14214f74,
0x4d95fc1d, 0x96b591af, 0x70f4ddd3,
0xbfbc09ec, 0x03bd9785, 0x7fac6dd0,
0x96eb27b3, 0x55fd3941, 0xda2547e6,
0x28507825, 0x530429f4, 0x0a2c86da,
0x68dc1462, 0xd7486900, 0x680ec0a4,
0x4f3ffea2, 0xe887ad8c, 0xb58ce006,
0xaace1e7c, 0xd3375fec, 0xce78a399,
0x20fe9e35, 0xd9f385b9, 0xee39d7ab,
0x1dc9faf7, 0x4b6d1856, 0x26a36631,
0x3a6efa74, 0xdd5b4332, 0x6841e7f7,
0xfb0af54e, 0xd8feb397, 0x454056ac,
0x55533a3a, 0x20838d87, 0xfe6ba9b7,
0x55a867bc, 0xa1159a58, 0xcca92963,
0xa62a4a56, 0x3f3125f9, 0x5ef47e1c,
0xfdf8e802, 0x04272f70, 0x80bb155c,
0x95c11548, 0xe4c66d22, 0x48c1133f,
0x07f9c9ee, 0x41041f0f, 0x404779a4,
0x325f51eb, 0xd59bc0d1, 0xf2bcc18f,
0x257b7834, 0x602a9c60, 0xdff8e8a3,
0x0e12b4c2, 0x02e1329e, 0xaf664fd1,
0x6b2395e0, 0x333e92e1, 0x3b240b62,
0x85b2a20e, 0xe6ba0d99, 0xde720c8c,
0xd0127845, 0x95b794fd, 0x647d0862,
0x5449a36f, 0x877d48fa, 0xc39dfd27,
0x0a476341, 0x992eff74, 0x3a6f6eab, 0xf4f8fd37,
0xa812dc60, 0xa1ebddf8, 0x991be14c,

0xdcd0e804,
0x690fed0b,
0xd9155ea3,
0x763bd6eb,
0xf42e312d,
0x782ef11c,
0x4bfb6350,
0xe2e1c3c9,
0xd5abea2a,
0x64e4c3fe,
0x6003604d,
0xd736fccc,
0x3c005e5f,
0xbf582e61,

0x46fcd9b9,
0x915f95e2,
0xc902de4c,
0x7574a99e,
0xc4324633,
0x1d6efe10,
0x2868f169,
0x4fcd7f52,
0x0de6d027,
0x61a806b5,
0x30dc7d62,
0xc2c21634,
0xce591d76,
0x7c927c24,
0xd39eb8fc,
0x4dad0fc4,
0x6c51133c,
0xddc6c837,
0x406000e0,
0x5ac52d1b,
0x99bc9bbe,
0xc700c47b,
0x6a366eb4,
0x6549c2c8,
0x4cd04dc6,
0xc67b5510, 0x6d672c37, 0x2765d43b,
0xf1290dc7, 0xcc00ffa3, 0xb5390f92,
0x667b9ffb, 0xcedb7d9c, 0xa091cf0b,
0xbb132f88, 0x515bad24, 0x7b9479bf,
0x37392eb3, 0xcc115979, 0x8026e297,
0x6842ada7, 0xc66a2b3b, 0x12754ccc,
0x6a124237, 0xb79251e7, 0x06a1bbe6,
0x1a6b1018, 0x11caedfa, 0x3d25bdd8,
0x44421659, 0x0a121386, 0xd90cec6e,
0x64af674e, 0xda86a85f, 0xbebfe988,
0x9dbc8057, 0xf0f7c086, 0x60787bf8,
0xd1fd8346, 0xf6381fb0, 0x7745ae04,
0x83426b33, 0xf01eab71, 0xb0804187,
0x77a057be, 0xbde8ae24, 0x55464299,
0x4e58f48f, 0xf2ddfda2, 0xf474ef38, 0x8789bdc2,
0x5366f9c3, 0xc8b38e74, 0xb475f255,
0x7aeb2661, 0x8b1ddf84, 0x846a0e79,
0x466e598e, 0x20b45770, 0x8cd55591,
0xb90bace1, 0xbb8205d0, 0x11a86248,
0xb77f19b6, 0xe0a9dc09, 0x662d09a1,
0xe85a1f02, 0x09f0be8c, 0x4a99a025,
0x1ab93d1d, 0x0ba5a4df, 0xa186f20f,
0xdcb7da83, 0x573906fe, 0xa1e2ce9b,
0x50115e01, 0xa70683fa, 0xa002b5c4,
0x9af88c27, 0x773f8641, 0xc3604c06,
0xf0177a28, 0xc0f586e0, 0x006058aa,
0x11e69ed7, 0x2338ea63, 0x53c2dd94,
0xbbcbee56, 0x90bcb6de, 0xebfc7da1,
0x6f05e409, 0x4b7c0188, 0x39720a3d,
0x86e3725f, 0x724d9db9, 0x1ac15bb4,
0xed545578, 0x08fca5b5, 0xd83d7cd3,
0x1e50ef5e, 0xb161e6f8, 0xa28514d9,
0x6fd5c7e7, 0x56e14ec4, 0x362abfce,
0xd79a3234, 0x92638212, 0x670efa8e,
0x3a39ce37, 0xd3faf5cf, 0xabc27737,
0x5cb0679e, 0x4fa33742, 0xd3822740,
0xd5118e9d, 0xbf0f7315, 0xd62d1c7e,
0xb78c1b6b, 0x21a19045, 0xb26eb1be,
0x5748ab2f, 0xbc946e79, 0xc6a376d2,
0x530ff8ee, 0x468dde7d, 0xd5730a1d,
0xbe5ee304,
0x9a86ee22,
0x9cf2d0a4,
0xba645bd6,
0xef5562e9,

0x3b3ee593,
0x022b8b51,
0x7c7d2d28,
0x5a88f54c,
0xed93fa9b,
0x79132e28,
0xe3d35e8c,
0x0564f0bd,
0xa93a072a,
0x26dcf319,
0x8aba3cbb,
0xccad925f,
0x9320f991,
0x774fbe32,
0x6413e680,
0x09072166,
0x1c20c8ae,
0x6bb4e3bb,
0xbcb4cdd5,
0xbf3c6f47,
0xf64e6370,
0xaf537d5d,
0x0115af84,
0xce6ea048,
0x277227f8,
0x344525bd,
0xa01fbac9,
0xa1e8aac7,
0xd50ada38,
0xe0b12b4f,
0x27d9459c,
0x9b941525,
0x12baa8d1,
0xcb03a442,
0x2939bbdb, 0xa9ba4650, 0xac9526e8,
0xa1fad5f0, 0x6a2d519a, 0x63ef8ce2,
0xc089c2b8, 0x43242ef6, 0xa51e03aa,
0x83c061ba, 0x9be96a4d, 0x8fe51550,
0x2826a2f9, 0xa73a3ae1, 0x4ba99586,
0xc72fefd3, 0xf752f7da, 0x3f046f69, 0x77fa0a59,
0x80e4a915, 0x87b08601, 0x9b09e6ad,
0xe990fd5a, 0x9e34d797, 0x2cf0b7d9,
0x96d5ac3a, 0x017da67d, 0xd1cf3ed6,
0x1f9f25cf, 0xadf2b89b, 0x5ad6b472,
0xe029ac71, 0xe019a5e6, 0x47b0acfd,
0xe8d3c48d, 0x283b57cc, 0xf8d56629,
0x785f0191, 0xed756055, 0xf7960e44,
0x15056dd4, 0x88f46dba, 0x03a16125,
0xc3eb9e15, 0x3c9057a2, 0x97271aec,
0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb,
0x7533d928, 0xb155fdf5, 0x03563482,
0x28517711, 0xc20ad9f8, 0xabcc5167,
0x4de81751, 0x3830dc8e, 0x379d5862,
0xea7a90c2, 0xfb3e7bce, 0x5121ce64,
0xa8b6e37e, 0xc3293d46, 0x48de5369,
0xa2ae0810, 0xdd6db224, 0x69852dfd,
0xb39a460a, 0x6445c0dd, 0x586cdecf,
0x5bbef7dd, 0x1b588d40, 0xccd2017f,
0xdda26a7e, 0x3a59ff45, 0x3e350a44,
0x72eacea8, 0xfa6484bb, 0x8d6612ae,
0xd29be463, 0x542f5d9e, 0xaec2771b,
0x740e0d8d, 0xe75b1357, 0xf8721671,
0x4040cb08, 0x4eb4e2cc, 0x34d2466a,
0xe1b00428, 0x95983a1d, 0x06b89fb4,
0x6f3f3b82, 0x3520ab82, 0x011a1d4b,
0x611560b1, 0xe7933fdc, 0xbb3a792b,
0xa08839e1, 0x51ce794b, 0x2f32c9b7,
0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3,
0x1a908749, 0xd44fbd9a, 0xd0dadecb,
0x0339c32a, 0xc6913667, 0x8df9317c,
0xf79e59b7, 0x43f5bb3a, 0xf2d519ff,
0xbf97222c, 0x15e6fc2a, 0x0f91fc71,
0xfae59361, 0xceb69ceb, 0xc2a86459,
0xb6c1075e, 0xe3056a0c, 0x10d25065,

```
                0xe0ec6e0e, 0x1698db3b, 0x4c98a0be,
0x3278e964,
                0x9f1f9532, 0xe0d392df, 0xd3a0342b,
0x8971f21e,
                0x1b0a7441, 0x4ba3348c, 0xc5be7120,
0xc37632d8,
                0xdf359f8d, 0x9b992f2e, 0xe60b6f47,
0x0fe3f11d,
                0xe54cda54, 0x1edad891, 0xce6279cf,
0xcd3e7e6f,
                0x1618b166, 0xfd2c1d05, 0x848fd2c5,
0xf6fb2299,
                0xf523f357, 0xa6327623, 0x93a83531,
0x56cccd02,
                0xacf08162, 0x5a75ebb5, 0x6e163697,
0x88d273cc,
                0xde966292, 0x81b949d0, 0x4c50901b,
0x71c65614,
                0xe6c6c7bd, 0x327a140a, 0x45e1d006,
0xc3f27b9a,
                0xc9aa53fd, 0x62a80f00, 0xbb25bfe2,
0x35bdd2f6,
                0x71126905, 0xb2040222, 0xb6cbcf7c,
0xcd769c2b,
                0x53113ec0, 0x1640e3d3, 0x38abbd60,
0x2547adf0,
                0xba38209c, 0xf746ce76, 0x77afa1c5,
0x20756060,
                0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0,
0x4cf9aa7e,
                0x1948c25c, 0x02fb8a8c, 0x01c36ae4,
0xd6ebe1f9,
                0x90d4f869, 0xa65cdea0, 0x3f09252d,
0xc208e69f,
                0xb74e6132, 0xce77e25b, 0x578fdfe3,
0x3ac372e6
        };

        // bcrypt IV: "OrpheanBeholderScryDoubt"
        static private final int bf_crypt_ciphertext[] = {
                0x4f727068, 0x65616e42, 0x65686f6c,
                0x64657253, 0x63727944, 0x6f756274
        };

        // Table for Base64 encoding
        static private final char base64_code[] = {
                '.', '/', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
                'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
                'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
                'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
                'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5',
                '6', '7', '8', '9'
        };

        // Table for Base64 decoding
        static private final byte index_64[] = {
                -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
                -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
                -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
                -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
                -1, -1, -1, -1, -1, -1, 0, 1, 54, 55,
                56, 57, 58, 59, 60, 61, 62, 63, -1, -1,
                -1, -1, -1, -1, -1, 2, 3, 4, 5, 6,
                7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
                -1, -1, -1, -1, -1, -1, -1, 28, 29, 30,
                31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
                41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
                51, 52, 53, -1, -1, -1, -1, -1
        };

        // Expanded Blowfish key
        private int P[];
        private int S[];

        /**
         * Encode a byte array using bcrypt's slightly-modified base64
         * encoding scheme. Note that this is *not* compatible with
         * the standard MIME-base64 encoding.
         *
         * @param d             the byte array to encode
         * @param len           the number of bytes to encode
         * @return  base64-encoded string
         * @exception IllegalArgumentException if the length is
invalid
         */
        private static String encode_base64(byte d[], int len)
                throws IllegalArgumentException {
                int off = 0;
                StringBuffer rs = new StringBuffer();
                int c1, c2;

                if (len <= 0 || len > d.length)
                        throw new IllegalArgumentException
("Invalid len");

                while (off < len) {
                        c1 = d[off++] & 0xff;
                        rs.append(base64_code[(c1 >> 2) &
0x3f]);
                        c1 = (c1 & 0x03) << 4;
                        if (off >= len) {
                                rs.append(base64_code[c1 & 0x3f]);
                                break;
                        }
                        c2 = d[off++] & 0xff;
                        c1 |= (c2 >> 4) & 0x0f;
                        rs.append(base64_code[c1 & 0x3f]);
                        c1 = (c2 & 0x0f) << 2;
                        if (off >= len) {
                                rs.append(base64_code[c1 & 0x3f]);
                                break;
                        }
                        c2 = d[off++] & 0xff;
                        c1 |= (c2 >> 6) & 0x03;
                        rs.append(base64_code[c1 & 0x3f]);
                        rs.append(base64_code[c2 & 0x3f]);
                }
                return rs.toString();
        }

        /**
         * Look up the 3 bits base64-encoded by the specified character,
         * range-checking againt conversion table
         * @param x             the base64-encoded value
         * @return  the decoded value of x
         */
        private static byte char64(char x) {
                if ((int)x < 0 || (int)x > index_64.length)
                        return -1;
                return index_64[(int)x];
        }

        /**
         * Decode a string encoded using bcrypt's base64 scheme to a
         * byte array. Note that this is *not* compatible with
         * the standard MIME-base64 encoding.
         * @param s the string to decode
         * @param maxolen       the maximum number of bytes to
decode
         * @return  an array containing the decoded bytes
         * @throws IllegalArgumentException if maxolen is invalid
         */
        private static byte[] decode_base64(String s, int maxolen)
                throws IllegalArgumentException {
                StringBuffer rs = new StringBuffer();
                int off = 0, slen = s.length(), olen = 0;
                byte ret[];
                byte c1, c2, c3, c4, o;

                if (maxolen <= 0)
                        throw new IllegalArgumentException
("Invalid maxolen");

                while (off < slen - 1 && olen < maxolen) {
                        c1 = char64(s.charAt(off++));
                        c2 = char64(s.charAt(off++));
                        if (c1 == -1 || c2 == -1)
```

```
                                        break;
                                o = (byte)(c1 << 2);
                                o |= (c2 & 0x30) >> 4;
                                rs.append((char)o);
                                if (++olen >= maxolen || off >= slen)
                                        break;
                                c3 = char64(s.charAt(off++));
                                if (c3 == -1)
                                        break;
                                o = (byte)((c2 & 0x0f) << 4);
                                o |= (c3 & 0x3c) >> 2;
                                rs.append((char)o);
                                if (++olen >= maxolen || off >= slen)
                                        break;
                                c4 = char64(s.charAt(off++));
                                o = (byte)((c3 & 0x03) << 6);
                                o |= c4;
                                rs.append((char)o);
                                ++olen;
                        }

                        ret = new byte[olen];
                        for (off = 0; off < olen; off++)
                                ret[off] = (byte)rs.charAt(off);
                        return ret;
                }

                /**
                 * Blowfish encipher a single 64-bit block encoded as
                 * two 32-bit halves
                 * @param lr                an array containing the two 32-bit half
blocks
                 * @param off               the position in the array of the blocks
                 */
                private final void encipher(int lr[], int off) {
                        int i, n, l = lr[off], r = lr[off + 1];

                        l ^= P[0];
                        for (i = 0; i <= BLOWFISH_NUM_ROUNDS - 2;)
{
                                // Feistel substitution on left word
                                n = S[(l >> 24) & 0xff];
                                n += S[0x100 | ((l >> 16) & 0xff)];
                                n ^= S[0x200 | ((l >> 8) & 0xff)];
                                n += S[0x300 | (l & 0xff)];
                                r ^= n ^ P[++i];

                                // Feistel substitution on right word
                                n = S[(r >> 24) & 0xff];
                                n += S[0x100 | ((r >> 16) & 0xff)];
                                n ^= S[0x200 | ((r >> 8) & 0xff)];
                                n += S[0x300 | (r & 0xff)];
                                l ^= n ^ P[++i];
                        }
                        lr[off] = r ^ P[BLOWFISH_NUM_ROUNDS + 1];
                        lr[off + 1] = l;
                }

                /**
                 * Cycically extract a word of key material
                 * @param data              the string to extract the data from
                 * @param offp              a "pointer" (as a one-entry array) to
the
                 * current offset into data
                 * @return  the next word of material from data
                 */
                private static int streamtoword(byte data[], int offp[]) {
                        int i;
                        int word = 0;
                        int off = offp[0];

                        for (i = 0; i < 4; i++) {
                                word = (word << 8) | (data[off] &
0xff);
                                off = (off + 1) % data.length;
                        }

                        offp[0] = off;
                        return word;
                }

                /**
                 * Initialise the Blowfish key schedule
                 */
                private void init_key() {
                        P = (int[])P_orig.clone();
                        S = (int[])S_orig.clone();
                }

                /**
                 * Key the Blowfish cipher
                 * @param key               an array containing the key
                 */
                private void key(byte key[]) {
                        int i;
                        int koffp[] = { 0 };
                        int lr[] = { 0, 0 };
                        int plen = P.length, slen = S.length;

                        for (i = 0; i < plen; i++)
                                P[i] = P[i] ^ streamtoword(key,
koffp);

                        for (i = 0; i < plen; i += 2) {
                                encipher(lr, 0);
                                P[i] = lr[0];
                                P[i + 1] = lr[1];
                        }

                        for (i = 0; i < slen; i += 2) {
                                encipher(lr, 0);
                                S[i] = lr[0];
                                S[i + 1] = lr[1];
                        }
                }

                /**
                 * Perform the "enhanced key schedule" step described by
                 * Provos and Mazieres in "A Future-Adaptable Password
Scheme"
                 * http://www.openbsd.org/papers/bcrypt-paper.ps
                 * @param data              salt information
                 * @param key               password information
                 */
                private void ekskey(byte data[], byte key[]) {
                        int i;
                        int koffp[] = { 0 }, doffp[] = { 0 };
                        int lr[] = { 0, 0 };
                        int plen = P.length, slen = S.length;

                        for (i = 0; i < plen; i++)
                                P[i] = P[i] ^ streamtoword(key,
koffp);

                        for (i = 0; i < plen; i += 2) {
                                lr[0] ^= streamtoword(data, doffp);
                                lr[1] ^= streamtoword(data, doffp);
                                encipher(lr, 0);
                                P[i] = lr[0];
                                P[i + 1] = lr[1];
                        }

                        for (i = 0; i < slen; i += 2) {
                                lr[0] ^= streamtoword(data, doffp);
                                lr[1] ^= streamtoword(data, doffp);
                                encipher(lr, 0);
                                S[i] = lr[0];
                                S[i + 1] = lr[1];
                        }
                }

                /**
                 * Perform the central password hashing step in the
                 * bcrypt scheme
                 * @param password      the password to hash
                 * @param salt              the binary salt to hash with the
password
                 * @param log_rounds    the binary logarithm of the number
                 * of rounds of hashing to apply
                 * @return  an array containing the binary hashed password
```

```
		 */
		private byte[] crypt_raw(byte password[], byte salt[], int
log_rounds) {
			int rounds, i, j;
			int cdata[] = (int[])bf_crypt_ciphertext.clone();
			int clen = cdata.length;
			byte ret[];

			if (log_rounds < 4 || log_rounds > 31)
				throw new IllegalArgumentException
("Bad number of rounds");
			rounds = 1 << log_rounds;
			if (salt.length != BCRYPT_SALT_LEN)
				throw new IllegalArgumentException
("Bad salt length");

			init_key();
			ekskey(salt, password);
			for (i = 0; i < rounds; i++) {
				key(password);
				key(salt);
			}

			for (i = 0; i < 64; i++) {
				for (j = 0; j < (clen >> 1); j++)
					encipher(cdata, j << 1);
			}

			ret = new byte[clen * 4];
			for (i = 0, j = 0; i < clen; i++) {
				ret[j++] = (byte)((cdata[i] >> 24) &
0xff);
				ret[j++] = (byte)((cdata[i] >> 16) &
0xff);
				ret[j++] = (byte)((cdata[i] >> 8) &
0xff);
				ret[j++] = (byte)(cdata[i] & 0xff);
			}
			return ret;
		}

		/**
		 * Hash a password using the OpenBSD bcrypt scheme
		 * @param password	the password to hash
		 * @param salt		the salt to hash with (perhaps
generated
		 * using BCrypt.gensalt)
		 * @return	the hashed password
		 */
		public static String hashpw(String password, String salt) {
			BCrypt B;
			String real_salt;
			byte passwordb[], saltb[], hashed[];
			char minor = (char)0;
			int rounds, off = 0;
			StringBuffer rs = new StringBuffer();

			if (salt.charAt(0) != '$' || salt.charAt(1) != '2')
				throw new IllegalArgumentException
("Invalid salt version");
			if (salt.charAt(2) == '$')
				off = 3;
			else {
				minor = salt.charAt(2);
				if (minor != 'a' || salt.charAt(3) != '$')
					throw new
IllegalArgumentException ("Invalid salt revision");
				off = 4;
			}

			// Extract number of rounds
			if (salt.charAt(off + 2) > '$')
				throw new IllegalArgumentException
("Missing salt rounds");
			rounds = Integer.parseInt(salt.substring(off, off +
2));

			real_salt = salt.substring(off + 3, off + 25);
			try {
				passwordb = (password + (minor >=
'a' ? "\000" : "")).getBytes("UTF-8");
			} catch (UnsupportedEncodingException uee) {
				throw new AssertionError("UTF-8 is
not supported");
			}

			saltb = decode_base64(real_salt,
BCRYPT_SALT_LEN);

			B = new BCrypt();
			hashed = B.crypt_raw(passwordb, saltb, rounds);

			rs.append("$2");
			if (minor >= 'a')
				rs.append(minor);
			rs.append("$");
			if (rounds < 10)
				rs.append("0");
			rs.append(Integer.toString(rounds));
			rs.append("$");
			rs.append(encode_base64(saltb, saltb.length));
			rs.append(encode_base64(hashed,
			    bf_crypt_ciphertext.length * 4 - 1));
			return rs.toString();
		}

		/**
		 * Generate a salt for use with the BCrypt.hashpw() method
		 * @param log_rounds	the log2 of the number of rounds of
		 * hashing to apply - the work factor therefore increases as
		 * 2**log_rounds.
		 * @param random		an instance of
SecureRandom to use
		 * @return	an encoded salt value
		 */
		public static String gensalt(int log_rounds, SecureRandom
random) {
			StringBuffer rs = new StringBuffer();
			byte rnd[] = new byte[BCRYPT_SALT_LEN];

			random.nextBytes(rnd);

			rs.append("$2a$");
			if (log_rounds < 10)
				rs.append("0");
			rs.append(Integer.toString(log_rounds));
			rs.append("$");
			rs.append(encode_base64(rnd, rnd.length));
			return rs.toString();
		}

		/**
		 * Generate a salt for use with the BCrypt.hashpw() method
		 * @param log_rounds	the log2 of the number of rounds of
		 * hashing to apply - the work factor therefore increases as
		 * 2**log_rounds.
		 * @return	an encoded salt value
		 */
		public static String gensalt(int log_rounds) {
			return gensalt(log_rounds, new SecureRandom());
		}

		/**
		 * Generate a salt for use with the BCrypt.hashpw() method,
		 * selecting a reasonable default for the number of hashing
		 * rounds to apply
		 * @return	an encoded salt value
		 */
		public static String gensalt() {
			return
gensalt(GENSALT_DEFAULT_LOG2_ROUNDS);
		}

		/**
		 * Check that a plaintext password matches a previously hashed
		 * one
		 * @param plaintext	the plaintext password to verify
		 * @param hashed	the previously-hashed password
		 * @return	true if the passwords match, false otherwise
```

```
		 */
		public static boolean checkpw(String plaintext, String hashed) {
			return (hashed.compareTo(hashpw(plaintext,
hashed)) == 0);
		}
}
```

DBConnection.java

```
package cepir.server;

import java.sql.*;


public class DBConnection{
	 public static Connection getConnection() {
	   Connection conn = null;
	   String url = "jdbc:mysql://localhost/";
//	   String dbName = "cepirDB";
//	   String driver = "com.mysql.jdbc.Driver";
//	   String userName = "root";
//	   String password = "";
	   String dbName = "EpiCollaboratory";
	   String driver = "com.mysql.jdbc.Driver";
	   String userName = "EpiCollaboratory";
	   String password = "uvmpHuLZCErbvm7S";
	   try {
			Class.forName(driver).newInstance();
			conn =
DriverManager.getConnection(url+dbName,userName,password);
			return conn;
	   } catch (Exception e) {
			e.printStackTrace();
			return null;
	   }
  }
}
```

DownloadFileService.java

```
package cepir.server;

import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.io.IOUtils;

import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
import com.mysql.jdbc.PreparedStatement;

@RemoteServiceRelativePath("download")
public class DownloadFileService extends HttpServlet{
	protected void doPost(HttpServletRequest request,
HttpServletResponse response)
	throws ServletException, IOException {
		processRequest(request, response);
	}

	protected void doGet(HttpServletRequest request,
HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }

	private void processRequest(HttpServletRequest request,
HttpServletResponse response) {
		Integer fileID =
Integer.valueOf(request.getParameter("f"));
		String table = request.getParameter("t"); //t:
tool,misc_file or reference
		try {
			Connection conn =
DBConnection.getConnection();
			PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT * FROM "+table+"
WHERE "+(table.equals("reference")?"ref":table)+"_id=?");
			ps.setInt(1, fileID);
			ResultSet rs = ps.executeQuery();
			rs.next();
			byte[] bbuf = new byte[1024];

			ServletOutputStream out =
response.getOutputStream();
			ServletContext context =
getServletConfig().getServletContext();

			File file = new
File(getServletContext().getRealPath("/")+"/uploads/"+(table.equals("refere
nce")?"ref":table)+"/"+(table.equals("tool")?"":rs.getString("proj_id")+"/")+

		rs.getString((table.equals("reference")?"ref":table)+"_path")+"/
"+rs.getString((table.equals("reference")?"ref":table)+"_name"));
			String mimetype =
context.getMimeType(rs.getString((table.equals("reference")?"ref":table)+"_
name"));

			response.setContentType((mimetype
!= null) ? mimetype : "application/octet-stream");
			response.setContentLength((int)
file.length());
			response.setHeader("Content-
Disposition", "attachement; filename=\"" +
rs.getString((table.equals("reference")?"ref":table)+"_name") + "\"");

			DataInputStream in = new
DataInputStream(new FileInputStream(file));

			int length;
			while ((in != null) && ((length =
in.read(bbuf)) != -1)) {
			out.write(bbuf, 0, length);
			}

			in.close();
			out.flush();
			out.close();
		} catch (SQLException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

ExcelServiceImpl.java

```
package cepir.server;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Date;
import java.util.Locale;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.hssf.util.HSSFColor;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.IndexedColors;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;

import jxl.Workbook;
```

```
import jxl.WorkbookSettings;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import jxl.write.WriteException;
import cepir.client.ExcelService;

public class ExcelServiceImpl extends SessionMngt implements
ExcelService{

	public void createExcel(Integer formID, String loc){
		try {
			File path = new File(loc);
			if (!path.exists()) {
				boolean status =
path.mkdirs();
			}
			String filename = loc + formID
+".xls";
			WorkbookSettings ws = new
WorkbookSettings();
			ws.setLocale(new Locale("en",
"EN"));
			WritableWorkbook workbook =
Workbook.createWorkbook(new File(filename), ws);
			workbook.createSheet("Sheet1", 0);
			workbook.write();
			workbook.close();
		} catch (IOException e) {
			e.printStackTrace();
		} catch (WriteException e) {
			e.printStackTrace();
		}
	}

	public void writeExcel(Integer formID, String loc) throws
IOException{
		try {
			File path = new File(loc);
			if (!path.exists()) {
				boolean status =
path.mkdirs();
			}
			FileOutputStream fileOut = new
FileOutputStream(loc + formID +".xls");
			HSSFWorkbook workbook = new
HSSFWorkbook();
			workbook.createSheet("Sheet1");
			workbook.write(fileOut);
			fileOut.flush();
			fileOut.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
	}
}
```

FormServiceImpl.java

```
package cepir.server;

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import com.mysql.jdbc.PreparedStatement;

import cepir.client.FormService;
import cepir.shared.DataEntryForm;
import cepir.shared.Project;
import cepir.shared.User;

public class FormServiceImpl extends SessionMngt implements
FormService{

	public ArrayList<DataEntryForm> getFormsSearchList(Integer
projID, String query){
		ArrayList<DataEntryForm> forms = new
ArrayList<DataEntryForm>();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from data_entry_form
where proj_id=? and form_name like ? order by form_name asc");
			ps.setInt(1, projID);
			ps.setString(2, "%"+query+"%");
			ResultSet rset = ps.executeQuery();
			while(rset.next()){
				DataEntryForm form =
new DataEntryForm();
				form.formID =
Integer.valueOf(rset.getString("form_id"));
				form.formName =
rset.getString("form_name");
				form.formDesc =
rset.getString("form_desc");
				form.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				form.dateUpdated =
java.sql.Timestamp.valueOf(rset.getString("date_updated"));
				User user = new User();
				user.username =
rset.getString("creator");
				form.creator = user;
				Project proj = new
Project();
				proj.projID =
Integer.valueOf(rset.getString("proj_id"));
				form.projID = proj;
				forms.add(form);
			}
			stmt.close();
			rset.close();
			ps.close();
			conn.close();
			return forms;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}

	public DataEntryForm getForm(Integer formID){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from data_entry_form
where form_id=?");
			ps.setInt(1, formID);
			ResultSet rset = ps.executeQuery();
			while(rset.next()){
				DataEntryForm form =
new DataEntryForm();
				form.formID =
Integer.valueOf(rset.getString("form_id"));
				form.formName =
rset.getString("form_name");
				form.formDesc =
rset.getString("form_desc");
				form.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				form.dateUpdated =
java.sql.Timestamp.valueOf(rset.getString("date_updated"));
				User user = new User();
				user.username =
rset.getString("creator");
				form.creator = user;
				Project proj = new
Project();
				proj.projID =
Integer.valueOf(rset.getString("proj_id"));
```

```
                                        form.projID = proj;
                                        stmt.close();
                                        return form;
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return null;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
        }

        public void deleteForm(Integer formID, Integer projID){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("delete from data_entry_form
where form_id=?");
                                ps.setInt(1, formID);
                                ps.executeUpdate();
                                File path = new
File(getServletContext().getRealPath("/") + "/data/" + projID + "/" + formID
+ ".xls");
                                path.delete();
                                stmt.close();
                                ps.close();
                                conn.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
        }

        public void deleteProjectData(Integer projID){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                File path = new
File(getServletContext().getRealPath("/") + "/data/" + projID + "/");
                                deleteDir(path);
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
data_entry_form WHERE proj_id=?");
                                ps.setString(1, projID.toString());
                                ps.executeUpdate();
                                stmt.close();
                                ps.close();
                                conn.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
        }

        public void deleteProjectData(Integer projID, String root){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                File path = new File(root + "/data/" +
projID + "/");
                                deleteDir(path);
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
data_entry_form WHERE proj_id=?");
                                ps.setString(1, projID.toString());
                                ps.executeUpdate();
                                stmt.close();
                                ps.close();
                                conn.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
        }

        public static boolean deleteDir(File dir) {
            if (dir.isDirectory()) {
              String[] children = dir.list();
              for (int i = 0; i < children.length; i++) {
                boolean success = deleteDir(new File(dir, children[i]));
                if (!success) {
                  return false;
                }
              }
            }
            return dir.delete();
        }

        public String saveForm(DataEntryForm form, Boolean
isUpdate){

        if(isUpdate==null||!checkDuplicateName(form,(isUpdate?true:f
alse))){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt =
conn.createStatement();
                                        PreparedStatement ps;

        if(isUpdate!=null&&isUpdate){
                                                form.creator
= getForm(form.formID).creator;

        form.dateCreated = getForm(form.formID).dateCreated;
                                                ps =
(PreparedStatement) conn.prepareStatement("delete from data_entry_form
where form_id=?");
                                                ps.setInt(1,
form.formID);

        ps.executeUpdate();
                                                }
                                                ps =
(PreparedStatement)conn.prepareStatement("INSERT INTO
data_entry_form
("+(isUpdate!=null&&isUpdate?"form_id,":"")+"form_name,form_desc,dat
e_created,date_updated,creator,proj_id) " +

                "VALUES
("+(isUpdate!=null&&isUpdate?form.formID+",":"")+"?,?,?,?,?,?)");
                                                ps.setString(1,
form.formName);
                                                ps.setString(2,
form.formDesc);
                                                ps.setString(3,
(isUpdate!=null&&isUpdate?form.dateCreated.toString():form.getTimestam
p().toString()));
                                                ps.setString(4,
form.getTimestamp().toString());
                                                ps.setString(5,
form.creator.username);
                                                ps.setInt(6,
form.projID.projID);
                                                ps.executeUpdate();
                                                stmt.close();
                                                ps.close();
                                                conn.close();
                                                if(!isUpdate){

        ExcelServiceImpl excel = new ExcelServiceImpl();
//
        excel.createExcel(getID(form.formName, form.projID.projID),
getServletContext().getRealPath("/") + "data/" + form.projID.projID + "/");

        excel.writeExcel(getID(form.formName, form.projID.projID),
getServletContext().getRealPath("/") + "data/" + form.projID.projID + "/");
                                                }
                                                return "ok";
                                } catch (SQLException e) {
                                        e.printStackTrace();
```

```
                                        return "Error adding
form";
                                }
                                catch (IOException e) {
                                        e.printStackTrace();
                                        return "Error adding
form";
                                }
                        }else{
                                return "Form name already exists";
                        }
                }

                private Integer getID(String formName, Integer projID) {
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from data_entry_form
where form_name=? and proj_id=?");
                                ps.setString(1, formName);
                                ps.setInt(2, projID);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        String id =
rset.getString("form_id");
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        rset.close();
                                        return
Integer.valueOf(id);
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return null;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
                }

                private Boolean checkDuplicateName(DataEntryForm form,
Boolean isUpdate){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from data_entry_form
where form_name=? and proj_id=?");
                                ps.setString(1, form.formName);
                                ps.setInt(2, form.projID.projID);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        if(isUpdate){

                if(!Integer.valueOf(rset.getString("form_id")).equals(form.form
ID)){

                stmt.close();

                rset.close();

                ps.close();

                conn.close();

                return true;
                                                }
                                        }else{
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return true;
                                        }
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return false;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

}
```

MailUtil.java

```
package cepir.server;

/*
 * Sudhir Ancha
 * Site :
http://www.javacommerce.com/displaypage.jsp?name=javamail.sql&id=182
74
 *
 * Shams Zawoad Ratul
 * Site: http://zawoad.blogspot.com/2010/05/sending-mail-using-javamail-
api-and.html
 */

import javax.mail.*;
import javax.mail.internet.*;

import cepir.client.MailService;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

import java.util.*;

public class MailUtil extends RemoteServiceServlet implements
MailService{

        private String SMTP_HOST_NAME = "localhost";
        private String SMTP_FROM = "accounts@cepir.upm.edu.ph";

        public Boolean postMail( String recipients[ ], String subject,
String message){
                try {
                        boolean debug = false;

                        //Set the host smtp address
                        Properties props = new Properties();
                        props.put("mail.smtp.host",
SMTP_HOST_NAME);
                        props.put("mail.from",
SMTP_FROM);

                        //Create a Session from the Properties
and the Authenticator
                        Session session =
Session.getInstance(props);
                        session.setDebug(debug);

                        // Create a MimeMessage
                        MimeMessage msg = new
MimeMessage(session);

                        // Set the from and to address
                        InternetAddress addressFrom = new
InternetAddress(SMTP_FROM);
                        msg.setFrom(addressFrom);

                        InternetAddress[] addressTo = new
InternetAddress[recipients.length];
                        for (int i = 0; i < recipients.length;
i++) {
                                addressTo[i] = new
InternetAddress(recipients[i]);
```

```
                }

        msg.setRecipients(Message.RecipientType.TO, addressTo);

                //Set message subject and text
                msg.setSubject(subject);
                msg.setText(message);

                Transport.send(msg);
            } catch (MessagingException
messagingException) {

        messagingException.printStackTrace();
                return false;
            } catch (Exception ex) {
                ex.printStackTrace();
                return false;
            }
            return true;
        }

}
```

MiscFileServiceImpl.java

```
package cepir.server;

import java.io.File;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import com.mysql.jdbc.PreparedStatement;

import cepir.client.MiscFileService;
import cepir.shared.MiscFile;
import cepir.shared.MiscFileFolder;
import cepir.shared.Project;
import cepir.shared.User;

public class MiscFileServiceImpl extends SessionMngt implements
MiscFileService{

        public MiscFile getMiscFile(Integer id){
            Connection conn =
DBConnection.getConnection();
            Statement stmt;
            try {
                stmt = conn.createStatement();
                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from misc_file where
misc_file_id=?");
                ps.setInt(1, id);
                ResultSet rset = ps.executeQuery();
                if(rset.next()){
                    MiscFile file = new
MiscFile();
                    file.setMiscFileID(id);

        file.setMiscFileName(rset.getString("misc_file_name"));

        file.setMiscFileDesc(rset.getString("misc_file_desc"));

        file.setMiscFileLink(rset.getString("misc_file_link"));

        file.setMiscFilePath(rset.getString("misc_file_path"));

        file.setDateCreated(java.sql.Timestamp.valueOf(rset.getString(
"date_created")));
                    User user = new User();
                    user.username =
rset.getString("creator");
                    file.setCreator(user);
                    Project proj = new
Project();
                    proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                    file.project = proj;
                    stmt.close();
                    rset.close();
                    ps.close();
                    conn.close();
                    return file;
                }
                stmt.close();
                rset.close();
                ps.close();
                conn.close();
                return null;
            } catch (SQLException e) {
                e.printStackTrace();
                return null;
            }
        }

        public String saveMiscFile(MiscFile file, Boolean isUpdate){

        if(isUpdate==null||!checkLinkIfDuplicate(file,(isUpdate?true:fa
lse))){
            Connection conn =
DBConnection.getConnection();
            Statement stmt;
            try {
                stmt =
conn.createStatement();
                PreparedStatement ps;

        if(isUpdate!=null&&isUpdate){

        file.dateCreated = getMiscFile(file.miscFileID).dateCreated;

        file.miscFilePath = getMiscFile(file.miscFileID).miscFilePath;
                    file.creator
= getMiscFile(file.miscFileID).creator;
                    ps =
(PreparedStatement) conn.prepareStatement("delete from misc_file where
misc_file_id=?");
                    ps.setInt(1,
file.miscFileID);

        ps.executeUpdate();
                }
                ps =
(PreparedStatement)conn.prepareStatement("INSERT INTO misc_file
("+(isUpdate!=null&&isUpdate?"misc_file_id,":"")+"misc_file_name,misc_
file_desc,misc_file_link,misc_file_path,date_created,creator,type,proj_id) "
+

        "VALUES
("+(isUpdate!=null&&isUpdate?file.miscFileID+",":"")+"?,?,?,?,?,?,?,?)");
                ps.setString(1,
file.miscFileName);
                ps.setString(2,
file.miscFileDesc);
                ps.setString(3,
file.miscFileLink);
                ps.setString(4,
file.miscFilePath);
                ps.setString(5,
(isUpdate!=null&&isUpdate?file.dateCreated.toString():file.getTimestamp()
.toString()));
                ps.setString(6,
file.creator.username);
                ps.setInt(7, file.type);
                ps.setInt(8,
file.project.projID);
                ps.executeUpdate();
                stmt.close();
                ps.close();
                conn.close();
                return "ok";
            } catch (SQLException e) {
                e.printStackTrace();
                return "Error adding
link";
            }
        }else{
            if(file.type==1){
```

```
                                        return "Filename already
exists in current location";
                                }else{
                                        return "Title already
exists in current location";
                                }
                        }
                }

                private Boolean checkLinkIfDuplicate(MiscFile file, Boolean
isUpdate){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from misc_file where
misc_file_path =? and misc_file_name=?");
                                ps.setString(1, file.miscFilePath);
                                ps.setString(2, file.miscFileName);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        if(isUpdate){

                if(!file.miscFileID.toString().equals(rset.getString("misc_file_i
d"))){

                stmt.close();

                rset.close();

                ps.close();

                conn.close();

                return true;
                                                }
                                        }else{
                                                stmt.close();
                                                rset.close();
                                                ps.close();

                conn.close();
                                                return true;
                                        }
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return false;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                public MiscFileFolder getFolder(Integer folderID){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from miscfile_folder
where miscfile_folder_id=?");
                                ps.setInt(1, folderID);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        MiscFileFolder folder =
new MiscFileFolder();

                folder.setMiscfileFolderID(folderID);

                folder.setMiscfileFolderName(rset.getString("miscfile_folder_n
ame"));

                folder.setMiscfileFolderDesc(rset.getString("miscfile_folder_de
sc"));

                folder.setMiscfileFolderPath(rset.getString("miscfile_folder_pa
th"));

                folder.setDateCreated(java.sql.Timestamp.valueOf(rset.getStrin
g("date_created")));
                                        User user = new User();
                                        user.username =
rset.getString("creator");
                                        folder.creator = user;
                                        Project proj = new
Project();
                                        proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                        folder.setProject(proj);
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return folder;
                                }
                                return null;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
                }

                public String saveDirectory(MiscFileFolder folder, Boolean
isUpdate){
                        String root = getServletContext().getRealPath("/")
+ "/uploads/misc_file/"+folder.project.projID+"/";

                if(isUpdate==null||!checkIfDuplicate(folder,(isUpdate?true:fals
e))){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt =
conn.createStatement();
                                        PreparedStatement ps;

                if(isUpdate!=null&&isUpdate){

                folder.dateCreated =
getFolder(folder.miscfileFolderID).dateCreated;

                folder.miscfileFolderPath =
getFolder(folder.miscfileFolderID).miscfileFolderPath;

                folder.creator = getFolder(folder.miscfileFolderID).creator;
                                                ps =
(PreparedStatement) conn.prepareStatement("delete from miscfile_folder
where miscfile_folder_id=?");
                                                ps.setInt(1,
folder.miscfileFolderID);

                ps.executeUpdate();
                                        }
                                        ps = (PreparedStatement)
conn.prepareStatement("insert into miscfile_folder
("+(isUpdate!=null&&isUpdate?"miscfile_folder_id,":"")+"miscfile_folder_
name,miscfile_folder_desc,miscfile_folder_path,date_created,creator,proj_id
) " +

                "values
("+(isUpdate!=null&&isUpdate?folder.miscfileFolderID+",":"")+"?,?,?,?,?,?
)");
                                        ps.setString(1,
folder.miscfileFolderName);
                                        ps.setString(2,
folder.miscfileFolderDesc);
                                        ps.setString(3,
folder.miscfileFolderPath);
                                        ps.setString(4,
(isUpdate!=null&&isUpdate?folder.dateCreated.toString():folder.getTimesta
mp().toString()));
                                        ps.setString(5,
folder.creator.username);
```

```
                                        ps.setInt(6,
folder.project.projID);
                                        ps.executeUpdate();
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        File path = new File(root
+ folder.miscfileFolderPath + getMiscFileFolderID(folder));
                                        if (!path.exists()) {
                                                boolean
status = path.mkdirs();
                                        }
                                        return "ok";
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return "Error adding
file";
                                }
                        }else{
                                return "Folder already exists in
current location";
                        }
                }

                private Boolean checkIfDuplicate(MiscFileFolder folder,
Boolean isUpdate){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from miscfile_folder
where miscfile_folder_path =? and miscfile_folder_name=?");
                                ps.setString(1,
folder.miscfileFolderPath);
                                ps.setString(2,
folder.miscfileFolderName);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        if(isUpdate){

                if(!folder.miscfileFolderID.toString().equals(rset.getString("mis
cfile_folder_id"))){

                stmt.close();

                rset.close();

                ps.close();

                conn.close();

                return true;
                                                }
                                        }else{
                                                stmt.close();
                                                rset.close();
                                                ps.close();


                conn.close();
                                                return true;
                                        }
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return false;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                private Integer getMiscFileFolderID(MiscFileFolder folder){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from miscfile_folder
where creator=? and miscfile_folder_path=? and miscfile_folder_name=?");
                                ps.setString(1,
folder.creator.username);
                                ps.setString(2,
folder.miscfileFolderPath);
                                ps.setString(3,
folder.miscfileFolderName);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        String id =
rset.getString("miscfile_folder_id");
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        rset.close();
                                        return
Integer.valueOf(id);
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return null;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
                }

                public Boolean checkIfDirExists(String currLoc){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                if(!currLoc.isEmpty()){
                                        stmt =
conn.createStatement();
                                        String[] parse =
currLoc.split("/");
                                        String path = "";
                                        for(int i = 0; i <
parse.length-1; i++){
                                                path +=
parse[i]+"/";
                                        }
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from miscfile_folder
where miscfile_folder_path =? and miscfile_folder_id=?");
                                        ps.setString(1, path);
                                        ps.setInt(2,
Integer.valueOf(parse[parse.length-1]));
                                        ResultSet rset =
ps.executeQuery();
                                        if(rset.next()){
                                                stmt.close();
                                                rset.close();
                                                ps.close();

                        conn.close();
                                                return true;
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return false;
                                }else{
                                        conn.close();
                                        return true;
                                }
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }
```

```
	public ArrayList<MiscFileFolder> getFolders(Integer projID,
String currLoc, String query){
		ArrayList<MiscFileFolder> folders = new
ArrayList<MiscFileFolder>();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from miscfile_folder
where miscfile_folder_path =? and proj_id=? and " +

	"(miscfile_folder_name like ? or miscfile_folder_desc like ?)
order by miscfile_folder_name asc");
			ps.setString(1, currLoc);
			ps.setInt(2, projID);
			ps.setString(3, "%"+query+"%");
			ps.setString(4, "%"+query+"%");
			ResultSet rset = ps.executeQuery();
//			ResultSet rset =
//
	stmt.executeQuery("select * from miscfile_folder where
miscfile_folder_path = '"+currLoc+"' and (miscfile_folder_name like '%"
//
	+query+"%' or miscfile_folder_desc like '%"+query+"%') order
by miscfile_folder_name asc");
			while(rset.next()){
				MiscFileFolder folder =
new MiscFileFolder();
				folder.miscfileFolderID
= Integer.valueOf(rset.getString("miscfile_folder_id"));

	folder.miscfileFolderName =
rset.getString("miscfile_folder_name");

	folder.miscfileFolderDesc =
rset.getString("miscfile_folder_desc");
				folder.miscfileFolderPath
= rset.getString("miscfile_folder_path");
				folder.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				User user = new User();
				user.username =
rset.getString("creator");
				folder.creator = user;
				Project proj = new
Project();
				proj.projID =
Integer.valueOf(rset.getString("proj_id"));
				folder.setProject(proj);
				folders.add(folder);
			}
			stmt.close();
			rset.close();
			ps.close();
			conn.close();
			return folders;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}

	public ArrayList<MiscFile> getFiles(Integer projID, String
currLoc, String query){
		ArrayList<MiscFile> files = new
ArrayList<MiscFile>();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from misc_file where
misc_file_path =? and proj_id=? and " +
				"(misc_file_name like ?
or misc_file_desc like ?) order by misc_file_name asc");
			ps.setString(1, currLoc);
			ps.setInt(2, projID);
			ps.setString(3, "%"+query+"%");
			ps.setString(4, "%"+query+"%");
			ResultSet rset = ps.executeQuery();
//			ResultSet rset =
//
	stmt.executeQuery("select * from tool where tool_path =
'"+currLoc+"' and (tool_name like '%"
//
	+query+"%' or tool_desc like '%"+query+"%') order by
tool_name asc");
			while(rset.next()){
				MiscFile file = new
MiscFile();
				file.miscFileID =
Integer.valueOf(rset.getString("misc_file_id"));
				file.miscFileName =
rset.getString("misc_file_name");
				file.miscFileDesc =
rset.getString("misc_file_desc");
				file.miscFilePath =
rset.getString("misc_file_path");
				file.miscFileLink =
rset.getString("misc_file_link");
				file.type =
Integer.valueOf(rset.getString("type"));
				file.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				User user = new User();
				user.username =
rset.getString("creator");
				file.creator = user;
				files.add(file);
			}
			stmt.close();
			rset.close();
			ps.close();
			conn.close();
			return files;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}

	public List<String> getFolderNamesFromIds(List<String>
ids){
		List<String> names = new ArrayList<String>();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			for(int i = 0; i < ids.size(); i++){
				if(!ids.get(i).isEmpty()){
					ResultSet
rset =

	stmt.executeQuery("select miscfile_folder_name from
miscfile_folder where miscfile_folder_id="+ids.get(i));

	if(rset.next()){

	names.add(rset.getString("miscfile_folder_name"));
					}else{

	names.add("");
					}
				}else{

	names.add("");
				}
			}
			stmt.close();
			conn.close();
			return names;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}
```

```
	public void deleteMiscFileFolder(Integer miscFileFolderID,
Integer projID, String currLoc){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			File path = new
File(getServletContext().getRealPath("/") + "/uploads/misc_file/" + projID +
"/" + currLoc + miscFileFolderID +"/");
			deleteDir(path);
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
miscfile_folder WHERE miscfile_folder_path LIKE ? or
miscfile_folder_path LIKE ?");
			ps.setString(1,
miscFileFolderID+"/%");
			ps.setString(2,
"%/"+miscFileFolderID+"/%");
			ps.executeUpdate();

			ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM misc_file
WHERE misc_file_path LIKE ? or misc_file_path LIKE ?");
			ps.setString(1,
miscFileFolderID+"/%");
			ps.setString(2,
"%/"+miscFileFolderID+"/%");
			ps.executeUpdate();

			ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
miscfile_folder WHERE miscfile_folder_id=?");
			ps.setInt(1, miscFileFolderID);
			ps.executeUpdate();
//			int query =
stmt.executeUpdate("DELETE FROM tool_folder WHERE
tool_folder_path LIKE '"+toolFolderID+"/%' OR tool_folder_path LIKE
'%/"+toolFolderID+"/%'");
//			query =
stmt.executeUpdate("DELETE FROM tool WHERE tool_path LIKE
'"+toolFolderID+"/%' OR tool_path LIKE '%/"+toolFolderID+"/%'");
//			query =
stmt.executeUpdate("DELETE FROM tool_folder WHERE
tool_folder_id="+toolFolderID);
			stmt.close();
			ps.close();
			conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public void deleteProjectMiscFiles(Integer projID){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			File path = new
File(getServletContext().getRealPath("/") + "/uploads/misc_file/" + projID +
"/");
			deleteDir(path);
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
miscfile_folder WHERE proj_id=?");
			ps.setString(1, projID.toString());
			ps.executeUpdate();

			ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM misc_file
WHERE proj_id=?");
			ps.setString(1, projID.toString());
			ps.executeUpdate();
			stmt.close();
			ps.close();
			conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public void deleteProjectMiscFiles(Integer projID, String root){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			File path = new File(root +
"/uploads/misc_file/" + projID + "/");
			deleteDir(path);
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
miscfile_folder WHERE proj_id=?");
			ps.setString(1, projID.toString());
			ps.executeUpdate();

			ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM misc_file
WHERE proj_id=?");
			ps.setString(1, projID.toString());
			ps.executeUpdate();
			stmt.close();
			ps.close();
			conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public void deleteMiscFile(Integer fileID, Integer projID,
String currLoc){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			File path = new
File(getServletContext().getRealPath("/") + "/uploads/misc_file/" + projID +
"/" +currLoc +getMiscFile(fileID).miscFileName);
			path.delete();
			int query =
stmt.executeUpdate("DELETE FROM misc_file WHERE
misc_file_id="+fileID);
			stmt.close();
			conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public static boolean deleteDir(File dir) {
	   if (dir.isDirectory()) {
	     String[] children = dir.list();
	     for (int i = 0; i < children.length; i++) {
	       boolean success = deleteDir(new File(dir, children[i]));
	       if (!success) {
	         return false;
	       }
	     }
	   }
	   return dir.delete();
	}
}
```

ProjectServiceImpl.java

```
package cepir.server;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import com.mysql.jdbc.PreparedStatement;

import cepir.client.ProjectService;
import cepir.shared.NewProjRequest;
import cepir.shared.ProjMemRequest;
```

```
import cepir.shared.ProjMemRequestPK;
import cepir.shared.Project;
import cepir.shared.Role;
import cepir.shared.User;

public class ProjectServiceImpl extends SessionMngt implements
ProjectService{

	public ArrayList<Project> getProjects(){
		ArrayList<Project> projects = new
ArrayList<Project>();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			ResultSet rset =

	stmt.executeQuery("select * from project order by proj_name
asc");
			while(rset.next()){
				Project proj = new
Project();
				proj.projID =
Integer.valueOf(rset.getString("proj_id"));
				proj.projName =
rset.getString("proj_name");
				proj.projDesc =
rset.getString("proj_desc");
				proj.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				proj.dateUpdated =
java.sql.Timestamp.valueOf(rset.getString("date_updated"));
				projects.add(proj);
			}
			stmt.close();
			rset.close();
			conn.close();
			return projects;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}

	public Project getProject(Integer projID){
		Project proj = new Project();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			ResultSet rset =

	stmt.executeQuery("select * from project where proj_id =
"+projID);
			while(rset.next()){
				proj.projID =
rset.getInt("proj_id");
				proj.projName =
rset.getString("proj_name");
				proj.projDesc =
rset.getString("proj_desc");
				proj.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				proj.dateUpdated =
java.sql.Timestamp.valueOf(rset.getString("date_updated"));
			}
			stmt.close();
			rset.close();
			conn.close();
			return proj;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}

	public ArrayList<Project> getProjectSearchList(String query){
		ArrayList<Project> projects = new
ArrayList<Project>();
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from project where
proj_name like ? or proj_desc like ? order by proj_name asc");
			ps.setString(1, "%"+query+"%");
			ps.setString(2, "%"+query+"%");
			ResultSet rset = ps.executeQuery();
			while(rset.next()){
				Project proj = new
Project();
				proj.projID =
Integer.valueOf(rset.getString("proj_id"));
				proj.projName =
rset.getString("proj_name");
				proj.projDesc =
rset.getString("proj_desc");
				proj.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
				proj.dateUpdated =
java.sql.Timestamp.valueOf(rset.getString("date_updated"));
				projects.add(proj);
			}
			stmt.close();
			rset.close();
			ps.close();
			conn.close();
			return projects;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}

	public void deleteProject(Integer projID){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		try {
			stmt = conn.createStatement();
			int query =
stmt.executeUpdate("DELETE FROM user_priv WHERE
proj_id="+projID);
			query =
stmt.executeUpdate("DELETE FROM proj_mem_request WHERE
proj_id="+projID);
			ReferenceServiceImpl ref = new
ReferenceServiceImpl();
			ref.deleteProjectReferences(projID,
getServletContext().getRealPath("/"));
			MiscFileServiceImpl file = new
MiscFileServiceImpl();
			file.deleteProjectMiscFiles(projID,
getServletContext().getRealPath("/"));
			FormServiceImpl form = new
FormServiceImpl();
			form.deleteProjectData(projID,
getServletContext().getRealPath("/"));
			query = stmt.executeUpdate("delete
from project where proj_id="+ projID);
			stmt.close();
			conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public ArrayList<User> getProjAdmins(Integer projID){
		Connection conn =
DBConnection.getConnection();
		Statement stmt;
		ArrayList<User> projAdmins = new
ArrayList<User>();
		try {
			stmt = conn.createStatement();
```

```
                                        ResultSet rset =

                stmt.executeQuery("SELECT u.username, u.first_name,
u.middle_name, u.last_name, u.email FROM user_priv up INNER JOIN role
r ON r.role_id = up.role_id " +

                "INNER JOIN project p ON p.proj_id = up.proj_id INNER
JOIN user u ON u.username = up.username WHERE r.role_name =  'Project
Administrator' AND p.proj_id ="+projID);
                                        while(rset.next()){
                                                User pAdmin = new
User();

                pAdmin.setUsername(rset.getString("username"));

                pAdmin.setFirstName(rset.getString("first_name"));

                pAdmin.setMiddleName(rset.getString("middle_name"));

                pAdmin.setLastName(rset.getString("last_name"));

                pAdmin.setEmail(rset.getString("email"));

                projAdmins.add(pAdmin);
                                        }
                                        stmt.close();
                                        rset.close();
                                        conn.close();
                                        return projAdmins;
                        } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                        }
                }

                public ArrayList<User> getProjMembers(Integer projID){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        ArrayList<User> projMembers = new
ArrayList<User>();
                        try {
                                        stmt = conn.createStatement();
                                        ResultSet rset =

                stmt.executeQuery("SELECT u.username, u.first_name,
u.middle_name, u.last_name, u.email, r.role_name FROM user_priv up " +

                "INNER JOIN role r ON r.role_id = up.role_id INNER JOIN
user u ON u.username = up.username WHERE up.proj_id ="+projID);
                                        while(rset.next()){
                                                User member = new
User();

                member.setUsername(rset.getString("username"));

                member.setPassword(rset.getString("role_name")); //just stored
project role here

                member.setFirstName(rset.getString("first_name"));

                member.setMiddleName(rset.getString("middle_name"));

                member.setLastName(rset.getString("last_name"));

                member.setEmail(rset.getString("email"));

                projMembers.add(member);
                                        }
                                        stmt.close();
                                        rset.close();
                                        conn.close();
                                        return projMembers;
                        } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                        }
                }

                public ArrayList<User> getProjMembersSearchList(Integer
projID, String query){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        ArrayList<User> projMembers = new
ArrayList<User>();
                        try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT u.username,
u.first_name, u.middle_name, u.last_name, u.email, r.role_name FROM
user_priv up " +

                "INNER JOIN role r ON r.role_id = up.role_id INNER JOIN
user u ON u.username = up.username WHERE up.proj_id = ?"+

                " AND (u.username LIKE  ? OR u.first_name LIKE  ? OR
u.middle_name LIKE  ? OR u.last_name LIKE ?) ORDER BY u.username
ASC");
                                        ps.setInt(1, projID);
                                        ps.setString(2, "%"+query+"%");
                                        ps.setString(3, "%"+query+"%");
                                        ps.setString(4, "%"+query+"%");
                                        ps.setString(5, "%"+query+"%");
                                        ResultSet rset = ps.executeQuery();
                                        while(rset.next()){
                                                User member = new
User();

                member.setUsername(rset.getString("username"));

                member.setPassword(rset.getString("role_name")); //just stored
project role here

                member.setFirstName(rset.getString("first_name"));

                member.setMiddleName(rset.getString("middle_name"));

                member.setLastName(rset.getString("last_name"));

                member.setEmail(rset.getString("email"));

                projMembers.add(member);
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return projMembers;
                        } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                        }
                }

                public Boolean projNameNotUsed(String projName){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from project where
proj_name=?");
                                        ps.setString(1, projName);
                                        ResultSet rset = ps.executeQuery();
                                        if(rset.next()){
                                                return false;
                                        }
                                        PreparedStatement ps2 =
(PreparedStatement) conn.prepareStatement("select * from
new_proj_request where new_proj_request_name=?");
                                        ps2.setString(1, projName);
                                        ResultSet rset2 = ps2.executeQuery();
                                        if(rset2.next()){
                                                return false;
                                        }
                                        stmt.close();
                                        rset.close();
```

```
                                        ps.close();
                                        conn.close();
                                        return true;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return false;
                                }
                        }

                        // used to check if user saved the same project name for the
same project ID
                        public Boolean projNameNotUsed(String projName, Integer
projID){
                                if(projID!=null){
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                stmt =
conn.createStatement();
                                                PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from project where
proj_name=?");
                                                ps.setString(1,
projName);
                                                ResultSet rset =
ps.executeQuery();
                                                if(rset.next()){

                        if(!rset.getString("proj_id").equals(projID.toString())){

                        return false;
                                                        }
                                                }
                                                PreparedStatement ps2 =
(PreparedStatement) conn.prepareStatement("select * from
new_proj_request where new_proj_request_name=?");
                                                ps2.setString(1,
projName);
                                                ResultSet rset2 =
ps2.executeQuery();
                                                if(rset2.next()){
                                                        return false;
                                                }
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return true;
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                                return false;
                                        }
                                }else{
                                        return projNameNotUsed(projName);
                                }
                        }

                        public Boolean checkIfSoleProjAdmin(String username){
                                ArrayList<Project> projects = getProjects();
                                for(int i = 0; i<projects.size(); i++){
                                        ArrayList<User> admins =
getProjAdmins(projects.get(i).projID);
                                        for(int j = 0; j < admins.size(); j++){

                        if(admins.get(j).username.equals(username) &&
admins.size()==1){
                                                        return true;
                                                }
                                        }
                                }
                                return false;
                        }

                        public Role getRoleInProj(String username, Integer projID){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        Role role = new Role();
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT r.role_id,
r.role_name FROM user_priv up INNER JOIN role r ON r.role_id =
up.role_id WHERE " +

                        "up.username = ? AND up.proj_id = ?");
                                        ps.setString(1, username);
                                        ps.setInt(2, projID);
                                        ResultSet rset = ps.executeQuery();
                                        while(rset.next()){
                                                role.roleID =
Integer.valueOf(rset.getString("role_id"));
                                                role.roleName =
rset.getString("role_name");
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return role;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                                }
                        }

                        public ArrayList<String> getMemberships(String username,
ArrayList<Project> projects){
                                ArrayList<String> memberships = new
ArrayList<String>();
                                for(int i = 0; i < projects.size(); i++){
                                        Role role = getRoleInProj(username,
projects.get(i).projID);

                        if(role.roleID==null||role.roleName==null){

                        if(getProjMemRequest(username,
projects.get(i).projID)==null){

                        memberships.add("-");
                                                }else{

                        memberships.add("Membership Request Pending");
                                                }
                                        }else{

                        memberships.add(role.roleName);
                                        }
                                }
                                return memberships;
                        }

                        /*
                         *              roleName = 'Project Administrator', 'Project
Member'
                         * */
                        public String addProjMembers(Integer projID,
ArrayList<String> projAdmins, Boolean isUpdate, String roleName){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps;
                                        if(isUpdate){
                                                ps =
(PreparedStatement)conn.prepareStatement("delete from user_priv where
proj_id = ?");
                                                ps.setInt(1, projID);
                                                ps.executeUpdate();
                                        }
                                        for(int ctr = 0; ctr<projAdmins.size();
ctr++){
                                                ps =
(PreparedStatement)conn.prepareStatement("insert into user_priv
(username,proj_id,role_id) select ?,?,(select role_id from role where
role_name=?)");
                                                ps.setString(1,
projAdmins.get(ctr));
```

```
                                        ps.setInt(2, projID);
                                        ps.setString(3,
roleName);
                                        ps.executeUpdate();
                                }
                                stmt.close();
                                conn.close();
                                return "ok";
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return "Error sending project request";
                        }
                }

                public String addProject(Project proj, ArrayList<String>
projAdmin, Boolean isUpdate){

                if((isUpdate==null)||(isUpdate?projNameNotUsed(proj.projNa
me,proj.projID):projNameNotUsed(proj.projName))){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt =
conn.createStatement();
                                        PreparedStatement ps;
                                        if(isUpdate!=null){

                if(isUpdate){

                proj.dateCreated = getProject(proj.projID).dateCreated;

                if(!projAdmin.isEmpty()){

                ps = (PreparedStatement)conn.prepareStatement("DELETE up
FROM user_priv up INNER JOIN role r ON r.role_id = up.role_id " +

                                "INNER JOIN project p ON p.proj_id
= up.proj_id WHERE r.role_name =  'Project Administrator' AND
p.proj_name=?");

                ps.setString(1, proj.projName);

                ps.executeUpdate();

                }

                ps = (PreparedStatement)conn.prepareStatement("delete from
project where proj_id = ?");

                ps.setInt(1, proj.projID);

                ps.executeUpdate();
                                                }
                                        }
                                        ps =
(PreparedStatement)conn.prepareStatement("insert into project
("+(isUpdate==null||!isUpdate?"":"proj_id,")+"proj_name,proj_desc,date_cr
eated,date_updated)
values("+(isUpdate==null||!isUpdate?"":proj.projID.toString()+",")+"?,?,?,?)
");
                                        ps.setString(1,
proj.projName);
                                        ps.setString(2,
proj.projDesc);
                                        ps.setString(3,
(isUpdate==null||!isUpdate?proj.getTimestamp().toString():proj.dateCreated.
toString()));
                                        ps.setString(4,
proj.getTimestamp().toString());
                                        ps.executeUpdate();
                                        for(int ctr = 0;
ctr<projAdmin.size(); ctr++){
                                                ps =
(PreparedStatement)conn.prepareStatement("insert into user_priv
(username,proj_id,role_id) select ?,(select proj_id from project where
proj_name=?),(select role_id from role where role_name='Project
Administrator')");

                ps.setString(1, projAdmin.get(ctr));
                ps.setString(2, proj.projName);

                ps.executeUpdate();
                                        }
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        return "ok";
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return "Error adding
project";
                                }
                        }else{
                                return "";
                        }
                }

                public String addProjRequest(NewProjRequest newProj){
                        if(projNameNotUsed(newProj.newProjName)){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt =
conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("insert into new_proj_request
(new_proj_request_name,new_proj_desc,username) values (?,?,?)");
                                        ps.setString(1,
newProj.newProjName);
                                        ps.setString(2,
newProj.newProjDesc);
                                        ps.setString(3,
newProj.user.username);
                                        ps.executeUpdate();
                                        stmt.close();
                                        return "ok";
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return "Error sending
project request";
                                }
                        }else{
                                return "";
                        }
                }


                public ArrayList<NewProjRequest> getProjRequests(){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                ArrayList<NewProjRequest> projects
= new ArrayList<NewProjRequest>();
                                stmt = conn.createStatement();
                                ResultSet rset =

                stmt.executeQuery("SELECT npr.new_proj_request_name,
npr.new_proj_desc, u.email, u.username " +

                "FROM new_proj_request npr INNER JOIN user u ON
npr.username = u.username ORDER BY npr.new_proj_request_name
ASC");
                                while(rset.next()){
                                        NewProjRequest proj =
new NewProjRequest();
                                        proj.newProjName =
rset.getString("new_proj_request_name");
                                        proj.newProjDesc =
rset.getString("new_proj_desc");
                                        User user = new User();
                                        user.username =
rset.getString("username");
                                        user.email =
rset.getString("email");
                                        proj.user = user;
                                        projects.add(proj);
```

```
                        }
                        stmt.close();
                        return projects;
                    } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                    }
                }

                public ArrayList<NewProjRequest> getProjRequests(String
username){ //ok
                    Connection conn =
DBConnection.getConnection();
                    try {
                        ArrayList<NewProjRequest> projects
= new ArrayList<NewProjRequest>();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT
npr.new_proj_request_id, npr.new_proj_request_name, npr.new_proj_desc,
u.email, u.username " +
                                                    "FROM
new_proj_request npr INNER JOIN user u ON npr.username = u.username
WHERE " +

                "npr.username =? ORDER BY npr.new_proj_request_name
ASC");
                        ps.setString(1, username);
                        ResultSet rset = ps.executeQuery();
                        while(rset.next()){
                            NewProjRequest proj =
new NewProjRequest();
                            proj.newProjRequestID =
Integer.valueOf(rset.getString("new_proj_request_id"));
                            proj.newProjName =
rset.getString("new_proj_request_name");
                            proj.newProjDesc =
rset.getString("new_proj_desc");
                            User user = new User();
                            user.username =
rset.getString("username");
                            user.email =
rset.getString("email");
                            proj.user = user;
                            projects.add(proj);
                        }
                        return projects;
                    } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                    }
                }

                public ArrayList<NewProjRequest>
getProjRequestsSearchList(String query){
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        ArrayList<NewProjRequest> projects
= new ArrayList<NewProjRequest>();
                        query.trim();
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from
new_proj_request where new_proj_request_name like ?"+
                                                    " or
new_proj_desc like ? order by new_proj_request_name asc");
                        ps.setString(1, "%"+query+"%");
                        ps.setString(2, "%"+query+"%");
                        ResultSet rset = ps.executeQuery();
                        while(rset.next()){
                            NewProjRequest proj =
new NewProjRequest();
                            proj.newProjName =
rset.getString("new_proj_request_name");
                            proj.newProjDesc =
rset.getString("new_proj_desc");
                            User user = new User();
                            user.username =
rset.getString("username");
                            proj.user = user;
                            projects.add(proj);
                        }
                        stmt.close();
                        return projects;
                    } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                    }
                }

                public void deleteProjRequest(String projName){
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("delete from new_proj_request
where new_proj_request_name=?");
                        ps.setString(1, projName);
                        ps.executeUpdate();
                        stmt.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }

                public void deleteUserPriv(String username){
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("DELETE up FROM user_priv
up INNER JOIN role r ON r.role_id = up.role_id "+
                                                    "INNER JOIN user u ON
u.username = up.username WHERE u.username=?");
                        ps.setString(1, username);
                        ps.executeUpdate();
                        stmt.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }

                public void deleteUserPriv(String username, Integer projID){
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("DELETE FROM user_priv
WHERE username = ? and proj_id = ?");
                        ps.setString(1, username);
                        ps.setInt(2, projID);
                        ps.executeUpdate();
                        stmt.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }

                public ArrayList<Project> getProjects(String username){
                    ArrayList<Project> projects = new
ArrayList<Project>();
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT p.proj_id,
p.proj_name, p.proj_desc, up.role_id FROM user_priv up " +
                                                    "INNER
JOIN project p ON p.proj_id = up.proj_id " +
```

```
                                                "INNER
JOIN user u ON up.username = u.username WHERE u.username = ?
ORDER BY p.proj_name ASC");
                                ps.setString(1, username);
                                ResultSet rset = ps.executeQuery();
                                while(rset.next()){
                                        Project proj = new
Project();
                                        proj.projName =
rset.getString("proj_name")+"="+rset.getString("role_id"); //used to store
what type of member
                                        proj.projDesc =
rset.getString("proj_desc");
                                        proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                        projects.add(proj);
                                }
                                stmt.close();
                                return projects;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
                }

                public ArrayList<Project> getProjectsWhereAdmin(String
username){
                        ArrayList<Project> projects = new
ArrayList<Project>();
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT p.proj_id,
p.proj_name, p.proj_desc, up.role_id FROM user_priv up " +
                                                "INNER
JOIN project p ON p.proj_id = up.proj_id " +
                                                "INNER
JOIN user u ON up.username = u.username WHERE u.username = ?");
                                ps.setString(1, username);
                                ResultSet rset = ps.executeQuery();
                                while(rset.next()){
                                        Project proj = new
Project();
                                        proj.projName =
rset.getString("proj_name")+"="+rset.getString("role_id"); //used to store
what type of member
                                        proj.projDesc =
rset.getString("proj_desc");
                                        proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                        projects.add(proj);
                                }
                                stmt.close();
                                return projects;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
                }

                public ProjMemRequest getProjMemRequest(String username,
Integer projID){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("SELECT u.username, u.email,
p.proj_id, p.proj_name, p.proj_desc, pmr.message FROM
proj_mem_request pmr INNER JOIN user u ON u.username =
pmr.username " +

                "INNER JOIN project p ON p.proj_id = pmr.proj_id WHERE
u.username = ? AND p.proj_id = ?");
                                ps.setString(1, username);
                                ps.setInt(2, projID);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        Project project = new
Project();
                                        project.projID =
Integer.valueOf(rset.getString("proj_id"));
                                        project.projName =
rset.getString("proj_name");
                                        project.projDesc =
rset.getString("proj_desc");
                                        User user = new User();
                                        user.username =
rset.getString("username");
                                        user.email =
rset.getString("email");
                                        ProjMemRequestPK
projPK = new ProjMemRequestPK();
                                        projPK.projID = project;
                                        projPK.username = user;
                                        ProjMemRequest proj =
new ProjMemRequest();
                                        proj.projMemRequestPK
= projPK;
                                        proj.message =
rset.getString("message");

                                        return proj;
                                }
                                stmt.close();
                                return null;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
                }

                public ArrayList<ProjMemRequest>
getProjMemRequests(String username){
                        ArrayList<ProjMemRequest> projects = new
ArrayList<ProjMemRequest>();
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("SELECT u.username, u.email,
p.proj_id, p.proj_name, p.proj_desc, pmr.message FROM
proj_mem_request pmr INNER JOIN user u ON u.username =
pmr.username " +

                "INNER JOIN project p ON p.proj_id = pmr.proj_id WHERE
u.username = ?");
                                ps.setString(1, username);
                                ResultSet rset = ps.executeQuery();
                                while(rset.next()){
                                        Project project = new
Project();
                                        project.projID =
Integer.valueOf(rset.getString("proj_id"));
                                        project.projName =
rset.getString("proj_name");
                                        project.projDesc =
rset.getString("proj_desc");
                                        User user = new User();
                                        user.username =
rset.getString("username");
                                        user.email =
rset.getString("email");
                                        ProjMemRequestPK
projPK = new ProjMemRequestPK();
                                        projPK.projID = project;
                                        projPK.username = user;
                                        ProjMemRequest proj =
new ProjMemRequest();
                                        proj.projMemRequestPK
= projPK;
                                        proj.message =
rset.getString("message");
                                        projects.add(proj);
```

```
                        }
                        stmt.close();
                        return projects;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                }
        }

        public ArrayList<ProjMemRequest>
getProjMemRequests(Integer projID, String query){
                ArrayList<ProjMemRequest> projects = new
ArrayList<ProjMemRequest>();
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("SELECT u.username, u.email,
u.first_name, u.middle_name, u.last_name, p.proj_name, p.proj_desc,
pmr.message " +
                                                "FROM
proj_mem_request pmr INNER JOIN user u ON u.username =
pmr.username " +
                                                "INNER
JOIN project p ON p.proj_id = pmr.proj_id WHERE p.proj_id = ? AND
(p.proj_name LIKE ? or u.username LIKE ? or pmr.message LIKE ? or
u.first_name LIKE ? or u.middle_name LIKE ? or u.last_name LIKE ?)");
                        ps.setInt(1, projID);
                        ps.setString(2, "%"+query+"%");
                        ps.setString(3, "%"+query+"%");
                        ps.setString(4, "%"+query+"%");
                        ps.setString(5, "%"+query+"%");
                        ps.setString(6, "%"+query+"%");
                        ps.setString(7, "%"+query+"%");
                        ResultSet rset = ps.executeQuery();
                        while(rset.next()){
                                Project project = new
Project();
                                project.projID = projID;
                                project.projName =
rset.getString("proj_name");
                                project.projDesc =
rset.getString("proj_desc");
                                User user = new User();
                                user.username =
rset.getString("username");
                                user.firstName =
rset.getString("first_name");
                                user.lastName =
rset.getString("last_name");
                                user.middleName =
rset.getString("middle_name");
                                user.email =
rset.getString("email");
                                ProjMemRequestPK
projPK = new ProjMemRequestPK();
                                projPK.projID = project;
                                projPK.username = user;
                                ProjMemRequest proj =
new ProjMemRequest();
                                proj.projMemRequestPK
= projPK;
                                proj.message =
rset.getString("message");
                                projects.add(proj);
                        }
                        stmt.close();
                        return projects;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                }
        }

        public ArrayList<ProjMemRequest>
getMemRequestsWhereAdmin(String username){
                ArrayList<Integer> projIDs =
getWhereProjAdmin(username);
                ArrayList<ProjMemRequest> requests = new
ArrayList<ProjMemRequest>();
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        for(int i = 0; i < projIDs.size(); i++){

        ArrayList<ProjMemRequest> request =
getProjMemRequests(projIDs.get(i),"");
                                requests.addAll(request);
                        }
                        stmt.close();
                        return requests;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                }
        }

        public ArrayList<ProjMemRequest>
getMemRequestsWhereAdminSearchList(String username, String query){
                ArrayList<Integer> projIDs =
getWhereProjAdmin(username);
                ArrayList<ProjMemRequest> requests = new
ArrayList<ProjMemRequest>();
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        for(int i = 0; i < projIDs.size(); i++){

        ArrayList<ProjMemRequest> request =
getProjMemRequests(projIDs.get(i), query);
                                requests.addAll(request);
                        }
                        stmt.close();
                        return requests;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                }
        }

        private ArrayList<Integer> getWhereProjAdmin(String
username){
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                ArrayList<Integer> projIDs = new
ArrayList<Integer>();
                try {
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT up.proj_id FROM
user_priv up INNER JOIN role r ON r.role_id = up.role_id " +
                                                "INNER
JOIN user u ON u.username = up.username WHERE r.role_name =  'Project
Administrator' AND u.username = ?");
                        ps.setString(1, username);
                        ResultSet rset = ps.executeQuery();
                        while(rset.next()){

        projIDs.add(Integer.valueOf(rset.getString("proj_id")));
                        }
                        return projIDs;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                }
        }

        public void deleteProjCreationRequest(Integer newProjID){
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
```

```
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("DELETE FROM
new_proj_request WHERE new_proj_request_id = ?");
                                        ps.setInt(1, newProjID);
                                        ps.executeUpdate();
                                        stmt.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
        }

        public void deleteMemRequest(String username, Integer
projID){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("DELETE FROM
proj_mem_request WHERE username = ? AND proj_id = ?");
                                        ps.setString(1, username);
                                        ps.setInt(2, projID);
                                        ps.executeUpdate();
                                        stmt.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
        }

        public void saveMemRequest(ProjMemRequest request){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("INSERT INTO
proj_mem_request (username, proj_id, message) VALUES (?,?,?)");
                                        ps.setString(1,
request.projMemRequestPK.username.username);
                                        ps.setInt(2,
request.projMemRequestPK.projID.projID);
                                        ps.setString(3, request.message);
                                        ps.executeUpdate();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
        }

        public void approveMemRequest(String username, Integer
projID, Integer memType, Boolean isUpdate){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps;
                                        if(isUpdate){
                                                ps = (PreparedStatement)
conn.prepareStatement("DELETE FROM user_priv WHERE username = ?
AND proj_id = ?");
                                                ps.setString(1,
username);
                                                ps.setInt(2, projID);
                                                ps.executeUpdate();
                                        }
                                        ps =
(PreparedStatement)conn.prepareStatement("INSERT INTO user_priv
(username,proj_id,role_id) VALUES (?,?,?)");
                                        ps.setString(1, username);
                                        ps.setInt(2, projID);
                                        ps.setInt(3, memType);
                                        ps.executeUpdate();
                                        stmt.close();
                                        deleteMemRequest(username,projID);
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
        }
}
```

ReferenceServiceImpl.java

```
package cepir.server;

import java.io.File;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import com.mysql.jdbc.PreparedStatement;

import cepir.client.ReferenceService;
import cepir.shared.Project;
import cepir.shared.Reference;
import cepir.shared.ReferenceFolder;
import cepir.shared.ReferenceType;
import cepir.shared.User;

public class ReferenceServiceImpl extends SessionMngt implements
ReferenceService {
        public Reference getReference(Integer id){
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference where
ref_id=?");
                        ps.setInt(1, id);
                        ResultSet rset = ps.executeQuery();
                        if(rset.next()){
                                Reference file = new
Reference();
                                file.setRefID(id);

                file.setRefName(rset.getString("ref_name"));

                file.setRefLink(rset.getString("ref_link"));

                file.setRefPath(rset.getString("ref_path"));

                file.setTitle(rset.getString("title"));

                file.setBookTitle(rset.getString("book_title"));

                file.setChapTitle(rset.getString("chap_title"));

                file.setJournTitle(rset.getString("journ_title"));

                file.setAuthor(rset.getString("author"));

                file.setEditor(rset.getString("editor"));

                file.setPublisher(rset.getString("publisher"));

                file.setPubDate(rset.getString("pub_date"));

                file.setPubPlace(rset.getString("pub_place"));

                file.setType(ReferenceType.valueOf(rset.getString("type").repl
ace(" ", "_")));

                file.setVolume(rset.getString("volume"));

                file.setIssue(rset.getString("issue"));

                file.setPageNum(rset.getString("page_num"));

                file.setArticleName(rset.getString("article_name"));

                file.setYear(rset.getString("year"));

                file.setSchool(rset.getString("school"));
```

```
file.setDateCreated(java.sql.Timestamp.valueOf(rset.getString(
"date_created")));
                                                User user = new User();
                                                user.username =
rset.getString("creator");
                                                file.setCreator(user);
                                                Project proj = new
Project();
                                                proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                                file.project = proj;
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return file;
                                        }
                                        return null;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                                }
                        }

                        public String saveReference(Reference file, Boolean isUpdate){

                        if(isUpdate==null||!checkLinkIfDuplicate(file,(isUpdate?true:fa
lse))){
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                stmt =
conn.createStatement();
                                                PreparedStatement ps;

                        if(isUpdate!=null&&isUpdate){

                        file.dateCreated = getReference(file.refID).dateCreated;
                                                                        file.refPath
= getReference(file.refID).refPath;
                                                                        file.creator
= getReference(file.refID).creator;
//
                        if(!getReference(file.refID).refName.isEmpty()){
//
                        file.refName = getReference(file.refID).refName;
//                                                                      }

                        if(!getReference(file.refID).refName.isEmpty()){

                        if(file.refName.isEmpty()){

                        file.refName = getReference(file.refID).refName;

                        }else{

                        String root = getServletContext().getRealPath("/") +
"/uploads/ref/"+file.project.projID+"/"+file.refPath;

                        File ref = new File(root+getReference(file.refID).refName+"/");

                        ref.delete();

                        }
                                                                        }
                                                                        ps =
(PreparedStatement) conn.prepareStatement("delete from reference where
ref_id=?");
                                                                        ps.setInt(1,
file.refID);

                        ps.executeUpdate();
                                                }
                                                ps =
(PreparedStatement)conn.prepareStatement("INSERT INTO reference
("+(isUpdate!=null&&isUpdate?"ref_id,":"")+"ref_name,ref_link,ref_path,d
ate_created," +
                                                "creator,proj_id,title,author,publisher,pub_date,type,book_title,
volume,issue,page_num,journ_title,chap_title,editor,pub_place,article_name
,year,school) " +

                                                "VALUES
("+(isUpdate!=null&&isUpdate?file.refID+",":"")+"?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?)");
                                                                        ps.setString(1,
file.refName);
                                                                        ps.setString(2,
file.refLink);
                                                                        ps.setString(3,
file.refPath);
                                                                        ps.setString(4,
(isUpdate!=null&&isUpdate?file.dateCreated.toString():file.getTimestamp()
.toString()));
                                                                        ps.setString(5,
file.creator.username);
                                                                        ps.setInt(6,
file.project.projID);
                                                                        ps.setString(7, file.title);
                                                                        ps.setString(8,
file.author);
                                                                        ps.setString(9,
file.publisher);
                                                                        ps.setString(10,
file.pubDate);
                                                                        ps.setString(11,
file.type.toString());
                                                                        ps.setString(12,
file.bookTitle);
                                                                        ps.setString(13,
file.volume);
                                                                        ps.setString(14,
file.issue);
                                                                        ps.setString(15,
file.pageNum);
                                                                        ps.setString(16,
file.journTitle);
                                                                        ps.setString(17,
file.chapTitle);
                                                                        ps.setString(18,
file.editor);
                                                                        ps.setString(19,
file.pubPlace);
                                                                        ps.setString(20,
file.articleName);
                                                                        ps.setString(21,
file.year);
                                                                        ps.setString(22,
file.school);

                                                                        ps.executeUpdate();
                                                                        stmt.close();
                                                                        ps.close();
                                                                        conn.close();
                                                                        return "ok";
                                                        } catch (SQLException e) {
                                                                        e.printStackTrace();
                                                                        return "Error adding
link";
                                                        }
                                        }else{
                                                        return "Reference already exists in
current location";
                                        }
                        }

                        private Boolean checkLinkIfDuplicate(Reference file, Boolean
isUpdate){
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                        stmt = conn.createStatement();
                                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference where
ref_path =? and ref_name=?");
                                                        ps.setString(1, file.refPath);
```

```
                                        ps.setString(2, file.refName);
                                        ResultSet rset = ps.executeQuery();
                                        if(rset.next()){
                                                if(isUpdate){

        if(!file.refID.toString().equals(rset.getString("ref_id"))){

        stmt.close();

        rset.close();

        ps.close();

        conn.close();

        return true;
                                                }
                                                }else{
                                                        stmt.close();
                                                        rset.close();
                                                        ps.close();

        conn.close();
                                                        return true;
                                                }
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return false;
                        } catch (SQLException e) {
                                        e.printStackTrace();
                                        return false;
                        }
        }

        public ReferenceFolder getFolder(Integer folderID){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference_folder
where ref_folder_id=?");
                                        ps.setInt(1, folderID);
                                        ResultSet rset = ps.executeQuery();
                                        if(rset.next()){
                                                ReferenceFolder folder =
new ReferenceFolder();

        folder.setRefFolderID(folderID);

        folder.setRefFolderName(rset.getString("ref_folder_name"));

        folder.setRefFolderDesc(rset.getString("ref_folder_desc"));

        folder.setRefFolderPath(rset.getString("ref_folder_path"));

        folder.setDateCreated(java.sql.Timestamp.valueOf(rset.getStrin
g("date_created")));
                                                User user = new User();
                                                user.username =
rset.getString("creator");
                                                folder.creator = user;
                                                Project proj = new
Project();
                                                proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                                folder.setProject(proj);
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return folder;
                                        }
                                        return null;
                        } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                        }
        }

        public String saveDirectory(ReferenceFolder folder, Boolean
isUpdate){
                        String root = getServletContext().getRealPath("/")
+ "/uploads/ref/"+folder.project.projID+"/";

        if(isUpdate==null||!checkIfDuplicate(folder,(isUpdate?true:fals
e))){
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                stmt =
conn.createStatement();
                                                PreparedStatement ps;

        if(isUpdate!=null&&isUpdate){

        folder.dateCreated = getFolder(folder.refFolderID).dateCreated;

        folder.refFolderPath =
getFolder(folder.refFolderID).refFolderPath;

        folder.creator = getFolder(folder.refFolderID).creator;
                                                ps =
(PreparedStatement) conn.prepareStatement("delete from reference_folder
where ref_folder_id=?");
                                                ps.setInt(1,
folder.refFolderID);

        ps.executeUpdate();
                                        }
                                        ps = (PreparedStatement)
conn.prepareStatement("insert into reference_folder
("+(isUpdate!=null&&isUpdate?"ref_folder_id,":"")+"ref_folder_name,ref_f
older_desc,ref_folder_path,date_created,creator,proj_id) " +

        "values
("+(isUpdate!=null&&isUpdate?folder.refFolderID+",":"")+"?,?,?,?,?,?)");
                                        ps.setString(1,
folder.refFolderName);
                                        ps.setString(2,
folder.refFolderDesc);
                                        ps.setString(3,
folder.refFolderPath);
                                        ps.setString(4,
(isUpdate!=null&&isUpdate?folder.dateCreated.toString():folder.getTimesta
mp().toString()));
                                        ps.setString(5,
folder.creator.username);
                                        ps.setInt(6,
folder.project.projID);
                                        ps.executeUpdate();
                                        stmt.close();
                                        File path = new File(root
+ folder.refFolderPath + getReferenceFolderID(folder));
                                        if (!path.exists()) {
                                                boolean
status = path.mkdirs();
                                        }
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        return "ok";
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return "Error adding
file";
                                }
                        }else{
                                        return "Folder already exists in
current location";
                        }
        }

        private Boolean checkIfDuplicate(ReferenceFolder folder,
Boolean isUpdate){
```

```
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference_folder
where ref_folder_path =? and ref_folder_name=?");
                        ps.setString(1, folder.refFolderPath);
                        ps.setString(2, folder.refFolderName);
                        ResultSet rset = ps.executeQuery();
                        if(rset.next()){
                                if(isUpdate){

        if(!folder.refFolderID.toString().equals(rset.getString("ref_fold
er_id"))){

        stmt.close();

        rset.close();

        ps.close();

        conn.close();

        return true;
                                                }
                                        }else{
                                                stmt.close();
                                                rset.close();
                                                ps.close();

        conn.close();
                                                return true;
                                        }
                        }
                        stmt.close();
                        rset.close();
                        ps.close();
                        conn.close();
                        return false;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return false;
                }
        }

        private Integer getReferenceFolderID(ReferenceFolder folder){
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference_folder
where creator=? and ref_folder_path=? and ref_folder_name=?");
                        ps.setString(1,
folder.creator.username);
                        ps.setString(2, folder.refFolderPath);
                        ps.setString(3, folder.refFolderName);
                        ResultSet rset = ps.executeQuery();
                        if(rset.next()){
                                return
Integer.valueOf(rset.getString("ref_folder_id"));
                        }
                        stmt.close();
                        rset.close();
                        ps.close();
                        conn.close();
                        return null;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                }
        }

        public Boolean checkIfDirExists(String currLoc){
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        if(!currLoc.isEmpty()){
                                stmt =
conn.createStatement();
                                String[] parse =
currLoc.split("/");
                                String path = "";
                                for(int i = 0; i <
parse.length-1; i++){
                                        path +=
parse[i]+"/";
                                }
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference_folder
where ref_folder_path =? and ref_folder_id=?");
                                ps.setString(1, path);
                                ps.setInt(2,
Integer.valueOf(parse[parse.length-1]));
                                ResultSet rset =
ps.executeQuery();
                                if(rset.next()){
                                        return true;
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return false;
                        }else{
                                conn.close();
                                return true;
                        }
                } catch (SQLException e) {
                        e.printStackTrace();
                        return false;
                }
        }

        public ArrayList<ReferenceFolder> getFolders(Integer projID,
String currLoc, String query){
                ArrayList<ReferenceFolder> folders = new
ArrayList<ReferenceFolder>();
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference_folder
where ref_folder_path =? and proj_id=? and " +

                "(ref_folder_name like ? or ref_folder_desc like ?) order by
ref_folder_name asc");
                        ps.setString(1, currLoc);
                        ps.setInt(2, projID);
                        ps.setString(3, "%"+query+"%");
                        ps.setString(4, "%"+query+"%");
                        ResultSet rset = ps.executeQuery();
                        while(rset.next()){
                                ReferenceFolder folder =
new ReferenceFolder();
                                folder.refFolderID =
Integer.valueOf(rset.getString("ref_folder_id"));
                                folder.refFolderName =
rset.getString("ref_folder_name");
                                folder.refFolderDesc =
rset.getString("ref_folder_desc");
                                folder.refFolderPath =
rset.getString("ref_folder_path");
                                folder.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
                                User user = new User();
                                user.username =
rset.getString("creator");
                                folder.creator = user;
                                Project proj = new
Project();
                                proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                folder.setProject(proj);
```

```
                                        folders.add(folder);
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return folders;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
        }

        public ArrayList<Reference> getFiles(Integer projID, String
currLoc, String query){
                        ArrayList<Reference> files = new
ArrayList<Reference>();
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from reference where
ref_path =? and proj_id=? and " +
                                        "(title like ? or author
like ? or chap_title like ? or article_name like ?) order by
title,chap_title,article_name asc");
                                ps.setString(1, currLoc);
                                ps.setInt(2, projID);
                                ps.setString(3, "%"+query+"%");
                                ps.setString(4, "%"+query+"%");
                                ps.setString(5, "%"+query+"%");
                                ps.setString(6, "%"+query+"%");
                                ResultSet rset = ps.executeQuery();
                                while(rset.next()){
                                        Reference file = new
Reference();

                file.setRefID(Integer.valueOf(rset.getString("ref_id")));

                file.setRefName(rset.getString("ref_name"));

                file.setRefLink(rset.getString("ref_link"));

                file.setRefPath(rset.getString("ref_path"));

                file.setTitle(rset.getString("title"));

                file.setBookTitle(rset.getString("book_title"));

                file.setChapTitle(rset.getString("chap_title"));

                file.setJournTitle(rset.getString("journ_title"));

                file.setAuthor(rset.getString("author"));

                file.setEditor(rset.getString("editor"));

                file.setPublisher(rset.getString("publisher"));

                file.setPubDate(rset.getString("pub_date"));

                file.setPubPlace(rset.getString("pub_place"));

                file.setType(ReferenceType.valueOf(rset.getString("type").repl
ace(" ", "_")));

                file.setVolume(rset.getString("volume"));

                file.setIssue(rset.getString("issue"));

                file.setPageNum(rset.getString("page_num"));

                file.setArticleName(rset.getString("article_name"));

                file.setYear(rset.getString("year"));

                file.setSchool(rset.getString("school"));

                file.setDateCreated(java.sql.Timestamp.valueOf(rset.getString(
"date_created")));
                                        User user = new User();
                                        user.username =
rset.getString("creator");
                                        file.setCreator(user);
                                        Project proj = new
Project();
                                        proj.projID =
Integer.valueOf(rset.getString("proj_id"));
                                        file.project = proj;
                                        files.add(file);
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return files;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
        }

        public List<String> getFolderNamesFromIds(List<String>
ids){
                        List<String> names = new ArrayList<String>();
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                for(int i = 0; i < ids.size(); i++){
                                        if(!ids.get(i).isEmpty()){
                                                ResultSet
rset =

        stmt.executeQuery("select ref_folder_name from
reference_folder where ref_folder_id="+ids.get(i));

        if(rset.next()){

        names.add(rset.getString("ref_folder_name"));
                                                }else{

        names.add("");
                                                }
                                        }else{

        names.add("");
                                        }
                                }
                                stmt.close();
                                conn.close();
                                return names;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
        }

        public void deleteReferenceFolder(Integer refFolderID, Integer
projID, String currLoc){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                File path = new
File(getServletContext().getRealPath("/") + "/uploads/ref/" + projID + "/" +
currLoc + refFolderID +"/");
                                deleteDir(path);
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
reference_folder WHERE ref_folder_path LIKE ? or ref_folder_path LIKE
?");
                                ps.setString(1, refFolderID+"/%");
                                ps.setString(2,
"%/"+refFolderID+"/%");
```

```
                                        ps.executeUpdate();

                                        ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM reference
WHERE ref_path LIKE ? or ref_path LIKE ?");
                                        ps.setString(1, refFolderID+"/%");
                                        ps.setString(2,
"%/"+refFolderID+"/%");
                                        ps.executeUpdate();

                                        ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
reference_folder WHERE ref_folder_id=?");
                                        ps.setInt(1, refFolderID);
                                        ps.executeUpdate();
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }

                        public void deleteProjectReferences(Integer projID){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        File path = new
File(getServletContext().getRealPath("/") + "/uploads/ref/" + projID + "/");
                                        deleteDir(path);
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
reference_folder WHERE proj_id=?");
                                        ps.setString(1, projID.toString());
                                        ps.executeUpdate();

                                        ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM reference
WHERE proj_id=?");
                                        ps.setString(1, projID.toString());
                                        ps.executeUpdate();
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }

                        public void deleteProjectReferences(Integer projID, String
root){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        File path = new File(root +
"/uploads/ref/" + projID + "/");
                                        deleteDir(path);
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM
reference_folder WHERE proj_id=?");
                                        ps.setString(1, projID.toString());
                                        ps.executeUpdate();

                                        ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM reference
WHERE proj_id=?");
                                        ps.setString(1, projID.toString());
                                        ps.executeUpdate();
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }

                        public void deleteReference(Integer fileID, Integer projID,
String currLoc){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        File path = new
File(getServletContext().getRealPath("/") + "/uploads/ref/" + projID + "/"
+currLoc +getReference(fileID).refName);
                                        path.delete();
                                        int query =
stmt.executeUpdate("DELETE FROM reference WHERE ref_id="+fileID);
                                        stmt.close();
                                        conn.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }

                        public static boolean deleteDir(File dir) {
                           if (dir.isDirectory()) {
                             String[] children = dir.list();
                             for (int i = 0; i < children.length; i++) {
                               boolean success = deleteDir(new File(dir, children[i]));
                               if (!success) {
                                 return false;
                               }
                             }
                           }
                           return dir.delete();
                        }
}
```

SessionMngt.java

```
package cepir.server;

import javax.servlet.http.HttpSession;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

public class SessionMngt extends RemoteServiceServlet{
                        public void storeSessionData(String key, String value){
                                HttpSession session = getSession();
                                session.setAttribute(key, value);
                        }

                        public String getSessionKeyData(String key) {
                                HttpSession session = getSession();
                                if(session.getAttribute("servContext")==null){
                                        storeSessionData("servContext",
getServletContext().getRealPath("/"));
                                }
                                if (session.getAttribute(key) != null) {
                                        return (String)
session.getAttribute(key);
                                }
                                return null;
                        }

                        public void closeSession(){
                                HttpSession session = getSession();
                                session.invalidate();
                        }

                        private HttpSession getSession() {
                                return
this.getThreadLocalRequest().getSession(true);
                        }
}
```

ToolServiceImpl.java

```
package cepir.server;

import java.io.File;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
```

```
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;

import com.google.gwt.core.client.GWT;
import com.mysql.jdbc.PreparedStatement;

import cepir.client.ToolService;
import cepir.shared.Tool;
import cepir.shared.ToolFolder;
import cepir.shared.User;

public class ToolServiceImpl extends SessionMngt implements
ToolService{

	public String saveDirectory(ToolFolder folder, Boolean
isUpdate){
		String root = getServletContext().getRealPath("/")
+ "/uploads/tool/";

	if(isUpdate==null||!checkIfDuplicate(folder,(isUpdate?true:fals
e))){
			Connection conn =
DBConnection.getConnection();
			Statement stmt;
			try {
				stmt =
conn.createStatement();
//				int query;
				PreparedStatement ps;


	if(isUpdate!=null&&isUpdate){

	folder.dateCreated =
getFolder(folder.toolFolderID).dateCreated;

	folder.toolFolderPath =
getFolder(folder.toolFolderID).toolFolderPath;

	folder.creator = getFolder(folder.toolFolderID).creator;
					ps =
(PreparedStatement) conn.prepareStatement("delete from tool_folder where
tool_folder_id=?");
					ps.setInt(1,
folder.toolFolderID);

	ps.executeUpdate();
//					query =
stmt.executeUpdate("DELETE FROM tool_folder WHERE
tool_folder_id="+folder.toolFolderID);
				}
				ps = (PreparedStatement)
conn.prepareStatement("insert into tool_folder
("+(isUpdate!=null&&isUpdate?"tool_folder_id,":"")+"tool_folder_name,to
ol_folder_desc,tool_folder_path,date_created,creator) " +

	"values
("+(isUpdate!=null&&isUpdate?folder.toolFolderID+",":"")+"?,?,?,?,?)");
					ps.setString(1,
folder.toolFolderName);
					ps.setString(2,
folder.toolFolderDesc);
					ps.setString(3,
folder.toolFolderPath);
					ps.setString(4,
(isUpdate!=null&&isUpdate?folder.dateCreated.toString():folder.getTimesta
mp().toString()));
					ps.setString(5,
folder.creator.username);
					ps.executeUpdate();
//					query =
stmt.executeUpdate("INSERT INTO tool_folder
("+(isUpdate!=null&&isUpdate?"tool_folder_id,":"")+"tool_folder_name,to
ol_folder_desc,tool_folder_path,date_created,creator) " +
//
	"VALUES
("+(isUpdate!=null&&isUpdate?folder.toolFolderID+",":"")+"'"+folder.tool
FolderName+"','"+folder.toolFolderDesc+"','"+folder.toolFolderPath+"','"+(i
sUpdate!=null&&isUpdate?folder.dateCreated:folder.getTimestamp().toStri
ng())+"','"+folder.creator.username+"')");
					stmt.close();
					ps.close();
					conn.close();
					File path = new File(root
+ folder.toolFolderPath + getToolFolderID(folder));
					if (!path.exists()) {
						boolean
status = path.mkdirs();
					}
					return "ok";
				} catch (SQLException e) {
					e.printStackTrace();
					return "Error adding
file";
				}
			}else{
				return "Folder already exists in
current location";
			}
		}

		public String saveTool(Tool tool, Boolean isUpdate){

		if(isUpdate==null||!checkLinkIfDuplicate(tool,(isUpdate?true:f
alse))){
			Connection conn =
DBConnection.getConnection();
			Statement stmt;
			try {
				stmt =
conn.createStatement();
//				int query;
				PreparedStatement ps;


		if(isUpdate!=null&&isUpdate){

		tool.dateCreated = getFile(tool.toolID).dateCreated;
					tool.toolPath
= getFile(tool.toolID).toolPath;
					tool.creator
= getFile(tool.toolID).creator;
					ps =
(PreparedStatement) conn.prepareStatement("delete from tool where
tool_id=?");
					ps.setInt(1,
tool.toolID);

		ps.executeUpdate();
//					query =
stmt.executeUpdate("DELETE FROM tool WHERE tool_id="+tool.toolID);
				}
				ps =
(PreparedStatement)conn.prepareStatement("INSERT INTO tool
("+(isUpdate!=null&&isUpdate?"tool_id,":"")+"tool_name,tool_desc,tool_li
nk,tool_path,date_created,creator,type) " +

		"VALUES
("+(isUpdate!=null&&isUpdate?tool.toolID+",":"")+"?,?,?,?,?,?,?)");
					ps.setString(1,
tool.toolName);
					ps.setString(2,
tool.toolDesc);
					ps.setString(3,
tool.toolLink);
					ps.setString(4,
tool.toolPath);
					ps.setString(5,
(isUpdate!=null&&isUpdate?tool.dateCreated.toString():tool.getTimestamp(
).toString()));
					ps.setString(6,
tool.creator.username);
					ps.setInt(7, tool.type);
					ps.executeUpdate();
//					query =
stmt.executeUpdate("INSERT INTO tool
("+(isUpdate!=null&&isUpdate?"tool_id,":"")+"tool_name,tool_desc,tool_li
nk,tool_path,date_created,creator,type) " +
```

```
//
		"VALUES
("+(isUpdate!=null&&isUpdate?tool.toolID+",":"")+"'"+tool.toolName+"','"
+tool.toolDesc+"','"+tool.toolLink+"','"+tool.toolPath+"','"+
//
		(isUpdate!=null&&isUpdate?tool.dateCreated:tool.getTimesta
mp().toString())+"','"+tool.creator.username+"','"+tool.type+"')");
                                                stmt.close();
                                                ps.close();
                                                conn.close();
                                                return "ok";
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                                return "Error adding
link";
                                        }
                                }else{
                                        if(tool.type==1){
                                                return "Filename already
exists in current location";
                                        }else{
                                                return "Title already
exists in current location";
                                        }
                                }
                        }

                        private Integer getToolFolderID(ToolFolder folder){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool_folder where
creator=? and tool_folder_path=? and tool_folder_name=?");
                                        ps.setString(1,
folder.creator.username);
                                        ps.setString(2, folder.toolFolderPath);
                                        ps.setString(3,
folder.toolFolderName);
                                        ResultSet rset = ps.executeQuery();
//
                        stmt.executeQuery("select * from tool_folder where creator =
'"+folder.creator.username+
//
                        "' and tool_folder_path = '"+folder.toolFolderPath+"' and
tool_folder_name = '"+folder.toolFolderName+"'");
                                        if(rset.next()){
                                                return
Integer.valueOf(rset.getString("tool_folder_id"));
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return null;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                                }
                        }

                        private Boolean checkLinkIfDuplicate(Tool tool, Boolean
isUpdate){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool where
tool_path =? and tool_name=?");
                                        ps.setString(1, tool.toolPath);
                                        ps.setString(2, tool.toolName);
                                        ResultSet rset = ps.executeQuery();
//                                      ResultSet rset =
//
                        stmt.executeQuery("select * from tool where tool_path =
'"+tool.toolPath+"' and tool_name = '"+tool.toolName+"'");
                                        if(rset.next()){
                                                if(isUpdate){

                        if(!tool.toolID.toString().equals(rset.getString("tool_id"))){

                        stmt.close();

                        rset.close();

                        ps.close();

                        conn.close();

                        return true;
                                                }else{
                                                        stmt.close();
                                                        rset.close();
                                                        ps.close();

                        conn.close();
                                                        return true;
                                                }
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return false;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return false;
                                }
                        }

                        private Boolean checkIfDuplicate(ToolFolder folder, Boolean
isUpdate){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool_folder where
tool_folder_path =? and tool_folder_name=?");
                                        ps.setString(1, folder.toolFolderPath);
                                        ps.setString(2,
folder.toolFolderName);
                                        ResultSet rset = ps.executeQuery();
//                                      ResultSet rset =
//
                        stmt.executeQuery("select * from tool_folder where
tool_folder_path = '"+folder.toolFolderPath+"' and tool_folder_name =
'"+folder.toolFolderName+"'");
                                        if(rset.next()){
                                                if(isUpdate){

                        if(!folder.toolFolderID.toString().equals(rset.getString("tool_fol
der_id"))){

                        stmt.close();

                        rset.close();

                        ps.close();

                        conn.close();

                        return true;
                                                }else{
                                                        stmt.close();
                                                        rset.close();
                                                        ps.close();

                        conn.close();
                                                        return true;
                                                }
                                        }
                                        stmt.close();
```

```
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return false;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return false;
                                }
                        }

                        public Boolean checkIfDirExists(String currLoc){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        if(!currLoc.isEmpty()){
                                                stmt =
conn.createStatement();
                                                String[] parse =
currLoc.split("/");
                                                String path = "";
                                                for(int i = 0; i <
parse.length-1; i++){
                                                        path +=
parse[i]+"/";
                                                }
                                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool_folder where
tool_folder_path =? and tool_folder_id=?");
                                                ps.setString(1, path);
                                                ps.setInt(2,
Integer.valueOf(parse[parse.length-1]));
                                                ResultSet rset =
ps.executeQuery();
//                                              ResultSet rset =
//
                        stmt.executeQuery("select * from tool_folder where
tool_folder_path = '"+path+"' and tool_folder_id="+parse[parse.length-1]);
                                                if(rset.next()){
                                                        stmt.close();
                                                        rset.close();
                                                        ps.close();


                        conn.close();
                                                        return true;
                                                }
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return false;
                                        }else{
                                                conn.close();
                                                return true;
                                        }
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return false;
                                }
                        }

                        public ToolFolder getFolder(Integer toolFolderID){
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool_folder where
tool_folder_id=?");
                                        ps.setInt(1, toolFolderID);
                                        ResultSet rset = ps.executeQuery();
//                                      ResultSet rset =
//
                        stmt.executeQuery("select * from tool_folder where
tool_folder_id = "+toolFolderID);
                                        if(rset.next()){
                                                ToolFolder folder = new
ToolFolder();
                                                folder.toolFolderID =
Integer.valueOf(rset.getString("tool_folder_id"));
                                                folder.toolFolderName =
rset.getString("tool_folder_name");
                                                folder.toolFolderDesc =
rset.getString("tool_folder_desc");
                                                folder.toolFolderPath =
rset.getString("tool_folder_path");
                                                folder.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
                                                User user = new User();
                                                user.username =
rset.getString("creator");
                                                folder.creator = user;
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return folder;
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
                                        conn.close();
                                        return null;
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return null;
                                }
                        }

                        // currLoc = /<parent folder>/...
                        public ArrayList<ToolFolder> getFolders(String currLoc,
String query){
                                ArrayList<ToolFolder> folders = new
ArrayList<ToolFolder>();
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt = conn.createStatement();
                                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool_folder where
tool_folder_path =? and " +

                        "(tool_folder_name like ? or tool_folder_desc like ?) order by
tool_folder_name asc");
                                        ps.setString(1, currLoc);
                                        ps.setString(2, "%"+query+"%");
                                        ps.setString(3, "%"+query+"%");
                                        ResultSet rset = ps.executeQuery();
//                                      ResultSet rset =
//
                        stmt.executeQuery("select * from tool_folder where
tool_folder_path = '"+currLoc+"' and (tool_folder_name like '%"
//
                        +query+"%' or tool_folder_desc like '%"+query+"%') order by
tool_folder_name asc");
                                        while(rset.next()){
                                                ToolFolder folder = new
ToolFolder();
                                                folder.toolFolderID =
Integer.valueOf(rset.getString("tool_folder_id"));
                                                folder.toolFolderName =
rset.getString("tool_folder_name");
                                                folder.toolFolderDesc =
rset.getString("tool_folder_desc");
                                                folder.toolFolderPath =
rset.getString("tool_folder_path");
                                                folder.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
                                                User user = new User();
                                                user.username =
rset.getString("creator");
                                                folder.creator = user;
                                                folders.add(folder);
                                        }
                                        stmt.close();
                                        rset.close();
                                        ps.close();
```

```
                        conn.close();
                        return folders;
                    } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                    }
                }

                public Tool getFile(Integer toolID){
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool where
tool_id=?");
                        ps.setInt(1, toolID);
                        ResultSet rset = ps.executeQuery();
//                      ResultSet rset =
//
                stmt.executeQuery("select * from tool where tool_id =
"+toolID);
                        if(rset.next()){
                            Tool tool = new Tool();
                            tool.toolID = toolID;
                            tool.toolName =
rset.getString("tool_name");
                            tool.toolDesc =
rset.getString("tool_desc");
                            tool.toolLink =
rset.getString("tool_link");
                            tool.toolPath =
rset.getString("tool_path");
                            tool.type =
Integer.valueOf(rset.getString("type"));
                            tool.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
                            User user = new User();
                            user.username =
rset.getString("creator");
                            tool.creator = user;
                            stmt.close();
                            rset.close();
                            ps.close();
                            conn.close();
                            return tool;
                        }
                        stmt.close();
                        rset.close();
                        ps.close();
                        conn.close();
                        return null;
                    } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                    }
                }

                public ArrayList<Tool> getFiles(String currLoc, String query){
                    ArrayList<Tool> tools = new ArrayList<Tool>();
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from tool where
tool_path =? and " +
                                "(tool_name like ? or
tool_desc like ?) order by tool_name asc");
                        ps.setString(1, currLoc);
                        ps.setString(2, "%"+query+"%");
                        ps.setString(3, "%"+query+"%");
                        ResultSet rset = ps.executeQuery();
//                      ResultSet rset =
//
                stmt.executeQuery("select * from tool where tool_path =
"'+currLoc+'" and (tool_name like '%"
//
                        +query+"%' or tool_desc like '%"+query+"%') order by
tool_name asc");
                        while(rset.next()){
                            Tool tool = new Tool();
                            tool.toolID =
Integer.valueOf(rset.getString("tool_id"));
                            tool.toolName =
rset.getString("tool_name");
                            tool.toolDesc =
rset.getString("tool_desc");
                            tool.toolPath =
rset.getString("tool_path");
                            tool.toolLink =
rset.getString("tool_link");
                            tool.type =
Integer.valueOf(rset.getString("type"));
                            tool.dateCreated =
java.sql.Timestamp.valueOf(rset.getString("date_created"));
                            User user = new User();
                            user.username =
rset.getString("creator");
                            tool.creator = user;
                            tools.add(tool);
                        }
                        stmt.close();
                        rset.close();
                        ps.close();
                        conn.close();
                        return tools;
                    } catch (SQLException e) {
                        e.printStackTrace();
                        return null;
                    }
                }

                public void deleteToolFolder(Integer toolFolderID, String
currLoc){
                    Connection conn =
DBConnection.getConnection();
                    Statement stmt;
                    try {
                        stmt = conn.createStatement();
                        File path = new
File(getServletContext().getRealPath("/") + "/uploads/tool/" + currLoc +
toolFolderID +"/");
                        deleteDir(path);
                        PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM tool_folder
WHERE tool_folder_path LIKE ? or tool_folder_path LIKE ?");
                        ps.setString(1, toolFolderID+"/%");
                        ps.setString(2,
"%/"+toolFolderID+"/%");
                        ps.executeUpdate();

                        ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM tool WHERE
tool_path LIKE ? or tool_path LIKE ?");
                        ps.setString(1, toolFolderID+"/%");
                        ps.setString(2,
"%/"+toolFolderID+"/%");
                        ps.executeUpdate();

                        ps =
(PreparedStatement)conn.prepareStatement("DELETE FROM tool_folder
WHERE tool_folder_id=?");
                        ps.setInt(1, toolFolderID);
                        ps.executeUpdate();
//                      int query =
stmt.executeUpdate("DELETE FROM tool_folder WHERE
tool_folder_path LIKE '"+toolFolderID+"/%' OR tool_folder_path LIKE
'%/"+toolFolderID+"/%'");
//                      query =
stmt.executeUpdate("DELETE FROM tool WHERE tool_path LIKE
'"+toolFolderID+"/%' OR tool_path LIKE '%/"+toolFolderID+"/%'");
//                      query =
stmt.executeUpdate("DELETE FROM tool_folder WHERE
tool_folder_id="+toolFolderID);
                        stmt.close();
                        ps.close();
```

```
                        conn.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

        public void deleteTool(Integer toolID, String currLoc){
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        File path = new
File(getServletContext().getRealPath("/") + "/uploads/tool/" + currLoc
+getFile(toolID).toolName);
                        path.delete();
                        int query =
stmt.executeUpdate("DELETE FROM tool WHERE tool_id="+toolID);
                        stmt.close();
                        conn.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

        public static boolean deleteDir(File dir) {
           if (dir.isDirectory()) {
             String[] children = dir.list();
             for (int i = 0; i < children.length; i++) {
               boolean success = deleteDir(new File(dir, children[i]));
               if (!success) {
                 return false;
               }
             }
           }
           return dir.delete();
        }

        private ArrayList<Integer> getFiles(String path){
                ArrayList<Integer> tools = new
ArrayList<Integer>();
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from tool_folder
where tool_folder_path=?");
                        ps.setString(1, path);
                        ResultSet rset = ps.executeQuery();
//
        stmt.executeQuery("select * from tool_folder where
tool_folder_path = '"+path+"'");
                        while(rset.next()){

        tools.add(Integer.valueOf(rset.getString("tool_folder_id")));
                        }
                        stmt.close();
                        rset.close();
                        ps.close();
                        conn.close();
                        return tools;
                } catch (SQLException e) {
                        e.printStackTrace();
                        return tools;
                }
        }

        public List<String> getFolderNamesFromIds(List<String>
ids){
                List<String> names = new ArrayList<String>();
                Connection conn =
DBConnection.getConnection();
                Statement stmt;
                try {
                        stmt = conn.createStatement();
                        for(int i = 0; i < ids.size(); i++){
                                if(!ids.get(i).isEmpty()){
                                                ResultSet
rset =

        stmt.executeQuery("select tool_folder_name from tool_folder
where tool_folder_id="+ids.get(i));

        if(rset.next()){

        names.add(rset.getString("tool_folder_name"));
                                                }else{

        names.add("");
                                                }
                                        }else{

        names.add("");
                                        }
                                }
                                stmt.close();
                                conn.close();
                                return names;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return null;
                        }
        }
}
```

UploadFileService.java

```
package cepir.server;

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
import com.mysql.jdbc.PreparedStatement;

@RemoteServiceRelativePath("upload")
public class UploadFileService extends HttpServlet{

        protected void doPost(HttpServletRequest req,
HttpServletResponse resp)
        throws ServletException, IOException {
                if (ServletFileUpload.isMultipartContent(req)) {
                        FileItemFactory factory = new
DiskFileItemFactory();
                        ServletFileUpload upload = new
ServletFileUpload(factory);
                        try {
                                String location = "";
                                String name = "";
                                List<FileItem> items =
upload.parseRequest(req);
                                for(FileItem item:
items){

        if(item.isFormField()){

        if(item.getFieldName().equalsIgnoreCase("currLoc")){
//currLoc should contain uploads/<type>/<projID>/...

        location = item.getString();

        }else if(item.getFieldName().equalsIgnoreCase("name")){

        name = item.getString();
```

```
		}
                                                        }else{

		File directory = new
File(getServletContext().getRealPath("/")+"/"+location);

		if(!directory.exists()){

		directory.mkdirs();

		}

		File uploadedFile = new
File(getServletContext().getRealPath("/")+"/"+location+name);

		if (uploadedFile.createNewFile()) {

		item.write(uploadedFile);

		resp.setStatus(HttpServletResponse.SC_CREATED);

		resp.getWriter().print("File was uploaded successfully");

		resp.flushBuffer();

		} else{

		resp.getWriter().print("The file already exists");

		}
                                                                }
                                                        }
                                        } catch (FileUploadException e) {
                                                e.printStackTrace();
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                        }
                                } else {

		resp.sendError(HttpServletResponse.SC_UNSUPPORTED_M
EDIA_TYPE,
                                        "Request contents type is not
supported by the servlet.");
                                }
		}
}
```

UserServiceImpl.java

```
package cepir.server;


import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashMap;

import javax.servlet.http.HttpSession;

import cepir.client.ProjectService;
import cepir.client.ProjectServiceAsync;
import cepir.client.UserService;
import cepir.server.DBConnection;
import cepir.shared.AcctRequest;
import cepir.shared.User;
import cepir.shared.YesNo;

import com.google.gwt.core.client.GWT;
import com.google.gwt.user.server.rpc.RemoteServiceServlet;
import com.mysql.jdbc.PreparedStatement;

public class UserServiceImpl extends SessionMngt implements
UserService{

		public Boolean checkIfUser(String username, String
password){ //ok
                        Connection conn =
DBConnection.getConnection();
                        try {
                                PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT * FROM user where
username=?");
                                ps.setString(1, username);
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        String hash =
rset.getString("password");

		if(BCrypt.checkpw(password, hash)){

		storeSessionData("username", username);

		storeSessionData("isSysAd", rset.getString("sysAd"));

		storeSessionData("isToolAd", rset.getString("toolAd"));
                                                                rset.close();
                                                                ps.close();

		conn.close();
                                                                return true;
                                        }
                                }
                                rset.close();
                                ps.close();
                                conn.close();
                                return false; //username/password
does not exist/match
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
		}

		public ArrayList<User> getUsersList(){
                        return getUsersList("username","asc");
		}

		/*
		 *		order must be ASC/DESC only
		 */
		public ArrayList<User> getUsersList(String colToSort, String
order){
                        if(getSessionKeyData("isSysAd")!=null){
                                ArrayList<User> users = new
ArrayList<User>();
                                Connection conn =
DBConnection.getConnection();
                                Statement stmt;
                                try {
                                        stmt =
conn.createStatement();
                                        ResultSet rset =

		stmt.executeQuery("select * from user
"+(colToSort.isEmpty()||order.isEmpty()?"":"order by "+colToSort+"
"+order));
                                        while(rset.next()){
                                                User user =
new User();

		user.username = rset.getString("username");

		user.password = rset.getString("password");
                                                                user.email =
rset.getString("email");

		user.lastName = rset.getString("last_name");

		user.firstName = rset.getString("first_name");

		user.middleName = rset.getString("middle_name");

		user.affiliation = rset.getString("affiliation");
                                                                user.sysAd
= YesNo.valueOf(rset.getString("sysAd"));
                                                                user.toolAd
= YesNo.valueOf(rset.getString("toolAd"));
```

```
		user.question = rset.getString("security_question");
                                                        user.answer
= rset.getString("answer");

		users.add(user);
                                                }
                                                stmt.close();
                                                conn.close();
                                                return users;
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                                return null;
                                        }
                                }
                                return null;
		}

		public ArrayList<User> getUserSearchList(String query){

		if(getSessionKeyData("isSysAd").equalsIgnoreCase("yes")){
                                        ArrayList<User> users = new
ArrayList<User>();
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                stmt =
conn.createStatement();
                                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from user where
username like ? or first_name like ? or middle_name like ? "+

		"or last_name like ? or email like ? or affiliation like ? order by
username asc");
                                                ps.setString(1,
"%"+query+"%");
                                                ps.setString(2,
"%"+query+"%");
                                                ps.setString(3,
"%"+query+"%");
                                                ps.setString(4,
"%"+query+"%");
                                                ps.setString(5,
"%"+query+"%");
                                                ps.setString(6,
"%"+query+"%");
                                                ResultSet rset =
ps.executeQuery();
                                                while(rset.next()){
                                                        User user =
new User();

		user.username = rset.getString("username");

		user.password = rset.getString("password");
                                                        user.email =
rset.getString("email");

		user.lastName = rset.getString("last_name");

		user.firstName = rset.getString("first_name");

		user.middleName = rset.getString("middle_name");

		user.affiliation = rset.getString("affiliation");
                                                        user.sysAd
= YesNo.valueOf(rset.getString("sysAd"));
                                                        user.toolAd
= YesNo.valueOf(rset.getString("toolAd"));

		user.question = rset.getString("security_question");
                                                        user.answer
= rset.getString("answer");

		users.add(user);
                                                }
                                                stmt.close();
                                                conn.close();
                                                return users;
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                                return null;
                                        }
                                }
                                return null;
		}

		public User getUser(String username){
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                User user = new User();
                                                stmt = conn.createStatement();
                                                PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from user where
username=?");
                                                ps.setString(1, username);
                                                ResultSet rset = ps.executeQuery();
                                                while(rset.next()){
                                                        user.username =
rset.getString("username");
                                                        user.password =
rset.getString("password");
                                                        user.email =
rset.getString("email");
                                                        user.lastName =
rset.getString("last_name");
                                                        user.firstName =
rset.getString("first_name");
                                                        user.middleName =
rset.getString("middle_name");
                                                        user.affiliation =
rset.getString("affiliation");
                                                        user.sysAd =
YesNo.valueOf(rset.getString("sysAd"));
                                                        user.toolAd =
YesNo.valueOf(rset.getString("toolAd"));
                                                        user.question =
rset.getString("security_question");
                                                        user.answer =
rset.getString("answer");
                                                }
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return user;
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                                return null;
                                        }
                                }
                                return null;
		}

		public User getSecurityQuestion(String username){
                                        Connection conn =
DBConnection.getConnection();
                                        Statement stmt;
                                        try {
                                                User user = new User();
                                                stmt = conn.createStatement();
                                                PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from user where
username=?");
                                                ps.setString(1, username);
                                                ResultSet rset = ps.executeQuery();
                                                while(rset.next()){
                                                        user.username =
rset.getString("username");
                                                        user.question =
rset.getString("security_question");
                                                        user.answer =
rset.getString("answer");
                                                }
                                                stmt.close();
                                                rset.close();
                                                ps.close();
                                                conn.close();
                                                return user;
```

```
			} catch (SQLException e) {
				e.printStackTrace();
				return null;
			}
		}

		public void resetPassword(String username, String password){
			Connection conn =
DBConnection.getConnection();
			Statement stmt;
			try {
				stmt = conn.createStatement();
				PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("update user set password=?
where username=?");
				ps.setString(1,
BCrypt.hashpw(password, BCrypt.gensalt()));
				ps.setString(2, username);
				ps.executeUpdate();
				stmt.close();
				ps.close();
				conn.close();
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}

		public ArrayList<AcctRequest> getAcctRequests(){
			ArrayList<AcctRequest> users = new
ArrayList<AcctRequest>();
			Connection conn =
DBConnection.getConnection();
			Statement stmt;
			try {
				stmt = conn.createStatement();
				ResultSet rset =

		stmt.executeQuery("select * from acct_request order by
last_name asc");
				while(rset.next()){
					AcctRequest user = new
AcctRequest();
					user.username =
rset.getString("username");
					user.email =
rset.getString("email");
					user.lastName =
rset.getString("last_name");
					user.firstName =
rset.getString("first_name");
					user.middleName =
rset.getString("middle_name");
					user.affiliation =
rset.getString("affiliation");
					user.password =
rset.getString("password");
					user.question =
rset.getString("security_question");
					user.answer =
rset.getString("answer");
					users.add(user);
				}
				stmt.close();
				rset.close();
				conn.close();
				return users;
			} catch (SQLException e) {
				e.printStackTrace();
				return null;
			}
		}

		public ArrayList<AcctRequest>
getAcctRequestSearchList(String query){
			ArrayList<AcctRequest> users = new
ArrayList<AcctRequest>();
			Connection conn =
DBConnection.getConnection();
			Statement stmt;
			try {
				stmt = conn.createStatement();
				PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("select * from acct_request
where username like ? or first_name like ? " +
"or
middle_name like ? or last_name like ? or email like ? or affiliation like ?
order by last_name asc");
				ps.setString(1, "%"+query+"%");
				ps.setString(2, "%"+query+"%");
				ps.setString(3, "%"+query+"%");
				ps.setString(4, "%"+query+"%");
				ps.setString(5, "%"+query+"%");
				ps.setString(6, "%"+query+"%");
				ResultSet rset = ps.executeQuery();
				while(rset.next()){
					AcctRequest user = new
AcctRequest();
					user.username =
rset.getString("username");
					user.email =
rset.getString("email");
					user.lastName =
rset.getString("last_name");
					user.firstName =
rset.getString("first_name");
					user.middleName =
rset.getString("middle_name");
					user.affiliation =
rset.getString("affiliation");
					user.password =
rset.getString("password");
					user.question =
rset.getString("security_question");
					user.answer =
rset.getString("answer");
					users.add(user);
				}
				stmt.close();
				rset.close();
				ps.close();
				conn.close();
				return users;
			} catch (SQLException e) {
				e.printStackTrace();
				return null;
			}
		}

		public String saveRequestForAccount(AcctRequest acct){

		if(emailIsNotUsed(acct.email)&&usernameIsNotUsed(acct.user
name)){
			Connection conn =
DBConnection.getConnection();
			Statement stmt;
			try {
				stmt =
conn.createStatement();
				PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("insert into acct_request
values(?,?,?,?,?,?,?,?,?)");
				ps.setString(1,
acct.email);
				ps.setString(2,
acct.username);
				ps.setString(3,
BCrypt.hashpw(acct.password, BCrypt.gensalt()));
				ps.setString(4,
acct.lastName);
				ps.setString(5,
acct.firstName);
				ps.setString(6,
acct.middleName);
				ps.setString(7,
acct.affiliation);
				ps.setString(8,
acct.question);
				ps.setString(9,
acct.answer);
				ps.executeUpdate();
```

```
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        return "Request sent.";
                                }catch (SQLException e) {
                                        e.printStackTrace();
                                        return "Error sending
request";
                                }
                        }else{
                                return "";
                        }
                }

                /*
                 * changePass - true if password is to be changed
                */
                public String updateUser(User user, Boolean changePass){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                if(!changePass){
                                        user.password =
getUser(user.username).password;
                                }
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("delete from user where
username = ?");
                                ps.setString(1, user.username);
                                ps.executeUpdate();
//                              int query  =
//
                stmt.executeUpdate("delete from user where
username='"+user.username+"'");
                                if(emailIsNotUsed(user.email)){
                                        ps =
(PreparedStatement)conn.prepareStatement("insert into user values
(?,?,?,?,?,?,?,?,?,?,?)");
                                        ps.setString(1,
user.username);
                                        ps.setString(2,
(!changePass?user.password:BCrypt.hashpw(user.password,
BCrypt.gensalt())));
                                        ps.setString(3,
user.email);
                                        ps.setString(4,
user.lastName);
                                        ps.setString(5,
user.firstName);
                                        ps.setString(6,
user.middleName);
                                        ps.setString(7,
user.affiliation);
                                        ps.setString(8,
user.sysAd.toString());
                                        ps.setString(9,
user.toolAd.toString());
                                        ps.setString(10,
user.question);
                                        ps.setString(11,
user.answer);
                                        ps.executeUpdate();

                if(user.username.equals(getSessionKeyData("username"))){

                storeSessionData("isSysAd", user.sysAd.toString());
                                        }
//                                      query =
stmt.executeUpdate("insert into user
values('"+user.username+"','"+(!changePass?user.password:BCrypt.hashpw(
user.password,
BCrypt.gensalt()))+"','"+user.email+"','"+user.lastName+"','"+
//
                user.firstName+"','"+user.middleName+"','"+user.affiliation+"','
"+user.sysAd+"','"+user.toolAd+"')");
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        return "updated";
                                }else{
                                        return "Email already in
use";
                                }
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return "Error updating user";
                        }
                }

                public Boolean addUser(User user, Boolean isEncrypted){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("insert into user values
(?,?,?,?,?,?,?,?,?,?,?)");
                                ps.setString(1, user.username);
                                ps.setString(2,
(isEncrypted?user.password:BCrypt.hashpw(user.password,
BCrypt.gensalt())));
                                ps.setString(3, user.email);
                                ps.setString(4, user.lastName);
                                ps.setString(5, user.firstName);
                                ps.setString(6, user.middleName);
                                ps.setString(7, user.affiliation);
                                ps.setString(8, user.sysAd.toString());
                                ps.setString(9,
user.toolAd.toString());
                                ps.setString(10, user.question);
                                ps.setString(11, user.answer);
                                ps.executeUpdate();
//                              int query  =
//
                stmt.executeUpdate("insert into user
values('"+user.username+"','"+(isEncrypted?user.password:BCrypt.hashpw(
user.password,
BCrypt.gensalt()))+"','"+user.email+"','"+user.lastName+"','"+
//
                user.firstName+"','"+user.middleName+"','"+user.affiliation+"','
"+user.sysAd+"','"+user.toolAd+"')");
                                stmt.close();
                                ps.close();
                                conn.close();
                                return true;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                public String addUserWithChecking(User user, Boolean
isEncrypted){

                if(usernameIsNotUsed(user.getUsername())&&emailIsNotUsed
(user.getEmail())){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt =
conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("insert into user values
(?,?,?,?,?,?,?,?,?,?,?)");
                                ps.setString(1,
user.username);
                                ps.setString(2,
(isEncrypted?user.password:BCrypt.hashpw(user.password,
BCrypt.gensalt())));
                                ps.setString(3,
user.email);
                                ps.setString(4,
user.lastName);
                                ps.setString(5,
user.firstName);
```

```
                                        ps.setString(6,
user.middleName);
                                        ps.setString(7,
user.affiliation);
                                        ps.setString(8,
user.sysAd.toString());
                                        ps.setString(9,
user.toolAd.toString());
                                        ps.setString(10,
user.question);
                                        ps.setString(11,
user.answer);
                                        ps.executeUpdate();
//                                      int query  =
//
                stmt.executeUpdate("insert into user
values('"+user.username+"','"+(isEncrypted?user.password:BCrypt.hashpw(
user.password,
BCrypt.gensalt()))+"','"+user.email+"','"+user.lastName+"','"+
//
                user.firstName+"','"+user.middleName+"','"+user.affiliation+"','
"+user.sysAd+"','"+user.toolAd+"')");
                                        stmt.close();
                                        ps.close();
                                        conn.close();
                                        return "ok";
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                        return "Error adding
user";
                                }
                        }else{
                                return "";
                        }
                }

                /*
                 * Returns true if username has not been used by a user nor an
acct requester
                 */
                public Boolean usernameIsNotUsed(String username){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from user where
username = ?");
                                ps.setString(1, username);
                                ResultSet rset = ps.executeQuery();
//
                stmt.executeQuery("select * from user where username='"+
username +"'");
                                if(rset.next()){
                                        return false;
                                }
                                ps =
(PreparedStatement)conn.prepareStatement("select * from acct_request
where username = ?");
                                ps.setString(1, username);
                                rset = ps.executeQuery();
//                              rset = stmt.executeQuery("select *
from acct_request where username='"+ username+"'");
                                if(rset.next()){
                                        return false;
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return true;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                /*
                 * Returns true if email has not been used by a user nor an acct
requester
                 */
                public Boolean emailIsNotUsed(String email){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("select * from user where email
= ?");
                                ps.setString(1, email);
                                ResultSet rset = ps.executeQuery();
//                              ResultSet rset =
//
                stmt.executeQuery("select * from acct_request where email='"+
email+"'");
                                if(rset.next()){
                                        return false;
                                }
                                ps =
(PreparedStatement)conn.prepareStatement("select * from acct_request
where email = ?");
                                ps.setString(1, email);
                                rset = ps.executeQuery();
//                              rset = stmt.executeQuery("select *
from user where email='"+ email+"'");
                                if(rset.next()){
                                        return false;
                                }
                                stmt.close();
                                rset.close();
                                ps.close();
                                conn.close();
                                return true;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                public void deleteUser(String username){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                ProjectServiceImpl proj = new
ProjectServiceImpl();
                                proj.deleteUserPriv(username);
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("delete from user where
username=?");
                                ps.setString(1, username);
                                ps.executeUpdate();
//                              int query =
//
                stmt.executeUpdate("delete from user where username='"+
username+"'");
                                stmt.close();
                                ps.close();
                                conn.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }

                public void deleteRequest(String username){
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement)conn.prepareStatement("delete from acct_request where
username=?");
                                ps.setString(1, username);
                                ps.executeUpdate();
//                              int query =
```

```
//
		stmt.executeUpdate("delete from acct_request where
username='"+ username+"'");
					stmt.close();
					ps.close();
					conn.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
		}

		public Boolean checkPasswordMatch(String pass, String hash){
			return BCrypt.checkpw(pass, hash);
		}


}
```

AcctRequest.java

```
package cepir.shared;

import java.io.Serializable;


public class AcctRequest implements Serializable {
		public String email;
		public String username;
		public String password;
		public String lastName;
		public String firstName;
		public String middleName;
		public String affiliation;
		public String question;
		public String answer;

		public AcctRequest() {
		}

		public AcctRequest(String email, String username, String
password,String lastName, String firstName,
					String middleName, String affiliation,
String question, String answer) {
			this.email = email;
			this.username = username;
			this.password = password;
			this.lastName = lastName;
			this.firstName = firstName;
			this.middleName = middleName;
			this.affiliation = affiliation;
			this.question = question;
			this.answer = answer;
		}

		public String getEmail() {
			return email;
		}

		public void setEmail(String email) {
			this.email = email;
		}

		public String getUsername() {
			return username;
		}

		public void setUsername(String username) {
			this.username = username;
		}

		public String getPassword() {
			return password;
		}

		public void setPassword(String password) {
			this.password = password;
		}

		public String getLastName() {
			return lastName;
		}

		public void setLastName(String lastName) {
			this.lastName = lastName;
		}

		public String getFirstName() {
			return firstName;
		}

		public void setFirstName(String firstName) {
			this.firstName = firstName;
		}

		public String getMiddleName() {
			return middleName;
		}

		public void setMiddleName(String middleName) {
			this.middleName = middleName;
		}

		public String getAffiliation() {
			return affiliation;
		}

		public void setAffiliation(String affiliation) {
			this.affiliation = affiliation;
		}

		public String getQuestion() {
			return question;
		}

		public void setQuestion(String question) {
			this.question = question;
		}

		public String getAnswer() {
			return answer;
		}

		public void setAnswer(String answer) {
			this.answer = answer;
		}

}
```

Base64Coder.java

```
package cepir.shared;

//Copyright 2003-2010 Christian d'Heureuse, Inventec Informatik AG,
Zurich, Switzerland
//www.source-code.biz, www.inventec.ch/chdh
//
//This module is multi-licensed and may be used under the terms
//of any of the following licenses:
//
//EPL, Eclipse Public License, V1.0 or later, http://www.eclipse.org/legal
//LGPL, GNU Lesser General Public License, V2.1 or later,
http://www.gnu.org/licenses/lgpl.html
//GPL, GNU General Public License, V2 or later,
http://www.gnu.org/licenses/gpl.html
//AL, Apache License, V2.0 or later, http://www.apache.org/licenses
//BSD, BSD License, http://www.opensource.org/licenses/bsd-license.php
//
//Please contact the author if you need another license.
//This module is provided "as is", without warranties of any kind.

/**
* A Base64 encoder/decoder.
*
* <p>
* This class is used to encode and decode data in Base64 format as described
in RFC 1521.
*
* <p>
* Project home page: <a href="http://www.source-
code.biz/base64coder/java/">www.source-
code.biz/base64coder/java</a><br>
```

```
* Author: Christian d'Heureuse, Inventec Informatik AG, Zurich,
Switzerland<br>
* Multi-licensed: EPL / LGPL / GPL / AL / BSD.
*/
public class Base64Coder {

//The line separator string of the operating system.
//private static final String systemLineSeparator =
System.getProperty("line.separator");
                private static final String systemLineSeparator = "CR+LF";

//Mapping table from 6-bit nibbles to Base64 characters.
private static char[]    map1 = new char[64];
static {
   int i=0;
   for (char c='A'; c<='Z'; c++) map1[i++] = c;
   for (char c='a'; c<='z'; c++) map1[i++] = c;
   for (char c='0'; c<='9'; c++) map1[i++] = c;
   map1[i++] = '+'; map1[i++] = '/'; }

//Mapping table from Base64 characters to 6-bit nibbles.
private static byte[]    map2 = new byte[128];
static {
   for (int i=0; i<map2.length; i++) map2[i] = -1;
   for (int i=0; i<64; i++) map2[map1[i]] = (byte)i; }

/**
* Encodes a string into Base64 format.
* No blanks or line breaks are inserted.
* @param s  A String to be encoded.
* @return   A String containing the Base64 encoded data.
*/
public static String encodeString (String s) {
return new String(encode(s.getBytes())); }

/**
* Encodes a byte array into Base 64 format and breaks the output into lines
of 76 characters.
* This method is compatible with
<code>sun.misc.BASE64Encoder.encodeBuffer(byte[])</code>.
* @param in  An array containing the data bytes to be encoded.
* @return    A String containing the Base64 encoded data, broken into lines.
*/
public static String encodeLines (byte[] in) {
return encodeLines(in, 0, in.length, 76, systemLineSeparator); }

/**
* Encodes a byte array into Base 64 format and breaks the output into lines.
* @param in            An array containing the data bytes to be encoded.
* @param iOff          Offset of the first byte in <code>in</code> to be
processed.
* @param iLen          Number of bytes to be processed in <code>in</code>,
starting at <code>iOff</code>.
* @param lineLen       Line length for the output data. Should be a multiple
of 4.
* @param lineSeparator The line separator to be used to separate the output
lines.
* @return              A String containing the Base64 encoded data, broken into
lines.
*/
public static String encodeLines (byte[] in, int iOff, int iLen, int lineLen,
String lineSeparator) {
int blockLen = (lineLen*3) / 4;
if (blockLen <= 0) throw new IllegalArgumentException();
int lines = (iLen+blockLen-1) / blockLen;
int bufLen = ((iLen+2)/3)*4 + lines*lineSeparator.length();
StringBuilder buf = new StringBuilder(bufLen);
int ip = 0;
while (ip < iLen) {
   int l = Math.min(iLen-ip, blockLen);
   buf.append (encode(in, iOff+ip, l));
   buf.append (lineSeparator);
   ip += l; }
return buf.toString(); }

/**
* Encodes a byte array into Base64 format.
* No blanks or line breaks are inserted in the output.
* @param in  An array containing the data bytes to be encoded.
* @return    A character array containing the Base64 encoded data.
*/
public static char[] encode (byte[] in) {
return encode(in, 0, in.length); }

/**
* Encodes a byte array into Base64 format.
* No blanks or line breaks are inserted in the output.
* @param in    An array containing the data bytes to be encoded.
* @param iLen  Number of bytes to process in <code>in</code>.
* @return      A character array containing the Base64 encoded data.
*/
public static char[] encode (byte[] in, int iLen) {
return encode(in, 0, iLen); }

/**
* Encodes a byte array into Base64 format.
* No blanks or line breaks are inserted in the output.
* @param in    An array containing the data bytes to be encoded.
* @param iOff  Offset of the first byte in <code>in</code> to be processed.
* @param iLen  Number of bytes to process in <code>in</code>, starting at
<code>iOff</code>.
* @return      A character array containing the Base64 encoded data.
*/
public static char[] encode (byte[] in, int iOff, int iLen) {
int oDataLen = (iLen*4+2)/3;       // output length without padding
int oLen = ((iLen+2)/3)*4;         // output length including padding
char[] out = new char[oLen];
int ip = iOff;
int iEnd = iOff + iLen;
int op = 0;
while (ip < iEnd) {
   int i0 = in[ip++] & 0xff;
   int i1 = ip < iEnd ? in[ip++] & 0xff : 0;
   int i2 = ip < iEnd ? in[ip++] & 0xff : 0;
   int o0 = i0 >>> 2;
   int o1 = ((i0 &   3) << 4) | (i1 >>> 4);
   int o2 = ((i1 & 0xf) << 2) | (i2 >>> 6);
   int o3 = i2 & 0x3F;
   out[op++] = map1[o0];
   out[op++] = map1[o1];
   out[op] = op < oDataLen ? map1[o2] : '='; op++;
   out[op] = op < oDataLen ? map1[o3] : '='; op++; }
return out; }

/**
* Decodes a string from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param s  A Base64 String to be decoded.
* @return   A String containing the decoded data.
* @throws   IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static String decodeString (String s) {
return new String(decode(s)); }

/**
* Decodes a byte array from Base64 format and ignores line separators, tabs
and blanks.
* CR, LF, Tab and Space characters are ignored in the input data.
* This method is compatible with
<code>sun.misc.BASE64Decoder.decodeBuffer(String)</code>.
* @param s  A Base64 String to be decoded.
* @return   An array containing the decoded data bytes.
* @throws   IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decodeLines (String s) {
char[] buf = new char[s.length()];
int p = 0;
for (int ip = 0; ip < s.length(); ip++) {
   char c = s.charAt(ip);
   if (c != ' ' && c != '\r' && c != '\n' && c != '\t')
      buf[p++] = c; }
return decode(buf, 0, p); }

/**
* Decodes a byte array from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
```

```
* @param s  A Base64 String to be decoded.
* @return   An array containing the decoded data bytes.
* @throws   IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decode (String s) {
return decode(s.toCharArray()); }

/**
* Decodes a byte array from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param in  A character array containing the Base64 encoded data.
* @return    An array containing the decoded data bytes.
* @throws    IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decode (char[] in) {
return decode(in, 0, in.length); }

/**
* Decodes a byte array from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param in    A character array containing the Base64 encoded data.
* @param iOff  Offset of the first character in <code>in</code> to be
processed.
* @param iLen  Number of characters to process in <code>in</code>,
starting at <code>iOff</code>.
* @return      An array containing the decoded data bytes.
* @throws      IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decode (char[] in, int iOff, int iLen) {
if (iLen%4 != 0) throw new IllegalArgumentException ("Length of Base64
encoded input string is not a multiple of 4.");
while (iLen > 0 && in[iOff+iLen-1] == '=') iLen--;
int oLen = (iLen*3) / 4;
byte[] out = new byte[oLen];
int ip = iOff;
int iEnd = iOff + iLen;
int op = 0;
while (ip < iEnd) {
   int i0 = in[ip++];
   int i1 = in[ip++];
   int i2 = ip < iEnd ? in[ip++] : 'A';
   int i3 = ip < iEnd ? in[ip++] : 'A';
   if (i0 > 127 || i1 > 127 || i2 > 127 || i3 > 127)
      throw new IllegalArgumentException ("Illegal character in Base64
encoded data.");
   int b0 = map2[i0];
   int b1 = map2[i1];
   int b2 = map2[i2];
   int b3 = map2[i3];
   if (b0 < 0 || b1 < 0 || b2 < 0 || b3 < 0)
      throw new IllegalArgumentException ("Illegal character in Base64
encoded data.");
   int o0 = ( b0       <<2) | (b1>>>4);
   int o1 = ((b1 & 0xf)<<4) | (b2>>>2);
   int o2 = ((b2 &   3)<<6) |  b3;
   out[op++] = (byte)o0;
   if (op<oLen) out[op++] = (byte)o1;
   if (op<oLen) out[op++] = (byte)o2; }
return out; }

//Dummy constructor.
private Base64Coder() {}

} // end class Base64Coder
```

DataEntryForm.java

```
package cepir.shared;

import java.sql.Timestamp;

public class DataEntryForm implements java.io.Serializable{
		public Integer formID;
		public Project projID;
		public String formName;
		public String formDesc;
		public User creator;
		public Timestamp dateCreated;
		public Timestamp dateUpdated;

		public DataEntryForm() {
		}

		public Integer getFormID() {
			return formID;
		}

		public void setFormID(Integer formID) {
			this.formID = formID;
		}

		public Project getProjID() {
			return projID;
		}

		public void setProjID(Project projID) {
			this.projID = projID;
		}

		public String getFormName() {
			return formName;
		}

		public void setFormName(String formName) {
			this.formName = formName;
		}

		public String getFormDesc() {
			return formDesc;
		}

		public void setFormDesc(String formDesc) {
			this.formDesc = formDesc;
		}

		public User getCreator() {
			return creator;
		}

		public void setCreator(User creator) {
			this.creator = creator;
		}

		public Timestamp getDateCreated() {
			return dateCreated;
		}

		public void setDateCreated(Timestamp dateCreated) {
			this.dateCreated = dateCreated;
		}

		public Timestamp getDateUpdated() {
			return dateUpdated;
		}

		public void setDateUpdated(Timestamp dateUpdated) {
			this.dateUpdated = dateUpdated;
		}

		public Timestamp getTimestamp(){
			java.util.Date today = new java.util.Date();
			return new java.sql.Timestamp(today.getTime());
		}

}
```

MiscFile.java

```
package cepir.shared;

import java.sql.Timestamp;

public class MiscFile implements java.io.Serializable{
		public Integer miscFileID;
		public String miscFileName;
		public String miscFileDesc;
		public String miscFileLink;
```

```
public String miscFilePath;
public User creator;
public Project project;
public Timestamp dateCreated;
public Integer type;

public MiscFile() {
}

public Integer getMiscFileID() {
		return miscFileID;
}

public void setMiscFileID(Integer miscFileID) {
		this.miscFileID = miscFileID;
}

public String getMiscFileName() {
		return miscFileName;
}

public void setMiscFileName(String miscFileName) {
		this.miscFileName = miscFileName;
}

public String getMiscFileDesc() {
		return miscFileDesc;
}

public void setMiscFileDesc(String miscFileDesc) {
		this.miscFileDesc = miscFileDesc;
}

public String getMiscFileLink() {
		return miscFileLink;
}

public void setMiscFileLink(String miscFileLink) {
		this.miscFileLink = miscFileLink;
}

public String getMiscFilePath() {
		return miscFilePath;
}

public void setMiscFilePath(String miscFilePath) {
		this.miscFilePath = miscFilePath;
}

public User getCreator() {
		return creator;
}

public void setCreator(User creator) {
		this.creator = creator;
}

public Timestamp getDateCreated() {
		return dateCreated;
}

public void setDateCreated(Timestamp dateCreated) {
		this.dateCreated = dateCreated;
}

public Project getProject() {
		return project;
}

public void setProject(Project project) {
		this.project = project;
}

public Integer getType() {
		return type;
}

public void setType(Integer type) {
		this.type = type;
}

		public Timestamp getTimestamp(){
			java.util.Date today = new java.util.Date();
			return new java.sql.Timestamp(today.getTime());
		}
}
```

MiscFileFolder.java

```
package cepir.shared;

import java.sql.Timestamp;

public class MiscFileFolder implements java.io.Serializable{
		public Integer miscfileFolderID;
		public String miscfileFolderName;
		public String miscfileFolderDesc;
		public String miscfileFolderPath;
		public User creator;
		public Project project;
		public Timestamp dateCreated;

		public MiscFileFolder() {
		}

		public Integer getMiscfileFolderID() {
			return miscfileFolderID;
		}
		public void setMiscfileFolderID(Integer miscfileFolderID) {
			this.miscfileFolderID = miscfileFolderID;
		}
		public String getMiscfileFolderName() {
			return miscfileFolderName;
		}
		public void setMiscfileFolderName(String
miscfileFolderName) {
			this.miscfileFolderName = miscfileFolderName;
		}
		public String getMiscfileFolderDesc() {
			return miscfileFolderDesc;
		}
		public void setMiscfileFolderDesc(String miscfileFolderDesc)
{
			this.miscfileFolderDesc = miscfileFolderDesc;
		}
		public String getMiscfileFolderPath() {
			return miscfileFolderPath;
		}
		public void setMiscfileFolderPath(String miscfileFolderPath) {
			this.miscfileFolderPath = miscfileFolderPath;
		}
		public User getCreator() {
			return creator;
		}
		public void setCreator(User creator) {
			this.creator = creator;
		}
		public Timestamp getDateCreated() {
			return dateCreated;
		}
		public void setDateCreated(Timestamp dateCreated) {
			this.dateCreated = dateCreated;
		}

		public Project getProject() {
			return project;
		}

		public void setProject(Project project) {
			this.project = project;
		}

		public Timestamp getTimestamp(){
			java.util.Date today = new java.util.Date();
			return new java.sql.Timestamp(today.getTime());
		}

}
```

NewProjRequest.java

```
package cepir.shared;
```

```
import java.io.Serializable;

public class NewProjRequest implements Serializable{
	public Integer newProjRequestID;
	public String newProjName;
	public String newProjDesc;
	public User user;

	public NewProjRequest() {
	}

	public NewProjRequest(String newProjName, User user) {
		this.newProjName = newProjName;
		this.user = user;
	}

	public NewProjRequest(String newProjName, String
newProjDesc,
				User user) {
		this.newProjName = newProjName;
		this.newProjDesc = newProjDesc;
		this.user = user;
	}

	public Integer getNewProjRequestID() {
		return newProjRequestID;
	}

	public void setNewProjRequestID(Integer newProjRequestID)
{
		this.newProjRequestID = newProjRequestID;
	}

	public String getNewProjName() {
		return newProjName;
	}

	public void setNewProjName(String newProjName) {
		this.newProjName = newProjName;
	}

	public String getNewProjDesc() {
		return newProjDesc;
	}

	public void setNewProjDesc(String newProjDesc) {
		this.newProjDesc = newProjDesc;
	}

	public User getUsername() {
		return user;
	}

	public void setUsername(User user) {
		this.user = user;
	}


}
```

Project.java

```
package cepir.shared;

import java.io.Serializable;
import java.sql.Timestamp;
import java.util.Calendar;

public class Project implements Serializable{

	private static final long serialVersionUID = 1L;

	public Integer projID;
	public String projName;
	public String projDesc;
	public Timestamp dateCreated;
	public Timestamp dateUpdated;

	public Project(){
	}

	public Project(String projName) {
		this.projName = projName;
//		this.dateCreated = getTimestamp();
//		this.dateUpdated = getTimestamp();
	}

	public Project(String projName, String projDesc) {
		this.projName = projName;
		this.projDesc = projDesc;
//		this.dateCreated = getTimestamp();
//		this.dateUpdated = getTimestamp();
	}

	public Integer getProjID() {
		return projID;
	}
	public void setProjID(Integer projID) {
		this.projID = projID;
	}
	public String getProjName() {
		return projName;
	}
	public void setProjName(String projName) {
		this.projName = projName;
	}
	public String getProjDesc() {
		return projDesc;
	}
	public void setProjDesc(String projDesc) {
		this.projDesc = projDesc;
	}
	public Timestamp getDateCreated() {
		return dateCreated;
	}
	public void setDateCreated(Timestamp dateCreated) {
		this.dateCreated = dateCreated;
	}
	public Timestamp getDateUpdated() {
		return dateUpdated;
	}
	public void setDateUpdated(Timestamp dateUpdated) {
		this.dateUpdated = dateUpdated;
	}

	public Timestamp getTimestamp(){
		java.util.Date today = new java.util.Date();
		return new java.sql.Timestamp(today.getTime());
	}
}
```

ProjMemRequest.java

```
package cepir.shared;

import java.io.Serializable;

public class ProjMemRequest implements Serializable{

	private static final long serialVersionUID = 1L;

	public ProjMemRequestPK projMemRequestPK;
	public String message;

	public ProjMemRequest() {
	}

	public ProjMemRequest(ProjMemRequestPK
projMemRequestPK, String message) {
		this.projMemRequestPK = projMemRequestPK;
		this.message = message;
	}

	public ProjMemRequestPK getProjMemRequestPK() {
		return projMemRequestPK;
	}

	public void setProjMemRequestPK(ProjMemRequestPK
projMemRequestPK) {
		this.projMemRequestPK = projMemRequestPK;
```

```
	}

	public String getMessage() {
		return message;
	}

	public void setMessage(String message) {
		this.message = message;
	}


}
```

ProjMemRequestPK.java

```
package cepir.shared;

import java.io.Serializable;

public class ProjMemRequestPK implements Serializable{

	private static final long serialVersionUID = 1L;

	public User username;
	public Project projID;

	public ProjMemRequestPK() {
	}

	public ProjMemRequestPK(User username, Project projID) {
		this.username = username;
		this.projID = projID;
	}

	public User getUsername() {
		return username;
	}

	public void setUsername(User username) {
		this.username = username;
	}

	public Project getProjID() {
		return projID;
	}

	public void setProjID(Project projID) {
		this.projID = projID;
	}


}
```

Reference.java

```
package cepir.shared;

import java.io.Serializable;

public class ProjMemRequestPK implements Serializable{

	private static final long serialVersionUID = 1L;

	public User username;
	public Project projID;

	public ProjMemRequestPK() {
	}

	public ProjMemRequestPK(User username, Project projID) {
		this.username = username;
		this.projID = projID;
	}

	public User getUsername() {
		return username;
	}

	public void setUsername(User username) {
		this.username = username;
	}

	public Project getProjID() {
		return projID;
	}

	public void setProjID(Project projID) {
		this.projID = projID;
	}


}
```

ReferenceFolder.java

```
package cepir.shared;

import java.sql.Timestamp;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

public class ReferenceFolder implements java.io.Serializable{
	public Integer refFolderID;
	public String refFolderName;
	public String refFolderDesc;
	public String refFolderPath;
	public User creator;
	public Project project;
	public Timestamp dateCreated;

	public ReferenceFolder() {
	}

	public Integer getRefFolderID() {
		return refFolderID;
	}

	public void setRefFolderID(Integer refFolderID) {
		this.refFolderID = refFolderID;
	}

	public String getRefFolderName() {
		return refFolderName;
	}

	public void setRefFolderName(String refFolderName) {
		this.refFolderName = refFolderName;
	}

	public String getRefFolderDesc() {
		return refFolderDesc;
	}

	public void setRefFolderDesc(String refFolderDesc) {
		this.refFolderDesc = refFolderDesc;
	}

	public String getRefFolderPath() {
		return refFolderPath;
	}

	public void setRefFolderPath(String refFolderPath) {
		this.refFolderPath = refFolderPath;
	}

	public User getCreator() {
		return creator;
	}

	public void setCreator(User creator) {
		this.creator = creator;
	}

	public Timestamp getDateCreated() {
		return dateCreated;
```

```
	}

	public void setDateCreated(Timestamp dateCreated) {
		this.dateCreated = dateCreated;
	}

	public Project getProject() {
		return project;
	}

	public void setProject(Project project) {
		this.project = project;
	}
	public Timestamp getTimestamp(){
		java.util.Date today = new java.util.Date();
		return new java.sql.Timestamp(today.getTime());
	}

}
```

ReferenceType.java

```
package cepir.shared;

public enum ReferenceType {
	Book,
	Book_Chapter{
		public String toString() {
			return name().replace("_", " ");
		}},
	Electronic_Article{
		public String toString() {
			return name().replace("_", " ");
		}},
	Electronic_Book{
		public String toString() {
			return name().replace("_", " ");
		}},
	Journal_Article{
		public String toString() {
			return name().replace("_", " ");
		}},
	Thesis,
	Unpublished_Work{
		public String toString() {
			return name().replace("_", " ");
		}}
	}
```

Role.java

```
package cepir.shared;

import java.io.Serializable;

public class Role implements Serializable{
	public Integer roleID;
	public String roleName;

	public Role() {
	}

	public Role(String roleName) {
		this.roleName = roleName;
	}

	public Integer getRoleID() {
		return roleID;
	}

	public void setRoleID(Integer roleID) {
		this.roleID = roleID;
	}

	public String getRoleName() {
		return roleName;
	}

	public void setRoleName(String roleName) {
		this.roleName = roleName;
	}

}
```

Tool.java

```
package cepir.shared;

import java.io.Serializable;
import java.sql.Timestamp;

public class Tool implements Serializable{
	public Integer toolID;
	public String toolName;
	public String toolDesc;
	public String toolLink;
	public String toolPath;
	public User creator;
	public Timestamp dateCreated;
	public Integer type; //1 - file, 2 - link

	public Tool() {
	}

	public Integer getToolID() {
		return toolID;
	}

	public void setToolID(Integer toolID) {
		this.toolID = toolID;
	}

	public String getToolName() {
		return toolName;
	}

	public void setToolName(String toolName) {
		this.toolName = toolName;
	}

	public String getToolDesc() {
		return toolDesc;
	}

	public void setToolDesc(String toolDesc) {
		this.toolDesc = toolDesc;
	}

	public String getToolLink() {
		return toolLink;
	}

	public void setToolLink(String toolLink) {
		this.toolLink = toolLink;
	}

	public String getToolPath() {
		return toolPath;
	}

	public void setToolPath(String toolPath) {
		this.toolPath = toolPath;
	}

	public User getCreator() {
		return creator;
	}

	public void setCreator(User creator) {
		this.creator = creator;
	}

	public Integer getType() {
		return type;
	}

	public void setType(Integer type) {
		this.type = type;
	}

	public Timestamp getTimestamp(){
		java.util.Date today = new java.util.Date();
```

```
                return new java.sql.Timestamp(today.getTime());
        }
}
```

ToolFolder.java

```
package cepir.shared;

import java.io.Serializable;
import java.sql.Timestamp;

public class ToolFolder implements Serializable{
        public Integer toolFolderID;
        public String toolFolderName;
        public String toolFolderDesc;
        public String toolFolderPath;
        public User creator;
        public Timestamp dateCreated;

        public ToolFolder() {
        }

        public Integer getToolFolderID() {
                return toolFolderID;
        }

        public void setToolFolderID(Integer toolFolderID) {
                this.toolFolderID = toolFolderID;
        }

        public String getToolFolderName() {
                return toolFolderName;
        }

        public void setToolFolderName(String toolFolderName) {
                this.toolFolderName = toolFolderName;
        }

        public String getToolFolderDesc() {
                return toolFolderDesc;
        }

        public void setToolFolderDesc(String toolFolderDesc) {
                this.toolFolderDesc = toolFolderDesc;
        }

        public String getToolFolderPath() {
                return toolFolderPath;
        }

        public void setToolFolderPath(String toolFolderPath) {
                this.toolFolderPath = toolFolderPath;
        }

        public User getCreator() {
                return creator;
        }

        public void setCreator(User creator) {
                this.creator = creator;
        }

        public Timestamp getDateCreated() {
                return dateCreated;
        }

        public void setDateCreated(Timestamp dateCreated) {
                this.dateCreated = dateCreated;
        }

        public Timestamp getTimestamp(){
                java.util.Date today = new java.util.Date();
                return new java.sql.Timestamp(today.getTime());
        }

}
```

User.java

```
package cepir.shared;

import java.io.Serializable;
import cepir.shared.YesNo;


public class User implements Serializable, Comparable<User>{

        private static final long serialVersionUID = 1L;

        public String username;
        public String password;
        public String email;
        public String lastName;
        public String firstName;
        public String middleName;
        public String affiliation;
        public YesNo sysAd;
        public YesNo toolAd;
        public String question;
        public String answer;

        public User(String username, String password, String email,
                        String lastName, String firstName,
String middleName, YesNo sysAd, String question, String answer) {
                this.username = username;
                this.password = password;
                this.email = email;
                this.lastName = lastName;
                this.firstName = firstName;
                this.middleName = middleName;
                this.sysAd = sysAd;
                this.question = question;
                this.answer = answer;
        }

        public User() {
        }

        public User(String username, String password, String email,
                        String lastName, String firstName,
String middleName,
                        String affiliation, YesNo sysAd,
YesNo toolAd, String question, String answer) {
                this.username = username;
                this.password = password;
                this.email = email;
                this.lastName = lastName;
                this.firstName = firstName;
                this.middleName = middleName;
                this.affiliation = affiliation;
                this.sysAd = sysAd;
                this.toolAd = toolAd;
                this.question = question;
                this.answer = answer;
        }

        public User(String username, String email, String lastName,
                        String firstName, String middleName,
YesNo sysAd, String affiliation, String question, String answer) {
                this.username = username;
                this.email = email;
                this.lastName = lastName;
                this.firstName = firstName;
                this.middleName = middleName;
                this.affiliation = affiliation;
                this.sysAd = sysAd;
                this.question = question;
                this.answer = answer;
        }



        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public String getPassword() {
```

```
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getMiddleName() {
		return middleName;
	}

	public void setMiddleName(String middleName) {
		this.middleName = middleName;
	}

	public String getAffiliation() {
		return affiliation;
	}

	public void setAffiliation(String affiliation) {
		this.affiliation = affiliation;
	}

	public YesNo getSysAd() {
		return sysAd;
	}

	public void setSysAd(YesNo sysAd) {
		this.sysAd = sysAd;
	}

	public YesNo getToolAd() {
		return toolAd;
	}

	public void setToolAd(YesNo toolAd) {
		this.toolAd = toolAd;
	}

	public String getQuestion() {
		return question;
	}

	public void setQuestion(String question) {
		this.question = question;
	}

	public String getAnswer() {
		return answer;
	}

	public void setAnswer(String answer) {
		this.answer = answer;
	}

	public int compareTo(User o) {
		return username.compareTo(o.username);
	}

	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((username == null) ? 0
: username.hashCode());
		return result;
	}

	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		User other = (User) obj;
		if (username == null) {
			if (other.username != null)
				return false;
		} else if (!username.equals(other.username))
			return false;
		return true;
	}

}
```

YesNo.java

```
package cepir.shared;

public enum YesNo {
	Yes, No;
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <!-- Servlets -->
<listener>
   <description>
		Used to cleanup when a session is destroyed</description>
   <display-name>ZK Session cleaner</display-name>
   <listener-class>org.zkoss.zk.ui.http.HttpSessionListener</listener-class>
  </listener>
  <servlet>
   <description>
		The ZK loader for ZUML pages</description>
   <servlet-name>zkLoader</servlet-name>
   <servlet-class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-class>
   <init-param>
     <param-name>update-uri</param-name>
     <param-value>/zkau</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
   <description>
		The asynchronous update engine for ZK</description>
   <servlet-name>auEngine</servlet-name>
   <servlet-class>org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>zkLoader</servlet-name>
   <url-pattern>*.zul</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
   <servlet-name>zkLoader</servlet-name>
   <url-pattern>*.zhtml</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
   <servlet-name>auEngine</servlet-name>
   <url-pattern>/zkau/*</url-pattern>
```

```
</servlet-mapping>
<servlet>
  <servlet-name>userServiceServlet</servlet-name>
  <servlet-class>cepir.server.UserServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>userServiceServlet</servlet-name>
  <url-pattern>/cepir/userService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>mailServiceServlet</servlet-name>
  <servlet-class>cepir.server.MailUtil</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>mailServiceServlet</servlet-name>
  <url-pattern>/cepir/mailService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>projectServiceServlet</servlet-name>
  <servlet-class>cepir.server.ProjectServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>projectServiceServlet</servlet-name>
  <url-pattern>/cepir/projectService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>formServiceServlet</servlet-name>
  <servlet-class>cepir.server.FormServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>formServiceServlet</servlet-name>
  <url-pattern>/cepir/formService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>toolServiceServlet</servlet-name>
  <servlet-class>cepir.server.ToolServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>toolServiceServlet</servlet-name>
  <url-pattern>/cepir/toolService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>refServiceServlet</servlet-name>
  <servlet-class>cepir.server.ReferenceServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>refServiceServlet</servlet-name>
  <url-pattern>/cepir/refService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>excelServiceServlet</servlet-name>
  <servlet-class>cepir.server.ExcelServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>excelServiceServlet</servlet-name>
  <url-pattern>/cepir/excelService</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>miscFileServiceServlet</servlet-name>
  <servlet-class>cepir.server.MiscFileServiceImpl</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>miscFileServiceServlet</servlet-name>
  <url-pattern>/cepir/miscFileService</url-pattern>
</servlet-mapping>

 <servlet>
  <servlet-name>uploadFileServiceServlet</servlet-name>
  <servlet-class>cepir.server.UploadFileService</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>uploadFileServiceServlet</servlet-name>
  <url-pattern>/cepir/upload</url-pattern>
 </servlet-mapping>

 <servlet>
  <servlet-name>downloadFileServiceServlet</servlet-name>
  <servlet-class>cepir.server.DownloadFileService</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>downloadFileServiceServlet</servlet-name>
  <url-pattern>/cepir/download</url-pattern>
 </servlet-mapping>


 <!-- Default page to serve -->
 <welcome-file-list>
  <welcome-file>CEpiR.html</welcome-file>
        <welcome-file>index.zul</welcome-file>
 </welcome-file-list>

</web-app>
```

CEpiR.html

```
<!doctype html>
<!-- The DOCTYPE declaration above will set the    -->
<!-- browser's rendering engine into               -->
<!-- "Standards Mode". Replacing this declaration  -->
<!-- with a "Quirks Mode" doctype may lead to some -->
<!-- differences in layout.                        -->

<html>
 <head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">

  <!--                                             -->
  <!-- Consider inlining CSS to reduce the number of requested files -->
  <!--                                             -->
  <link type="text/css" rel="stylesheet" href="CEpiR.css">

  <!--                                  -->
  <!-- Any title is fine                -->
  <!--                                  -->
  <title>Collaboratory for Epidemiological Research - (CEpiR)</title>

  <!--                                  -->
  <!-- This script loads your compiled module.   -->
  <!-- If you add any GWT meta tags, they must   -->
  <!-- be added before this line.                -->
  <!--                                  -->
  <script type="text/javascript" language="javascript"
src="cepir/cepir.nocache.js"></script>
 </head>

 <!--                                  -->
 <!-- The body can have arbitrary html, or      -->
 <!-- you can leave the body empty if you want  -->
 <!-- to create a completely dynamic UI.        -->
 <!--                                  -->
 <body id="background">

  <!-- OPTIONAL: include this if you want history support -->
  <iframe src="javascript:"" id="__gwt_historyFrame" tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>

  <!-- RECOMMENDED if your web app will not function without
JavaScript enabled -->
  <noscript>
   <div style="width: 22em; position: absolute; left: 50%; margin-left: -
11em; color: red; background-color: white; border: 1px solid red; padding:
4px; font-family: sans-serif">
    Your web browser must have JavaScript enabled
    in order for this application to display correctly.
```

```
    </div>
   </noscript>

  </body>
</html>
```

CEpiR.css

```
a{
		text-decoration:none;
}

#background{
		background-color: #374370;
}

.center{
		background-color: #E2E5E8;
}

.cursor{
		cursor: pointer;
}

.errorMsg{
		color: #F2583E;
}

.flexTable{
		 table-layout:fixed;
		 border-collapse: collapse;
}

.gwt-Frame{
		border: 0px;
}

.gwt-Hyperlink{
		color:#000000;
}

.gwt-Hyperlink a:hover{
		color:#5790B3;
}

.header{
		color:#3D72A4;
		font-size: 20px;
		font-weight: bold;
		font-family: arial;
}

.header2{
		color:#3D72A4;
		padding: 5px;
		font-size: 15px;
		font-family: arial;
		font-weight: bold;
		background-color: #9AB4E3;
}

.header3{
		color:#4782A6;
		font-size: 13px;
		font-family: arial;
		font-weight: bold;
}

.header4{
		color:#4782A6;
		font-size: 30px;
		font-family: arial;
		font-weight: bold;
}

.header5 {
		color: white;
		padding: 5px;
		font-variant: normal;
		font-size: 14px;
		font-family: arial;
		font-weight: bold;
		background-color: steelblue;
}

.icon {
		border: 0px;
}

.icon2 {
		color: White;
		padding: 2px;
		border: 1px solid cadetblue;
		font-family: arial;
		font-weight: bold;
		text-align: center;
		background-color: cornflowerblue;
}
.icon3 {
		color: White;
		padding: 2px;
		border: 1px solid lightcoral;
		font-family: arial;
		font-weight: bold;
		text-align: center;
		background-color: orangered;
}

.idLabel{
		color: #0099cc;
		font-weight: bold;
}

/*for project description in project home*/
.labelPadding {
		padding-left: 40px;
		text-align: justify;
}

.link{
  color: #0099cc;
}

.link a:hover{
   color: #0099CC;
}

.okMsg{
		color: #3AE05C;
}

.tableHeader{
		color: #E2E5E8;
		padding: 2px;
		text-align: center;
		background-color: #4782A6;
}

.tableHighlight{
		/*background-color: #48AFF7;*/
		background-color: #A2C2FA;
		color: #334552;
}
.unauthErrorBox {
		width: 5px red;
		border: 5px solid Red;
}
.recordBorder {
		border-bottom: 1px dotted cornflowerblue;
}

.recordsTable {
		border-collapse: collapse;
		border:1px solid cornflowerblue;
		border-padding:3px;
		cell-spacing:0px;
}
.richTextArea {
		border-top: 1px outset;
		border-right: 0px;
		border-bottom: 0px;
```

```
	border-left: 0px;
	font-family: Arial;
	background-color: white;
}

.sideBorder {
	border-left: 1px dotted cornflowerblue;
	border-right: 1px dotted cornflowerblue;
}
```

index.zul

```
<?page title="Edit data - CEpiR"?>
<zk>
	<div id="main" height="100%" width="100%"
apply="data.InitWindow">
		<menubar>
			<menu label="File">
				<menupopup>
					<menuitem
id="save" label="Save"></menuitem>
					<menuitem
id="export" label="Export to Excel"></menuitem>
				</menupopup>
			</menu>
			<menu label="Edit">
				<menupopup>
					<menuitem
id="cut" label="Cut"></menuitem>
					<menuitem
id="copy" label="Copy"></menuitem>
					<menuitem
id="paste" label="Paste"></menuitem>
				</menupopup>
			</menu>
			<menuitem label="Formula List"
id="formulas"></menuitem>
		</menubar>
		<menupopup id="rightClickPopup">
			<menu label="Shift cells">
				<menupopup>
					<menuitem
id="shiftCellUp" label="Shift cells up"></menuitem>
					<menuitem
id="shiftCellDown" label="Shift cells down" />
					<menuitem
id="shiftCellLeft" label="Shift cells left" />
					<menuitem
id="shiftCellRight" label="Shift cells right"></menuitem>
				</menupopup>
			</menu>
			<menu label="Insert">
				<menupopup>
					<menuitem
id="insertRow" label="Insert Row"></menuitem>
					<menuitem
id="insertCol" label="Insert Column"></menuitem>
				</menupopup>
			</menu>
			<menu label="Delete">
				<menupopup>
					<menuitem
id="deleteRow" label="Delete Row"></menuitem>
					<menuitem
id="deleteCol" label="Delete Column"></menuitem>
				</menupopup>
			</menu>
		</menupopup>
		<div height="3px"></div>
		<combobox id="focusCombobox"
mold="rounded"
			style="text-align:center;">
		</combobox><textbox id="formulaEditor"
cols="50" />
		<div height="3px"></div>
		<window id="formula" title="List of formulas"
visible="false" border="normal" position="center" width="500px"
closable="true">


		</window>
		<spreadsheet id="sheet" ctrlKeys="^c^v^x"/>
	</div>
</zk>
```

timeout.zul

```
<window width="80%" title="Session Timeout" border="normal">
	<vbox>
The page or component you request is no longer available.
This is normally caused by timeout, opening too many Web pages, or
rebooting
the server.
	</vbox>
</window>
```

zk.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<zk>
	<device-config>
		<device-type>ajax</device-type>
		<timeout-uri>/timeout.zul</timeout-uri><!-- An
empty URL can cause the browser to reload the same URL -->
	</device-config>
</zk>
```

Base64Coder.java

```
package data;

//Copyright 2003-2010 Christian d'Heureuse, Inventec Informatik AG,
Zurich, Switzerland
//www.source-code.biz, www.inventec.ch/chdh
//
//This module is multi-licensed and may be used under the terms
//of any of the following licenses:
//
//EPL, Eclipse Public License, V1.0 or later, http://www.eclipse.org/legal
//LGPL, GNU Lesser General Public License, V2.1 or later,
http://www.gnu.org/licenses/lgpl.html
//GPL, GNU General Public License, V2 or later,
http://www.gnu.org/licenses/gpl.html
//AL, Apache License, V2.0 or later, http://www.apache.org/licenses
//BSD, BSD License, http://www.opensource.org/licenses/bsd-license.php
//
//Please contact the author if you need another license.
//This module is provided "as is", without warranties of any kind.

/**
* A Base64 encoder/decoder.
*
* <p>
* This class is used to encode and decode data in Base64 format as described
in RFC 1521.
*
* <p>
* Project home page: <a href="http://www.source-
code.biz/base64coder/java/">www.source-
code.biz/base64coder/java</a><br>
* Author: Christian d'Heureuse, Inventec Informatik AG, Zurich,
Switzerland<br>
* Multi-licensed: EPL / LGPL / GPL / AL / BSD.
*/
public class Base64Coder {

//The line separator string of the operating system.
//private static final String systemLineSeparator =
System.getProperty("line.separator");
	private static final String systemLineSeparator = "CR+LF";

//Mapping table from 6-bit nibbles to Base64 characters.
private static char[]    map1 = new char[64];
static {
  int i=0;
  for (char c='A'; c<='Z'; c++) map1[i++] = c;
  for (char c='a'; c<='z'; c++) map1[i++] = c;
  for (char c='0'; c<='9'; c++) map1[i++] = c;
  map1[i++] = '+'; map1[i++] = '/'; }

//Mapping table from Base64 characters to 6-bit nibbles.
private static byte[]    map2 = new byte[128];
static {
```

```
   for (int i=0; i<map2.length; i++) map2[i] = -1;
   for (int i=0; i<64; i++) map2[map1[i]] = (byte)i; }

/**
* Encodes a string into Base64 format.
* No blanks or line breaks are inserted.
* @param s  A String to be encoded.
* @return   A String containing the Base64 encoded data.
*/
public static String encodeString (String s) {
return new String(encode(s.getBytes())); }

/**
* Encodes a byte array into Base 64 format and breaks the output into lines
of 76 characters.
* This method is compatible with
<code>sun.misc.BASE64Encoder.encodeBuffer(byte[])</code>.
* @param in  An array containing the data bytes to be encoded.
* @return    A String containing the Base64 encoded data, broken into lines.
*/
public static String encodeLines (byte[] in) {
return encodeLines(in, 0, in.length, 76, systemLineSeparator); }

/**
* Encodes a byte array into Base 64 format and breaks the output into lines.
* @param in            An array containing the data bytes to be encoded.
* @param iOff          Offset of the first byte in <code>in</code> to be
processed.
* @param iLen          Number of bytes to be processed in <code>in</code>,
starting at <code>iOff</code>.
* @param lineLen       Line length for the output data. Should be a multiple
of 4.
* @param lineSeparator The line separator to be used to separate the output
lines.
* @return              A String containing the Base64 encoded data, broken into
lines.
*/
public static String encodeLines (byte[] in, int iOff, int iLen, int lineLen,
String lineSeparator) {
int blockLen = (lineLen*3) / 4;
if (blockLen <= 0) throw new IllegalArgumentException();
int lines = (iLen+blockLen-1) / blockLen;
int bufLen = ((iLen+2)/3)*4 + lines*lineSeparator.length();
StringBuilder buf = new StringBuilder(bufLen);
int ip = 0;
while (ip < iLen) {
   int l = Math.min(iLen-ip, blockLen);
   buf.append (encode(in, iOff+ip, l));
   buf.append (lineSeparator);
   ip += l; }
return buf.toString(); }

/**
* Encodes a byte array into Base64 format.
* No blanks or line breaks are inserted in the output.
* @param in  An array containing the data bytes to be encoded.
* @return    A character array containing the Base64 encoded data.
*/
public static char[] encode (byte[] in) {
return encode(in, 0, in.length); }

/**
* Encodes a byte array into Base64 format.
* No blanks or line breaks are inserted in the output.
* @param in    An array containing the data bytes to be encoded.
* @param iLen  Number of bytes to process in <code>in</code>.
* @return      A character array containing the Base64 encoded data.
*/
public static char[] encode (byte[] in, int iLen) {
return encode(in, 0, iLen); }

/**
* Encodes a byte array into Base64 format.
* No blanks or line breaks are inserted in the output.
* @param in    An array containing the data bytes to be encoded.
* @param iOff  Offset of the first byte in <code>in</code> to be processed.
* @param iLen  Number of bytes to process in <code>in</code>, starting at
<code>iOff</code>.
* @return      A character array containing the Base64 encoded data.
*/
public static char[] encode (byte[] in, int iOff, int iLen) {
int oDataLen = (iLen*4+2)/3;       // output length without padding
int oLen = ((iLen+2)/3)*4;         // output length including padding
char[] out = new char[oLen];
int ip = iOff;
int iEnd = iOff + iLen;
int op = 0;
while (ip < iEnd) {
   int i0 = in[ip++] & 0xff;
   int i1 = ip < iEnd ? in[ip++] & 0xff : 0;
   int i2 = ip < iEnd ? in[ip++] & 0xff : 0;
   int o0 = i0 >>> 2;
   int o1 = ((i0 &   3) << 4) | (i1 >>> 4);
   int o2 = ((i1 & 0xf) << 2) | (i2 >>> 6);
   int o3 = i2 & 0x3F;
   out[op++] = map1[o0];
   out[op++] = map1[o1];
   out[op] = op < oDataLen ? map1[o2] : '='; op++;
   out[op] = op < oDataLen ? map1[o3] : '='; op++; }
return out; }

/**
* Decodes a string from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param s  A Base64 String to be decoded.
* @return   A String containing the decoded data.
* @throws   IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static String decodeString (String s) {
return new String(decode(s)); }

/**
* Decodes a byte array from Base64 format and ignores line separators, tabs
and blanks.
* CR, LF, Tab and Space characters are ignored in the input data.
* This method is compatible with
<code>sun.misc.BASE64Decoder.decodeBuffer(String)</code>.
* @param s  A Base64 String to be decoded.
* @return   An array containing the decoded data bytes.
* @throws   IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decodeLines (String s) {
char[] buf = new char[s.length()];
int p = 0;
for (int ip = 0; ip < s.length(); ip++) {
   char c = s.charAt(ip);
   if (c != ' ' && c != '\r' && c != '\n' && c != '\t')
      buf[p++] = c; }
return decode(buf, 0, p); }

/**
* Decodes a byte array from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param s  A Base64 String to be decoded.
* @return   An array containing the decoded data bytes.
* @throws   IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decode (String s) {
return decode(s.toCharArray()); }

/**
* Decodes a byte array from Base64 format.
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param in  A character array containing the Base64 encoded data.
* @return    An array containing the decoded data bytes.
* @throws    IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decode (char[] in) {
return decode(in, 0, in.length); }

/**
* Decodes a byte array from Base64 format.
```

```
* No blanks or line breaks are allowed within the Base64 encoded input
data.
* @param in    A character array containing the Base64 encoded data.
* @param iOff  Offset of the first character in <code>in</code> to be
processed.
* @param iLen  Number of characters to process in <code>in</code>,
starting at <code>iOff</code>.
* @return      An array containing the decoded data bytes.
* @throws      IllegalArgumentException If the input is not valid Base64
encoded data.
*/
public static byte[] decode (char[] in, int iOff, int iLen) {
if (iLen%4 != 0) throw new IllegalArgumentException ("Length of Base64
encoded input string is not a multiple of 4.");
while (iLen > 0 && in[iOff+iLen-1] == '=') iLen--;
int oLen = (iLen*3) / 4;
byte[] out = new byte[oLen];
int ip = iOff;
int iEnd = iOff + iLen;
int op = 0;
while (ip < iEnd) {
  int i0 = in[ip++];
  int i1 = in[ip++];
  int i2 = ip < iEnd ? in[ip++] : 'A';
  int i3 = ip < iEnd ? in[ip++] : 'A';
  if (i0 > 127 || i1 > 127 || i2 > 127 || i3 > 127)
    throw new IllegalArgumentException ("Illegal character in Base64
encoded data.");
  int b0 = map2[i0];
  int b1 = map2[i1];
  int b2 = map2[i2];
  int b3 = map2[i3];
  if (b0 < 0 || b1 < 0 || b2 < 0 || b3 < 0)
    throw new IllegalArgumentException ("Illegal character in Base64
encoded data.");
  int o0 = ( b0       <<2) | (b1>>>4);
  int o1 = ((b1 & 0xf)<<4) | (b2>>>2);
  int o2 = ((b2 &   3)<<6) |  b3;
  out[op++] = (byte)o0;
  if (op<oLen) out[op++] = (byte)o1;
  if (op<oLen) out[op++] = (byte)o2; }
return out; }

//Dummy constructor.
private Base64Coder() {}

} // end class Base64Coder
```

DBConnection.java

```
package data;
import java.sql.*;

public class DBConnection{
                public static Connection getConnection() {
                  Connection conn = null;
                  String url = "jdbc:mysql://localhost/";
//                String dbName = "cepirDB";
//                String driver = "com.mysql.jdbc.Driver";
//                String userName = "root";
//                String password = "";
                  String dbName = "EpiCollaboratory";
                  String driver = "com.mysql.jdbc.Driver";
                  String userName = "EpiCollaboratory";
                  String password = "uvmpHuLZCErbvm7S";
                  try {
                    Class.forName(driver).newInstance();
                    conn =
DriverManager.getConnection(url+dbName,userName,password);
                     return conn;
                  } catch (Exception e) {
                    e.printStackTrace();
                    return null;
                  }
                 }
                }
```

Function.java

```
package data;

public class Function {
                private String funcName;
                private String format;
                private String details;

                public Function(String funcName, String format, String details)
{
                        this.funcName = funcName;
                        this.format = format;
                        this.details = details;
                }

                public String getFuncName() {
                        return funcName;
                }

                public void setFuncName(String funcName) {
                        this.funcName = funcName;
                }

                public String getFormat() {
                        return format;
                }

                public void setFormat(String format) {
                        this.format = format;
                }

                public String getDetails() {
                        return details;
                }

                public void setDetails(String details) {
                        this.details = details;
                }

}
```

InitWindow.java

```
package data;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.List;

import org.zkoss.poi.ss.usermodel.Cell;
import org.zkoss.poi.ss.usermodel.CellStyle;
import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.Executions;
import org.zkoss.zk.ui.event.Event;
import org.zkoss.zk.ui.event.KeyEvent;
import org.zkoss.zk.ui.event.SelectionEvent;
import org.zkoss.zk.ui.util.Clients;
import org.zkoss.zk.ui.util.GenericForwardComposer;
import org.zkoss.zss.model.Book;
import org.zkoss.zss.model.Exporter;
import org.zkoss.zss.model.Exporters;
import org.zkoss.zss.model.Range;
import org.zkoss.zss.model.Ranges;
import org.zkoss.zss.model.Worksheet;
import org.zkoss.zss.ui.Rect;
import org.zkoss.zss.ui.Spreadsheet;
import org.zkoss.zss.ui.event.CellEvent;
import org.zkoss.zss.ui.event.CellMouseEvent;
import org.zkoss.zss.ui.impl.Utils;
import org.zkoss.zul.*;

import com.mysql.jdbc.PreparedStatement;

public class InitWindow extends GenericForwardComposer{
                Div main;
                Spreadsheet sheet;
                Worksheet currSheet;
                Menupopup rightClickPopup;
```

```
	int rowIndex;
	int colIndex;

	String formID;
	String projID;

	Combobox focusCombobox;
	Comboitem position;
	Textbox formulaEditor;
	Cell currentCell;
	Range currentRange;

	List<Function> functions;
	Window formula;
	Div popupList;
	Listbox fxnList;
	Label format;
	Label details;

	Rect selectedRect;
	private final static String KEY_SOURCE_RANGE =
"KEY_SOURCE_RANGE";
	private final static String KEY_SOURCE_RECT =
"KEY_SOURCE_RECT";
	private final static String KEY_IS_CUT = "KEY_IS_CUT";

	//file menu listeners
	public void onClick$save() throws IOException{
		try {
			Book wb = sheet.getBook();
			Exporter c =
Exporters.getExporter("excel");
			ByteArrayOutputStream baos = new
ByteArrayOutputStream();
			c.export(wb, baos);
//			File excel = new
File("D://workspace/CEpiR/war/data/"+projID+"/"+formID+".xls");
//			File excel = new
File(application.getRealPath("/").replaceFirst("dataLoad/",
"cepir/")+"data/"+projID+"/"+formID+".xls");
			File excel = new
File(application.getRealPath("/").replaceFirst("dataLoad",
"cepir")+"data/"+projID+"/"+formID+".xls");
			FileOutputStream file = new
FileOutputStream(excel);
			baos.writeTo(file);
			baos.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
	}

	public void onClick$close(){
		Clients.evalJavaScript("window.close();");
	}

	public void onClick$export(Event evt) throws IOException {
		Book wb = sheet.getBook();
		Exporter c = Exporters.getExporter("excel");
		ByteArrayOutputStream baos = new
ByteArrayOutputStream();
		c.export(wb, baos);
		Filedownload.save(baos.toByteArray(),
"application/file",
				wb.getBookName());
	}

	//cell right click listeners
	public void doAfterCompose(Component comp) throws
Exception {
		super.doAfterCompose(comp);
		rightClickPopup.setWidgetListener("onOpen",
"this.$f('" + sheet.getId() + "').focus(false)");
		ListFunctions fxn = new ListFunctions();
		functions = fxn.getList();

		fxnList = new Listbox();
		setupHeader();
		fxnList.setRows(2);
		fxnList.setParent(formula);
		formula.setParent(main);
		fxnList.setModel(new ListModelList(functions));
		fxnList.setItemRenderer(new ListitemRenderer() {

			public void render(Listitem list,
Object data) throws Exception {
				final Function func =
(Function)data;
				list.setValue(func);

				new
Listcell(func.getFuncName()).setParent(list);
				new
Listcell(func.getFormat()).setParent(list);
				new
Listcell(func.getDetails()).setParent(list);
			}
		});
	}

	private void setupHeader() {
		Listheader name = new Listheader("Function");
		Listheader format = new Listheader("Format");
		Listheader desc = new Listheader("Description");
		Listhead head = new Listhead();
		head.isSizable();
		name.setParent(head);
		format.setParent(head);
		desc.setParent(head);
		head.setParent(fxnList);
	}

	public void onCellRightClick$sheet(CellMouseEvent event) {
		rowIndex = event.getRow();
		colIndex = event.getColumn();
		currSheet = event.getSheet();
		rightClickPopup.open(event.getPageX(),
event.getPageY());
	}

	public void onClick$insertRow(){
		Ranges.range(currSheet, rowIndex, 0,
currSheet.getLastRowNum(), 255).move(1,0);
	}

	public void onClick$deleteRow() {
		Ranges.range(currSheet, rowIndex, 0, rowIndex,
255).delete(Range.SHIFT_UP);
	}

	public void onClick$deleteCol() {
		int tRow = currSheet.getFirstRowNum();
		int bRow = currSheet.getLastRowNum();
		Ranges.range(currSheet, tRow, colIndex, bRow,
colIndex).delete(Range.SHIFT_LEFT);
	}

	public void onClick$insertCol() {
		int tRow = currSheet.getFirstRowNum();
		int bRow = currSheet.getLastRowNum();
		Ranges.range(currSheet, tRow, colIndex, bRow,
254).move(0,1);
	}

	public void onClick$shiftCellLeft() {
		Range rng = Ranges.range(currSheet, rowIndex,
colIndex);
		rng.delete(Range.SHIFT_LEFT);
	}

	public void onClick$shiftCellUp() {
		final Range rng = Ranges.range(currSheet,
rowIndex, colIndex);
		rng.delete(Range.SHIFT_UP);
	}

	public void onClick$shiftCellRight() {
```

```
                        Ranges.range(currSheet, rowIndex, colIndex,
rowIndex, 254).move(0, 1);
                }

                public void onClick$shiftCellDown() {
                        Ranges.range(currSheet, rowIndex, colIndex,
currSheet.getLastRowNum(), colIndex).move(1, 0);
                }

                //on load functions
                public void onCreate(){
                        String username =
Executions.getCurrent().getParameter("u");
                        projID =
Executions.getCurrent().getParameter("p");
                        formID =
Executions.getCurrent().getParameter("f");

                        if(checkPriv(username,projID,formID)){
                                sheet.setWidth("100%");
                                sheet.setHeight("100%");
                                sheet.setMaxcolumns(256);
                                sheet.setMaxrows(5000);
//
                sheet.setSrc(application.getRealPath("/").replaceFirst("dataLoa
d/", "cepir/")+"data/"+projID+"/"+formID+".xls");

                sheet.setSrc(application.getRealPath("/").replaceFirst("dataLoa
d", "cepir")+"data/"+projID+"/"+formID+".xls");
//
                sheet.setSrc("D://workspace/CEpiR/war/data/"+projID+"/"+for
mID+".xls");
                                sheet.setParent(main);
                        }
                }

                private Boolean checkPriv(String username, String projID,
String formID) {
                        Connection conn =
DBConnection.getConnection();
                        Statement stmt;
                        try {
                                stmt = conn.createStatement();
                                PreparedStatement ps =
(PreparedStatement) conn.prepareStatement("SELECT * FROM user_priv
up INNER JOIN role r ON r.role_id = up.role_id WHERE " +
                                "up.username = ? AND up.proj_id =
?");
                                ps.setString(1, username);
                                ps.setInt(2, Integer.valueOf(projID));
                                ResultSet rset = ps.executeQuery();
                                if(rset.next()){
                                        stmt.close();
                                        conn.close();
                                        return true;
                                }
                                stmt.close();
                                conn.close();
                                return false;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                //formula box
                public void onCellFocused$sheet(CellEvent event) {
                        int row = event.getRow();
                        int col = event.getColumn();
                        rowIndex = event.getRow();
                        colIndex = event.getColumn();
                        focusCombobox.setText(sheet.getRowtitle(row) +
sheet.getColumntitle(col));

                        Worksheet sheet = this.sheet.getSelectedSheet();
                        currentCell = Utils.getCell(sheet, row, col);
                        currentRange = Ranges.range(sheet, row, col);

                formulaEditor.setText(currentRange.getEditText());
                }

                public void onOK$formulaEditor() {
                        Worksheet sheet = this.sheet.getSelectedSheet();
                        currentCell = Utils.getCell(sheet, rowIndex,
colIndex);
                        if(sheet.getRow(rowIndex)==null){
                                sheet.createRow(rowIndex);

                sheet.getRow(rowIndex).createCell(colIndex);
                                currentCell = Utils.getCell(sheet,
rowIndex, colIndex);
                        }else if(currentCell==null){

                sheet.getRow(rowIndex).createCell(colIndex);
                                currentCell = Utils.getCell(sheet,
rowIndex, colIndex);
                        }

                currentRange.setEditText(formulaEditor.getText());
                        this.sheet.focusTo(currentCell.getRowIndex(),
currentCell.getColumnIndex());
                }


                //edit menu functions
                public void onCellSelection$sheet() {
                        selectedRect = sheet.getSelection();
                }

                public void onClick$cut(){
                        onCut();
                }

                public void onClick$copy(){
                        onCopy();
                }

                public void onClick$paste(){
                        onPaste();
                }

                public void onCtrlKey$sheet(KeyEvent evt) {
                        if (evt.isCtrlKey()) {
                                switch (evt.getKeyCode()) {
                                case 'C':
                                        onCopy();
                                        break;
                                case 'V':
                                        onPaste();
                                        break;
                                case 'X':
                                        onCut();
                                        break;
                                }

                        }
                }

                private void onCopy() {
                        sheet.setAttribute(KEY_IS_CUT,
Boolean.valueOf(false));
                        setSource();
                        sheet.setHighlight(sheet.getSelection());
                }

                private void onCut() {
                        sheet.setAttribute(KEY_IS_CUT,
Boolean.valueOf(true));
                        setSource();

                        /* Highlight cut source */
                        sheet.setHighlight(sheet.getSelection());
                }
                private void setSource() {
                        Rect selectedRect = sheet.getSelection();
                        Range range =
Ranges.range(sheet.getSelectedSheet(),
                                        selectedRect.getTop(),
                                        selectedRect.getLeft(),
```

```
		selectedRect.getBottom(),
                                                selectedRect.getRight());
                        sheet.setAttribute(KEY_SOURCE_RANGE,
range);
                        sheet.setAttribute(KEY_SOURCE_RECT,
selectedRect);
                }

                private void onPaste() {
                        Range srcRange =
(Range)sheet.getAttribute(KEY_SOURCE_RANGE);
                        Rect srcRect =
(Rect)sheet.getAttribute(KEY_SOURCE_RECT);

                        if (srcRange != null && srcRect != null) {

                                int columnCount = srcRect.getRight()
- srcRect.getLeft();
                                int rowCount = srcRect.getBottom() -
srcRect.getTop();
                                Rect dstRect = sheet.getSelection();
                                int rowIndex = dstRect.getTop();
                                int columnIndex = dstRect.getLeft();

                                Range dst =
Ranges.range(sheet.getSelectedSheet(),
                                                                rowIndex,

                columnIndex,
                                                                rowIndex +
rowCount,

                columnIndex + columnCount);
                                srcRange.copy(dst);

                                clearHighlightIfNeeded();
                                clearCutRangeIfNeeded();
                        }
                }

                private void clearCutRangeIfNeeded() {
                        if (!isCut())
                                return;

                        Range srcRange =
(Range)sheet.getAttribute(KEY_SOURCE_RANGE);
                        srcRange.clearContents();
                        /* clear cell style*/
                        CellStyle defaultCellStyle =
sheet.getBook().getCellStyleAt((short)0);
                        srcRange.setStyle(defaultCellStyle);
                }

                private void clearHighlightIfNeeded() {
                        sheet.setHighlight(null);
                }

                private boolean isCut() {
                        Boolean isCut =
(Boolean)sheet.getAttribute(KEY_IS_CUT);
                        return isCut != null && isCut;
                }

                public void onClick$formulas(){
                        formula.doPopup();
                }

                public void onSelect$fxnList(SelectionEvent event){
                        Label format = new Label();
                        format.setParent(formula);

                format.setValue(functions.get(fxnList.getSelectedIndex()).getF
ormat());
                        Label details = new Label();
                        details.setParent(formula);

                details.setValue(functions.get(fxnList.getSelectedIndex()).getD
etails());
                }
}
```

ListFunctions.java

```
package data;

import java.util.ArrayList;
import java.util.List;

import org.zkoss.zk.ui.util.GenericForwardComposer;
import org.zkoss.zul.Listbox;

public class ListFunctions extends GenericForwardComposer{

        public List<Function> getList(){
                List<Function> functions = new
ArrayList<Function>();
                functions.add(new
Function("ABS","ABS(number)","Returns the absolute value of a
number"));
                functions.add(new
Function("ACOS","ACOS(number)","Returns the arccosine of a number"));
                functions.add(new
Function("ACOSH","ACOSH(number)","Returns the inverse hyperbolic
cosine of a number"));
                functions.add(new
Function("ASIN","ASIN(number)","Returns the arcsine of a number"));
                functions.add(new
Function("ASINH","ASINH(number)","Returns the inverse hyperbolic sine
of a number"));
                functions.add(new
Function("ATAN","ATAN(number)","Returns the arctangent of a
number"));
                functions.add(new
Function("ATANH","ATANH(number)","Returns the inverse hyperbolic
tangent of a number"));
                functions.add(new
Function("AVEDEV","AVEDEV(number1, number2,...)","Returns the
average of the absolute deviations of data points from their mean. AVEDEV
is a measure of the variability in a data set."));
                functions.add(new Function("AVERAGE",
"AVERAGE(number1, number2,...)", "Returns the average of its
arguments"));
                functions.add(new
Function("BINOMDIST","BINOMDIST(number_s, trials, probability_s,
cumulative)","Returns the individual term binomial distribution
probability"));
                functions.add(new
Function("CEILING","CEILING(number, significance)","Rounds a number
to the nearest integer or to the nearest multiple of significance"));
                functions.add(new
Function("COMBIN","COMBIN(number, number_chosen)","Returns the
number of combinations for a given number of objects"));
                functions.add(new
Function("COS","COS(number)","Returns the cosine of a number"));
                functions.add(new
Function("COSH","COSH(number)","Returns the hyperbolic cosine of a
number"));
                functions.add(new
Function("COUNT","COUNT(value1, value2,...)","Counts how many
numbers are in the list of arguments"));
                functions.add(new
Function("COUNTA","COUNTA(value1, value2,...)","Counts how many
values are in the list of arguments"));
                functions.add(new
Function("DEGREES","DEGREES(angle)","Converts radians to degrees"));
                functions.add(new
Function("EVEN","EVEN(number)","Rounds a number up to the nearest
even integer"));
                functions.add(new
Function("EXP","EXP(number)","Returns e raised to the power of a given
number"));
                functions.add(new
Function("FACT","FACT(number)","Returns the factorial of a number"));
                functions.add(new
Function("FACTDOUBLE","FACTDOUBLE(number)","Returns the
double factorial of a number"));
                functions.add(new
Function("FLOOR","FLOOR(number, significance)","Rounds a number
down, toward zero"));
```

```
                functions.add(new
Function("INT","INT(number)","Rounds a number down to the nearest
integer"));
                functions.add(new
Function("LN","LN(number)","Returns the natural logarithm of a
number"));
                functions.add(new
Function("LOG","LOG(number, base)","Returns the logarithm of a number
to a specified base"));
                functions.add(new
Function("LOG10","LOG10(number)","Returns the base-10 logarithm of a
number"));
                functions.add(new
Function("MAX","MAX(number1, number2,...)","Returns the maximum
value in a list of arguments"));
                functions.add(new
Function("MIN","MIN(number1, number2,...)","Returns the minimum value
in a list of arguments"));
                functions.add(new
Function("MOD","MOD(number, divisor)","Returns the remainder from
division"));
                functions.add(new
Function("ODD","ODD(number)","Rounds a number up to the nearest odd
integer"));
                functions.add(new Function("PI","PI( )","Returns
the value of pi"));
                functions.add(new
Function("POWER","POWER(number, power)","Returns the result of a
number raised to a power"));
                functions.add(new
Function("PRODUCT","PRODUCT(number1, number2, ...)","Multiplies its
arguments"));
                functions.add(new
Function("QUOTIENT","QUOTIENT(numerator, denominator)","Returns
the integer portion of a division"));
                functions.add(new
Function("RADIANS","RADIANS(angle)","Converts degrees to radians"));
                functions.add(new Function("RAND","RAND(
)","Returns a random number between 0 and 1"));
                functions.add(new
Function("RANDBETWEEN","RANDBETWEEN(bottom, top)","Returns a
random number between the numbers you specify"));
                functions.add(new
Function("ROUND","ROUND(number, num_digits)","Rounds a number to
a specified number of digits"));
                functions.add(new
Function("ROUNDDOWN","ROUNDDOWN(number,
num_digits)","Rounds a number down, toward zero"));
                functions.add(new
Function("ROUNDUP","ROUNDUP(number,num_digits)","Rounds a
number up, away from zero"));
                functions.add(new
Function("SIGN","SIGN(number)","Returns the sign of a number"));
                functions.add(new
Function("SIN","SIN(number)","Returns the sine of the given angle"));
                functions.add(new
Function("SINH","SINH(number)","Returns the hyperbolic sine of the given
angle"));
                functions.add(new
Function("SQRT","SQRT(number)","Returns a positive square root"));
                functions.add(new
Function("SUM","SUM(number1, number2, number3, ...)","Adds its
arguments"));
                functions.add(new
Function("STDEV","STDEV(number1, number2,...)","Estimates standard
deviation based on a sample"));
                functions.add(new
Function("TAN","TAN(number)","Returns the tangent of a number"));
                functions.add(new
Function("TANH","TANH(number)","Returns the hyperbolic tangent of a
number"));
                functions.add(new
Function("TRUNC","TRUNC(number, num_digits)","Truncates a number
to an integer"));
                return functions;
        }
}
```

XII. ACKNOWLEDGEMENT

First of all, I would like to thank the Lord for all the blessings He has given me and for giving me the strength and will to finish my SP. For without him, nothing is possible. Secondly, I would like to thank my parents and family for giving me the opportunity to study at UP Manila and for supporting me all the way. You have given me so much and I would like to like to express my gratitude to you by graduating.

I would also like to thank all my professors especially my adviser, Prof Richard Bryann Chua, for giving me guidance in my SP and for not letting me give up. I have learned so much in the past 4 years thanks to all of you.

Next, I would like to thank all my friends. Thank you Fathy and Joanna for always trying to help me with my SP even though you do not understand what I am doing. Thank you for your unwavering support and care and for always being there especially when I needed a friend to lean on. And oh, thanks for letting me stay over your house at those times when I still do not have internet at home :)

Thank you Arvi, Iehl, Jam, Kai, Mon, Tin and Yun for being my best friends in college. Thank you for all those FUN memories we have shared and for always believing in me. Thank you for being there for me ever since and for understanding me especially when I am in a bad mood. We started being friends since day 2 of college and look, we're still together. I could not thank you enough and I am so lucky to have you as my friends.

Thank you Alvin, Chard, Dawi, Maq, Quindoy, Ryan and other Dota Boys for being my secondary friends. Thank you for making me laugh every time with all your craziness and witty thoughts. But more importantly, thank you for being my bus-mates especially at late hours. I truly appreciate your companionship.

Thank you Kristin, Nikko, Pash and Ruth for still being my friends even though we do not spend much time together anymore. I really enjoy hanging out with you guys especially at *Ihawan*. Let's go there again sometime!

Thank you so much Block 12 Batch 2007 for making my college life a very memorable one! I hope to meet you again in the future and I wish you all the best in the world.