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

Fine Grained Access Security Module for DentISt: Dental Information System(DentISt) 3.0

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

Submitted by:

Angela Bernadette S. Jarabelo

April 2013

ACCEPTANCE SHEET

The Special Problem entitled "Fine Grained Access Security Module for DentISt: Dental Information System(DentISt) 3.0" prepared and submitted by Angela Bernadette S. Jarabelo 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. Perlita E. Gasmen, M.Sc. (*candidate*)	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____
6. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
7. Geoffrey A. Solano, Ph.D.(*candidate*)	_____	_____
8. Bernie B. Terrado, M.Sc. (*candidate*)	_____	_____

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

Avegail D. Carpio, 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

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

DentISt 3.0 is the third version of the electronic patient record in UPCD. Compare to the earlier version, the new one no longer uses OpenMRS as its EMR engine thus fixing the compatibility issues that was seen in the earlier version and give a more student-teacher approach.

As security in a system is one of the main concerns in converting from paper-based to electronic-based; DentISt has one module concerning the fine-grained access control of the system. To strengthen the access control, the usual placement of the access control in the application layer is change to the database layer and also using some HIPAA policies for sensitive data. Having the access control in the database layer helps us to have a multi-user database login and the use of stored procedure to avoid SQL injection. Some policies also provide us to provide three important security features- data integrity, confidentiality and availability and audit trail of the system.

Keywords: Dental Information System, Fine-grained access control, Dental Informatics

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Information systems are gradually replacing the paper-based records for years now and though not yet penetrated as the same way in other sectors, information technology has paved its way to the health industry. This paves way to the rise of the electronic medical record (EMR) in the medical field. There are a lot of open source nowadays. The popular ones are OpenEMR[1], OpenMRS[2], Tolven[3] and iTrust[4] [3].

Dental informatics is the application of information technology to improve dental practice, research, education and management [4]. While it fails in comparison compare to other medical fields' advancement, dental informatics has the potential to improve the efficiency, robustness and overall quality of dental care. Only a few dental information systems are available as of today and most of them are not free and are expensive.

As the leading school in the Philippines, the use of dental record system in the University of the Philippines Manila College of Dentistry will further improve the training of students both academically and clinically. A robust, effective and secure dental system will also help the workload of clinicians in accessing patient records.

The first dental system used in UPCD was created by UP Manila Computer Science students taking up Software Engineering Course. As it is the first, problems were inevitable and the system contained bugs. As an attempt to fix the problem, UPCD staffs have tried to reformat the computer but ended up erasing the system data and the software data as well [5]. Using the free and open source electronic medical record (EMR) system, OpenMRS, Aurielle Lee

[1] `http://www.open-emr.org/`
[2] `http://openmrs.org/`
[3] `http://home.tolven.org/`
[4] `http://www.itrust.in/`

developed a new dental information system called Open DentIS in 2011. Open DentIS makes use of the OpenMRS concept feature by creating a standardized dental lexicon based on UPCD terminologies. In 2012, Christina Balsita continued Aurielle Lee's Open DentIS to form the second version called DentISt [5]. Enhancement of the functionalities of the system was made specifically fixing the errors and loading of the dental chart [6].

B. Statement of the Problem

Though OpenMRS is a free and open source electronic medical record system, it lacks resources and documentation online and not widely used in the Philippines. Therefore developing a new module is a tedious process. Deployment of OpenMRS is not easy as the installation of different softwares and applications that will be needed is a long and complicated procedure. There are also some compatibility issues which caused errors on some important functionalities of the system. The system is not able to run properly in newer versions of web browser (Mozilla Firefox version 11.0). The dental chart and other main features of DentISt cannot be updated due to this. Additionally, there are some issues of compatibility between OpenMRS and versions of Apache Tomcat later than 6.0.29. It returns errors when updating and creating patient dental records. Currently, OpenMRS modules that rely on certain custom expression language functions throw a java.lang.ClassNotFoundException exception [7].

OpenMRS was created with doctors as the primary users of the system thus its usability and effectiveness may be affected. Additionally, considering DentISt's university setting, students are also allowed to use the system. Therefore their accomplishment report entries must be approved first by their respective professor before being considered a qualified data input. However, as OpenMRS considers its users as eligible doctors, adding a student-teacher concept to the system is hard. As deployment and adding new modules and functionalities in the system are complicated and difficult, further study and development may

come as a major problem in the future.

To comply with the HIPAA (Health Insurance Portability and Accountability Act) standards regarding data integrity, confidentiality and availability, OpenMRS uses the Role Based Access Control Model where data access is based on a subject's role description. However with the growing complexity of health information systems, the traditional RBAC model implemented in the OpenMRS cannot secure data protection and security problems in compliance of HIPAA for separation of duty/minimal privilege principle, encryption of sensitive data according to HIPAA standards and core RBAC module. As RBAC is implemented in the OpenMRS at its core application, enhancing the access control model requires re-working the architecture of the application. Assuming the proper delegation of access control policy to the application systems are implemented, the database tends to trust these different application systems which create the loopholes in the security and protection of the system. Also, access control in the application layer considers a one- database user approach in the OpenMRS. A problem on having this one-database user approach is every query runs with the super- user privileges thus violating the minimum privilege principle and a result to a potential damage due to malicious access.

C. Objectives of the Study

To create the Dental Information System (DentISt) 3.0 which is reengineered from DentISt 2.0 but no longer built on the OpenMRS platform with the following major revisions:

1. the roles and the functionalities in DentISt 3.0 are:
    (a) to allow the student clinician of a section to
        i. manage patient records of the patient assigned to the section
        ii. manage his consultation appointments in the section
        iii. do general query for patients

(b) to allow the faculty of a section to

i. perform all the functionalities of a student clinician

ii. confirm the new entries and updates of a student clinician on a patient record

iii. view the student clinicians and faculty of the section

iv. view the list of consultation appointment schedules of the student clinicians and faculty of the section

v. view statistics of the section

(c) to allow the system administrator to

i. manage user accounts, roles and privileges

ii. manage sections, privileges of the users in the section, and users in the section

iii. view statistics for the whole UP College of Dentistry

2. the security design of DentISt 3.0 will follow the HIPAA recommendations with fine-grained database access policy done from the database management system (DBMS) instead of in the application

D. Significance of the Project

As the popularity of electronic medical record systems increases, a dental information system can help make a centralized, accessible and secure patient information system. Not only does it reduce costs, it minimizes incorrect medical records (by handwriting) and also store the latest dental health information.

Reengineering the previous system to an EMR without the OpenMRS platform can help future developers in their study and implementation of new ideas to the dental system. Deployment and adding new modules and functionalities in the system wouldn't be a problem as well. Codes which have been previously been difficult and complicated because of its OpenMRS architecture become more understandable and modification and changes become less time

consuming. Since the OpenMRS development methodology will not anymore be implemented in the system, it will be easier to make DentISt more usable to non-doctors and student-teacher approach for the system becomes possible to implement by first having teacher approval before new entry from student can be considered final.

As HIPAA standards concern about the data integrity, confidentiality and availability, implementing its security policies in the system can protect data from being accessed by unauthorized users, provide anonymity to listed sensitive health information by encryption and offer the least privilege principle to limit access of users.

To maintain the same controls in various applications, placing the access control in the database layer instead in the usual application layer can provide a multi- database user approach where access control policies are implemented to every user and applications ensuring that the database can know the user's identity and its permissions. Also since application layer no longer needs to issue SQL codes in performing database access, the relevance of stored procedures allows better fine grained access control and allows application to be no longer dependent on the table structure thus decreased the time consumed in production and SQL injection attacks.

E. Scope and Limitations

1. Security policies are based on HIPAA (Health Insurance Portability and Accountability Act) of the USA and the following are implemented in DentISt 3.0:
   (a) All data considered sensitive must be encrypted when stored in the database
   (b) Access to sensitive data or PHI (protected health information) must be:

i. Authorized by a designated HIPAA Officer who knows the policies.

ii. Authenticated by logging in as a unique user using own username and password.

iii. Revoked when such access is no longer necessary to perform authorized job responsibilities.

iv. Consistent with the minimum privilege principle.

(c) Requires re-authentication, after a period of inactivity

(d) Opens sessions are automatically disconnected after a period of inactivity

(e) Allows only authorized users to have physical access into facilities where PHI is stored and resides.

(f) Regular audit of information system activity.

2. Sensitive data are based on HIPAA standards and are the following:

   (a) Patient Name

   (b) Address

   (c) Birth Date

   (d) Telephone number

3. The initial roles that will be included are:

   (a) Student clinician

   (b) Faculty

   (c) System administrator

4. The initial sections that will be included are:

   (a) Oral diagnosis

   (b) Oral medicine

   (c) Prosthodontrics

(d) Operative dentistry

5. Clinicians in Oral Diagnosis roles are the only one who can add patient records and also the only one to add/edit the following forms:
   (a) Patient Information Form
   (b) Physical Assessment Form
   (c) Vital Signs Forms
   (d) Dental History Forms
   (e) Medical History Forms
   (f) Social History Forms
   (g) Soft Tissue Exam Form
   (h) Radiographic Exam Form
   (i) Treatment Plan Form
6. The general query for patients can be done with the following criteria:
   (a) Male patients
   (b) Female patients
   (c) Patients within an age group
   (d) Patients having a certain job
   (e) Patients living in an area
   (f) Patients needing a specific treatment
   (g) Patients with specific dental condition
7. The statistics that will be generated for a specified time period includes:
   (a) number of patients with specific dental condition
   (b) number of patients who underwent a particular treatment
   (c) number of patients needing a specific treatment
   (d) number of male patients
   (e) number of female patients

(f) number of patients living in a particular location

(g) number of patients treated/registered over the time period

F. Assumptions

1. User account without any role cannot do anything in the system.
2. There will be separate roles for the student clinician and faculty for each section.
3. No user account will be assigned both student clinician and faculty roles at the same time.
4. Entries input by users with faculty role are automatically final and don't need any approval.
5. The clinician in attendant of the patient may not be the clinician in the next encounter.

II. Review of Related Literature

As the application of computer science crosses over to various fields, even the field of dental medicine has shown interest in applying technology in this medical track. Biomedical informatics is a maturing discipline of significant scale and scope. One of its sub disciplines in particular, dental informatics, though small, is emerging as a growing discipline [1]. Dental informatics is an information science that helps improve the dental practices, research, education and management [4].

EMR (electronic medical record) systems replace the paper-based records to provide single, shareable, up to date information system. These systems have helped reduce the medical errors, facilitating the electronic transfer of patient records across facilities, and improving the efficiency of clinicians [8]. The use of EMR has also paved its way to dental offices.

As investment in health care information technology is now greater than any other sector, the use of Open Source Software (OSS) has significantly showed growth. Its popularity is caused by its ability to be integrated and interoperated with other systems and most especially the lowered cost for development, acquisition, and maintenance [9]. Some examples of popular open sourced electronic medical records include OpenMRS, OpenEMR, iTrust and Tolven [3].

Despite its potential, health care providers are slow to adapt health care information system. As of 2009, only 17 percent of U.S. physicians use either a minimally functional or a comprehensive electronic records system [10]. EMR implementation has often failed mainly because of physician's inefficiency in using the system. System designed without the users as the center of the design process causes its usability problems [11]. Usability is one of the factors in adopting a system.

OpenMRS, an open source electronic medical record system, started back on 2004 from the idea of Paul Biondich and Bruke Mamlin. As OpenMRS is

free, it is being used as a promising electronic medical record for developing countries. OpenMRS's main components are based on its conceptual database structure, independent of required information types, and having a web application that enables access through a user interface [12]. However, OpenMRS has little documentation and implementation in the Philippines thus deployment and adding new functionalities and modules in the stated EMR cause difficulty and problems in future developments and enhancements of the system [6]. OpenMRS also has a problem in terms of its usability as it considers the fact its users are all doctors.

Using electronic medical records versus paper based records can help us to better secure and keep track of information. However, data security has been mostly overlooked in information system. As there are broad and diverse set of controls applicable to data access, it is presumed that information has been protected. However, that reason alone has taken away the centralized access control of the system and made it accessible for serious attacks and weakness in the database [8]. Even if identifiers such as names are encrypted, the adversary can use linking, homogeneity and background attacks to re-identify individual data records or sensitive information (PHI) [13]. Protection of electronic PHI (protected health information) revolves around these 3 concepts: confidentiality or protection of data from unauthorized disclosure, integrity or prevention from unauthorized data access, and availability or the identification of and recovery from hardware and software errors or malicious activity resulting in the denial of data availability [14].

Health information is still not homogenous and may cause some constraints on the processing of data [15]. HIPAA (Health Insurance Portability and Accountability Act) has been passed in the USA in August 1996 as an attempt to reduce healthcare costs, improve efficiency, simplification of administrative procedures and mandate healthcare organizations to implement standard formats health systems [16]. Enforcing HIPAA standards can ensure every access to a

system and its resources controlled and only authorized accesses can take place [16]. Traditional access control includes Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role Based Access Control (RBAC). Particularly, RBAC is widely acknowledged for providing security and administration of data access by users of a system and is mostly used in following the HIPAA standards. However, because of the growing complexity of health information, traditional access controls lack support of privacy-related policies [17]. OpenMRS, for example, has failed the evaluation criteria that include using the HIPAA requirements. Specifically, OpenMRS failed to have a separation of duty and emergency access procedures [3].

RBAC in a system can be implemented as a module or core application. Module allows the modification to a more robust security model whereas the core requires the re-working of the design/architecture of the whole system. OpenMRS implements RBAC to its system as a core application thus changes to the security policies means re-working the code. This increases time consumed for production and may result to security risks [3]. Access control in the application level can overlook some possible loopholes that can cause a breakthrough to its security policies and open system to malicious attacks [18]. Although many advances have been developed to enhance a more secure application, trusting the application, which was developed under time constraint, to protect and secure data present a large risk to the database [19]. Also existing scenarios in database, use a single user acts as both the data owner and the querier. However, most practical applications do not serve only one user. Having a one database user approach in a system, like the OpenMRS, causes the database to have its only user as the database owner itself thus violating the RBAC principle of minimal privilege and inconsistency of access between users [20].

Implementing the access control mechanism to the database layer instead of the application layer solves the problems of re-coding the architecture and

helps minimize the possible security loopholes overlooked because of the large amount of code [18]. Access control in the database also ensures that security policies are consistently applied to every user and application thus allowing a multi-database user approach [21].

Stored procedures can be use to break the dependency of the application to data structure thus prevents SQL injection attacks. Stored procedures, which have an irreplaceable importance in any database application, also allows better fine grained access control, data integrity and decrease time consumed in production [22]. Another form of fine grained authorization can be implemented by using views, tabled value functions, that enables to determine for a user the only part of the database that interests her. Views provides user-specific parameter values [23].

As MySQL has turned into an open core database, PostgreSQL always has been and still remains a true open source database. Users can freely suggest, create, submit, and have features accepted without interference or delay from a commercial entity. Security has been the main focus of PostgreSQL, which contains tools and options for securing data integrity, stability and accesses to the database [24]. As PostgreSQL focuses on security, security enhanced (SE)- PostgreSQL is introduced as a built in enhancement or optional access control feature that provides a mandatory access control (MAC) based on the Security-Enhanced Linux (SELinux) security policy [25].

Using [21] correction criteria to the PostgreSQL database can enforce fine grained access control that satisfy the evaluation criteria of being sound, secure and maximum. The newly enhanced query evaluation engine of the PostgreSQL addresses the issues concern on access control at the application and allows a FGAC access depending on the security policy at row or even cell level scheme [26].

III. Theoretical Framework

A. Dental Informatics

Dental informatics is an information science that helps improve the dental practices, research, education and management [4]. It is a small growing sub discipline of biomedical informatics that has been developed into a research discipline of significant scale and scope [1].

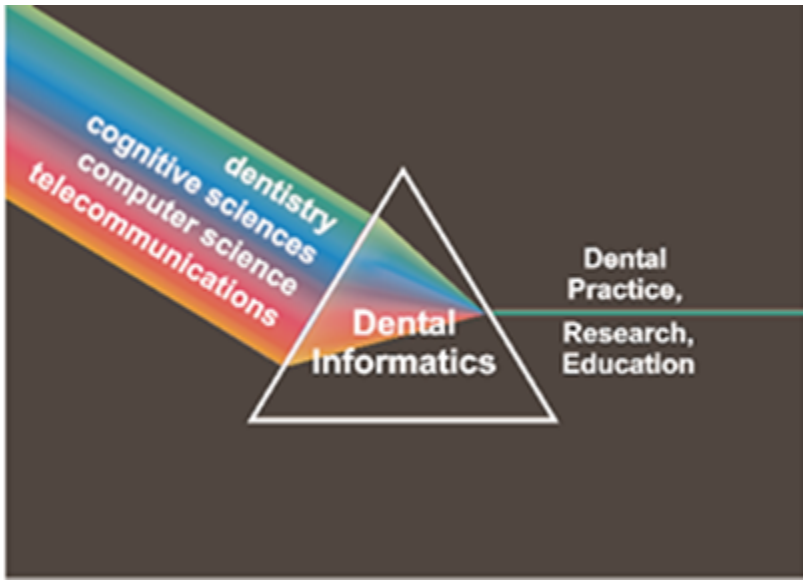


Figure 1: Dental informatics combines its methodological foundations to address problems in practice, research, and education [1].

Figure 1 shows how combination of the one or more components of information science (cognitive sciences, computer science and telecommunications) to the domain area of dentistry to develop solutions in dental practice, research and education-dental informatics.

As defined by the Wolrd Health Organization(WHO), dentistry is the science and art of preventing, diagnosing and treating diseases, injuries and malformations of the teeth, jaws, and mouth [27].

Computer science is a discipline that involves the understanding and design of computers and computational processes. The emphasis of computer science

is on how it is represented, process, manipulated, and managed in computer systems.

Cognitive science is a research area that draws on several fields (such as psychology, artificial intelligence, linguistics, and philosophy) to develop theories of perception, thinking, and learning. The central hypothesis of cognitive science is that thinking can best be understood in terms of representational structures in the mind and computational procedures that operate on those structures. Since biomedicine is sated with complex cognitive operations (such as diagnosis, treatment planning, and evaluation), cognitive science represents a significant component in dental informatics.

Telecommunications is the science that deals with communication at a distance. Telecommunication concerns on how computers communicate with each other, how communication traffic is routed, how bandwidth is used most efficiently, and how communication can be kept secure.

B. UP College of Dentistry

The UP College of Dentistry was first established as a Dept. Of Dentistry of the College of Medicine and Surgery on February 8, 1915. Upon the recommendation of the late Dean Antonio G. Sison of the College of Medicine, the Board of Regents of the University passed a resolution changing the status of the School of Dentistry to an independent unit as the College of Dentistry on October 21, 1948 [28].

As a newly established independent unit of the University of the Philippines, the College of Dentistry envisions itself to be the country's premier academic institution providing quality dental education, training, research and service characterized by global competence, social sensitivity and responsible leadership in the continuous pursuit of excellence for the service of God and the nation [28].

1. UPCD Structure

The UP College of Dentistry consists of different sections where clinicians work and perform treatments on patients and the following sections are [6]:

- Oral Diagnosis
- Oral Medicine - Periodontics, Oral Surgery, Endodontics
- Prosthodontics - Removable Prosthodontics, Fixed Partial Prosthodontics
- Operative Dentistry - Orthodontics, Pedodontics, Restorative Dentistry

Oral diagnosis is in charge of accepting patients and gathering the patient information. Oral Medicine is concerned with the oral health care of patients. The Prosthodontics section specializes with the diagnosis, treatment planning, rehabilitation and maintenance of the oral function, comfort, appearance and health of patients with clinical conditions associated with missing or deficient teeth and/or oral and maxillofacial tissues using biocompatible substitutes. Operative dentistry focuses primarily on the diagnosis, prevention, treatment and prognosis of diseases or trauma to teeth.

Patients may also be endorsed to outside sections or clinics that can perform specific exams or treatments not covered by UPCD [6].

C. Health Insurance Portability and Accountability Act (HIPAA)

Health Insurance Portability and Accountability Act (HIPAA) is an act signed by Bill Clinton and passed in August 1996. Two main goals of HIPAA include the improvement in system effectiveness and protection of confidentiality. HIPAA is an attempt to reduce health costs, to call for simplification of administrative procedures and to mandate health care organizations to implement standard formats for all transactions [16].

1. HIPAA Privacy Rules pertaining to protected health information (PHI) [16]:
   (a) Role-based access controls to ensure that only appropriate people have access to the minimum necessary PHI to perform their job responsibilities
   (b) Safeguards to ensure that the PHI does not get altered or destroyed in an unauthorized manner.
   (c) A formal process for ending a person's employment or a user's access so that inappropriate access to PHI does not occur.
   (d) Secured control of media (such as papers, diskettes, tapes, laptops, personal digital assistants, CDs, etc.) containing PHI to ensure that unauthorized use or disclosure does not occur.
   (e) Allowing only authorized persons to have physical access into your facilities where PHI is stored and resides.
   (f) A well-defined change control process.
   (g) Regular audits of information system activity.
   (h) PHI sent across open systems, such as the Internet, is protected from unauthorized access.
2. HIPAA Statements from the Security Rule that deals specifically with electronic protected health information (EPHI) [29]:
   (a) Access Control- Implement technical policies and procedures for electronic information systems that maintain electronic protected health information to allow access only to those persons or software programs that have been granted access rights.
      i. Unique User Identification- Assign a unique name and/or number for identifying and tracking user identity.
      ii. Automatic Log-off- Implement electronic procedures that terminate an electronic session after a predetermined time of inactivity.

iii. Emergency Access Procedure: Establish (and implement as needed) procedures for obtaining necessary electronic protected health information during an emergency.

iv. Encryption and Decryption- Implement a mechanism to encrypt and decrypt electronic protected health Information.

(b) Audit Control- Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information.

(c) Integrity- Implement policies and procedures to protect electronic protected health information from improper alteration or destruction.

The overall goals of the security rule revolve around the confidentiality, integrity, and availability of data.

D. Security Model

Security policy outlines about how data is accessed, what level of security is required, and what actions should be taken when these requirements are not met. To apply this security policy to a system, a security model provides a deeper design and analysis of the requirements necessary to properly support and implement it. A security model presents a framework of the abstract goals of the policy to information systems by specifying explicit data structures and techniques necessary to enforce security policy [30].

As information security is made up of data confidentiality, availability and integrity, access control is the security model's level of granularity for defining security policy [31]. Access control is critical to preserving the confidentiality, availability and integrity of information. Traditional access control, who uses single-user privilege and implements non-cryptography technique, are the following [16]:

- Discretionary Access Control (DAC)- Access Control Lists

- Mandatory Access Control (MAC)- Bell Lapadula model, Biba model, Chinese wall model
- Role-based Access Control (RBAC)

Because of the growing complexity of health information, traditional access controls lack support of privacy-related policies. In order to address the shortcoming, different extensions and enhancement are created to support the security policies [17].

1. Fine Grained Access Control

As compared to the traditional access controls that focus on limiting data access at the table level, fine grained access control provides a way to restrict data access at the row or even at the cell level. Fine grained access control, or also known as content-based access control, is determined based on the content of the row where the data is located and the accessibility of a data item specified. Fine grained access control also provides data access to respect individual preferences and comply with many enacted privacy laws [21].

Fine grained authorization can be improved more by implementing it to the database layer. FGAC at the database level solves the drawbacks found in the application layer such as [23]:

- Authorization checks that are distributed over a large body of code, requiring more programmer effort, and increasing the chances of security problems due to programmer or design errors
- Single database user log in as every query runs with the super user privileges with respect to all data managed by the application and causes a potential risk for malicious attacks
- Information from different organizations may resides in one application service provider model thus can provide organizations who are not willing to trust the application to protect its data.

2. Fine Grained Access Control with Stored Procedures

Stored procedure is a set of Structured Query Language SQL statements with an assigned name that's stored in the database in compiled form so that it can be shared by a number of programs. By having procedures that handle the database work for your interface, you eliminate the need to modify the application source code to improve a query's performance. Stored procedure can also preserve data integrity and improve productivity [22].

Without having to rewrite the same statements to the application, stored procedures applies several policies to the table thus helps to provide a fine grained access control in the database level [32].

3. Fine Grained Access Control Features of PostgreSQL

Security has been the main focus of PostgreSQL, which contains tools and options for securing data integrity, stability and accesses to the database. In PostgreSQL, users can freely suggest, create, submit, and have features accepted without interference or delay from a commercial entity. [24]. As PostgresSQL focuses on security, a built in enhancement or an optional access control feature was created called Security Enhanced (SE)- PostgreSQL. SE-Postgresql applies a fine grained mandatory access control (MAC) for database objects and makes access control decisions based on SELinux security policy. SE-Postgresql with the SELinux uses a client-server approach to make access control assessment [25].

PostgreSQL also features on rich privilege management. Host based access defines what authentication and verification method for users (or roles) with Login Attribute and who wants to connect to a database. Once the user is provided to have a right to connect to the database, the user must also have the usage on the schema before looking at the rights or grants of the user to access or modify the contents of the relation. Before giving an ownership access, PostgreSQL provides a GRANT (to give right) and REVOKE (to remove right)

on the specified target [2].

Enforcing a fine grained access control in the database level must satisfy three evaluation criteria of being sound, secure and maximum. The algorithm is sound when the answer returned is the same as the result returned without the access control, secure when it does not leak the information not allowed by the policy, and maximum when it returns as much information as possible, while satisfying the first two property [21]. FGAC on PostgreSQL has provided a better access control mechanism and at the same time comparable to the database without the FGAC implementation [26].

E. XACML

XACML stands for eXtensible Access Control Markup Language created by the standardized effort of OASIS. XACML is designed to express access control policies in XML as a way to challenge the problem of managing distributed and heterogeneous network system that needs an enforcement of security policies. XACML may start of as a policy set (set of policies) or policy. XACML policy starts with a root element of Policy which contains a set of rules, a target, and a rule-combining algorithm. A target consists of the set of resources, subjects and actions that must be applied to a given request. Meanwhile, a rule is a set of condition that applies a more specified or fine-grained access given the target and the rule-combining algorithm that speaks about whether it permits or deny a certain access [33]. Figure 2 shows the structure of an XACML

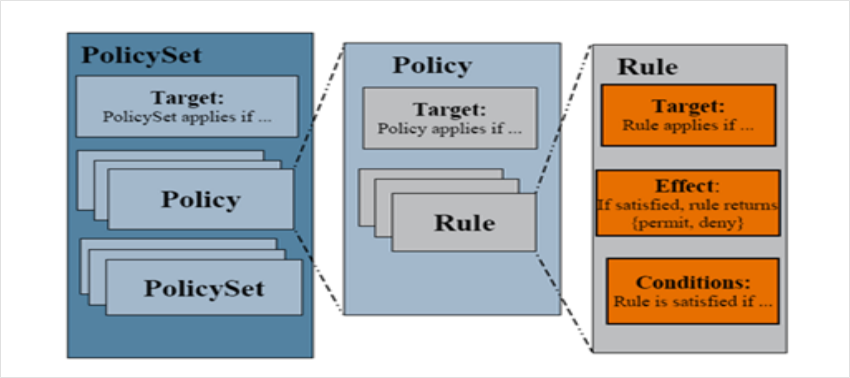


Figure 2: XACML Structure

Figure 3 shows a simple example of a XACML for a user role- teacher. XACML starts with a policy tag that contains the target element containing the three important elements of subject, resource and action. Subject has a string attribute value of teacher. Action has a string attribute value of edit. Finally, resource has a string attribute value of sub1. Generally, the exampled xacml means a teacher has a privilege to edit the sub1 data.

```
<Policy  xmlns="urn:oasis:names:tc:xacml:1.0:policy"
         RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">teacher</AttributeValue>
          <SubjectAttributeDesignator AttributeId="db_acad_users_user_role"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>

    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">edit</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string" />
        </ActionMatch>
      </Action>
    </Actions>

    <Resources>

      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">sub1</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
        </ResourceMatch>
      </Resource>

  </Target>
</Policy>
```

Figure 3: XACML Example

Applying XACML to the database, subject and resource can refer to existing attributes in the database. Enhancing database access control using XACML policy can be divided into three steps. First step is policy processing where we extract the subjects, resources and actions and use it as the users, tables and query actions, respectively. Second step is policy transformation where we can transform the three parts obtained in the first step to its specified SQL GRANTS. Finally, third step is the policy enforcement where the GRANT statements are now enforced in the database [34].

IV. Design and Implementation

A. Context Diagram

The DentISt 3.0 will have four main types of roles- the System Administrator, the Workflow Administrator, the Faculty, and the Student Clinician. The context diagram is shown below in Figure 4.

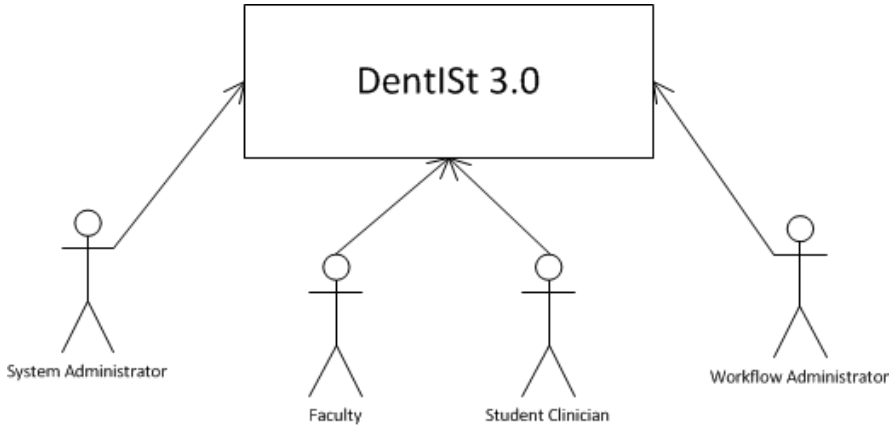


Figure 4: Context Diagram of DentISt

B. Use Case Diagrams

Student clinician can manage patient record and perform workflow tasks. On the other hand, faculty can perform student clinician functionalities and view the student clinicians and faculty of the section. Both the faculty and the system administrator can view statistics. Workflow administrator can manage system workflow. Finally, the system administrator has the privilege to manage the user accounts, roles and sections. Figure 5 shows the top level use case diagram of the DentISt where use cases in gray indicate that they are for the workflow management module, use cases in white for the fine grained access control module and use cases in black for both.

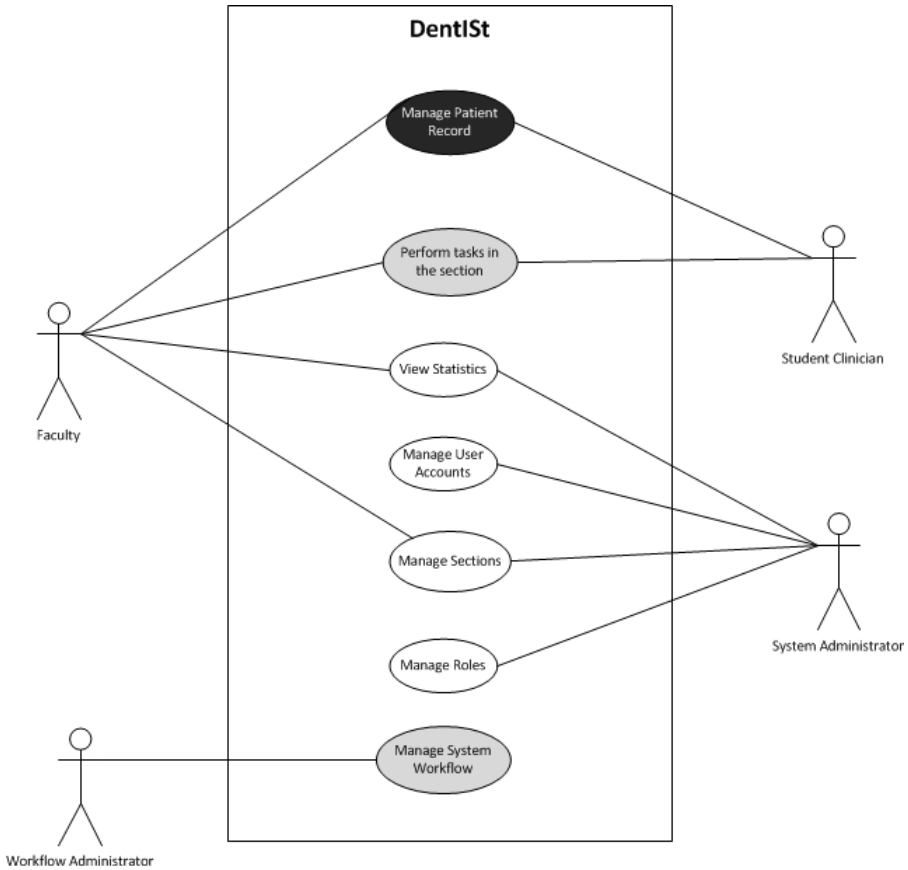


Figure 5: TopLevelUseCase of DentISt

1. View Statistics

System administrator and faculty can view statistics based on the field set by the system administrator on their query. Figure 6 shows the View Statistics Use Case Diagram of the System Administrator and Faculty in DentISt.

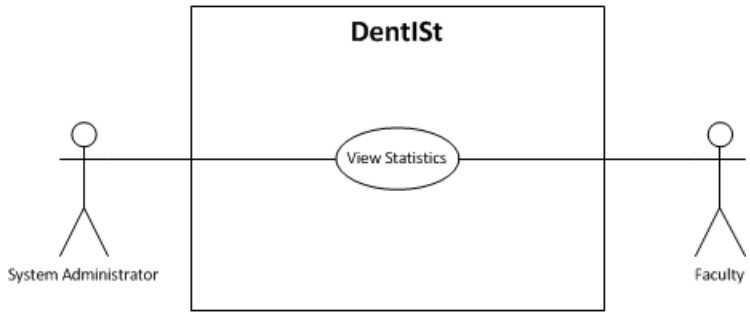


Figure 6: View Statistics Use Case Diagram of the System Administrator and Faculty of DentISt

Activity Diagram of the View Statistics are shown Figure 7.

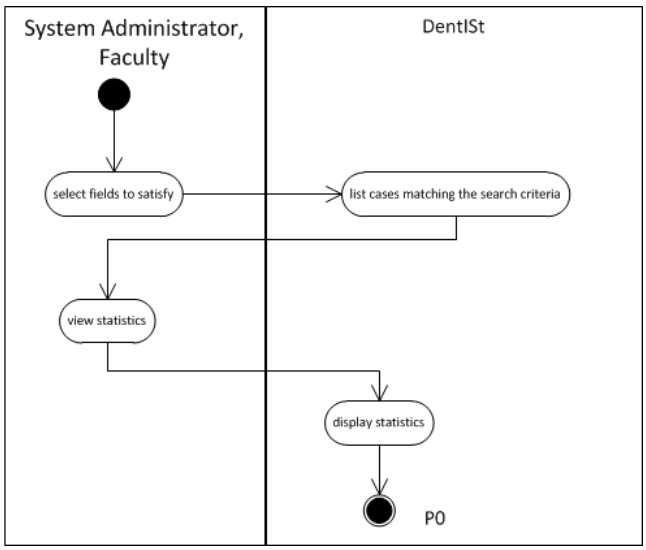


Figure 7: Search For Patients according to Specified Parameters Activity Diagram of DentISt

2. Manage Patient Records

The Manage Patient Records Use Case involves the student clinician and faculty role. Both can add and edit patient records in the section. They also have a privilege to search and view patients meeting different criterion. Criterion includes the sex, age, occupation, address, chief complaints, dental status chart and needed services. Use case in gray indicates it is part of the workflow module. Figure 8 shows its user case diagram. Figure 9 shows the Query Patient Use Case.

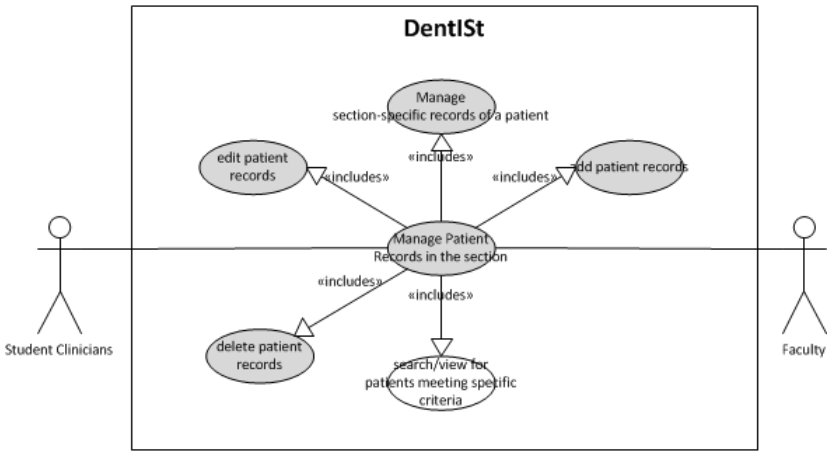


Figure 8: Manage Patient Records Use Case Diagram of Student Clinicians in other sections and Clinicians in Oral Diagnosis of DentISt

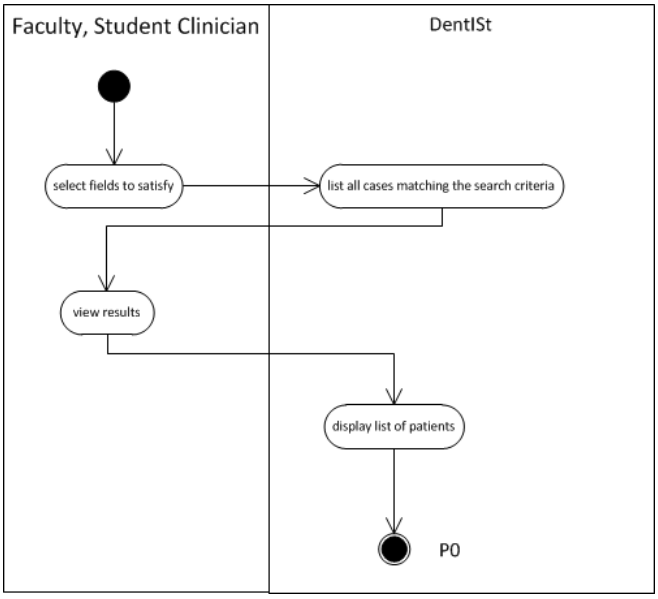


Figure 9: Search and View Patient Record Activity Diagram of DentISt

3. Manage User Accounts

System administrator handles the management of user accounts. System administrator can add, edit and delete user accounts and add and remove privileges according to assigned role of faculty, student clinicians or system administrator. The SA can also search and view the user accounts. Figure 10 shows the Manage User Accounts Use Case Diagram.

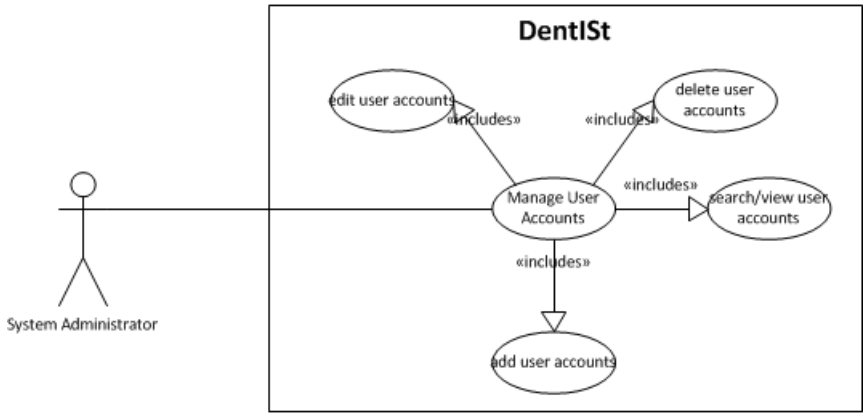


Figure 10: Manage User Accounts Use Case Diagram of the System Administrator

Figures 11, 12, 13 and 14 show the Activity Diagrams for Manage User

Account.

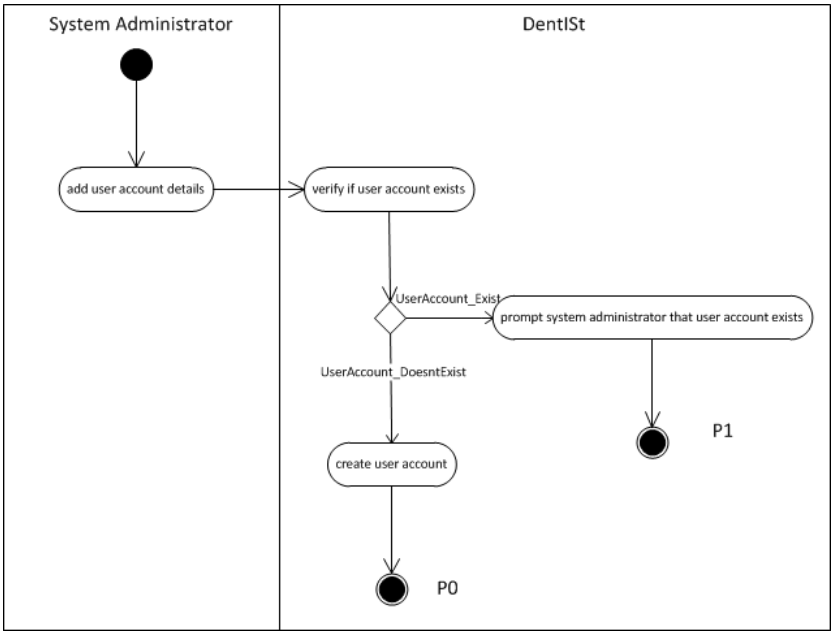


Figure 11: Add User Accounts Activity Diagram of DentISt

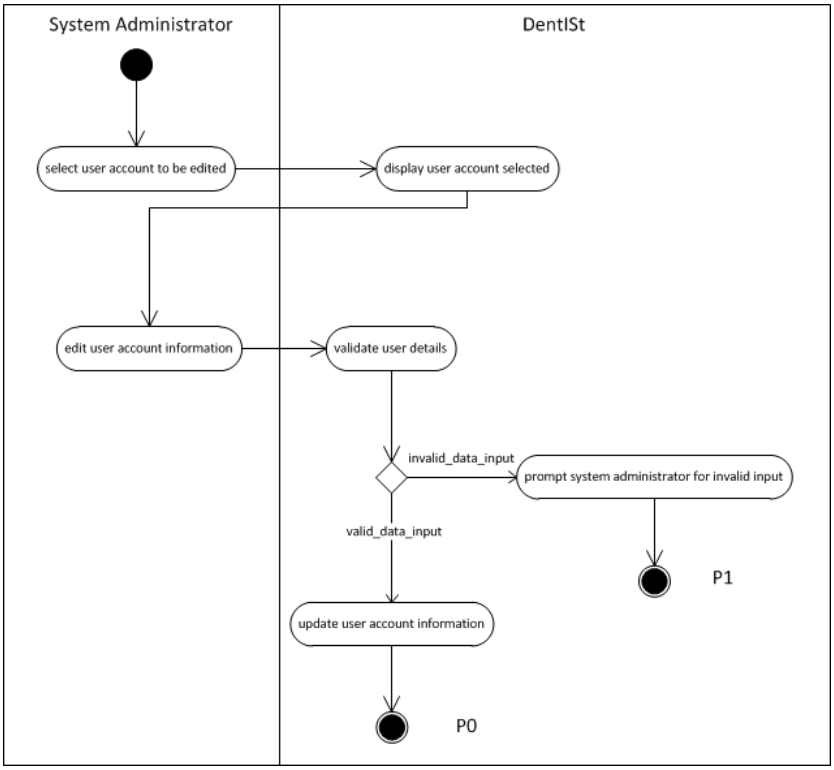


Figure 12: Edit User Accounts Activity Diagram of DentISt

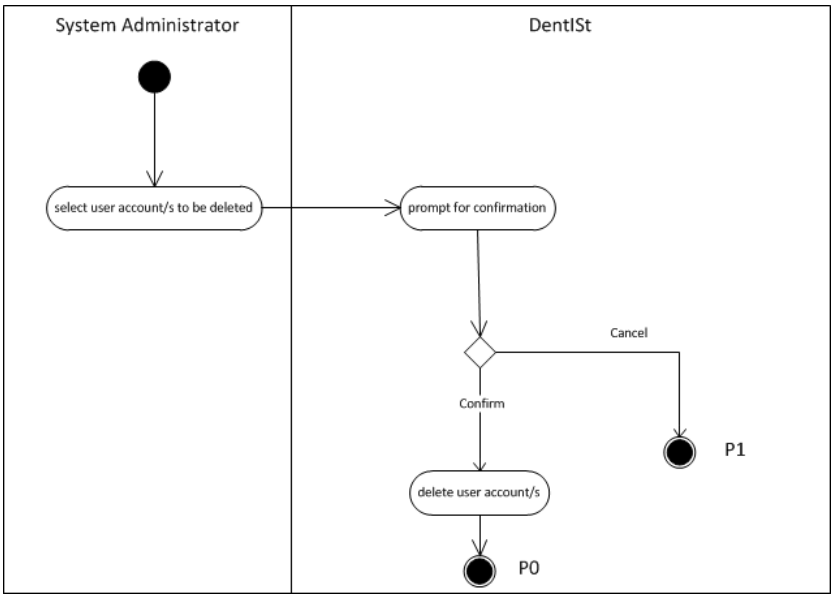


Figure 13: Delete User Accounts Activity Diagram of DentISt

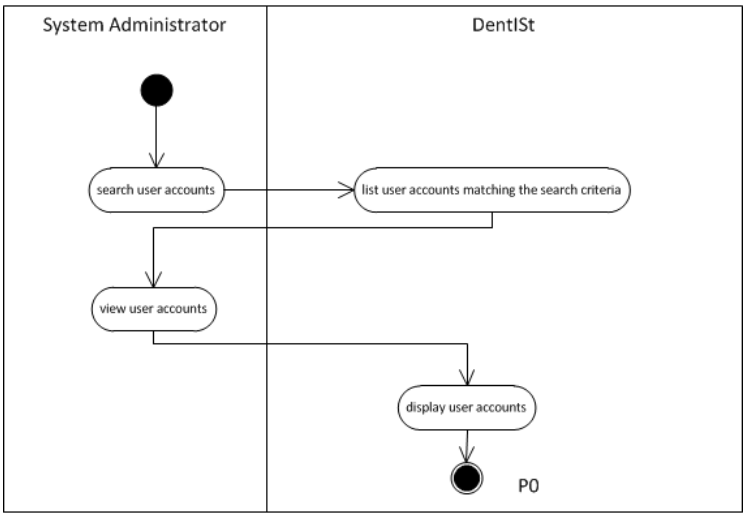


Figure 14: Search and View User Accounts Activity Diagram of DentISt

4. Manage Roles

The Manage Roles Use Case Diagrams involves only the system administrator who can add, edit and delete roles. Figure 15 shows its Use Case Diagram.

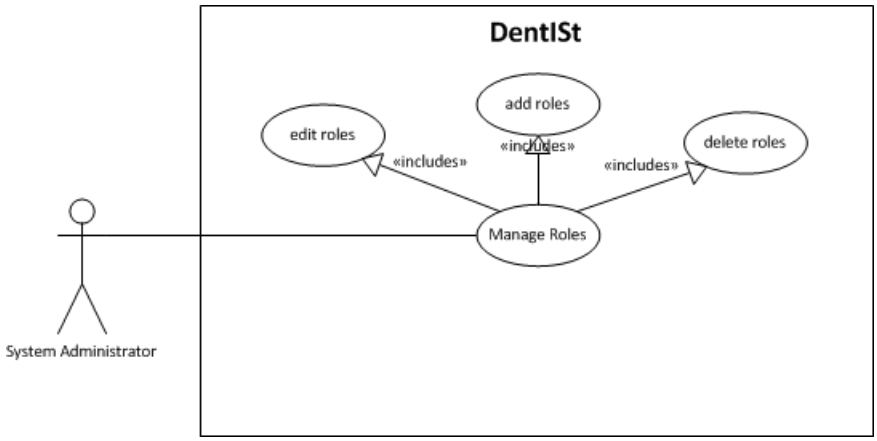


Figure 15: Manage User Accounts Use Case Diagram of the System Administrator

Figure 16, 17 and 18 show the Activity diagram of the Manage Roles.

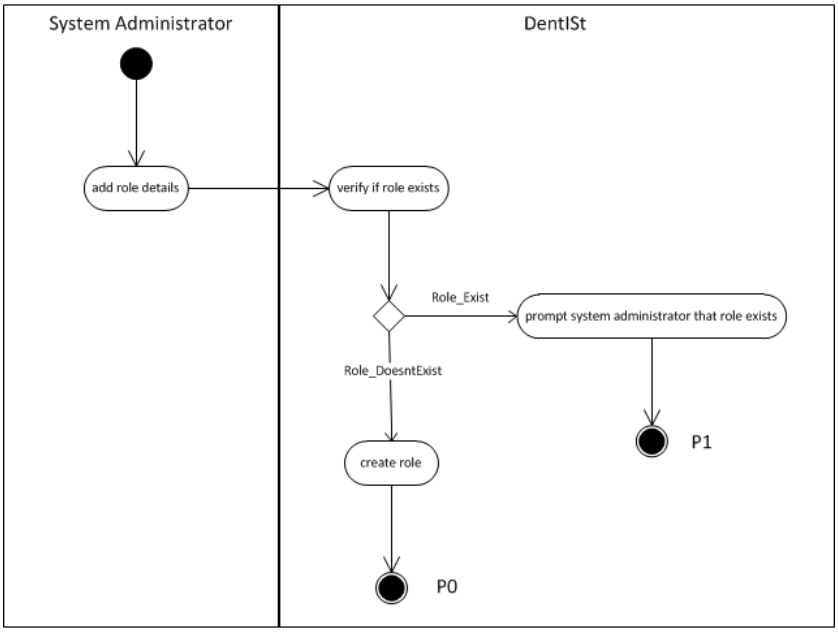


Figure 16: Add Roles Activity Diagram of DentISt

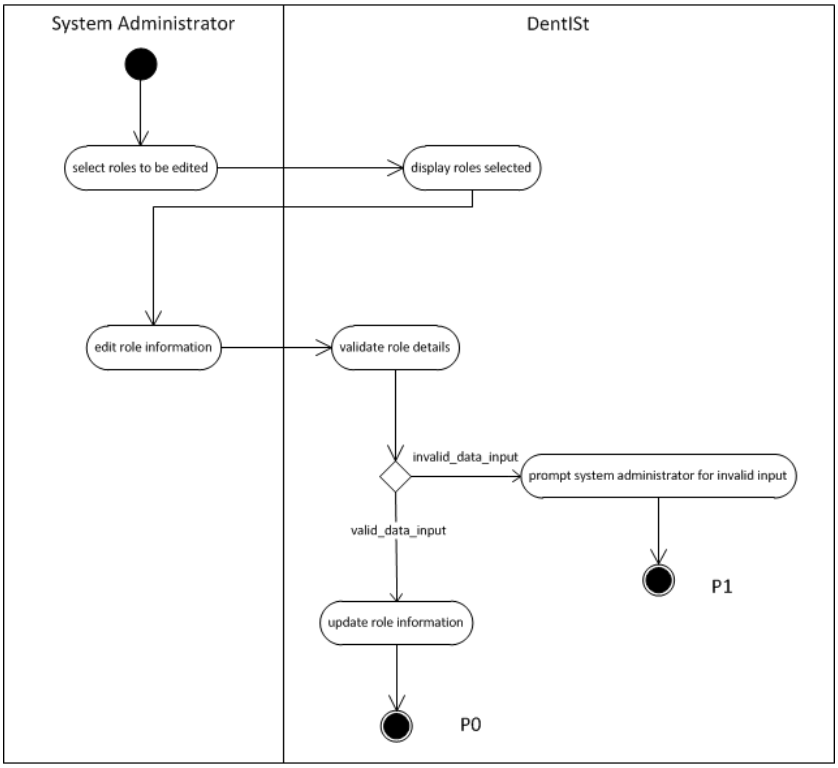


Figure 17: Edit Roles Activity Diagram of DentISt

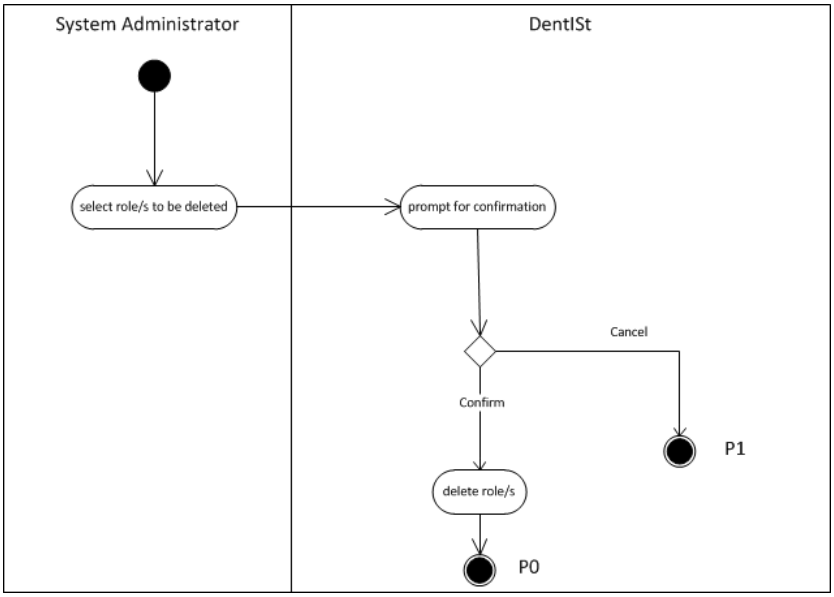


Figure 18: Delete Roles Activity Diagram of DentISt

5. Manage Sections

The Manage Sections Use Case Diagram involves the system administrator and faculty. In DentISt, system administrator can add, edit and delete sections. Both, system administrator and faculty role, have the privilege of search and view in the section. The Manage Sections Use Case Diagram of System Administrator and Faculty Clinician is shown in Figure 19. Use cases in gray are specified in the workflow module of Dentist 3.0.

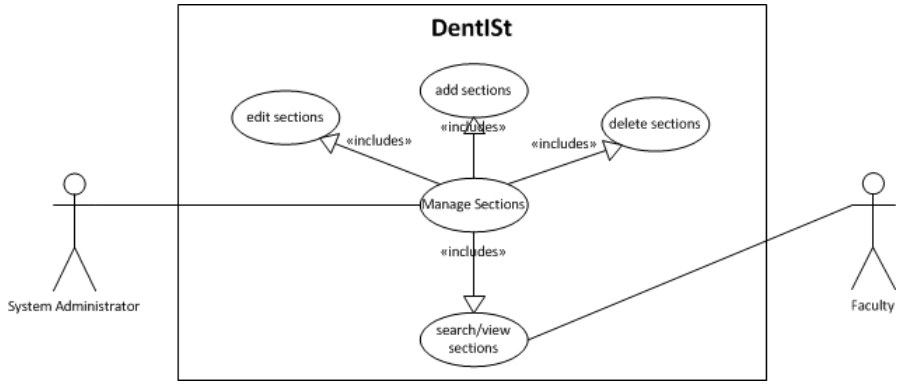


Figure 19: Manage Sections Use Case Diagram of the System Administrator and Faculty

Figures 20, 21, 22 and 23 show the Activity Diagrams for Manage Sections.

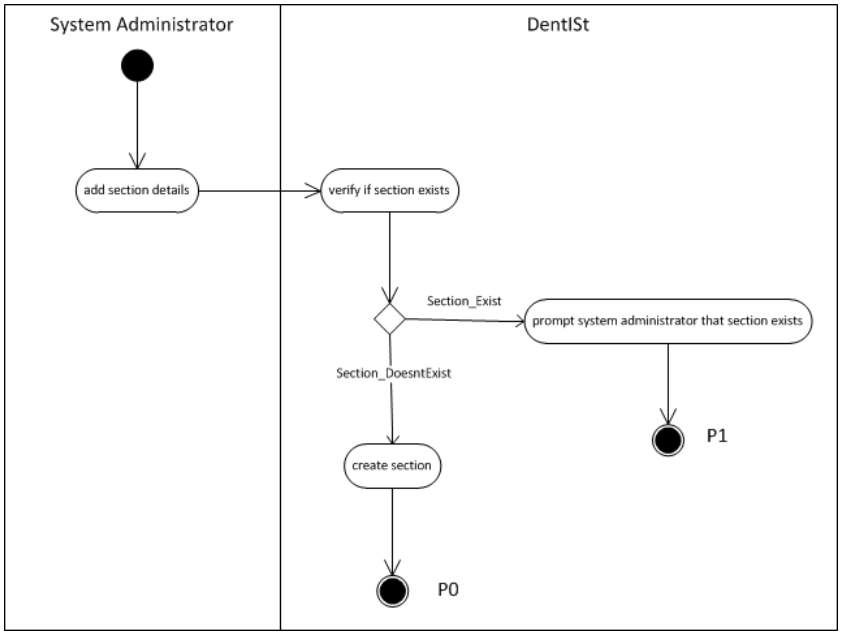


Figure 20: Add Sections Activity Diagram of DentISt

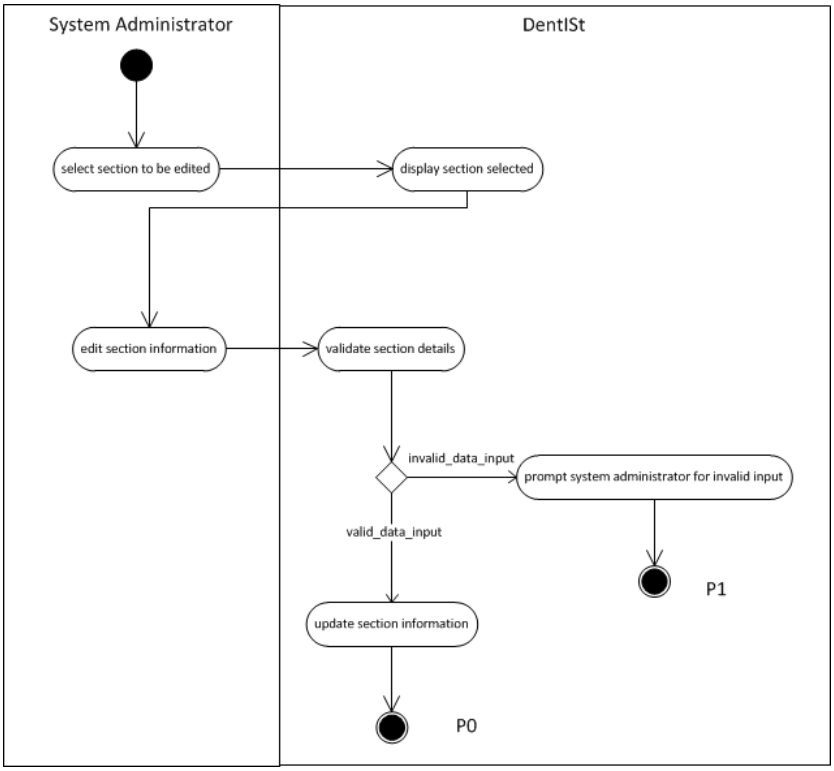


Figure 21: Edit Sections Activity Diagram of DentISt

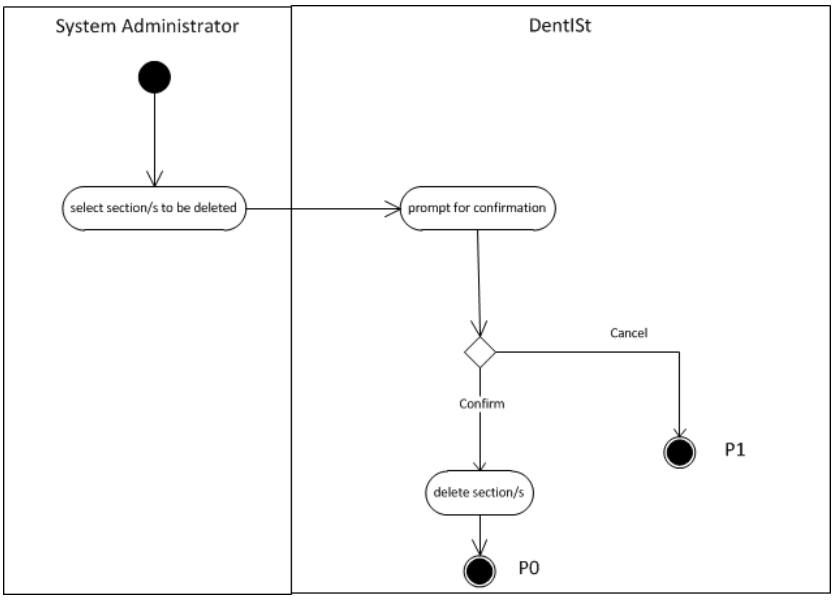


Figure 22: Delete Sections Activity Diagram of DentISt

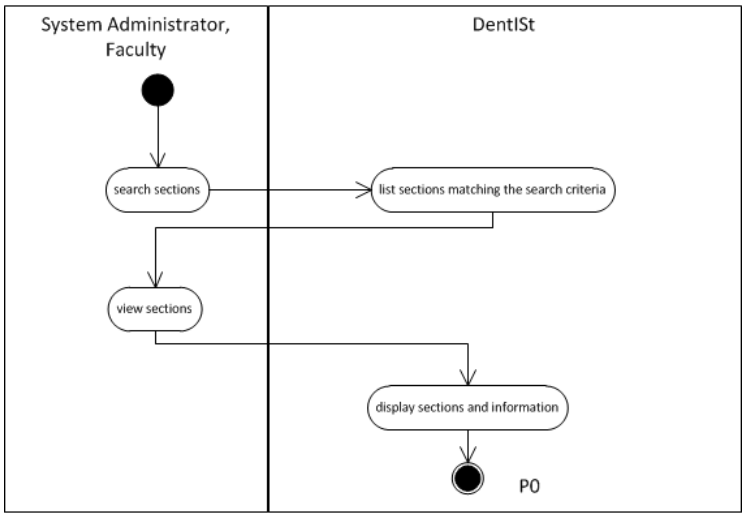


Figure 23: Search and View Sections Activity Diagram of DentISt

C. Entity Relationship Diagram (ERD)

Figure 24 shows the ERD of the User cluster that illustrates the relationships of the tables user, section, privilege, role, section_user, role_privilege and user_role.

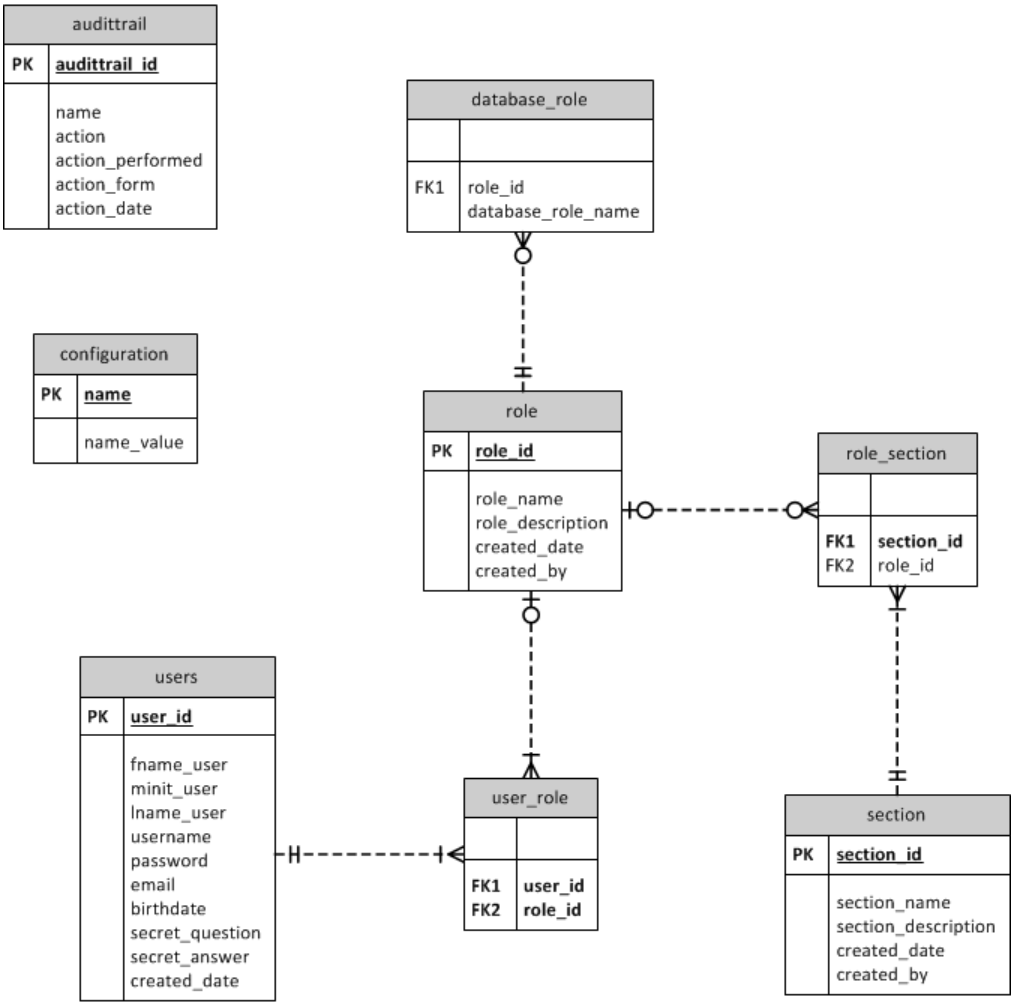


Figure 24: ERD for User Cluster

The ERD of the patient cluster in terms of patient information is shown in Figure 25. The tables connected to patient table are the following - patient additional info, patient vital signs, patient physical assessment, patient social history, patient appointment, treatment plan, patient medical history and patient dental history.

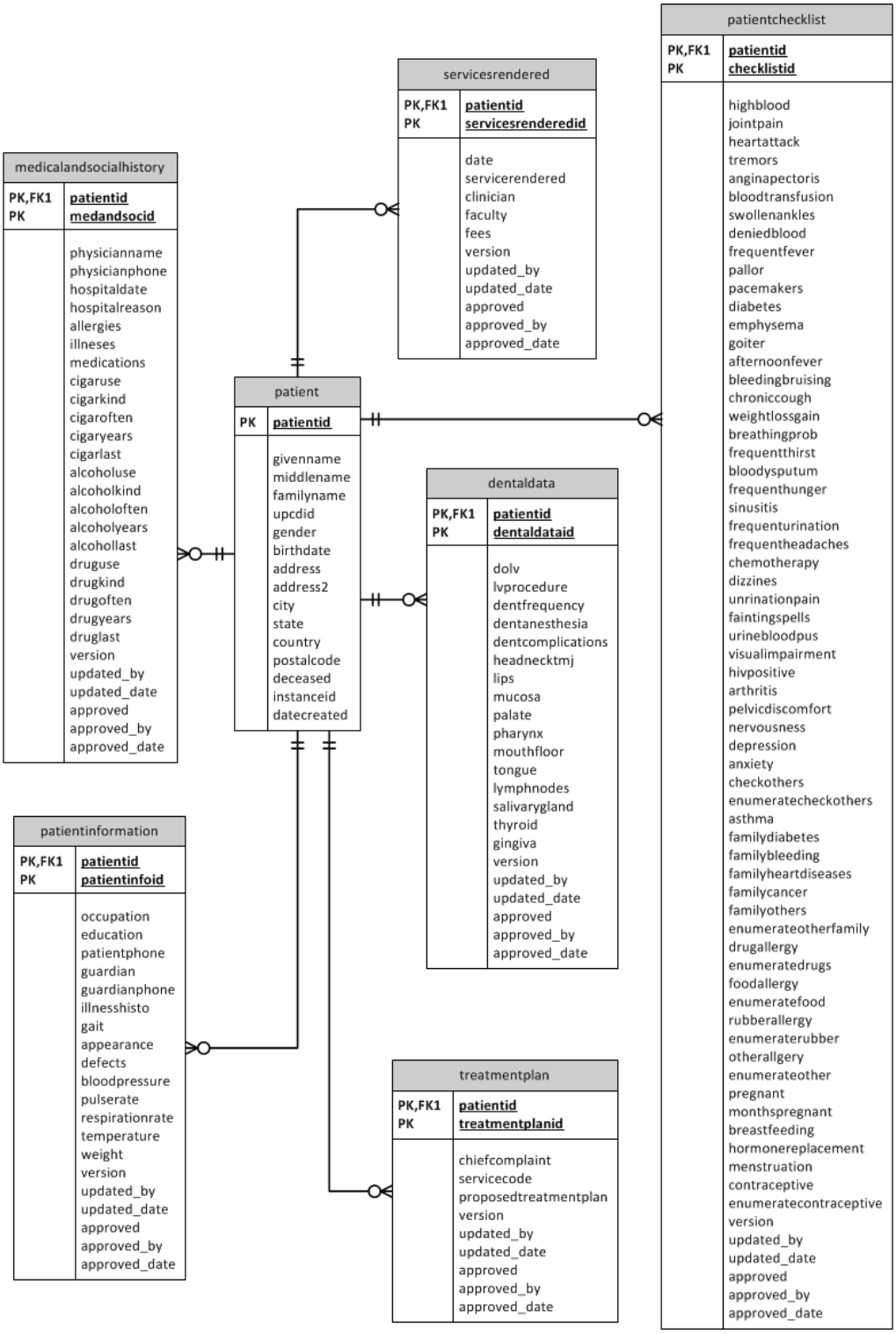


Figure 25: ERD for Patient Cluster in terms of patient information

Figure 26 illustrates the DentISt ERD of the same patient cluster above in terms of the dental chart and the dental status surfaces table - caries status, recurrent status, amalgam status, composite status, glassionomer status and tempfilling status.

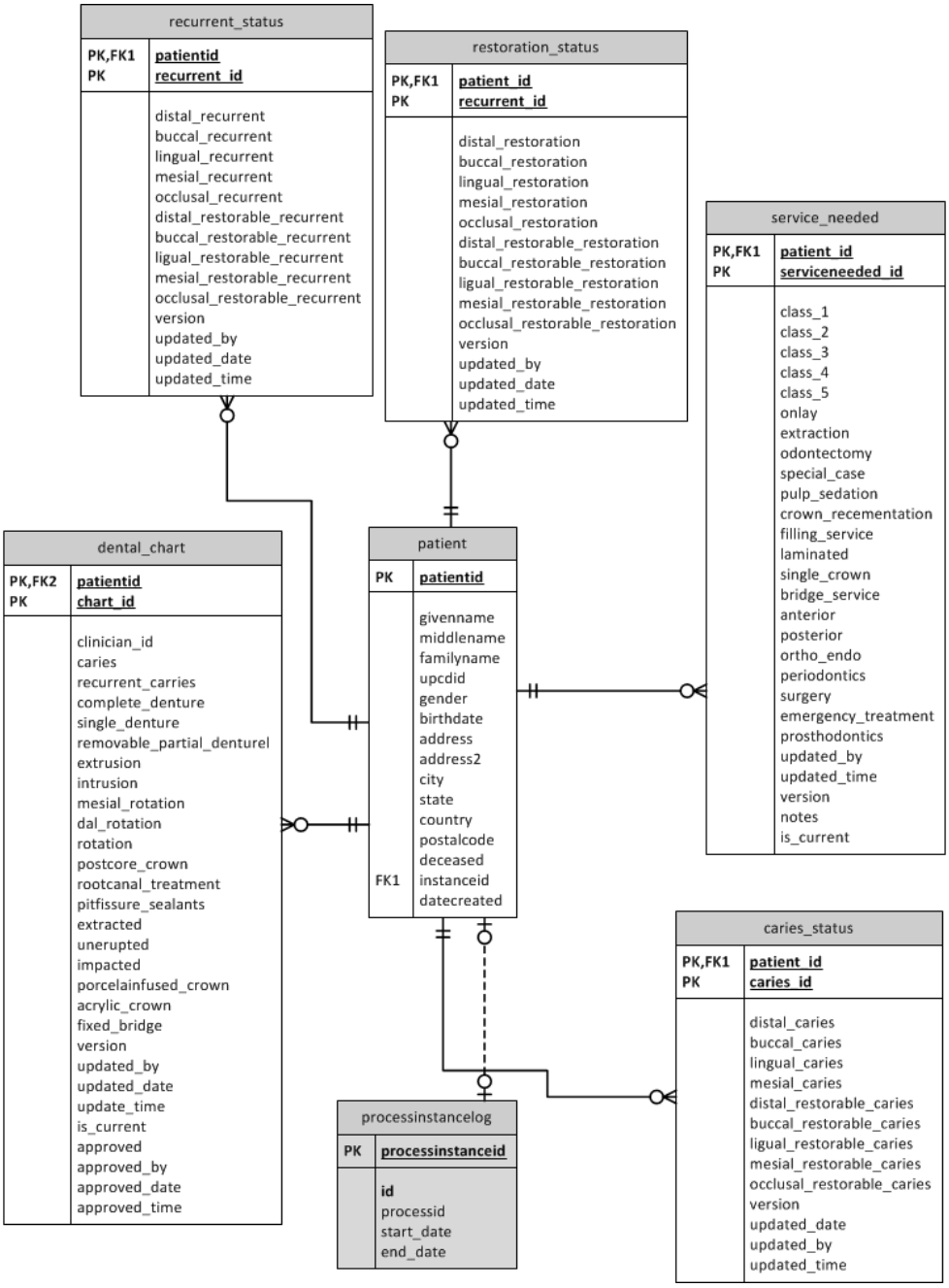


Figure 26: ERD for Patient Cluster in terms of dental chart and dental status

D. Data Dictionary

In encrypting the password of the users, the crypt function which is in the pgcrypto module of PostgreSQL is used. The crypt function is a one way (non-reversible) encryption function that accepts two parameters: password text and the salt text. Salt text is a random value that enables users with the same password to have different encrypted passwords. The salt is generated with the gen_salt function with the parameter bf as the hashing algorithm parameter [35].

Attribute	Data Type	Description
user_id	bigserial	user identifier
fname_user	character varying(128)	first name of user
minit_user	character varying(10)	middle initial of user
lname_user	character varying(128)	last name of user
username	character varying(128)	username of user
password	character varying(128)	password of user
email	character varying(255)	email of user
secret_question	character varying(255)	question for forgotten password
secret_answer	character varying(255)	answer to question for forgotten password
created_date	character varying (128)	current date the user is created

Table 1: users Table

Attribute	Data Type	Description
section_id	bigserial	section identifier
section_name	character varying(50)	name of section
section_description	character varying(128)	description of section
created_by	character varying(128)	user who created the section
created_date	character varying (128)	current date the section is created

Table 2: section Table

Attribute	Data Type	Description
role_id	bigserial	role identifier
role_name	character varying(50)	name of role
role_description	character varying(128)	description of role
created_by	character varying(128)	user who created the role
created_date	character varying (128)	current date the role is created

Table 3: role Table

Attribute	Data Type	Description
section_id	integer(11)	section identifier the role is assigned
role_id	integer(11)	role identifier assigned to a section

Table 4: role_section Table

Attribute	Data Type	Description
user_id	integer(11)	user identifier the role is assigned
role_id	integer(11)	role identifier assigned to a user

Table 5: user_role Table

Attribute	Data Type	Description
role_id	integer	role identifier assigned to a role
database_role_name	character varying (100)	equivalent database name of a role

Table 6: database_role Table

Attribute	Data Type	Description
audittrail_id	bigserial	audittrail identifier
name	bytea	name of user who performed action in the system
action	bytea	action done in the system
action_performed	bytea	who or what the action was performed
action_form	bytea	what form the action was done
action_date	bytea	date the action was done

Table 7: auddittrail Table

Attribute	Data Type	Description
name	character varying (150)	name of the configuration setting
name_value	character varying (500)	value of the specified name

Table 8: configuration Table

In compliance to the HIPAA policy, sensitive patient data are encrypted with encrypt function with three parameters - data, key (which is stored in the configuration file) and bf as the type of algorithm, Since encrypt is a two-way encryption function, the encrypted data are decrypted with the decrypt function with the same parameters as those used in encryption [35].

The following data are stored in encrypted format (with the encrypt function) in the database:

1. patient
   (a) givenname
   (b) middlename
   (c) familyname
   (d) birthdate
   (e) address
   (f) address2
   (g) city
   (h) state
   (i) country
   (j) postalcode
2. patientinformation
   (a) patientphone
   (b) guradian
   (c) guardianphone

Attribute	Data Type	Description
patientid	bigint	patient identifier
givenname	bytea	patient given name
middlename	bytea	patient middle name
familyname	bytea	patient family name
upcdid	character varying(50)	patient UPCD identifier
gender	character varying(10)	patient gender
birthdate	bytea	patient birthdate
address	bytea	patient address
address2	bytea	patient address
city	bytea	patient address
state	bytea	patient address
country	bytea	patient address
postalcode	bytea	patient address
deceased	character varying(10)	is patient deceased
instanceid	bigint	patient current case id
datecreated	character varying(50)	date patient record is created

Table 9: Patient Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
patientinfoid	bigint	patient information identifier
occupation	character varying(50)	occupation of patient
education	character varying(50)	educational attainment of patient
patientphone	bytea	contact number of patient
guardian	bytea	guardian of patient
guardianphone	bytea	contact number of guardian of patient
illnesshisto	character varying(200)	present illness of patient
gait	character varying(200)	gait
appearance	character varying(200)	appearance
defects character	varying(200)	defects
bloodpressure	character varying(20)	blood pressure of patient
pulserate	character varying(20)	pulse rate of patient
respirationrate	character varying(5)	respiration rate of patient
temperature	character varying(5)	temperature of patient
weight	character varying(7)	weight of patient
version	integer	version of patient information record of patient
updated_by	character varying(100)	user who updated the record
updated_date	character varying(50)	date the record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(50)	user who approved the record
approved_date	character varying(50)	date the record is approved

Table 10: PatientInformation Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
medandsocid	bigint	medical and social history identifier
physicianname	character varying(100)	physician of patient
physicianphone	character varying(100)	contact number of physician of patient
hospitaldate	character varying(100)	date of latest hospitalization of patient
hospitalreason	character varying(100)	reason for latest hospitalization
allergies	character varying(100)	allergies
illneses	character varying	illnesses
medications	character varying(100)	medications
childhood	character varying(100)	childhood diseases history
cigaruse	character varying(10)	Is the patient using or have used tobacco,cigarette?
cigarkind	character varying(100)	What kind does the patient smoke?
cigaroften	character varying(100)	How often does the patient smoke?
cigaryears	character varying(20)	How many years has the patient been smoking?
cigarlast	character varying(50)	If patient already stopped, how long since last used?
alcoholuse	character varying(10)	Does the patient drink alcoholic beverage?
alcoholkind	character varying(50)	What kind does the patient drink?
alcoholoften	character varying(20)	How often does the patient drink?
alcoholyears	character varying(5)	How many years has the patient been drinking?
alcohollast	character varying(50)	If patient already stopped, how long since last used?

Table 11: MedicalandSocialHistory Table

Attribute	Data Type	Description
druguse	character varying(10)	Has the patient used drugs for recreation purposes?
drugkind	character varying(50)	What kind of drug?
drugoften	character varying(20)	How often does the patient use drugs?
drugyears	character varying(5)	How many years has the patient been using?
druglast	character varying(50)	If patient already stopped, how long since last used?
version	integer	version of medical and social history record of patient
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date record is updated
approved	character varying(50)	approved status of patient
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date record is approved

Table 12: MedicalandSocialHistory Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
dentaldataid	bigint	dental data identifier
dolv	character varying(50)	date of last visit
lvprocedure	character varying(50)	last visit procedure
dentfrequency	character varying(25)	frequency of dental visit
dentanesthesia	character varying(25)	exposure and response to local anesthesia
dentcomplications	character varying(25)	complications during and or after dental procedure
headnecktmj	character varying(100)	head and neck TMJ
lips	character varying(100)	lips
mucosa	character varying(100)	mucosa
palate	character varying(100)	palate
pharynx	character varying(100)	pharynx
mouthfloor	character varying(100)	mouth floor
tongue	character varying(100)	tongue
lymphnodes	character varying(100)	lymphnodes
salivarygland	character varying(100)	salivary gland
thyroid	character varying(100)	thyroid
gingiva	character varying(100)	gingiva
version	integer	version of dental data record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date the record is approved

Table 13: DentalData Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
treatmentplanid	bigint	treatment plan identifier
chiefcomplaint	character varying(35)	chief complaint of patient
servicecode	character varying(100)	service codes of treatment type
proposedtreatment	character varying(200)	proposed treatment
version	integer	version of treatment plan record
updated_by	character varying(100)	user who updated the record
updated_date	character varying(50)	date record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date the record is approved

Table 14: TreatmentPlan Table

Attribute	Data Type	Description
patientid	bigint	patient identifier
servicesrenderedid	bigint	services rendered identifier
date	character varying(50)	date when service is rendered by patient
servicerendered	character varying(100)	type of service rendered by patient
clinician	character varying(100)	clinician who performed the service
faculty	character varying(100)	faculty clinician who checked the work of the clinician
fees	character varying(20)	fees paid by patient
version	integer	version of service rendered
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
approved	character varying(50)	approved status of record
approved_by	character varying(100)	user who approved the record
approved_date	character varying(50)	date when record is approved

Table 15: ServicesRendered Table

Attribute	Data Type	Description
caries_id	bigint	caries identifier
patient_id	bigint	patient identifier
distal_caries	integer[]	distal surface with caries
buccal_caries	integer[]	buccal surface with caries
lingual_caries	integer[]	lingual surface with caries
mesial_caries	integer[]	mesial surface with caries
occlusal_caries	integer[]	occlusal surface with caries
distal_restorable_caries	character varying[]	variation of distal surface with caries
buccal_restorable_caries	character varying[]	variation of buccal surface with caries
lingual_restorable_caries	character varying[]	variation of lingual surface with caries
mesial_restorable_caries	character varying[]	variation of mesial surface with caries
occlusal_restorable_caries	character varying[]	variation of occlusal surface with caries
version	bigint	version of caries status record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated

Table 16: caries_status Table

Attribute	Data Type	Description
recurrent_id	bigint	recurrent caries identifier
patient_id	bigint	patient identifier
distal_recurrent	integer[]	distal surface with recurrent caries
buccal_recurrent	integer[]	buccal surface with recurrent caries
lingual_recurrent	integer[]	lingual surface with recurrent caries
mesial_recurrent	integer[]	mesial surface with recurrent caries
occlusal_recurrent	integer[]	occlusal surface with recurrent caries
distal_restorable_recurrent	character varying[]	variation of distal surface with recurrent caries
buccal_restorable_recurrent	character varying[]	variation of buccal surface with recurrent caries
lingual_restorable_recurrent	character varying[]	variation of lingual surface with recurrent caries
mesial_restorable_recurrent	character varying[]	variation of mesial surface with recurrent caries
occlusal_restorable_recurrent	character varying[]	variation of occlusal surface with recurrent caries
version	bigint	version of recurrent caries status record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated

Table 17: recurrent_status Table

Attribute	Data Type	Description
restoration_id	bigint	restoration identifier
patient_id	bigint	patient identifier
distal_restoration	integer[]	distal surface with restoration
buccal_restoration	integer[]	buccal surface with restoration
lingual_restoration	integer[]	lingual surface with restoration
mesial_restoration	integer[]	mesial surface with restoration
occlusal_restoration	integer[]	occlusal surface with restoration
distal_restorable_restoration	character varying[]	variation of distal surface with restoration
buccal_restorable_restoration	character varying[]	variation of buccal surface with restoration
lingual_restorable_restoration	character varying[]	variation of lingual surface with restoration
mesial_restorable_restoration	character varying[]	variation of mesial surface with restoration
occlusal_restorable_restoration	character varying[]	variation of occlusal surface with restoration
version	bigint	version of restoration status record
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated

Table 18: restoration_status Table

Attribute	Data Type	Description
serviceneeded_id	bigint	service needed identifier
patient_id	bigint	patient identifier
class_1	integer[]	tooth numbers that need class 1 treatment
class_2	integer[]	tooth numbers that need class 2 treatment
class_3	integer[]	tooth numbers that need class 3 treatment
class_4	integer[]	tooth numbers that need class 4 treatment
class_5	integer[]	tooth numbers that need class 1 treatment
onlay	integer[]	tooth numbers that need onlay treatment
extraction	integer[]	tooth numbers that need extraction
odontectomy	integer[]	tooth numbers that odontectomy
special_case	integer[]	tooth numbers that are special cases
pulp_sedation	integer[]	pulp sedation treatment
crown_recementation	integer[]	recementation of crowns
filling_service	integer[]	temporary fillings
laminated	integer[]	tooth numbers that need laminated fixed partial denture
single_crown	integer[]	tooth numbers that need single crown fixed partial denture
bridge_service	integer[]	tooth numbers that need bridge fixed partial denture
anterior	integer[]	anterior endodontics
posterior	integer[]	anterior endodontics
ortho_endo	integer[]	other endodontics

Table 19: service_needed Table

Attribute	Data Type	Description
periodontics	character varying(10)	management of periodontal disease
surgery	character varying(50)	surgery
emergency_treatment	character varying(50)	emergency treatment
prosthodontics	character varying(50)	prosthodontics
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated
version	integer	version of service needed record
notes	character varying(500)	additional notes
is_current	character varying(10)	is the record the latest version?

Table 20: service_needed Table

Attribute	Data Type	Description
dental_chart_id	bigint	dental chart identifier
patient_id	bigint	patient identifier
clinician_id	bigint	clinician of patient
caries	integer[]	tooth numbers with caries
recurrent_caries	integer[]	tooth numbers with recurrent caries
restoration	integer[]	tooth numbers with restoration
removable_partial_denture	integer[]	removable partial denture
extrusion	integer[]	tooth numbers with extrusion
intrusion	integer[]	tooth numbers with intrusion
mesial_rotation	integer[]	tooth numbers with mesial rotation
distal_rotation	integer[]	tooth numbers with distal rotation
rotation	integer[]	tooth numbers with rotation
postcore_crown	integer[]	tooth numbers with post core crown
rootcanal_treatment	integer[]	tooth numbers with root canal treatment
pitfissure_sealants	integer[]	tooth numbers with pit and fissure sealant
extracted	integer[]	extracted teeth
missing	integer[]	missing teeth
unerupted	integer[]	unerupted teeth
impacted	integer[]	impacted teeth
porcelain_crown	integer[]	tooth numbers with porcelain crown
acrylic_crown	integer[]	tooth numbers with acrylic crown

Table 21: dentalchart Table

Attribute	Data Type	Description
metal_crown	integer[]	tooth numbers with metal crown
porcelain_infused	integer[]	tooth numbers with porcelain fused to metal crown
fixed_bridge	integer[]	fixed bridge
version	integer	dental chart record version
updated_by	character varying(50)	user who updated the record
updated_date	character varying(50)	date the record is updated
updated_time	character varying(50)	time the record is updated
is_current	character varying(50)	is the record the latest dental chart version?
approved	character varying(50)	approved status of the record
approved_by	character varying(50)	user who approved the record
approved_date	character varying(50)	date the record is approved
approved_time	character varying(50)	time the record is approved
complete_denture	character varying(10)	complete denture
single_denture	character varying(50)	single denture

Table 22: dentalchart Table

E. XACML and PostgreSQL Setting

Using the PostgreSQL setting, privileges of users/roles in the schema can be implemented in the database. Access control policies of DentISt 3.0 are presented in XACML (access control xml files) and XACML for clinicians are separated according to their section. As discussed in section E. in the Theoretical Framework, XACML is used as a way to standardized representation of access control policies.

Target element is in three parts- subject, resource and action. In relation to DentISt, subjects are the set of user/role in the database, resources are the set of tables in the database and actions are the set of operations in the database. Figure 27 shows the standard XACML to be used in the DentIst.

```
<Policy PolicyId="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">-user/role-</AttributeValue>
          <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string" />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">-tablename-</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="http://www.w3.org/2001/XMLSchema#string" />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">-databaseoperation-</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string" />
        </ActionMatch>
      </Action>

    </Actions>
  </Target>
 <Rule RuleId="IfPermit-tablename-" Effect="Permit">
    <Condition>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
    </Condition>
 </Rule>
</Policy>
```

Figure 27: Standard XACML framework for access control in DentISt

These three parts are used to transform the XACML to its equivalent SQL statements that provides the GRANTS of a user (subject) to a table (resource) with the following operations (action). The rule combining algorithm also provides the indication of whether the XACML is for permission (GRANT) or denial (REVOKE) of privileges. The transformed standardized statement of an XACML is in Figure 28.

```
GRANT -action- ON TABLE -resource- TO -subject-;
```

Figure 28: Standard XACML framework for access control in DentISt

Figure 29 shows a XACML file in DentISt. while Figure 30 shows the equivalent SQL file generated from the exampled XACML.

```
<Policy PolicyId="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentIStProsthoStudent</AttributeValue>
            <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
    <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentIStProsthoFac</AttributeValue>
            <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
</Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">user_role</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http:
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">select</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
</Policy>
```

Figure 29: User Role XACML file for Prosthodontics section

```
GRANT select ON TABLE "user_role" TO "DentIStProsthoStudent";
GRANT select ON TABLE "user_role" TO "DentIStProsthoFac";
```

Figure 30: Equivalent SQL files of User Role XACML file

V. Architecture

A. System Architecture

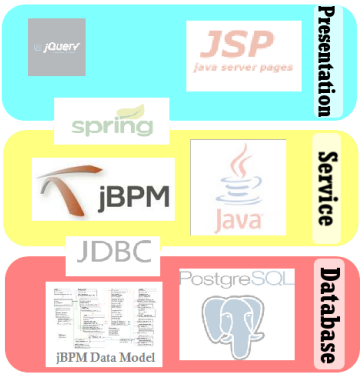


Figure 31: System Architecture of DentISt

Figure 31 shows the system architecture of DentISt. The system has three layers- presentation, service and database. The presentation layer has the web interface that is coded in JSP and uses JQuery as its Javascript framework. The presentation and service layers are connected using the Spring framework. The service layer uses jBPM 5.2 to manage the workflow and other services. It was developed with the Java programming language and uses JDBC to connect to PostgreSQL at the database layer.

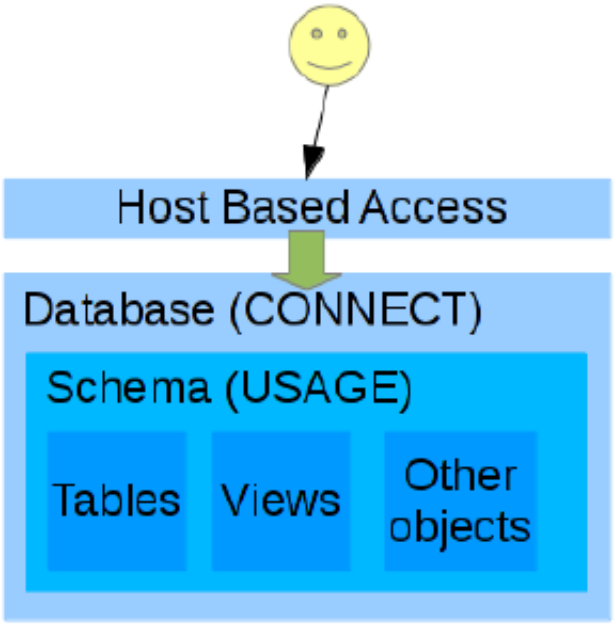


Figure 32: PostgreSQL management in DentISt [2]

Figure 32 shows the structural design of what happens inside the PostgreSQL database. A role runs into the host base access (HBA) before being able to allow a database connection. The HBA is the configuration in the database that defines the authentication of a user to enter the database. For the host base access to allow a connection, a role must have the right to CONNECT to the database and must have the right of USAGE to the schema. A role can only access or modify the contents of the objects (tables, etc) if it has the privilege.

B. Technical Architecture

DentISt 3.0 is compatible for the operating system:

- Ubuntu Linux 12.04.2 or Redhat Linux

It will also use the following software:

- Apache Tomcat 6.25
- PostgreSQL 8 or 9 with pgcrypt module

The client side must have any of the following compatible web browsers:

- Mozilla Firefox 16.0.1
- Google Chrome 22.0.1229.94
- Safari 5.1.7
- Opera 12.02

VI. Results

A. User-System Interface

When you open the site, Figures 33 which is the login page is first displayed.

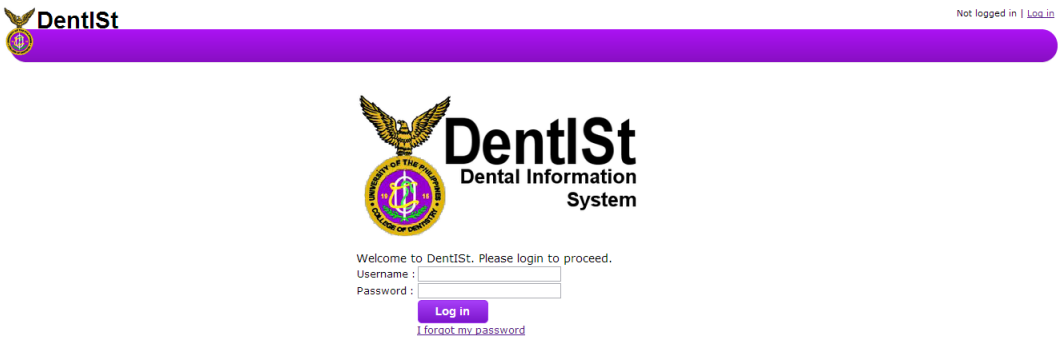


Figure 33: Login Page for DentISt

After logging in, the header or the navigation menu of DentISt is shown and its tabs are dependent to what role/s the login user have. Users use stored procedures for every database call needed. Figures 34, 35, 36 and 37 are the homepage for the roles system administrator, student clinician, faculty, and workflow administrator, respectively.

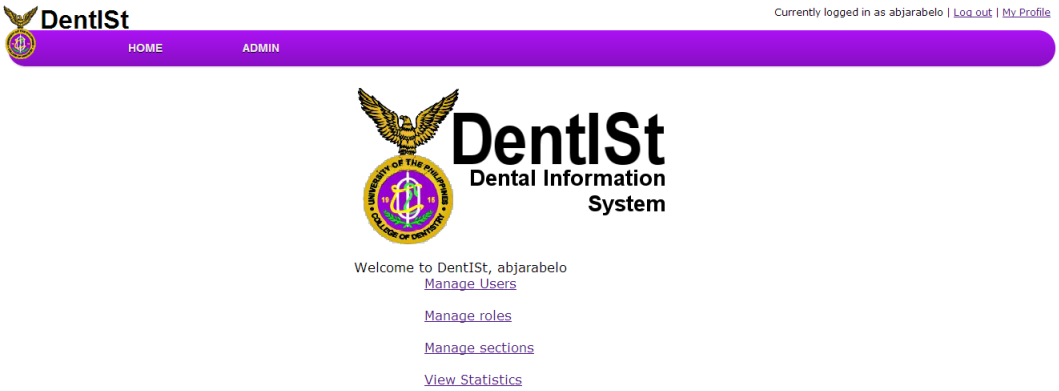


Figure 34: Homepage for DentISt-System Administrator

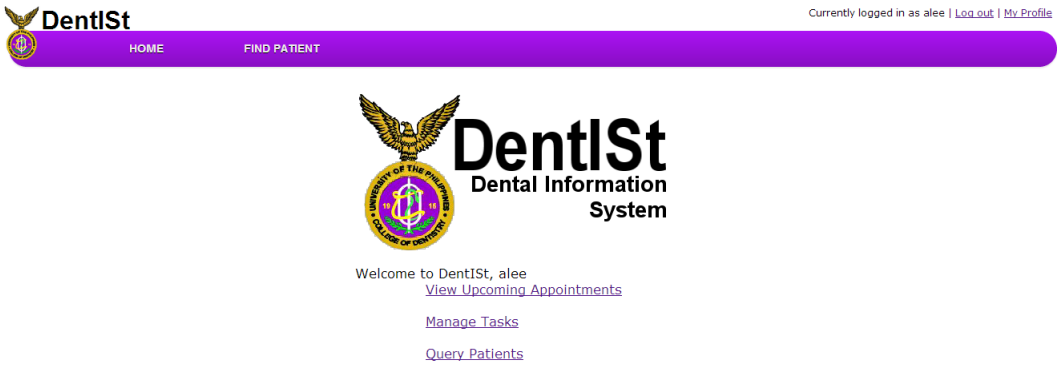


Figure 35: Homepage for DentISt-Student Clinician

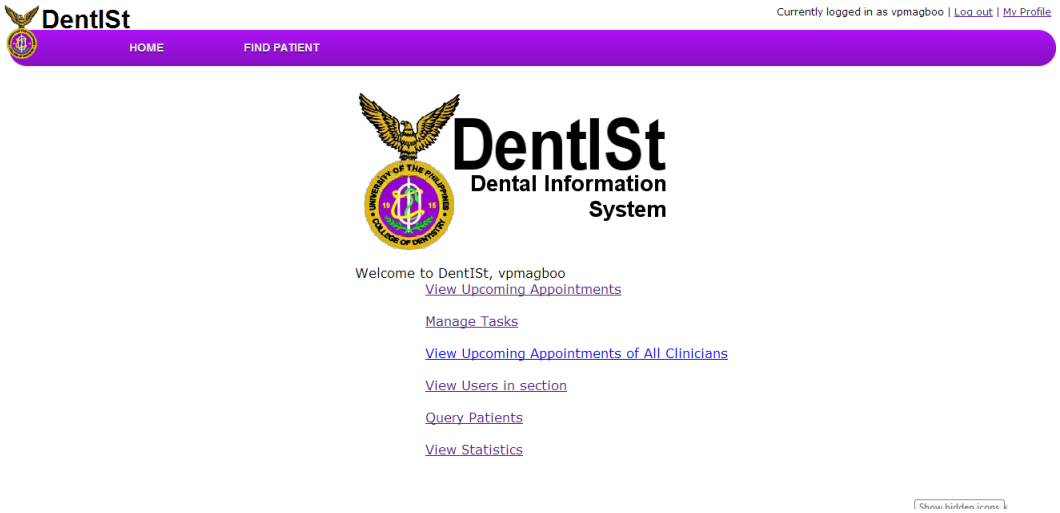


Figure 36: Homepage for DentISt-Faculty

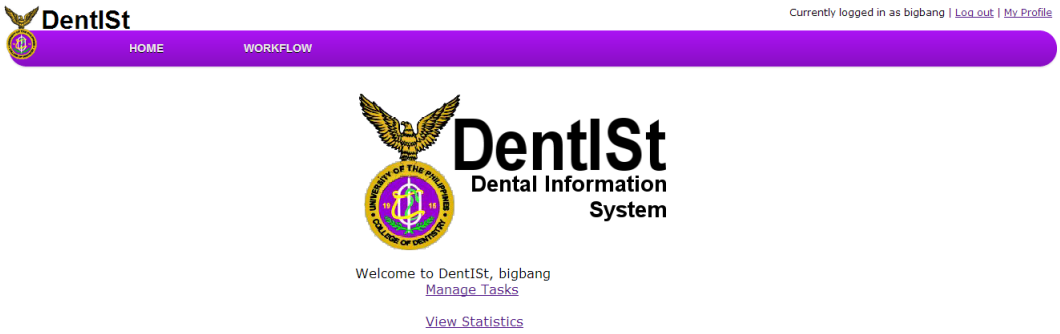


Figure 37: Homepage for DentISt-Workflow Administrator

System administrator is responsible for the management of users, roles and sections and other system administrator functions such as view statistics, email settings and audit trail.

In the management of users, adding a new user requires the input of the name, username, password, email address, secret question and secret answer. Figure 38 shows the add user form.

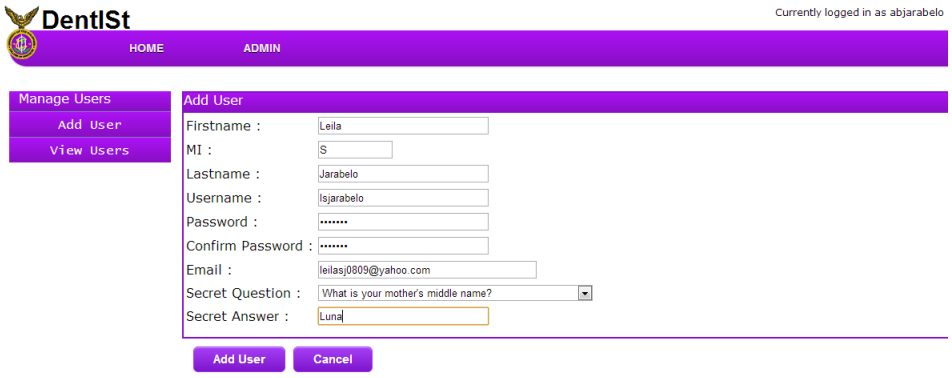


Figure 38: Add User form in DentISt

Once the user is added, we can directly go to the edit user form, where two tabs can be seen- for the edit user form(Figure 39) and the add user role form (Figure 40). Figure 41 also gives View Users form.

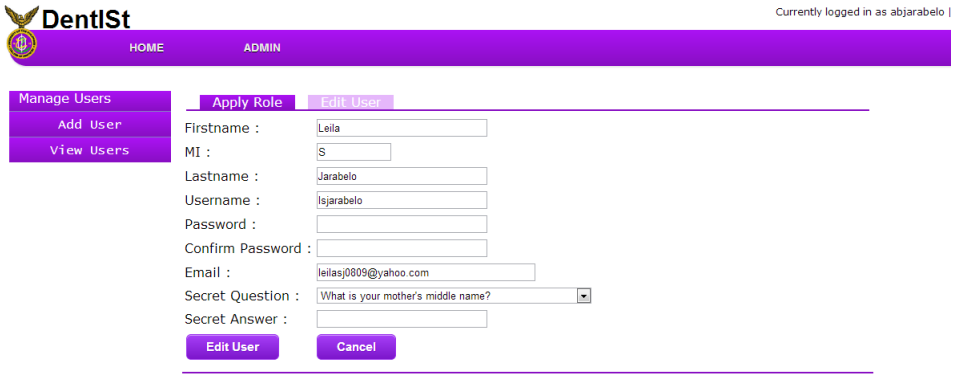


Figure 39: Edit User form in DentISt

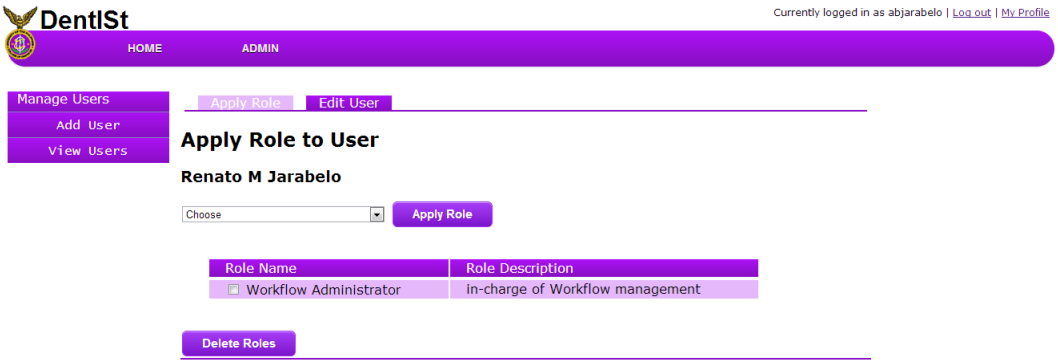


Figure 40: Apply role to user form in DentISt

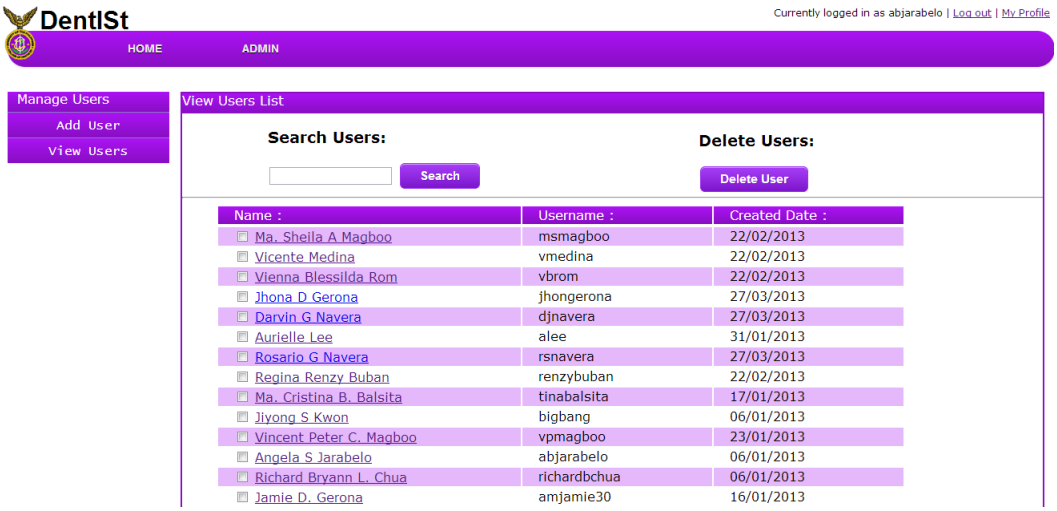


Figure 41: View User form in DentISt

As continuation with the add role, management of roles also includes view, delete and edit roles. The view roles form includes the delete function as well and is shown in Figure 42. There are two tabs for updating the role- editing the role(Figure 43) and viewing users in that role (Figure 44).

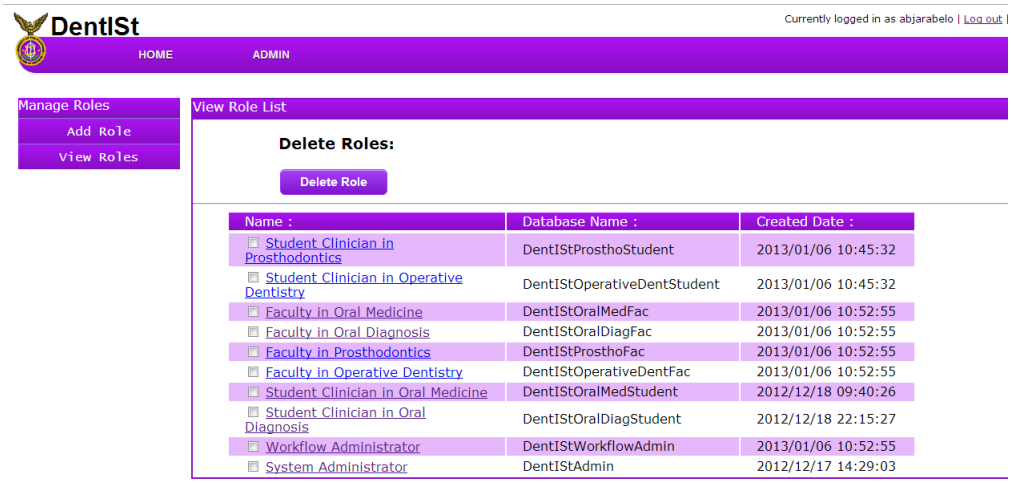


Figure 42: View Role form in DentISt

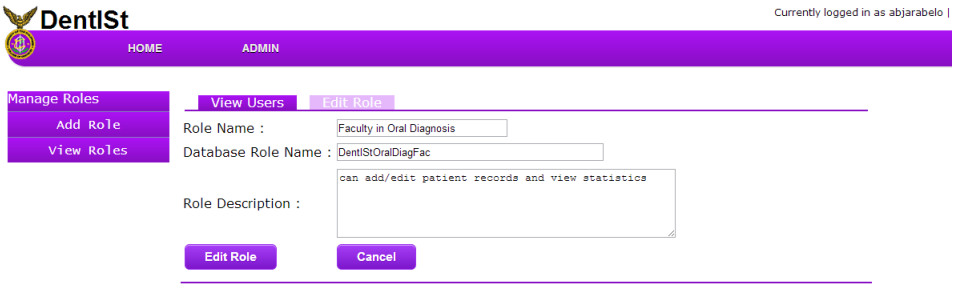


Figure 43: Edit Role form in DentISt

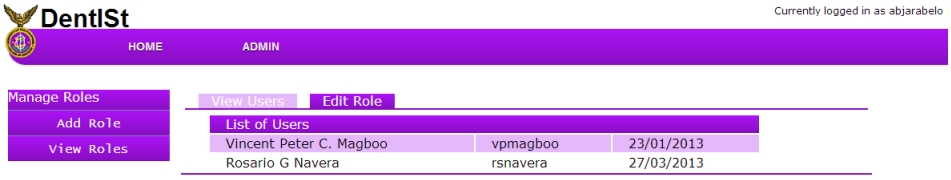


Figure 44: View users given a role in DentISt

For the management of sections, a system administrator can also add, edit, view, and delete sections. Figure 45 and 46 show the add section form, view sections with delete functions respectively. It also include some section tabs for editing the section(Figure 47) and adding roles to section(Figure 48).

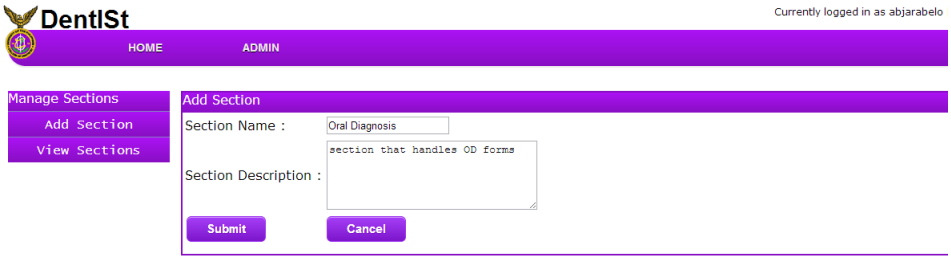


Figure 45: Add Section form in DentISt

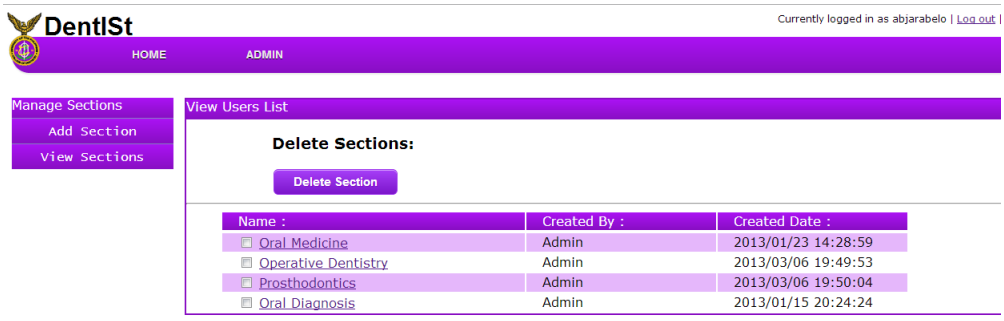


Figure 46: View Section form in DentISt

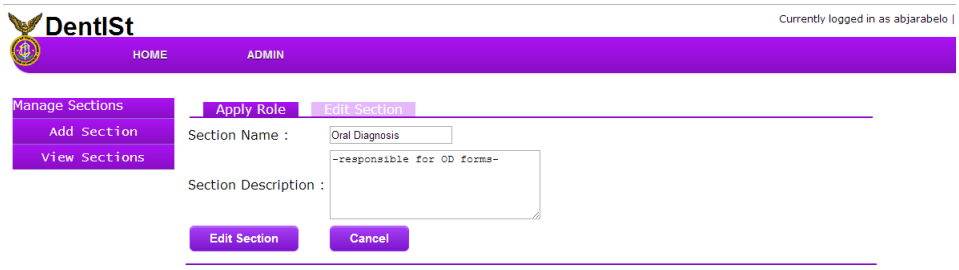


Figure 47: Edit Section form in DentISt

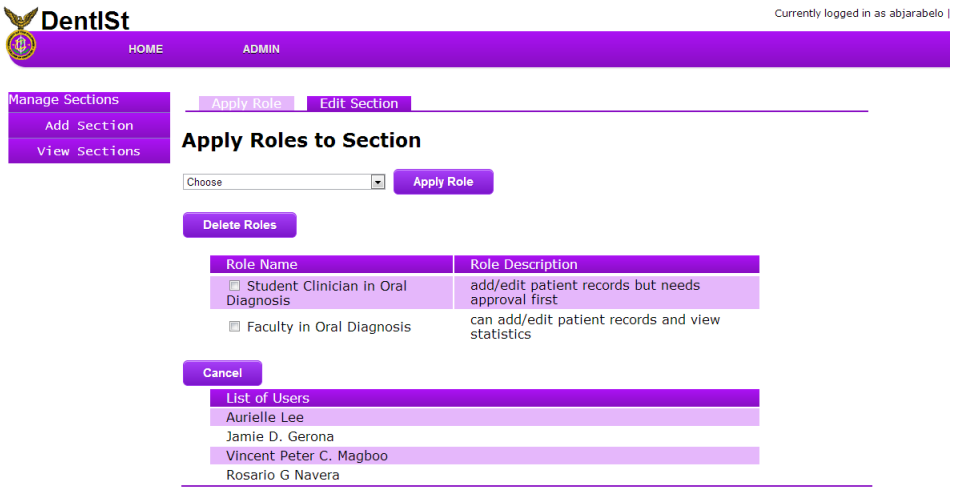


Figure 48: Add roles to a section form in DentISt

Other system administrator rights also include the change of email settings, view statistics and audit trail. Figure 49 shows the email settings form that is used as the email being acknowledge when a person has forgotten his/her password.

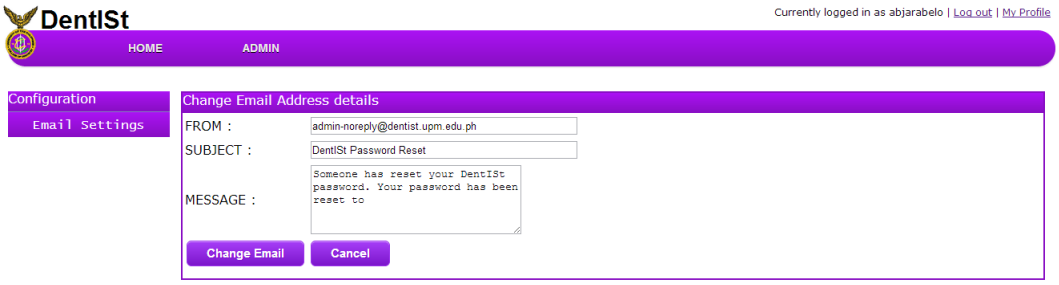


Figure 49: Email setting form in DentISt

Figure 50 is the view statistics of the whole UPCD which generates a report in a specific period of time. The system administrator can search according to its demographic, dental condition, service needed and the service done to a patient. Figure 51 shows the result page of the statistics.

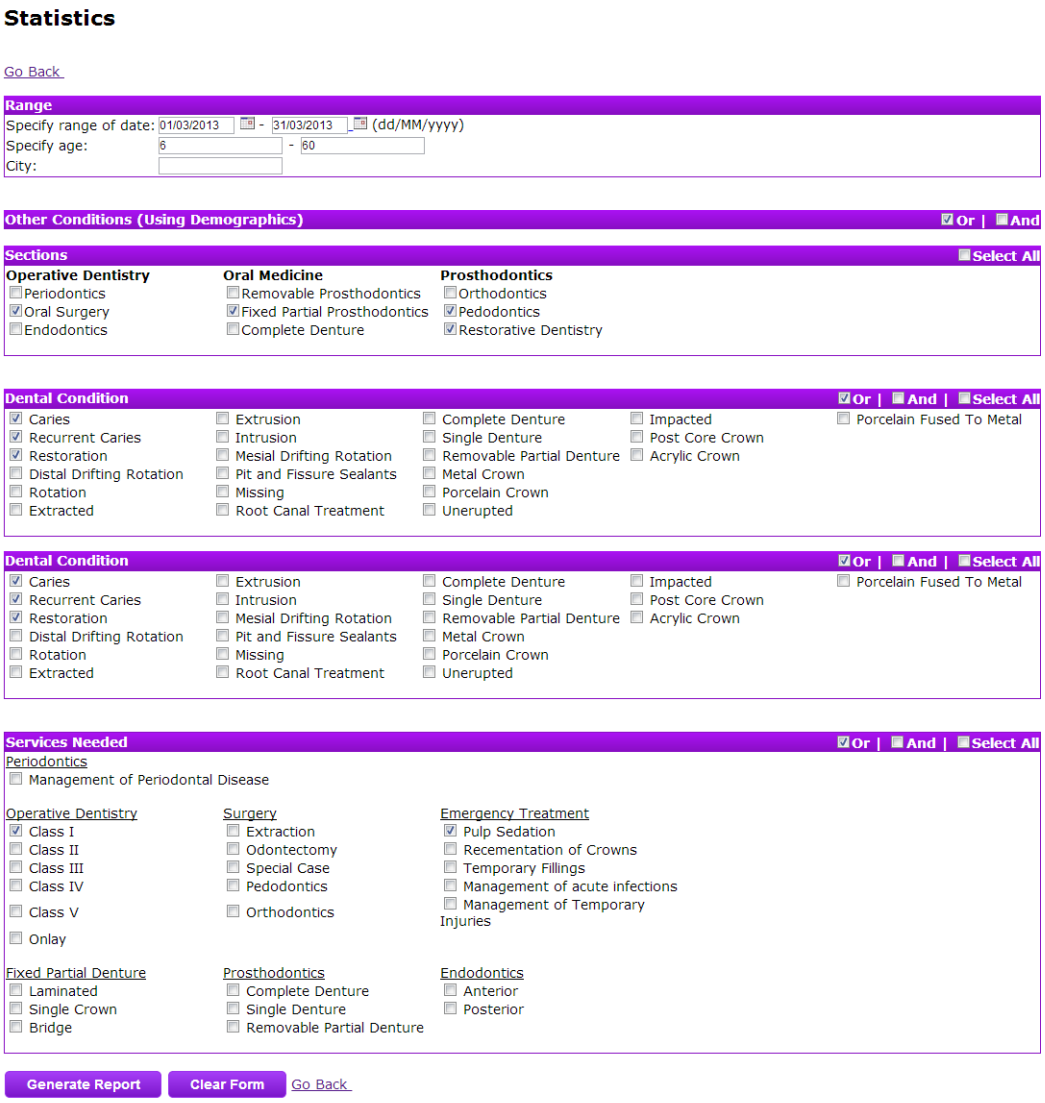


Figure 50: Statistics form in DentISt

Statistics

Patients registered from 01/03/2013 to 31/03/2013

Total # of Females	5	71%
Total # of Males	2	28%
AGE GROUP (6 - 60)		
# of Females	5	71%
# of Males	2	28%

Other Conditions from 01/03/2013 to 31/03/2013 (OR)

Sections

Condition	# of Cases	Females	% Females	Males	% Males	% Females (over total females)	% Males (over total males)	% Cases (over total patients)
Oral Surgery	0	0	0%	0	0%	0%	0%	0%
Fixed Partial Prosthodontics	0	0	0%	0	0%	0%	0%	0%
Restorative Dentistry	1		0%		100%	0%	50%	14%
Pedodontics	0	0	0%	0	0%	0%	0%	0%
Total # of Cases	**1**							

Sections

Condition	# of Cases	Females	% Females	Males	% Males	% Females (over total females)	% Males (over total males)	% Cases (over total patients)
Oral Surgery	0	0	0%	0	0%	0%	0%	0%
Fixed Partial Prosthodontics	0	0	0%	0	0%	0%	0%	0%
Restorative Dentistry	1		0%		100%	0%	50%	14%
Pedodontics	0	0	0%	0	0%	0%	0%	0%
Total # of Cases	**1**							

Dental Status Chart (OR)

Condition	# of Cases	Females	% Females	Males	% Males	% Females (over total females)	% Males (over total males)	% Cases (over total patients)
Caries	3	3	100%	0	0%	60%	0%	42%
Recurrent Caries	2	2	100%	0	0%	40%	0%	28%
restoration	2	2	100%	0	0%	40%	0%	28%
Total # of Cases	**7**							

Services Needed (OR)

Condition	# of Cases	Females	% Females	Males	% Males	% Females (over total females)	% Males (over total males)	% Cases (over total patients)
Class 1	2	2	100%	0	0%	40%	0%	28%
Pulp Sedation	0	0	0%	0	0%	0%	0%	0%
Total # of Cases	**2**							

Figure 51: View Statistics result in DentISt

Meanwhile, the audit trail is a special functionality of the system. Figure 52 shows the audit trail of the system and can search the specific actions and forms within a specific period. For the example, audit trail searches for the audit log of viewing the Medical and Social History form and gives who, what resource and action, and when it was done in a period of time.

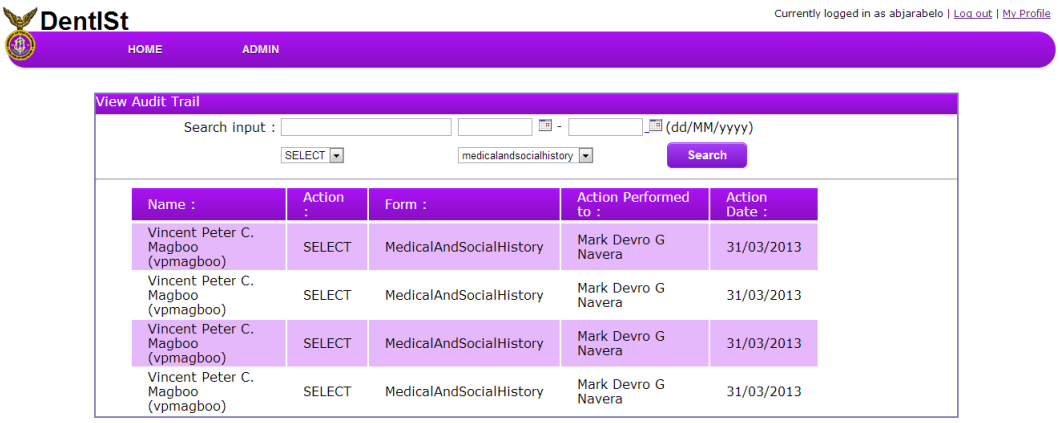


Figure 52: Audit Trail searching for insertion of patients in a specific period of time in DentISt

Aside from the management of patient record, a faculty can view clinicians, both faculty and student clinician, in his specific section, view statistics in a section and query patient. Figure 53 shows the page of where a faculty can see the clinicians in the section where he belongs. A faculty can also search for either faculty or student clinicians in a certain section.

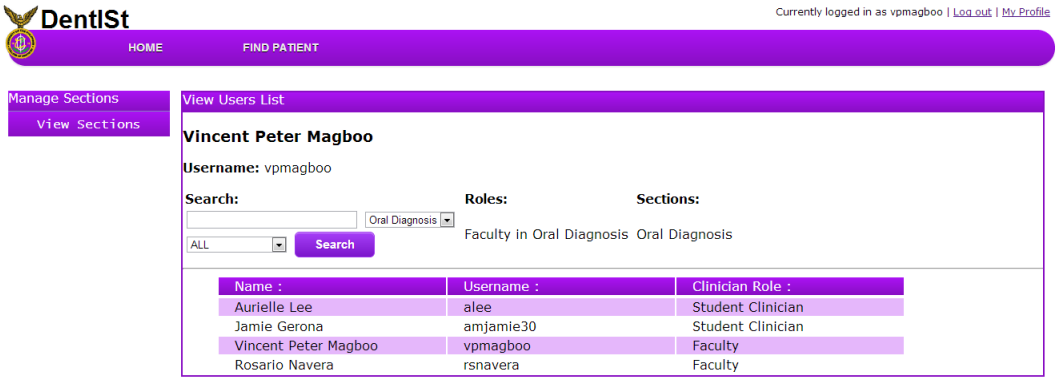


Figure 53: View clinicians in a section in DentISt

Figure 54 is for the view statistics in a specific section where a faculty can generate the statistics in a given period of time in his/her own section.

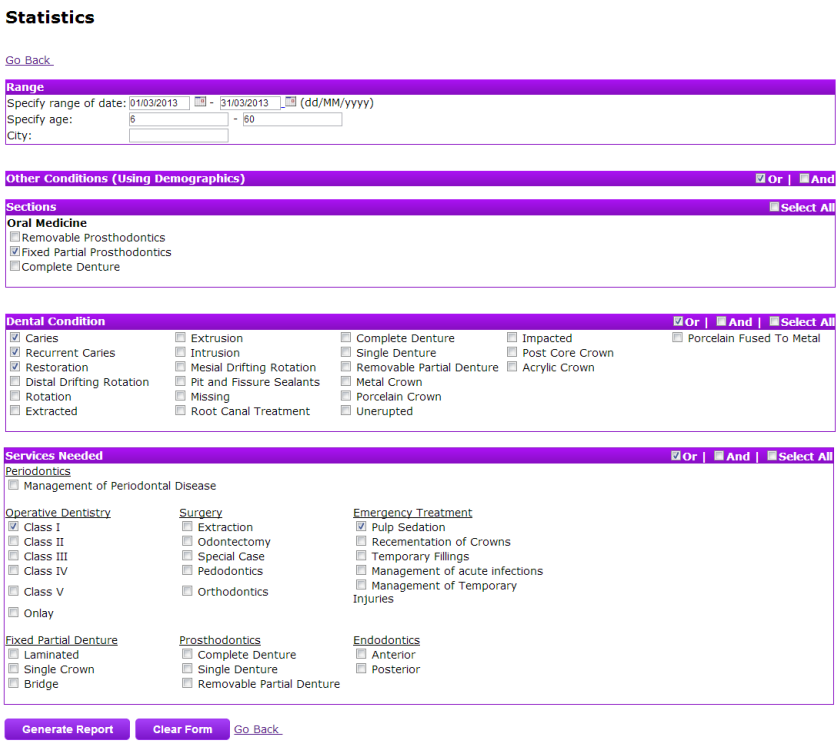


Figure 54: View Statistics in a specific section (Oral Medicine) in DentISt

Lastly Figure 55 shows the search patient form page that can be both used by the faculty and student clinician. A clinician can query patients according to their demographic, dental condition, service needed and service done. Figure 56 is the end result of the query patients where patient ids are link to its own record for read-only check.

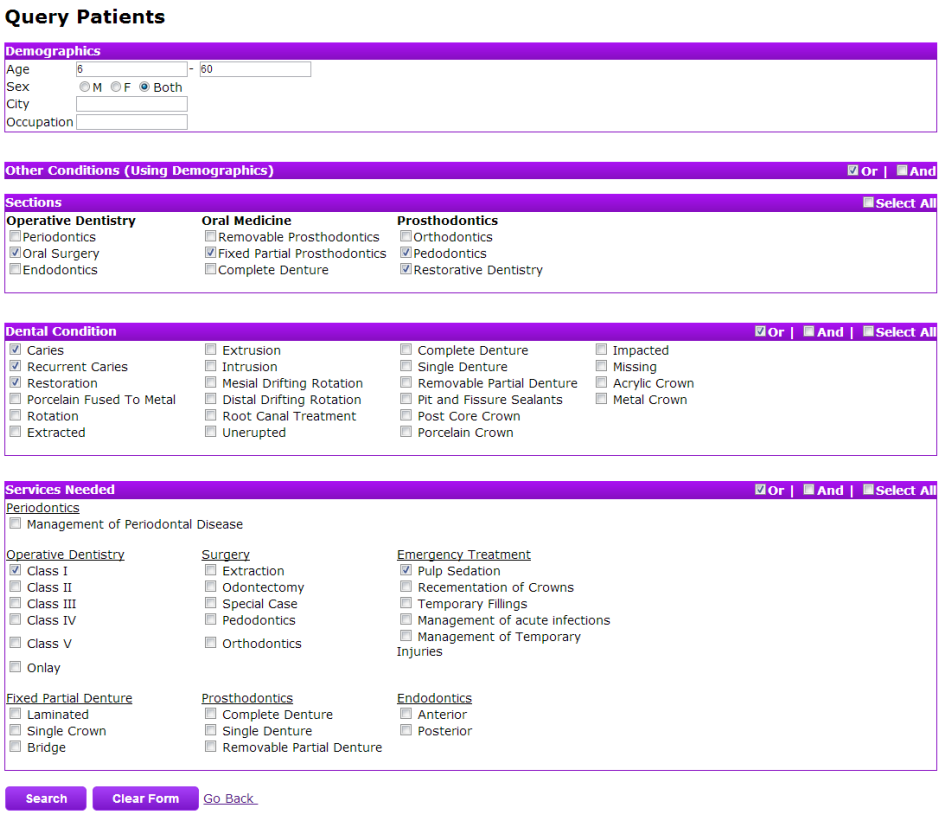
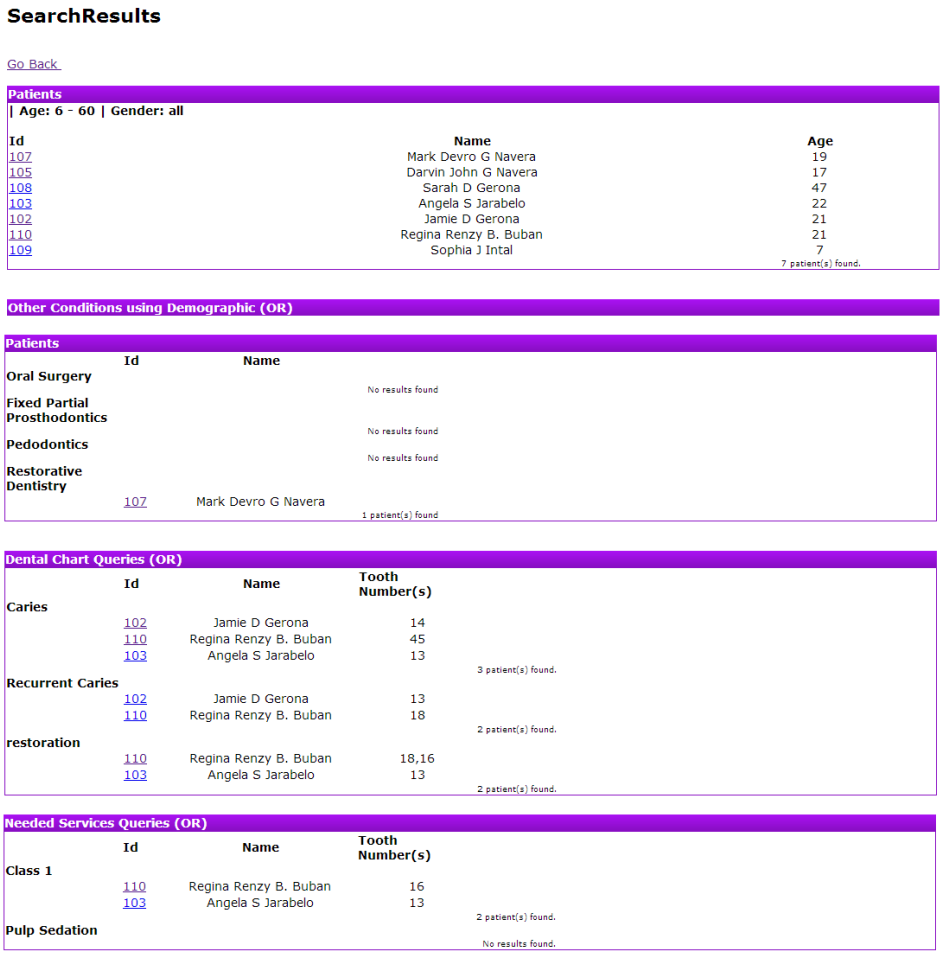


Figure 55: Query patient form in DentISt



SearchResults

Go Back

Patients

| Age: 6 - 60 | Gender: all

Id	Name	Age
107	Mark Devro G Navera	19
105	Darvin John G Navera	17
108	Sarah D Gerona	47
103	Angela S Jarabelo	22
102	Jamie D Gerona	21
110	Regina Renzy B. Buban	21
109	Sophia J Intal	7

7 patient(s) found.

Other Conditions using Demographic (OR)

Patients

	Id	Name
Oral Surgery		
	No results found	
Fixed Partial Prosthodontics		
	No results found	
Pedodontics		
	No results found	
Restorative Dentistry		
	107	Mark Devro G Navera

1 patient(s) found

Dental Chart Queries (OR)

	Id	Name	Tooth Number(s)
Caries			
	102	Jamie D Gerona	14
	110	Regina Renzy B. Buban	45
	103	Angela S Jarabelo	13

3 patient(s) found.

Recurrent Caries			
	102	Jamie D Gerona	13
	110	Regina Renzy B. Buban	18

2 patient(s) found.

restoration			
	110	Regina Renzy B. Buban	18,16
	103	Angela S Jarabelo	13

2 patient(s) found.

Needed Services Queries (OR)

	Id	Name	Tooth Number(s)
Class 1			
	110	Regina Renzy B. Buban	16
	103	Angela S Jarabelo	13

2 patient(s) found.

Pulp Sedation			

No results found.

Figure 56: Query patient result in DentISt

B. Procedures for Adding Section

For every new role in a section that needs to be added there must be an equivalent database role specified. The new database login role is provided and made by the database administrator. Once the database role is created and has the permission to connect to the database, Figure 57 shows how to add a role in the system.

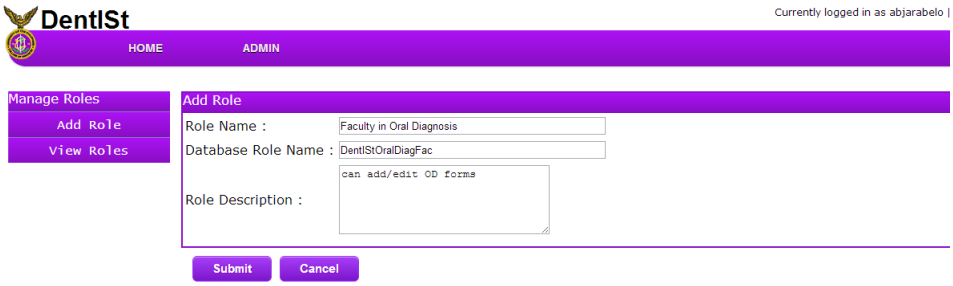


Figure 57: Add Role in DentISt

To standardize how policies are presented, XACML (XML for access control) are used to represent the policies needed for the system. Figure 58 shows an example of how a simple XACML looks like. Target element contains the subject (who can access), resource (what can be access) and action (what actions can be done to the access).

```
<Policy PolicyId="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combinin
  <Description>Grants for Roles</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentISt Admin</AttributeValue>
          <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string" />
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentISt Prostho Student</AttributeValue>
            <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
    <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentISt Operative Dent Student</AttributeValue>
            <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
    <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentISt Oral Med Fac</AttributeValue>
            <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
    <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentISt Prostho Fac</AttributeValue>
            <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
    <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">DentISt Operative Dent Fac</AttributeValue>
```

Figure 58: XAML for DentISt table Role

For the access on forms, the workflow administrator provides the database administrator with a BPMN2 export file containing information regarding the forms created. Using a stand-alone application, we can convert the BPMN2 file to XACML files per forms created. Figure 60 shows the stand-alone program opening a bpmn2 file and converting it to XACML files.

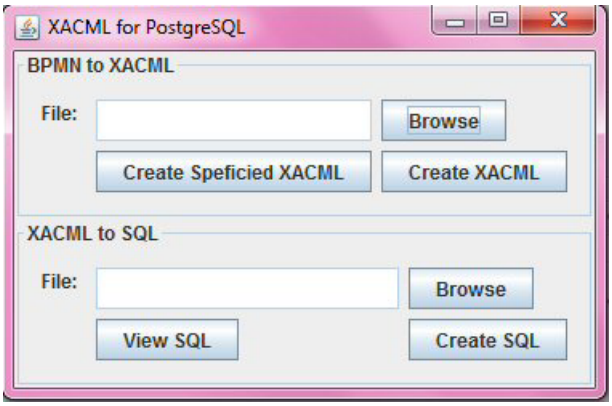


Figure 59: XACML for PostgreSQL in DentISt

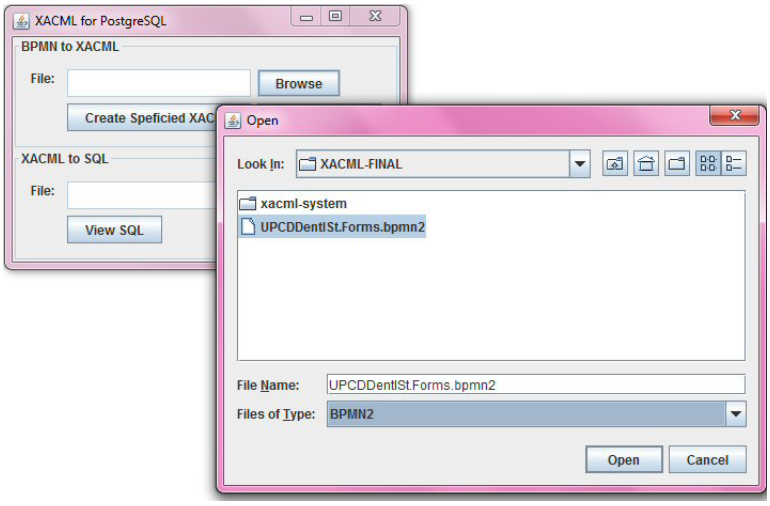


Figure 60: BPMN2 file to XACML files in DentISt

After opening the BPMN2 file, user can choose to select a certain form to create its specific XACML file or choose to create ALL the XACML files in the BPMN2 workflow file. Figure 61 shows when the user has chosen to create a specific XACML file.

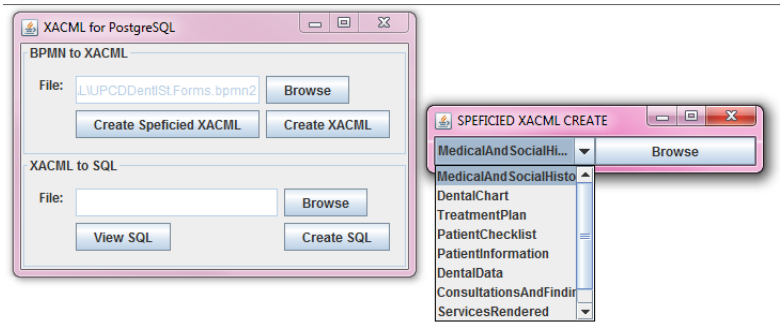


Figure 61: Create specific XACML file for a form in DentISt

Figure 62 shows how a user choose a XACML file and Figure 63 gives the equivalent generated SQL file from the file chosen.

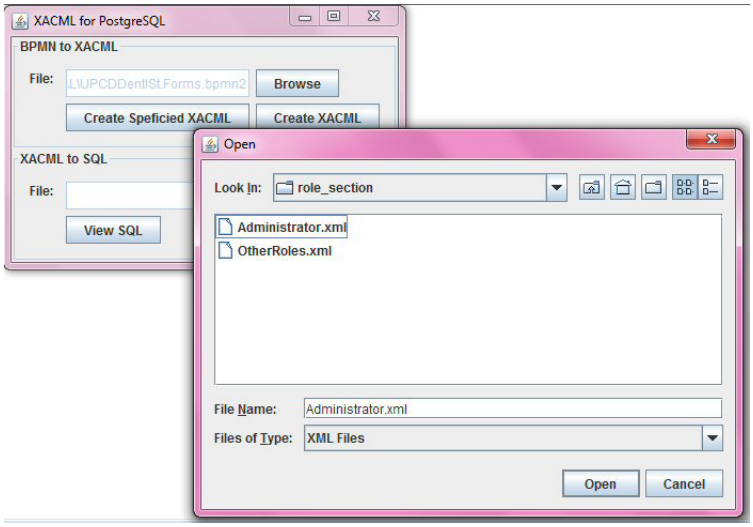


Figure 62: XACML file to SQL file in DentISt

```
GRANT select ON TABLE "user_role" TO "DentIStProsthoStudent";
GRANT select ON TABLE "user_role" TO "DentIStProsthoFac";
```

Figure 63: XACML file to SQL file in DentISt

Using the generated SQL files that provide the access control GRANTS of a role in a section, the database administrator uploads the statements to the database. A role in a section now has the minimum privileges as stated by their access control policy. Figure 64 shows an example of how privileges are minimized on each role (example is for role table).

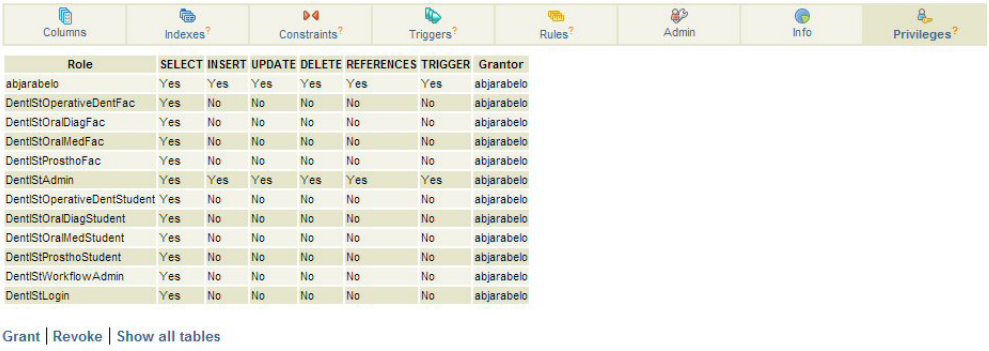


Figure 64: Privileges in role table-database of DentISt

VII. Discussion

DentISt or Dental Information System 3.0 is the 3rd version of the DentISt. DentISt is specifically created for a potential electronic patient record to be used by the College of Dentistry-University of the Philippines. The system can store patient dental records that also include some information regarding medical, social and dental history, physical assessment, vital sign, dental status chart and many more. DentISt 3.0 is divided into two modules that focus on two separate topics- the Workflow and Fine Grained Access Control Module. For the latter module, it concentrates on complying with HIPAA (Health Insurance Portability and Accountability Act) standards regarding data integrity, confidentiality and availability, as it implements some security policies regarding electronic sensitive information in the system to protect it from being accessed by unauthorized users and making it anonymous to access offenders. It is also in charge of bringing the access control from the application layer to the database thus achieving a more sound and secure access of information and making it fine-grained.

As compared to the two previous versions of DentISt, the third version does not use an OpenMRS platform thus solving some compatibility issues like the inability to use dental chart when it comes to web issues and exceptions concerning the inserting and updating of patient records using the OpenMRS functionality. Building away from the OpenMRS platform also helped in setting up a friendlier approach to its primary users that concerns with the student-faculty concept and matches the university setting of the system. Developing and adding new modules and functionalities are implemented easier as compared to using OpenMRS that lacks resources and documentations online.

DentISt 3.0 presents a different approach in using roles compare to the second version. Using the initial four sections- Oral Diagnosis, Oral Medicine, Prosthodontics and Restorative Dentistry, each section provides two different

roles which are faculty and student clinician. Faculty and student clinician are also differentiated by having the student clinician's work be approved first before being considered as a legitimate case. Entries by faculty are automatically considered as final and do not need any more approval. Aside from that, faculty can view the other clinicians in his specific section and also the statistics in its section. Statistics will help clinician and administrative staff to generate a yearly report as well to know the number of cases treated and services needed by the patient. Both clinician and faculty can query and view patient records. Given that roles are considered by what kind of clinician and what section they are in, these roles in a section have their own set of minimum privileges. To prevent access of unauthorized users, instead of the super-user or single user database approach, roles in a section are used as the users in the database. A multi-user database approach helps us ensure a more sound and secure access in the database level. A particular user with a given role can only attain their required minimum privilege according to the policy and avoid unnecessary access of some records. The system also provides two other roles initially created for DentISt which are the system administrator- that handles the assigning of users and roles and the workflow administrator- that handles the workflow of the system.

Encryption of the sensitive data according to the HIPAA policy like name, phone, address, etc. is done to provide anonymity to the patient records. With the help of the one of the many security-inspired PostgreSQL modules one and two way encryptions can be used easily with the use of stored functions, specifically made to help PostgreSQL users to prevent easy availability of patient information by converting data to non-readable information.

Development of new form in the workflow modules also produces a new set of privileges for the roles in the system. To avoid the unnecessary task of reworking the architecture of the system, access control is arranged in the database itself as compared to the usual placement in the application layer.

Using an export file that contains the name and the roles allowed in the forms, a stand-alone application generates SQL files from XACML (XML for access control policies) that handles the privileges of roles in the database tables. Strict ruling also comply in the communication between the FGAC and workflow modules. Names of the new forms are strictly used as the database table name to allow the generation of SQLs matching the policies to work effectively and efficiently.

To test for the fine- grained access control of the database, some testing has been done to show the good security of the system. Roles have their own set of minimum privileges and thus some are not allowed to do particular operations depending on its access control policy. Figure 65 shows the privileges of roles in table role.

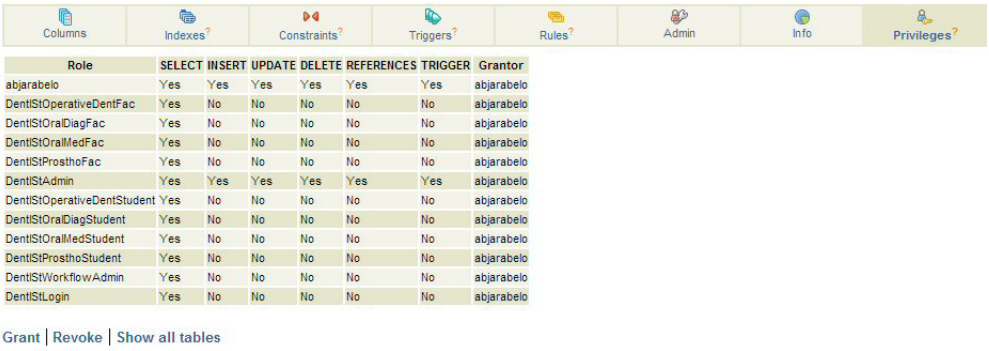


Figure 65: Privileges in role table-database of DentISt

SQL statements that contain an operation but do not have the privilege will be denied by the database. Using the role "DentIStOralDiagFac" who does not have an INSERT privilege for the table role, his call for the stored procedure of insert role will be denied. Figure 66 shows the warning message of the example.

Stored procedures are used to transfer SQL calls from the application to the database. This elevates the protection of the system from malicious attacks such as SQLIA (SQL Injection Attack) and improves productivity by not rewriting redundant statements inside application. SQL Injection attacks can be avoided

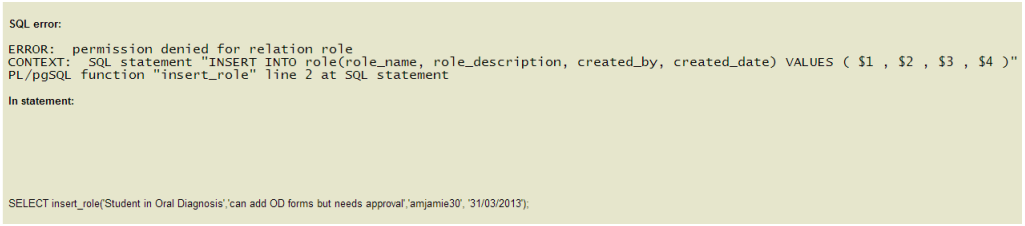


Figure 66: Privilege deny in database

by using stored procedure with its parameterized values, avoiding the use of dynamic SQL queries as much as possible and using a low privilege role in the database.

As an example of an SQL injection attack [36], A tester or hacker may try to inject SQL queries to be able to login. Figure 67 shows an example where the login input is John' or 'x'='x as username and Smith' or 'x'='x as password. Since 'x'='x' is always true, the intruder may be able to login even without knowing any username and password.

```
SELECT * FROM Users WHERE User_Name = '" & strUserName & _
        "' AND  Password = '" & strPassword & "';"

SELECT * FROM Users WHERE User_Name = 'John' or _
        'x'='x' AND Password = 'Smith' or 'x'='x';
```

Figure 67: SQL Injection example

As DentISt uses stored procedures for SQL queries, these stored procedures are designed to have parameterized values (checking what kind of field the value has) and not in dynamic SQL query format. As compare to Figure 67, login in DenISt is done by calling this stored procedure as shown in Figure 68.

Function	Arguments	Returns	Programming language
login_user	"par1" text, "par2" text	text	plpgsql

Definition

```
1   DECLARE
2           id INTEGER;
3   BEGIN
4           SELECT user_id INTO id FROM users WHERE users.username=par1 AND
5           users.password = crypt(par2, users.password);
6
7           RETURN id;
8   END
9
10
```

Function Costing

Execution Cost: 100	Result Rows: 0

Properties

Figure 68: Stored procedure for Login

To test if the stored procedure will work in the login injection, input of the

username and password above can be seen in Figure 69 while Figure 70 shows how the system does not allow a login.

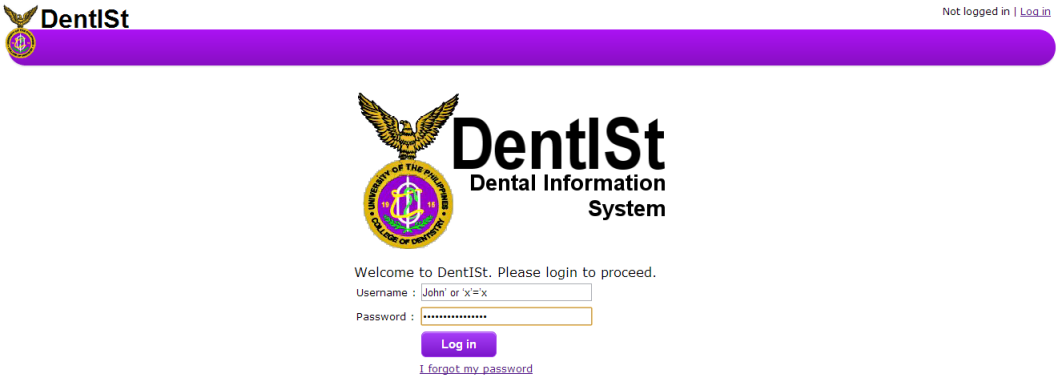


Figure 69: SQL injection and example

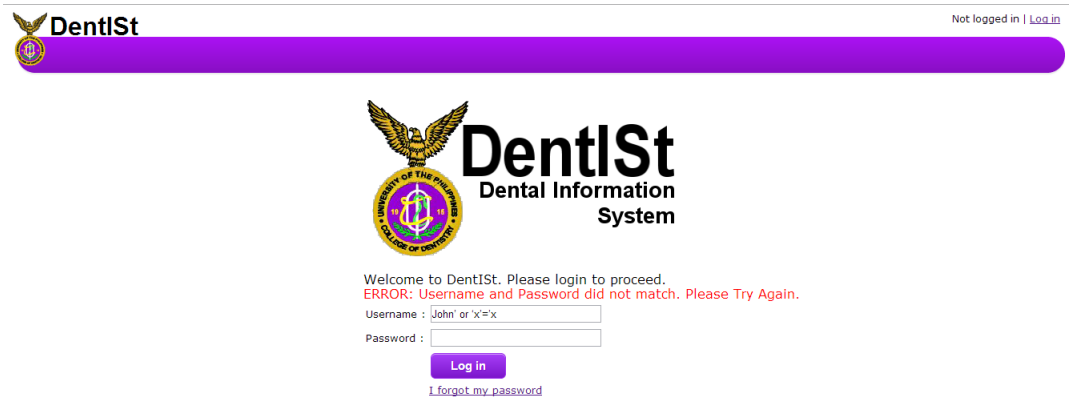


Figure 70: DentISt denies Login

Using the 'DentIStLogin' role in the database itself, Figure 71 shows how the stored procedure does not provide any result.

SQL injection can also cause serious dilemma in the database. Given any SQL query with a dynamic approach they can input in any text field an operation function (specifically a DROP or DELETE function). Since all database calls are done in a stored procedure with a non-dynamic approach, parame-

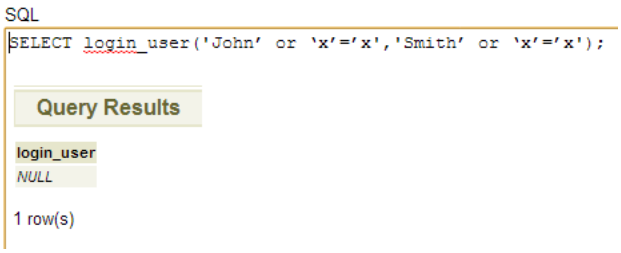


Figure 71: Stored procedure for Login in DentISt

terized values and more importantly roles used are with minimum privileges. Figure 72 shows how the database rejects the function because of the parameter input. Errors are handled accordingly in the system.

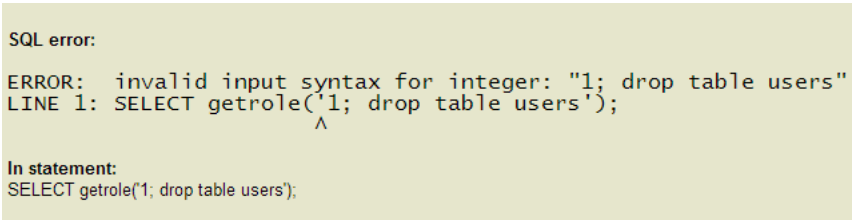


Figure 72: ERROR in stored procedure after SQL injection

As clinicians in the workflow are divided accordingly in a section, clinicians on other section cannot easily insert or update the record of a certain patient. Also with workflow's management, a clinician can claim a case of a patient and must deny other clinicians to access the records aside from viewing it.

To test if the patient records are secured from any attacks, stored procedures are provided to check for a user's access on the workflow. Figure 73 shows the stored procedure that checks if a clinician or a role is allowed to handle that task in the workflow (either the clinician owning the case or clinician in a section).Figures 74 to 79 denied permission.

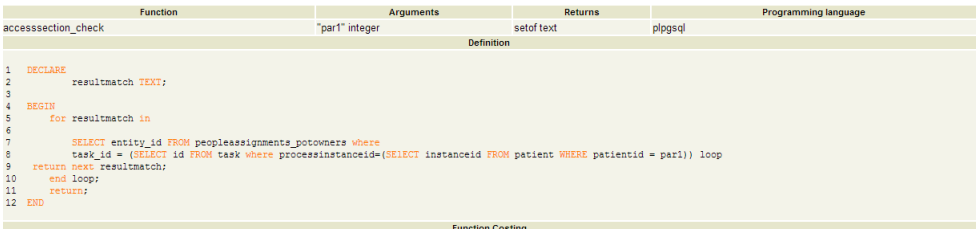


Figure 73: Stored procedure for privilege check

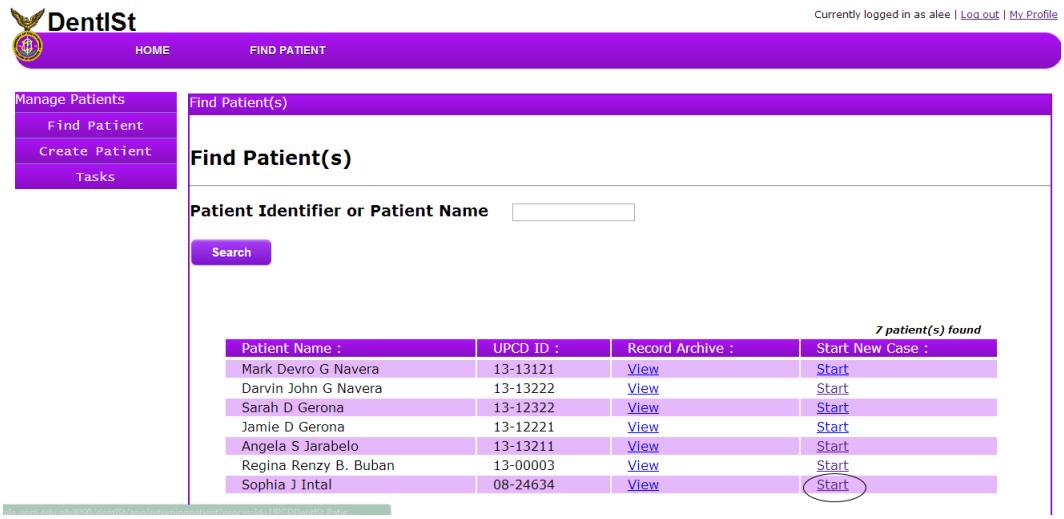


Figure 74: alee starts a new case

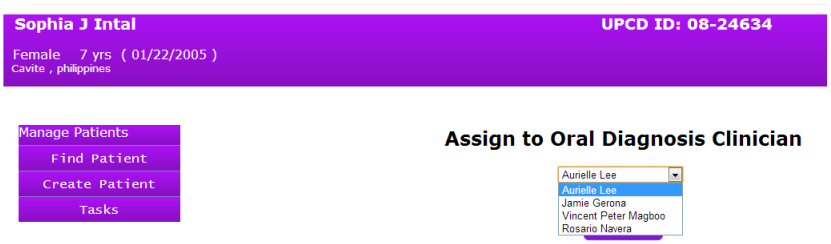


Figure 75: alee assigns a clinician to the patient

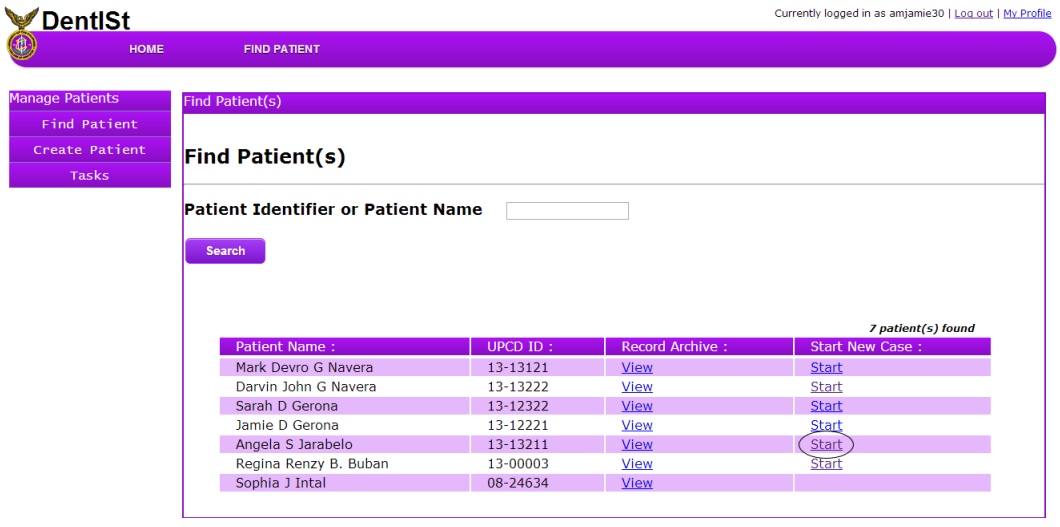


Figure 76: amjamie30 starts another case

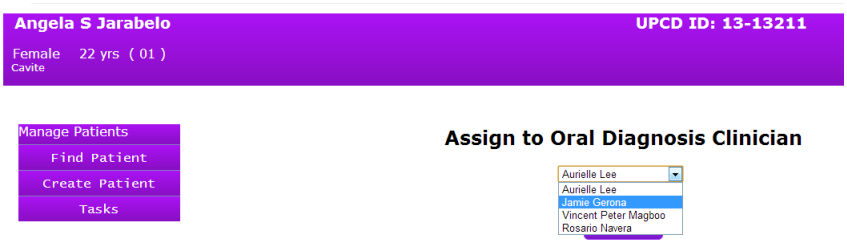


Figure 77: amjamie30 assigning a clinician to the patient

Figure 78: amjamie30 changes the patientid owned by another clinician in the URL

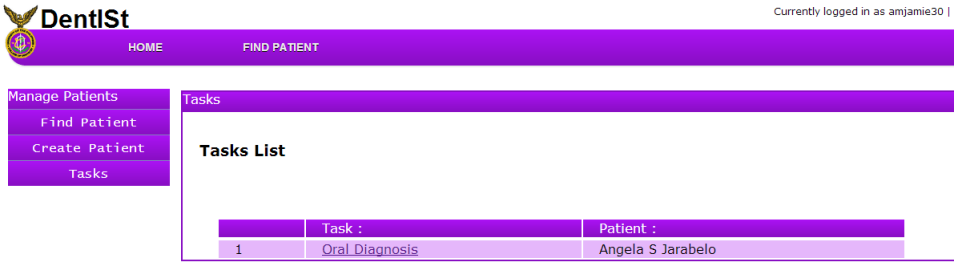


Figure 79: amjamie30 is not allowed in the record and returns to its task list

VIII. Conclusion

Dental Information System 3.0 is the third version of DentISt that offers a good management of patient dental records electronically and will help in the future in the shift from paper- based to electronic based in the College of Dentistry-University of the Philippines Manila.

Without using OpenMRS as an EMR platform in the system, deployment of the system will be easier and some compatibility issues can be solved. Some issues include the disability of some functionality in the system like for the dental chart. Dental chart has also been modified as said by Dean Vicente Medina and Dr. Charlie Atienza of UPCD to simplify its usage and to make it more readable for clinicians. Error that returns a java.lang.ClassNotFoundException exception of the OpenMRS platform when a user updates and creates a patient record also no longer exists even when Apache Tomcat is upgraded. Usability issue of the system has been change into a student- faculty approach to apply in the university setting of the system. Deployment and adding new modules functionalities may turn out easier to develop in the future.

Complying with the HIPAA policy for electronic information helps in maintaining data integrity, confidentiality and availability. Encryption of sensitive data such as the name, address, etc. is done to attain anonymity of data information by using stored functions specifically made by PostgreSQL for encrypting data. Audit trail in the system is done as well to ensure the knowledge of who accessed what and when. Also, access control that formerly resides on the core application shifted to the database layer. Access control in the database will reduce the re-working of the architecture and also helped in turning the one database user approach to a multi- user approach. A multi-user database approach assures a more fine grained access that provides the user with its set of minimum privileges according to its roles and avoids potential damage due to malicious access of using a super-user privilege. Allowing roles to access ex-

clusively what they have complies with the minimum privilege principle. This includes setting up the right privileges to roles in a section and the specific administrators.

Aside from having the initial sections of Oral Diagnosis, Oral Medicine, Restorative Dentistry and Prosthodontics, we subdivide clinicians to faculty and students clinicians. Both can manage patient records but with the student clinician having their entries be approved first. Some functionalities that have been modified are the query of patients which is done by both the faculty and student clinician to present the list of patient and a link to its specific record. For the administrative role, statistics of the whole UPCD can be generated to summarize reports for a specific period. Faculty can also view and list the clinicians in the section. Administrator roles are of two parts- the system administrator, who manages the user accounts, roles, sections and its privileges, and workflow administrator, who manages the tasks and the workflow of the system. Stored procedures are also use to avoid attacks from outside users like injection attacks and to provide a better access control in the system.

When adding a new form done in the workflow module, re-working the architecture can be lessen because of the existence of the access control in the database layer. Grants to roles for a form with its specified table can be generated using an application that converts an exported file from the workflow that contains the forms and its list of users to an XACML file. Using the XACML file, a standard XML used to proclaim access control policies, SQL files for the access of tables can be created and uploaded by the database administrator. This provides a good communication between the workflow and fine grained module when adding new functionalities or module to the system in future use.

IX. Recommendation

Other dental forms for the three other sections can still be added to the system. Tables and access control for each form can easily implemented by communicating with the workflow module. Addition of the other forms can help UPCD in slowly shifting from paper-based to electronic record.

Integration of the concept dictionary relating to UPCD can help in error checking, validation and in managing information/ data collected more easily. As statistics and query patients are organized as a standard search using the initial forms, once new forms are added, other factors can be considered to be added as well in the search for patients and statistics.

As XACML is used as the standard representation of the tables and its access control, XACML's rule element can be used as a reference for a more cell-level access control given that a file from the workflow can be exported having the field names of the particular form and also the use of the attribute of rule combinging algorithm to know if one is permitted or denied. Initial application can only generate SQL files when access control is considered permitted.

X. Bibliography

[1] T. Schleyer, "Dental Informatics: An Emerging Biomedical Informatics Discipline," *Advances in Dental Research*, vol. 17, pp. 4–8, Dec. 2003.

[2] N. Thuvin, "Managin rights in postgresql," *Dalibo*, 2011.

[3] E. Helms and L. Williams, "Evaluating access control of open source electronic health record systems," in *Proceedings of the 3rd Workshop on Software Engineering in Health Care*, SEHC '11, (New York, NY, USA), pp. 63–70, ACM, 2011.

[4] J. Eisner, "The future of dental informatics," *Eur J dent Educ*, pp. 61–69, 1999.

[5] A. J. Lee, "Developing a dental information system with openmrs (open dentis)," 2011.

[6] C. Balsita, "Dentist: Dental information system 2.0," 2012.

[7] "Openmrs: Troubleshooting your installation." `http://en.flossmanuals.net/openmrs-guide/troubleshooting/`. Accessed on October, 2012.

[8] C. L. Grand and D. Sarel, "Database security, compliance and audit," *Information Systems Control Journal*, vol. 5, 2008.

[9] G. Munoz-Cornejo, C. B. Seaman, and A. G. Koru, "An empirical investigation into the adoption of open source software in hospitals," *International Journal of Healthcare Information Systems and Informatics*, vol. 3, no. 3, pp. 16–37, 2008.

[10] A. K. Jha, C. M. DesRoches, E. G. Campbell, K. Donelan, S. R. Rao, T. G. Ferris, A. Shields, S. Rosenbaum, and D. Blumenthal, "Use of electronic health records in U.S. hospitals.," *The New England journal of medicine*, vol. 360, pp. 1628–38, Apr. 2009.

[11] J. B. Smelcar, H. Miller-Jacobs, and L. Kantrovich, "Usability of electronic medical records," *Journal of Usability Studies*, vol. 4, pp. 70–84, February 2009.

[12] C. Tanasie, "Open source medical software: Openmrs," *Open Source Science Journal*, vol. 3, no. 1, pp. 5 – 19, 2011.

[13] X. Sun, H. Wang, J. Li, and Y. Zhang, "Satisfying Privacy Requirements Before Data Anonymization," *The Computer Journal*, vol. 55, pp. 422–437, Mar. 2011.

[14] M. C. Murray, "Database security: What students need to know," *Journal of Information Technology Education*, vol. 9, 2010.

[15] C. T. Di Iorio, F. Carinci, M. Brillante, J. Azzopardi, P. Beck, N. Bratina, S. G. Cunningham, C. De Beaufort, N. Debacker, P. Jarosz-Chobot, M. Jecht, U. Lindblad, T. Moulton, Z. Metelko, A. Nagy, G. Olympios, S. Pruna, M. Rø der, S. Skeie, F. Storms, and M. Massi Benedetti, "Cross-border flow of health information: is 'privacy by design' enough? Privacy performance assessment in EUBIROD.," *European journal of public health*, May 2012.

[16] W. A. Al-Hamdani, "Cryptography based access control in healthcare web systems," in *2010 Information Security Curriculum Development Conference*, InfoSecCD '10, (New York, NY, USA), pp. 66–79, ACM, 2010.

[17] S. Braghin, A. Coen-Porisini, P. Colombo, S. Sicari, and A. Trombetta, "Introducing privacy in a hospital information system," in *Proceedings of the fourth international workshop on Software engineering for secure systems*, SESS '08, (New York, NY, USA), pp. 9–16, ACM, 2008.

[18] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending query rewriting techniques for fine-grained access control," *Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04*, p. 551, 2004.

[19] A. Roichman and E. Gudes, "Fine-grained access control to web databases," *Proceedings of the 12th ACM symposium on Access control models and technologies - SACMAT '07*, p. 31, 2007.

[20] Y. Yang, D. X.H, R. Deng, and F. Bao, "Multi-user private queries over encrypted databases," *Int. J. High Performance Computing and Networking*, vol. 1, 2008.

[21] Q. Wang, T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J.-W. Byun, "On the correctness criteria of fine-grained access control in relational databases," in *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pp. 555–566, VLDB Endowment, 2007.

[22] S. Kakade, R. Thakare, B. Sapare, and D. B. Meshram, "Maintaining stored procedures in database application," *International Journal of Computer Science and Communication Networks*, vol. 2, pp. 400–403, 2010.

[23] S. Chaudhuri, T. Dutta, and S. Sudarshan, "Fine grained authorization through predicated grants," in *ICDE*, pp. 1174–1183, 2007.

[24] "Postgres plus 8.4 vs. mysql 5.5." `http://www.enterprisedb.com`. Accessed on October, 2012.

[25] "Se-postgresql." `http://wiki.postgresql.org/wiki/SEPostgreSQLv8.4`, 2012. Accessed on October, 2012.

[26] V. Arjun, "A database level implementation to enforce fine grained access control," Master's thesis, North Carolina State University, 2008.

[27] "Dentist. what is a dentist?." `http://www.ada.org.au/dentalprofessionals/dentist.aspx`, October 2012. Accessed on October, 2012.

[28] "University of the philippines college of dentistry." `http://cd.upm.edu.ph`, October 2012. Accessed on October, 2012.

[29] "Hipaa security series," tech. rep., USA Department of Health and Human Services, 2007.

[30] M. Gregg, *CISSP Exam Cram, Second Edition.* Pearson Certification, second ed., 2009.

[31] D. Aspinall, "Security models," tech. rep., School of Informatics University of Edinburgh, 2012.

[32] "Oracle database." `http://docs.oracle.com/`, 2003. Accessed on October, 2012.

[33] C. Ardagna, S. D. C. d. V. E. Damiani, and P. Samarati, "Xml-based access control language," *Elsevier Ltd.*, 2004.

[34] S. Jahid, C. A. Gunter, I. Hoque, and H. Okhravi, "Myabdac: compiling xacml policies for attribute-based database access control," in *Proceedings of the first ACM conference on Data and application security and privacy*, CODASPY '11, (New York, NY, USA), pp. 97–108, ACM, 2011.

[35] "Postgresql: pgcrypto module." `http://www.postgresql.org/docs/8.3/static/pgcrypto.html`. Accessed on April, 2013.

[36] H. Chaudhary, "Database security testing in the light of sql injection attack," 2011.

XI. Appendix

A. XACML

Listing 1: Administrator-users

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
      urn:oasis:names:tc:xacml:2.0
      :policy:schema:os"
      RuleCombiningAlgId="
      urn:oasis:names:tc:xacml:1.0:rule-
      combining-algorithm:permit-overrides
      ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">
                DentISt Admin</
                AttributeValue>
          <SubjectAttributeDesignator
                AttributeId="role"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">users</
                AttributeValue>
          <ResourceAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:resource:resource-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">all</
                AttributeValue>
          <ActionAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:action:action-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
 <Rule RuleId="IfPermitUsers" Effect="
       Permit">
<Description>users_user_id_seq</
      Description>
<Condition>
<AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#boolean">true</
      AttributeValue>
</Condition>
</Rule>

</Policy>
```

Listing 2: Administrator-role

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
      urn:oasis:names:tc:xacml:2.0
      :policy:schema:os"
      RuleCombiningAlgId="
      urn:oasis:names:tc:xacml:1.0:rule-
      combining-algorithm:permit-overrides
      ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
```

```
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">
              DentIStAdmin</
              AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="role"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">role</
              AttributeValue>
          <ResourceAttributeDesignator
              AttributeId="
              urn:oasis:names:tc:xacml:1
              .0:resource:resource-id"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">all</
              AttributeValue>
          <ActionAttributeDesignator
              AttributeId="
              urn:oasis:names:tc:xacml:1
              .0:action:action-id"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
 <Rule RuleId="IfPermitRole" Effect="
     Permit">
<Description>role_role_id_seq</
    Description>
<Condition>
<AttributeValue DataType="http://www.w3.
    org/2001/XMLSchema#boolean">true</
    AttributeValue>
</Condition>
</Rule>

</Policy>
```

Listing 3: Administrator-section

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
    urn:oasis:names:tc:xacml:2.0
    :policy:schema:os"
    RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:1.0:rule-
    combining-algorithm:permit-overrides
    ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">
              DentIStAdmin</
              AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="role"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">section<
              /AttributeValue>
          <ResourceAttributeDesignator
              AttributeId="
              urn:oasis:names:tc:xacml:1
              .0:resource:resource-id"
```

```
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="
                urn:oasis:names:tc:xacml:1.0
                :function:string-equal">
              <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">all</
                AttributeValue>
              <ActionAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:action:action-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
      <Rule RuleId="IfPermitSection" Effect="
          Permit">
      <Description>section_section_id_seq</
          Description>
      <Condition>
      <AttributeValue DataType="http://www.w3.
          org/2001/XMLSchema#boolean">true</
          AttributeValue>
      </Condition>
      </Rule>

      </Policy>
```

Listing 4: Administrator-userrole

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
      urn:oasis:names:tc:xacml:2.0
      :policy:schema:os"
      RuleCombiningAlgId="
      urn:oasis:names:tc:xacml:1.0:rule-
      combining-algorithm:permit-overrides
      ">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="
                urn:oasis:names:tc:xacml:1.0
                :function:string-equal">
            <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">
                DentISt Admin</
                AttributeValue>
            <SubjectAttributeDesignator
                AttributeId="role"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="
                urn:oasis:names:tc:xacml:1.0
                :function:string-equal">
            <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">
                user_role</AttributeValue>
            <ResourceAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:resource:resource-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="
                urn:oasis:names:tc:xacml:1.0
                :function:string-equal">
            <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">all</
                AttributeValue>
            <ActionAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:action:action-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Rule RuleId="IfPermitUserRole" Effect="
        Permit">
```

```
<Condition>
<AttributeValue DataType="http://www.w3.
    org/2001/XMLSchema#boolean">true</
    AttributeValue>
</Condition>
</Rule>

</Policy>
```

Listing 5: Administrator-databaserole

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
    urn:oasis:names:tc:xacml:2.0
    :policy:schema:os"
    RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:1.0:rule-
    combining-algorithm:permit-overrides
    ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">
              DentIStAdmin</
              AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="role"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">
              database_role</
              AttributeValue>
          <ResourceAttributeDesignator
              AttributeId="
              urn:oasis:names:tc:xacml:1
              .0:resource:resource-id"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">all</
              AttributeValue>
          <ActionAttributeDesignator
              AttributeId="
              urn:oasis:names:tc:xacml:1
              .0:action:action-id"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
 <Rule RuleId="IfPermitDatabaseRole"
      Effect="Permit">
<Condition>
<AttributeValue DataType="http://www.w3.
    org/2001/XMLSchema#boolean">true</
    AttributeValue>
</Condition>
</Rule>

</Policy>
```

Listing 6: OtherRoles-users

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
    urn:oasis:names:tc:xacml:2.0
    :policy:schema:os"
    RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:1.0:rule-
    combining-algorithm:permit-overrides
    ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
```

```
                :function:string-equal">
            <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">
                DentIStAdmin</
                AttributeValue>
            <SubjectAttributeDesignator
                AttributeId="role"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
          </SubjectMatch>
        </Subject>
    <Subject>
    <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
     <AttributeValue DataType="http://www.
        w3.org/2001/XMLSchema#string">
        DentIStProsthoStudent</
        AttributeValue>
     <SubjectAttributeDesignator
        AttributeId="role" DataType="
        http://www.w3.org/2001/XMLSchema#
        string"/>
    </SubjectMatch>
   </Subject>
   <Subject>
    <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
     <AttributeValue DataType="http://www.
        w3.org/2001/XMLSchema#string">
        DentIStOperativeDentStudent</
        AttributeValue>
     <SubjectAttributeDesignator
        AttributeId="role" DataType="
        http://www.w3.org/2001/XMLSchema#
        string"/>
    </SubjectMatch>
   </Subject>
  <Subject>
  <SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStOralMedFac</AttributeValue>
  <SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStProsthoFac</AttributeValue>
  <SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStOperativeDentFac</
      AttributeValue>
  <SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStWorkflowAdmin</AttributeValue
      >
  <SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStOralMedStudent</
      AttributeValue>
  <SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStOralDiagStudent</
      AttributeValue>
```

```
<SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
      urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
<AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#string">
      DentIStOralDiagFac</AttributeValue>
<SubjectAttributeDesignator AttributeId="
      role" DataType="http://www.w3.org
      /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">users</
                AttributeValue>
          <ResourceAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:resource:resource-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">select</
                AttributeValue>
          <ActionAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:action:action-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
        </ActionMatch>
      </Action>
      <Action>
        <ActionMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">update</
                AttributeValue>
          <ActionAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:action:action-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
 <Rule RuleId="IfPermitUsers" Effect="
       Permit">
<Description>users_user_id_seq</
      Description>
<Condition>
<AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#boolean">true</
      AttributeValue>
</Condition>
</Rule>

</Policy>
```

Listing 7: OtherRoles-role

```
<Policy PolicyId="
      urn:oasis:names:tc:xacml:2.0
      :policy:schema:os"
      RuleCombiningAlgId="
      urn:oasis:names:tc:xacml:1.0:rule-
      combining-algorithm:permit-overrides
      ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
          <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">
                DentIStAdmin</
                AttributeValue>
          <SubjectAttributeDesignator
                AttributeId="role"
```

```
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
            </SubjectMatch>
          </Subject>
  <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStProsthoStudent</
         AttributeValue>
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStOperativeDentStudent</
         AttributeValue>
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStOralMedFac</AttributeValue
         >
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStProsthoFac</AttributeValue
         >
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStOperativeDentFac</
         AttributeValue>
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStWorkflowAdmin</
         AttributeValue>
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStOralMedStudent</
         AttributeValue>
   <SubjectAttributeDesignator
         AttributeId="role" DataType="
         http://www.w3.org/2001/XMLSchema#
         string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
   <AttributeValue DataType="http://www.
         w3.org/2001/XMLSchema#string">
         DentIStOralDiagStudent</
         AttributeValue>
   <SubjectAttributeDesignator
```

```
        AttributeId="role" DataType="
        http://www.w3.org/2001/XMLSchema#
        string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
       urn:oasis:names:tc:xacml:1.0
       :function:string-equal">
   <AttributeValue DataType="http://www.
        w3.org/2001/XMLSchema#string">
        DentIStOralDiagFac</
        AttributeValue>
   <SubjectAttributeDesignator
        AttributeId="role" DataType="
        http://www.w3.org/2001/XMLSchema#
        string"/>
  </SubjectMatch>
 </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
             urn:oasis:names:tc:xacml:1.0
             :function:string-equal">
          <AttributeValue DataType="http:
               //www.w3.org/2001/
               XMLSchema#string">role</
               AttributeValue>
          <ResourceAttributeDesignator
               AttributeId="
               urn:oasis:names:tc:xacml:1
               .0:resource:resource-id"
               DataType="http://www.w3.
               org/2001/XMLSchema#string"
                />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="
             urn:oasis:names:tc:xacml:1.0
             :function:string-equal">
          <AttributeValue DataType="http:
               //www.w3.org/2001/
               XMLSchema#string">select</
               AttributeValue>
          <ActionAttributeDesignator
               AttributeId="
               urn:oasis:names:tc:xacml:1
               .0:action:action-id"
               DataType="http://www.w3.
               org/2001/XMLSchema#string"
                />
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="IfPermitRole" Effect="
       Permit">
<Description>role_role_id_seq</
     Description>
<Condition>
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#boolean">true</
     AttributeValue>
</Condition>
</Rule>
</Policy>
```

Listing 8: OtherRoles-section

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="
     urn:oasis:names:tc:xacml:2.0
     :policy:schema:os"
     RuleCombiningAlgId="
     urn:oasis:names:tc:xacml:1.0:rule-
     combining-algorithm:permit-overrides
     ">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="
             urn:oasis:names:tc:xacml:1.0
             :function:string-equal">
          <AttributeValue DataType="http:
               //www.w3.org/2001/
               XMLSchema#string">
               DentIStAdmin</
               AttributeValue>
          <SubjectAttributeDesignator
               AttributeId="role"
               DataType="http://www.w3.
               org/2001/XMLSchema#string"
                />
        </SubjectMatch>
      </Subject>
  <Subject>
  <SubjectMatch MatchId="
       urn:oasis:names:tc:xacml:1.0
       :function:string-equal">
   <AttributeValue DataType="http://www.
        w3.org/2001/XMLSchema#string">
        DentIStProsthoStudent</
        AttributeValue>
   <SubjectAttributeDesignator
        AttributeId="role" DataType="
```

```
        http://www.w3.org/2001/XMLSchema#
        string"/>
  </SubjectMatch>
 </Subject>
 <Subject>
  <SubjectMatch MatchId="
       urn:oasis:names:tc:xacml:1.0
       :function:string-equal">
   <AttributeValue DataType="http://www.
        w3.org/2001/XMLSchema#string">
        DentIStOperativeDentStudent</
        AttributeValue>
   <SubjectAttributeDesignator
        AttributeId="role" DataType="
        http://www.w3.org/2001/XMLSchema#
        string"/>
  </SubjectMatch>
 </Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStOralMedFac</AttributeValue>
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStProsthoFac</AttributeValue>
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStOperativeDentFac</
     AttributeValue>
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStWorkflowAdmin</AttributeValue
     >
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStOralMedStudent</
     AttributeValue>
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStOralDiagStudent</
     AttributeValue>
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
<Subject>
<SubjectMatch MatchId="
     urn:oasis:names:tc:xacml:1.0
     :function:string-equal">
<AttributeValue DataType="http://www.w3.
     org/2001/XMLSchema#string">
     DentIStOralDiagFac</AttributeValue>
<SubjectAttributeDesignator AttributeId="
     role" DataType="http://www.w3.org
     /2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
             urn:oasis:names:tc:xacml:1.0
             :function:string-equal">
```

```
            <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">section<
                /AttributeValue>
            <ResourceAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:resource:resource-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="
              urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http:
                //www.w3.org/2001/
                XMLSchema#string">select</
                AttributeValue>
            <ActionAttributeDesignator
                AttributeId="
                urn:oasis:names:tc:xacml:1
                .0:action:action-id"
                DataType="http://www.w3.
                org/2001/XMLSchema#string"
                 />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Rule RuleId="IfPermitSection" Effect="
        Permit">
  <Description>section_section_id_seq</
      Description>
  <Condition>
  <AttributeValue DataType="http://www.w3.
      org/2001/XMLSchema#boolean">true</
      AttributeValue>
  </Condition>
  </Rule>
  </Policy>
```

Listing 9: OtherRoles-userrole

```
<Policy PolicyId="
    urn:oasis:names:tc:xacml:2.0
    :policy:schema:os"
    RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:1.0:rule-
    combining-algorithm:permit-overrides
    ">
  <Target>
    <Subjects>
    <Subject>
    <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
      <AttributeValue DataType="http://www.
          w3.org/2001/XMLSchema#string">
          DentIStProsthoStudent</
          AttributeValue>
      <SubjectAttributeDesignator
          AttributeId="role" DataType="
          http://www.w3.org/2001/XMLSchema#
          string"/>
    </SubjectMatch>
  </Subject>
  <Subject>
    <SubjectMatch MatchId="
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
      <AttributeValue DataType="http://www.
          w3.org/2001/XMLSchema#string">
          DentIStProsthoFac</AttributeValue
          >
      <SubjectAttributeDesignator
          AttributeId="role" DataType="
          http://www.w3.org/2001/XMLSchema#
          string"/>
    </SubjectMatch>
  </Subject>
</Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="
            urn:oasis:names:tc:xacml:1.0
            :function:string-equal">
          <AttributeValue DataType="http:
              //www.w3.org/2001/
              XMLSchema#string">
              user_role</AttributeValue>
          <ResourceAttributeDesignator
              AttributeId="
              urn:oasis:names:tc:xacml:1
              .0:resource:resource-id"
              DataType="http://www.w3.
              org/2001/XMLSchema#string"
               />
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="
```

```
        urn:oasis:names:tc:xacml:1.0
        :function:string-equal">
      <AttributeValue DataType="http:
          //www.w3.org/2001/
          XMLSchema#string">select</
          AttributeValue>
      <ActionAttributeDesignator
          AttributeId="
          urn:oasis:names:tc:xacml:1
          .0:action:action-id"
          DataType="http://www.w3.
          org/2001/XMLSchema#string"
           />
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
</Policy>
```

B. Source Code

Listing 10: AdminController

```
package com.dentist.version.three.
    controller;

import java.util.ArrayList;

import javax.servlet.http.
    HttpServletRequest;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.
    Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
    annotation.ModelAttribute;
import org.springframework.web.bind.
    annotation.RequestMapping;
import org.springframework.web.bind.
    annotation.RequestMethod;

import com.dentist.version.three.db.
    AdminSP;
import com.dentist.version.three.db.
    UserSP;
import com.dentist.version.three.form.
    AuditTrail;
import com.dentist.version.three.form.
    Configuration;
import com.dentist.version.three.form.
    DeleteFunction;
import com.dentist.version.three.form.
    EmailAddress;

@Controller
public class AdminController {

 @RequestMapping(value = "/
     changeEmailAdmin.html", method =
     RequestMethod.GET)
 public String emailForm(Model model,
     HttpServletRequest request,
     HttpServletResponse response,
     HttpSession session) throws
     Exception {

  session=request.getSession(false);

  EmailAddress emailAddress= new
      EmailAddress();
  ArrayList<Configuration> allConfigList=
      new ArrayList<Configuration>();
  AdminSP adminSP= new AdminSP();
  adminSP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
  allConfigList= adminSP.allConfigList();

  for(Configuration config: allConfigList
      ){
   if(config.getName().equals("smtp_from"
       )){
    emailAddress.setSmtp_from(config.
        getName_value());
   }
   if(config.getName().equals("smtp_subj"
       )){
    emailAddress.setSmtp_subj(config.
        getName_value());
   }
   if(config.getName().equals("
       smtp_message")){
    emailAddress.setSmtp_message(config.
        getName_value());
   }
  }



  model.addAttribute("emailAddress",
      emailAddress);
  return "changeEmailAdmin";


 }

 @RequestMapping(value = "/
     changeEmailAdmin.html", method =
     RequestMethod.POST)
 public String emailoutput(
     @ModelAttribute("emailAddress")
     EmailAddress emailAddress, Model
     model, HttpServletRequest request,
     HttpServletResponse response,
     HttpSession session){

  session=request.getSession(false);

  AdminSP adminSP= new AdminSP();


  try {
   adminSP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
   adminSP.changeEmailAdd(emailAddress.
```

```
        getSmtp_from(),emailAddress.
        getSmtp_subj(),emailAddress.
        getSmtp_message());
   String success="SUCCESS: You have
        successfully changed your
        password.";
          model.addAttribute("success",
              success);
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM_ERROR");
   return "Error";
  }




  model.addAttribute("emailAddress",
       emailAddress);
  return "changeEmailAdmin";

 }
 @RequestMapping(value = "/auditTrailView
      .html", method = RequestMethod.GET)
 public String auditTrailView(Model model
      , HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception {

  session=request.getSession(false);
  ArrayList<AuditTrail> allAuditList= new
        ArrayList<AuditTrail>();
  ArrayList<String> dentistTables= new
       ArrayList<String>();
  dentistTables.add("ALL");
  AdminSP adminSP= new AdminSP();


  try {



  ArrayList<String> tempTables=adminSP.
       allDentistTables();

   for(int i=0; i<tempTables.size();i++){
    if(tempTables.get(i).equalsIgnoreCase
         ("audittrail"))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("caries_status"
         ))
     tempTables.remove(i);
    else if (tempTables.get(i).trim().
         equalsIgnoreCase("
         recurrent_status"))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("
         restoration_status"))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("configuration"
         ))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("database_role"
         ))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("
         database_tomcat_role"))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("approveupdates
         "))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("
         processinstanceinfo"))
     tempTables.remove(i);
    else if (tempTables.get(i).
         equalsIgnoreCase("
         AdditionalDemographics"))
     tempTables.remove(i);
    else
     dentistTables.add(tempTables.get(i))
          ;
   }
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   System.out.println("BOOM_ERROR");
   return "Error";
  }
   ArrayList<String> actions= new
        ArrayList<String>();
      actions.add("ALL");
     actions.add("INSERT");
     actions.add("SELECT");
     actions.add("UPDATE");
     actions.add("DELETE");




    model.addAttribute("actions", actions
         );
  model.addAttribute("auditTrail", new
       AuditTrail());
```

```
 model.addAttribute("allAuditList",
      allAuditList);
 model.addAttribute("dentistTables",
      dentistTables);

 return "auditTrail";
}


@RequestMapping(value = "/
     viewSearchAudit.html", method =
     RequestMethod.GET)
public String viewSearchAudit(
     @ModelAttribute AuditTrail
     auditTrail,HttpServletRequest
     request,
         HttpServletResponse response,
              Model model,HttpSession
              session) throws Exception {

 session=request.getSession(false);

 ArrayList<String> dentistTables= new
      ArrayList<String>();
 dentistTables.add("ALL");
 AdminSP adminSP= new AdminSP();
 String nameSearch= "%"+request.
      getParameter("name")+"%";
 String actionSearch= request.
      getParameter("action");
 String action_formSearch= request.
      getParameter("action_form");

 System.out.println("CHECK::_"+
      nameSearch);
 ArrayList<AuditTrail> allAuditList= new
       ArrayList<AuditTrail>();

 try {
  adminSP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

  ArrayList<AuditTrail> tempAudit=
       adminSP.searchAuditTrail(
       nameSearch,actionSearch);

  if(action_formSearch.equalsIgnoreCase(
       "ALL")){
   for(int i=0;i<tempAudit.size();i++){
    if(checkWithinDate(tempAudit.get(i).
         getAction_date().trim(),request
         ))
     allAuditList.add(tempAudit.get(i));
   }
  }else{
 for(int i=0;i<tempAudit.size();i++){
  if(tempAudit.get(i).getAction_form()
       .trim().equalsIgnoreCase(
       action_formSearch) &&
    checkWithinDate(tempAudit.get(i).
         getAction_date().trim(),
         request))
   allAuditList.add(tempAudit.get(i));
 }

 auditTrail.setAction_form(
      action_formSearch);
 auditTrail.setAction(actionSearch);
}
ArrayList<String> tempTables=adminSP.
     allDentistTables();

for(int i=0; i<tempTables.size();i++){
 if(tempTables.get(i).equalsIgnoreCase
      ("audittrail"))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("caries_status"
      ))
  tempTables.remove(i);
 else if (tempTables.get(i).trim().
      equalsIgnoreCase("
      recurrent_status"))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("
      restoration_status"))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("configuration"
      ))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("database_role"
      ))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("
      database_tomcat_role"))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("approveupdates
      "))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("
      processinstanceinfo"))
  tempTables.remove(i);
 else if (tempTables.get(i).
      equalsIgnoreCase("
```

```
        AdditionalDemographics"))
    tempTables.remove(i);
   else
    dentistTables.add(tempTables.get(i))
        ;
  }
 } catch (Exception e) {
  e.printStackTrace();
  System.out.println("BOOM_ERROR");
  return "Error";
 }

  ArrayList<String> actions= new
      ArrayList<String>();
     actions.add("ALL");
    actions.add("INSERT");
    actions.add("SELECT");
    actions.add("UPDATE");
    actions.add("DELETE");

   model.addAttribute("actions", actions
       );

 model.addAttribute("allAuditList",
      allAuditList);
 model.addAttribute("auditTrail",
      auditTrail);
 model.addAttribute("dentistTables",
      dentistTables);



 return "auditTrail";

}

protected boolean checkWithinDate(String
      date, HttpServletRequest request)
    throws Exception {
 String[] datefrom= new String[3];
 String[] dateto= new String[3];
 if(request.getParameter("datefrom")!=
      null || !request.getParameter("
      datefrom").equals("")
   || request.getParameter("datefrom").
       length()>0 || !request.
       getParameter("datefrom").isEmpty
       ()){
  datefrom = request.getParameter("
      datefrom").split("/");
 }
 else
  return true;
 if(request.getParameter("dateto")!=null
       || !request.getParameter("dateto
      ").equals("")
   || request.getParameter("dateto").
       length()>0 || !request.
       getParameter("date").isEmpty()){
  dateto = request.getParameter("dateto"
      ).split("/");
 }
 else
  return true;

 if(datefrom[0].isEmpty() || dateto[0].
     isEmpty())
  return true;

 int ddf = Integer.parseInt(datefrom[0])
     ;
 int mmf = Integer.parseInt(datefrom[1])
     ;
 int yyf = Integer.parseInt(datefrom[2])
     ;
 int ddt = Integer.parseInt(dateto[0]);
 int mmt = Integer.parseInt(dateto[1]);
 int yyt = Integer.parseInt(dateto[2]);
 if(date==null)
  return false;
 String[] finalDate= new String[3];
 finalDate=date.split("/");
 int dd = Integer.parseInt(finalDate[0])
     ;
 int mm = Integer.parseInt(finalDate[1])
     ;
 int yy =0;
 if(finalDate[2].length()>2){
  String[] temp= finalDate[2].split("_")
      ;

  yy=Integer.parseInt(temp[0]);
 }
 else{
  yy=Integer.parseInt(finalDate[2]);
 }

 System.out.println("CHECK_:"+ddf+"-"+
     mmf+"-"+yyf);
 System.out.println("CHECK_:"+ddt+"-"+
     mmt+"-"+yyt);
 System.out.println("CHECK_:"+dd+"-"+mm+
     "-"+yy);
 if(yyf < yy && yy < yyt) {
  return true;
 }
 else if(yyf == yy && yy < yyt){
  if(mmf < mm) {
   return true;
  }
  else if(mmf == mm){
   if(ddf <= dd)
    return true;
```

```
    }
   }
   else if(yyt == yy && yyf < yy){
    if(mm < mmt) {
     return true;
    }
    else if(mm == mmt){
     if(dd <= ddt)
      return true;
    }
   }
   else if(yyf == yy && yy == yyt){
    if(mmf < mm && mm < mmt) {
     return true;
    }
    else if(mmf == mm && mm < mmt){
     if(ddf <= dd)
      return true;
    }
    else if(mmf < mm && mm == mmt){
     if(dd <= ddt)
      return true;
    }
    else if(mmf == mm && mm == mmt){
     if(ddf <= dd && dd <= ddt)
      return true;
    }
   }
   return false;
  }

}
```

Listing 11: AdminSP

```
package com.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.ArrayList;

import com.dentist.version.three.form.
    AuditTrail;
import com.dentist.version.three.form.
    Configuration;
import com.dentist.version.three.form.
    EmailAddress;
import com.dentist.version.three.form.
    User;
import com.dentist.version.three.mapper.
    AuditMapper;
import com.dentist.version.three.mapper.
    UserMapper;

public class AdminSP {
 private String database_username;
 private final String database_password="
     yes?bakitpo?";

 public void setDatabase_username(String
     database_username) {
  this.database_username =
      database_username;
 }


 public String getDatabase_username() {
  return database_username;
 }

 //list of config
 public ArrayList<Configuration>
     allConfigList() throws Exception{
  ArrayList<Configuration> listConfig=
      new ArrayList<Configuration>();
  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  conn.setAutoCommit(false);
    CallableStatement calstat=conn.
        prepareCall("{call_listconfig()}
        ");

     ResultSet rs = calstat.executeQuery
         ();

     while(rs.next()){
    Configuration config= new
        Configuration();

    config.setName(rs.getString("name"))
        ;
    config.setName_value(rs.getString("
        name_value"));

    listConfig.add(config);
        }


    conn.close();
    calstat.close();
    System.out.println("Successful_call_
        for_listconfig_function");
```

```
        return listConfig;
    }

    //change email address in profile
     public void changeEmailAdd(String
          smtp_from, String smtp_subj,String
           smtp_message) throws Exception{

      Class.forName("org.postgresql.Driver")
           .newInstance();
        Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt",
            database_username,
            database_password);

          CallableStatement calstat=conn.
               prepareCall("{call_
               update_config(?,?,?)}");
               calstat.setString(1,smtp_from)
                    ;
               calstat.setString(2,smtp_subj)
                    ;
               calstat.setString(3,
                    smtp_message);

               ResultSet rs = calstat.
                    executeQuery();

               conn.close();
               calstat.close();
               System.out.println("Your_data_
                    has_been_inserted_into_
                    table.");


      }

    //list the configuration settings for
         email
     public ArrayList<Configuration>
          configListChange() throws
          Exception{
      ArrayList<Configuration> listConfig=
           new ArrayList<Configuration>();
      Class.forName("org.postgresql.Driver")
           .newInstance();
        Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt","
            postgres","root");
       conn.setAutoCommit(false);
        CallableStatement calstat=conn.
            prepareCall("{call_listconfig()
            }");

          ResultSet rs = calstat.
               executeQuery();

           while(rs.next()){
          Configuration config= new
               Configuration();

          config.setName(rs.getString("name")
               );
          config.setName_value(rs.getString("
               name_value"));


          listConfig.add(config);
               }


          conn.close();
          calstat.close();
          System.out.println("Successful_call
               _for_listconfig_function");
           return listConfig;
    }

    //INSERT AUDITTRAIL
     public void insertAuditTrail(String
          name, String action,String
          action_performed, String
          action_form, String action_date)
          throws Exception{

      Class.forName("org.postgresql.Driver"
           ).newInstance();
        Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt",
            database_username,
            database_password);

         CallableStatement calstat=conn.
              prepareCall("{call_
              insert_audittrail(?,?,?,?,?)
              }");
              calstat.setString(1,name);
              calstat.setString(2,action);
              calstat.setString(3,
                   action_performed);
              calstat.setString(4,
                   action_form);
              calstat.setString(5,
                   action_date);

              ResultSet rs = calstat.
                   executeQuery();


              conn.close();
```

```
            calstat.close();
            System.out.println("Your data
                has been inserted into
                table.");


    }
//list all audittrail
    public ArrayList<AuditTrail>
        allAuditTrail() throws Exception{
     ArrayList<AuditTrail> listAudit= new
         ArrayList<AuditTrail>();
     Class.forName("org.postgresql.Driver"
         ).newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt",
            database_username,
            database_password);
      conn.setAutoCommit(false);
       CallableStatement calstat=conn.
            prepareCall("{call
            listauditrail()}");

        ResultSet rs = calstat.
             executeQuery();

        while(rs.next()){
       AuditTrail auditTrail= new
            AuditTrail();
            AuditMapper auditMap= new
                AuditMapper();
            auditTrail= auditMap.mapRow(
                rs, 6);


            listAudit.add(auditTrail);
           }


       conn.close();
       calstat.close();
       System.out.println("Successful
            call for listauditrail
            function");
        return listAudit;
    }


    //list of search audit
    public ArrayList<AuditTrail>
        searchAuditTrail(String
        specifiedSearch, String
        actionSearch) throws Exception{
     ArrayList<AuditTrail> listAudit= new
         ArrayList<AuditTrail>();
     Class.forName("org.postgresql.Driver"
         ).newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt",
            database_username,
            database_password);
      conn.setAutoCommit(false);
       CallableStatement calstat=conn.
            prepareCall("{call
            listsearchaudit(?,?)}");
        calstat.setString(1,
             specifiedSearch);
        calstat.setString(2, actionSearch
             );

        ResultSet rs = calstat.
             executeQuery();

        while(rs.next()){
            AuditTrail auditTrail= new
                AuditTrail();
            AuditMapper auditmap= new
                AuditMapper();
          auditTrail= auditmap.mapRow(rs,
               6);


          listAudit.add(auditTrail);
           }


       conn.close();
       calstat.close();
       System.out.println("Successful
            call for listspecifiedusers
            function");
        return listAudit;
    }


    //check for privilege count a role
    public int allTablesList(String
        database_role) throws Exception{

     int privilegeCount=0;

     Class.forName("org.postgresql.Driver"
         ).newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt","
            postgres","root");
      conn.setAutoCommit(false);
       CallableStatement calstat=conn.
            prepareCall("{call
            setUserRoleResult(?)}");
        calstat.setString(1,
```

```
            database_role);

        ResultSet rs = calstat.
            executeQuery();

        while(rs.next()){
       privilegeCount=rs.getInt(1);
           }


       conn.close();
       calstat.close();
       System.out.println("Successful
            call for listconfig function"
            );
        return privilegeCount;
   }

   //privilege check
   public boolean checkPrivilege(String
        database_role, String table_name,
         String action) throws Exception{


    Class.forName("org.postgresql.Driver"
         ).newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql
           ://localhost:5432/DentISt","
           postgres","root");
     conn.setAutoCommit(false);
      CallableStatement calstat=conn.
           prepareCall("{call
           checkprivilege(?,?,?)}");
        calstat.setString(1,
             database_role);
        calstat.setString(2, table_name);
        calstat.setString(3, action);

        ResultSet rs = calstat.
            executeQuery();
        boolean privilegeCheck=false;
        while(rs.next()){
       privilegeCheck=rs.getBoolean(1);
       //tableNames.add(rs.getString(1));
        System.out.println("Privilege:"+
            rs.getBoolean(1));
           }


       conn.close();
       calstat.close();
       System.out.println("Successful
            call for checkprivilege
            function");
        return privilegeCheck;
   }

   //list all dentist tables
   public ArrayList<String>
        allDentistTables() throws
        Exception{
    ArrayList<String> list= new ArrayList
        <String>();
    Class.forName("org.postgresql.Driver"
         ).newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql
           ://localhost:5432/DentISt","
           postgres","root");
     conn.setAutoCommit(false);
      CallableStatement calstat=conn.
           prepareCall("{call
           getdentisttables()}");

       ResultSet rs = calstat.
            executeQuery();

       while(rs.next()){

       list.add(rs.getString(1));
           }


       conn.close();
       calstat.close();
       System.out.println("Successful
            call for listaudittrail
            function");
        return list;
   }
}
```

Listing 12: AuditMapper

```
package com.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
    RowMapper;

import com.dentist.version.three.form.
    AuditTrail;

public class AuditMapper implements
    RowMapper<AuditTrail> {
   public AuditTrail mapRow(ResultSet rs
```

```
        , int rowNum) throws
        SQLException {
    AuditTrail auditTrail = new
        AuditTrail();
    auditTrail.setAuditTrail_id(rs.
        getInt("auditTrail_id"));
    auditTrail.setName(rs.getString("
        name"));
      auditTrail.setAction(rs.getString
          ("action"));
      auditTrail.setAction_performed(rs
          .getString("action_performed
          "));
      auditTrail.setAction_form(rs.
          getString("action_form"));
      auditTrail.setAction_date(rs.
          getString("action_date"));

      return auditTrail;
    }
  }
```

Listing 13: AuditTrail

```
package com.dentist.version.three.form;

public class AuditTrail {
 private int auditTrail_id;
 private String name;
 private String action;
 private String action_performed;
 private String action_form;
 private String action_date;
 public void setAuditTrail_id(int
     auditTrail_id) {
  this.auditTrail_id = auditTrail_id;
 }
 public int getAuditTrail_id() {
  return auditTrail_id;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getName() {
  return name;
 }
 public void setAction(String action) {
  this.action = action;
 }
 public String getAction() {
  return action;
 }
 public void setAction_performed(String
     action_performed) {
  this.action_performed =
      action_performed;
 }
 public String getAction_performed() {
  return action_performed;
 }
 public void setAction_date(String
     action_date) {
  this.action_date = action_date;
 }
 public String getAction_date() {
  return action_date;
 }
 public void setAction_form(String
     action_form) {
  this.action_form = action_form;
 }
 public String getAction_form() {
  return action_form;
 }

}
```

Listing 14: Configuration

```
package com.dentist.version.three.form;

public class Configuration {
 private String name;
 private String name_value;
 public void setName(String name) {
  this.name = name;
 }
 public String getName() {
  return name;
 }
 public void setName_value(String
     name_value) {
  this.name_value = name_value;
 }
 public String getName_value() {
  return name_value;
 }
}
```

Listing 15: ConsultationMapper

```
package com.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
    RowMapper;

import com.dentist.version.three.form.
    ConsultationsReferrals;

public class ConsultationMapper
    implements RowMapper<
    ConsultationsReferrals>{
 public ConsultationsReferrals mapRow(
     ResultSet rs, int rowNum) throws
     SQLException {

  ConsultationsReferrals consult= new
      ConsultationsReferrals();

   consult.setPatientid(rs.getInt("
       patientid"));
   consult.setConsultationid(rs.getInt("
       consultationid"));
   consult.setConsultationreason(rs.
       getString("consultationreason"));
   consult.setConsultfrom(rs.getString("
       consultfrom"));
   consult.setConsultto(rs.getString("
       consultto"));
   consult.setConsultfindings(rs.
       getString("consultfindings"));
   consult.setConsultclinician(rs.
       getString("consultclinician"));
   consult.setConsultcliniciannature(rs.
       getString("consultcliniciannature
       "));
   consult.setVersion(rs.getInt("version"
       ));
   consult.setUpdated_by(rs.getString("
       updated_by"));
   consult.setUpdated_date(rs.getString("
       updated_date"));
   consult.setUpdated_time(rs.getString("
       updated_time"));
   consult.setApproved(rs.getString("
       approved"));
   consult.setApproved_by(rs.getString("
       approved_by"));
   consult.setApproved_date(rs.getString(
       "approved_date"));
   consult.setApproved_time(rs.getString(
       "approved_time"));

   return consult;
  }
}
```

Listing 16: ConsultationReferrals

```
package com.dentist.version.three.form;

public class ConsultationsReferrals {

 private int patientid;
 private int consultationid;
 private String consultationreason;
 private String consultfrom;
 private String consultto;
 private String consultfindings;
 private String consultclinician;
 private String consultcliniciannature;
 private int version;
 private String updated_by;
 private String updated_date;
 private String updated_time;
 private String approved;
 private String approved_by;
 private String approved_date;
 private String approved_time;
 public void setPatientid(int patientid)
     {
  this.patientid = patientid;
 }
 public int getPatientid() {
  return patientid;
 }
 public void setConsultationid(int
     consultationid) {
  this.consultationid = consultationid;
 }
 public int getConsultationid() {
  return consultationid;
 }
 public void setConsultationreason(String
      consultationreason) {
  this.consultationreason =
      consultationreason;
 }
 public String getConsultationreason() {
  return consultationreason;
 }
 public void setConsultfrom(String
     consultfrom) {
  this.consultfrom = consultfrom;
 }
```

```
 public String getConsultfrom() {
  return consultfrom;
 }
 public void setConsultto(String
      consultto) {
  this.consultto = consultto;
 }
 public String getConsultto() {
  return consultto;
 }
 public void setConsultfindings(String
      consultfindings) {
  this.consultfindings = consultfindings;
 }
 public String getConsultfindings() {
  return consultfindings;
 }
 public void setConsultclinician(String
      consultclinician) {
  this.consultclinician =
       consultclinician;
 }
 public String getConsultclinician() {
  return consultclinician;
 }
 public void setConsultcliniciannature(
      String consultcliniciannature) {
  this.consultcliniciannature =
       consultcliniciannature;
 }
 public String getConsultcliniciannature
      () {
  return consultcliniciannature;
 }
 public void setVersion(int version) {
  this.version = version;
 }
 public int getVersion() {
  return version;
 }
 public void setUpdated_by(String
      updated_by) {
  this.updated_by = updated_by;
 }
 public String getUpdated_by() {
  return updated_by;
 }
 public void setUpdated_date(String
      updated_date) {
  this.updated_date = updated_date;
 }
 public String getUpdated_date() {
  return updated_date;
 }
 public void setUpdated_time(String
      updated_time) {
  this.updated_time = updated_time;
 }
 public String getUpdated_time() {
  return updated_time;
 }
 public void setApproved(String approved)
       {
  this.approved = approved;
 }
 public String getApproved() {
  return approved;
 }
 public void setApproved_by(String
      approved_by) {
  this.approved_by = approved_by;
 }
 public String getApproved_by() {
  return approved_by;
 }
 public void setApproved_date(String
      approved_date) {
  this.approved_date = approved_date;
 }
 public String getApproved_date() {
  return approved_date;
 }
 public void setApproved_time(String
      approved_time) {
  this.approved_time = approved_time;
 }
 public String getApproved_time() {
  return approved_time;
 }

}
```

Listing 17: DeleteFunction

```
package com.dentist.version.three.form;

public class DeleteFunction {

 int[] deleteObject;
 private int key_id;

 public void setDeleteObject(int[]
      deleteObject){
  this.deleteObject=deleteObject;
 }
 public int[] getDeleteObject(){
  return deleteObject;
 }
 public void setKey_id(int key_id) {
```

```
 this.key_id = key_id;
}
public int getKey_id() {
 return key_id;
}
}
```

Listing 18: DentalChart

```
package com.dentist.version.three.form;

import java.util.ArrayList;

public class DentalChart {

 private int dental_chart_id;
 private int patient_id;
 private int clinician_id;
 private Integer[] caries;
 private Integer[] recurrentcaries;
 private Integer[] restoration;
 private String complete_denture;
 private String single_denture;
 private Integer[]
     removable_partial_denture;
 private Integer[] extrusion;
 private Integer[] intrusion;
 private Integer[] mesial_rotation;
 private Integer[] distal_rotation;
 private Integer[] rotation;
 private Integer[] postcore_crown;
 private Integer[] rootcanal_treatment;
 private Integer[] pitfissure_sealants;
 private Integer[] extracted;
 private Integer[] missing;
 private Integer[] unerupted;
 private Integer[] impacted;
 private Integer[] porcelain_crown;
 private Integer[] acrylic_crown;
 private Integer[] metal_crown;
 private Integer[] porcelain_infused;
 private Integer[] fixed_bridge;
 private int version;
 private String updated_by;
 private String updated_date;
 private String updated_time;
 private String is_current;
 private String approved;
 private String approved_by;
 private String approved_date;
 private String approvated_time;


 //String private
 private String[] caries_string;
 private String[] recurrentcaries_string;
 private String[] restoration_string;
 private String[]
     removable_partial_denture_string;
 private String[] extrusion_string;
 private String[] intrusion_string;
 private String[] mesial_rotation_string;
 private String[] distal_rotation_string;
 private String[] rotation_string;
 private String[] postcore_crown_string;
 private String[]
     rootcanal_treatment_string;
 private String[]
     pitfissure_sealants_string;
 private String[] extracted_string;
 private String[] missing_string;
 private String[] unerupted_string;
 private String[] impacted_string;
 private String[] porcelain_crown_string;
 private String[] acrylic_crown_string;
 private String[] metal_crown_string;
 private String[]
     porcelain_infused_string;
 private String[] fixed_bridge_string;

 private Integer[] splitStringToInteger(
     String[] temp){
  Integer[] result= new Integer[temp.
      length];
  String tempResult= "";
  for(int i=0; i<temp.length;i++){
   tempResult=temp[i];
   if(temp[i].indexOf("{")!=-1 || temp[i
       ].indexOf("}")!=-1){
    if(temp[i].indexOf("{")!=-1)
     tempResult=tempResult.replace("{", "
         ").trim();
     if(temp[i].indexOf("}")!=-1)
     tempResult=tempResult.replace("}", "
         ").trim();

    System.out.println("Check:_"+
        tempResult);
   }

   result[i]=Integer.parseInt(tempResult)
       ;
  }
  return result;
 }

 private String[] splitStringToSpecific(
     String[] temp){
  ArrayList<String> tempResult= new
```

```
        ArrayList<String>();
  String tempAr= "";
  String[] result=null;
  if(temp!=null){
  for(int i=0; i<temp.length;i++){
   if(temp[i].length()>2)
   tempResult.add(temp[i]);
  }
  result= new String[tempResult.size()];
  for(int j=0;j<tempResult.size();j++){
   tempAr=tempResult.get(j);

   if(tempResult.get(j).indexOf("{")!=-1
        || tempResult.get(j).indexOf("}")
        !=-1){
    if(tempResult.get(j).indexOf("{")
         !=-1)
     tempAr=tempAr.replace("{", "").trim
          ();
     if(tempResult.get(j).indexOf("}")
          !=-1)
      tempAr=tempAr.replace("}", "").trim
           ();

   }
   System.out.println("Check:_"+ tempAr +
        "_noms");
   result[j]=tempAr;
  }
  }
  else{
   result=new String[1];
   result[0]="-1";
  }

  return result;
 }

 public void setValuesNeeded(String name,
       String value){
  String[] partial=null;
  if(name.equals("caries")){
   partial=value.split(",");
   this.setCaries(partial);
  }
  else if(name.equals("recurrent_caries")
       ){
   partial=value.split(",");
   this.setRecurrentcaries(partial);
  }
  else if(name.equals("restoration")){
   partial=value.split(",");
   this.setRestoration(partial);
  }
  else if(name.equals("
       removable_partial_denture")){
   partial=value.split(",");
   this.setRemovable_partial_denture(
        partial);
  }
  else if(name.equals("extrusion")){
   partial=value.split(",");
   this.setExtrusion(partial);
  }
  else if(name.equals("intrusion")){
   partial=value.split(",");
   this.setIntrusion(partial);
  }
  else if(name.equals("mesial_rotation"))
       {
   partial=value.split(",");
   this.setMesial_rotation(partial);
  }
  else if(name.equals("distal_rotation"))
       {
   partial=value.split(",");
   this.setDistal_rotation(partial);
  }
  else if(name.equals("rotation")){
   partial=value.split(",");
   this.setRotation(partial);
  }
  else if(name.equals("postcore_crown")){
   partial=value.split(",");
   this.setPostcore_crown(partial);
  }
  else if(name.equals("
       rootcanal_treatment")){
   partial=value.split(",");
   this.setRootcanal_treatment(partial);
  }
  else if(name.equals("
       pitfissure_sealants")){
   partial=value.split(",");
   this.setPitfissure_sealants(partial);
  }
  else if(name.equals("extracted")){
   partial=value.split(",");
   this.setExtracted(partial);
  }
  else if(name.equals("missing")){
   partial=value.split(",");
   this.setMissing(partial);
  }
  else if(name.equals("unerupted")){
   partial=value.split(",");
   this.setUnerupted(partial);
  }
  else if(name.equals("impacted")){
   partial=value.split(",");
   this.setImpacted(partial);
  }
```

```
 else if(name.equals("porcelain_crown"))
     {
  partial=value.split(",");
  this.setPorcelain_crown(partial);
 }
 else if(name.equals("acrylic_crown")){
  partial=value.split(",");
  this.setAcrylic_crown(partial);
 }
 else if(name.equals("metal_crown")){
  partial=value.split(",");
  this.setMetal_crown(partial);
 }
 else if(name.equals("porcelain_infused"
     )){
  partial=value.split(",");
  this.setPorcelain_infused(partial);
 }
 else if(name.equals("fixed_bridge")){
  partial=value.split(",");
  this.setFixed_bridge(partial);
 }
}
```

```
public void setDental_chart_id(int
    dental_chart_id) {
 this.dental_chart_id = dental_chart_id;
}
public int getDental_chart_id() {
 return dental_chart_id;
}
public void setPatient_id(int patient_id
    ) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setClinician_id(int
    clinician_id) {
 this.clinician_id = clinician_id;
}
public int getClinician_id() {
 return clinician_id;
}
public void setCaries(String[] caries) {
 Integer[] cariesResult=null;
if (caries != null) {
 cariesResult=splitStringToInteger(
     caries);
}
 this.setCaries_string(
     splitStringToSpecific(caries));
 this.caries = cariesResult;
}
```

```
public Integer[] getCaries() {
 return caries;
}

public void setRecurrentcaries(String[]
    recurrentcaries) {
 Integer[] recurrentResult=null;
 if (recurrentcaries != null) {

 recurrentResult=splitStringToInteger(
     recurrentcaries);

 }
 this.setRecurrentcaries_string(
     splitStringToSpecific(
     recurrentcaries));
 this.recurrentcaries = recurrentResult;
}
public Integer[] getRecurrentcaries() {
 return recurrentcaries;
}
```

```
public void setRestoration(String[]
    restoration) {
 Integer[] restorationtResult=null;
 if (restoration != null) {

 restorationtResult=splitStringToInteger
     (restoration);

 }
 this.setRestoration_string(
     splitStringToSpecific(restoration)
     );
 this.restoration = restorationtResult;
}


public Integer[] getRestoration() {
 return restoration;
}


public void setRemovable_partial_denture
    (String[] removable_partial_denture
    ) {
 Integer[]
     removable_partial_dentureResult=
     null;
 if (removable_partial_denture != null)
     {
 removable_partial_dentureResult=
     splitStringToInteger(
```

```
        removable_partial_denture);
    }
    this.
        setRemovable_partial_denture_string
        (splitStringToSpecific(
        removable_partial_denture));
    this.removable_partial_denture =
        removable_partial_dentureResult;
}


public Integer[]
    getRemovable_partial_denture() {
    return removable_partial_denture;
}


public void setExtrusion(String[]
    extrusion) {
    Integer[] extrusionResult=null;
    if (extrusion != null) {
    extrusionResult=splitStringToInteger(
        extrusion);
    }
    this.setExtrusion_string(
        splitStringToSpecific(extrusion));
    this.extrusion = extrusionResult;
}


public Integer[] getExtrusion() {
    return extrusion;
}


public void setIntrusion(String[]
    intrusion) {
    Integer[] intrusionResult=null;
    if (intrusion != null) {
    intrusionResult=splitStringToInteger(
        intrusion);
    }
    this.setIntrusion_string(
        splitStringToSpecific(intrusion));
    this.intrusion = intrusionResult;
}


public Integer[] getIntrusion() {
    return intrusion;
}


public void setMesial_rotation(String[]
    mesial_rotation) {
    Integer[] mesial_rotationResult=null;
    if (mesial_rotation != null) {
    mesial_rotationResult=
        splitStringToInteger(
        mesial_rotation);
    }
    this.setMesial_rotation_string(
        splitStringToSpecific(
        mesial_rotation));
    this.mesial_rotation =
        mesial_rotationResult;
}


public Integer[] getMesial_rotation() {
    return mesial_rotation;
}


public void setDistal_rotation(String[]
    distal_rotation) {
    Integer[] distal_rotationResult=null;
    if (distal_rotation != null) {
    distal_rotationResult=
        splitStringToInteger(
        distal_rotation);
    }
    this.setDistal_rotation_string(
        splitStringToSpecific(
        distal_rotation));
    this.distal_rotation =
        distal_rotationResult;
}


public Integer[] getDistal_rotation() {
    return distal_rotation;
}


public void setRotation(String[]
    rotation) {

    Integer[] rotationResult=null;
    if (rotation != null) {
    rotationResult=splitStringToInteger(
        rotation);
    }
    this.setRotation_string(
        splitStringToSpecific(rotation));
    this.rotation = rotationResult;
}


public Integer[] getRotation() {
    return rotation;
}
```

```
public void setPostcore_crown(String[]
    postcore_crown) {
 Integer[] postcore_crownResult=null;

 if (postcore_crown != null ) {
 postcore_crownResult=
     splitStringToInteger(
     postcore_crown);
 }
 this.setPostcore_crown_string(
     splitStringToSpecific(
     postcore_crown));
 this.postcore_crown =
     postcore_crownResult;
}


public Integer[] getPostcore_crown() {
 return postcore_crown;
}


public void setRootcanal_treatment(
    String[] rootcanal_treatment) {
 Integer[] rootcanal_treatmentResult=
     null;

 if (rootcanal_treatment != null) {
 rootcanal_treatmentResult=
     splitStringToInteger(
     rootcanal_treatment);
 }
 this.setRootcanal_treatment_string(
     splitStringToSpecific(
     rootcanal_treatment));
 this.rootcanal_treatment =
     rootcanal_treatmentResult;
}


public Integer[] getRootcanal_treatment
    () {
 return rootcanal_treatment;
}


public void setPitfissure_sealants(
    String[] pitfissure_sealants) {
 Integer[] pitfissure_sealantsResult=
     null;

 if (pitfissure_sealants != null ) {
 pitfissure_sealantsResult=
     splitStringToInteger(
     pitfissure_sealants);
 }
 this.setPitfissure_sealants_string(
     splitStringToSpecific(
     pitfissure_sealants));
 this.pitfissure_sealants =
     pitfissure_sealantsResult;
}


public Integer[] getPitfissure_sealants
    () {
 return pitfissure_sealants;
}


public void setExtracted(String[]
    extracted) {
 Integer[] extractedResult=null;

 if (extracted != null) {
 extractedResult=splitStringToInteger(
     extracted);
 }
 this.setExtracted_string(
     splitStringToSpecific(extracted));
 this.extracted = extractedResult;
}


public Integer[] getExtracted() {
 return extracted;
}


public void setMissing(String[] missing)
     {
 Integer[] missingResult=null;

 if (missing != null ) {
 missingResult=splitStringToInteger(
     missing);
 }
 this.setMissing_string(
     splitStringToSpecific(missing));
 this.missing = missingResult;
}


public Integer[] getMissing() {
 return missing;
}


public void setUnerupted(String[]
    unerupted) {
```

```
 Integer[] uneruptedResult=null;


 if (unerupted != null ) {
 uneruptedResult=splitStringToInteger(
      unerupted);
 }
 this.setUnerupted_string(
      splitStringToSpecific(unerupted));
 this.unerupted = uneruptedResult;
}


public Integer[] getUnerupted() {
 return unerupted;
}


public void setImpacted(String[]
     impacted) {
 Integer[] impactedResult=null;

 if (impacted != null ) {
 impactedResult=splitStringToInteger(
      impacted);
 }

 this.setImpacted_string(
      splitStringToSpecific(impacted));
 this.impacted = impactedResult;
}


public Integer[] getImpacted() {
 return impacted;
}


public void setPorcelain_crown(String[]
     porcelain_crown) {
 Integer[] porcelain_crownResult=null;

 if (porcelain_crown != null) {
 porcelain_crownResult=
      splitStringToInteger(
      porcelain_crown);
 }
 this.setPorcelain_crown_string(
      splitStringToSpecific(
      porcelain_crown));
 this.porcelain_crown =
      porcelain_crownResult;
}


public Integer[] getPorcelain_crown() {
 return porcelain_crown;
}


public void setVersion(int version) {
 this.version = version;
}


public int getVersion() {
 return version;
}


public void setUpdated_by(String
     updated_by) {
 this.updated_by = updated_by;
}


public String getUpdated_by() {
 return updated_by;
}


public void setUpdated_time(String
     updated_time) {
 this.updated_time = updated_time;
}


public String getUpdated_time() {
 return updated_time;
}


public void setIs_current(String
     is_current) {
 this.is_current = is_current;
}


public String getIs_current() {
 return is_current;
}


public void setApproved(String approved)
      {
 this.approved = approved;
}


public String getApproved() {
 return approved;
}
```

```
public void setApproved_by(String
    approved_by) {
 this.approved_by = approved_by;
}


public String getApproved_by() {
 return approved_by;
}


public void setApproved_date(String
    approved_date) {
 this.approved_date = approved_date;
}


public String getApproved_date() {
 return approved_date;
}


public void setApprovated_time(String
    approvated_time) {
 this.approvated_time = approvated_time;
}


public String getApprovated_time() {
 return approvated_time;
}


public void setAcrylic_crown(String[]
    acrylic_crown) {
 Integer[] acrylic_crownResult=null;

 if (acrylic_crown != null) {
 acrylic_crownResult=
     splitStringToInteger(acrylic_crown
     );
 }
 this.setAcrylic_crown_string(
     splitStringToSpecific(
     acrylic_crown));
 this.acrylic_crown =
     acrylic_crownResult;
}


public Integer[] getAcrylic_crown() {
 return acrylic_crown;
}


public void setMetal_crown(String[]
    metal_crown) {
 Integer[] metal_crownResult=null;

 if (metal_crown != null ) {
 metal_crownResult=splitStringToInteger(
     metal_crown);
 }
 this.setMetal_crown_string(
     splitStringToSpecific(metal_crown)
     );
 this.metal_crown = metal_crownResult;
}


public Integer[] getMetal_crown() {
 return metal_crown;
}


public void setPorcelain_infused(String
    [] porcelain_infused) {
 Integer[] porcelain_infusedResult=null;

 if (porcelain_infused != null ) {
 porcelain_infusedResult=
     splitStringToInteger(
     porcelain_infused);
 }

 this.setPorcelain_infused_string(
     splitStringToSpecific(
     porcelain_infused));
 this.porcelain_infused =
     porcelain_infusedResult;
}


public Integer[] getPorcelain_infused()
    {
 return porcelain_infused;
}


public void setFixed_bridge(String[]
    fixed_bridge) {
 Integer[] fixed_bridgeResult=null;

 if (fixed_bridge != null) {
 fixed_bridgeResult=splitStringToInteger
     (fixed_bridge);
 }

 this.setFixed_bridge_string(
     splitStringToSpecific(fixed_bridge
```

```
      ));
 this.fixed_bridge = fixed_bridgeResult;
}


public Integer[] getFixed_bridge() {
 return fixed_bridge;
}


public void setUpdated_date(String
     updated_date) {
 this.updated_date = updated_date;
}


public String getUpdated_date() {
 return updated_date;
}

public void setRecurrentcaries_string(
     String[] recurrentcaries_string) {
 this.recurrentcaries_string =
      recurrentcaries_string;
}

public String[]
     getRecurrentcaries_string() {
 return recurrentcaries_string;
}

public void setRestoration_string(String
     [] restoration_string) {
 this.restoration_string =
      restoration_string;
}

public String[] getRestoration_string()
     {
 return restoration_string;
}

public void
     setRemovable_partial_denture_string
     (
  String[]
       removable_partial_denture_string)
        {
 this.removable_partial_denture_string =
       removable_partial_denture_string;
}

public String[]
     getRemovable_partial_denture_string
     () {
 return removable_partial_denture_string
      ;
}

public void setExtrusion_string(String[]
      extrusion_string) {
 this.extrusion_string =
      extrusion_string;
}

public String[] getExtrusion_string() {
 return extrusion_string;
}

public void setIntrusion_string(String[]
      intrusion_string) {
 this.intrusion_string =
      intrusion_string;
}

public String[] getIntrusion_string() {
 return intrusion_string;
}

public void setMesial_rotation_string(
     String[] mesial_rotation_string) {
 this.mesial_rotation_string =
      mesial_rotation_string;
}

public String[]
     getMesial_rotation_string() {
 return mesial_rotation_string;
}

public void setDistal_rotation_string(
     String[] distal_rotation_string) {
 this.distal_rotation_string =
      distal_rotation_string;
}

public String[]
     getDistal_rotation_string() {
 return distal_rotation_string;
}

public void setRotation_string(String[]
     rotation_string) {
 this.rotation_string = rotation_string;
}

public String[] getRotation_string() {
 return rotation_string;
}

public void setPostcore_crown_string(
     String[] postcore_crown_string) {
```

```
 this.postcore_crown_string =
      postcore_crown_string;
}

public String[] getPostcore_crown_string
    () {
 return postcore_crown_string;
}

public void
    setRootcanal_treatment_string(
  String[] rootcanal_treatment_string) {
 this.rootcanal_treatment_string =
      rootcanal_treatment_string;
}

public String[]
    getRootcanal_treatment_string() {
 return rootcanal_treatment_string;
}

public void
    setPitfissure_sealants_string(
  String[] pitfissure_sealants_string) {
 this.pitfissure_sealants_string =
      pitfissure_sealants_string;
}

public String[]
    getPitfissure_sealants_string() {
 return pitfissure_sealants_string;
}

public void setExtracted_string(String[]
      extracted_string) {
 this.extracted_string =
      extracted_string;
}

public String[] getExtracted_string() {
 return extracted_string;
}

public void setMissing_string(String[]
    missing_string) {
 this.missing_string = missing_string;
}
public String[] getMissing_string() {
 return missing_string;
}

public void setUnerupted_string(String[]
      unerupted_string) {
 this.unerupted_string =
      unerupted_string;
}

public String[] getUnerupted_string() {
 return unerupted_string;
}

public void setImpacted_string(String[]
    impacted_string) {
 this.impacted_string = impacted_string;
}

public String[] getImpacted_string() {
 return impacted_string;
}

public void setPorcelain_crown_string(
    String[] porcelain_crown_string) {
 this.porcelain_crown_string =
      porcelain_crown_string;
}

public String[]
    getPorcelain_crown_string() {
 return porcelain_crown_string;
}

public void setAcrylic_crown_string(
    String[] acrylic_crown_string) {
 this.acrylic_crown_string =
      acrylic_crown_string;
}

public String[] getAcrylic_crown_string
    () {
 return acrylic_crown_string;
}

public void setMetal_crown_string(String
    [] metal_crown_string) {
 this.metal_crown_string =
      metal_crown_string;
}

public String[] getMetal_crown_string()
    {
 return metal_crown_string;
}

public void setPorcelain_infused_string(
    String[] porcelain_infused_string)
    {
 this.porcelain_infused_string =
      porcelain_infused_string;
}

public String[]
    getPorcelain_infused_string() {
```

```
 return porcelain_infused_string;
}

public void setFixed_bridge_string(
    String[] fixed_bridge_string) {
 this.fixed_bridge_string =
     fixed_bridge_string;
}

public String[] getFixed_bridge_string()
    {
 return fixed_bridge_string;
}

public void setCaries_string(String[]
    caries_string) {
 this.caries_string = caries_string;
}

public String[] getCaries_string() {
 return caries_string;
}

public void setComplete_denture(String
    complete_denture) {
 this.complete_denture =
     complete_denture;
}

public String getComplete_denture() {
 return complete_denture;
}

public void setSingle_denture(String
    single_denture) {
 this.single_denture = single_denture;
}

public String getSingle_denture() {
 return single_denture;
}


}
```

Listing 19: GeneratePassword

```
package com.dentist.version.three.mapper;
/*
JSPWiki - a JSP-based WikiWiki clone.

Licensed to the Apache Software
    Foundation (ASF) under one
or more contributor license agreements.
    See the NOTICE file
distributed with this work for additional
    information
regarding copyright ownership.  The ASF
    licenses this file
to you under the Apache License, Version
    2.0 (the
"License"); you may not use this file
    except in compliance
with the License.  You may obtain a copy
    of the License at

   http://www.apache.org/licenses/LICENSE
       -2.0

Unless required by applicable law or
    agreed to in writing,
software distributed under the License is
    distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR
    CONDITIONS OF ANY
KIND, either express or implied.  See the
    License for the
specific language governing permissions
    and limitations
under the License.
*/

import java.security.SecureRandom;
import java.util.Random;

public class GeneratePassword
{

private static final Random RANDOM = new
    SecureRandom();
/** Length of password. @see #
    generateRandomPassword() */
public static final int PASSWORD_LENGTH =
    8;
/**
* Generate a random String suitable for
    use as a temporary password.
*
* @return String suitable for use as a
    temporary password
* @since 2.4
*/
public String generateRandomPassword()
{
  // Pick from some letters that won't be
      easily mistaken for each
  // other. So, for example, omit o O and
      0, 1 l and L.
  String letters = "
```

```
        abcdefghjkmnpqrstuvwxyzABCDEFGHJKMNPQRSTUVWXYZ23456789
        ";

    String pw = "";
    for (int i=0; i<PASSWORD_LENGTH; i++)
    {
        int index = (int)(RANDOM.nextDouble()*letters.length());
        pw += letters.substring(index, index+1);
    }
    return pw;
}
}
```

Listing 20: LoginController

```
package com.dentist.version.three.controller;

import java.io.File;
import java.io.FileWriter;
import java.io.InputStream;
import java.io.PrintWriter;
import java.net.URL;


import java.util.ArrayList;

import javax.mail.internet.MimeMessage;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;


import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.dentist.version.three.db.AdminSP;
import com.dentist.version.three.db.LoginSP;
import com.dentist.version.three.db.RoleSP;
import com.dentist.version.three.db.SectionSP;
import com.dentist.version.three.db.UserSP;
import com.dentist.version.three.form.Configuration;
import com.dentist.version.three.form.EmailAddress;
import com.dentist.version.three.form.Login;
import com.dentist.version.three.form.PrivilegeLink;
import com.dentist.version.three.form.Role;
import com.dentist.version.three.form.TargetExtract;
import com.dentist.version.three.form.UPCDIDGenerator;
import com.dentist.version.three.form.User;
import com.dentist.version.three.mapper.MailUtil;
import com.dentist.version.three.xacml.PolicyReader;
import com.dentist.version.three.xacml.XACMLExtract;
import com.dentist.version.three.xacml.XMLCreate;



@Controller
public class LoginController{


 private ArrayList<String> tempUsernames;
 ArrayList<PrivilegeLink> privilege= new ArrayList<PrivilegeLink>();


 @RequestMapping(value = "/loginForm.html
```

```
    ", method = RequestMethod.GET)
public String loginForm(Model model,
    HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

  if(privilege!=null && privilege.size()
      >0)
   privilege.clear();

        Login login= new Login();

        /*
         * UPCDID CHECK
         *
         */
        UPCDIDGenerator idgenerator=
            new UPCDIDGenerator();
        String id= idgenerator.generate
            ("DentIStAdmin");
        System.out.println("UPCDID: "+
            id);
        /*
         *
         * END
         *
         */
 model.addAttribute("login",login);
 return "loginForm";

}


@RequestMapping(value = "/forgotPassword
    .html", method = RequestMethod.GET)
public String forgotPassword(Model model
    , HttpServletRequest request)
    throws Exception {
 User user= new User();
 model.addAttribute("user", user);
 return "forgotPasswordUN";

}

@RequestMapping(value = "/
    forgotPasswordUN.html", method =
    RequestMethod.GET)
public String forgotPasswordUN(
    HttpServletRequest request,
        HttpServletResponse response,
            Model model) throws
            Exception {
 String errorMsg="ERROR: No such user
     exists. Please try again.";
 LoginSP loginSP= new LoginSP();
 User user= new User();
 String username= request.getParameter("
     username");
 int user_id= loginSP.getUserID(username
     );
 if(user_id==0){
 model.addAttribute("errorMsg",errorMsg)
     ;

 }
 user= loginSP.getUser(user_id);
 String applicable="YES";
 user.setSecret_answer("");
 model.addAttribute("user", user);
 model.addAttribute("applicable",
     applicable);
 return "forgotPasswordUN";

}

@RequestMapping(value = "/
    forgotPasswordRetrieve.html",
    method = RequestMethod.POST)
public String retrievalPassword(
    HttpServletRequest request,Model
    model,
  @ModelAttribute("user") User user,
          HttpSession session) throws
              Exception {
 session.setAttribute("PASSWORDFIND", "
     PASSWORDFIND");
 ArrayList<Configuration> allConfigList=
      new ArrayList<Configuration>();
 EmailAddress emailAddress= new
     EmailAddress();
 if(request.getAttribute("confirm")!=
     null)
  request.removeAttribute("confirm");
 LoginSP loginSP= new LoginSP();
 int user_id=0;
 String newPassword= loginSP.
     resetPassword(user.getUser_id(),
     user.getSecret_question(), user.
     getSecret_answer());
 System.out.println(newPassword);
 user_id= loginSP.loginUser(user.
     getUsername(), newPassword);
 System.out.println(user_id);
 if(user_id==0){
  request.setAttribute("confirm", "NO");
 }
 else{
  AdminSP adminSP= new AdminSP();
  allConfigList= adminSP.
      configListChange();

  for(Configuration config:
```

```
        allConfigList){
    if(config.getName().equals("smtp_from
        ")){
     emailAddress.setSmtp_from(config.
         getName_value());
    }
    if(config.getName().equals("smtp_subj
        ")){
     emailAddress.setSmtp_subj(config.
         getName_value());
    }
    if(config.getName().equals("
        smtp_message")){
     emailAddress.setSmtp_message(config.
         getName_value());
    }
   }
  String subText= emailAddress.
      getSmtp_message()+ newPassword;
     //    ApplicationContext context =
         new
         ClassPathXmlApplicationContext
         ("spring-mail.xml");
        String[] recipientList={user.
            getEmail()};
        MailUtil smtpMailSender = new
            MailUtil();
       // smtpMailSender.
            setSmtp_auth_user(
            emailAddress.getSmtp_from());
        boolean checkLang= smtpMailSender
            .postMail(recipientList,
            emailAddress.getSmtp_subj(),
             subText,emailAddress.
            getSmtp_from());
        if(checkLang)
         System.out.println("MAILED_TO_
             YOU");
       // MailService mailService = (
            MailService) context.getBean
            ("mailService");


        //mailService.sendMail("angela.
            jarabelo@gmail.com", user.
            getEmail(), "Reset Password
            ",subText );


      //   mailService.sendAlertMail("
           Exception occurred");
  }
  model.addAttribute("user", user);

  return "success";

}
```

```
@RequestMapping(value = "/index.html",
     method = RequestMethod.GET)
public String indexForm(Model model,
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 session=request.getSession(false);

  if(privilege!=null && privilege.size()
      >0)
   privilege.clear();

 String[] currentRoleList= (String[])
     session.getAttribute("
     currentRoleList");

 boolean studentCheck=false;
 boolean facultyCheck=false;
 boolean adminCheck=false;
 boolean workflowCheck=false;
 for(int i=0; i<currentRoleList.length;i
     ++){
 // privilegeCheck= adminSP.
     checkPrivilege(boom[i], "users", "
     insert");
  if(currentRoleList[i].indexOf("System"
      )!=-1){

   PrivilegeLink userPriv= new
       PrivilegeLink();
   userPriv.setName("userURL");
   userPriv.setValue("Manage_Users");
   userPriv.setUrl(request.
       getContextPath()+"/forms/dentist
       /simpleForm/userForm.html");

   PrivilegeLink rolePriv= new
       PrivilegeLink();
   rolePriv.setName("roleURL");
   rolePriv.setValue("Manage_roles");
   rolePriv.setUrl(request.
       getContextPath()+"/forms/dentist
       /simpleForm/roleForm.html");

   PrivilegeLink sectionPriv= new
       PrivilegeLink();
   sectionPriv.setName("sectionURL");
   sectionPriv.setValue("Manage_sections
       ");
   sectionPriv.setUrl(request.
       getContextPath()+"/forms/dentist
       /simpleForm/sectionForm.html");


   privilege.add(userPriv);
   privilege.add(rolePriv);
```

```
privilege.add(sectionPriv);

if(!adminCheck && !facultyCheck && !
    workflowCheck){

PrivilegeLink statPriv= new
    PrivilegeLink();
statPriv.setName("viewStatURL");
statPriv.setValue("View Statistics");
statPriv.setUrl(request.
    getContextPath()+"/forms/dentist
    /search/statisticsform.html");
privilege.add(statPriv);
 }



adminCheck=true;
}
 //for faculty
 if(currentRoleList[i].toString().
     toLowerCase().indexOf("faculty")
     !=-1){
  if(!facultyCheck){
  PrivilegeLink sectionPriv= new
      PrivilegeLink();
  sectionPriv.setName("sectionURL");
  sectionPriv.setValue("View Users in
      section");
  sectionPriv.setUrl(request.
      getContextPath()+"/forms/
      dentist/simpleForm/
      viewFacultySection.html");
  privilege.add(sectionPriv);

   PrivilegeLink queryPriv= new
       PrivilegeLink();
   queryPriv.setName("viewQueryURL");
   queryPriv.setValue("Query Patients"
       );
   queryPriv.setUrl(request.
       getContextPath()+"/forms/
       dentist/search/
       viewQueryPatients.html");
   privilege.add(queryPriv);

  if(!adminCheck && !facultyCheck){
  PrivilegeLink statPriv= new
      PrivilegeLink();
  statPriv.setName("viewStatURL");
  statPriv.setValue("View Statistics")
      ;
  statPriv.setUrl(request.
      getContextPath()+"/forms/
      dentist/search/statisticsform.
      html");
  privilege.add(statPriv);
  }

  facultyCheck=true;
  }
 }
 if(currentRoleList[i].toString().
     toLowerCase().indexOf("student")
     !=-1){
  if(!studentCheck){

   PrivilegeLink queryPriv= new
       PrivilegeLink();
   queryPriv.setName("viewQueryURL");
   queryPriv.setValue("Query Patients"
       );
   queryPriv.setUrl(request.
       getContextPath()+"/forms/
       dentist/search/
       viewQueryPatients.html");
   privilege.add(queryPriv);

   studentCheck=true;

  }
 }
 if(currentRoleList[i].toString().
     toLowerCase().indexOf("workflow"
     )!=-1){
  if(!workflowCheck){

 if(!adminCheck && !facultyCheck && !
     workflowCheck){
  PrivilegeLink statPriv= new
      PrivilegeLink();
  statPriv.setName("viewStatURL");
  statPriv.setValue("View Statistics")
      ;
  statPriv.setUrl(request.
      getContextPath()+"/forms/
      dentist/search/statisticsform.
      html");
  privilege.add(statPriv);
  }

  workflowCheck=true;
  }
 }

}



model.addAttribute("privilege",
    privilege);
```

```
 return "index";

}

@RequestMapping(value = "/loginOutput.
    html", method = RequestMethod.POST)
public String loginProcess(
    HttpServletRequest request,Login
    login, Model model,
  @ModelAttribute("loginuser") Login
      loginuser,
          HttpSession session){

 if(session!=null) {
  session.invalidate();
  }
  // Create a new session for the user.
  session= request.getSession(true);
  if(privilege!=null &&privilege.size()
      >0)
  privilege.clear();

  String userName = (String) session.
      getAttribute("sessionUser");
  User user = new User();

 String[] currentRoleList;
 String[] currentDatabaseList;
 int user_id=0;
 int role_id=0;
 Role role=new Role();
 String role_name="";
 //UserSP userSP= new UserSP();
// RoleSP roleSP= new RoleSP();
 //SectionSP sectionSP= new SectionSP();
 AdminSP adminSP= new AdminSP();

 String database_role_name="";
 String error="";
 //multi role
  ArrayList<Integer> roles_id= new
      ArrayList<Integer>();
  int countPriv=0;
  int countString=0;
  String tempDatabaseRole="";
  LoginSP loginSP= new LoginSP();
 try {
  user_id= loginSP.loginUser(login.
      getUsername(), login.getPassword
      ());

  roles_id= loginSP.getUserRole(user_id)
      ;
  currentRoleList= new String[roles_id.
      size()];
  currentDatabaseList= new String[
      roles_id.size()];

 for(int temp_role_id: roles_id){
  int tempCount=0;
  tempDatabaseRole=loginSP.
      getDatabaseRole(temp_role_id);
  System.out.println(tempDatabaseRole);
  currentDatabaseList[countString]=
      tempDatabaseRole;
  tempCount=adminSP.allTablesList(
      tempDatabaseRole);
  if(countPriv<tempCount){
   countPriv=tempCount;
   role_id=temp_role_id;
   //System.out.println("temp: "+
       role_id);

  }
  Role tempRole= new Role();
  tempRole=loginSP.getRole(temp_role_id
      );
  currentRoleList[countString]=
      tempRole.getRole_name();
  System.out.println("CHECK LANG::: "+
      currentRoleList[countString]);
  countString++;
 }

 user=loginSP.getUser(user_id);


 System.out.println("new Role: "+
     role_id);
 role= loginSP.getRole(role_id);
 role_name=role.getRole_name();

if(user_id==0){

 model.addAttribute("error","ERROR:
     Username and Password did not
     match. Please Try Again.");
 return "loginForm";
}
else{
 database_role_name= loginSP.
     getDatabaseRole(role_id);

}


 loginSP.setDatabase_username(
     database_role_name);
 System.out.println("ENTERED AS::: "+
```

```
        database_role_name);
loginuser.setUsername(login.
        getUsername());
loginuser.setPassword(login.
        getPassword());
session.setAttribute("sessionUser",
        loginuser.getUsername());
session.setAttribute("sessionUserId",
        user_id);
session.setAttribute("sessionRole",
        role_name);
session.setAttribute("sessionUserRole"
        , database_role_name);
session.setAttribute("currentRoleList"
        , currentRoleList);
session.setAttribute("
        currentDatabaseList",
        currentDatabaseList);
session.setAttribute("sessionName",
        user.getFname_user()+" "+user.
        getLname_user());


 if(privilege!=null && privilege.size
      ()>0)
  privilege.clear();


 boolean studentCheck=false;
 boolean facultyCheck=false;
 boolean adminCheck=false;
 boolean workflowCheck=false;
 for(int i=0; i<currentRoleList.length
      ;i++){
 // privilegeCheck= adminSP.
       checkPrivilege(boom[i], "users",
        "insert");
  if(currentRoleList[i].indexOf("
       System")!=-1){

   PrivilegeLink userPriv= new
        PrivilegeLink();
   userPriv.setName("userURL");
   userPriv.setValue("Manage_Users");
   userPriv.setUrl(request.
        getContextPath()+"/forms/
        dentist/simpleForm/userForm.
        html");

   PrivilegeLink rolePriv= new
        PrivilegeLink();
   rolePriv.setName("roleURL");
   rolePriv.setValue("Manage_roles");
   rolePriv.setUrl(request.
        getContextPath()+"/forms/
        dentist/simpleForm/roleForm.
        html");

   PrivilegeLink sectionPriv= new
        PrivilegeLink();
   sectionPriv.setName("sectionURL");
   sectionPriv.setValue("Manage_
        sections");
   sectionPriv.setUrl(request.
        getContextPath()+"/forms/
        dentist/simpleForm/sectionForm
        .html");



   privilege.add(userPriv);
   privilege.add(rolePriv);
   privilege.add(sectionPriv);

   if(!adminCheck && !facultyCheck &&
        !workflowCheck){

   PrivilegeLink statPriv= new
        PrivilegeLink();
   statPriv.setName("viewStatURL");
   statPriv.setValue("View_Statistics"
        );
   statPriv.setUrl(request.
        getContextPath()+"/forms/
        dentist/search/statisticsform.
        html");
   privilege.add(statPriv);
    }



   adminCheck=true;
  }
  //for faculty
  if(currentRoleList[i].toString().
        toLowerCase().indexOf("faculty
        ")!=-1){
   if(!facultyCheck){
   PrivilegeLink sectionPriv= new
        PrivilegeLink();
   sectionPriv.setName("sectionURL");
   sectionPriv.setValue("View_Users_
        in_section");
   sectionPriv.setUrl(request.
        getContextPath()+"/forms/
        dentist/simpleForm/
        viewFacultySection.html");
   privilege.add(sectionPriv);

    PrivilegeLink queryPriv= new
        PrivilegeLink();
    queryPriv.setName("viewQueryURL")
        ;
```

```
  queryPriv.setValue("Query
       Patients");
  queryPriv.setUrl(request.
       getContextPath()+"/forms/
       dentist/search/
       viewQueryPatients.html");
  privilege.add(queryPriv);

 if(!adminCheck && !facultyCheck){
 PrivilegeLink statPriv= new
      PrivilegeLink();
 statPriv.setName("viewStatURL");
 statPriv.setValue("View Statistics
      ");
 statPriv.setUrl(request.
      getContextPath()+"/forms/
      dentist/search/statisticsform
      .html");
 privilege.add(statPriv);
 }

 facultyCheck=true;
 }
}
if(currentRoleList[i].toString().
     toLowerCase().indexOf("student
     ")!=-1){
 if(!studentCheck){

  PrivilegeLink queryPriv= new
       PrivilegeLink();
  queryPriv.setName("viewQueryURL")
       ;
  queryPriv.setValue("Query
       Patients");
  queryPriv.setUrl(request.
       getContextPath()+"/forms/
       dentist/search/
       viewQueryPatients.html");
  privilege.add(queryPriv);

  studentCheck=true;

 }
}
if(currentRoleList[i].toString().
     toLowerCase().indexOf("
     workflow")!=-1){
 if(!workflowCheck){

if(!adminCheck && !facultyCheck &&
     !workflowCheck){
 PrivilegeLink statPriv= new
      PrivilegeLink();
 statPriv.setName("viewStatURL");
 statPriv.setValue("View Statistics
      ");
 statPriv.setUrl(request.
      getContextPath()+"/forms/
      dentist/search/statisticsform
      .html");
 privilege.add(statPriv);
 }

 workflowCheck=true;
 }
}

}



 model.addAttribute("privilege",
      privilege);

 return "index";

}catch (Exception e) {
 // TODO Auto-generated catch block
 error= "ERROR: Username may not be a
      member of the system. Please Try
      Again";
 model.addAttribute("error",error);
 return "loginForm";

}

}

@RequestMapping(value = "/logout.html",
     method = RequestMethod.GET)
    public String logoutUser(Model model
         , HttpServletRequest request,
         HttpServletResponse response) {
 HttpSession session = request.
      getSession(false);

 try {
  session.removeAttribute("sessionUser")
       ;
    session.removeAttribute("
         sessionUserRole");
    session.removeAttribute("
         sessionUserId");
    session.removeAttribute("
         sessionRole");
    session.removeAttribute("
         currentRoleList");
  session.removeAttribute("
       currentDatabaseList");
  session.removeAttribute("sessionName")
```

```
        ;

    session.invalidate();

    privilege.clear();
          Login login= new Login();
    model.addAttribute("login",login);
   } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    System.out.println("BOOM_ERROR");
    return "loginForm";

   }


          return "loginForm";
      }

@RequestMapping(value = "/myProfile",
     method = RequestMethod.GET)
public String myProfileView(
     @RequestParam(value="user_id",
     required = true) int user_id,
     ModelMap model,
    HttpServletRequest request,
        HttpServletResponse response,
        HttpSession session) throws
        Exception{

   session=request.getSession(false);


   User user= new User();
   String allUsernames="";
   UserSP userSP= new UserSP();
   try {

    userSP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());
    tempUsernames= userSP.getUsernameList
        ();
    user=userSP.getUser(user_id);
   } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    System.out.println("BOOM_ERROR");
    return "Error";

   }

   for(int i=0;i<tempUsernames.size();i++)
   {
         if (allUsernames == "")
          allUsernames = tempUsernames.get
              (i);
         else
          allUsernames = allUsernames + "
              ," + tempUsernames.get(i);
   }
   //List of secret_questions
    ArrayList<String> secret_questions=
        new ArrayList<String>();

     secret_questions.add("What is your
         mother's middle name?");
     secret_questions.add("What is your
         favorite cartoon?");
     secret_questions.add("What was your
         childhood nickname?");
     secret_questions.add("What is the
         name of your favorite childhood
         friend? ");
     secret_questions.add("What is the
         first name of your oldest niece
         ?");
     secret_questions.add("Who is your
         favorite author?");
     secret_questions.add("What is the
         title of your favorite book?");
     secret_questions.add("What was the
         last name of your favorite
         teacher?");
     secret_questions.add("What was the
         name of your first pet?");
     secret_questions.add("What is the
         name of your all-time favorite
         sports team?");

      // ModelMap modelMap = new
          ModelMap();
      model.put("user",user);
      model.put("allUsernames",
          allUsernames);
      model.put("secret_questions",
          secret_questions);

      return "myProfile";
}


@RequestMapping(value = "/myProfile",
    method = RequestMethod.POST)
public String editmyProfile(
    @ModelAttribute("user") User user,
        BindingResult result, ModelMap
            model,
        HttpServletRequest request,
            HttpServletResponse
            response,HttpSession
            session) throws Exception{
```

```
session=request.getSession(false);


UserSP userSP= new UserSP();

try {

  userSP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 tempUsernames= userSP.getUsernameList
     ();
 userSP.changeProfile(user.getUser_id()
     ,user.getEmail());
 String success="SUCCESS: You have
     successfully changed your email
     address.";
      model.put("success",success);
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}


      model.addAttribute("user", user);

 return "myProfile";

}

@RequestMapping(value = "/changePassword
    ", method = RequestMethod.GET)
public String changePass(@RequestParam(
    value="user_id", required = true)
    int user_id,ModelMap model,
 HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{

session=request.getSession(false);


User user= new User();
UserSP userSP= new UserSP();

try {

  userSP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 user=userSP.getUser(user_id);

} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();

}

      model.put("user",user);

      return "changePassword";
}

@RequestMapping(value = "/changePassword
    ", method = RequestMethod.POST)
public String changePasswordPOST(
    @ModelAttribute("user") User user,
    ModelMap model,
 HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{

session=request.getSession(false);


UserSP userSP= new UserSP();

try {

  userSP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 userSP.changePassword(user.getUser_id
      (),user.getPassword());
 String success="SUCCESS: You have
      successfully changed your
      password.";
       model.put("success",success);
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}
      model.put("user",user);

      return "changePassword";
}


@RequestMapping(value = "/changeSecret",
     method = RequestMethod.GET)
public String changeSec(@RequestParam(
    value="user_id", required = true)
    int user_id,ModelMap model,
 HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
```

```
        Exception{

session=request.getSession(false);


User user= new User();
UserSP userSP= new UserSP();

try {

  userSP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

 user=userSP.getUser(user_id);
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}
//List of secret_questions
 ArrayList<String> secret_questions=
      new ArrayList<String>();

   secret_questions.add("What is your
        mother's middle name?");
   secret_questions.add("What is your
        favorite cartoon?");
   secret_questions.add("What was your
        childhood nickname?");
   secret_questions.add("What is the
        name of your favorite childhood
        friend? ");
   secret_questions.add("What is the
        first name of your oldest niece
        ?");
   secret_questions.add("Who is your
        favorite author?");
   secret_questions.add("What is the
        title of your favorite book?");
   secret_questions.add("What was the
        last name of your favorite
        teacher?");
   secret_questions.add("What was the
        name of your first pet?");
   secret_questions.add("What is the
        name of your all-time favorite
        sports team?");

      model.put("secret_questions",
           secret_questions);
      user.setSecret_answer("");

      model.put("user",user);
      return "changeSecret";
}

@RequestMapping(value = "/changeSecret",
      method = RequestMethod.POST)
public String changeSecretPOST(
    @ModelAttribute("user") User user,
    ModelMap model,
  HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{

session=request.getSession(false);


UserSP userSP= new UserSP();

try {

  userSP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

 userSP.changeSecret(user.getUser_id(),
      user.getSecret_question(),user.
      getSecret_answer());
 String success="SUCCESS: You have
      successfully changed your Secret
      Question & Answer.";
       model.put("success",success);

} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}

System.out.println("SECRET:::: "+user.
     getSecret_answer());
//List of secret_questions
 ArrayList<String> secret_questions=
      new ArrayList<String>();

   secret_questions.add("What is your
        mother's middle name?");
   secret_questions.add("What is your
        favorite cartoon?");
   secret_questions.add("What was your
        childhood nickname?");
   secret_questions.add("What is the
        name of your favorite childhood
        friend? ");
   secret_questions.add("What is the
        first name of your oldest niece
        ?");
   secret_questions.add("Who is your
        favorite author?");
   secret_questions.add("What is the
        title of your favorite book?");
```

```
        secret_questions.add("What was the
            last name of your favorite
            teacher?");
        secret_questions.add("What was the
            name of your first pet?");
        secret_questions.add("What is the
            name of your all-time favorite
            sports team?");

        model.put("secret_questions",
            secret_questions);

        model.put("user",user);

        return "changeSecret";
    }

}
```

Listing 21: LoginFilter

```
package com.dentist.version.three.mapper;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.
    HttpServletRequest;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet Filter implementation class
     LoginFilter
 */
public class LoginFilter implements
    Filter {

    /**
     * Default constructor.
     */
    public LoginFilter() {
        // TODO Auto-generated
            constructor stub
    }

 /**
  * @see Filter#destroy()
  */
 public void destroy() {
  // TODO Auto-generated method stub
 }

 /**
  * @see Filter#doFilter(ServletRequest,
      ServletResponse, FilterChain)
  */
 public void doFilter(ServletRequest req,
     ServletResponse res, FilterChain
     chain) throws IOException,
     ServletException {
 // TODO Auto-generated method stub
 // place your code here
  HttpServletRequest request = (
      HttpServletRequest) req;
       HttpServletResponse response = (
           HttpServletResponse) res;
   HttpSession session = request.
       getSession(false);

  // System.out.println("FILTER:: " +
      request.getContextPath() );
   if(request.getRequestURI().equals(
       request.getContextPath()+"/forms
       /dentist/login/logout.html")){

           // Set standard HTTP/1.1 no
               -cache headers.
           response.setHeader("Cache-
               Control", "no-store,
               no-cache,must-
               revalidate");
           // Set IE extended HTTP/1.1
               no-cache headers (use
               addHeader).
           response.addHeader("Cache-
               Control", "post-check
               =0,pre-check=0");

           // Set standard HTTP/1.0 no
               -cache header.
           response.setHeader("Pragma"
               , "no-cache");
         //  response.sendRedirect(
             request.getRequestURI())
             ;
        // Set to expire far in the
            past.
         response.setHeader("Expires",
             "Sat,6May1995 12:00:00
             GMT");
       chain.doFilter(req, res); //
```

```
            Logged-in user found, so
            just continue request.


        }
    else{

        if ((session == null || session.
            getAttribute("sessionUser")
            == null)&&
          (!request.getRequestURI().equals
              (request.getContextPath()+"
              /forms/dentist/login/
              loginForm.html")) &&
          (!request.getRequestURI().equals
              (request.getContextPath()+"
              /forms/dentist/login/
              loginOutput.html")) &&
          (!request.getRequestURI().equals
              (request.getContextPath()+"
              /forms/dentist/login/
              forgotPassword.html")) &&
          (!request.getRequestURI().equals
              (request.getContextPath()+"
              /forms/dentist/login/
              forgotPasswordUN.html")) &&
          (!request.getRequestURI().equals
              (request.getContextPath()+"
              /forms/dentist/login/
              forgotPasswordRetrieve.html
              ")))) {
          System.out.println(request.
              getRequestURI());



          request.getRequestDispatcher("/
              forms/dentist/login/
              loginForm.html").forward(req
              , res);
    }
        else{
            response.setHeader("Cache-
                Control", "no-store,no-
                cache,must-revalidate");
            // Set IE extended HTTP/1.1
                no-cache headers (use
                addHeader).
            response.addHeader("Cache-
                Control", "post-check
                =0,pre-check=0");

            // Set standard HTTP/1.0 no-
                cache header.
            response.setHeader("Pragma",
                "no-cache");
          //  response.sendRedirect(
                request.getRequestURI());
        // Set to expire far in the past
            .
          response.setHeader("Expires", "
              Sat,6May1995 12:00:00
              GMT");

        chain.doFilter(req, res); //
            Logged-in user found, so
            just continue request.

        }
    }


}

/**
 * @see Filter#init(FilterConfig)
 */
public void init(FilterConfig fConfig)
    throws ServletException {
 // TODO Auto-generated method stub
}

}
```

Listing 22: LoginSP

```
package com.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

import com.dentist.version.three.form.
    Role;
import com.dentist.version.three.form.
    User;
import com.dentist.version.three.mapper.
    GeneratePassword;
import com.dentist.version.three.mapper.
    RoleMapper;
import com.dentist.version.three.mapper.
    UserMapper;

public class LoginSP {

 private String database_username;
 private final String database_password="
```

```
    yes?bakitpo?";

public void setDatabase_username(String
    database_username) {
 this.database_username =
     database_username;
}


public String getDatabase_username() {
 return database_username;
}

 public int getUserID(String username)
     throws Exception{
  Class.forName("org.postgresql.Driver")
      .newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt","
       postgres","root");
   conn.setAutoCommit(false);

     PreparedStatement calstat=conn.
         prepareCall("{call_getUserID
         (?)}");
        calstat.setString(1, username);
         ResultSet rs= calstat.
             executeQuery();
       int result=0;

       while(rs.next()){
               System.out.println(rs.
                   getInt(1));
               result= rs.getInt(1);
              }


         conn.close();
         calstat.close();

         return result;


 }

//get User
 public User getUser(int user_id) throws
      Exception{
  Class.forName("org.postgresql.Driver")
      .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt","
        postgres","root");
   conn.setAutoCommit(false);
  CallableStatement calstat=conn.
      prepareCall("{call_getuser(?)}"
      );
    calstat.setInt(1, user_id);

    ResultSet rs = calstat.
        executeQuery();
       User user= new User();
       UserMapper usermap= new
           UserMapper();
   while(rs.next()){

   user= usermap.mapRow(rs, 10);

       }


   conn.close();
   calstat.close();
   System.out.println("Successful_call
       _for_getuser_function");
    return user;
}

//reset password
 public String resetPassword(int
     user_id, String secret_question,
     String secret_answer) throws
     Exception{
  GeneratePassword password= new
      GeneratePassword();
  Class.forName("org.postgresql.Driver"
      ).newInstance();
  String generatePassword= password.
      generateRandomPassword();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt","
        postgres","root");

     CallableStatement calstat=conn.
         prepareCall("{call_
         retrievepassword(?,?,?,?)}")
         ;
       //calstat.registerOutParameter
           (10, java.sql.Types.
           INTEGER);
         calstat.setInt(1,user_id);
          calstat.setString(2,
              secret_question);
         calstat.setString(3,
             secret_answer);
         calstat.setString(4,
             generatePassword);
```

```
            ResultSet rs= calstat.
                executeQuery();



            conn.close();
            calstat.close();
            System.out.println("Successul
                in resetting password."
                );

        return generatePassword;
    }

    //login User
    public int loginUser(String username,
        String password) throws Exception
        {
     Class.forName("org.postgresql.Driver"
         ).newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt","
            postgres","root");
     // conn.setAutoCommit(false);
       CallableStatement calstat=conn.
            prepareCall("{call login_user
            (?,?)}");
         calstat.setString(1, username);
         calstat.setString(2, password);

         ResultSet rs = calstat.
              executeQuery();
         int user_id=0;
        while(rs.next()){
              user_id=rs.getInt(1);
             }

         conn.close();
        calstat.close();
        System.out.println("Successful
             call for login user function"
             );

        return user_id;
    }

    // getuserrole
    public ArrayList<Integer> getUserRole(
        int user_id) throws Exception{

     Class.forName("org.postgresql.Driver"
         ).newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:postgresql
            ://localhost:5432/DentISt","
            postgres","root");
     // conn.setAutoCommit(false);
       CallableStatement calstat=conn.
            prepareCall("{call getuserrole
            (?)}");
         calstat.setInt(1, user_id);

         ResultSet rs = calstat.
              executeQuery();

         ArrayList<Integer> role_id= new
              ArrayList<Integer>();
         while(rs.next()){
              role_id.add(rs.getInt(1));
              System.out.println("THIS IS
                   IT :"+role_id);
             }

         conn.close();
        calstat.close();
        System.out.println("Successful
             call for login user function"
             );

        return role_id;
    }


    //get Role
    public Role getRole(int role_id)
        throws Exception{
     Class.forName("org.postgresql.Driver"
         ).newInstance();
     Connection conn=DriverManager.
          getConnection("jdbc:postgresql
          ://localhost:5432/DentISt","
          postgres","root");
      conn.setAutoCommit(false);
       CallableStatement calstat=conn.
            prepareCall("{call getrole(?)}
            ");
         calstat.setInt(1, role_id);

         ResultSet rs = calstat.
              executeQuery();
              Role role= new Role();
              RoleMapper rolemap= new
                   RoleMapper();
         while(rs.next()){

          role= rolemap.mapRow(rs, 5);

             }


         conn.close();
```

```
        calstat.close();
        System.out.println("Successful
            call for getrole function");
         return role;
    }

    //get database role
    public String getDatabaseRole(int
        role_id) throws Exception{
     Class.forName("org.postgresql.Driver"
        ).newInstance();
     Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt","
        postgres","root");
      conn.setAutoCommit(false);
       CallableStatement calstat=conn.
          prepareCall("{call
          getdatabaserole(?)}");
        calstat.setInt(1, role_id);

        ResultSet rs = calstat.
            executeQuery();
        Role role= new Role();
        RoleMapper rolemap= new
            RoleMapper();
        String database_role="";
      while(rs.next()){

       database_role= rs.getString(1);
        }


      conn.close();
      calstat.close();
      System.out.println("Successful
          call for getrole function");
       return database_role;
    }


}
```

Listing 23: MailUtil

```
package com.dentist.version.three.mapper;


/*
 * Sudhir Ancha
 * Site : http://www.javacommerce.com/
     displaypage.jsp?name=javamail.sql&
     id=18274
 *
 * 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 java.util.*;

public class MailUtil{


 private String SMTP_HOST_NAME = "
     localhost";
 //private String SMTP_FROM = "
     admin@dentist.upm.edu.ph";

 public Boolean postMail( String
     recipients[], String subject,
     String message, String from){
  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", 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(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();
   System.out.println("NAGERROR_TOINKS");
   return false;
  } catch (Exception ex) {
   ex.printStackTrace();
   System.out.println("NAGERROR_TOINKS");
   return false;
  }
  return true;
 }


}
```

Listing 24: QuerySP

```
package com.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

import com.dentist.version.three.form.
    AuditTrail;
import com.dentist.version.three.form.
    ConsultationsReferrals;
import com.dentist.version.three.form.
    DentalChart;
import com.dentist.version.three.form.
    Patient;
import com.dentist.version.three.form.
    PatientInformation;
import com.dentist.version.three.form.
    Section;
import com.dentist.version.three.form.
    ServiceNeeded;
import com.dentist.version.three.mapper.
    AuditMapper;
import com.dentist.version.three.mapper.
    ConsultationMapper;
import com.dentist.version.three.mapper.
    DentalChartMapper;
import com.dentist.version.three.mapper.
    PatientInfoMapper;
import com.dentist.version.three.mapper.
    PatientMapper;
import com.dentist.version.three.mapper.
    SectionMapper;
import com.dentist.version.three.mapper.
    ServicesNeededMapper;

public class QuerySP {

 private String database_username;
 private final String database_password="
     yes?bakitpo?";

 public void setDatabase_username(String
     database_username) {
  this.database_username =
      database_username;
 }


 public String getDatabase_username() {
  return database_username;
 }

 //list of all patients
 public ArrayList<Patient> allPatients()
     throws Exception{
  ArrayList<Patient> list= new ArrayList<
      Patient>();
  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
        prepareCall("{call
        listallpatient()}");

     ResultSet rs = calstat.executeQuery
         ();

     while(rs.next()){
      Patient patient= new Patient();
      PatientMapper patientMap= new
          PatientMapper();
     patient= patientMap.mapRow(rs, 17);
```

```
        list.add(patient);
           }


      conn.close();
      calstat.close();
      System.out.println("Successful call
           for listallpatient function");
       return list;
}

//list of all dentalchart
public ArrayList<DentalChart>
     allDentalChart() throws Exception{
 ArrayList<DentalChart> list= new
      ArrayList<DentalChart>();
 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call
        listalldentalchart()}");

     ResultSet rs = calstat.executeQuery
          ();

     while(rs.next()){
      DentalChart dentalChart= new
           DentalChart();
      DentalChartMapper map= new
           DentalChartMapper();
    dentalChart= map.mapRow(rs, 35);

     list.add(dentalChart);
     //System.out.println(rs.getString
          (1));
        }


      conn.close();
      calstat.close();
      System.out.println("Successful call
           for listalldentalchart function
           ");
       return list;
}

//list of all serviceneeded
public ArrayList<ServiceNeeded>
     allServiceNeeded() throws Exception
     {
 ArrayList<ServiceNeeded> list= new
      ArrayList<ServiceNeeded>();
 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call
        listallserviceneeded()}");

     ResultSet rs = calstat.executeQuery
          ();

     while(rs.next()){
      ServiceNeeded service= new
           ServiceNeeded();
      ServicesNeededMapper map= new
           ServicesNeededMapper();
    service= map.mapRow(rs, 30);

     list.add(service);
     //System.out.println(rs.getString
          (1));
        }


      conn.close();
      calstat.close();
      System.out.println("Successful call
           for listallserviceneeded
           function");
       return list;
}

//list of all patients
public ArrayList<ConsultationsReferrals>
     allConsultation() throws Exception
     {
 ArrayList<ConsultationsReferrals> list=
      new ArrayList<
      ConsultationsReferrals>();
 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call
        listallconsultations()}");
```

```
        ResultSet rs = calstat.executeQuery
            ();

        while(rs.next()){
        ConsultationsReferrals consult= new
            ConsultationsReferrals();
        ConsultationMapper consultMapper=
            new ConsultationMapper();
        consult= consultMapper.mapRow(rs,
            16);

        list.add(consult);
           }


        conn.close();
        calstat.close();
        System.out.println("Successful call
            for listallconsultations
            function");
         return list;
}


//allpatientinfo
public ArrayList<PatientInformation>
     allPatientInfo() throws Exception{
 ArrayList<PatientInformation> list= new
       ArrayList<PatientInformation>();
 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call
       listallpatientinfo()}");

        ResultSet rs = calstat.executeQuery
            ();

        while(rs.next()){
        PatientInformation patient= new
            PatientInformation();
         PatientInfoMapper patientMap= new
              PatientInfoMapper();
        patient= patientMap.mapRow(rs, 24);

        list.add(patient);
           }
        conn.close();
        calstat.close();
        System.out.println("Successful call
            for listallpatient function");
         return list;
}

//get Patient
public Patient getPatient(int patient_id
    ) throws Exception{


 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call getpatient(?)
       }");
     calstat.setInt(1, patient_id);

      ResultSet rs = calstat.executeQuery
          ();
          Patient patient= new Patient();
          PatientMapper patientmap= new
              PatientMapper();
     while(rs.next()){

     patient= patientmap.mapRow(rs, 17);


          }


     conn.close();
     calstat.close();
     System.out.println("Successful call
         for getpatient function");
      return patient;
}

 public ArrayList<ServiceNeeded>
      getServiceNeeded(int patient_id)
      throws Exception{
  ArrayList<ServiceNeeded> list= new
       ArrayList<ServiceNeeded>();
  Class.forName("org.postgresql.Driver")
       .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
```

```
    conn.setAutoCommit(false);
     CallableStatement calstat=conn.
          prepareCall("{call_
          listspecifiedserviceneeded(?)}"
          );
        calstat.setInt(1, patient_id);

       ResultSet rs = calstat.
            executeQuery();

       while(rs.next()){
        ServiceNeeded service= new
             ServiceNeeded();
        ServicesNeededMapper map= new
             ServicesNeededMapper();
      service= map.mapRow(rs, 30);

       list.add(service);
          }


      conn.close();
      calstat.close();
      System.out.println("Successful_call
           _for_listallserviceneeded_
           function");
       return list;
}


public ArrayList<DentalChart>
     getDentalChart(int patient_id)
     throws Exception{

 ArrayList<DentalChart> list= new
      ArrayList<DentalChart>();


 Class.forName("org.postgresql.Driver")
      .newInstance();
 Connection conn=DriverManager.
      getConnection("jdbc:postgresql://
      localhost:5432/DentISt",
      database_username,
      database_password);
  conn.setAutoCommit(false);

    PreparedStatement calstat=conn.
         prepareCall("{call_
         listspecifieddentalchart(?)}")
         ;
       calstat.setInt(1, patient_id);

        ResultSet rs = calstat.
             executeQuery();

        while(rs.next()){
         DentalChart dentalChart= new
              DentalChart();
         DentalChartMapper map= new
              DentalChartMapper();
        dentalChart= map.mapRow(rs,
             35);

         list.add(dentalChart);
            }
       conn.close();
       calstat.close();

       return list;


}

//getpatientinfo

public ArrayList<PatientInformation>
     getPatientInfo(int patient_id)
     throws Exception{

 ArrayList<PatientInformation> list=
      new ArrayList<PatientInformation
      >();


 Class.forName("org.postgresql.Driver")
      .newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  conn.setAutoCommit(false);

   CallableStatement calstat=conn.
        prepareCall("{call_
        getpatientinfo(?)}");
     calstat.setInt(1, patient_id);

      ResultSet rs = calstat.
           executeQuery();

       while(rs.next()){
       PatientInformation patient= new
            PatientInformation();
        PatientInfoMapper patientMap=
             new PatientInfoMapper()
            ;
       patient= patientMap.mapRow(rs,
            24);

        list.add(patient);
```

```
            //System.out.println(rs.
                getString(1));
            }


    conn.close();
    calstat.close();
    System.out.println("Successful call
        for getpatient function");
     return list;
}


//get Consultation
public ArrayList<ConsultationsReferrals
    > getConsultations(int patient_id)
     throws Exception{

 ArrayList<ConsultationsReferrals> list
     = new ArrayList<
     ConsultationsReferrals>();


 Class.forName("org.postgresql.Driver")
     .newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
  conn.setAutoCommit(false);

   CallableStatement calstat=conn.
       prepareCall("{call
       getconsultationandfindingsquery
       (?)}");
     calstat.setInt(1, patient_id);

       ResultSet rs = calstat.
           executeQuery();

        while(rs.next()){
        ConsultationsReferrals consult=
            new
            ConsultationsReferrals();
          ConsultationMapper patientMap
            = new ConsultationMapper
            ();
        consult= patientMap.mapRow(rs,
            16);

        list.add(consult);
        //System.out.println(rs.
            getString(1));
        }


    conn.close();
    calstat.close();
    System.out.println("Successful call
        for getpatient function");
     return list;
}

//getpatientDIlist
public ArrayList<Integer>
    getpatientIDList() throws
    Exception{
 ArrayList<Integer> listpatientids= new
     ArrayList<Integer>();
 Class.forName("org.postgresql.Driver")
     .newInstance();
 System.out.println("Check"+
     database_username);
 Connection conn=DriverManager.
     getConnection("jdbc:postgresql://
     localhost:5432/DentISt",
     database_username,
     database_password);


  conn.setAutoCommit(false);
  CallableStatement calstat=conn.
      prepareCall("{call
      listallpatientids()}");

     ResultSet rs = calstat.
         executeQuery();

     while(rs.next()){
    listpatientids.add(rs.getInt(1));
      }


    conn.close();
    calstat.close();
    System.out.println("Successful call
        for listallusername function"
        );

  return listpatientids;

}
}
```

Listing 25: Role

```
package com.dentist.version.three.form;


//table object
public class Role {
 private int role_id;
 private String role_name;
 private String role_description;
 private String created_by;
 private String created_date;
 private String database_role;
 public void setRole_id(int role_id) {
  this.role_id = role_id;
 }
 public int getRole_id() {
  return role_id;
 }
 public void setRole_name(String
      role_name) {
  this.role_name = role_name;
 }
 public String getRole_name() {
  return role_name;
 }
 public void setRole_description(String
      role_description) {
  this.role_description =
       role_description;
 }
 public String getRole_description() {
  return role_description;
 }
 public void setCreated_by(String
      created_by) {
  this.created_by = created_by;
 }
 public String getCreated_by() {
  return created_by;
 }
 public void setCreated_date(String
      created_date) {
  this.created_date = created_date;
 }
 public String getCreated_date() {
  return created_date;
 }
 public void setDatabase_role(String
      database_role) {
  this.database_role = database_role;
 }
 public String getDatabase_role() {
  return database_role;
 }



}
```

Listing 26: RoleMapper

```
package com.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
     RowMapper;

import com.dentist.version.three.form.
     Role;

public class RoleMapper implements
     RowMapper<Role> {
    public Role mapRow(ResultSet rs, int
         rowNum) throws SQLException {
        Role role = new Role();
        role.setRole_id(rs.getInt("
             role_id"));
        role.setRole_name(rs.getString("
             role_name"));
        role.setRole_description(rs.
             getString("role_description"
             ));
        role.setCreated_by(rs.getString("
             created_by"));
        role.setCreated_date(rs.getString
             ("created_date"));


        return role;
    }
 }
```

Listing 27: RoleSection

```
package com.dentist.version.three.form;

public class RoleSection {
 private String role_name;
 private int section_id;
 public void setRole_name(String
      role_name) {
  this.role_name = role_name;
 }
```

```
 public String getRole_name() {
  return role_name;
 }
 public void setSection_id(int section_id
     ) {
  this.section_id = section_id;
 }
 public int getSection_id() {
  return section_id;
 }


}
```

Listing 28: RoleSP

```
package com.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.ArrayList;

import com.dentist.version.three.form.
    Role;
import com.dentist.version.three.form.
    User;
import com.dentist.version.three.mapper.
    RoleMapper;
import com.dentist.version.three.mapper.
    UserMapper;



public class RoleSP {

 private String database_username;
 private final String database_password="
     yes?bakitpo?";

 public void setDatabase_username(String
     database_username) {
  this.database_username =
      database_username;
 }



 public String getDatabase_username() {
  return database_username;
 }



 // insert role
  public int executeRole(String role_name
      , String role_description,String
      created_by, String created_date)
      throws Exception{

  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

     CallableStatement calstat=conn.
         prepareCall("{call_insert_role
         (?,?,?,?)}");
        calstat.setString(1,role_name);
        calstat.setString(2,
            role_description);
        calstat.setString(3,created_by)
            ;
        calstat.setString(4,
            created_date);

        ResultSet rs = calstat.
            executeQuery();
        int role_id=0;
        while(rs.next()){
            role_id= rs.getInt(1);
          }

        conn.close();
        calstat.close();
        System.out.println("Your_data_
            has_been_inserted_into_
            table.");

        return role_id;
 }

 //list all rolenames
 public ArrayList<String> listAllRoles()
     throws Exception{
  ArrayList<String> listRoles= new
      ArrayList<String>();
  Class.forName("org.postgresql.Driver").
      newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
   conn.setAutoCommit(false);
    CallableStatement calstat=conn.
        prepareCall("{call_
```

```
        listallrolenames()}");

      ResultSet rs = calstat.executeQuery
          ();

      while(rs.next()){
          System.out.println(rs.
              getString(1));
          listRoles.add(rs.getString(1))
              ;
        }


    conn.close();
    calstat.close();
    System.out.println("Role data
        selected.");
     return listRoles;
}


//list all roles
public ArrayList<Role> allRoleList()
    throws Exception{
 ArrayList<Role> listRoles= new
     ArrayList<Role>();
 Class.forName("org.postgresql.Driver").
     newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call listallroles
       ()}");

     ResultSet rs = calstat.executeQuery
         ();

     while(rs.next()){
         Role role= new Role();
         RoleMapper rolemap= new
             RoleMapper();
         role= rolemap.mapRow(rs, 5);


       listRoles.add(role);
     //System.out.println(rs.getString
         (1));
        }


    conn.close();
    calstat.close();
    System.out.println("Successful call
        for listallroles function");
     return listRoles;
}



//get Role
public Role getRole(int role_id) throws
    Exception{
 Class.forName("org.postgresql.Driver").
     newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call getrole(?)}")
       ;
     calstat.setInt(1, role_id);

     ResultSet rs = calstat.executeQuery
         ();
         Role role= new Role();
         RoleMapper rolemap= new
             RoleMapper();
    while(rs.next()){

    role= rolemap.mapRow(rs, 5);


        }


    conn.close();
    calstat.close();
    System.out.println("Successful call
        for getrole function");
     return role;
}

//update role
 public void updateRole(int role_id,
     String role_name, String
     role_description, String
     database_role) throws Exception{

  Class.forName("org.postgresql.Driver")
      .newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
```

```
        CallableStatement calstat=conn.
            prepareCall("{call_
            update_roles(?,?,?,?)}");
          //calstat.registerOutParameter
              (10, java.sql.Types.INTEGER
              );
            calstat.setInt(1,role_id);
              calstat.setString(2,
                  role_name);
            calstat.setString(3,
                role_description);
            calstat.setString(4,
                database_role);

            System.out.println(
                role_description);

            ResultSet rs= calstat.
                executeQuery();

            conn.close();
            calstat.close();
            System.out.println("Your_data_
                has_been_updated_into_
                table.");


    }
//delete User
 public void deleteRoles(int role_id)
      throws Exception{
  Class.forName("org.postgresql.Driver")
       .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  // conn.setAutoCommit(false);
    CallableStatement calstat=conn.
        prepareCall("{call_delete_roles
        (?)}");
      calstat.setInt(1, role_id);

      ResultSet rs = calstat.
          executeQuery();



     conn.close();
     calstat.close();
     System.out.println("Successful_call
         _for_DELETE_function");
 }

 //get list of users in a role
  public  ArrayList<User>
      getListUserRole(int role_id)
      throws Exception{

 ArrayList<User> userLists= new
      ArrayList<User>();

 Class.forName("org.postgresql.Driver"
      ).newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);

    CallableStatement calstat=conn.
        prepareCall("{call_
        listuserrole(?)}");
      //calstat.registerOutParameter
          (10, java.sql.Types.
          INTEGER);
        calstat.setInt(1,role_id);

        ResultSet rs= calstat.
            executeQuery();

        while(rs.next()){

         User user= new User();
              UserMapper usermap=
                  new UserMapper();
             user= usermap.mapRow(rs,
                 5);



            userLists.add(user);
        }

        conn.close();
        calstat.close();
        System.out.println("Function_
            has_been_called.");

        return userLists;
 }

//insert database role
public void insert_databaseRole(int
    role_id, String database_role)
    throws Exception{

 Class.forName("org.postgresql.Driver"
      ).newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
```

```
            database_password);

        CallableStatement calstat=conn.
             prepareCall("{call_
             insert_database_role(?,?)}")
             ;
             calstat.setInt(1,role_id);
             calstat.setString(2,
                  database_role);

             ResultSet rs = calstat.
                  executeQuery();

             conn.close();
             calstat.close();
             System.out.println("Your_data
                  _has_been_inserted_into_
                  table.");

    }

  //get database role
   public String getDatabaseRole(int
        role_id) throws Exception{
    Class.forName("org.postgresql.Driver"
         ).newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql
           ://localhost:5432/DentISt",
           database_username,
           database_password);
      conn.setAutoCommit(false);
       CallableStatement calstat=conn.
           prepareCall("{call_
           getdatabaserole(?)}");
        calstat.setInt(1, role_id);

        ResultSet rs = calstat.
             executeQuery();
            Role role= new Role();
            RoleMapper rolemap= new
                 RoleMapper();
            String database_role="";
       while(rs.next()){

        database_role= rs.getString(1);
        //System.out.println(rs.getString
             (1));
            }


        conn.close();
        calstat.close();
        System.out.println("Successful_
             call_for_getrole_function");
        return database_role;
    }


}
```

Listing 29: RoleUser

```
package com.dentist.version.three.form;

public class RoleUser {
 private String role_name;
 private int user_id;

 public void setUser_id(int user_id) {
  this.user_id = user_id;
 }
 public int getUser_id() {
  return user_id;
 }
 public void setRole_name(String
      role_name) {
  this.role_name = role_name;
 }
 public String getRole_name() {
  return role_name;
 }


}
```

Listing 30: SearchController

```
package com.dentist.version.three.
     controller;

import javax.servlet.http.
     HttpServletRequest;
import javax.servlet.http.
     HttpServletResponse;
import org.springframework.stereotype.
     Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
     annotation.RequestMapping;
import org.springframework.web.bind.
     annotation.RequestMethod;

@Controller
public class SearchController {
```

```
@RequestMapping(value = "/
    viewQueryPatients.html", method =
    RequestMethod.GET)
public String viewQueryPatient(Model
    model, HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

 return "queryPatients";

}
```

```
@RequestMapping(value = "/statisticsform
    .html", method = RequestMethod.GET)
public String viewStatisticsForm(Model
    model, HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

 return "statisticsform";

}

}
```

Listing 31: SearchResultController

```
package com.dentist.version.three.
    controller;

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

import javax.servlet.http.
    HttpServletRequest;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.
    Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
    annotation.ModelAttribute;
import org.springframework.web.bind.
    annotation.RequestMapping;
import org.springframework.web.bind.
    annotation.RequestMethod;
import org.springframework.web.bind.
    annotation.RequestParam;
import org.springframework.web.servlet.
    ModelAndView;

import com.dentist.version.three.db.
    AdminSP;
import com.dentist.version.three.db.
    QuerySP;
import com.dentist.version.three.form.
    ConsultationsReferrals;
import com.dentist.version.three.form.
    DentalChart;
import com.dentist.version.three.form.
    Patient;
import com.dentist.version.three.form.
    PatientInformation;
import com.dentist.version.three.form.
    ServiceNeeded;

@Controller
public class SearchResultController {
 @RequestMapping(value = "/viewResults.
    html", method = RequestMethod.POST)
 public ModelAndView viewQueryPatient(
    Model model, HttpServletRequest
    request, HttpServletResponse
    response) throws Exception {

  return new ModelAndView("searchResults"
    );

 }

 /*
  * MODEL ATTRIBUTE
  */

 @ModelAttribute("theConsultations")
 protected ArrayList<
    ConsultationsReferrals>
    getConsultArrayList(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
  session=request.getSession(false);
  QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
   ArrayList<ConsultationsReferrals>
      consultArrayList=querySP.
      allConsultation();
   ArrayList<ConsultationsReferrals>
      finalConsultList= new ArrayList<
      ConsultationsReferrals>();
   List<Integer> checkRepeat= new
      ArrayList<Integer>();
```

```
  int index=0;
  for(ConsultationsReferrals consult:
       consultArrayList){
   ArrayList<ConsultationsReferrals>
        tempConsult=querySP.
        getConsultations(consult.
        getPatientid());
   int latest=0;
   if(checkRepeat.size()==0){
   if(tempConsult.size() != 0){
    latest = tempConsult.size()-1;
    finalConsultList.add(tempConsult.get
         (latest));
    checkRepeat.add(tempConsult.get(
         latest).getPatientid());
   }
   }else{

    for(int i=0; i<checkRepeat.size();i
         ++){
     if(consult.getPatientid()==
          checkRepeat.get(i))
      break;
     else{
      if(tempConsult.size() != 0){
       latest = tempConsult.size()-1;
       finalConsultList.add(tempConsult.
            get(latest));
       checkRepeat.add(tempConsult.get(
            latest).getPatientid());
       break;
      }
     }

    }

   }
  }

  return finalConsultList;
}

@ModelAttribute("perio")
public String perio(
   @RequestParam(value="perio", required
        = false) String perio)
   {
    return perio;
}
@ModelAttribute("rpd")
public String rpd(
   @RequestParam(value="rpd", required =
        false) String rpd)
   {
    return rpd;
}
@ModelAttribute("resto")
public String resto(
   @RequestParam(value="resto", required
        = false) String resto)
   {
    return resto;
}
@ModelAttribute("os")
public String os(
   @RequestParam(value="os", required =
        false) String os)
   {
    return os;
}
@ModelAttribute("fpd")
public String fpd(
   @RequestParam(value="fpd", required =
        false) String fpd)
   {
    return fpd;
}
@ModelAttribute("pedo")
public String pedo(
   @RequestParam(value="pedo", required
        = false) String pedo)
   {
    return pedo;
}
@ModelAttribute("endo")
public String endo(
   @RequestParam(value="endo", required
        = false) String endo)
   {
    return endo;
}
@ModelAttribute("cd")
public String cd(
   @RequestParam(value="cd", required =
        false) String cd)
   {
    return cd;
}
@ModelAttribute("ortho")
public String ortho(
   @RequestParam(value="ortho", required
        = false) String ortho)
   {
    return ortho;
}
@ModelAttribute("periodontics")
public String periodontics(
   @RequestParam(value="periodontics",
        required = false) String
        periodontics)
   {
    return periodontics;
```

```
}

@ModelAttribute("class1")
public String class1(
   @RequestParam(value="class1",
        required = false) String class1)
   {
    return class1;
}
@ModelAttribute("class2")
public String class2(
   @RequestParam(value="class2",
        required = false) String class2)
   {
    return class2;
}
@ModelAttribute("class3")
public String class3(
   @RequestParam(value="class3",
        required = false) String class3)
   {
    return class3;
}
@ModelAttribute("class4")
public String class4(
   @RequestParam(value="class4",
        required = false) String class4)
   {
    return class4;
}
@ModelAttribute("class5")
public String class5(
   @RequestParam(value="class5",
        required = false) String class5)
   {
    return class5;
}
@ModelAttribute("onlay")
public String onlay(
   @RequestParam(value="onlay", required
        = false) String onlay)
   {
    return onlay;
}
@ModelAttribute("extraction")
public String extraction(
   @RequestParam(value="extraction",
        required = false) String
        extraction)
   {
    return extraction;
}
@ModelAttribute("odontectomy")
public String odontectomy(
   @RequestParam(value="odontectomy",
        required = false) String
        odontectomy)
   {
    return odontectomy;
}
@ModelAttribute("specialcase")
public String specialcase(
   @RequestParam(value="specialcase",
        required = false) String
        specialcase)
   {
    return specialcase;
}
@ModelAttribute("pedodontics")
public String pedodontics(
   @RequestParam(value="pedodontics",
        required = false) String
        pedodontics)
   {
    return pedodontics;
}
@ModelAttribute("orthodontics")
public String orthodontics(
   @RequestParam(value="orthodontics",
        required = false) String
        orthodontics)
   {
    return orthodontics;
}
@ModelAttribute("pulpsedation")
public String pulpsedation(
   @RequestParam(value="pulpsedation",
        required = false) String
        pulpsedation)
   {
    return pulpsedation;
}
@ModelAttribute("crownrecementation")
public String crownrecementation(
   @RequestParam(value="
        crownrecementation", required =
        false) String crownrecementation
        )
   {
    return crownrecementation;
}
@ModelAttribute("tempfillingservice")
public String tempfillingservice(
   @RequestParam(value="
        tempfillingservice", required =
        false) String tempfillingservice
        )
   {
    return tempfillingservice;
}
@ModelAttribute("acuteinfections")
public String acuteinfections(
```

```
    @RequestParam(value="acuteinfections"
        , required = false) String
        acuteinfections)
    {
     return acuteinfections;
}
@ModelAttribute("traumaticinjuries")
public String traumaticinjuries(
    @RequestParam(value="
        traumaticinjuries", required =
        false) String traumaticinjuries)
    {
     return traumaticinjuries;
}
@ModelAttribute("laminated")
public String laminated(
    @RequestParam(value="laminated",
        required = false) String
        laminated)
    {
     return laminated;
}
@ModelAttribute("singlecrown")
public String singlecrown(
    @RequestParam(value="singlecrown",
        required = false) String
        singlecrown)
    {
     return singlecrown;
}
@ModelAttribute("bridge")
public String bridge(
    @RequestParam(value="bridge",
        required = false) String bridge)
    {
     return bridge;
}
@ModelAttribute("anterior")
public String anterior(
    @RequestParam(value="anterior",
        required = false) String
        anterior)
    {
     return anterior;
}
@ModelAttribute("posterior")
public String posterior(
    @RequestParam(value="posterior",
        required = false) String
        posterior)
    {
     return posterior;
}
@ModelAttribute("completedentservice")
public String completedentservice(
    @RequestParam(value="
        completedentservice", required =
         false) String
        completedentservice)
    {
     return completedentservice;
}
@ModelAttribute("singledentservice")
public String singledentservice(
    @RequestParam(value="
        singledentservice", required =
        false) String singledentservice)
    {
     return singledentservice;
}
@ModelAttribute("removablepartialservice
    ")
public String removablepartialservice(
    @RequestParam(value="
        removablepartialservice",
        required = false) String
        removablepartialservice)
    {
     return removablepartialservice;
}
/**
 * This class returns the form backing
     object. This can be a string, a
     boolean, or a normal java
 * pojo. The bean name defined in the
     ModelAttribute annotation and the
     type can be just
 * defined by the return type of this
     method
 */

@ModelAttribute("theServicesNeeded")
protected ArrayList<ServiceNeeded>
    getServicesNeeded(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 ArrayList<ServiceNeeded> services =null
     ;
 session=request.getSession(false);

 ArrayList<Patient> patients=null;


 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
  services=querySP.allServiceNeeded();

 return services;
```

```
}
@ModelAttribute("theDentalChart")
protected ArrayList<DentalChart>
     getDentalChart(HttpServletRequest
     request,HttpSession session) throws
      Exception {
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
  ArrayList<DentalChart> chart=querySP.
       allDentalChart();

 return chart;
}

/*
@ModelAttribute("theTreatmentPlan")
protected ArrayList<TreatmentPlan>
     getTreatmentPlans(
     HttpServletRequest request) throws
     Exception {
 ArrayList<TreatmentPlan> treatmentplan
      = Context.getService(DentalService
      .class).getAllTreatmentPlans();

 return treatmentplan;
}
*/
@ModelAttribute("caries")
public String caries(
   @RequestParam(value="caries",
        required = false) String caries)
   {
    return caries;
}

@ModelAttribute("recurrentcaries")
public String recurrentcaries(
   @RequestParam(value="recurrentcaries"
        , required = false) String
        recurrentcaries)
   {
    return recurrentcaries;
}
@ModelAttribute("restoration")
public String restoration(
   @RequestParam(value="restoration",
        required = false) String
        restoration)
   {
    return restoration;
}
@ModelAttribute("amalgam")
public String amalgam(
   @RequestParam(value="amalgam",
        required = false) String amalgam
        )
   {
    return amalgam;
}
@ModelAttribute("composite")
public String composite(
   @RequestParam(value="composite",
        required = false) String
        composite)
   {
    return composite;
}
@ModelAttribute("glassionomer")
public String glassionomer(
   @RequestParam(value="glassionomer",
        required = false) String
        glassionomer)
   {
    return glassionomer;
}
@ModelAttribute("tempfilling")
public String tempfilling(
   @RequestParam(value="tempfilling",
        required = false) String
        tempfilling)
   {
    return tempfilling;
}
@ModelAttribute("extrusion")
public String extrusion(
   @RequestParam(value="extrusion",
        required = false) String
        extrusion)
   {
    return extrusion;
}

@ModelAttribute("intrusion")
public String intrusion(
   @RequestParam(value="intrusion",
        required = false) String
        intrusion)
   {
    return intrusion;
}

@ModelAttribute("mesialdrift")
public String mesialdrift(
   @RequestParam(value="mesialdrift",
        required = false) String
        mesialdrift)
   {
    return mesialdrift;
```

```
}

@ModelAttribute("distaldrift")
public String distaldrift(
   @RequestParam(value="distaldrift",
        required = false) String
        distaldrift)
   {
    return distaldrift;
}

@ModelAttribute("completedenture")
public String completedenture(
   @RequestParam(value="completedenture"
        , required = false) String
        completedenture)
   {
    return completedenture;
}

@ModelAttribute("singledenture")
public String singledenture(
   @RequestParam(value="singledenture",
        required = false) String
        singledenture)
   {
    return singledenture;
}

@ModelAttribute("removablepartial")
public String removablepartial(
   @RequestParam(value="removablepartial
        ", required = false) String
        removablepartial)
   {
    return removablepartial;
}

@ModelAttribute("rotation")
public String rotation(
   @RequestParam(value="rotation",
        required = false) String
        rotation)
   {
    return rotation;
}

@ModelAttribute("postcorecrown")
public String postcorecrown(
   @RequestParam(value="postcorecrown",
        required = false) String
        postcorecrown)
   {
    return postcorecrown;
}

@ModelAttribute("rootcanal")
public String rootcanal(
   @RequestParam(value="rootcanal",
        required = false) String
        rootcanal)
   {
    return rootcanal;
}

@ModelAttribute("pitandfissure")
public String pitandfissure(
   @RequestParam(value="pitandfissure",
        required = false) String
        pitandfissure)
   {
    return pitandfissure;
}

@ModelAttribute("extracted")
public String extracted(
   @RequestParam(value="extracted",
        required = false) String
        extracted)
   {
    return extracted;
}

@ModelAttribute("missing")
public String missing(
   @RequestParam(value="missing",
        required = false) String missing
        )
   {
    return missing;
}

@ModelAttribute("unerupted")
public String unerupted(
   @RequestParam(value="unerupted",
        required = false) String
        unerupted)
   {
    return unerupted;
}

@ModelAttribute("impacted")
public String impacted(
   @RequestParam(value="impacted",
        required = false) String
        impacted)
   {
    return impacted;
}

@ModelAttribute("porcelainfused")
public String porcelainfused(
```

```
    @RequestParam(value="porcelainfused",
         required = false) String
        porcelainfused)
    {
     return porcelainfused;
}

@ModelAttribute("acryliccrown")
public String acryliccrown(
    @RequestParam(value="acryliccrown",
        required = false) String
        acryliccrown)
    {
     return acryliccrown;
}

@ModelAttribute("metalcrown")
public String metalcrown(
    @RequestParam(value="metalcrown",
        required = false) String
        metalcrown)
    {
     return metalcrown;
}

@ModelAttribute("porcelaincrown")
public String porcelaincrown(
    @RequestParam(value="porcelaincrown",
         required = false) String
        porcelaincrown)
    {
     return porcelaincrown;
}

@ModelAttribute("restorable")
public String restorable(
    @RequestParam(value="restorable",
        required = false) String
        restorable)
    {
     return restorable;
}

@ModelAttribute("nonrestorable")
public String nonrestorable(
    @RequestParam(value="nonrestorable",
        required = false) String
        nonrestorable)
    {
     return nonrestorable;
}

@ModelAttribute("thePatientInfo")
protected ArrayList<PatientInformation>
     getPatientInfo(HttpServletRequest
     request,HttpSession session) throws
         Exception {
 ArrayList<PatientInformation> info =
     null;

 session=request.getSession(false);

 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
  info=querySP.allPatientInfo();

 return info;
}

@ModelAttribute("patient")
protected ArrayList<Patient> getPatients
     (HttpServletRequest request,
     HttpSession session) throws
     Exception {

 session=request.getSession(false);

 ArrayList<Patient> patients=null;


 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
  patients=querySP.allPatients();

 return patients;
}

@ModelAttribute("agefrom")
public String agefrom(
   @RequestParam(value="agefrom",
        required = false) String agefrom
        )
   {
    return agefrom;
}
@ModelAttribute("ageto")
public String ageto(
   @RequestParam(value="ageto", required
        = false) String ageto)
   {
    return ageto;
}
@ModelAttribute("sex")
public String sex(
   @RequestParam(value="sex", required =
        false) String sex)
```

```
    {
     return sex;
}
@ModelAttribute("city")
public String city(
    @RequestParam(value="city", required
        = false) String city)
    {
     return city;
}
@ModelAttribute("occupation")
public String occupation(
    @RequestParam(value="occupation",
         required = false) String
         occupation)
    {
     return occupation;
}

@ModelAttribute("or1")
public String or1(
    @RequestParam(value="or1", required =
          false) String or1)
    {
     return or1;
}
@ModelAttribute("and1")
public String and1(
    @RequestParam(value="and1", required
        = false) String and1)
    {
     return and1;
}
@ModelAttribute("or2")
public String or2(
    @RequestParam(value="or2", required =
          false) String or2)
    {
     return or2;
}
@ModelAttribute("and2")
public String and2(
    @RequestParam(value="and2", required
        = false) String and2)
    {
     return and2;
}

@ModelAttribute("condd")
public String condd(
    @RequestParam(value="condd", required
         = false) String condd)
    {
     return condd;
}
//-QUERIES-//
```

```
protected boolean checkWithinAge(String
     age, HttpServletRequest request)
     throws Exception {
 String agefrom = request.getParameter("
      agefrom");
 String ageto = request.getParameter("
      ageto");
 if(agefrom==null || agefrom.isEmpty())
  agefrom = "0";
 if(ageto==null || ageto.isEmpty())
  ageto = "100";
 int agef = Integer.parseInt(agefrom);
 int aget = Integer.parseInt(ageto);

 if(age==null || age.trim().isEmpty())
  return false;

  int ageCompare = Integer.parseInt(age.
       trim());


 if(agef < ageCompare && ageCompare <
      aget)
  return true;
 else return false;

}
protected boolean checkGender(String
     gender, HttpServletRequest request)
      throws Exception {
 String sex = request.getParameter("sex"
      );
 //String sexF = request.getParameter("
      sexF");
 if(sex == null || sex.contains(",")) {
  return true;
 }
 if(gender== null || gender.trim().
      isEmpty())
  return false;
 if(sex.equalsIgnoreCase(gender.trim())
      || sex.equals("all"))
  return true;
 else return false;
}



protected boolean checkOccupation(String
      occupation, HttpServletRequest
     request) throws Exception {
 String job = request.getParameter("
      occupation");
 if(job.isEmpty())
  return true;
```

```
 if(occupation== null || occupation.trim
     ().isEmpty())
  return false;
 if(occupation.trim().toLowerCase().
     indexOf(job.toLowerCase()) != -1)
  return true;
 else return false;
}



protected boolean checkCity(String
    occupation, HttpServletRequest
    request) throws Exception {
 String city = request.getParameter("
     city");
 if(city.isEmpty())
  return true;
 if(occupation==null || occupation.trim
     ().isEmpty())
  return false;
 if(occupation.trim().toLowerCase().
     indexOf(city.toLowerCase()) != -1)
  return true;
 else return false;
}




protected void addtoList(ArrayList<
    String> list, Patient patient,
    String toothno, HttpServletRequest
    request, HttpSession session)
    throws Exception {
 String cond= request.getParameter("
     condd");
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());

 ArrayList<PatientInformation> info =
     querySP.getPatientInfo(patient.
     getPatient_id());
 String occupation = "";



 if(info.size() != 0){
  occupation = info.get(info.size()-1).
      getOccupation();
  if(occupation==null)
   occupation="";
 }

 if(cond.equalsIgnoreCase("and3")){
  if(checkCity(patient.getCity(),request
      ) && checkOccupation(occupation,
      request) && checkGender(patient.
      getGender(), request) &&
      checkWithinAge(patient.getAge(),
      request)) {
   list.add(String.valueOf(patient.
       getPatient_id()));
   list.add(patient.getGivenName()+" "+
       patient.getMiddleName()+ " "+
       patient.getFamilyName());
   list.add(toothno);
  }
 }
 else if(cond.equalsIgnoreCase("or3")){
   list.add(String.valueOf(patient.
       getPatient_id()));
   list.add(patient.getGivenName()+" "+
       patient.getMiddleName()+ " "+
       patient.getFamilyName());
   list.add(toothno);


  }

}

protected static String join(final
    StringBuffer buff, final Object
    array[],
     final String delim)
  {
     boolean haveDelim = (delim != null
         );

     for (int i = 0; i < array.length;
         i++)
     {
        buff.append(array[i]);

        // if this is the last element
            then don't append delim
        if (haveDelim && (i + 1) <
            array.length)
        {
           buff.append(delim);
        }
     }

     return buff.toString();
  }

  /**
   * Join an array of strings into one
       delimited string.
   *
   * @param array   Array of objects to
       join as strings.
   * @param delim   Delimiter to join
```

```
        strings with or <i>null</i>.
     * @return          Joined string.
     */
    protected String join(final Object
        array[], final String delim)
    {
       return join(new StringBuffer(),
           array, delim);
    }
//
    ------------------------------------------------------------


/*
 * NEW SET
 */

@ModelAttribute("cariespatients")
protected ArrayList<String>
     getCariesPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

 session=request.getSession(false);


 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getCaries()[0]!=0 &&
       charts.get(i).getCaries() != null
       && charts.get(i).getIs_current()
       .equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String caries= this.join(charts.get(i
        ).getCaries(), ",");
   addtoList(list, patient,caries,
        request,session);
  }
 }

 return list;
}

@ModelAttribute("recurrentcariespatients
     ")
protected ArrayList<String>
     getRecurrentcariesPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();


 QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getRecurrentcaries()
       [0]!=0 && charts.get(i).
       getRecurrentcaries() != null &&
       charts.get(i).getIs_current().
       equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String recaries= this.join(charts.get
        (i).getRecurrentcaries(), ",");
   addtoList(list, patient,recaries,
        request,session);
  }
 }

 return list;
}

@ModelAttribute("restorationpatients")
protected ArrayList<String>
     getrestorationcariesPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

 QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getRestoration()
       [0]!=0 && charts.get(i).
       getRestoration() != null &&
       charts.get(i).getIs_current().
```

```
        equals("yes")) {
    Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
    String restoration= this.join(charts.
        get(i).getRestoration(), ",");
    addtoList(list, patient,restoration,
        request,session);
   }
  }

  return list;
 }

/*
 @ModelAttribute("amalgampatients")
 protected ArrayList<String>
     getAmalgamsPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
  ArrayList<String> list = new ArrayList<
      String>();

  boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";
  AdminSP adminSP= new AdminSP();
  QuerySP querySP= new QuerySP();
  if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
   errorMessage="Session has expired. Pls
       log again.";
  else{
   String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
       i++){
    privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "dentalchart", "select");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   querySP.setDatabase_username(
       currentRole);

  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getAmalgam() != null
       && charts.get(i).getIs_current().
       equals("yes")) {
    Patient patient = this.
        getServicesNeeded(request,
        session);
    addtoList(list, patient, charts.get(i
        ).getAmalgam(), request);
   }
  }
 }
  return list;
 }

 @ModelAttribute("compositepatients")
 protected ArrayList<String>
     getCompositePatients(
     HttpServletRequest request) throws
     Exception {
  ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
  ArrayList<String> ArrayList = new
      ArrayList<String>();
  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getComposite() !=
       null && charts.get(i).
       getIs_current().equals("yes")) {
    Patient patient = this.
        getServicesNeeded(request,
        session);
    addtoList(ArrayList, patient, charts.
        get(i).getComposite(), request);
   }
  }
  return ArrayList;
 }

 @ModelAttribute("glassionomerpatients")
 protected ArrayList<String>
     getGlassionomerPatients(
     HttpServletRequest request) throws
     Exception {
  ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
  ArrayList<String> ArrayList = new
      ArrayList<String>();
  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getGlassionomer() !=
       null && charts.get(i).
       getIs_current().equals("yes")) {
    Patient patient = this.
        getServicesNeeded(request,
        session);
    addtoList(ArrayList, patient, charts.
        get(i).getGlassionomer(),
        request);
   }
  }
```

```
 return ArrayList;
}

@ModelAttribute("tempfillingdcpatients")
protected ArrayList<String>
     getTempfillingDCPatients(
     HttpServletRequest request) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> ArrayList = new
      ArrayList<String>();
 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getTempfilling() !=
       null && charts.get(i).
       getIs_current().equals("yes")) {
   Patient patient = this.
        getServicesNeeded(request,
        session);
   addtoList(ArrayList, patient, charts.
        get(i).getTempfilling(), request
        );
  }
 }
 return ArrayList;
}
*/
@ModelAttribute("extrusionpatients")
protected ArrayList<String>
     getExtrusionPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

 QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getExtrusion()[0]!=0
       && charts.get(i).getExtrusion()
       != null && charts.get(i).
       getIs_current().equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String extrusion= this.join(charts.
        get(i).getExtrusion(), ",");
   addtoList(list, patient,extrusion,
        request,session);
  }
 }


 return list;
}

@ModelAttribute("intrusionpatients")
protected ArrayList<String>
     getIntrusionPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getIntrusion()[0]!=0
       && charts.get(i).getIntrusion()
       != null && charts.get(i).
       getIs_current().equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String intrusion= this.join(charts.
        get(i).getIntrusion(), ",");
   addtoList(list, patient,intrusion,
        request,session);
  }
 }

 return list;
}

@ModelAttribute("mesialdriftpatients")
protected ArrayList<String>
     getMesialdriftPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
```

```
        getAttribute("sessionUserRole").
        toString());



   for(int i= 0; i < charts.size(); i++) {
    if(charts.get(i).getMesial_rotation()
        [0]!=0 && charts.get(i).
        getMesial_rotation() != null &&
        charts.get(i).getIs_current().
        equals("yes")) {
     Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
     String mesial_rotation= this.join(
         charts.get(i).getMesial_rotation
         (), ",");
     addtoList(list, patient,
         mesial_rotation, request,session
         );
    }
   }

   return list;
  }

  @ModelAttribute("distaldriftpatients")
  protected ArrayList<String>
      getDistaldriftPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
   ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
   ArrayList<String> list = new ArrayList<
       String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

   for(int i= 0; i < charts.size(); i++) {
    if(charts.get(i).getDistal_rotation()
        [0]!=0 && charts.get(i).
        getDistal_rotation() != null &&
        charts.get(i).getIs_current().
        equals("yes")) {
     Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
     String distal_rotation= this.join(
         charts.get(i).getDistal_rotation
         (), ",");
     addtoList(list, patient,
         distal_rotation, request,session
         );
    }
   }

   return list;
  }

  @ModelAttribute("rotationpatients")
  protected ArrayList<String>
      getRotationPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
   ArrayList<DentalChart> charts =this.
       getDentalChart(request,session);
   ArrayList<String> list = new ArrayList<
       String>();


 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());


   for(int i= 0; i < charts.size(); i++) {
    if(charts.get(i).getRotation()[0]!=0
        && charts.get(i).getRotation() !=
         null && charts.get(i).
        getIs_current().equals("yes")) {
     Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
     String rotation= this.join(charts.get
         (i).getRotation(), ",");
     addtoList(list, patient,rotation,
         request,session);
    }
   }

   return list;
  }

  @ModelAttribute("extractedpatients")
  protected ArrayList<String>
      getExtractedPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
   ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
   ArrayList<String> list = new ArrayList<
       String>();

 QuerySP querySP= new QuerySP();
```

```
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < charts.size(); i++) {
 if(charts.get(i).getExtracted()[0]!=0
     && charts.get(i).getExtracted()
     != null && charts.get(i).
     getIs_current().equals("yes")) {
  Patient patient = querySP.getPatient(
      charts.get(i).getPatient_id());
  String extracted= this.join(charts.
      get(i).getExtracted(), ",");
  addtoList(list, patient,extracted,
      request,session);
 }
}


return list;
}

@ModelAttribute("completedentpatients")
protected ArrayList<String>
    getCompletedentPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
ArrayList<DentalChart> charts = this.
    getDentalChart(request,session);
ArrayList<String> list = new ArrayList<
    String>();

QuerySP querySP= new QuerySP();

querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < charts.size(); i++) {
 if(charts.get(i).getComplete_denture()
     != null && charts.get(i).
     getIs_current().equals("yes")) {
  Patient patient = querySP.getPatient(
      charts.get(i).getPatient_id());
  addtoList(list, patient,charts.get(i)
      .getComplete_denture(), request,
      session);
 }
}

return list;
}

@ModelAttribute("singledentpatients")
protected ArrayList<String>
    getSingledentPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
ArrayList<DentalChart> charts = this.
    getDentalChart(request,session);
ArrayList<String> list = new ArrayList<
    String>();

QuerySP querySP= new QuerySP();

querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < charts.size(); i++) {
 if(charts.get(i).getSingle_denture()
     != null && charts.get(i).
     getIs_current().equals("yes")) {
  Patient patient = querySP.getPatient(
      charts.get(i).getPatient_id());
  addtoList(list, patient, charts.get(i
      ).getSingle_denture(), request,
      session);
 }
}

return list;
}

@ModelAttribute("
    removablepartialdentpatients")
protected ArrayList<String>
    getRemovablepartialdentPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
ArrayList<DentalChart> charts = this.
    getDentalChart(request,session);
ArrayList<String> list = new ArrayList<
    String>();

QuerySP querySP= new QuerySP();

querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < charts.size(); i++) {
 if(charts.get(i).
```

```
        getRemovable_partial_denture()
        [0]!=0 && charts.get(i).
        getRemovable_partial_denture() !=
         null && charts.get(i).
        getIs_current().equals("yes")) {
     Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
     String removable= this.join(charts.
         get(i).
         getRemovable_partial_denture(),
         ",");
     addtoList(list, patient, removable,
         request,session);
    }
   }

   return list;
  }

  @ModelAttribute("pitandfissurepatients")
  protected ArrayList<String>
      getPitandfissurePatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
   ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
   ArrayList<String> list = new ArrayList<
       String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());


   for(int i= 0; i < charts.size(); i++) {
    if(charts.get(i).
        getPitfissure_sealants()[0]!=0 &&
         charts.get(i).
        getPitfissure_sealants() != null
        && charts.get(i).getIs_current().
        equals("yes")) {
     Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
     String pitfissure_sealants= this.join
         (charts.get(i).
         getPitfissure_sealants(), ",");
     addtoList(list, patient,
         pitfissure_sealants, request,
         session);
    }
   }

   return list;
  }

  @ModelAttribute("rootcanalpatients")
  protected ArrayList<String>
      getRootcanalPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
   ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
   ArrayList<String> list = new ArrayList<
       String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());


   for(int i= 0; i < charts.size(); i++) {
    if(charts.get(i).
        getRootcanal_treatment()[0]!=0 &&
         charts.get(i).
        getRootcanal_treatment() != null
        && charts.get(i).getIs_current().
        equals("yes")) {
     Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
     String rootcanal= this.join(charts.
         get(i).getRootcanal_treatment(),
          ",");
     addtoList(list, patient,rootcanal,
         request,session);
    }
   }

   return list;
  }

  @ModelAttribute("uneruptedpatients")
  protected ArrayList<String>
      getUneruptedPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
   ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
   ArrayList<String> list = new ArrayList<
       String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
```

```
  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getUnerupted()[0]!=0
       && charts.get(i).getUnerupted()
       != null && charts.get(i).
       getIs_current().equals("yes")) {
    Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
    String unerupted= this.join(charts.
        get(i).getUnerupted(), ",");
    addtoList(list, patient, unerupted,
        request,session);
   }
  }

  return list;
 }

 @ModelAttribute("impactedpatients")
 protected ArrayList<String>
     getImpactedPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
  ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getImpacted()[0]!=0
       && charts.get(i).getImpacted() !=
        null && charts.get(i).
       getIs_current().equals("yes")) {
    Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
    String impacted= this.join(charts.get
        (i).getImpacted(), ",");
    addtoList(list, patient, impacted,
        request,session);
   }
  }


  return list;
 }

 @ModelAttribute("missingpatients")
 protected ArrayList<String>
     getMissingPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
  ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getMissing()[0]!=0&&
       charts.get(i).getMissing() !=
       null && charts.get(i).
       getIs_current().equals("yes")) {
    Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
    String missing= this.join(charts.get(
        i).getMissing(), ",");
    addtoList(list, patient, missing,
        request,session);
   }
  }

  return list;
 }

 @ModelAttribute("acryliccrownpatients")
 protected ArrayList<String>
     getAcryliccrownPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
  ArrayList<String> list = new ArrayList<
      String>();

  QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
```

```
 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getAcrylic_crown()
       [0]!=0 && charts.get(i).
       getAcrylic_crown() != null &&
       charts.get(i).getIs_current().
       equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String acrylic_crown= this.join(
        charts.get(i).getAcrylic_crown()
        , ",");
   addtoList(list, patient,
        acrylic_crown, request,session);
  }
 }

 return list;
}

@ModelAttribute("metalcrownpatients")
protected ArrayList<String>
     getMetalcrownPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getMetal_crown()
       [0]!=0 && charts.get(i).
       getMetal_crown() != null &&
       charts.get(i).getIs_current().
       equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String metal_crown= this.join(charts.
        get(i).getMetal_crown(), ",");
   addtoList(list, patient, metal_crown,
        request,session);
  }
 }

 return list;
}
```

```
@ModelAttribute("postcorecrownpatients")
protected ArrayList<String>
     getPostcorecrownPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();
QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 for(int i= 0; i < charts.size(); i++) {
  if(charts.get(i).getPostcore_crown()
       [0]!=0 && charts.get(i).
       getPostcore_crown() != null &&
       charts.get(i).getIs_current().
       equals("yes")) {
   Patient patient = querySP.getPatient(
        charts.get(i).getPatient_id());
   String postcore_crown= this.join(
        charts.get(i).getPostcore_crown
        (), ",");
   addtoList(list, patient,
        postcore_crown, request,session)
        ;
  }
 }

 return list;
}

@ModelAttribute("porcelaincrownpatients"
     )
protected ArrayList<String>
     getPorcelaincrownPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 ArrayList<DentalChart> charts = this.
      getDentalChart(request,session);
 ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
```

```
  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getPorcelain_crown()
        [0]!=0 && charts.get(i).
        getPorcelain_crown() != null &&
        charts.get(i).getIs_current().
        equals("yes")) {
    Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
    String porcelian_crown= this.join(
         charts.get(i).getPorcelain_crown
         (), ",");
    addtoList(list, patient,
         porcelian_crown, request,session
         );
   }
  }

  return list;
 }

 @ModelAttribute("porcelainfusedpatients"
     )
 protected ArrayList<String>
     getPorcelainfusedPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
  ArrayList<String> list = new ArrayList<
       String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());



  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getPorcelain_infused
        ()[0]!=0 && charts.get(i).
        getPorcelain_infused() != null &&
         charts.get(i).getIs_current().
        equals("yes")) {
    Patient patient = querySP.getPatient(
         charts.get(i).getPatient_id());
    String porcelian_infused= this.join(
         charts.get(i).
         getPorcelain_infused(), ",");
    addtoList(list, patient,
         porcelian_infused, request,
         session);
   }
  }

  return list;
 }
/*
 @ModelAttribute("rsetorablepatients")
 protected ArrayList<String>
      getRestorablePatients(
      HttpServletRequest request) throws
      Exception {
  ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
  ArrayList<String> ArrayList = new
       ArrayList<String>();
  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getRestorable() !=
        null && charts.get(i).
        getIs_current().equals("yes")) {
    Patient patient = this.
         getServicesNeeded(request,
         session);
    addtoList(ArrayList, patient, charts.
         get(i).getRestorable(), request)
         ;
   }
  }
  return ArrayList;
 }

 @ModelAttribute("nonrestorablepatients")
 protected ArrayList<String>
      getNonrestorablePatients(
      HttpServletRequest request) throws
      Exception {
  ArrayList<DentalChart> charts = this.
       getDentalChart(request,session);
  ArrayList<String> ArrayList = new
       ArrayList<String>();
  for(int i= 0; i < charts.size(); i++) {
   if(charts.get(i).getNonrestorable() !=
        null && charts.get(i).
        getIs_current().equals("yes")) {
    Patient patient = this.
         getServicesNeeded(request,
         session);
    addtoList(ArrayList, patient, charts.
         get(i).getNonrestorable(),
         request);
   }
  }
  return ArrayList;
 }
*/
 //
```

```
 @ModelAttribute("periodonticspatients")
 protected ArrayList<String>
      getPeriodonticsPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  ArrayList<ServiceNeeded> services =
       this.getServicesNeeded(request,
       session);
  ArrayList<String> list = new ArrayList<
       String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());



  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).getPeriodontics()
        != null && services.get(i).
        getIs_current().equals("yes")) {
    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
    addtoList(list, patient, "", request,
         session);
   }
  }

  return list;
 }

 @ModelAttribute("class1patients")
 protected ArrayList<String>
      getClass1Patients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  ArrayList<ServiceNeeded> services =
       this.getServicesNeeded(request,
       session);
  ArrayList<String> list = new ArrayList<
       String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());



  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).getClass_1()[0]!=0
       && services.get(i).getClass_1()
       != null && services.get(i).
       getIs_current().equals("yes")) {
    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
    String class_1= this.join(services.
         get(i).getClass_1(), ",");
    addtoList(list, patient, class_1,
         request,session);
   }
  }


  return list;
 }

 @ModelAttribute("class2patients")
 protected ArrayList<String>
      getClass2Patients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  ArrayList<ServiceNeeded> services =
       this.getServicesNeeded(request,
       session);
  ArrayList<String> list = new ArrayList<
       String>();
QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());



  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).getClass_2()[0]!=0
       && services.get(i).getClass_2()
       != null && services.get(i).
       getIs_current().equals("yes")) {
    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
    String class_2= this.join(services.
         get(i).getClass_2(), ",");
    addtoList(list, patient, class_2,
         request,session);
   }
  }

  return list;
```

```
}

@ModelAttribute("class3patients")
protected ArrayList<String>
    getClass3Patients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 ArrayList<ServiceNeeded> services =
     this.getServicesNeeded(request,
     session);
 ArrayList<String> list = new ArrayList<
     String>();
QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());


 for(int i= 0; i < services.size(); i++)
     {


  if(services.get(i).getClass_3()[0]!=0
      && services.get(i).getClass_3()
      != null && services.get(i).
      getIs_current().equals("yes")) {
   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
   String class_3= this.join(services.
       get(i).getClass_3(), ",");
   addtoList(list, patient, class_3,
       request,session);
  }
 }

 return list;
}

@ModelAttribute("class4patients")
protected ArrayList<String>
    getClass4Patients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 ArrayList<ServiceNeeded> services =
     this.getServicesNeeded(request,
     session);
 ArrayList<String> list = new ArrayList<
     String>();
QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());


 for(int i= 0; i < services.size(); i++)
     {
  if(services.get(i).getClass_4()[0]!=0
      && services.get(i).getClass_4()
      != null && services.get(i).
      getIs_current().equals("yes")) {
   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
   String class_4= this.join(services.
       get(i).getClass_4(), ",");
   addtoList(list, patient, class_4,
       request,session);
  }
 }

 return list;
}

@ModelAttribute("class5patients")
protected ArrayList<String>
    getClass5Patients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 ArrayList<ServiceNeeded> services =
     this.getServicesNeeded(request,
     session);
 ArrayList<String> list = new ArrayList<
     String>();

QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());


 for(int i= 0; i < services.size(); i++)
     {
  if(services.get(i).getClass_5()[0]!=0
      && services.get(i).getClass_5()
      != null && services.get(i).
      getIs_current().equals("yes")) {
   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
   String class_5= this.join(services.
       get(i).getClass_4(), ",");
   addtoList(list, patient, class_5,
```

```
            request,session);
    }
   }

   return list;
  }

  @ModelAttribute("onlaypatients")
  protected ArrayList<String>
       getOnlayPatients(HttpServletRequest
        request,HttpSession session)
       throws Exception {
   ArrayList<ServiceNeeded> services =
        this.getServicesNeeded(request,
        session);
   ArrayList<String> list = new ArrayList<
        String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());



   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getOnlay()[0]!=0 &&
         services.get(i).getOnlay() !=
         null && services.get(i).
         getIs_current().equals("yes")) {
     Patient patient = querySP.getPatient(
          services.get(i).getPatient_id())
          ;
     String onlay= this.join(services.get(
          i).getOnlay(), ",");
     addtoList(list, patient,onlay,
          request,session);
    }
   }

   return list;
  }

  @ModelAttribute("extraction5patients")
  protected ArrayList<String>
       getExtractionPatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
   ArrayList<ServiceNeeded> services =
        this.getServicesNeeded(request,
        session);
   ArrayList<String> list = new ArrayList<
        String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());



   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getExtraction()
         [0]!=0 && services.get(i).
         getExtraction() != null &&
         services.get(i).getIs_current().
         equals("yes")) {
     Patient patient = querySP.getPatient(
          services.get(i).getPatient_id())
          ;
     String extraction= this.join(services
          .get(i).getExtraction(), ",");
     addtoList(list, patient, extraction,
          request,session);
    }
   }

   return list;
  }

  @ModelAttribute("odontectomypatients")
  protected ArrayList<String>
       getOdontectomyPatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
   ArrayList<ServiceNeeded> services =
        this.getServicesNeeded(request,
        session);
   ArrayList<String> list = new ArrayList<
        String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());



   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getOdontectomy()
         [0]!=0 && services.get(i).
         getOdontectomy() != null &&
         services.get(i).getIs_current().
         equals("yes")) {
```

```
    Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    String odontectomy= this.join(
        services.get(i).getOdontectomy()
        , ",");
    addtoList(list, patient, odontectomy,
         request,session);
   }
  }

  return list;
 }

 @ModelAttribute("specialcasepatients")
 protected ArrayList<String>
     getSpecialcasePatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<ServiceNeeded> services =
      this.getServicesNeeded(request,
      session);
  ArrayList<String> list = new ArrayList<
      String>();
QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


  for(int i= 0; i < services.size(); i++)
      {
   if(services.get(i).getSpecial_case()
       [0]!=0 && services.get(i).
       getSpecial_case() != null &&
       services.get(i).getIs_current().
       equals("yes")) {
    Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    String special_case= this.join(
        services.get(i).getSpecial_case
        (), ",");
    addtoList(list, patient, special_case
        , request,session);
   }
  }

  return list;
 }

 @ModelAttribute("pulpsedationpatients")
 protected ArrayList<String>
     getPulpsedationPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<ServiceNeeded> services =
      this.getServicesNeeded(request,
      session);
  ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


  for(int i= 0; i < services.size(); i++)
      {
   if(services.get(i).getPulp_sedation()
       [0]!=0 && services.get(i).
       getPulp_sedation() != null &&
       services.get(i).getIs_current().
       equals("yes")) {
    Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    String pulp_sedation= this.join(
        services.get(i).getPulp_sedation
        (), ",");
    addtoList(list, patient,
        pulp_sedation, request,session);
   }
  }

  return list;
 }

 @ModelAttribute("
     crownrecementationpatients")
 protected ArrayList<String>
     getCrownrecementationPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  ArrayList<ServiceNeeded> services =
      this.getServicesNeeded(request,
      session);
  ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
```

```
  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).
        getCrown_recementation()[0]!=0 &&
         services.get(i).
        getCrown_recementation() != null
        && services.get(i).getIs_current
        ().equals("yes")) {
    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
    String crown_recementation= this.join
         (services.get(i).
         getCrown_recementation(), ",");
    addtoList(list, patient,
         crown_recementation, request,
         session);
   }
  }

  return list;
 }

 @ModelAttribute("tempfillingpatients")
 protected ArrayList<String>
      getTempfillingPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  ArrayList<ServiceNeeded> services =
       this.getServicesNeeded(request,
       session);
  ArrayList<String> list = new ArrayList<
       String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());




  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).getFilling_service
        ()[0]!=0 && services.get(i).
        getFilling_service() != null &&
        services.get(i).getIs_current().
        equals("yes")) {
    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
    String filling_service= this.join(
         services.get(i).
         getFilling_service(), ",");
    addtoList(list, patient,
         filling_service, request,session
         );
   }
  }

  return list;
 }

 @ModelAttribute("laminatedpatients")
 protected ArrayList<String>
      getLaminatedPatients(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  ArrayList<ServiceNeeded> services =
       this.getServicesNeeded(request,
       session);
  ArrayList<String> list = new ArrayList<
       String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());




  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).getLaminated()
        [0]!=0 && services.get(i).
        getLaminated() != null &&
        services.get(i).getIs_current().
        equals("yes")) {
    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
    String laminated= this.join(services.
         get(i).getLaminated(), ",");
    addtoList(list, patient, laminated,
         request,session);
   }
  }

  return list;
 }

 @ModelAttribute("singlecrownpatients")
 protected ArrayList<String>
```

```
        getSinglecrownPatients(
        HttpServletRequest request,
        HttpSession session) throws
        Exception {
    ArrayList<ServiceNeeded> services =
         this.getServicesNeeded(request,
         session);
    ArrayList<String> list = new ArrayList<
         String>();
QuerySP querySP= new QuerySP();

    querySP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());



    for(int i= 0; i < services.size(); i++)
         {
     if(services.get(i).getSingle_crown()
         [0]!=0 && services.get(i).
         getSingle_crown() != null &&
         services.get(i).getIs_current().
         equals("yes")) {
      Patient patient = querySP.getPatient(
           services.get(i).getPatient_id())
           ;
      String single_crown= this.join(
           services.get(i).getSingle_crown
           (), ",");
      addtoList(list, patient, single_crown
           , request,session);
     }
    }

    return list;
  }

  @ModelAttribute("bridgeservicepatients")
  protected ArrayList<String>
       getBridgeservicePatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
    ArrayList<ServiceNeeded> services =
         this.getServicesNeeded(request,
         session);
    ArrayList<String> list = new ArrayList<
         String>();

QuerySP querySP= new QuerySP();

    querySP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());



    for(int i= 0; i < services.size(); i++)
         {
     if(services.get(i).getBridge_service()
         [0]!=0 && services.get(i).
         getBridge_service() != null &&
         services.get(i).getIs_current().
         equals("yes")) {
      Patient patient = querySP.getPatient(
           services.get(i).getPatient_id())
           ;
      String bridge_service= this.join(
           services.get(i).
           getBridge_service(), ",");
      addtoList(list, patient,
           bridge_service, request,session)
           ;
     }
    }

    return list;
  }

  @ModelAttribute("anteriorpatients")
  protected ArrayList<String>
       getAnteriorPatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
    ArrayList<ServiceNeeded> services =
         this.getServicesNeeded(request,
         session);
    ArrayList<String> list = new ArrayList<
         String>();

QuerySP querySP= new QuerySP();

    querySP.setDatabase_username(session.
         getAttribute("sessionUserRole").
         toString());



    for(int i= 0; i < services.size(); i++)
         {
     if(services.get(i).getAnterior()[0]!=0
          && services.get(i).getAnterior()
          != null && services.get(i).
         getIs_current().equals("yes")) {
      Patient patient = querySP.getPatient(
           services.get(i).getPatient_id())
           ;
      String anterior= this.join(services.
           get(i).getAnterior(), ",");
      addtoList(list, patient, anterior,
           request,session);
```

```
    }
   }

   return list;
  }

  @ModelAttribute("posteriorpatients")
  protected ArrayList<String>
       getPosteriorPatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
   ArrayList<ServiceNeeded> services =
        this.getServicesNeeded(request,
        session);
   ArrayList<String> list = new ArrayList<
        String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());

   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getPosterior()
         [0]!=0 && services.get(i).
         getPosterior() != null &&
         services.get(i).getIs_current().
         equals("yes")) {
     Patient patient = querySP.getPatient(
          services.get(i).getPatient_id())
          ;
     String posterior= this.join(services.
          get(i).getPosterior(), ",");
     addtoList(list, patient, posterior,
          request,session);
    }
   }

   return list;
  }

  @ModelAttribute("pedodonticspatients")
  protected ArrayList<String>
       getPedodonticsPatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
   ArrayList<ServiceNeeded> services =
        this.getServicesNeeded(request,
        session);
   ArrayList<String> list = new ArrayList<
        String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());


   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getSurgery() !=
         null) {
     if(services.get(i).getSurgery().
          contains("pedodontics") &&
          services.get(i).getIs_current().
          equals("yes")) {
      Patient patient = querySP.getPatient
           (services.get(i).getPatient_id
           ());
      addtoList(list, patient, "", request
           ,session);
     }
    }
   }

   return list;
  }

  @ModelAttribute("orthodonticspatients")
  protected ArrayList<String>
       getOrthodonticsPatients(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
   ArrayList<ServiceNeeded> services =
        this.getServicesNeeded(request,
        session);
   ArrayList<String> list = new ArrayList<
        String>();

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());


   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getSurgery() !=
         null) {
     if(services.get(i).getSurgery().
          contains("orthodontics") &&
          services.get(i).getIs_current().
          equals("yes")) {
      Patient patient = querySP.getPatient
```

```
            (services.get(i).getPatient_id
            ());
        addtoList(list, patient, "", request
            , session);
      }
    }
  }

  return list;
}

@ModelAttribute("acuteinfectionspatients
    ")
protected ArrayList<String>
    getAcuteinfectionsPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
  ArrayList<ServiceNeeded> services =
      this.getServicesNeeded(request,
      session);
  ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



  for(int i= 0; i < services.size(); i++)
      {
    if(services.get(i).
        getEmergency_treatment() != null)
        {
      if(services.get(i).
          getEmergency_treatment().
          contains("acute_infections") &&
          services.get(i).getIs_current().
          equals("yes")) {
        Patient patient = querySP.getPatient
            (services.get(i).getPatient_id
            ());
        addtoList(list, patient, "", request
            , session);
      }
    }
  }

  return list;
}

@ModelAttribute("
    traumaticinjuriespatients")
protected ArrayList<String>
    getTraumaticinjuriesPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
  ArrayList<ServiceNeeded> services =
      this.getServicesNeeded(request,
      session);
  ArrayList<String> list = new ArrayList<
      String>();


QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



  for(int i= 0; i < services.size(); i++)
      {
    if(services.get(i).
        getEmergency_treatment() != null)
        {
      if(services.get(i).
          getEmergency_treatment().
          contains("traumatic_injuries")
          && services.get(i).getIs_current
          ().equals("yes")) {
        Patient patient = querySP.getPatient
            (services.get(i).getPatient_id
            ());
        addtoList(list, patient, "", request
            , session);
      }
    }
  }

  return list;
}

@ModelAttribute("completedenturepatients
    ")
protected ArrayList<String>
    getCompletedenturePatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
  ArrayList<ServiceNeeded> services =
      this.getServicesNeeded(request,
      session);
  ArrayList<String> list = new ArrayList<
      String>();

QuerySP querySP= new QuerySP();
```

```
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < services.size(); i++)
    {
 if(services.get(i).getProsthodontics()
    != null) {
  if(services.get(i).getProsthodontics
    ().contains("complete_denture")
    && services.get(i).getIs_current
    ().equals("yes")) {
   Patient patient = querySP.getPatient
    (services.get(i).getPatient_id
    ());
   addtoList(list, patient, "", request
    ,session);
  }
 }
}

return list;
}

@ModelAttribute("singledenturepatients")
protected ArrayList<String>
    getSingledenturePatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
ArrayList<ServiceNeeded> services =
    this.getServicesNeeded(request,
    session);
ArrayList<String> list = new ArrayList<
    String>();

QuerySP querySP= new QuerySP();

querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < services.size(); i++)
    {
 if(services.get(i).getProsthodontics()
    != null) {
  if(services.get(i).getProsthodontics
    ().contains("single_denture") &&
     services.get(i).getIs_current()
    .equals("yes")) {
   Patient patient = querySP.getPatient
    (services.get(i).getPatient_id
    ());
   addtoList(list, patient, "", request
    ,session);
  }
 }
}

return list;
}

@ModelAttribute("
    removablepartialpatients")
protected ArrayList<String>
    getRemovablepartialPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
ArrayList<ServiceNeeded> services =
    this.getServicesNeeded(request,
    session);
ArrayList<String> list = new ArrayList<
    String>();

QuerySP querySP= new QuerySP();

querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



for(int i= 0; i < services.size(); i++)
    {
 if(services.get(i).getProsthodontics()
    != null) {
  if(services.get(i).getProsthodontics
    ().contains("removable_partial")
    && services.get(i).
    getIs_current().equals("yes")) {
   Patient patient = querySP.getPatient
    (services.get(i).getPatient_id
    ());
   addtoList(list, patient, "", request
    ,session);
  }
 }
}



return list;
}

@ModelAttribute("agegrouppatients")
protected List<String>
    getAgeGroupPatients(
```

```
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<String> list = new ArrayList<
    String>();
List<Patient> patients = this.
    getPatients(request,session);


QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



for(int i = 0; i < patients.size(); i
    ++) {
 ArrayList<PatientInformation> info =
     querySP.getPatientInfo(patients.
     get(i).getPatient_id());
 String complaint = "";
 String occupation = "";
 System.out.println("BOOM patient:"+
     patients.get(i).getPatient_id());
 System.out.println("BOOM :"+info.size
     ());


 if(info.size() != 0){
  occupation = info.get(info.size()-1).
      getOccupation();
  if(occupation==null)
   occupation="";
 }

 if(checkCity(patients.get(i).getCity()
     ,request) && checkOccupation(
     occupation,request) &&
     checkGender(patients.get(i).
     getGender(), request) &&
     checkWithinAge(patients.get(i).
     getAge(), request)) {
  list.add(String.valueOf(patients.get(
      i).getPatient_id()));
  list.add(patients.get(i).getGivenName
      ()+" "+ patients.get(i).
      getMiddleName()+ " "+patients.
      get(i).getFamilyName());
  list.add(patients.get(i).getAge().
      toString());
 }
 }
 return list;
}
@ModelAttribute("getPatientOrAnd")
protected List<String> getPatientsOrAnd(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<String> list = new ArrayList<
     String>();
 List<Patient> patients = this.
     getPatients(request,session);


 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());



 for(int i = 0; i < patients.size(); i
     ++) {
  ArrayList<PatientInformation> info =
      querySP.getPatientInfo(patients.
      get(i).getPatient_id());
  String complaint = "";
  String occupation = "";
  System.out.println("BOOM patient:"+
      patients.get(i).getPatient_id());
  System.out.println("BOOM :"+info.size
      ());


  if(info.size() != 0){
   occupation = info.get(info.size()-1).
       getOccupation();
   if(occupation==null)
    occupation="";
  }

  String cond= request.getParameter("
      condd");
  System.out.println(cond);
  if(cond.equalsIgnoreCase("and3")){
  if(checkCity(patients.get(i).getCity()
      ,request) && checkOccupation(
      occupation,request) &&
      checkGender(patients.get(i).
      getGender(), request) &&
      checkWithinAge(patients.get(i).
      getAge(), request)) {
   list.add(String.valueOf(patients.get(
       i).getPatient_id()));
```

```
    list.add(patients.get(i).getGivenName
        ()+" "+ patients.get(i).
        getMiddleName()+ " "+patients.
        get(i).getFamilyName());
    list.add(patients.get(i).getAge().
        toString());
   }
   }
  else if(cond.equalsIgnoreCase("or3")){
    list.add(String.valueOf(patients.get(
        i).getPatient_id()));
    list.add(patients.get(i).getGivenName
        ()+" "+ patients.get(i).
        getMiddleName()+ " "+patients.
        get(i).getFamilyName());
    list.add("0");

   }

  }
  return list;
}


/*
 *
 * RESULTS
 *
 */


@ModelAttribute("results")
protected List<String>
     getResultsPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<String> glist = getPatientsOrAnd(
      request,session);



 int i = 0;
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 while(i < glist.size()) {
  if(i % 3 == 0) {
   List<DentalChart> chart = querySP.
        getDentalChart(Integer.parseInt(
        glist.get(i)));
   String caries = request.getParameter(
        "caries");
   if(caries == null) caries="no";
   String recurrent = request.
        getParameter("recurrentcaries");
   if(recurrent == null) recurrent="no";
   String restoration = request.
        getParameter("restoration");
   if(restoration == null) restoration="
        no";
   String extrusion = request.
        getParameter("extrusion");
   if(extrusion == null) extrusion="no";
   String intrusion = request.
        getParameter("intrusion");
   if(intrusion == null) intrusion="no";
   String mesial = request.getParameter(
        "mesialdrift");
   if(mesial == null) mesial="no";
   String distal = request.getParameter(
        "distaldrift");
   if(distal == null) distal="no";
   String rotation = request.
        getParameter("rotation");
   if(rotation == null) rotation="no";
   String extracted = request.
        getParameter("extracted");
   if(extracted == null) extracted="no";
   String complete = request.
        getParameter("completedenture");
   if(complete == null) complete="no";
   String single = request.getParameter(
        "singledenture");
   if(single == null) single="no";
   String removable = request.
        getParameter("removablepartial")
        ;
   if(removable == null) removable="no";
   String pitandfissure = request.
        getParameter("pitandfissure");
   if(pitandfissure == null)
        pitandfissure="no";
   String missing = request.getParameter
        ("missing");
   if(missing == null) missing="no";
   String rootcanal = request.
        getParameter("rootcanal");
   if(rootcanal == null) rootcanal="no";
   String impacted = request.
        getParameter("impacted");
   if(impacted == null) impacted="no";
   String postcore = request.
        getParameter("postcorecrown");
   if(postcore == null) postcore="no";
   String acrylic = request.getParameter
        ("acryliccrown");
   if(acrylic == null) acrylic="no";
   String metal = request.getParameter("
        metalcrown");
   if(metal == null) metal="no";
```

```
String porcelain = request.
    getParameter("porcelaincrown");
if(porcelain == null) porcelain="no";
String unerupted = request.
    getParameter("unerupted");
if(unerupted == null) unerupted="no";
String porcelainfused = request.
    getParameter("porcelainfused");
if(porcelainfused == null)
    porcelainfused="no";
boolean valid = true;
for(int j = 0; j < chart.size(); j++)
    {
 if(chart.get(j).getIs_current().
     equals("yes")){
  if((chart.get(j).getCaries()[0] ==
      0 && caries.equals("yes")) ||
      (chart.get(j).
      getRecurrentcaries()[0] == 0
      && recurrent.equals("yes")) ||
       (chart.get(j).getRestoration
      () == null && restoration.
      equals("yes"))
    || (chart.get(j).getExtrusion()
        [0] == 0 && extrusion.equals
        ("yes")) || (chart.get(j).
        getIntrusion()[0] == 0 &&
        intrusion.equals("yes"))
    || (chart.get(j).
        getMesial_rotation()[0] == 0
        && mesial.equals("yes")) ||
        (chart.get(j).
        getDistal_rotation()[0] == 0
        && distal.equals("yes"))
    || (chart.get(j).getRotation()[0]
        == 0 && rotation.equals("
        yes")) || (chart.get(j).
        getExtracted()[0] == 0 &&
        extracted.equals("yes"))
    || (chart.get(j).getImpacted()[0]
        == 0 && impacted.equals("
        yes")) || (chart.get(j).
        getMissing()[0] == 0 &&
        missing.equals("yes"))
    || (chart.get(j).
        getRootcanal_treatment()[0]
        == 0 && rootcanal.equals("
        yes")) || (chart.get(j).
        getPitfissure_sealants()[0]
        == 0 && pitandfissure.equals
        ("yes"))
    || (chart.get(j).
        getPostcore_crown()[0] == 0
        && postcore.equals("yes"))
        || (chart.get(j).
        getAcrylic_crown()[0] == 0
        && acrylic.equals("yes"))
    || (chart.get(j).getMetal_crown()
        [0] == 0 && metal.equals("
        yes")) || (chart.get(j).
        getPorcelain_crown()[0] == 0
        && porcelain.equals("yes"))
    || (chart.get(j).getUnerupted()
        [0] == 0 && unerupted.equals
        ("yes")) || (chart.get(j).
        getPorcelain_infused()[0] ==
        0 && porcelainfused.equals(
        "yes"))
    || (chart.get(j).
        getComplete_denture() ==
        null && complete.equals("yes
        ")) || (chart.get(j).
        getSingle_denture() == null
        && single.equals("yes")
    || (chart.get(j).
        getRemovable_partial_denture
        ()[0] == 0 && removable.
        equals("yes")))) {
      valid = false;
     }
    }
   }
   if(valid) {
    i = i+3;
   }
   else {
    glist.remove(i);
    glist.remove(i);
    glist.remove(i);
   }
  }

 }
 String cond= request.getParameter("
     condd");
 System.out.println(cond);
 return glist;
}

@ModelAttribute("results2")
protected List<String>
    getResultsPatients2(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<String> glist = getPatientsOrAnd(
     request,session);
 int i = 0;
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());
```

```
while(i < glist.size()) {
 if(i % 3 == 0) {
  System.out.println("Patient; "+ glist
      .get(i));
  List<ServiceNeeded> service =querySP.
      getServiceNeeded(Integer.
      parseInt(glist.get(i)));
  String periodontics = request.
      getParameter("periodontics");
  if(periodontics == null) periodontics
      ="no";
  String class1 = request.getParameter(
      "class1");
  if(class1 == null) class1="no";
  String class2 = request.getParameter(
      "class2");
  if(class2 == null) class2="no";
  String class3 = request.getParameter(
      "class3");
  if(class3 == null) class3="no";
  String class4 = request.getParameter(
      "class4");
  if(class4 == null) class4="no";
  String class5 = request.getParameter(
      "class5");
  if(class5 == null) class5="no";
  String onlay = request.getParameter("
      onlay");
  if(onlay == null) onlay="no";
  String extraction = request.
      getParameter("extraction");
  if(extraction == null) extraction="no
      ";
  String odontectomy = request.
      getParameter("odontectomy");
  if(odontectomy == null) odontectomy="
      no";
  String specialcase = request.
      getParameter("specialcase");
  if(specialcase == null) specialcase="
      no";
  String pedodontics = request.
      getParameter("pedodontics");
  if(pedodontics == null) pedodontics="
      no";
  String orthodontics = request.
      getParameter("orthodontics");
  if(orthodontics == null) orthodontics
      ="no";
  String pulpsedation = request.
      getParameter("pulpsedation");
  if(pulpsedation == null) pulpsedation
      ="no";
  String crownrecementation = request.
      getParameter("crownrecementation
      ");
  if(crownrecementation == null)
      crownrecementation="no";
  String tempfillingservice = request.
      getParameter("tempfillingservice
      ");
  if(tempfillingservice == null)
      tempfillingservice="no";
  String acuteinfections = request.
      getParameter("acuteinfections");
  if(acuteinfections == null)
      acuteinfections="no";
  String traumaticinjuries = request.
      getParameter("traumaticinjuries"
      );
  if(traumaticinjuries == null)
      traumaticinjuries="no";
  String laminated = request.
      getParameter("laminated");
  if(laminated == null) laminated="no";
  String singlecrown = request.
      getParameter("singlecrown");
  if(singlecrown == null) singlecrown="
      no";
  String bridgeservice = request.
      getParameter("bridgeservice");
  if(bridgeservice == null)
      bridgeservice="no";
  String anterior = request.
      getParameter("anterior");
  if(anterior == null) anterior="no";
  String posterior = request.
      getParameter("posterior");
  if(posterior == null) posterior="no";
  String completedentservice = request.
      getParameter("
      completedentservice");
  if(completedentservice == null)
      completedentservice="no";
  String singledentservice = request.
      getParameter("singledentservice"
      );
  if(singledentservice == null)
      singledentservice="no";
  String removablepartialservice =
      request.getParameter("
      removablepartialservice");
  if(removablepartialservice == null)
      removablepartialservice="no";
  boolean valid = true;
  for(int j = 0; j < service.size(); j
      ++) {
   if(service.get(j).getIs_current().
       equals("yes")){
    if((service.get(j).getPeriodontics
        () == null && periodontics.
        equals("yes")) || (service.get
```

```
(j).getClass_1()[0] == 0 &&
class1.equals("yes"))
|| (service.get(j).getClass_2()
[0] == 0 && class2.equals("
yes")) || (service.get(j).
getClass_3()[0] == 0 &&
class3.equals("yes"))
|| (service.get(j).getClass_4()
[0] == 0 && class4.equals("
yes")) || (service.get(j).
getClass_5()[0] == 0 &&
class5.equals("yes"))
|| (service.get(j).getOnlay()[0]
== 0 && onlay.equals("yes"))
|| (service.get(j).
getSpecial_case()[0] == 0 &&
specialcase.equals("yes"))
|| (service.get(j).getExtraction
()[0] == 0 && extraction.
equals("yes")) || (service.
get(j).getOdontectomy()[0]
== 0 && odontectomy.equals("
yes"))
|| (service.get(j).getSurgery()
== null && pedodontics.
equals("yes")) || (service.
get(j).getSurgery() == null
&& orthodontics.equals("yes"
))
|| (service.get(j).
getEmergency_treatment() ==
null && acuteinfections.
equals("yes")) || (service.
get(j).
getEmergency_treatment() ==
null && traumaticinjuries.
equals("yes"))
|| (service.get(j).
getPulp_sedation()[0] == 0
&& pulpsedation.equals("yes"
)) || (service.get(j).
getCrown_recementation()[0]
== 0 && crownrecementation.
equals("yes"))
|| (service.get(j).
getFilling_service()[0] == 0
&& tempfillingservice.
equals("yes")) || (service.
get(j).getLaminated()[0] ==
0 && laminated.equals("yes")
)
|| (service.get(j).
getSingle_crown()[0] == 0 &&
singlecrown.equals("yes"))
|| (service.get(j).
getBridge_service()[0] == 0
&& bridgeservice.equals("yes
"))
|| (service.get(j).
getProsthodontics() == null
&& completedentservice.
equals("yes")) || (service.
get(j).getProsthodontics()
== null && singledentservice
.equals("yes"))
|| (service.get(j).
getProsthodontics() == null
&& removablepartialservice.
equals("yes")) || (service.
get(j).getAnterior()[0] == 0
&& anterior.equals("yes")
|| (service.get(j).getPosterior()
[0] == 0 && posterior.equals
("yes")))) {
valid = false;
}
}
}
if(valid) i++;
else {
glist.remove(i);
glist.remove(i);
glist.remove(i);
}
}
else i++;

}
String cond= request.getParameter("
condd");
System.out.println(cond);
return glist;
}
}
```

Listing 32: section

```
package com.dentist.version.three.form;


//table object
public class Section {
private int section_id;
private String section_name;
private String section_description;
private String created_by;
private String created_date;
public void setSection_id(int section_id
) {
```

```
 this.section_id = section_id;
}
public int getSection_id() {
 return section_id;
}
public void setSection_name(String
     section_name) {
 this.section_name = section_name;
}
public String getSection_name() {
 return section_name;
}
public void setSection_description(
     String section_description) {
 this.section_description =
      section_description;
}
public String getSection_description() {
 return section_description;
}
public void setCreated_by(String
     created_by) {
 this.created_by = created_by;
}
public String getCreated_by() {
 return created_by;
}
public void setCreated_date(String
     created_date) {
 this.created_date = created_date;
}
public String getCreated_date() {
 return created_date;
}

}
```

Listing 33: SectionMapper

```
package com.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
    RowMapper;

import com.dentist.version.three.form.
    Section;


public class SectionMapper implements
    RowMapper<Section> {
    public Section mapRow(ResultSet rs,
        int rowNum) throws SQLException
        {
        Section section = new Section();
        section.setSection_id(rs.getInt("
            section_id"));
        section.setSection_name(rs.
            getString("section_name"));
        section.setSection_description(rs
            .getString("
            section_description"));
        section.setCreated_by(rs.
            getString("created_by"));
        section.setCreated_date(rs.
            getString("created_date"));

        return section;
    }
}
```

Listing 34: SectionSP

```
package com.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.ArrayList;

import com.dentist.version.three.form.
    Role;
import com.dentist.version.three.form.
    Section;
import com.dentist.version.three.form.
    User;
import com.dentist.version.three.form.
    UserRoleSection;
import com.dentist.version.three.mapper.
    RoleMapper;
import com.dentist.version.three.mapper.
    SectionMapper;
import com.dentist.version.three.mapper.
    UserMapper;

public class SectionSP {

 private String database_username;
 private final String database_password="
     yes?bakitpo?";
```

```
public void setDatabase_username(String
    database_username) {
 this.database_username =
     database_username;
}


public String getDatabase_username() {
 return database_username;
}


//insert section
 public void executeSection(String
     section_name, String
     section_description,String
     created_by, String created_date)
     throws Exception{

 Class.forName("org.postgresql.Driver").
     newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);

     CallableStatement calstat=conn.
         prepareCall("{call
         insert_section(?,?,?,?)}");
         calstat.setString(1,
             section_name);
         calstat.setString(2,
             section_description);
         calstat.setString(3,created_by)
             ;
         calstat.setString(4,
             created_date);

         ResultSet rs = calstat.
             executeQuery();
         conn.close();
         calstat.close();
         System.out.println("Your data
             has been inserted into
             table.");


 }

//list all section names
public ArrayList<String> listAllSection
    () throws Exception{
 ArrayList<String> listallsection= new
     ArrayList<String>();
 Class.forName("org.postgresql.Driver").
     newInstance();
  Connection conn=DriverManager.
      getConnection("jdbc:postgresql
      ://localhost:5432/DentISt",
      database_username,
      database_password);
 conn.setAutoCommit(false);
  CallableStatement calstat=conn.
      prepareCall("{call
      listallsectionnames()}");

   ResultSet rs = calstat.executeQuery
       ();

   while(rs.next()){
       System.out.println(rs.
           getString(1));
      listallsection.add(rs.getString
          (1));
      }


  conn.close();
  calstat.close();
  System.out.println("Section list
      selected.");
   return listallsection;
}


//list all sections
public ArrayList<Section> allSectionList
    () throws Exception{
 ArrayList<Section> listSections= new
     ArrayList<Section>();
 Class.forName("org.postgresql.Driver").
     newInstance();
  Connection conn=DriverManager.
      getConnection("jdbc:postgresql
      ://localhost:5432/DentISt",
      database_username,
      database_password);
 conn.setAutoCommit(false);
  CallableStatement calstat=conn.
      prepareCall("{call
      listallsections()}");

   ResultSet rs = calstat.executeQuery
       ();

   while(rs.next()){
       Section section= new Section();
       SectionMapper sectionmap= new
           SectionMapper();
      section= sectionmap.mapRow(rs, 5)
          ;
```

```
        listSections.add(section);
        //System.out.println(rs.getString
            (1));
           }


      conn.close();
      calstat.close();
      System.out.println("Successful call
          for listallsections function");
       return listSections;
}

//delete section
public void deleteSections(int
     section_id) throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
 // conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call
        delete_sections(?)}");
      calstat.setInt(1, section_id);

      ResultSet rs = calstat.executeQuery
           ();



      conn.close();
      calstat.close();
      System.out.println("Successful call
          for DELETE function");
}



//get Section
public Section getSection(int section_id
     ) throws Exception{
 Class.forName("org.postgresql.Driver").
      newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call getsection(?)
        }");
      calstat.setInt(1, section_id);

      ResultSet rs = calstat.executeQuery
           ();
          Section section= new Section();
          SectionMapper sectionmap= new
              SectionMapper();
       while(rs.next()){

       section= sectionmap.mapRow(rs, 5);

        //System.out.println(rs.getString
            (1));
           }


      conn.close();
      calstat.close();
      System.out.println("Successful call
          for getsection function");
       return section;
}

//update section
 public void updateSection(int
     section_id, String section_name,
      String section_description) throws
       Exception{

  Class.forName("org.postgresql.Driver")
      .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

     CallableStatement calstat=conn.
         prepareCall("{call
         update_sections(?,?,?)}");
       //calstat.registerOutParameter
           (10, java.sql.Types.INTEGER
           );
        calstat.setInt(1,section_id);
          calstat.setString(2,
              section_name);
        calstat.setString(3,
            section_description);

        ResultSet rs= calstat.
            executeQuery();

        conn.close();
        calstat.close();
        System.out.println("Your data
            has been updated into
```

```
            table.");


 }

 //insert role to section
 public void insertRoleSection(int
     section_id, String role_name)
     throws Exception{

  Class.forName("org.postgresql.Driver")
      .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

      CallableStatement calstat=conn.
          prepareCall("{call_
          apply_section(?,?)}");
         calstat.setInt(1,section_id);
         calstat.setString(2,role_name)
             ;


         ResultSet rs = calstat.
             executeQuery();
         conn.close();
         calstat.close();
         System.out.println("Your_data_
             has_been_inserted_into_
             table.");


 }

//list roles in a section
 public  ArrayList<Role>
     getListRoleSection(int section_id)
      throws Exception{

  ArrayList<Role> roleLists= new
      ArrayList<Role>();

  Class.forName("org.postgresql.Driver")
      .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

      CallableStatement calstat=conn.
          prepareCall("{call_
          listrolesection(?)}");
      //calstat.registerOutParameter
          (10, java.sql.Types.INTEGER
          );
        calstat.setInt(1,section_id);

        ResultSet rs= calstat.
            executeQuery();

        while(rs.next()){

         Role role= new Role();
               RoleMapper rolemap= new
                   RoleMapper();
              role= rolemap.mapRow(rs,
                  5);


            roleLists.add(role);
        }

        conn.close();
        calstat.close();
        System.out.println("Function_
            has_been_called.");

        return roleLists;
    }


//list of users in a section
 public  ArrayList<User>
     getListUserSection(int role_id)
     throws Exception{

  ArrayList<User> userLists= new
      ArrayList<User>();

  Class.forName("org.postgresql.Driver")
      .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

      CallableStatement calstat=conn.
          prepareCall("{call_
          listusersection(?)}");
        //calstat.registerOutParameter
          (10, java.sql.Types.INTEGER
          );
        calstat.setInt(1,role_id);

        ResultSet rs= calstat.
            executeQuery();
```

```
            while(rs.next()){

             User user= new User();
                    UserMapper usermap= new
                        UserMapper();
                   user= usermap.mapRow(rs,
                        10);
             System.out.println("CHECK_:"+
                    user.getUser_id());
              userLists.add(user);
            }

            conn.close();
            calstat.close();
            System.out.println("Function_
                  has_been_called.");

            return userLists;
    }

  //delete user role section
   public void deleteRoleSections(int
        role_id) throws Exception{
    Class.forName("org.postgresql.Driver")
         .newInstance();
      Connection conn=DriverManager.
           getConnection("jdbc:postgresql
           ://localhost:5432/DentISt",
           database_username,
           database_password);
    // conn.setAutoCommit(false);
      CallableStatement calstat=conn.
           prepareCall("{call_
           delete_rolesections(?)}");
        calstat.setInt(1, role_id);

        ResultSet rs = calstat.
             executeQuery();



       conn.close();
       calstat.close();
       System.out.println("Successful_call
            _for_DELETE_function");
    }

    //list sections of a user
     public  ArrayList<Section>
          getListSectionsOfUser(int user_id
          ) throws Exception{

      ArrayList<Section> sectionLists= new
           ArrayList<Section>();

      Class.forName("org.postgresql.Driver"
           ).newInstance();
        Connection conn=DriverManager.
             getConnection("jdbc:postgresql
             ://localhost:5432/DentISt",
             database_username,
             database_password);

         CallableStatement calstat=conn.
              prepareCall("{call_
              listsectionofusers(?)}");
           //calstat.registerOutParameter
                (10, java.sql.Types.
                INTEGER);
             calstat.setInt(1,user_id);

             ResultSet rs= calstat.
                  executeQuery();

             while(rs.next()){

              Section section= new Section
                   ();
                SectionMapper
                     sectionmap= new
                     SectionMapper();
               section= sectionmap.
                    mapRow(rs, 5);

                System.out.println("BEEP:"
                     +section.getSection_id
                     ());
                sectionLists.add(section);
             }

             conn.close();
             calstat.close();
             System.out.println("Function_
                  has_been_called.");

             return sectionLists;
      }
  }
```

Listing 35: ServicesNeeded

```
package com.dentist.version.three.form;

import java.util.ArrayList;

public class ServiceNeeded {

 private int serviceneeded_id;
 private int patient_id;
```

```
private Integer[] class_1;
private Integer[] class_2;
private Integer[] class_3;
private Integer[] class_4;
private Integer[] class_5;
private Integer[] onlay;
private Integer[] extraction;
private Integer[] odontectomy;
private Integer[] special_case;
private Integer[] pulp_sedation;
private Integer[] crown_recementation;
private Integer[] filling_service;
private Integer[] laminated;
private Integer[] single_crown;
private Integer[] bridge_service;
private Integer[] anterior;
private Integer[] posterior;
private Integer[] ortho_endo;
private String periodontics;
private String surgery;
private String emergency_treatment;
private String prosthodontics;
private String updated_by;
private String updated_date;
private String updated_time;
private int version;
private String notes;
private String is_current;
/*
 * String[] version
 */
private String[] class_1_string;
private String[] class_2_string;
private String[] class_3_string;
private String[] class_4_string;
private String[] class_5_string;
private String[] onlay_string;
private String[] extraction_string;
private String[] odontectomy_string;
private String[] special_case_string;
private String[] pulp_sedation_string;
private String[]
    crown_recementation_string;
private String[] filling_service_string;
private String[] laminated_string;
private String[] single_crown_string;
private String[] bridge_service_string;
private String[] anterior_string;
private String[] posterior_string;
private String[] ortho_endo_string;

/*
 * private functions
 */

private Integer[] splitStringToInteger(
    String[] temp){
 Integer[] result= new Integer[temp.
     length];
 String tempResult= "";
 for(int i=0; i<temp.length;i++){
  tempResult=temp[i];
  if(temp[i].indexOf("{")!=-1 || temp[i
      ].indexOf("}")!=-1){
   if(temp[i].indexOf("{")!=-1)
    tempResult=tempResult.replace("{", "
        ").trim();
   if(temp[i].indexOf("}")!=-1)
   tempResult=tempResult.replace("}", "
        ").trim();

   System.out.println("Check:_"+
       tempResult);
  }

  result[i]=Integer.parseInt(tempResult)
      ;
 }
 return result;
}

private String[] splitStringToSpecific(
    String[] temp){
 ArrayList<String> tempResult= new
     ArrayList<String>();
 String tempAr= "";
 String[] result=null;
 if(temp!=null){
 for(int i=0; i<temp.length;i++){
  if(temp[i].length()>2)
  tempResult.add(temp[i]);
 }
 result= new String[tempResult.size()];
 for(int j=0;j<tempResult.size();j++){
  tempAr=tempResult.get(j);

  if(tempResult.get(j).indexOf("{")!=-1
      || tempResult.get(j).indexOf("}")
      !=-1){
   if(tempResult.get(j).indexOf("{")
       !=-1)
    tempAr=tempAr.replace("{", "").trim
        ();
   if(tempResult.get(j).indexOf("}")
       !=-1)
    tempAr=tempAr.replace("}", "").trim
        ();

   System.out.println("Check:_"+ tempAr
       +"_noms");
  }
  result[j]=tempAr;
```

```
 }
 }
 else{
  result=new String[1];
  result[0]="-1";
 }

 return result;
}

public void setValuesNeeded(String name,
      String value){
 String[] partial=null;
 if(name.equals("class_1")){
  partial=value.split(",");
  this.setClass_1(partial);
 }
 else if(name.equals("class_2")){
  partial=value.split(",");
  this.setClass_2(partial);
 }
 else if(name.equals("class_3")){
  partial=value.split(",");
  this.setClass_3(partial);
 }
 else if(name.equals("class_4")){
  partial=value.split(",");
  this.setClass_4(partial);
 }
 else if(name.equals("class_5")){
  partial=value.split(",");
  this.setClass_5(partial);
 }
 else if(name.equals("onlay")){
  partial=value.split(",");
  this.setOnlay(partial);
 }
 else if(name.equals("extraction")){
  partial=value.split(",");
  this.setExtraction(partial);
 }
 else if(name.equals("odontectomy")){
  partial=value.split(",");
  this.setOdontectomy(partial);
 }
 else if(name.equals("special_case")){
  partial=value.split(",");
  this.setSpecial_case(partial);
 }
 else if(name.equals("pulp_sedation")){
  partial=value.split(",");
  this.setPulp_sedation(partial);
 }
 else if(name.equals("
      crown_recementation")){
  partial=value.split(",");
  this.setCrown_recementation(partial);
 }
 else if(name.equals("filling_service"))
      {
  partial=value.split(",");
  this.setFilling_service(partial);
 }
 else if(name.equals("laminated")){
  partial=value.split(",");
  this.setLaminated(partial);
 }
 else if(name.equals("single_crown")){
  partial=value.split(",");
  this.setSingle_crown(partial);
 }
 else if(name.equals("bridge_service")){
  partial=value.split(",");
  this.setBridge_service(partial);
 }
 else if(name.equals("anterior")){
  partial=value.split(",");
  this.setAnterior(partial);
 }
 else if(name.equals("posterior")){
  partial=value.split(",");
  this.setPosterior(partial);
 }
 else if(name.equals("ortho_endo")){
  partial=value.split(",");
  this.setOrtho_endo(partial);
 }
}


/*
 * Getters AND Setters
 */


public void setServiceneeded_id(int
     serviceneeded_id) {
 this.serviceneeded_id =
      serviceneeded_id;
}
public int getServiceneeded_id() {
 return serviceneeded_id;
}
public void setPatient_id(int patient_id
     ) {
 this.patient_id = patient_id;
}
public int getPatient_id() {
 return patient_id;
}
public void setClass_1(String[] class_1)
      {
```

```
 Integer[] class_1Result=null;
 if (class_1 != null) {
 class_1Result=splitStringToInteger(
      class_1);
 }
 this.setClass_1_string(
      splitStringToSpecific(class_1));
 this.class_1 = class_1Result;
}
public Integer[] getClass_1() {
 return class_1;
}
public void setClass_2(String[] class_2)
      {

 Integer[] class_2Result=null;
 if (class_2 != null) {
 class_2Result=splitStringToInteger(
      class_2);
 }
 this.setClass_2_string(
      splitStringToSpecific(class_2));

 this.class_2 = class_2Result;
}
public Integer[] getClass_2() {
 return class_2;
}
public void setClass_3(String[] class_3)
      {
 Integer[] class_3Result=null;
 if (class_3 != null) {
 class_3Result=splitStringToInteger(
      class_3);
 }
 this.setClass_3_string(
      splitStringToSpecific(class_3));

 this.class_3 = class_3Result;
}
public Integer[] getClass_3() {
 return class_3;
}
public void setClass_4(String[] class_4)
      {

 Integer[] class_4Result=null;
 if (class_4 != null ) {
  class_4Result=splitStringToInteger(
       class_4);
 }
 this.setClass_4_string(
      splitStringToSpecific(class_4));
 this.class_4 = class_4Result;
}
public Integer[] getClass_4() {
 return class_4;
}
public void setClass_5(String[] class_5)
      {

 Integer[] class_5Result=null;
 if (class_5 != null) {
  class_5Result=splitStringToInteger(
       class_5);
 }
 this.setClass_5_string(
      splitStringToSpecific(class_5));


 this.class_5 = class_5Result;
}
public Integer[] getClass_5() {
 return class_5;
}
public void setOnlay(String[] onlay) {

 Integer[] onlayResult=null;
 if (onlay != null) {
  onlayResult=splitStringToInteger(onlay
       );
 }
 this.setOnlay_string(
      splitStringToSpecific(onlay));


 this.onlay = onlayResult;
}
public Integer[] getOnlay() {
 return onlay;
}
public void setExtraction(String[]
     extraction) {


 Integer[] extractionResult=null;
 if (extraction != null ) {
  extractionResult=splitStringToInteger(
       extraction);
 }
 this.setExtraction_string(
      splitStringToSpecific(extraction))
      ;


 this.extraction = extractionResult;
}
public Integer[] getExtraction() {
 return extraction;
```

```
}
public void setOdontectomy(String[]
    odontectomy) {

 Integer[] odontectomyResult=null;
 if (odontectomy != null) {
  odontectomyResult=splitStringToInteger
      (odontectomy);
 }
 this.setOdontectomy_string(
     splitStringToSpecific(odontectomy)
     );

 this.odontectomy = odontectomyResult;
}
public Integer[] getOdontectomy() {
 return odontectomy;
}
public void setSpecial_case(String[]
    special_case) {


 Integer[] special_caseResult=null;
 if (special_case != null) {
  special_caseResult=
      splitStringToInteger(special_case
      );
 }
 this.setSpecial_case_string(
     splitStringToSpecific(special_case
     ));



 this.special_case = special_caseResult;
}
public Integer[] getSpecial_case() {
 return special_case;
}
public void setPulp_sedation(String[]
    pulp_sedation) {

 Integer[] pulp_sedationResult=null;
 if (pulp_sedation != null) {
  pulp_sedationResult=
      splitStringToInteger(
      pulp_sedation);
 }
 this.setPulp_sedation_string(
     splitStringToSpecific(
     pulp_sedation));

 this.pulp_sedation =
     pulp_sedationResult;
}
public Integer[] getPulp_sedation() {
 return pulp_sedation;
}
public void setCrown_recementation(
    String[] crown_recementation) {

 Integer[] crown_recementationResult=
     null;
 if (crown_recementation != null) {
  crown_recementationResult=
      splitStringToInteger(
      crown_recementation);
 }
 this.setCrown_recementation_string(
     splitStringToSpecific(
     crown_recementation));

 this.crown_recementation =
     crown_recementationResult;
}
public Integer[] getCrown_recementation
    () {
 return crown_recementation;
}
public void setFilling_service(String[]
    filling_service) {

 Integer[] filling_serviceResult=null;
 if (filling_service != null) {
  filling_serviceResult=
      splitStringToInteger(
      filling_service);
 }
 this.setFilling_service_string(
     splitStringToSpecific(
     filling_service));

 this.filling_service =
     filling_serviceResult;
}
public Integer[] getFilling_service() {
 return filling_service;
}
public void setLaminated(String[]
    laminated) {

 Integer[] laminatedResult=null;
 if (laminated != null) {
  laminatedResult=splitStringToInteger(
      laminated);
 }
 this.setLaminated_string(
     splitStringToSpecific(laminated));

 this.laminated = laminatedResult;
}
public Integer[] getLaminated() {
```

```
 return laminated;
}
public void setBridge_service(String[]
     bridge_service) {

 Integer[] bridge_serviceResult=null;
 if (bridge_service != null) {
  bridge_serviceResult=
       splitStringToInteger(
       bridge_service);
 }
 this.setBridge_service_string(
      splitStringToSpecific(
      bridge_service));

 this.bridge_service =
      bridge_serviceResult;
}
public Integer[] getBridge_service() {
 return bridge_service;
}
public void setSingle_crown(String[]
     single_crown) {

 Integer[] single_crownResult=null;
 if (single_crown != null ) {
  single_crownResult=
       splitStringToInteger(single_crown
       );
 }
 this.setSingle_crown_string(
      splitStringToSpecific(single_crown
      ));

 this.single_crown = single_crownResult;
}
public Integer[] getSingle_crown() {
 return single_crown;
}
public void setAnterior(String[]
     anterior) {

 Integer[] anteriorResult=null;
 if (anterior != null) {
  anteriorResult=splitStringToInteger(
       anterior);
 }
 this.setAnterior_string(
      splitStringToSpecific(anterior));



 this.anterior = anteriorResult;
}
public Integer[] getAnterior() {
 return anterior;
}
public void setPosterior(String[]
     posterior) {

 Integer[] posteriorResult=null;
 if (posterior != null) {
  posteriorResult=splitStringToInteger(
       posterior);
 }
 this.setPosterior_string(
      splitStringToSpecific(posterior));



 this.posterior = posteriorResult;
}
public Integer[] getPosterior() {
 return posterior;
}
public void setOrtho_endo(String[]
     ortho_endo) {

 Integer[] ortho_endoResult=null;
 if (ortho_endo != null ) {
  ortho_endoResult=splitStringToInteger(
       ortho_endo);
 }
 this.setOrtho_endo_string(
      splitStringToSpecific(ortho_endo))
      ;

 this.ortho_endo = ortho_endoResult;
}
public Integer[] getOrtho_endo() {
 return ortho_endo;
}
public void setPeriodontics(String
     periodontics) {
 this.periodontics = periodontics;
}
public String getPeriodontics() {
 return periodontics;
}
public void setEmergency_treatment(
     String emergency_treatment) {
 this.emergency_treatment =
      emergency_treatment;
}
public String getEmergency_treatment() {
 return emergency_treatment;
}
public void setProsthodontics(String
     prosthodontics) {
 this.prosthodontics = prosthodontics;
}
public String getProsthodontics() {
 return prosthodontics;
```

```
}
public void setUpdated_by(String
    updated_by) {
 this.updated_by = updated_by;
}
public String getUpdated_by() {
 return updated_by;
}
public void setUpdated_date(String
    updated_date) {
 this.updated_date = updated_date;
}
public String getUpdated_date() {
 return updated_date;
}
public void setUpdated_time(String
    updated_time) {
 this.updated_time = updated_time;
}
public String getUpdated_time() {
 return updated_time;
}
public void setVersion(int version) {
 this.version = version;
}
public int getVersion() {
 return version;
}

public void setClass_1_string(String[]
    class_1_string) {
 this.class_1_string = class_1_string;
}

public String[] getClass_1_string() {
 return class_1_string;
}

public void setClass_2_string(String[]
    class_2_string) {
 this.class_2_string = class_2_string;
}

public String[] getClass_2_string() {
 return class_2_string;
}

public void setClass_3_string(String[]
    class_3_string) {
 this.class_3_string = class_3_string;
}

public String[] getClass_3_string() {
 return class_3_string;
}

public void setClass_4_string(String[]
    class_4_string) {
 this.class_4_string = class_4_string;
}

public String[] getClass_4_string() {
 return class_4_string;
}

public void setClass_5_string(String[]
    class_5_string) {
 this.class_5_string = class_5_string;
}

public String[] getClass_5_string() {
 return class_5_string;
}

public void setOnlay_string(String[]
    onlay_string) {
 this.onlay_string = onlay_string;
}

public String[] getOnlay_string() {
 return onlay_string;
}

public void setExtraction_string(String
    [] extraction_string) {
 this.extraction_string =
     extraction_string;
}

public String[] getExtraction_string() {
 return extraction_string;
}

public void setOdontectomy_string(String
    [] odontectomy_string) {
 this.odontectomy_string =
     odontectomy_string;
}

public String[] getOdontectomy_string()
    {
 return odontectomy_string;
}

public void setSpecial_case_string(
    String[] special_case_string) {
 this.special_case_string =
     special_case_string;
}

public String[] getSpecial_case_string()
    {
```

```
 return special_case_string;
}

public void setPulp_sedation_string(
     String[] pulp_sedation_string) {
 this.pulp_sedation_string =
      pulp_sedation_string;
}

public String[] getPulp_sedation_string
     () {
 return pulp_sedation_string;
}

public void
     setCrown_recementation_string(
  String[] crown_recementation_string) {
 this.crown_recementation_string =
      crown_recementation_string;
}

public String[]
     getCrown_recementation_string() {
 return crown_recementation_string;
}

public void setFilling_service_string(
     String[] filling_service_string) {
 this.filling_service_string =
      filling_service_string;
}

public String[]
     getFilling_service_string() {
 return filling_service_string;
}

public void setLaminated_string(String[]
      laminated_string) {
 this.laminated_string =
      laminated_string;
}

public String[] getLaminated_string() {
 return laminated_string;
}

public void setSingle_crown_string(
     String[] single_crown_string) {
 this.single_crown_string =
      single_crown_string;
}

public String[] getSingle_crown_string()
      {
 return single_crown_string;
}

public void setBridge_service_string(
     String[] bridge_service_string) {
 this.bridge_service_string =
      bridge_service_string;
}

public String[] getBridge_service_string
     () {
 return bridge_service_string;
}

public void setAnterior_string(String[]
     anterior_string) {
 this.anterior_string = anterior_string;
}

public String[] getAnterior_string() {
 return anterior_string;
}

public void setPosterior_string(String[]
      posterior_string) {
 this.posterior_string =
      posterior_string;
}

public String[] getPosterior_string() {
 return posterior_string;
}

public void setOrtho_endo_string(String
     [] ortho_endo_string) {
 this.ortho_endo_string =
      ortho_endo_string;
}

public String[] getOrtho_endo_string() {
 return ortho_endo_string;
}

public void setSurgery(String surgery) {
 this.surgery = surgery;
}

public String getSurgery() {
 return surgery;
}

public void setNotes(String notes) {
 this.notes = notes;
}

public String getNotes() {
 return notes;
```

```
}

public void setIs_current(String
    is_current) {
 this.is_current = is_current;
}

public String getIs_current() {
 return is_current;
}
}
```

Listing 36: ServiceNeededMapper

```
package com.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
    RowMapper;

import com.dentist.version.three.form.
    ServiceNeeded;

public class ServicesNeededMapper
    implements RowMapper<ServiceNeeded>{

 public ServiceNeeded mapRow(ResultSet rs
     , int rowNum) throws SQLException {

  ServiceNeeded serviceNeeded= new
      ServiceNeeded();

   serviceNeeded.setServiceneeded_id(rs.
       getInt("serviceneeded_id"));
   serviceNeeded.setPatient_id(rs.getInt(
       "patient_id"));
   serviceNeeded.setValuesNeeded("class_1
       ",rs.getArray("class_1").toString
       ());
   serviceNeeded.setValuesNeeded("class_2
       ",rs.getArray("class_2").toString
       ());
   serviceNeeded.setValuesNeeded("class_3
       ",rs.getArray("class_3").toString
       ());
   serviceNeeded.setValuesNeeded("class_4
       ",rs.getArray("class_4").toString
       ());
   serviceNeeded.setValuesNeeded("class_5
       ",rs.getArray("class_5").toString
       ());
   serviceNeeded.setValuesNeeded("onlay",
       rs.getArray("onlay").toString());
   serviceNeeded.setValuesNeeded("
       extraction",rs.getArray("
       extraction").toString());
   serviceNeeded.setValuesNeeded("
       odontectomy",rs.getArray("
       odontectomy").toString());
   serviceNeeded.setValuesNeeded("
       special_case",rs.getArray("
       special_case").toString());
   serviceNeeded.setValuesNeeded("
       pulp_sedation",rs.getArray("
       pulp_sedation").toString());
   serviceNeeded.setValuesNeeded("
       crown_recementation",rs.getArray(
       "crown_recementation").toString()
       );
   serviceNeeded.setValuesNeeded("
       filling_service",rs.getArray("
       filling_service").toString());
   serviceNeeded.setValuesNeeded("
       laminated",rs.getArray("laminated
       ").toString());
   serviceNeeded.setValuesNeeded("
       single_crown",rs.getArray("
       single_crown").toString());
   serviceNeeded.setValuesNeeded("
       bridge_service",rs.getArray("
       bridge_service").toString());
   serviceNeeded.setValuesNeeded("
       anterior",rs.getArray("anterior")
       .toString());
   serviceNeeded.setValuesNeeded("
       posterior",rs.getArray("posterior
       ").toString());
   serviceNeeded.setValuesNeeded("
       ortho_endo",rs.getArray("
       ortho_endo").toString());
   serviceNeeded.setPeriodontics(rs.
       getString("periodontics"));
   serviceNeeded.setSurgery(rs.getString(
       "surgery"));
   serviceNeeded.setEmergency_treatment(
       rs.getString("emergency_treatment
       "));
   serviceNeeded.setProsthodontics(rs.
       getString("prosthodontics"));
   serviceNeeded.setUpdated_by(rs.
       getString("updated_by"));
   serviceNeeded.setUpdated_date(rs.
```

```
            getString("updated_date"));
        serviceNeeded.setUpdated_time(rs.
            getString("updated_time"));
        serviceNeeded.setVersion(rs.getInt("
            version"));
        serviceNeeded.setNotes(rs.getString("
            notes"));
        serviceNeeded.setIs_current(rs.
            getString("is_current"));

        return serviceNeeded;
      }
    }
```

Listing 37: SimpleFormController

```
package com.dentist.version.three.
    controller;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

import javax.servlet.http.
    HttpServletRequest;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;



import org.dentist.version.three.
    processserver.
    HumanTaskStartupServlet;
import org.springframework.stereotype.
    Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.
    BindingResult;
import org.springframework.web.bind.
    annotation.ModelAttribute;
import org.springframework.web.bind.
    annotation.RequestMapping;
import org.springframework.web.bind.
    annotation.RequestMethod;
import org.springframework.web.bind.
    annotation.RequestParam;

import com.dentist.version.three.db.
    AdminSP;
import com.dentist.version.three.db.
    DentalChartSP;
import com.dentist.version.three.db.
    RoleSP;
import com.dentist.version.three.db.
    SectionSP;
import com.dentist.version.three.db.
    UserSP;
import com.dentist.version.three.db.
    XacmlSP;
import com.dentist.version.three.form.
    CariesStatus;
import com.dentist.version.three.form.
    DeleteFunction;
import com.dentist.version.three.form.
    DentalChart;
import com.dentist.version.three.form.
    PrivilegeRole;
import com.dentist.version.three.form.
    RecurrentStatus;
import com.dentist.version.three.form.
    RestorationStatus;
import com.dentist.version.three.form.
    Role;
import com.dentist.version.three.form.
    RoleSection;
import com.dentist.version.three.form.
    RoleUser;
import com.dentist.version.three.form.
    Section;
import com.dentist.version.three.form.
    ServiceNeeded;
import com.dentist.version.three.form.
    TargetExtract;
import com.dentist.version.three.form.
    User;
import com.dentist.version.three.form.
    UserRoleSection;
import com.dentist.version.three.xacml.
    PolicyReader;
import com.dentist.version.three.xacml.
    XACMLExtract;

@Controller
public class SimpleFormController{

private int userID;
private ArrayList<String> allRoles;
private ArrayList<String> tempUsernames;
private ArrayList<String> tempRolenames;
private ArrayList<String>
    tempSectionnames;

//user form
@RequestMapping(value = "/userForm.html",
     method = RequestMethod.GET)
```

```
public String simpleForm(Model model,
    HttpServletRequest request,
    HttpServletResponse response,
    HttpSession session) throws
    Exception{

 session=request.getSession(false);

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";

 String allUsernames="";
 UserSP userSP= new UserSP();
 AdminSP adminSP= new AdminSP();
 try {
  if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
   errorMessage="Session has expired. Pls
       log again.";
  else{
   String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
       i++){
    privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "users", "insert");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
       privilegeCheck);
   userSP.setDatabase_username(
       currentRole);
    adminSP.setDatabase_username(
        currentRole);

   tempUsernames= userSP.getUsernameList
       ();
  }

 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
  System.out.println("ERROR: "+
      errorMessage);
  return "Error";
 }

 for(int i=0;i<tempUsernames.size();i++)
 {
      if (allUsernames == "")
       allUsernames = tempUsernames.get(
           i);
       else
        allUsernames = allUsernames + ",
            " + tempUsernames.get(i);
 }
 //List of secret_questions
  ArrayList<String> secret_questions= new
      ArrayList<String>();

    secret_questions.add("What is your
        mother's middle name?");
    secret_questions.add("What is your
        favorite cartoon?");
    secret_questions.add("What was your
        childhood nickname?");
    secret_questions.add("What is the
        name of your favorite childhood
        friend? ");
    secret_questions.add("What is the
        first name of your oldest niece?
        ");
    secret_questions.add("Who is your
        favorite author?");
    secret_questions.add("What is the
        title of your favorite book?");
    secret_questions.add("What was the
        last name of your favorite
        teacher?");
    secret_questions.add("What was the
        name of your first pet?");
    secret_questions.add("What is the
        name of your all-time favorite
        sports team?");


model.addAttribute(new User());
model.addAttribute("allUsernames",
    allUsernames);
model.addAttribute("secret_questions",
    secret_questions);

 return "userForm";
}

// user form result
@RequestMapping(value = "/useroutput.html
    ", method = RequestMethod.POST)

public String simple(@ModelAttribute User
     user, Model model,
    HttpServletRequest request,
    HttpServletResponse response,
    HttpSession session) throws
```

```
    Exception{

session=request.getSession(false);

//date time
 DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
   //get current date time with Date()
   Date date = new Date();

//get current date time with Calendar()

   String dateString= String.valueOf(
        dateFormat.format(date));

//end for date time
   boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";

user.setCreated_date(dateString);
UserSP userSP= new UserSP();
AdminSP adminSP= new AdminSP();
try {
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
       log again.";
 else{
  String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
       i++){
   privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "users", "insert");
   if(privilegeCheck){
    currentRole=currentRoleList[i];
    break;
   }
  }
  System.out.println(currentRole+"-"+
       privilegeCheck);
  userSP.setDatabase_username(
       currentRole);
  adminSP.setDatabase_username(
       currentRole);

 userSP.executeUser(user.getFname_user()
      ,user.getMinit_user(),user.
      getLname_user(),
   user.getUsername(),user.getPassword()
        ,user.getEmail(), user.
        getSecret_question(),user.
        getSecret_answer(), user.
        getCreated_date());
 userID=userSP.getUserID(user.
      getUsername());
 try{
  /** start add new user to jbpm**/
  HumanTaskStartupServlet.taskSession.
       addUser(new org.jbpm.task.User(
       user.getUsername()));
  /**end add new user to jbpm**/
  }
  catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
  }
 //audittrail
 int sessionUserID= Integer.parseInt(
      session.getAttribute("
      sessionUserId").toString());
 User sessionUser= userSP.getUser(
      sessionUserID);
 String action_performed=user.
      getUsername();
 String name= sessionUser.getFname_user
      ()+" "+ sessionUser.getMinit_user
      () + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";

 adminSP.insertAuditTrail(name, "INSERT"
      , action_performed, "Users",
      dateString);
 }
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
  System.out.println("ERROR: "+
       errorMessage);
  return "Error";
 }
 user.setUser_id(userID);

 model.addAttribute("user", user);

 return "useroutput";
}

//role form
@RequestMapping(value = "/roleForm.html",
      method = RequestMethod.GET)
public String roleForm(Model model,
     HttpServletRequest request,
     HttpServletResponse response,
     HttpSession session) throws
     Exception{
```

```
session=request.getSession(false);

boolean privilegeCheck=false;
String errorMessage="";
String currentRole="";
AdminSP adminSP= new AdminSP();

String allRolenames="";
RoleSP roleSP= new RoleSP();

try {
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
      log again.";
 else{
  String[] currentRoleList= (String[])
      session.getAttribute("
      currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
      i++){
   privilegeCheck= adminSP.
       checkPrivilege(currentRoleList[i
       ], "role", "insert");
   if(privilegeCheck){
    currentRole=currentRoleList[i];
    break;
   }
  }
  System.out.println(currentRole+"-"+
      privilegeCheck);

  roleSP.setDatabase_username(
      currentRole);
  adminSP.setDatabase_username(
      currentRole);

 tempRolenames= roleSP.listAllRoles();
}
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 return "Error";
}

for(int i=0;i<tempRolenames.size();i++)
{
      if (allRolenames == "")
       allRolenames = tempRolenames.get(
           i);
       else
        allRolenames = allRolenames + ",
            " + tempRolenames.get(i);
 }

model.addAttribute("allRolenames",
     allRolenames);
model.addAttribute(new Role());
 return "roleForm";

}



//role form result
@RequestMapping(value = "/roleoutput.html
     ", method = RequestMethod.POST)

public String roleResult(@ModelAttribute
     Role role, Model model,
     HttpServletRequest request,
     HttpServletResponse response,
     HttpSession session) throws
     Exception{

 session=request.getSession(false);
 //date time
  DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
        date));



    String dateString= dateFormat.format(
        date);

 //end for date time



    boolean privilegeCheck=false;
     String errorMessage="";
     String currentRole="";
     AdminSP adminSP= new AdminSP();

 role.setCreated_by(session.getAttribute(
     "sessionUser").toString());
 role.setCreated_date(dateString);

 //start database connection by using
     datasource

 RoleSP roleSP= new RoleSP();
 UserSP userSP= new UserSP();
  int role_id=0;
 try {
  if(session.getAttribute("
```

```
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
 errorMessage="Session has expired. Pls
      log again.";
else{
 String[] currentRoleList= (String[])
      session.getAttribute("
      currentDatabaseList");
 for(int i=0; i<currentRoleList.length;
      i++){
  privilegeCheck= adminSP.
       checkPrivilege(currentRoleList[i
       ], "role", "insert");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
      privilegeCheck);

   adminSP.setDatabase_username(
        currentRole);
roleSP.setDatabase_username(currentRole
     );
role_id= roleSP.executeRole(role.
     getRole_name(), role.
     getRole_description(),role.
     getCreated_by(),role.
     getCreated_date());


try{
 /** start add new group to jbpm**/
 HumanTaskStartupServlet.taskSession.
      addGroup(new org.jbpm.task.Group(
      role.getRole_name()));
 /**end add new group to jbpm**/
 }
 catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }
if(role_id!=0){
 roleSP.insert_databaseRole(role_id,
      role.getDatabase_role());
}

int sessionUserID= Integer.parseInt(
     session.getAttribute("
     sessionUserId").toString());
User sessionUser= userSP.getUser(
     sessionUserID);
String action_performed=role.
     getRole_name();
String name= sessionUser.getFname_user
      ()+" "+ sessionUser.getMinit_user
      () + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";
 adminSP.insertAuditTrail(name, "INSERT"
      , action_performed, "Role",
      dateString);
 }
 } catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }

 model.addAttribute("role", role);

 return "roleoutput";
}



//role form result
@RequestMapping(value = "/addUserRole",
     method = RequestMethod.GET)

public String viewAddUserRole(
     @RequestParam(value="user_id",
     required = true) int user_id,
       Model model, HttpServletRequest
            request, HttpServletResponse
            response,HttpSession session)
             throws Exception{


 session=request.getSession(false);
 //database connection

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();

 ArrayList<Role> currentRoleLists= new
      ArrayList<Role>();

 DeleteFunction deleteFunction= new
      DeleteFunction();

 RoleSP roleSP= new RoleSP();
 User user= new User();
 UserSP userSP= new UserSP();

 try {
  if(session.getAttribute("
       currentDatabaseList")==null ||
       session.getAttribute("
       sessionUserRole")==null)
   errorMessage="Session has expired. Pls
```

```
        log again.";
    else{
     String[] currentRoleList= (String[])
          session.getAttribute("
          currentDatabaseList");
     for(int i=0; i<currentRoleList.length;
          i++){
      privilegeCheck= adminSP.
           checkPrivilege(currentRoleList[i
           ], "user_role", "select");
      if(privilegeCheck){
       currentRole=currentRoleList[i];
       break;
      }
     }
     System.out.println(currentRole+"-"+
          privilegeCheck);



    deleteFunction.setKey_id(user_id);

     userSP.setDatabase_username(
          currentRole);
     roleSP.setDatabase_username(
          currentRole);
     adminSP.setDatabase_username(
          currentRole);

    allRoles= roleSP.listAllRoles();
    user= userSP.getUser(user_id);
     currentRoleLists= userSP.
          getListRolesofUser(user_id);

    }
   } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }
   RoleUser roleuser= new RoleUser();
   roleuser.setUser_id(user_id);
   System.out.println("HOY ::: "+ roleuser.
        getUser_id());
   model.addAttribute("roleuser",roleuser);
   model.addAttribute("user",user);
   model.addAttribute("allRoles",allRoles);
   model.addAttribute("currentRoleLists",
        currentRoleLists);
   model.addAttribute("deleteFunction",
        deleteFunction);

   return "AddUserRole";
  }

  //user form
  @RequestMapping(value = "/UserRoleConfirm
       .html", method = RequestMethod.POST)
  public String simpleForm(@ModelAttribute
       RoleUser roleuser, @ModelAttribute
       User user, Model model
    , HttpServletRequest request,
        HttpServletResponse response,
        HttpSession session) throws
        Exception{

   session=request.getSession(false);
   ArrayList<Role> currentRoleLists= new
        ArrayList<Role>();
   ArrayList<Role> tempCurrentRoleLists=
        new ArrayList<Role>();

   boolean privilegeCheck=false;
   String errorMessage="";
   String currentRole="";
   AdminSP adminSP= new AdminSP();

   //date time
    DateFormat dateFormat = new
        SimpleDateFormat("dd/MM/yyyy HH:mm
        :ss");
     //get current date time with Date()
     Date date = new Date();
     System.out.println(dateFormat.format(
          date));

     String dateString= dateFormat.format(
          date);

   //end for date time

   DeleteFunction deleteFunction= new
        DeleteFunction();

   UserSP userSP= new UserSP();
   RoleSP roleSP= new RoleSP();
   String[] compareString;
   //bool
   boolean error= false;
   try {
    if(session.getAttribute("
         currentDatabaseList")==null ||
         session.getAttribute("
         sessionUserRole")==null)
     errorMessage="Session has expired. Pls
          log again.";
    else{
     String[] currentRoleList= (String[])
          session.getAttribute("
          currentDatabaseList");
     for(int i=0; i<currentRoleList.length;
          i++){
      privilegeCheck= adminSP.
           checkPrivilege(currentRoleList[i
```

```
        ], "user_role", "insert");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
       privilegeCheck);

 userSP.setDatabase_username(
       currentRole);
 roleSP.setDatabase_username(
       currentRole);
 adminSP.setDatabase_username(
       currentRole);



 deleteFunction.setKey_id(roleuser.
       getUser_id());

 tempCurrentRoleLists= userSP.
       getListRolesofUser(roleuser.
       getUser_id());
 compareString= new String[
       tempCurrentRoleLists.size()];
 int compareCount=0;
 for(Role tempCurrentRoleList:
       tempCurrentRoleLists){
  //for Faculty
  if(tempCurrentRoleList.getRole_name()
        .indexOf("Faculty")!=-1){
   compareString[compareCount]="faculty
         ";
  }
  //for Student Clinician
  else if(tempCurrentRoleList.
        getRole_name().indexOf("Student"
        )!=-1){
   compareString[compareCount]="student
         clinician";
  }
  else{
   compareString[compareCount]="
         administrator";
  }
  compareCount++;
 }
for(int i=0;i<compareString.length;i++)
     {
 if(roleuser.getRole_name().toLowerCase
       ().indexOf("administrator")!=-1){
   error=false;
   break;
 }
 if(roleuser.getRole_name().toLowerCase
       ().indexOf(compareString[i])==-1
        && !compareString[i].equals("
        administrator"))
    error=true;
     break;
 }
 String success="SUCCESS: You have
       successfully added a role to a
       user";
        model.addAttribute("success",
             success);
}
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}
if(error==true){
 try{
  allRoles= roleSP.listAllRoles();
  user= userSP.getUser(roleuser.
        getUser_id());
   currentRoleLists= userSP.
         getListRolesofUser(roleuser.
         getUser_id());
 model.addAttribute("errorMessage","
       ERROR: Faculty and Student
       Clinician Roles cannot be used
       concurrently in one user.");
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }
}
else{
 try{

userSP.addUserRole(roleuser.getUser_id()
      , roleuser.getRole_name());
allRoles= roleSP.listAllRoles();
user= userSP.getUser(roleuser.getUser_id
      ());
 currentRoleLists= userSP.
       getListRolesofUser(roleuser.
       getUser_id());



int sessionUserID= Integer.parseInt(
      session.getAttribute("sessionUserId
      ").toString());
  User sessionUser= userSP.getUser(
        sessionUserID);
  String name= sessionUser.getFname_user
        ()+" "+ sessionUser.getMinit_user
        () + " "+ sessionUser.
        getLname_user()+" ("+ sessionUser
        .getUsername()+")";
  adminSP.insertAuditTrail(name, "INSERT
```

```
        ", user.getUsername() +" (" +
        roleuser.getRole_name()+")","
        user_role" ,dateString);



 System.out.println(roleuser.getUser_id()
     +"   "+roleuser.getRole_name());
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
 model.addAttribute("currentRoleLists",
      currentRoleLists);
 model.addAttribute("roleuser",roleuser);
 model.addAttribute("user",user);
 model.addAttribute("allRoles",allRoles);
 model.addAttribute("deleteFunction",
      deleteFunction);

 return "AddUserRole";
}

//DELETE USERS
//user form
@RequestMapping(value = "/deleteUserRole.
     html", method = RequestMethod.POST)
public String deleteUserRole(
     @ModelAttribute User user, Model
     model, HttpServletRequest request,
     HttpServletResponse response,
   HttpSession session,@ModelAttribute
        DeleteFunction deleteFunction)
        throws Exception{

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();

 session=request.getSession(false);
 ArrayList<Role> currentRoleLists= new
      ArrayList<Role>();

 RoleUser roleuser= new RoleUser();

 UserSP userSP= new UserSP();
 RoleSP roleSP= new RoleSP();
 int[] deleteRoles= deleteFunction.
      getDeleteObject();

 int user_id=deleteFunction.getKey_id();
 roleuser.setUser_id(user_id);
 System.out.println("User_id_of_Delete::
      "+user_id);
 try {
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
       log again.";
 else{
  String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
       i++){
   privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "user_role", "delete");
   if(privilegeCheck){
    currentRole=currentRoleList[i];
    break;
   }
  }
  System.out.println(currentRole+"-"+
       privilegeCheck);
  userSP.setDatabase_username(
       currentRole);
  roleSP.setDatabase_username(
       currentRole);
  adminSP.setDatabase_username(
       currentRole);

  for(int i=0; i<deleteRoles.length; i
       ++){
   try {
    System.out.println("Check_function::
         "+deleteRoles[i]);
    userSP.deleteUserRole(user_id,
         deleteRoles[i]);
   } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }
  }
  allRoles= roleSP.listAllRoles();
  user= userSP.getUser(user_id);
  currentRoleLists= userSP.
       getListRolesofUser(user_id);
 }
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }

 //System.out.println(roleuser.getUser_id
      ()+"  -  "+roleuser.getRole_name())
      ;

 model.addAttribute("currentRoleLists",
```

```
        currentRoleLists);
    model.addAttribute("roleuser",roleuser);
    model.addAttribute("user",user);
    model.addAttribute("allRoles",allRoles);
    model.addAttribute("deleteFunction",
        deleteFunction);

    return "AddUserRole";
}

@RequestMapping(value = "/viewAllUsers.
    html", method = RequestMethod.GET)
public String viewUsers(@ModelAttribute
    DeleteFunction deleteFunction, Model
     model
  , HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{

    session=request.getSession(false);
    ArrayList<User> userLists = new
        ArrayList<User>();
    UserSP userSP= new UserSP();

    boolean privilegeCheck=false;
    String errorMessage="";
    String currentRole="";
    AdminSP adminSP= new AdminSP();


    try {
     if(session.getAttribute("
         currentDatabaseList")==null ||
         session.getAttribute("
         sessionUserRole")==null)
      errorMessage="Session has expired. Pls
          log again.";
     else{
      String[] currentRoleList= (String[])
          session.getAttribute("
          currentDatabaseList");
      for(int i=0; i<currentRoleList.length;
          i++){
       privilegeCheck= adminSP.
           checkPrivilege(currentRoleList[i
           ], "users", "select");
       if(privilegeCheck){
        currentRole=currentRoleList[i];
        break;
       }
      }
      System.out.println(currentRole+"-"+
          privilegeCheck);
      userSP.setDatabase_username(
          currentRole);
      adminSP.setDatabase_username(
          currentRole);

      userLists= userSP.allUserList();
      }
     } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
      return "Error";
     }
     model.addAttribute("userLists",userLists
         );
     model.addAttribute("deleteFunction",
         deleteFunction);
     model.addAttribute(new User());

     return "viewAllUsers";
    }

    @RequestMapping(value = "/deleteUsers",
        method = RequestMethod.POST)
    public String deleteUsers(@ModelAttribute
        DeleteFunction deleteFunction,
        Model model, HttpServletRequest
        request, HttpServletResponse
        response,HttpSession session) throws
        Exception{

     session=request.getSession(false);

     //date time
      DateFormat dateFormat = new
          SimpleDateFormat("dd/MM/yyyy");
       //get current date time with Date()
       Date date = new Date();
       System.out.println(dateFormat.format(
           date));


       String dateString= dateFormat.format(
           date);

     //end for date time

       boolean privilegeCheck=false;
      String errorMessage="";
      String currentRole="";
      AdminSP adminSP= new AdminSP();

     int[] deleteUsers= deleteFunction.
         getDeleteObject();

     ArrayList<User> userLists = new
         ArrayList<User>();

     ArrayList<User> auditUser = new
```

```
        ArrayList<User>();
    UserSP userSP= new UserSP();
    if(session.getAttribute("
        currentDatabaseList")==null ||
        session.getAttribute("
        sessionUserRole")==null)
     errorMessage="Session has expired. Pls
         log again.";
    else{
     String[] currentRoleList= (String[])
         session.getAttribute("
         currentDatabaseList");
     for(int i=0; i<currentRoleList.length;i
         ++){
      privilegeCheck= adminSP.checkPrivilege
          (currentRoleList[i], "users", "
          delete");
      if(privilegeCheck){
       currentRole=currentRoleList[i];
       break;
      }
     }
     System.out.println(currentRole+"-"+
         privilegeCheck);
    }
     userSP.setDatabase_username(currentRole
         );
     adminSP.setDatabase_username(
         currentRole);

    for(int i=0; i<deleteUsers.length; i++){
     try {
      auditUser.add(userSP.getUser(
          deleteUsers[i]));
      userSP.deleteUser(deleteUsers[i]);

     } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     }
    }



    try {
     userSP.setDatabase_username(currentRole
         );
     userLists= userSP.allUserList();
     int sessionUserID= Integer.parseInt(
         session.getAttribute("
         sessionUserId").toString());
     User sessionUser= userSP.getUser(
         sessionUserID);
     String action_performed="Users";
     System.out.println(action_performed);
     String name= sessionUser.getFname_user
         ()+" "+ sessionUser.getMinit_user
         () + " "+ sessionUser.
         getLname_user()+" ("+ sessionUser.
         getUsername()+")";

     for(int i=0; i<deleteUsers.length; i++)
         {
      adminSP.insertAuditTrail(name, "DELETE
          ", auditUser.get(i).getUsername()
          ,"Users",dateString);

     }
     } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     }

     model.addAttribute("userLists",userLists
         );
     model.addAttribute("deleteFunction",
         deleteFunction);
     model.addAttribute(new User());

     return "viewAllUsers";
    }


    @RequestMapping(value = "/editUserForm",
         method = RequestMethod.GET)
    public String editUserForm(@RequestParam(
         value="user_id", required = true)
         int user_id,
            ModelMap model, HttpServletRequest
                request, HttpServletResponse
                response,HttpSession session
               ) throws Exception{

     session=request.getSession(false);

     //date time
      DateFormat dateFormat = new
          SimpleDateFormat("dd/MM/yyyy");
        //get current date time with Date()
        Date date = new Date();
        System.out.println(dateFormat.format(
            date));


        String dateString= dateFormat.format(
            date);

     //end for date time



      boolean privilegeCheck=false;
      String errorMessage="";
```

```
 String currentRole="";
 AdminSP adminSP= new AdminSP();


User user= new User();
String allUsernames="";
UserSP userSP= new UserSP();

try {
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
       log again.";
 else{
  String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
       i++){
   privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "users", "update");
   if(privilegeCheck){
    currentRole=currentRoleList[i];
    break;
   }
  }
  System.out.println(currentRole+"-"+
       privilegeCheck);

  userSP.setDatabase_username(
       currentRole);
  adminSP.setDatabase_username(
       currentRole);

 tempUsernames= userSP.getUsernameList()
      ;
 user=userSP.getUser(user_id);


 //audittrail
 int sessionUserID= Integer.parseInt(
      session.getAttribute("
      sessionUserId").toString());
 User sessionUser= userSP.getUser(
      sessionUserID);
 String action_performed=user.
      getUsername();
 String name= sessionUser.getFname_user
      ()+" "+ sessionUser.getMinit_user
      () + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";
 adminSP.insertAuditTrail(name, "SELECT"
      , action_performed, "Users",
      dateString);
 }
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}

for(int i=0;i<tempUsernames.size();i++)
{
      if (allUsernames == "")
       allUsernames = tempUsernames.get(
            i);
       else
        allUsernames = allUsernames + ",
             " + tempUsernames.get(i);
}
//List of secret_questions
 ArrayList<String> secret_questions= new
       ArrayList<String>();


   secret_questions.add("What is your
        mother's middle name?");
   secret_questions.add("What is your
        favorite cartoon?");
   secret_questions.add("What was your
        childhood nickname?");
   secret_questions.add("What is the
        name of your favorite childhood
        friend? ");
   secret_questions.add("What is the
        first name of your oldest niece?
        ");
   secret_questions.add("Who is your
        favorite author?");
   secret_questions.add("What is the
        title of your favorite book?");
   secret_questions.add("What was the
        last name of your favorite
        teacher?");
   secret_questions.add("What was the
        name of your first pet?");
   secret_questions.add("What is the
        name of your all-time favorite
        sports team?");

   user.setSecret_answer("");
     // ModelMap modelMap = new ModelMap
          ();
      model.put("user",user);
      model.put("allUsernames",
           allUsernames);
      model.put("secret_questions",
           secret_questions);
```

```
        return "editUserForm";
}


@RequestMapping(value = "/editUserForm",
     method = RequestMethod.POST)
public String editUserOutput(
     @ModelAttribute("user") User user,
          BindingResult result, ModelMap
               model, HttpServletRequest
               request, HttpServletResponse
                response,HttpSession
               session) throws Exception{

 session=request.getSession(false);
 //date time
  DateFormat dateFormat = new
       SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
         date));



    String dateString= dateFormat.format(
         date);

 //end for date time

    boolean privilegeCheck=false;
     String errorMessage="";
     String currentRole="";
     AdminSP adminSP= new AdminSP();

 UserSP userSP= new UserSP();
 try {
  if(session.getAttribute("
       currentDatabaseList")==null ||
       session.getAttribute("
       sessionUserRole")==null)
   errorMessage="Session has expired. Pls
        log again.";
  else{
   String[] currentRoleList= (String[])
        session.getAttribute("
        currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
        i++){
    privilegeCheck= adminSP.
         checkPrivilege(currentRoleList[i
         ], "users", "update");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   userSP.setDatabase_username(
        currentRole);
   adminSP.setDatabase_username(
        currentRole);

tempUsernames= userSP.getUsernameList()
     ;
userSP.updateUser(user.getUser_id(),
     user.getFname_user(),user.
     getMinit_user(),user.getLname_user
     (),
  user.getUsername(),user.getPassword()
       ,user.getEmail(), user.
       getSecret_question(),user.
       getSecret_answer());
try {
 /** start add new user to jbpm**/
 HumanTaskStartupServlet.taskSession.
      addUser(new org.jbpm.task.User(
      user.getUsername()));
 /**end add new user to jbpm**/
 }
 catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }

int sessionUserID= Integer.parseInt(
     session.getAttribute("
     sessionUserId").toString());
User sessionUser= userSP.getUser(
     sessionUserID);
String action_performed=user.
     getUsername();
System.out.println(action_performed);
String name= sessionUser.getFname_user
     ()+" "+ sessionUser.getMinit_user
     () + " "+ sessionUser.
     getLname_user()+" ("+ sessionUser.
     getUsername()+")";

adminSP.insertAuditTrail(name, "UPDATE"
     , action_performed,"Users",
     dateString);

String success="SUCCESS: You have
     successfully edit a user form.";
      model.put("success",success);
}
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}


      model.addAttribute("user", user);
```

```
  return "editUserForm";

}

//FOR ROLES

@RequestMapping(value = "/editRoleForm",
     method = RequestMethod.GET)
public String editRoleForm(@RequestParam(
     value="role_id", required = true)
     int role_id, ModelMap model,
  HttpServletRequest request,
       HttpServletResponse response,
       HttpSession session) throws
       Exception{

 session=request.getSession(false);

 //date time
  DateFormat dateFormat = new
       SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
         date));



    String dateString= dateFormat.format(
         date);

 //end for date time

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();
 UserSP userSP= new UserSP();
 Role role= new Role();
 String allRolenames="";
 RoleSP RoleSP= new RoleSP();
 System.out.println("Toot: "+role_id);
 try {
  if(session.getAttribute("
       currentDatabaseList")==null ||
       session.getAttribute("
       sessionUserRole")==null)
   errorMessage="Session has expired. Pls
        log again.";
  else{
   String[] currentRoleList= (String[])
        session.getAttribute("
        currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
        i++){
    privilegeCheck= adminSP.
         checkPrivilege(currentRoleList[i
         ], "role", "update");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
        privilegeCheck);
   RoleSP.setDatabase_username(
        currentRole);
   userSP.setDatabase_username(
        currentRole);
   adminSP.setDatabase_username(
        currentRole);
  tempRolenames= RoleSP.listAllRoles();
  String database_role= RoleSP.
       getDatabaseRole(role_id);
  System.out.println(database_role);
  role=RoleSP.getRole(role_id);
  role.setDatabase_role(database_role);

  int sessionUserID= Integer.parseInt(
       session.getAttribute("
       sessionUserId").toString());
  User sessionUser= userSP.getUser(
       sessionUserID);
  String action_performed=role.
       getRole_name();
  String name= sessionUser.getFname_user
       ()+" "+ sessionUser.getMinit_user
       () + " "+ sessionUser.
       getLname_user()+" ("+ sessionUser.
       getUsername()+")";
  adminSP.insertAuditTrail(name, "SELECT"
       , action_performed, "Role",
       dateString);
  }
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }

for(int i=0;i<tempRolenames.size();i++)
{
       if (allRolenames == "")
        allRolenames = tempRolenames.get(
             i);
        else
         allRolenames = allRolenames + ",
              " + tempRolenames.get(i);
}

      // ModelMap modelMap = new ModelMap
           ();
       model.put("role",role);
```

```
        model.put("allRolenames",
            allRolenames);

        return "editRoleForm";
}

//SAVE EDIT RESULT

@RequestMapping(value = "/editRoleForm",
    method = RequestMethod.POST)
public String editUserOutput(
    @ModelAttribute("role") Role role,
        BindingResult result, ModelMap
            model,
        HttpServletRequest request,
            HttpServletResponse response
            ,HttpSession session) throws
             Exception{

 session=request.getSession(false);

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();
 //date time
  DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
        date));


    String dateString= dateFormat.format(
        date);

 //end for date time

 RoleSP roleSP= new RoleSP();
 UserSP userSP= new UserSP();
 try {
  if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
   errorMessage="Session has expired. Pls
       log again.";
  else{
   String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
       i++){
    privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "role", "update");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
       privilegeCheck);
   roleSP.setDatabase_username(
       currentRole);
   userSP.setDatabase_username(
       currentRole);
   adminSP.setDatabase_username(
       currentRole);

tempRolenames= roleSP.listAllRoles();
roleSP.updateRole(role.getRole_id(),
    role.getRole_name(),role.
    getRole_description(),role.
    getDatabase_role());

try{
 /** start add new group to jbpm**/
 HumanTaskStartupServlet.taskSession.
     addGroup(new org.jbpm.task.Group(
     role.getRole_name()));
 /**end add new group to jbpm**/
 }
 catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }
int sessionUserID= Integer.parseInt(
    session.getAttribute("
    sessionUserId").toString());
User sessionUser= userSP.getUser(
    sessionUserID);
String action_performed=role.
    getRole_name();
System.out.println(action_performed);
String name= sessionUser.getFname_user
    ()+" "+ sessionUser.getMinit_user
    () + " "+ sessionUser.
    getLname_user()+" ("+ sessionUser.
    getUsername()+")";
adminSP.insertAuditTrail(name, "UPDATE"
    , action_performed,"Role",
    dateString);

String success="SUCCESS: You have
    successfully edit a role.";
     model.put("success",success);
}
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
```

```
 }


        model.addAttribute("role", role);

  return "editRoleForm";

}

//VIEW
@RequestMapping(value = "/viewAllRoles.
     html", method = RequestMethod.GET)
public String viewRoles(@ModelAttribute
     DeleteFunction deleteFunction, Model
      model
  , HttpServletRequest request,
       HttpServletResponse response,
       HttpSession session) throws
       Exception{

 session=request.getSession(false);

 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();

 ArrayList<Role> roleLists = new
      ArrayList<Role>();
 RoleSP roleSP= new RoleSP();

 try {
  if(session.getAttribute("
       currentDatabaseList")==null ||
       session.getAttribute("
       sessionUserRole")==null)
   errorMessage="Session has expired. Pls
        log again.";
  else{
   String[] currentRoleList= (String[])
        session.getAttribute("
        currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
        i++){
    privilegeCheck= adminSP.
         checkPrivilege(currentRoleList[i
         ], "role", "select");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
        privilegeCheck);
   roleSP.setDatabase_username(
        currentRole);
   roleLists= roleSP.allRoleList();
   for(Role roleList: roleLists){
    String database_role= roleSP.
         getDatabaseRole(roleList.
         getRole_id());
    roleList.setDatabase_role(
         database_role);
   }
   }
   } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   return "Error";
  }

  model.addAttribute("roleLists",roleLists
       );
  model.addAttribute("deleteFunction",
       deleteFunction);
  model.addAttribute(new Role());

  return "viewAllRoles";
 }


@RequestMapping(value = "/deleteRoles",
     method = RequestMethod.POST)
public String deleteRoles(@ModelAttribute
      DeleteFunction deleteFunction,
     Model model,
  HttpServletRequest request,
       HttpServletResponse response,
       HttpSession session) throws
       Exception{

 session=request.getSession(false);
 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();
 //date time
  DateFormat dateFormat = new
       SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
         date));


    String dateString= dateFormat.format(
         date);

 //end for date time
 int[] deleteRoles= deleteFunction.
      getDeleteObject();
```

```
ArrayList<Role> auditRoles = new
    ArrayList<Role>();
ArrayList<Role> roleLists = new
    ArrayList<Role>();
RoleSP roleSP= new RoleSP();
UserSP userSP= new UserSP();
if(session.getAttribute("
    currentDatabaseList")==null ||
    session.getAttribute("
    sessionUserRole")==null)
 errorMessage="Session has expired. Pls
     log again.";
else{
 String[] currentRoleList= (String[])
     session.getAttribute("
     currentDatabaseList");
 for(int i=0; i<currentRoleList.length;i
     ++){
  privilegeCheck= adminSP.checkPrivilege
      (currentRoleList[i], "role", "
      delete");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
     privilegeCheck);
 roleSP.setDatabase_username(currentRole
     );
 adminSP.setDatabase_username(
     currentRole);


}
for(int i=0; i<deleteRoles.length; i++){
 try {
  auditRoles.add(roleSP.getRole(
      deleteRoles[i]));
  roleSP.deleteRoles(deleteRoles[i]);
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }
}



try {

 roleLists= roleSP.allRoleList();
 for(Role roleList: roleLists){
  String database_role= roleSP.
      getDatabaseRole(roleList.
      getRole_id());
  roleList.setDatabase_role(
      database_role);
 }

 int sessionUserID= Integer.parseInt(
     session.getAttribute("
     sessionUserId").toString());
 User sessionUser= userSP.getUser(
     sessionUserID);
 String action_performed="Role";
 System.out.println(action_performed);
 String name= sessionUser.getFname_user
     ()+" "+ sessionUser.getMinit_user
     () + " "+ sessionUser.
     getLname_user()+" ("+ sessionUser.
     getUsername()+")";
 for(int i=0; i<deleteRoles.length; i++)
     {
 adminSP.insertAuditTrail(name, "DELETE"
     , auditRoles.get(i).getRole_name()
     ,"Role", dateString);
 }

 } catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}

 model.addAttribute("roleLists",roleLists
     );
 model.addAttribute("deleteFunction",
     deleteFunction);
 model.addAttribute(new Role());

 return "viewAllRoles";
}

//role form
@RequestMapping(value = "/sectionForm.
    html", method = RequestMethod.GET)
public String sectionForm(Model model,
    HttpServletRequest request,
    HttpServletResponse response,
    HttpSession session) throws
    Exception{

 session=request.getSession(false);
 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();

 String allSectionnames="";
 SectionSP sectionSP= new SectionSP();
 try {
  if(session.getAttribute("
      currentDatabaseList")==null ||
```

```
        session.getAttribute("
        sessionUserRole")==null)
    errorMessage="Session has expired. Pls
        log again.";
   else{
    String[] currentRoleList= (String[])
        session.getAttribute("
        currentDatabaseList");
    for(int i=0; i<currentRoleList.length;
        i++){
     privilegeCheck= adminSP.
         checkPrivilege(currentRoleList[i
         ], "section", "insert");
     if(privilegeCheck){
      currentRole=currentRoleList[i];
      break;
     }
    }
    System.out.println(currentRole+"-"+
        privilegeCheck);

    sectionSP.setDatabase_username(
        currentRole);
    adminSP.setDatabase_username(
        currentRole);

   tempSectionnames= sectionSP.
       listAllSection();



   }
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   return "Error";
  }

  for(int i=0;i<tempSectionnames.size();i
      ++)
  {
        if (allSectionnames == "")
         allSectionnames =
             tempSectionnames.get(i);
         else
          allSectionnames =
              allSectionnames + "," +
              tempSectionnames.get(i);
  }

 model.addAttribute("allSectionnames",
     allSectionnames);
 model.addAttribute(new Section());
  return "sectionForm";

 }
```

```
 //section form result
 @RequestMapping(value = "/sectionoutput.
     html", method = RequestMethod.POST)

 public String sectionResult(
     @ModelAttribute Section section,
     Model model, HttpServletRequest
     request, HttpServletResponse
     response,HttpSession session) throws
      Exception{

  session=request.getSession(false);

  //date time
   DateFormat dateFormat = new
       SimpleDateFormat("dd/MM/yyyy");
     //get current date time with Date()
     Date date = new Date();
     System.out.println(dateFormat.format(
         date));



     String dateString= dateFormat.format(
         date);

  //end for date time


     section.setCreated_by("Admin");
     section.setCreated_date(dateString);

  //start database connection by using
       datasource



     boolean privilegeCheck=false;
      String errorMessage="";
      String currentRole="";
      AdminSP adminSP= new AdminSP();
  SectionSP sectionSP= new SectionSP();
  UserSP userSP= new UserSP();
  try {
   if(session.getAttribute("
       currentDatabaseList")==null ||
       session.getAttribute("
       sessionUserRole")==null)
    errorMessage="Session has expired. Pls
        log again.";
   else{
    String[] currentRoleList= (String[])
        session.getAttribute("
        currentDatabaseList");
    for(int i=0; i<currentRoleList.length;
        i++){
     privilegeCheck= adminSP.
```

```
            checkPrivilege(currentRoleList[i
            ], "section", "insert");
      if(privilegeCheck){
       currentRole=currentRoleList[i];
       break;
      }
     }
     System.out.println(currentRole+"-"+
          privilegeCheck);

     sectionSP.setDatabase_username(
          currentRole);
     userSP.setDatabase_username(
          currentRole);
     adminSP.setDatabase_username(
          currentRole);

    sectionSP.executeSection(section.
         getSection_name(), section.
         getSection_description(),section.
         getCreated_by(),section.
         getCreated_date());

    int sessionUserID= Integer.parseInt(
         session.getAttribute("
         sessionUserId").toString());
      User sessionUser= userSP.getUser(
           sessionUserID);
      String action_performed=section.
           getSection_name();
      System.out.println(action_performed);
      String name= sessionUser.
           getFname_user()+" "+ sessionUser
           .getMinit_user() + " "+
           sessionUser.getLname_user()+" ("
           + sessionUser.getUsername()+")";
      adminSP.insertAuditTrail(name, "
           INSERT", action_performed,"
           Section" ,dateString);
    }
    } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }

   model.addAttribute("section",section);

   return "sectionoutput";
}


//VIEW
@RequestMapping(value = "/viewAllSections
     .html", method = RequestMethod.GET)
public String viewSections(
     @ModelAttribute DeleteFunction
     deleteFunction, Model model,
     HttpServletRequest request,
     HttpServletResponse response,
     HttpSession session) throws
     Exception{

session=request.getSession(false);
ArrayList<Section> sectionLists = new
     ArrayList<Section>();
SectionSP sectionSP= new SectionSP();


boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();
try {
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
       log again.";
 else{
  String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
       i++){
   privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "section", "select");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
        privilegeCheck);

   sectionSP.setDatabase_username(
        currentRole);
   adminSP.setDatabase_username(
        currentRole);

  sectionLists= sectionSP.allSectionList
       ();
 }
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 return "Error";
}

model.addAttribute("sectionLists",
     sectionLists);
```

```
 model.addAttribute("deleteFunction",
      deleteFunction);
 model.addAttribute(new Section());

 return "viewAllSections";
}


@RequestMapping(value = "/deleteSections"
     , method = RequestMethod.POST)
public String deleteSection(
     @ModelAttribute DeleteFunction
     deleteFunction, Model model,
     HttpServletRequest request,
     HttpServletResponse response,
     HttpSession session) throws
     Exception{

 session=request.getSession(false);

 //date time
  DateFormat dateFormat = new
       SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
         date));



    String dateString= dateFormat.format(
         date);

 //end for date time

 int[] deleteSection= deleteFunction.
      getDeleteObject();


 boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";
  AdminSP adminSP= new AdminSP();

  ArrayList<Section> auditSection = new
       ArrayList<Section>();

 ArrayList<Section> sectionLists = new
      ArrayList<Section>();
 SectionSP sectionSP= new SectionSP();
 UserSP userSP= new UserSP();
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
       log again.";
else{
 String[] currentRoleList= (String[])
      session.getAttribute("
      currentDatabaseList");
 for(int i=0; i<currentRoleList.length;i
      ++){
  privilegeCheck= adminSP.checkPrivilege
       (currentRoleList[i], "section", "
       delete");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
      privilegeCheck);

sectionSP.setDatabase_username(
     currentRole);
 adminSP.setDatabase_username(
      currentRole);



for(int i=0; i<deleteSection.length; i
     ++){
 try {
  auditSection.add(sectionSP.getSection(
       deleteSection[i]));
  sectionSP.deleteSections(deleteSection
       [i]);
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }
}
}

try {


 sectionLists= sectionSP.allSectionList
      ();

 int sessionUserID= Integer.parseInt(
      session.getAttribute("
      sessionUserId").toString());
   User sessionUser= userSP.getUser(
        sessionUserID);
   String action_performed="Section";
   System.out.println(action_performed);
   String name= sessionUser.
        getFname_user()+" "+ sessionUser
        .getMinit_user() + " "+
        sessionUser.getLname_user()+" ("
        + sessionUser.getUsername()+")";
```

```
    for(int i=0; i<deleteSection.length;
        i++){
    adminSP.insertAuditTrail(name, "
        DELETE", auditSection.get(i).
        getSection_name(),"Section",
        dateString);

    }
  } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }

 model.addAttribute("sectionLists",
     sectionLists);
 model.addAttribute("deleteFunction",
     deleteFunction);
 model.addAttribute(new Role());

 return "viewAllSections";
}


//FOR SECTIONS

@RequestMapping(value = "/editSectionForm
    ", method = RequestMethod.GET)
public String editSectionForm(
    @RequestParam(value="section_id",
    required = true) int section_id,
    ModelMap model,
  HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{

 session=request.getSession(false);

 //date time
  DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
    //get current date time with Date()
    Date date = new Date();
    System.out.println(dateFormat.format(
        date));



    String dateString= dateFormat.format(
        date);

 //end for date time

 boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";
  AdminSP adminSP= new AdminSP();
ArrayList<Role> currentRoleLists= new
    ArrayList<Role>();
ArrayList<ArrayList<User>>
    currentUserLists= new ArrayList<
    ArrayList<User>>();
DeleteFunction deleteFunction= new
    DeleteFunction();
Section section= new Section();
String allSectionnames="";
SectionSP sectionSP= new SectionSP();
RoleSP roleSP= new RoleSP();
RoleSection roleSection= new RoleSection
    ();
UserSP userSP= new UserSP();
deleteFunction.setKey_id(section_id);
try {
 if(session.getAttribute("
     currentDatabaseList")==null ||
     session.getAttribute("
     sessionUserRole")==null)
  errorMessage="Session has expired. Pls
      log again.";
 else{
  String[] currentRoleList= (String[])
      session.getAttribute("
      currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
      i++){
   privilegeCheck= adminSP.
       checkPrivilege(currentRoleList[i
       ], "section", "delete");
   if(privilegeCheck){
    currentRole=currentRoleList[i];
    break;
   }
  }
  System.out.println(currentRole+"-"+
      privilegeCheck);
  sectionSP.setDatabase_username(
      currentRole);
  roleSP.setDatabase_username(
      currentRole);
  adminSP.setDatabase_username(
      currentRole);
  userSP.setDatabase_username(
      currentRole);
  currentRoleLists= sectionSP.
      getListRoleSection(section_id);
  tempSectionnames= sectionSP.
      listAllSection();
  section=sectionSP.getSection(section_id
      );
  allRoles= roleSP.listAllRoles();

  for (Role currentRoleListss :
```

```
        currentRoleLists) {
    currentUserLists.add(sectionSP.
        getListUserSection(
        currentRoleListss.getRole_id()));
  }

  int sessionUserID= Integer.parseInt(
        session.getAttribute("
        sessionUserId").toString());
  User sessionUser= userSP.getUser(
        sessionUserID);
  String action_performed=section.
        getSection_name();
  System.out.println(action_performed);
  String name= sessionUser.getFname_user
        ()+"_"+ sessionUser.getMinit_user
        () + "_"+ sessionUser.
        getLname_user()+"_("+ sessionUser.
        getUsername()+")";
  adminSP.insertAuditTrail(name, "SELECT"
        , action_performed ,"Section" ,
        dateString);


  }
} catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
}

for(int i=0;i<tempSectionnames.size();i
    ++)
{
       if (allSectionnames == "")
        allSectionnames =
            tempSectionnames.get(i);
        else
         allSectionnames =
             allSectionnames + "," +
             tempSectionnames.get(i);
}
roleSection.setSection_id(section.
    getSection_id());

  model.addAttribute("deleteFunction",
        deleteFunction);
  model.addAttribute("section",section);
  model.addAttribute("roleSection",
        roleSection);
    model.addAttribute("allSectionnames"
        ,allSectionnames);
    model.addAttribute("allRoles",
        allRoles);
    model.addAttribute("currentUserLists
        ",currentUserLists);
    model.addAttribute("currentRoleLists
        ", currentRoleLists);
      return "editSectionForm";
}

//ADD ROLE TO SECTION

@RequestMapping(value = "/addSectionRole"
    , method = RequestMethod.GET)
public String addRoleSections(
    @RequestParam(value="section_id",
    required = true) int section_id,
    @ModelAttribute DeleteFunction
    deleteFunction, @ModelAttribute("
    roleSection") RoleSection
    roleSection, ModelMap model
  , HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{

 session=request.getSession(false);

 boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";
  AdminSP adminSP= new AdminSP();


 ArrayList<Role> currentRoleLists= new
     ArrayList<Role>();
 Section section= new Section();
 ArrayList<ArrayList<User>>
     currentUserLists= new ArrayList<
     ArrayList<User>>();
 SectionSP sectionSP= new SectionSP();
 RoleSP roleSP= new RoleSP();
 deleteFunction.setKey_id(section_id);

 try {

  if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
   errorMessage="Session_has_expired._Pls
       _log_again.";
  else{
   String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
       i++){
    privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "role_section", "insert");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
```

```
            break;
          }
        }
        System.out.println(currentRole+"-"+
            privilegeCheck);
        sectionSP.setDatabase_username(
            currentRole);
        roleSP.setDatabase_username(
            currentRole);
        adminSP.setDatabase_username(
            currentRole);

      section=sectionSP.getSection(section_id
          );
      //sectionSP.insertRoleSection(
          roleSection.getSection_id(),
          roleSection.getRole_name());
      currentRoleLists= sectionSP.
          getListRoleSection(section_id);
      allRoles= roleSP.listAllRoles();
      for (Role currentRoleListss :
          currentRoleLists) {
       currentUserLists.add(sectionSP.
          getListUserSection(
          currentRoleListss.getRole_id()));
      }
     }
     } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     }


        model.addAttribute("deleteFunction",
            deleteFunction);
        model.addAttribute("section",section);
        model.addAttribute("roleSection",
            roleSection);
          model.addAttribute("allRoles",
              allRoles);
          model.addAttribute("
              currentUserLists",
              currentUserLists);
          model.addAttribute("
              currentRoleLists",
              currentRoleLists);
      return "addSectionRole";

    }


    //SAVE EDIT RESULT

    @RequestMapping(value = "/editSectionForm
        ", method = RequestMethod.POST)
    public String editSectionOutput(
        @ModelAttribute("section") Section
        section,
          BindingResult result, ModelMap
              model,
          HttpServletRequest request,
              HttpServletResponse response,
              HttpSession session) throws
              Exception{

    boolean privilegeCheck=false;
     String errorMessage="";
     String currentRole="";
     AdminSP adminSP= new AdminSP();
    session=request.getSession(false);
    //date time
     DateFormat dateFormat = new
         SimpleDateFormat("dd/MM/yyyy");
       //get current date time with Date()
       Date date = new Date();
       System.out.println(dateFormat.format(
            date));



       String dateString= dateFormat.format(
            date);

    //end for date time

    String allSectionnames="";
    SectionSP sectionSP= new SectionSP();
    UserSP userSP= new UserSP();

    try {
     if(session.getAttribute("
         currentDatabaseList")==null ||
         session.getAttribute("
         sessionUserRole")==null)
      errorMessage="Session has expired. Pls
          log again.";
     else{
      String[] currentRoleList= (String[])
          session.getAttribute("
          currentDatabaseList");
      for(int i=0; i<currentRoleList.length;
          i++){
       privilegeCheck= adminSP.
           checkPrivilege(currentRoleList[i
           ], "section", "update");
       if(privilegeCheck){
        currentRole=currentRoleList[i];
        break;
       }
      }
      System.out.println(currentRole+"-"+
          privilegeCheck);
      sectionSP.setDatabase_username(
```

```
        currentRole);
    adminSP.setDatabase_username(
        currentRole);
    userSP.setDatabase_username(
        currentRole);

  tempSectionnames= sectionSP.
      listAllSection();
  sectionSP.updateSection(section.
      getSection_id(),section.
      getSection_name(),section.
      getSection_description());



  int sessionUserID= Integer.parseInt(
      session.getAttribute("
      sessionUserId").toString());
    User sessionUser= userSP.getUser(
        sessionUserID);
    String action_performed=section.
        getSection_name();
    System.out.println(action_performed);
    String name= sessionUser.
        getFname_user()+" "+ sessionUser
        .getMinit_user() + " "+
        sessionUser.getLname_user()+" ("
        + sessionUser.getUsername()+")";
    adminSP.insertAuditTrail(name, "
        UPDATE", action_performed,"
        role_section" ,dateString);


    String success="SUCCESS: You have
        successfully edited a section
        form.";
          model.put("success",success);
  }
  } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }
  model.addAttribute("section",section);
     model.addAttribute("allSectionnames"
         ,allSectionnames);

  return "editSectionForm";

}



@RequestMapping(value = "/
    deleteRoleSection.html", method =
    RequestMethod.POST)
public String deleteRoleSection(
    @ModelAttribute DeleteFunction
    deleteFunction,@ModelAttribute("
    section") Section section,
    @ModelAttribute("roleSection")
    RoleSection roleSection,
      BindingResult result, ModelMap
          model
      , HttpServletRequest request,
          HttpServletResponse response,
          HttpSession session) throws
          Exception{

session=request.getSession(false);

boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 AdminSP adminSP= new AdminSP();

 //date time
  DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
   //get current date time with Date()
   Date date = new Date();
   System.out.println(dateFormat.format
       (date));



   String dateString= dateFormat.format
       (date);

 //end for date time

 ArrayList<Role> currentRoleLists= new
     ArrayList<Role>();

ArrayList<Role> auditRole= new ArrayList
    <Role>();

ArrayList<ArrayList<User>>
    currentUserLists= new ArrayList<
    ArrayList<User>>();

SectionSP sectionSP= new SectionSP();
RoleSP roleSP= new RoleSP();
UserSP userSP= new UserSP();
if(session.getAttribute("
    currentDatabaseList")==null ||
    session.getAttribute("
    sessionUserRole")==null)
 errorMessage="Session has expired. Pls
     log again.";
else{
 String[] currentRoleList= (String[])
     session.getAttribute("
     currentDatabaseList");
 for(int i=0; i<currentRoleList.length;i
     ++){
```

```
 privilegeCheck= adminSP.checkPrivilege
      (currentRoleList[i], "
      role_section", "delete");
 if(privilegeCheck){
  currentRole=currentRoleList[i];
  break;
 }
}
System.out.println(currentRole+"-"+
     privilegeCheck);
sectionSP.setDatabase_username(
     currentRole);
roleSP.setDatabase_username(currentRole
     );
adminSP.setDatabase_username(
     currentRole);


int[] deleteRoles= deleteFunction.
    getDeleteObject();
int section_id=deleteFunction.getKey_id
    ();
try {

 for(int i=0; i<deleteRoles.length; i++)
     {
  try {
   auditRole.add(roleSP.getRole(
       deleteRoles[i]));
   sectionSP.deleteRoleSections(
       deleteRoles[i]);
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
 roleSection.setSection_id(section_id);

 System.out.println("BUMP_:"+section_id)
     ;
 section=sectionSP.getSection(section_id
     );
 currentRoleLists= sectionSP.
     getListRoleSection(section_id);
 allRoles= roleSP.listAllRoles();

 for (Role currentRoleListss :
     currentRoleLists) {
  currentUserLists.add(sectionSP.
      getListUserSection(
      currentRoleListss.getRole_id()));
 }
 int sessionUserID= Integer.parseInt(
     session.getAttribute("
     sessionUserId").toString());
 User sessionUser= userSP.getUser(
     sessionUserID);
 String action_performed="("+section.
     getSection_name()+")";
 System.out.println(action_performed);
 String name= sessionUser.getFname_user
     ()+"_"+ sessionUser.getMinit_user
     () + "_"+ sessionUser.
     getLname_user()+"_("+ sessionUser.
     getUsername()+")";


 for(int i=0; i<deleteRoles.length; i++)
     {
 adminSP.insertAuditTrail(name, "DELETE"
     , auditRole.get(i).getRole_name()+
     "_"+action_performed,"role_section
     " ,dateString);

 }
} catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}

}
 model.addAttribute("deleteFunction",
     deleteFunction);
 model.addAttribute("section",section);
 model.addAttribute("roleSection",
     roleSection);
    model.addAttribute("allRoles",
        allRoles);
    model.addAttribute("currentUserLists
        ",currentUserLists);
    model.addAttribute("currentRoleLists
        ", currentRoleLists);
 return "addSectionRole";


}


@RequestMapping(value = "/addSectionRole"
    , method = RequestMethod.POST)
public String addRoleSections(
    @ModelAttribute DeleteFunction
    deleteFunction, @ModelAttribute("
    roleSection") RoleSection
    roleSection, ModelMap model,
  HttpServletRequest request,
      HttpServletResponse response,
      HttpSession session) throws
      Exception{


 session=request.getSession(false);
```

```
//date time
 DateFormat dateFormat = new
      SimpleDateFormat("dd/MM/yyyy");
   //get current date time with Date()
   Date date = new Date();
   System.out.println(dateFormat.format(
        date));


   String dateString= dateFormat.format(
        date);

//end for date time


  boolean privilegeCheck=false;
  String errorMessage="";
   String currentRole="";
   AdminSP adminSP= new AdminSP();

ArrayList<Role> currentRoleLists= new
     ArrayList<Role>();
Section section= new Section();
ArrayList<ArrayList<User>>
     currentUserLists= new ArrayList<
     ArrayList<User>>();
SectionSP sectionSP= new SectionSP();
RoleSP roleSP= new RoleSP();
deleteFunction.setKey_id(roleSection.
     getSection_id());
UserSP userSP= new UserSP();

try {
 if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
  errorMessage="Session has expired. Pls
       log again.";
 else{
  String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
  for(int i=0; i<currentRoleList.length;
       i++){
   privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "role_section", "insert");
   if(privilegeCheck){
    currentRole=currentRoleList[i];
    break;
   }
  }
  System.out.println(currentRole+"-"+
       privilegeCheck);
  sectionSP.setDatabase_username(
       currentRole);
  roleSP.setDatabase_username(
       currentRole);
  adminSP.setDatabase_username(
       currentRole);
  userSP.setDatabase_username(
       currentRole);


 sectionSP.insertRoleSection(roleSection
      .getSection_id(),roleSection.
      getRole_name());
 section=sectionSP.getSection(
      roleSection.getSection_id());
 currentRoleLists= sectionSP.
      getListRoleSection(roleSection.
      getSection_id());
 allRoles= roleSP.listAllRoles();
 for (Role currentRoleListss :
      currentRoleLists) {
  currentUserLists.add(sectionSP.
       getListUserSection(
       currentRoleListss.getRole_id()));
 }
 int sessionUserID= Integer.parseInt(
      session.getAttribute("
      sessionUserId").toString());
 User sessionUser= userSP.getUser(
      sessionUserID);
 String action_performed=roleSection.
      getRole_name()+" ("+section.
      getSection_name()+")";
 System.out.println(action_performed);
 String name= sessionUser.getFname_user
      ()+" "+ sessionUser.getMinit_user
      () + " "+ sessionUser.
      getLname_user()+" ("+ sessionUser.
      getUsername()+")";
 adminSP.insertAuditTrail(name, "INSERT"
      , action_performed,"role_section",
      dateString);

 String success="SUCCESS: You have
      successfully added a role in a
      section.";
       model.put("success",success);
 }
 } catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}


  model.addAttribute("deleteFunction",
       deleteFunction);
  model.addAttribute("section",section);
  model.addAttribute("roleSection",
```

```
        roleSection);
      model.addAttribute("allRoles",
          allRoles);
      model.addAttribute("
          currentUserLists",
          currentUserLists);
      model.addAttribute("
          currentRoleLists",
          currentRoleLists);
  return "addSectionRole";

}


@RequestMapping(value = "/viewUserRole",
    method = RequestMethod.GET)
public String viewUserRole(@RequestParam(
    value="role_id", required = true)
    int role_id,
      Model model,
      HttpServletRequest request,
          HttpServletResponse response,
          HttpSession session) throws
          Exception{

 session=request.getSession(false);

 boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";
  AdminSP adminSP= new AdminSP();
 ArrayList<User> currentUserLists= new
     ArrayList<User>();
 Role role= new Role();
 RoleSP roleSP= new RoleSP();

 try {
  if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
      sessionUserRole")==null)
   errorMessage="Session has expired. Pls
       log again.";
  else{
   String[] currentRoleList= (String[])
       session.getAttribute("
       currentDatabaseList");
   for(int i=0; i<currentRoleList.length;
       i++){
    privilegeCheck= adminSP.
        checkPrivilege(currentRoleList[i
        ], "role", "select");
    if(privilegeCheck){
     currentRole=currentRoleList[i];
     break;
    }
   }
   System.out.println(currentRole+"-"+
       privilegeCheck);
   roleSP.setDatabase_username(
       currentRole);
   adminSP.setDatabase_username(
       currentRole);

  currentUserLists= roleSP.
      getListUserRole(role_id);
  role= roleSP.getRole(role_id);
  }
 } catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }

 model.addAttribute("currentUserLists",
     currentUserLists);
 model.addAttribute("role", role);
  return "viewUserRole";

}


@RequestMapping(value = "/
    viewSpecificedUsers.html", method =
    RequestMethod.GET)
public String viewSpecificUsers(
    HttpServletRequest request,
        HttpServletResponse response,
            Model model,HttpSession
            session) throws Exception {

 session=request.getSession(false);


  boolean privilegeCheck=false;
  String errorMessage="";
  String currentRole="";
  AdminSP adminSP= new AdminSP();

 DeleteFunction deleteFunction = new
     DeleteFunction();
 UserSP userSP= new UserSP();
 String nameSearch= "%"+request.
     getParameter("username")+"%";
 System.out.println("CHECK:: "+nameSearch
     );
 ArrayList<User> userLists = new
     ArrayList<User>();

 try {
  if(session.getAttribute("
      currentDatabaseList")==null ||
      session.getAttribute("
```

```
        sessionUserRole")==null)
    errorMessage="Session has expired. Pls
        log again.";
   else{
    String[] currentRoleList= (String[])
        session.getAttribute("
        currentDatabaseList");
    for(int i=0; i<currentRoleList.length;
        i++){
     privilegeCheck= adminSP.
         checkPrivilege(currentRoleList[i
         ], "role", "select");
     if(privilegeCheck){
      currentRole=currentRoleList[i];
      break;
     }
    }
    System.out.println(currentRole+"-"+
        privilegeCheck);
    userSP.setDatabase_username(
        currentRole);
    adminSP.setDatabase_username(
        currentRole);

   userLists= userSP.specifiedUserList(
       nameSearch);
   }
  } catch (Exception e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   return "Error";
  }
  model.addAttribute("userLists",userLists
      );
  model.addAttribute(new User());
   model.addAttribute("deleteFunction",
       deleteFunction);


  return "viewAllUsers";

}



/*
 *
 * VIEWING FOR FACULTY
 *
 */


@RequestMapping(value = "/
    viewFacultySection.html", method =
    RequestMethod.GET)
public String viewFacultySection(Model
    model, HttpServletRequest request,
    HttpServletResponse response,
    HttpSession session) throws
    Exception {

session=request.getSession(false);
boolean privilegeCheck=false;
String errorMessage="";
String currentRole="";
ArrayList<UserRoleSection> currentLists=
     new ArrayList<UserRoleSection>();

ArrayList<ArrayList<User>>
    currentUserLists= new ArrayList<
    ArrayList<User>>();
ArrayList<Role> currentRoleLists= new
    ArrayList<Role>();
ArrayList<Section> currentSectionLists=
    new ArrayList<Section>();

AdminSP adminSP= new AdminSP();
UserSP userSP= new UserSP();
RoleSP roleSP= new RoleSP();
SectionSP sectionSP= new SectionSP();
try{
if(session.getAttribute("
    currentDatabaseList")==null ||
    session.getAttribute("
    sessionUserRole")==null)
 errorMessage="Session has expired. Pls
     log again.";
else{
 String[] currentRoleList= (String[])
     session.getAttribute("
     currentDatabaseList");
 for(int i=0; i<currentRoleList.length;i
     ++){
  privilegeCheck= adminSP.checkPrivilege
      (currentRoleList[i], "section", "
      select");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
     privilegeCheck);

 userSP.setDatabase_username(currentRole
     );
 roleSP.setDatabase_username(currentRole
     );
 sectionSP.setDatabase_username(
     currentRole);
 adminSP.setDatabase_username(
     currentRole);
```

```
currentSectionLists= sectionSP.
     getListSectionsOfUser(Integer.
     parseInt(session.getAttribute("
     sessionUserId").toString()));
currentRoleLists= sectionSP.
     getListRoleSection(
     currentSectionLists.get(0).
     getSection_id());
allRoles= roleSP.listAllRoles();
for (Role currentRoleListss :
     currentRoleLists) {
 currentUserLists.add(sectionSP.
      getListUserSection(
      currentRoleListss.getRole_id()));
}
for(ArrayList<User> currentUserList:
     currentUserLists){
 for(User currentUser: currentUserList)
     {
  System.out.println("OOPS:_"+
       currentUser.getUsername());

  UserRoleSection tempUserRoleSection=
       new UserRoleSection();

  tempUserRoleSection.setUser_name(
       currentUser.getUsername());
  tempUserRoleSection.setFull_name(
       currentUser.getFname_user()+"_"+
       currentUser.getLname_user());

  //for user_role
  String role="";
  ArrayList<Role> tempRole= new
       ArrayList<Role>();
  tempRole=userSP.getListRolesofUser(
       currentUser.getUser_id());
  for(Role checkRole: tempRole){
   if(checkRole.getRole_name().
        toLowerCase().indexOf("student"
        )!=-1){
    role="Student_Clinician";
    break;
   }
   else if(checkRole.getRole_name().
        toLowerCase().indexOf("faculty"
        )!=-1){
    role="Faculty";
    break;
   }
   else{
    role="Administrator";
   }

  }
    tempUserRoleSection.setUser_role(role
         );

    currentLists.add(tempUserRoleSection)
         ;
   }
  }

 }
 } catch (Exception e) {
  // TODO Auto-generated catch block
  System.out.println("ERROR");
  e.printStackTrace();
 }
 model.addAttribute("currentLists",
      currentLists);
 model.addAttribute("userRoleSection",new
       UserRoleSection());
 model.addAttribute("currentSectionLists"
      ,currentSectionLists);

 return "viewFacultySection";

}

@RequestMapping(value = "/
     viewSpecificFacultySection.html",
     method = RequestMethod.GET)
public String searchFacultySection(Model
     model, HttpServletRequest request,
     HttpServletResponse response,
   HttpSession session) throws Exception {

 session=request.getSession(false);
 boolean privilegeCheck=false;
 String errorMessage="";
 String currentRole="";
 ArrayList<UserRoleSection> currentLists=
      new ArrayList<UserRoleSection>();

 ArrayList<ArrayList<User>>
      currentUserLists= new ArrayList<
      ArrayList<User>>();
 ArrayList<Role> currentRoleLists= new
      ArrayList<Role>();
 ArrayList<Section> currentSectionLists=
      new ArrayList<Section>();

 AdminSP adminSP= new AdminSP();
 UserSP userSP= new UserSP();
 RoleSP roleSP= new RoleSP();
 SectionSP sectionSP= new SectionSP();

 String searchString= "";

 if(request.getParameter("user_name").
```

```
    isEmpty() || request.getParameter("
    user_name") !=null)
 searchString=request.getParameter("
     user_name");

int section_id =Integer.parseInt(request
    .getParameter("section_id"));
String pickUserRole =request.
    getParameter("user_role").toString
    ();
System.out.println(pickUserRole);
try{
if(session.getAttribute("
    currentDatabaseList")==null ||
    session.getAttribute("
    sessionUserRole")==null)
 errorMessage="Session has expired. Pls
     log again.";
else{
 String[] currentRoleList= (String[])
     session.getAttribute("
     currentDatabaseList");
 for(int i=0; i<currentRoleList.length;i
     ++){
  privilegeCheck= adminSP.checkPrivilege
      (currentRoleList[i], "section", "
      select");
  if(privilegeCheck){
   currentRole=currentRoleList[i];
   break;
  }
 }
 System.out.println(currentRole+"-"+
     privilegeCheck);

 userSP.setDatabase_username(currentRole
     );
 roleSP.setDatabase_username(currentRole
     );
 sectionSP.setDatabase_username(
     currentRole);
 adminSP.setDatabase_username(
     currentRole);

 currentSectionLists= sectionSP.
     getListSectionsOfUser(Integer.
     parseInt(session.getAttribute("
     sessionUserId").toString()));

 currentRoleLists= sectionSP.
     getListRoleSection(section_id);

 allRoles= roleSP.listAllRoles();
 for (Role currentRoleListss :
     currentRoleLists) {
  currentUserLists.add(sectionSP.
      getListUserSection(
      currentRoleListss.getRole_id()));
 }
 for(ArrayList<User> currentUserList:
     currentUserLists){
  for(User currentUser: currentUserList)
      {
   if(searchString.isEmpty() ||
       searchString.equals(""))
   {
   System.out.println("OOPS: "+
       currentUser.getUsername());
   UserRoleSection tempUserRoleSection=
       new UserRoleSection();

   if(pickUserRole.equalsIgnoreCase("all
       ")){
    String role="";
    ArrayList<Role> tempRole= new
        ArrayList<Role>();
    tempRole=userSP.getListRolesofUser(
        currentUser.getUser_id());
    for(Role checkRole: tempRole){
     if(checkRole.getRole_name().
         toLowerCase().indexOf("student
         ")!=-1){
      role="Student Clinician";
      break;
     }
     else if(checkRole.getRole_name().
         toLowerCase().indexOf("faculty
         ")!=-1){
      role="Faculty";
      break;
     }
     else{
      role="Administrator";
     }

    }
    tempUserRoleSection.setUser_name(
        currentUser.getUsername());
    tempUserRoleSection.setFull_name(
        currentUser.getFname_user()+" "
        +currentUser.getLname_user());
    tempUserRoleSection.setUser_role(
        role);
    currentLists.add(tempUserRoleSection
        );

   }
   else{
   //for user_role
   String role="";
   ArrayList<Role> tempRole= new
```

```
        ArrayList<Role>();
    tempRole=userSP.getListRolesofUser(
        currentUser.getUser_id());
    for(Role checkRole: tempRole){

     if(checkRole.getRole_name().
         toLowerCase().indexOf("student"
         )!=-1 && pickUserRole.
         toLowerCase().indexOf("student"
         )!=-1){
      role="Student Clinician";


      tempUserRoleSection.setUser_name(
           currentUser.getUsername());
      tempUserRoleSection.setFull_name(
           currentUser.getFname_user()+"
           "+currentUser.getLname_user())
           ;
      tempUserRoleSection.setUser_role(
           role);
      currentLists.add(
           tempUserRoleSection);

      break;
     }
     else if(checkRole.getRole_name().
         toLowerCase().indexOf("faculty"
         )!=-1 && pickUserRole.
         toLowerCase().indexOf("faculty"
         )!=-1){
      role="Faculty";


      tempUserRoleSection.setUser_name(
           currentUser.getUsername());
      tempUserRoleSection.setFull_name(
           currentUser.getFname_user()+"
           "+currentUser.getLname_user())
           ;
      tempUserRoleSection.setUser_role(
           role);
      currentLists.add(
           tempUserRoleSection);

      break;
     }

     else{
      role="Admin";
     }
    }
}

}
    else{
if(currentUser.getUsername().
    toLowerCase().indexOf(
    searchString)!=-1 || currentUser
    .getFname_user().toLowerCase().
    indexOf(searchString)!=-1
  || currentUser.getLname_user().
      toLowerCase().indexOf(
      searchString)!=-1){
 UserRoleSection tempUserRoleSection=
      new UserRoleSection();

 if(pickUserRole.equalsIgnoreCase("
     all")){
  String role="";
  ArrayList<Role> tempRole= new
      ArrayList<Role>();
  tempRole=userSP.getListRolesofUser(
      currentUser.getUser_id());
  for(Role checkRole: tempRole){
   if(checkRole.getRole_name().
       toLowerCase().indexOf("
       student")!=-1){
    role="Student Clinician";
    break;
   }
   else if(checkRole.getRole_name().
       toLowerCase().indexOf("
       faculty")!=-1){
    role="Faculty";
    break;
   }
   else{
    role="Administrator";
   }

  }
  tempUserRoleSection.setUser_name(
      currentUser.getUsername());
  tempUserRoleSection.setFull_name(
      currentUser.getFname_user()+"
      "+currentUser.getLname_user())
      ;
  tempUserRoleSection.setUser_role(
      role);
  currentLists.add(
      tempUserRoleSection);

 }
 else{
 //for user_role
 ArrayList<Role> tempRole= new
     ArrayList<Role>();
 tempRole=userSP.getListRolesofUser(
     currentUser.getUser_id());
 String role="";
 for(Role checkRole: tempRole){
```

```
                if(checkRole.getRole_name().
                    toLowerCase().indexOf("student
                    ")!=-1 && pickUserRole.
                    toLowerCase().indexOf("student
                    ")!=-1){
                 role="Student_Clinician";


                 tempUserRoleSection.setUser_name(
                      currentUser.getUsername());
                 tempUserRoleSection.setFull_name(
                      currentUser.getFname_user()+"
                      _"+currentUser.getLname_user
                      ());
                 tempUserRoleSection.setUser_role(
                      role);
                 currentLists.add(
                      tempUserRoleSection);

                 break;
                }
                else if(checkRole.getRole_name().
                    toLowerCase().indexOf("faculty
                    ")!=-1 && pickUserRole.
                    toLowerCase().indexOf("faculty
                    ")!=-1){
                 role="Faculty";
                 tempUserRoleSection.setUser_name(
                      currentUser.getUsername());
                 tempUserRoleSection.setFull_name(
                      currentUser.getFname_user()+"
                      _"+currentUser.getLname_user
                      ());
                 tempUserRoleSection.setUser_role(
                      role);
                 currentLists.add(
                      tempUserRoleSection);

                 break;
                }
                else{
                 role="Admin";
                }
               }
                    }


                    }


                   }// end of search string condition
                  }


               }
               } catch (Exception e) {
                // TODO Auto-generated catch block
                System.out.println("ERROR");
                e.printStackTrace();
               }

               UserRoleSection userRoleSection= new
                    UserRoleSection();
               userRoleSection.setSection_id(section_id
                    );
               model.addAttribute("currentLists",
                    currentLists);
               model.addAttribute("userRoleSection",
                    userRoleSection);
               model.addAttribute("currentSectionLists"
                    ,currentSectionLists);

               return "viewFacultySection";

              }

              @RequestMapping(value = "/dentistDesigner
                   .html", method = RequestMethod.GET)
              public String viewDesigner(Model model) {


               return "dentistDesigner";
              }

              }
```

Listing 38: StatisticsController

```
package com.dentist.version.three.
    controller;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;

import javax.servlet.http.
    HttpServletRequest;
import javax.servlet.http.
    HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import org.springframework.stereotype.
    Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.
    annotation.ModelAttribute;
import org.springframework.web.bind.
    annotation.RequestMapping;
import org.springframework.web.bind.
    annotation.RequestMethod;
import org.springframework.web.bind.
    annotation.RequestParam;
import org.springframework.web.servlet.
    ModelAndView;

import com.dentist.version.three.db.
    QuerySP;
import com.dentist.version.three.form.
    ConsultationsReferrals;
import com.dentist.version.three.form.
    DentalChart;
import com.dentist.version.three.form.
    Patient;
import com.dentist.version.three.form.
    PatientInformation;
import com.dentist.version.three.form.
    ServiceNeeded;

@Controller
public class StatisticsController {

 @RequestMapping(value = "/
     statisticsResults.html", method =
     RequestMethod.POST)
 public ModelAndView viewQueryPatient(
     Model model, HttpServletRequest
     request, HttpServletResponse
     response) throws Exception {

  return new ModelAndView("
      statisticsResults");


 }

 private Date parseDate(String date,
     String format) throws
     ParseException
 {
     SimpleDateFormat formatter = new
         SimpleDateFormat(format);
     return formatter.parse(date);
 }

 @ModelAttribute("condd")
 public String condd(
    @RequestParam(value="condd", required
        = false) String condd)
    {
     return condd;
 }

 @ModelAttribute("periodontics")
 public String periodontics(
    @RequestParam(value="periodontics",
        required = false) String
        periodontics)
    {
     return periodontics;
 }


 @ModelAttribute("periocount")
 protected List<Integer> getPerioCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

  String condition= request.getParameter(
      "condd");

  for(int i= 0; i < consult.size(); i++)
      {
   if(consult.get(i).getConsultto().trim
       ().equalsIgnoreCase("PERIO")) {
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge(),
          request) && checkCity(patient.
         getCity(), request)) {
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"))
       fcount++;
      else mcount++;
     }
```

```
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
}

 @ModelAttribute("rpdcount")
 protected List<Integer> getRPDCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

String condition= request.getParameter("
    condd");

  for(int i= 0; i < consult.size(); i++)
      {
   if(consult.get(i).getConsultto().trim
       ().equalsIgnoreCase("RPD")) {
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge(),
         request) && checkCity(patient.
        getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
}

 @ModelAttribute("orthocount")
 protected List<Integer> getOrthoCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

String condition= request.getParameter("
    condd");

  for(int i= 0; i < consult.size(); i++)
      {
   if(consult.get(i).getConsultto().trim
       ().equalsIgnoreCase("Ortho")) {
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge(),
```

```
            request) && checkCity(patient.
            getCity(), request)) {
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("female"))
         fcount++;
        else mcount++;
       }
      }//end if condition and3
      else{

       if(checkWithinDate(patient.
            getDatecreated(), request)){
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("female"))
         fcount++;
        else mcount++;
       }//end else
      }
     }

    }//end for loop
     return savelist(count, fcount, mcount,
          request, session);
    }

    @ModelAttribute("oscount")
    protected List<Integer> getOSCount(
         HttpServletRequest request,
         HttpSession session) throws
         Exception {
     List<ConsultationsReferrals> consult =
          this.getConsultation(request,
          session);
     int fcount = 0;
     int mcount = 0;
     int count = 0;

     session=request.getSession(false);
     QuerySP querySP= new QuerySP();
     querySP.setDatabase_username(session.
          getAttribute("sessionUserRole").
          toString());

    String condition= request.getParameter("
        condd");

     for(int i= 0; i < consult.size(); i++)
          {
      if(consult.get(i).getConsultto().trim
           ().equalsIgnoreCase("OS")) {
       Patient patient = querySP.getPatient(
            consult.get(i).getPatientid());
       //Date date = parseDate(patient.
            getDatecreated(), "dd/MM/yyyy");
      if(condition.equals("and3")){
       if(checkWithinDate(patient.
            getDatecreated(), request)&&
            checkWithinAge(patient.getAge(),
             request) && checkCity(patient.
            getCity(), request)) {
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("female"))
         fcount++;
        else mcount++;
       }
      }//end if condition and3
      else{

       if(checkWithinDate(patient.
            getDatecreated(), request)){
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("female"))
         fcount++;
        else mcount++;
       }//end else
      }
     }

    }//end for loop
     return savelist(count, fcount, mcount,
          request, session);
    }

    @ModelAttribute("fpdcount")
    protected List<Integer> getFPDCount(
         HttpServletRequest request,
         HttpSession session) throws
         Exception {
     List<ConsultationsReferrals> consult =
          this.getConsultation(request,
          session);
     int fcount = 0;
     int mcount = 0;
     int count = 0;

     session=request.getSession(false);
     QuerySP querySP= new QuerySP();
     querySP.setDatabase_username(session.
          getAttribute("sessionUserRole").
          toString());

    String condition= request.getParameter("
        condd");

     for(int i= 0; i < consult.size(); i++)
          {
      if(consult.get(i).getConsultto().trim
           ().equalsIgnoreCase("FPD")) {
```

```
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge(),
         request) && checkCity(patient.
        getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
 }

 @ModelAttribute("pedocount")
 protected List<Integer> getPedoCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

String condition= request.getParameter("
    condd");
  for(int i= 0; i < consult.size(); i++)
      {
   if(consult.get(i).getConsultto().trim
       ().equalsIgnoreCase("PEDO")) {
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge(),
         request) && checkCity(patient.
        getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
 }


 @ModelAttribute("endocount")
 protected List<Integer> getEndoCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
```

```
        toString());

String condition= request.getParameter("
    condd");

  for(int i= 0; i < consult.size(); i++)
      {
   if(consult.get(i).getConsultto().trim
       ().equalsIgnoreCase("ENDO")) {
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge(),
         request) && checkCity(patient.
        getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
 }

 @ModelAttribute("cdcount")
 protected List<Integer> getCDCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;
  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

String condition= request.getParameter("
    condd");

  for(int i= 0; i < consult.size(); i++)
      {
   if(consult.get(i).getConsultto().trim
       ().equalsIgnoreCase("CD")) {
    Patient patient = querySP.getPatient(
        consult.get(i).getPatientid());
    //Date date = parseDate(patient.
        getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge(),
         request) && checkCity(patient.
        getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
 }

 @ModelAttribute("restocount")
 protected List<Integer> getRestoCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<ConsultationsReferrals> consult =
      this.getConsultation(request,
      session);
```

```
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

String condition= request.getParameter("
     condd");

  for(int i= 0; i < consult.size(); i++)
       {
   if(consult.get(i).getConsultto().trim
        ().equalsIgnoreCase("Resto")) {
    Patient patient = querySP.getPatient(
         consult.get(i).getPatientid());
    //Date date = parseDate(patient.
         getDatecreated(), "dd/MM/yyyy");
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge(),
          request) && checkCity(patient.
         getCity(), request)) {
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
         getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
 }

 @ModelAttribute("class1count")
 protected List<Integer> getClass1Count(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  List<ServiceNeeded> services = this.
       getServicesNeeded(request, session
       );
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());

String condition= request.getParameter("
     condd");

for(int i= 0; i < services.size(); i++) {
 if(services.get(i).getClass_1()[0]!=0 &&
       services.get(i).getClass_1() !=
      null && services.get(i).
      getIs_current().equals("yes")) {

  Patient patient = querySP.getPatient(
       services.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge(),
          request) && checkCity(patient.
         getCity(), request)) {
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
         getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
 }
```

```
@ModelAttribute("class2count")
protected List<Integer> getClass2Count(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<ServiceNeeded> services = this.
     getServicesNeeded(request, session
     );
 int fcount = 0;
 int mcount = 0;
 int count = 0;
//
 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());



 String condition= request.getParameter(
     "condd");

 for(int i= 0; i < services.size(); i++)
     {
  if(services.get(i).getClass_2()[0]!=0
     && services.get(i).getClass_2()
     != null && services.get(i).
     getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
         getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("class3count")
protected List<Integer> getClass3Count(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<ServiceNeeded> services = this.
     getServicesNeeded(request, session
     );
 int fcount = 0;
 int mcount = 0;
 int count = 0;
 //
 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());

 String condition= request.getParameter(
     "condd");

 for(int i= 0; i < services.size(); i++)
     {
  if(services.get(i).getClass_3()[0]!=0
     && services.get(i).getClass_3()
     != null && services.get(i).
     getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
```

```
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("class4count")
protected List<Integer> getClass4Count(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < services.size(); i++)
       {
  if(services.get(i).getClass_4()[0]!=0
       && services.get(i).getClass_4()
       != null && services.get(i).
       getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("class5count")
protected List<Integer> getClass5Count(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < services.size(); i++)
       {
  if(services.get(i).getClass_5()[0]!=0
       && services.get(i).getClass_5()
       != null && services.get(i).
       getIs_current().equals("yes")) {
```

```
   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
         getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("onlaycount")
protected List<Integer> getOnlayCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<ServiceNeeded> services = this.
     getServicesNeeded(request, session
     );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());
 String condition= request.getParameter(
     "condd");

 for(int i= 0; i < services.size(); i++)
     {
  if(services.get(i).getOnlay()[0]!=0 &&
      services.get(i).getOnlay() !=
      null && services.get(i).
      getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
         getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("extractioncount")
protected List<Integer>
    getExtractionCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<ServiceNeeded> services = this.
     getServicesNeeded(request, session
     );
 int fcount = 0;
```

```
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

String condition= request.getParameter("
     condd");

 for(int i= 0; i < services.size(); i++)
       {
  if(services.get(i).getExtraction()
       [0]!=0 && services.get(i).
       getExtraction() != null &&
       services.get(i).getIs_current().
       equals("yes")) {

   Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("odontectomycount")
protected List<Integer>
     getOdontectomyCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

String condition= request.getParameter("
     condd");

 for(int i= 0; i < services.size(); i++)
       {
  if(services.get(i).getOdontectomy()
       [0]!=0 && services.get(i).
       getOdontectomy() != null &&
       services.get(i).getIs_current().
       equals("yes")) {

   Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
```

```
            ))
          fcount++;
         else mcount++;
        }//end else
       }
      }

    }//end for loop
   return savelist(count, fcount, mcount,
        request, session);
}

@ModelAttribute("specialcasecount")
protected List<Integer>
     getSpecialcaseCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;


 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


 String condition= request.getParameter(
      "condd");

  for(int i= 0; i < services.size(); i
       ++) {
   if(services.get(i).getSpecial_case()
        [0]!=0 && services.get(i).
        getSpecial_case() != null &&
        services.get(i).getIs_current().
        equals("yes")) {


    Patient patient = querySP.getPatient
         (services.get(i).getPatient_id
         ());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request))
           {
        count++;
        if(patient.getGender().trim().
             toLowerCase().equals("female
             "))
         fcount++;
        else mcount++;
       }//end if condition and3
       else{

        if(checkWithinDate(patient.
             getDatecreated(), request)){
         count++;
         if(patient.getGender().trim().
              toLowerCase().equals("female
              "))
          fcount++;
         else mcount++;
        }//end else
       }
      }

    }//end for loop
   return savelist(count, fcount, mcount,
        request, session);
}

@ModelAttribute("pedodonticscount")
protected List<Integer>
     getPedodonticsCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

  for(int i= 0; i < services.size(); i
       ++) {
   if(services.get(i).getSurgery() !=
        null && services.get(i).
        getSurgery().toLowerCase().
```

```
        indexOf("pedodontics") != -1 &&
        services.get(i).getIs_current().
        equals("yes")) {

    Patient patient = querySP.getPatient
        (services.get(i).getPatient_id
        ());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request))
          {
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female
           "))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
          getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female
           "))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("orthodonticscount")
protected List<Integer>
    getOrthodonticsCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<ServiceNeeded> services = this.
     getServicesNeeded(request, session
     );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());


 String condition= request.getParameter(
     "condd");

  for(int i= 0; i < services.size(); i
      ++) {
   if(services.get(i).getSurgery() !=
       null && services.get(i).
       getSurgery().toLowerCase().
       indexOf("pedodontics") != -1 &&
       services.get(i).getIs_current().
       equals("yes")) {

    Patient patient = querySP.getPatient
        (services.get(i).getPatient_id
        ());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request))
          {
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female
           "))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
          getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female
           "))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("pulpsedationcount")
```

```
protected List<Integer>
    getPulpsedationCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<ServiceNeeded> services = this.
    getServicesNeeded(request, session
    );
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());

String condition= request.getParameter(
    "condd");

for(int i= 0; i < services.size(); i++)
    {
 if(services.get(i).getPulp_sedation()
    [0]!=0 && services.get(i).
    getPulp_sedation() != null &&
    services.get(i).getIs_current().
    equals("yes")) {

  Patient patient = querySP.getPatient(
      services.get(i).getPatient_id())
      ;
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }//end else
   }
  }

  }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("crownrecementationcount
    ")
protected List<Integer>
    getCrownrecementationCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<ServiceNeeded> services = this.
    getServicesNeeded(request, session
    );
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());


String condition= request.getParameter("
    condd");

 for(int i= 0; i < services.size(); i++)
     {
  if(services.get(i).
      getCrown_recementation()[0]!=0 &&
       services.get(i).
      getCrown_recementation() != null
      && services.get(i).getIs_current
      ().equals("yes")) {

   Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
          toLowerCase().equals("female"
```

```
            ))
          fcount++;
         else mcount++;
        }
       }//end if condition and3
       else{

        if(checkWithinDate(patient.
            getDatecreated(), request)){
         count++;
         if(patient.getGender().trim().
             toLowerCase().equals("female"
             ))
          fcount++;
         else mcount++;
        }//end else
       }
      }

    }//end for loop
   return savelist(count, fcount, mcount,
       request, session);
  }

  @ModelAttribute("tempfillingservicecount
      ")
  protected List<Integer>
       getFillingserviceCount(
       HttpServletRequest request,
       HttpSession session) throws
       Exception {
   List<ServiceNeeded> services = this.
        getServicesNeeded(request, session
        );
   int fcount = 0;
   int mcount = 0;
   int count = 0;

   Date convertedDate=null;
   session=request.getSession(false);
   QuerySP querySP= new QuerySP();
   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());


 String condition= request.getParameter("
      condd");

   for(int i= 0; i < services.size(); i++)
        {
    if(services.get(i).getFilling_service
        ()[0]!=0 && services.get(i).
        getFilling_service() != null &&
        services.get(i).getIs_current().
        equals("yes")) {

     Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
          getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
  return savelist(count, fcount, mcount,
      request, session);
 }

 @ModelAttribute("acuteinfectionscount")
 protected List<Integer>
      getAcuteinfectionsCount(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  List<ServiceNeeded> services = this.
       getServicesNeeded(request, session
       );
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  Date convertedDate=null;
  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
```

```
String condition= request.getParameter(
    "condd");

  for(int i= 0; i < services.size(); i
      ++) {
   if(services.get(i).
       getEmergency_treatment() !=
       null && services.get(i).
       getEmergency_treatment().
       toLowerCase().indexOf("acute_
       infections") != -1  && services
       .get(i).getIs_current().equals(
       "yes")) {

    Patient patient = querySP.
        getPatient(services.get(i).
        getPatient_id());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
          getDatecreated(), request)&&
           checkWithinAge(patient.
          getAge(), request) &&
          checkCity(patient.getCity(),
           request)) {
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("
            female"))
         fcount++;
        else mcount++;
       }
      }//end if condition and3
      else{

       if(checkWithinDate(patient.
           getDatecreated(), request)){
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("
            female"))
         fcount++;
        else mcount++;
       }//end else
      }
     }

    }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("traumaticinjuriescount"
    )
protected List<Integer>
    getTraumaticinjuriesCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<ServiceNeeded> services = this.
    getServicesNeeded(request, session
    );
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());


String condition= request.getParameter(
    "condd");

for(int i= 0; i < services.size(); i++)
    {
 if(services.get(i).
     getEmergency_treatment() != null
     && services.get(i).
     getEmergency_treatment().
     toLowerCase().indexOf("acute_
     infections") != -1 && services.
     get(i).getIs_current().equals("
     yes")) {

  Patient patient = querySP.getPatient(
      services.get(i).getPatient_id())
      ;
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
```

```
            toLowerCase().equals("female"
            ))
          fcount++;
         else mcount++;
        }//end else
       }
      }

    }//end for loop
   return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("laminatedcount")
protected List<Integer>
     getLaminatedCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;


 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


String condition= request.getParameter("
     condd");

 for(int i= 0; i < services.size(); i++)
       {
  if(services.get(i).getLaminated()
       [0]!=0 && services.get(i).
       getLaminated() != null &&
       services.get(i).getIs_current().
       equals("yes")) {

   Patient patient = querySP.getPatient(
        services.get(i).getPatient_id())
        ;
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("singlecrowncount")
protected List<Integer>
     getSinglecrownCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;


 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



String condition= request.getParameter("
     condd");

  for(int i= 0; i < services.size(); i++)
       {
   if(services.get(i).getSingle_crown()
       [0]!=0 && services.get(i).
```

```
        getSingle_crown() != null &&
        services.get(i).getIs_current().
        equals("yes")) {

    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("bridgeservicecount")
protected List<Integer>
     getBridgeserviceCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

   for(int i= 0; i < services.size(); i
       ++) {
    if(services.get(i).getBridge_service
        ()[0]!=0 && services.get(i).
        getBridge_service() != null &&
        services.get(i).getIs_current()
        .equals("yes")) {

     Patient patient = querySP.
          getPatient(services.get(i).
          getPatient_id());
      if(condition.equals("and3")){
       if(checkWithinDate(patient.
            getDatecreated(), request)&&
            checkWithinAge(patient.
            getAge(), request) &&
            checkCity(patient.getCity(),
            request)) {
        count++;
        if(patient.getGender().trim().
             toLowerCase().equals("
             female"))
         fcount++;
        else mcount++;
       }
      }//end if condition and3
      else{

       if(checkWithinDate(patient.
            getDatecreated(), request)){
        count++;
        if(patient.getGender().trim().
             toLowerCase().equals("
             female"))
         fcount++;
        else mcount++;
       }//end else
      }
     }

    }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
 }

 @ModelAttribute("anteriorcount")
```

```
protected List<Integer> getAnteriorCount
    (HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<ServiceNeeded> services = this.
    getServicesNeeded(request, session
    );
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



String condition= request.getParameter(
    "condd");

  for(int i= 0; i < services.size(); i
      ++) {
   if(services.get(i).getAnterior()
       [0]!=0 && services.get(i).
       getAnterior() != null &&
       services.get(i).getIs_current()
       .equals("yes")) {

    Patient patient = querySP.
        getPatient(services.get(i).
        getPatient_id());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.
          getAge(), request) &&
          checkCity(patient.getCity(),
          request)) {
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("
           female"))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
          getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("
           female"))
        fcount++;
       else mcount++;
      }//end else
     }
    }

  }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("posteriorcount")
protected List<Integer>
    getPosteriorCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<ServiceNeeded> services = this.
    getServicesNeeded(request, session
    );
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());


String condition= request.getParameter(
    "condd");

for(int i= 0; i < services.size(); i++)
    {
 if(services.get(i).getPosterior()
     [0]!=0 && services.get(i).
     getPosterior() != null &&
     services.get(i).getIs_current().
     equals("yes")) {

  Patient patient = querySP.getPatient(
      services.get(i).getPatient_id())
      ;
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
```

```
            toLowerCase().equals("female"
            ))
          fcount++;
         else mcount++;
        }
       }//end if condition and3
       else{

        if(checkWithinDate(patient.
             getDatecreated(), request)){
         count++;
         if(patient.getGender().trim().
             toLowerCase().equals("female"
             ))
          fcount++;
         else mcount++;
        }//end else
       }
      }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("
     completedentservicecount")
protected List<Integer>
     getCompletedentserviceCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



String condition= request.getParameter("
     condd");

for(int i= 0; i < services.size(); i++)
     {
 if(services.get(i).getProsthodontics()
      != null && services.get(i).
      getProsthodontics().indexOf("
      complete_denture") != -1 &&
      services.get(i).getIs_current().
      equals("yes")) {

  Patient patient = querySP.getPatient(
       services.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge()
         , request) && checkCity(patient
         .getCity(), request)) {
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female")
          )
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
         getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female")
          )
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}
@ModelAttribute("singledentservicecount"
     )
protected List<Integer>
     getSingledentserviceCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<ServiceNeeded> services = this.
      getServicesNeeded(request, session
      );
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
```

```
        getAttribute("sessionUserRole").
        toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < services.size(); i++)
       {
  if(services.get(i).getProsthodontics()
        != null && services.get(i).
       getProsthodontics().indexOf("
       complete_denture") != -1 &&
       services.get(i).getIs_current().
       equals("yes")) {

    Patient patient = querySP.getPatient(
         services.get(i).getPatient_id())
         ;
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}
@ModelAttribute("
     removablepartialservicecount")
protected List<Integer>
     getRemovablepartiaserviceCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
List<ServiceNeeded> services = this.
     getServicesNeeded(request, session
     );
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());



String condition= request.getParameter(
     "condd");

for(int i= 0; i < services.size(); i++)
      {
 if(services.get(i).getProsthodontics()
       != null && services.get(i).
      getProsthodontics().indexOf("
      removable_partial") != -1 &&
      services.get(i).getIs_current().
      equals("yes")) {

  Patient patient = querySP.getPatient(
       services.get(i).getPatient_id())
       ;
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
         getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
      fcount++;
```

```
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("class1")
public String class1(
   @RequestParam(value="class1",
        required = false) String class1)
   {
    return class1;
}
@ModelAttribute("class2")
public String class2(
   @RequestParam(value="class2",
        required = false) String class2)
   {
    return class2;
}
@ModelAttribute("class3")
public String class3(
   @RequestParam(value="class3",
        required = false) String class3)
   {
    return class3;
}
@ModelAttribute("class4")
public String class4(
   @RequestParam(value="class4",
        required = false) String class4)
   {
    return class4;
}
@ModelAttribute("class5")
public String class5(
   @RequestParam(value="class5",
        required = false) String class5)
   {
    return class5;
}
@ModelAttribute("onlay")
public String onlay(
   @RequestParam(value="onlay", required
        = false) String onlay)
   {
    return onlay;
}
@ModelAttribute("extraction")
public String extraction(
   @RequestParam(value="extraction",
        required = false) String
        extraction)
   {
    return extraction;
}
@ModelAttribute("odontectomy")
public String odontectomy(
   @RequestParam(value="odontectomy",
        required = false) String
        odontectomy)
   {
    return odontectomy;
}
@ModelAttribute("specialcase")
public String specialcase(
   @RequestParam(value="specialcase",
        required = false) String
        specialcase)
   {
    return specialcase;
}
@ModelAttribute("pedodontics")
public String pedodontics(
   @RequestParam(value="pedodontics",
        required = false) String
        pedodontics)
   {
    return pedodontics;
}
@ModelAttribute("orthodontics")
public String orthodontics(
   @RequestParam(value="orthodontics",
        required = false) String
        orthodontics)
   {
    return orthodontics;
}
@ModelAttribute("pulpsedation")
public String pulpsedation(
   @RequestParam(value="pulpsedation",
        required = false) String
        pulpsedation)
   {
    return pulpsedation;
}
@ModelAttribute("crownrecementation")
public String crownrecementation(
   @RequestParam(value="
        crownrecementation", required =
        false) String crownrecementation
        )
   {
    return crownrecementation;
}
@ModelAttribute("tempfillingservice")
public String tempfillingservice(
   @RequestParam(value="
```

```
        tempfillingservice", required =
        false) String tempfillingservice
        )
    {
     return tempfillingservice;
}
@ModelAttribute("acuteinfections")
public String acuteinfections(
    @RequestParam(value="acuteinfections"
        , required = false) String
        acuteinfections)
    {
     return acuteinfections;
}
@ModelAttribute("traumaticinjuries")
public String traumaticinjuries(
    @RequestParam(value="
        traumaticinjuries", required =
        false) String traumaticinjuries)
    {
     return traumaticinjuries;
}
@ModelAttribute("laminated")
public String laminated(
    @RequestParam(value="laminated",
        required = false) String
        laminated)
    {
     return laminated;
}
@ModelAttribute("singlecrown")
public String singlecrown(
    @RequestParam(value="singlecrown",
        required = false) String
        singlecrown)
    {
     return singlecrown;
}
@ModelAttribute("bridge")
public String bridge(
    @RequestParam(value="bridge",
        required = false) String bridge)
    {
     return bridge;
}
@ModelAttribute("anterior")
public String anterior(
    @RequestParam(value="anterior",
        required = false) String
        anterior)
    {
     return anterior;
}
@ModelAttribute("posterior")
public String posterior(
    @RequestParam(value="posterior",
        required = false) String
        posterior)
    {
     return posterior;
}
@ModelAttribute("completedentservice")
public String completedentservice(
    @RequestParam(value="
        completedentservice", required =
         false) String
        completedentservice)
    {
     return completedentservice;
}
@ModelAttribute("singledentservice")
public String singledentservice(
    @RequestParam(value="
        singledentservice", required =
        false) String singledentservice)
    {
     return singledentservice;
}
@ModelAttribute("removablepartialservice
    ")
public String removablepartialservice(
    @RequestParam(value="
        removablepartialservice",
        required = false) String
        removablepartialservice)
    {
     return removablepartialservice;
}

@ModelAttribute("perio")
public String perio(
    @RequestParam(value="perio", required
         = false) String perio)
    {
     return perio;
}
@ModelAttribute("rpd")
public String rpd(
    @RequestParam(value="rpd", required =
         false) String rpd)
    {
     return rpd;
}
@ModelAttribute("resto")
public String resto(
    @RequestParam(value="resto", required
         = false) String resto)
    {
     return resto;
}
@ModelAttribute("os")
public String os(
```

```
    @RequestParam(value="os", required =
        false) String os)
    {
     return os;
}
@ModelAttribute("fpd")
public String fpd(
    @RequestParam(value="fpd", required =
         false) String fpd)
    {
     return fpd;
}
@ModelAttribute("pedo")
public String pedo(
    @RequestParam(value="pedo", required
        = false) String pedo)
    {
     return pedo;
}
@ModelAttribute("endo")
public String endo(
    @RequestParam(value="endo", required
        = false) String endo)
    {
     return endo;
}
@ModelAttribute("cd")
public String cd(
    @RequestParam(value="cd", required =
        false) String cd)
    {
     return cd;
}
@ModelAttribute("ortho")
public String ortho(
    @RequestParam(value="ortho", required
         = false) String ortho)
    {
     return ortho;
}



/**
 * This class returns the form backing
     object. This can be a string, a
     boolean, or a normal java
 * pojo. The bean name defined in the
     ModelAttribute annotation and the
     type can be just
 * defined by the return type of this
     method
 */

@ModelAttribute("theServicesNeeded")
protected ArrayList<ServiceNeeded>
    getServicesNeeded(
        HttpServletRequest request,
        HttpSession session) throws
        Exception {
 ArrayList<ServiceNeeded> services =null
      ;
 session=request.getSession(false);



 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
  services=querySP.allServiceNeeded();

 return services;
}
@ModelAttribute("theDentalChart")
protected ArrayList<DentalChart>
    getDentalChart(HttpServletRequest
    request,HttpSession session) throws
     Exception {
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
  ArrayList<DentalChart> chart=querySP.
      allDentalChart();

 return chart;
}
@ModelAttribute("checklang")
protected Patient checkLang(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {

 List<Patient> patients = this.
     getPatients(request,session);

 return patients.get(0);
}

@ModelAttribute("cariescount")
protected List<Integer> getCariesCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<DentalChart> chart = this.
     getDentalChart(request,session);
 int fcount = 0;
 int mcount = 0;
 int count = 0;
```

```
Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());

String condition= request.getParameter(
    "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getCaries()[0]!=0 &&
     chart.get(i).getCaries() != null
     && chart.get(i).getIs_current().
     equals("yes")) {

  Patient patient = querySP.getPatient(
      chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("totals")
protected List<Integer> getTotal(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<Patient> patients = this.
    getPatients(request,session);
List<PatientInformation> info = this.
    getPatientInfo(request,session);

String convertedDate="";
double ftotal = 0.0, mtotal = 0.0,
    total=0.0;
for(int i = 0; i < patients.size(); i
    ++) {
 //Date date = parseDate(patients.get(i
     ).getDatecreated().trim(), "dd/MM
     /yyyy");
 if(checkWithinDate(patients.get(i).
     getDatecreated(), request)&&
     patients.get(i).getGender().
     toLowerCase().trim().
     equalsIgnoreCase("female"))
  ftotal++;
 if(checkWithinDate(patients.get(i).
     getDatecreated(), request)&&
     patients.get(i).getGender().
     toLowerCase().trim().
     equalsIgnoreCase("male"))
  mtotal++;
 total++;
}
double a = ftotal/total*100;
double b = mtotal/total*100;
List<Integer> totals=  Arrays.asList((
    int) ftotal, (int) a, (int) mtotal
    , (int) b);
return totals;
}

@ModelAttribute("agegrouppatients")
protected List<Integer>
    getAgeGroupPatients(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<Patient> patients = this.
    getPatients(request,session);
double ftotal = 0.0, mtotal = 0.0,
    total=0.0;
for(int i = 0; i < patients.size(); i
    ++) {
 //Date date = parseDate(patients.get(i
     ).getDatecreated(), "dd/MM/yyyy")
     ;
 if(checkWithinDate(patients.get(i).
     getDatecreated(), request)&&
     checkWithinAge(patients.get(i).
     getAge(), request)) {
  if(patients.get(i).getGender().trim()
```

```
        .toLowerCase().equalsIgnoreCase(
        "female"))
    ftotal++;
   if(patients.get(i).getGender().trim()
        .toLowerCase().equalsIgnoreCase(
        "male"))
    mtotal++;
   total++;
  }
 }
 double a = ftotal/total*100;
 double b = mtotal/total*100;
 List<Integer> totals=  Arrays.asList((
      int) ftotal, (int) a, (int) mtotal
      , (int) b);
 return totals;
}

@ModelAttribute("citygrouppatients")
protected List<Integer>
     getCityGroupPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<Patient> patients = this.
      getPatients(request,session);
 double ftotal = 0.0, mtotal = 0.0,
      total=0.0;
 for(int i = 0; i < patients.size(); i
      ++) {
 // Date date = parseDate(patients.get(i
      ).getDatecreated(), "dd/MM/yyyy");
  if(checkWithinDate(patients.get(i).
       getDatecreated(), request)&&
       checkCity(patients.get(i).getCity
       (), request)) {
   if(patients.get(i).getGender().trim()
        .toLowerCase().equalsIgnoreCase(
        "female"))
    ftotal++;
   if(patients.get(i).getGender().trim()
        .toLowerCase().equalsIgnoreCase(
        "male"))
    mtotal++;
   total++;
  }
 }
 double a = ftotal/total*100;
 double b = mtotal/total*100;
 List<Integer> totals=  Arrays.asList((
      int) ftotal, (int) a, (int) mtotal
      , (int) b);
 return totals;
}

protected List<Integer> savelist(int
     count, int fcount, int mcount,
     HttpServletRequest request,
     HttpSession session) throws
     Exception{
 ArrayList<Patient> patients = this.
      getPatients(request,session);
 double ftotal = 0.0, mtotal = 0.0;
 for(int i = 0; i < patients.size(); i
      ++) {
  if(patients.get(i).getGender().trim().
       toLowerCase().equalsIgnoreCase("
       female"))
   ftotal++;
  if(patients.get(i).getGender().trim().
       toLowerCase().equalsIgnoreCase("
       male"))
   mtotal++;
 }
 System.out.println("CHECK_TOTALS:_"+
      fcount+"-"+ftotal);
 double a = (fcount/ftotal)*100;
 double b = (mcount/mtotal)*100;
 double c = (fcount+mcount)/(ftotal+
      mtotal)*100;
 List<Integer> condition = new ArrayList
      <Integer>();
 double cnt = (double)count;
 if(count != 0)
  condition =  Arrays.asList(count,
       fcount, (int)(fcount/cnt*100),
       mcount, (int)(mcount/cnt*100), (
       int)a, (int)b, (int)c);
 else condition = Arrays.asList(0, 0, 0,
       0, 0, 0, 0, 0);
 return condition;
}

protected boolean checkWithinDate(String
      date, HttpServletRequest request)
     throws Exception {
 String[] datefrom= new String[3];
 String[] dateto= new String[3];
 if(request.getParameter("datefrom")!=
      null)
  datefrom = request.getParameter("
       datefrom").split("/");
 if(request.getParameter("dateto")!=null
      )
  dateto = request.getParameter("dateto"
       ).split("/");


 int ddf = Integer.parseInt(datefrom[0])
      ;
 int mmf = Integer.parseInt(datefrom[1])
      ;
```

```
int yyf = Integer.parseInt(datefrom[2])
    ;
int ddt = Integer.parseInt(dateto[0]);
int mmt = Integer.parseInt(dateto[1]);
int yyt = Integer.parseInt(dateto[2]);
if(date==null)
 return false;

String[] finalDate= new String[3];
finalDate=date.split("/");
int dd = Integer.parseInt(finalDate[0])
    ;
int mm = Integer.parseInt(finalDate[1])
    ;
int yy = Integer.parseInt(finalDate[2])
    ;
System.out.println("CHECK :"+ddf+"-"+
    mmf+"-"+yyf);
System.out.println("CHECK :"+ddt+"-"+
    mmt+"-"+yyt);
System.out.println("CHECK :"+dd+"-"+mm+
    "-"+yy);
if(yyf < yy && yy < yyt) {
 return true;
}
else if(yyf == yy && yy < yyt){
 if(mmf < mm) {
  return true;
 }
 else if(mmf == mm){
  if(ddf <= dd)
   return true;
 }
}
else if(yyt == yy && yyf < yy){
 if(mm < mmt) {
  return true;
 }
 else if(mm == mmt){
  if(dd <= ddt)
   return true;
 }
}
else if(yyf == yy && yy == yyt){
 if(mmf < mm && mm < mmt) {
  return true;
 }
 else if(mmf == mm && mm < mmt){
  if(ddf <= dd)
   return true;
 }
 else if(mmf < mm && mm == mmt){
  if(dd <= ddt)
   return true;
 }
 else if(mmf == mm && mm == mmt){
  if(ddf <= dd && dd <= ddt)
   return true;
  }
 }
 return false;
}

protected boolean checkWithinAge(String
    age, HttpServletRequest request)
    throws Exception {
 String agefrom = request.getParameter("
     agefrom");
 String ageto = request.getParameter("
     ageto");
 if(agefrom==null || agefrom.isEmpty())
  agefrom = "0";
 if(ageto==null || ageto.isEmpty())
  ageto = "100";
 int agef = Integer.parseInt(agefrom);
 int aget = Integer.parseInt(ageto);


 if(age==null || age.trim().isEmpty())
  return false;

  int ageCompare = Integer.parseInt(age.
      trim());


 if(agef < ageCompare && ageCompare <
     aget)
  return true;
 else return false;

}

protected boolean checkCity(String
    occupation, HttpServletRequest
    request) throws Exception {
 String city = request.getParameter("
     city");
 if(city.isEmpty())
  return true;
 if(occupation==null || occupation.trim
     ().isEmpty())
  return false;
 if(occupation.trim().toLowerCase().
     indexOf(city.toLowerCase()) != -1)
  return true;
 else return false;
}

@ModelAttribute("recurrentcariescount")
protected List<Integer>
```

```
    getRecurrentCariesCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<DentalChart> chart = this.
     getDentalChart(request, session);
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());

String condition= request.getParameter(
     "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getRecurrentcaries()
      [0]!=0 && chart.get(i).
      getRecurrentcaries() != null &&
      chart.get(i).getIs_current().
      equals("yes")) {

  Patient patient = querySP.getPatient(
       chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
         getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
      fcount++;
     else mcount++;
    }//end else
   }
  }
  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("restorationcount")
protected List<Integer>
     getrestorationCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());


 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getRestoration()[0]!=0
       && chart.get(i).getRestoration()
       != null && chart.get(i).
       getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
```

```
        if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
         fcount++;
        else mcount++;
       }//end else
      }
     }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
 }

/*
 @ModelAttribute("amalgamcount")
 protected ArrayList<Integer>
      getAmalgamCount(HttpServletRequest
      request) throws Exception {
  ArrayList<DentalChart> chart = Context.
       getService(DentalService.class).
       getAllChart();
  int fcount = 0;
  int mcount = 0;
  int count = 0;
  for(int i= 0; i < chart.size(); i++) {
   if(chart.get(i).getAmalgam() != null
        && chart.get(i).getIscurrent().
        equals("yes")) {
    Patient patient = Context.
         getPatientService().getPatient(
         chart.get(i).getPatientid());
    if(checkWithinDate(patient.
         getDateCreated(), request) &&
         checkWithinAge(patient.getAge(),
          request)) {
     count++;
     if(patient.getGender().toUpperCase()
          .equals("F"))
      fcount++;
     else mcount++;
    }
   }
  }
  return savelist(count, fcount, mcount);
 }

 @ModelAttribute("compositecount")
 protected ArrayList<Integer>
      getCompositeCount(
      HttpServletRequest request) throws
      Exception {
  ArrayList<DentalChart> chart = Context.
       getService(DentalService.class).
       getAllChart();
  int fcount = 0;
  int mcount = 0;
  int count = 0;
  for(int i= 0; i < chart.size(); i++) {
   if(chart.get(i).getComposite() != null
        && chart.get(i).getIscurrent().
       equals("yes")) {
    Patient patient = Context.
         getPatientService().getPatient(
         chart.get(i).getPatientid());
    if(checkWithinDate(patient.
         getDateCreated(), request) &&
         checkWithinAge(patient.getAge(),
          request)) {
     count++;
     if(patient.getGender().toUpperCase()
          .equals("F"))
      fcount++;
     else mcount++;
    }
   }
  }
  return savelist(count, fcount, mcount);
 }

 @ModelAttribute("glassionomercount")
 protected ArrayList<Integer>
      getGlassIonomerCount(
      HttpServletRequest request) throws
      Exception {
  ArrayList<DentalChart> chart = Context.
       getService(DentalService.class).
       getAllChart();
  int fcount = 0;
  int mcount = 0;
  int count = 0;
  for(int i= 0; i < chart.size(); i++) {
   if(chart.get(i).getGlassionomer() !=
        null && chart.get(i).getIscurrent
        ().equals("yes")) {
    Patient patient = Context.
         getPatientService().getPatient(
         chart.get(i).getPatientid());
    if(checkWithinDate(patient.
         getDateCreated(), request) &&
         checkWithinAge(patient.getAge(),
          request)) {
     count++;
     if(patient.getGender().toUpperCase()
          .equals("F"))
      fcount++;
     else mcount++;
    }
   }
  }
  return savelist(count, fcount, mcount);
 }
```

```
 @ModelAttribute("tempfillingcount")
 protected ArrayList<Integer>
      getTempfillingCount(
      HttpServletRequest request) throws
      Exception {
  ArrayList<DentalChart> chart = Context.
       getService(DentalService.class).
       getAllChart();
  int fcount = 0;
  int mcount = 0;
  int count = 0;
  for(int i= 0; i < chart.size(); i++) {
   if(chart.get(i).getTemp_filling() !=
        null && chart.get(i).
        getIs_current().equals("yes")) {
    Patient patient = Context.
         getPatientService().getPatient(
         chart.get(i).getPatientid());
    if(checkWithinDate(patient.
         getDateCreated(), request) &&
         checkWithinAge(patient.getAge(),
          request)) {
     count++;
     if(patient.getGender().toUpperCase()
          .equals("F"))
      fcount++;
     else mcount++;
    }
   }
  }
  return savelist(count, fcount, mcount);
 }
*/
 @ModelAttribute("extrusioncount")
 protected List<Integer>
      getExtrusionCount(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  List<DentalChart> chart = this.
       getDentalChart(request, session);
  int fcount = 0;
  int mcount = 0;
  int count = 0;

  Date convertedDate=null;
  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());



  String condition= request.getParameter(
       "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getExtrusion()[0]!=0
       && chart.get(i).getExtrusion() !=
        null && chart.get(i).
       getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("intrusioncount")
protected List<Integer>
     getIntrusionCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request,session);
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
```

```
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



String condition= request.getParameter(
    "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getIntrusion()[0]!=0
     && chart.get(i).getIntrusion() !=
      null && chart.get(i).
     getIs_current().equals("yes")) {

  Patient patient = querySP.getPatient(
      chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("mesialdriftcount")
protected List<Integer>
    getMesialdriftCount(
    HttpServletRequest request,
    HttpSession session) throws
    Exception {
List<DentalChart> chart =  this.
    getDentalChart(request, session);
int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());



String condition= request.getParameter(
    "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getMesial_rotation()
     [0]!=0 && chart.get(i).
     getMesial_rotation() != null &&
     chart.get(i).getIs_current().
     equals("yes")) {

  Patient patient = querySP.getPatient(
      chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }//end else
   }
  }
```

```
  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("distaldriftcount")
protected List<Integer>
     getDistaldriftCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getDistal_rotation()
       [0]!=0 && chart.get(i).
       getDistal_rotation() != null &&
       chart.get(i).getIs_current().
       equals("yes")) {


   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("completedenturecount")
protected List<Integer>
     getCompletedentureCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
 int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getComplete_denture()
       != null && chart.get(i).
       getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
```

```
            ))
          fcount++;
         else mcount++;
        }
       }//end if condition and3
       else{

        if(checkWithinDate(patient.
            getDatecreated(), request)){
         count++;
         if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
          fcount++;
         else mcount++;
        }//end else
       }
      }

    }//end for loop
   return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("singledenturecount")
protected List<Integer>
     getSingledentureCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getSingle_denture() !=
       null && chart.get(i).
      getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("removablepartialcount")
protected List<Integer>
     getRemovablepartialCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");
```

```
 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).
       getRemovable_partial_denture()
       [0]!=0 && chart.get(i).
       getRemovable_partial_denture() !=
        null && chart.get(i).
       getIs_current().equals("yes")) {

    Patient patient = querySP.getPatient(
         chart.get(i).getPatient_id());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("rotationcount")
protected List<Integer> getRotatioCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getRotation()[0]!=0 &&
       chart.get(i).getRotation() !=
       null && chart.get(i).
       getIs_current().equals("yes")) {

    Patient patient = querySP.getPatient(
         chart.get(i).getPatient_id());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("postcorecrowncount")
protected List<Integer>
     getPostcorecrownCount(
     HttpServletRequest request,
     HttpSession session) throws
```

```
    Exception {
List<DentalChart> chart = this.
     getDentalChart(request, session);
      int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
     getAttribute("sessionUserRole").
     toString());



String condition= request.getParameter(
     "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getPostcore_crown()
      [0]!=0 && chart.get(i).
      getPostcore_crown() != null &&
      chart.get(i).getIs_current().
      equals("yes")) {

  Patient patient = querySP.getPatient(
       chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
         getDatecreated(), request)&&
         checkWithinAge(patient.getAge
         (), request) && checkCity(
         patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
         getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
          toLowerCase().equals("female"
          ))
      fcount++;
     else mcount++;
    }//end else
   }
  }
 }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("rootcanalcount")
protected List<Integer>
     getRootcanalCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getRootcanal_treatment
       ()[0]!=0 && chart.get(i).
       getRootcanal_treatment() != null
       && chart.get(i).getIs_current().
       equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
```

```
        count++;
        if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
         fcount++;
        else mcount++;
       }//end else
      }
     }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("pitandfissurecount")
protected List<Integer>
     getPitandfissureCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getPitfissure_sealants
       ()[0]!=0 && chart.get(i).
       getPitfissure_sealants() != null
       && chart.get(i).getIs_current().
       equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("extractedcount")
protected List<Integer>
     getExtractedCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getExtracted()[0]!=0
       && chart.get(i).getExtracted() !=
        null && chart.get(i).
       getIs_current().equals("yes")) {

   Patient patient = querySP.getPatient(
```

```
            chart.get(i).getPatient_id());
        if(condition.equals("and3")){
         if(checkWithinDate(patient.
              getDatecreated(), request)&&
              checkWithinAge(patient.getAge
              (), request) && checkCity(
              patient.getCity(), request)) {
          count++;
          if(patient.getGender().trim().
               toLowerCase().equals("female"
               ))
           fcount++;
          else mcount++;
         }
        }//end if condition and3
        else{

         if(checkWithinDate(patient.
              getDatecreated(), request)){
          count++;
          if(patient.getGender().trim().
               toLowerCase().equals("female"
               ))
           fcount++;
          else mcount++;
         }//end else
        }
       }

      }//end for loop
     return savelist(count, fcount, mcount,
          request, session);
    }

    @ModelAttribute("missingcount")
    protected List<Integer> getMissingCount(
         HttpServletRequest request,
         HttpSession session) throws
         Exception {
     List<DentalChart> chart = this.
          getDentalChart(request, session);
           int fcount = 0;
     int mcount = 0;
     int count = 0;

     Date convertedDate=null;
     session=request.getSession(false);
     QuerySP querySP= new QuerySP();
     querySP.setDatabase_username(session.
          getAttribute("sessionUserRole").
          toString());



     String condition= request.getParameter(
          "condd");

     for(int i= 0; i < chart.size(); i++) {
      if(chart.get(i).getMissing()[0]!=0 &&
           chart.get(i).getMissing() != null
            && chart.get(i).getIs_current().
           equals("yes")) {

       Patient patient = querySP.getPatient(
            chart.get(i).getPatient_id());
       if(condition.equals("and3")){
        if(checkWithinDate(patient.
             getDatecreated(), request)&&
             checkWithinAge(patient.getAge
             (), request) && checkCity(
             patient.getCity(), request)) {
         count++;
         if(patient.getGender().trim().
              toLowerCase().equals("female"
              ))
          fcount++;
         else mcount++;
        }
       }//end if condition and3
       else{

        if(checkWithinDate(patient.
             getDatecreated(), request)){
         count++;
         if(patient.getGender().trim().
              toLowerCase().equals("female"
              ))
          fcount++;
         else mcount++;
        }//end else
       }
      }

      }//end for loop
     return savelist(count, fcount, mcount,
          request, session);
    }

    @ModelAttribute("uneruptedcount")
    protected List<Integer>
         getUneruptedCount(
         HttpServletRequest request,
         HttpSession session) throws
         Exception {
     List<DentalChart> chart = this.
          getDentalChart(request, session);
           int fcount = 0;
     int mcount = 0;
     int count = 0;

     Date convertedDate=null;
     session=request.getSession(false);
```

```
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());


String condition= request.getParameter(
    "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getUnerupted()[0]!=0
     && chart.get(i).getUnerupted() !=
      null && chart.get(i).
     getIs_current().equals("yes")) {

  Patient patient = querySP.getPatient(
      chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
 return savelist(count, fcount, mcount,
     request, session);
}

@ModelAttribute("impactedcount")
protected List<Integer> getImpactedCount
    (HttpServletRequest request,
    HttpSession session) throws
    Exception {
 List<DentalChart> chart = this.
     getDentalChart(request, session);
  int fcount = 0;
int mcount = 0;
int count = 0;

Date convertedDate=null;
session=request.getSession(false);
QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
    getAttribute("sessionUserRole").
    toString());


String condition= request.getParameter(
    "condd");

for(int i= 0; i < chart.size(); i++) {
 if(chart.get(i).getImpacted()[0]!=0 &&
     chart.get(i).getImpacted() !=
     null && chart.get(i).
     getIs_current().equals("yes")) {

  Patient patient = querySP.getPatient(
      chart.get(i).getPatient_id());
   if(condition.equals("and3")){
    if(checkWithinDate(patient.
        getDatecreated(), request)&&
        checkWithinAge(patient.getAge
        (), request) && checkCity(
        patient.getCity(), request)) {
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }
   }//end if condition and3
   else{

    if(checkWithinDate(patient.
        getDatecreated(), request)){
     count++;
     if(patient.getGender().trim().
         toLowerCase().equals("female"
         ))
      fcount++;
     else mcount++;
    }//end else
   }
  }

 }//end for loop
return savelist(count, fcount, mcount,
    request, session);
```

```
}

@ModelAttribute("porcelainfusedcount")
protected List<Integer>
     getPorcelainfusedCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getPorcelain_crown()
       [0]!=0 && chart.get(i).
       getPorcelain_crown() != null &&
       chart.get(i).getIs_current().
       equals("yes")) {


   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }
    }//end if condition and3
    else{

     if(checkWithinDate(patient.
          getDatecreated(), request)){
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
      else mcount++;
     }//end else
    }
   }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("acryliccrowncount")
protected List<Integer>
     getAcryliccrownCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getAcrylic_crown()
       [0]!=0 && chart.get(i).
       getAcrylic_crown() != null &&
       chart.get(i).getIs_current().
       equals("yes")) {

   Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
    if(condition.equals("and3")){
     if(checkWithinDate(patient.
          getDatecreated(), request)&&
          checkWithinAge(patient.getAge
          (), request) && checkCity(
          patient.getCity(), request)) {
      count++;
      if(patient.getGender().trim().
           toLowerCase().equals("female"
           ))
       fcount++;
```

```
        else mcount++;
       }
      }//end if condition and3
      else{

       if(checkWithinDate(patient.
            getDatecreated(), request)){
        count++;
        if(patient.getGender().trim().
             toLowerCase().equals("female"
             ))
         fcount++;
        else mcount++;
       }//end else
      }
     }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
}

@ModelAttribute("metalcrowncount")
protected List<Integer>
     getMetalcrownCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");

 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getAcrylic_crown()
       [0]!=0 && chart.get(i).
       getAcrylic_crown() != null &&
       chart.get(i).getIs_current().
       equals("yes")) {

    Patient patient = querySP.getPatient(
         chart.get(i).getPatient_id());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

  }//end for loop
 return savelist(count, fcount, mcount,
      request, session);
}

@ModelAttribute("porcelaincrowncount")
protected List<Integer>
     getPorcelaincrownCount(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<DentalChart> chart = this.
      getDentalChart(request, session);
       int fcount = 0;
 int mcount = 0;
 int count = 0;

 Date convertedDate=null;
 session=request.getSession(false);
 QuerySP querySP= new QuerySP();
 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());



 String condition= request.getParameter(
      "condd");
```

```
 for(int i= 0; i < chart.size(); i++) {
  if(chart.get(i).getPorcelain_crown()
       [0]!=0 && chart.get(i).
       getPorcelain_crown() != null &&
       chart.get(i).getIs_current().
       equals("yes")) {

    Patient patient = querySP.getPatient(
        chart.get(i).getPatient_id());
     if(condition.equals("and3")){
      if(checkWithinDate(patient.
           getDatecreated(), request)&&
           checkWithinAge(patient.getAge
           (), request) && checkCity(
           patient.getCity(), request)) {
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }
     }//end if condition and3
     else{

      if(checkWithinDate(patient.
           getDatecreated(), request)){
       count++;
       if(patient.getGender().trim().
            toLowerCase().equals("female"
            ))
        fcount++;
       else mcount++;
      }//end else
     }
    }

   }//end for loop
  return savelist(count, fcount, mcount,
       request, session);
 }
/*
 @ModelAttribute("restorablecount")
 protected List<Integer>
      getRestorableCount(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  List<DentalChart> chart = this.
       getDentalChart(request, session);
        int fcount = 0;
  int mcount = 0;
  int count = 0;

  Date convertedDate=null;
  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());


  for(int i= 0; i < chart.size(); i++) {
   if(chart.get(i).getRestorable() !=
        null && chart.get(i).
        getIs_current().equals("yes")) {
    Patient patient = Context.
         getPatientService().getPatient(
         chart.get(i).getPatientid());
    if(checkWithinDate(patient.
         getDateCreated(), request) &&
         checkWithinAge(patient.getAge(),
          request)) {
     count++;
     if(patient.getGender().toUpperCase()
          .equals("F"))
      fcount++;
     else mcount++;
    }
   }
  }
  return savelist(count, fcount, mcount);
 }

 @ModelAttribute("nonrestorablecount")
 protected List<Integer>
      getNonrestorableCount(
      HttpServletRequest request,
      HttpSession session) throws
      Exception {
  List<DentalChart> chart = this.
       getDentalChart(request, session);
        int fcount = 0;
  int mcount = 0;
  int count = 0;

  Date convertedDate=null;
  session=request.getSession(false);
  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());


  for(int i= 0; i < chart.size(); i++) {
   if(chart.get(i).getNonrestorable() !=
        null && chart.get(i).getIscurrent
        ().equals("yes")) {
    Patient patient = Context.
         getPatientService().getPatient(
         chart.get(i).getPatientid());
    if(checkWithinDate(patient.
```

```
            getDateCreated(), request) &&
            checkWithinAge(patient.getAge(),
             request)) {
        count++;
        if(patient.getGender().toUpperCase()
            .equals("F"))
         fcount++;
        else mcount++;
       }
      }
     }
     return savelist(count, fcount, mcount);
    }
    */
    /*
    @ModelAttribute("theTreatmentPlan")
    protected ArrayList<TreatmentPlan>
         getTreatmentPlans(
         HttpServletRequest request) throws
         Exception {
     ArrayList<TreatmentPlan> treatmentplan
          = Context.getService(DentalService
          .class).getAllTreatmentPlans();
     return treatmentplan;
    }
    */
    @ModelAttribute("caries")
    public String caries(
       @RequestParam(value="caries",
            required = false) String caries)
       {
        return caries;
    }

    @ModelAttribute("recurrentcaries")
    public String recurrentcaries(
       @RequestParam(value="recurrentcaries"
            , required = false) String
            recurrentcaries)
       {
        return recurrentcaries;
    }
    @ModelAttribute("restoration")
    public String restoration(
       @RequestParam(value="restoration",
            required = false) String
            restoration)
       {
        return restoration;
    }
    /* @ModelAttribute("composite")
    public String composite(
       @RequestParam(value="composite",
            required = false) String
            composite)
       {
        return composite;
    }
    @ModelAttribute("glassionomer")
    public String glassionomer(
       @RequestParam(value="glassionomer",
            required = false) String
            glassionomer)
       {
        return glassionomer;
    }
    @ModelAttribute("tempfilling")
    public String tempfilling(
       @RequestParam(value="tempfilling",
            required = false) String
            tempfilling)
       {
        return tempfilling;
    }*/
    @ModelAttribute("extrusion")
    public String extrusion(
       @RequestParam(value="extrusion",
            required = false) String
            extrusion)
       {
        return extrusion;
    }

    @ModelAttribute("intrusion")
    public String intrusion(
       @RequestParam(value="intrusion",
            required = false) String
            intrusion)
       {
        return intrusion;
    }

    @ModelAttribute("mesialdrift")
    public String mesialdrift(
       @RequestParam(value="mesialdrift",
            required = false) String
            mesialdrift)
       {
        return mesialdrift;
    }

    @ModelAttribute("distaldrift")
    public String distaldrift(
       @RequestParam(value="distaldrift",
            required = false) String
            distaldrift)
       {
        return distaldrift;
    }

    @ModelAttribute("completedenture")
    public String completedenture(
```

```
    @RequestParam(value="completedenture"
        , required = false) String
        completedenture)
    {
     return completedenture;
}

@ModelAttribute("singledenture")
public String singledenture(
    @RequestParam(value="singledenture",
        required = false) String
        singledenture)
    {
     return singledenture;
}

@ModelAttribute("removablepartial")
public String removablepartial(
    @RequestParam(value="removablepartial
        ", required = false) String
        removablepartial)
    {
     return removablepartial;
}

@ModelAttribute("rotation")
public String rotation(
    @RequestParam(value="rotation",
        required = false) String
        rotation)
    {
     return rotation;
}

@ModelAttribute("postcorecrown")
public String postcorecrown(
    @RequestParam(value="postcorecrown",
        required = false) String
        postcorecrown)
    {
     return postcorecrown;
}

@ModelAttribute("rootcanal")
public String rootcanal(
    @RequestParam(value="rootcanal",
        required = false) String
        rootcanal)
    {
     return rootcanal;
}

@ModelAttribute("pitandfissure")
public String pitandfissure(
    @RequestParam(value="pitandfissure",
        required = false) String
        pitandfissure)
    {
     return pitandfissure;
}

@ModelAttribute("extracted")
public String extracted(
    @RequestParam(value="extracted",
        required = false) String
        extracted)
    {
     return extracted;
}

@ModelAttribute("missing")
public String missing(
    @RequestParam(value="missing",
        required = false) String missing
        )
    {
     return missing;
}

@ModelAttribute("unerupted")
public String unerupted(
    @RequestParam(value="unerupted",
        required = false) String
        unerupted)
    {
     return unerupted;
}

@ModelAttribute("impacted")
public String impacted(
    @RequestParam(value="impacted",
        required = false) String
        impacted)
    {
     return impacted;
}

@ModelAttribute("porcelainfused")
public String porcelainfused(
    @RequestParam(value="porcelainfused",
         required = false) String
        porcelainfused)
    {
     return porcelainfused;
}

@ModelAttribute("acryliccrown")
public String acryliccrown(
    @RequestParam(value="acryliccrown",
        required = false) String
        acryliccrown)
    {
```

```
    return acryliccrown;
}

@ModelAttribute("metalcrown")
public String metalcrown(
   @RequestParam(value="metalcrown",
        required = false) String
        metalcrown)
   {
    return metalcrown;
}

@ModelAttribute("porcelaincrown")
public String porcelaincrown(
   @RequestParam(value="porcelaincrown",
         required = false) String
        porcelaincrown)
   {
    return porcelaincrown;
}
/*
@ModelAttribute("restorable")
public String restorable(
   @RequestParam(value="restorable",
        required = false) String
        restorable)
   {
    return restorable;
}

@ModelAttribute("nonrestorable")
public String nonrestorable(
   @RequestParam(value="nonrestorable",
        required = false) String
        nonrestorable)
   {
    return nonrestorable;
}
*/
@ModelAttribute("thePatientInfo")
protected ArrayList<PatientInformation>
     getPatientInfo(HttpServletRequest
     request,HttpSession session) throws
      Exception {
  ArrayList<PatientInformation> info =
       null;

 session=request.getSession(false);

 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
       getAttribute("sessionUserRole").
       toString());
  info=querySP.allPatientInfo();
 return info;
}

@ModelAttribute("theConsultations")
protected ArrayList<
     ConsultationsReferrals>
     getConsultation(HttpServletRequest
     request, HttpSession session)
     throws Exception {
session=request.getSession(false);
QuerySP querySP= new QuerySP();

 querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
 ArrayList<ConsultationsReferrals>
      consultArrayList=querySP.
      allConsultation();
 ArrayList<ConsultationsReferrals>
      finalConsultList= new ArrayList<
      ConsultationsReferrals>();
 List<Integer> checkRepeat= new
      ArrayList<Integer>();
 int index=0;
 for(ConsultationsReferrals consult:
      consultArrayList){
  ArrayList<ConsultationsReferrals>
       tempConsult=querySP.
       getConsultations(consult.
       getPatientid());
  int latest=0;
  if(checkRepeat.size()==0){
  if(tempConsult.size() != 0){
   latest = tempConsult.size()-1;
   finalConsultList.add(tempConsult.get
        (latest));
   checkRepeat.add(tempConsult.get(
        latest).getPatientid());
  }
  }else{

   for(int i=0; i<checkRepeat.size();i
       ++){
    if(consult.getPatientid()==
        checkRepeat.get(i))
     break;
    else{
     if(tempConsult.size() != 0){
      latest = tempConsult.size()-1;
      finalConsultList.add(tempConsult.
           get(latest));
      checkRepeat.add(tempConsult.get(
           latest).getPatientid());
      break;
     }
    }
```

```
    }

   }
  }

  return finalConsultList;
}


@ModelAttribute("patient")
protected ArrayList<Patient> getPatients
    (HttpServletRequest request,
    HttpSession session) throws
    Exception {

 session=request.getSession(false);

 ArrayList<Patient> patients=null;


 QuerySP querySP= new QuerySP();

  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());
  patients=querySP.allPatients();

 return patients;
}

@ModelAttribute("datefrom")
public String datefrom(
   @RequestParam(value="datefrom",
        required = false) String
        datefrom)
   {
    return datefrom;
}
@ModelAttribute("dateto")
public String dateto(
   @RequestParam(value="dateto",
        required = false) String dateto)
   {
    return dateto;
}
@ModelAttribute("agefrom")
public String agefrom(
   @RequestParam(value="agefrom",
        required = false) String agefrom
        )
   {
    return agefrom;
}
@ModelAttribute("ageto")
public String ageto(
   @RequestParam(value="ageto", required
        = false) String ageto)
   {
    return ageto;
}
@ModelAttribute("sex")
public String sex(
   @RequestParam(value="sex", required =
        false) String sex)
   {
    return sex;
}
@ModelAttribute("occupation")
public String occupation(
   @RequestParam(value="occupation",
        required = false) String
        occupation)
   {
    return occupation;
}
@ModelAttribute("city")
public String city(
   @RequestParam(value="city", required
        = false) String city)
   {
    return city;
}
@ModelAttribute("agecheck")
public String agecheck(
   @RequestParam(value="agecheck",
        required = false) String
        agecheck)
   {
    return agecheck;
}
@ModelAttribute("sexcheck")
public String sexcheck(
   @RequestParam(value="sexcheck",
        required = false) String
        sexcheck)
   {
    return sexcheck;
}
@ModelAttribute("occupationcheck")
public String occupationcheck(
   @RequestParam(value="occupationcheck"
        , required = false) String
        occupationcheck)
   {
    return occupationcheck;
}



@ModelAttribute("citycheck")
public String citycheck(
   @RequestParam(value="citycheck",
```

```
        required = false) String
        citycheck)
    {
     return citycheck;
}

@ModelAttribute("or1")
public String or1(
    @RequestParam(value="or1", required =
         false) String or1)
    {
     return or1;
}
@ModelAttribute("and1")
public String and1(
    @RequestParam(value="and1", required
        = false) String and1)
    {
     return and1;
}
@ModelAttribute("or2")
public String or2(
    @RequestParam(value="or2", required =
         false) String or2)
    {
     return or2;
}
@ModelAttribute("and2")
public String and2(
    @RequestParam(value="and2", required
        = false) String and2)
    {
     return and2;
}
@ModelAttribute("agegrouppatients2")
protected List<String>
     getAgeGroupPatients2(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<String> list = new ArrayList<
      String>();
  ArrayList<Integer> listpatientids= new
       ArrayList<Integer>();
  boolean patient_exist=false;

 QuerySP querySP= new QuerySP();

   querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());

   String condition= request.
        getParameter("condd");

 ArrayList<Patient> patients = this.
      getPatients(request,session);
 for(int i = 0; i < patients.size(); i
     ++) {
 int patient_id= patients.get(i).
     getPatient_id();
  if(!listpatientids.isEmpty() ||
      listpatientids!=null){
   for(Integer patientidss :
       listpatientids){
    if(patientidss==patient_id)
     patient_exist=true;
   }
  }
  if(patient_exist){
   if(condition.equals("and3")){
    if(checkWithinDate(patients.get(i).
        getDatecreated(), request)&&
        checkWithinAge(patients.get(i).
        getAge(), request) && checkCity
        (patients.get(i).getCity(),
        request)) {

    list.add(Integer.toString(patients.
        get(i).getPatient_id()));
    //list.add(patients.get(i).
        getGivenName()+" "+ patients.get
        (i).getMiddleName()+ " "+
        patients.get(i).getFamilyName())
        ;
    //list.add(patients.get(i).getAge().
        toString());
   }
  }else{

   if(checkWithinDate(patients.get(i).
       getDatecreated(), request)){
    list.add(Integer.toString(patients.
        get(i).getPatient_id()));

   }//end else
  }
   }
   patient_exist=false;
  }

 return list;
}

@ModelAttribute("results")
protected List<Integer>
     getResultsPatients(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
 List<String> glist =
      getAgeGroupPatients2(request,
```

```
        session);
int count = 0, fcount = 0, mcount = 0;
int i = 0;

QuerySP querySP= new QuerySP();
querySP.setDatabase_username(session.
        getAttribute("sessionUserRole").
        toString());

while(i < glist.size()) {
 List<DentalChart> chart = querySP.
        getDentalChart(Integer.parseInt(
        glist.get(i)));

 String caries = request.getParameter("
        caries");
 if(caries == null) caries="no";
 String recurrent = request.
        getParameter("recurrentcaries");
 if(recurrent == null) recurrent="no";
 String restoration = request.
        getParameter("restoration");
 if(restoration == null) restoration="
        no";
 String extrusion = request.
        getParameter("extrusion");
 if(extrusion == null) extrusion="no";
 String intrusion = request.
        getParameter("intrusion");
 if(intrusion == null) intrusion="no";
 String mesial = request.getParameter("
        mesialdrift");
 if(mesial == null) mesial="no";
 String distal = request.getParameter("
        distaldrift");
 if(distal == null) distal="no";
 String rotation = request.getParameter
        ("rotation");
 if(rotation == null) rotation="no";
 String extracted = request.
        getParameter("extracted");
 if(extracted == null) extracted="no";
 String complete = request.getParameter
        ("completedenture");
 if(complete == null) complete="no";
 String single = request.getParameter("
        singledenture");
 if(single == null) single="no";
 String removable = request.
        getParameter("removablepartial");
 if(removable == null) removable="no";
 String pitandfissure = request.
        getParameter("pitandfissure");
 if(pitandfissure == null)
        pitandfissure="no";
 String missing = request.getParameter(
        "missing");
 if(missing == null) missing="no";
 String rootcanal = request.
        getParameter("rootcanal");
 if(rootcanal == null) rootcanal="no";
 String impacted = request.getParameter
        ("impacted");
 if(impacted == null) impacted="no";
 String postcore = request.getParameter
        ("postcorecrown");
 if(postcore == null) postcore="no";
 String acrylic = request.getParameter(
        "acryliccrown");
 if(acrylic == null) acrylic="no";
 String metal = request.getParameter("
        metalcrown");
 if(metal == null) metal="no";
 String porcelain = request.
        getParameter("porcelaincrown");
 if(porcelain == null) porcelain="no";
 String unerupted = request.
        getParameter("unerupted");
 if(unerupted == null) unerupted="no";
 String porcelainfused = request.
        getParameter("porcelainfused");
 if(porcelainfused == null)
        porcelainfused="no";
 boolean valid = true;
 for(int j = 0; j < chart.size(); j++)
        {
  if(chart.get(j).getIs_current().
         equals("yes")){
   if((chart.get(j).getCaries()[0] == 0
          && caries.equals("yes")) || (
          chart.get(j).getRecurrentcaries
          ()[0] == 0 && recurrent.equals(
          "yes")) || (chart.get(j).
          getRestoration() == null &&
          restoration.equals("yes"))
      || (chart.get(j).getExtrusion()[0]
           == 0 && extrusion.equals("
          yes")) || (chart.get(j).
          getIntrusion()[0] == 0 &&
          intrusion.equals("yes"))
      || (chart.get(j).
          getMesial_rotation()[0] == 0
          && mesial.equals("yes")) || (
          chart.get(j).
          getDistal_rotation()[0] == 0
          && distal.equals("yes"))
      || (chart.get(j).getRotation()[0]
          == 0 && rotation.equals("yes"
          )) || (chart.get(j).
          getExtracted()[0] == 0 &&
          extracted.equals("yes"))
      || (chart.get(j).getImpacted()[0]
```

```
            == 0 && impacted.equals("yes"
            )) || (chart.get(j).
            getMissing()[0] == 0 &&
            missing.equals("yes"))
        || (chart.get(j).
            getRootcanal_treatment()[0]
            == 0 && rootcanal.equals("yes
            ")) || (chart.get(j).
            getPitfissure_sealants()[0]
            == 0 && pitandfissure.equals(
            "yes"))
        || (chart.get(j).getPostcore_crown
            ()[0] == 0 && postcore.equals
            ("yes")) || (chart.get(j).
            getAcrylic_crown()[0] == 0 &&
             acrylic.equals("yes"))
        || (chart.get(j).getMetal_crown()
            [0] == 0 && metal.equals("yes
            ")) || (chart.get(j).
            getPorcelain_crown()[0] == 0
            && porcelain.equals("yes"))
        || (chart.get(j).getUnerupted()[0]
             == 0 && unerupted.equals("
            yes")) || (chart.get(j).
            getPorcelain_infused()[0] ==
            0 && porcelainfused.equals("
            yes"))
        || (chart.get(j).
            getComplete_denture() == null
             && complete.equals("yes"))
            || (chart.get(j).
            getSingle_denture() == null
            && single.equals("yes")
        || (chart.get(j).
            getRemovable_partial_denture
            ()[0] == 0 && removable.
            equals("yes")))) {
      valid = false;
     }
    }
   }
   if(valid) {
    Patient patient = querySP.getPatient(
        Integer.parseInt(glist.get(i)));
    i++;
    count++;
    if(patient.getGender().trim().
        toLowerCase().equals("female"))
     fcount++;
    else mcount++;
   }
   else glist.remove(i);
  }
  return savelist(count, fcount, mcount,
      request,session);
 }

 @ModelAttribute("results2")
 protected List<Integer>
     getResultsPatients2(
     HttpServletRequest request,
     HttpSession session) throws
     Exception {
  List<String> glist =
      getAgeGroupPatients2(request,
      session);
  int count = 0, fcount = 0, mcount = 0;
  int i = 0;

  QuerySP querySP= new QuerySP();
  querySP.setDatabase_username(session.
      getAttribute("sessionUserRole").
      toString());

  while(i < glist.size()) {
   List<ServiceNeeded> service = querySP.
       getServiceNeeded(Integer.parseInt
       (glist.get(i)));
   String periodontics = request.
       getParameter("periodontics");
   if(periodontics == null) periodontics=
       "no";
   String class1 = request.getParameter("
       class1");
   if(class1 == null) class1="no";
   String class2 = request.getParameter("
       class2");
   if(class2 == null) class2="no";
   String class3 = request.getParameter("
       class3");
   if(class3 == null) class3="no";
   String class4 = request.getParameter("
       class4");
   if(class4 == null) class4="no";
   String class5 = request.getParameter("
       class5");
   if(class5 == null) class5="no";
   String onlay = request.getParameter("
       onlay");
   if(onlay == null) onlay="no";
   String extraction = request.
       getParameter("extraction");
   if(extraction == null) extraction="no"
       ;
   String odontectomy = request.
       getParameter("odontectomy");
   if(odontectomy == null) odontectomy="
       no";
   String specialcase = request.
       getParameter("specialcase");
   if(specialcase == null) specialcase="
       no";
```

```
String pedodontics = request.
    getParameter("pedodontics");
if(pedodontics == null) pedodontics="
    no";
String orthodontics = request.
    getParameter("orthodontics");
if(orthodontics == null) orthodontics=
    "no";
String pulpsedation = request.
    getParameter("pulpsedation");
if(pulpsedation == null) pulpsedation=
    "no";
String crownrecementation = request.
    getParameter("crownrecementation"
    );
if(crownrecementation == null)
    crownrecementation="no";
String tempfillingservice = request.
    getParameter("tempfillingservice"
    );
if(tempfillingservice == null)
    tempfillingservice="no";
String acuteinfections = request.
    getParameter("acuteinfections");
if(acuteinfections == null)
    acuteinfections="no";
String traumaticinjuries = request.
    getParameter("traumaticinjuries")
    ;
if(traumaticinjuries == null)
    traumaticinjuries="no";
String laminated = request.
    getParameter("laminated");
if(laminated == null) laminated="no";
String singlecrown = request.
    getParameter("singlecrown");
if(singlecrown == null) singlecrown="
    no";
String bridgeservice = request.
    getParameter("bridgeservice");
if(bridgeservice == null)
    bridgeservice="no";
String anterior = request.getParameter
    ("anterior");
if(anterior == null) anterior="no";
String posterior = request.
    getParameter("posterior");
if(posterior == null) posterior="no";
String completedentservice = request.
    getParameter("completedentservice
    ");
if(completedentservice == null)
    completedentservice="no";
String singledentservice = request.
    getParameter("singledentservice")
    ;
if(singledentservice == null)
    singledentservice="no";
String removablepartialservice =
    request.getParameter("
    removablepartialservice");
if(removablepartialservice == null)
    removablepartialservice="no";
boolean valid = true;
for(int j = 0; j < service.size(); j
    ++) {
 if(service.get(j).getIs_current().
     equals("yes")){
  if((service.get(j).getPeriodontics()
      == null && periodontics.equals
      ("yes")) || (service.get(j).
      getClass_1()[0] == 0 && class1.
      equals("yes"))
    || (service.get(j).getClass_2()[0]
       == 0 && class2.equals("yes")
      ) || (service.get(j).
      getClass_3()[0] == 0 &&
      class3.equals("yes"))
    || (service.get(j).getClass_4()[0]
       == 0 && class4.equals("yes")
      ) || (service.get(j).
      getClass_5()[0] == 0 &&
      class5.equals("yes"))
    || (service.get(j).getOnlay()[0]
      == 0 && onlay.equals("yes"))
      || (service.get(j).
      getSpecial_case()[0] == 0 &&
      specialcase.equals("yes"))
    || (service.get(j).getExtraction()
      [0] == 0 && extraction.equals
      ("yes")) || (service.get(j).
      getOdontectomy()[0] == 0 &&
      odontectomy.equals("yes"))
    || (service.get(j).getSurgery() ==
       null && pedodontics.equals("
      yes")) || (service.get(j).
      getSurgery() == null &&
      orthodontics.equals("yes"))
    || (service.get(j).
      getEmergency_treatment() ==
      null && acuteinfections.
      equals("yes")) || (service.
      get(j).getEmergency_treatment
      () == null &&
      traumaticinjuries.equals("yes
      "))
    || (service.get(j).
      getPulp_sedation()[0] == 0 &&
       pulpsedation.equals("yes"))
      || (service.get(j).
      getCrown_recementation()[0]
      == 0 && crownrecementation.
```

```
            equals("yes"))
        || (service.get(j).
            getFilling_service()[0] == 0
            && tempfillingservice.equals(
            "yes")) || (service.get(j).
            getLaminated()[0] == 0 &&
            laminated.equals("yes"))
        || (service.get(j).getSingle_crown
            ()[0] == 0 && singlecrown.
            equals("yes")) || (service.
            get(j).getBridge_service()[0]
             == 0 && bridgeservice.equals
            ("yes"))
        || (service.get(j).
            getProsthodontics() == null
            && completedentservice.equals
            ("yes")) || (service.get(j).
            getProsthodontics() == null
            && singledentservice.equals("
            yes"))
        || (service.get(j).
            getProsthodontics() == null
            && removablepartialservice.
            equals("yes")) || (service.
            get(j).getAnterior()[0] == 0
            && anterior.equals("yes")
            || (service.get(j).getPosterior()
                [0] == 0 && posterior.equals(
                "yes")))) {
          valid = false;
        }
       }
      }
      if(valid) {
       Patient patient = querySP.getPatient(
           Integer.parseInt(glist.get(i)));
       i++;
       count++;
       if(patient.getGender().trim().
           toLowerCase().equals("female"))
        fcount++;
       else mcount++;
      }
      else glist.remove(i);
     }
     return savelist(count, fcount, mcount,
         request,session);
    }

}
```

Listing 39: User

```
package com.dentist.version.three.form;

//User object
public class User {

 private int user_id;
 private String fname_user;
 private String minit_user;
 private String lname_user;
 private String username;
 private String password;
 private String email;
 private String secret_question;
 private String secret_answer;
 private String created_date;
 public void setUser_id(int user_id) {
  this.user_id = user_id;
 }
 public int getUser_id() {
  return user_id;
 }
 public void setFname_user(String
     fname_user) {
  this.fname_user = fname_user;
 }
 public String getFname_user() {
  return fname_user;
 }
 public void setMinit_user(String
     minit_user) {
  this.minit_user = minit_user;
 }
 public String getMinit_user() {
  return minit_user;
 }
 public void setLname_user(String
     lname_user) {
  this.lname_user = lname_user;
 }
 public String getLname_user() {
  return lname_user;
 }
 public void setUsername(String username)
     {
  this.username = username;
 }
 public String getUsername() {
  return username;
 }
 public void setPassword(String password)
     {
  this.password = password;
 }
 public String getPassword() {
```

```
 return password;
}
public void setEmail(String email) {
 this.email = email;
}
public String getEmail() {
 return email;
}
public void setSecret_question(String
    secret_question) {
 this.secret_question = secret_question;
}
public String getSecret_question() {
 return secret_question;
}

public void setCreated_date(String
    created_date) {
 this.created_date = created_date;
}
public String getCreated_date() {
 return created_date;
}
public void setSecret_answer(String
    secret_answer) {
 this.secret_answer = secret_answer;
}
public String getSecret_answer() {
 return secret_answer;
}


}
```

Listing 40: UserMapper

```
package com.dentist.version.three.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.
    RowMapper;

import com.dentist.version.three.form.
    User;


public class UserMapper implements
    RowMapper<User> {
    public User mapRow(ResultSet rs, int
        rowNum) throws SQLException {
       User user = new User();
      user.setUser_id(rs.getInt("user_id"
          ));
      user.setFname_user(rs.getString("
          fname_user"));
      user.setMinit_user(rs.getString("
          minit_user"));
      user.setLname_user(rs.getString("
          lname_user"));
      user.setUsername(rs.getString("
          username"));
      user.setPassword(rs.getString("
          password"));
      user.setEmail(rs.getString("email")
          );
      user.setSecret_question(rs.
          getString("secret_question"));
      user.setSecret_answer(rs.getString(
          "secret_answer"));
      user.setCreated_date(rs.getString("
          created_date"));



        return user;
    }
 }
```

Listing 41: UserRoleSection

```
package com.dentist.version.three.form;

import java.util.ArrayList;


public class UserRoleSection {

 private String user_name;
 private int section_id;
 private String user_role;
 private ArrayList<Section> user_section;
 private String full_name;

public void setUser_name(String
    user_name) {
 this.user_name = user_name;
}
public String getUser_name() {
 return user_name;
}

public void setUser_section(ArrayList<
    Section> user_section) {
```

```
 this.user_section = user_section;
}
public ArrayList<Section>
     getUser_section() {
 return user_section;
}
public void setSection_id(int section_id
     ) {
 this.section_id = section_id;
}
public int getSection_id() {
 return section_id;
}
public void setFull_name(String
     full_name) {
 this.full_name = full_name;
}
public String getFull_name() {
 return full_name;
}
public void setUser_role(String
     user_role) {
 this.user_role = user_role;
}
public String getUser_role() {
 return user_role;
}

}
```

Listing 42: UserSP

```
package com.dentist.version.three.db;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import com.dentist.version.three.form.
    Role;
import com.dentist.version.three.form.
    User;
import com.dentist.version.three.mapper.
    GeneratePassword;
import com.dentist.version.three.mapper.
    RoleMapper;
import com.dentist.version.three.mapper.
    UserMapper;

public class UserSP {

 private String database_username;
 private final String database_password="
     yes?bakitpo?";

 public void setDatabase_username(String
     database_username) {
  this.database_username =
      database_username;
 }


 public String getDatabase_username() {
  return database_username;
 }

 //insert user
  public void executeUser(String
      fname_user, String minit_user,
      String lname_user, String username
      ,
     String password, String email,
         String secret_question, String
         secret_answer, String
         created_date) throws Exception{

   Class.forName("org.postgresql.Driver")
       .newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

     CallableStatement calstat=conn.
         prepareCall("{call
         insert_users
         (?,?,?,?,?,?,?,?,?)}");

       calstat.setString(1,
           fname_user);
      calstat.setString(2,minit_user
          );
      calstat.setString(3,lname_user
          );
      calstat.setString(4, username)
          ;
      calstat.setString(5, password)
          ;
      calstat.setString(6, email);
      calstat.setString(7,
          secret_question);
```

```
            calstat.setString(8,
                 secret_answer);
            calstat.setString(9,
                 created_date);



            ResultSet rs= calstat.
                 executeQuery();

            conn.close();
            calstat.close();
            System.out.println("Your_data_
                 has_been_inserted_into_
                 table.");


 }

// get user id
public int getUserID(String username)
     throws Exception{
  Class.forName("org.postgresql.Driver")
       .newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
   conn.setAutoCommit(false);

     PreparedStatement calstat=conn.
          prepareCall("{call_getUserID
          (?)}");
         calstat.setString(1, username);
          ResultSet rs= calstat.
               executeQuery();
        int result=0;

        while(rs.next()){
                 System.out.println(rs.
                      getInt(1));
                 result= rs.getInt(1);
               }


          conn.close();
          calstat.close();

          return result;


}

// add role to user
public void addUserRole(int user,
     String role) throws Exception{

  Class.forName("org.postgresql.Driver")
       .newInstance();
    Connection conn=DriverManager.
         getConnection("jdbc:postgresql
         ://localhost:5432/DentISt",
         database_username,
         database_password);

      CallableStatement calstat=conn.
           prepareCall("{call_apply_role
           (?,_?)}");
          calstat.setInt(1,user);
          calstat.setString(2,role);


          ResultSet rs = calstat.
               executeQuery();
          conn.close();
          calstat.close();
   System.out.println("Added_role_to_a_
        user");

}

//list of username
public ArrayList<String>
     getUsernameList() throws Exception
     {
  ArrayList<String> listUsername= new
       ArrayList<String>();
  Class.forName("org.postgresql.Driver")
       .newInstance();
  System.out.println("Check"+
       database_username);
  Connection conn=DriverManager.
       getConnection("jdbc:postgresql://
       localhost:5432/DentISt",
       database_username,
       database_password);


    conn.setAutoCommit(false);
    CallableStatement calstat=conn.
         prepareCall("{call_
         listallusername()}");

      ResultSet rs = calstat.
           executeQuery();

      while(rs.next()){
          // System.out.println(rs.
               getString(1));
     listUsername.add(rs.getString(1));
          }
```

```
    conn.close();
    calstat.close();
    System.out.println("Successful call
        for listallusername function"
        );

  return listUsername;

}

//list of users
public ArrayList<User> allUserList()
    throws Exception{
 ArrayList<User> listUsers= new
     ArrayList<User>();
 Class.forName("org.postgresql.Driver")
     .newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call listallusers
       ()}");

     ResultSet rs = calstat.
         executeQuery();

     while(rs.next()){
         User user= new User();
         UserMapper usermap= new
             UserMapper();
         user= usermap.mapRow(rs, 10);



         listUsers.add(user);
     //System.out.println(rs.getString
         (1));
        }



    conn.close();
    calstat.close();
    System.out.println("Successful call
        for listallusers function");
     return listUsers;
}

//get User
public User getUser(int user_id) throws
     Exception{
 Class.forName("org.postgresql.Driver")
     .newInstance();
  Connection conn=DriverManager.
      getConnection("jdbc:postgresql
      ://localhost:5432/DentISt",
      database_username,
      database_password);
 conn.setAutoCommit(false);
  CallableStatement calstat=conn.
      prepareCall("{call getuser(?)}"
      );
   calstat.setInt(1, user_id);

    ResultSet rs = calstat.
        executeQuery();
       User user= new User();
       UserMapper usermap= new
           UserMapper();
    while(rs.next()){

    user= usermap.mapRow(rs, 10);

     //System.out.println(rs.getString
         (1));
       }



    conn.close();
    calstat.close();
    System.out.println("Successful call
        for getuser function");
     return user;
}

//update User
 public void updateUser(int user_id,
     String fname_user, String
     minit_user,String lname_user,
     String username,
    String password, String email,
        String secret_question, String
         secret_answer) throws
        Exception{

  Class.forName("org.postgresql.Driver"
      ).newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);

     CallableStatement calstat=conn.
         prepareCall("{call
         update_users
         (?,?,?,?,?,?,?,?,?)}");
       //calstat.registerOutParameter
           (10, java.sql.Types.
```

```
            INTEGER);
        calstat.setInt(1,user_id);
          calstat.setString(2,
              fname_user);
        calstat.setString(3,
            minit_user);
        calstat.setString(4,
            lname_user);
        calstat.setString(5, username
            );
        calstat.setString(6, password
            );
        calstat.setString(7, email);
        calstat.setString(8,
            secret_question);
        calstat.setString(9,
            secret_answer);



        ResultSet rs= calstat.
            executeQuery();



        conn.close();
        calstat.close();
        System.out.println("Your data
            has been updated into
            table.");


  }
//get User
 public void deleteUser(int user_id)
     throws Exception{
  Class.forName("org.postgresql.Driver"
      ).newInstance();
    Connection conn=DriverManager.
        getConnection("jdbc:postgresql
        ://localhost:5432/DentISt",
        database_username,
        database_password);
  // conn.setAutoCommit(false);
    CallableStatement calstat=conn.
        prepareCall("{call
        delete_users(?)}");
      calstat.setInt(1, user_id);

      ResultSet rs = calstat.
          executeQuery();



    conn.close();
    calstat.close();
    System.out.println("Successful
        call for DELETE function");
}


// getuserrole
public ArrayList<Integer> getUserRole(
    int user_id) throws Exception{

 Class.forName("org.postgresql.Driver"
     ).newInstance();
   Connection conn=DriverManager.
       getConnection("jdbc:postgresql
       ://localhost:5432/DentISt",
       database_username,
       database_password);
 // conn.setAutoCommit(false);
   CallableStatement calstat=conn.
       prepareCall("{call getuserrole
       (?)}");
     calstat.setInt(1, user_id);

     ResultSet rs = calstat.
         executeQuery();

     ArrayList<Integer> role_id= new
         ArrayList<Integer>();
     while(rs.next()){
         role_id.add(rs.getInt(1));
         System.out.println("THIS IS
             IT :"+role_id);
       }

     conn.close();
    calstat.close();
    System.out.println("Successful
        call for login user function"
        );

    return role_id;
}



//change password in profile
 public void changePassword(int
     user_id, String newPassword){
  try {
   Class.forName("org.postgresql.
       Driver").newInstance();

    Connection conn=DriverManager.
        getConnection("jdbc:
        postgresql://localhost:5432/
        DentISt",database_username,
        database_password);

     CallableStatement calstat=conn.
```

```
            prepareCall("{call
            changepassword(?,?)}");
          //calstat.registerOutParameter
              (10, java.sql.Types.
              INTEGER);
            calstat.setInt(1,user_id);
              calstat.setString(2,
                  newPassword);

            ResultSet rs= calstat.
                executeQuery();

            conn.close();
            calstat.close();
            System.out.println("
                  Successul in changing
                  password.");
    } catch (InstantiationException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
    } catch (IllegalAccessException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
    } catch (ClassNotFoundException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
    } catch (SQLException e) {
     System.out.println("ERROR");
     // TODO Auto-generated catch block
     e.printStackTrace();
    }
    }


//change secret q&a in profile
    public void changeSecret(int user_id,
          String question, String answer)
          throws Exception{
     Class.forName("org.postgresql.Driver
          ").newInstance();
       Connection conn=DriverManager.
            getConnection("jdbc:
            postgresql://localhost:5432/
            DentISt",database_username,
            database_password);

          CallableStatement calstat=conn.
              prepareCall("{call
              changesecret(?,?,?)}");
            //calstat.registerOutParameter
                (10, java.sql.Types.
                INTEGER);
              calstat.setInt(1,user_id);
                calstat.setString(2,
                    question);
                calstat.setString(3,
                    answer);

              ResultSet rs= calstat.
                  executeQuery();

              conn.close();
              calstat.close();
              System.out.println("
                    Successul in changing
                    secret Q&A.");

    }


//update profile
    public void changeProfile(int user_id
        , String email){
     try {
      Class.forName("org.postgresql.
          Driver").newInstance();

       Connection conn=DriverManager.
            getConnection("jdbc:
            postgresql://localhost:5432/
            DentISt",database_username,
            database_password);

          CallableStatement calstat=conn.
              prepareCall("{call
              changeprofile(?,?)}");
            //calstat.registerOutParameter
                (10, java.sql.Types.
                INTEGER);
              calstat.setInt(1,user_id);
                calstat.setString(2, email
                    );

              ResultSet rs= calstat.
                  executeQuery();

              conn.close();
              calstat.close();
              System.out.println("
                    Successul in changing
                    email.");
     } catch (InstantiationException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     } catch (IllegalAccessException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     } catch (ClassNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     } catch (SQLException e) {
      System.out.println("ERROR");
```

```
  // TODO Auto-generated catch block
  e.printStackTrace();
 }
 }


//list of searched users
public ArrayList<User>
     specifiedUserList(String
     specifiedSearch) throws
     Exception{
 ArrayList<User> listUsers= new
      ArrayList<User>();
 Class.forName("org.postgresql.Driver
     ").newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:
        postgresql://localhost:5432/
        DentISt",database_username,
        database_password);
  conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call
        listspecifiedusers(?)}");
         calstat.setString(1,
              specifiedSearch);

     ResultSet rs = calstat.
          executeQuery();

     while(rs.next()){
          User user= new User();
          UserMapper usermap= new
               UserMapper();
          user= usermap.mapRow(rs, 10)
               ;



          listUsers.add(user);
     //System.out.println(rs.
          getString(1));
         }


    conn.close();
    calstat.close();
    System.out.println("Successful
         call for listspecifiedusers
         function");
     return listUsers;
}

//delete role of a user
public void deleteUserRole(int
     user_id,int role_id) throws
     Exception{
 Class.forName("org.postgresql.Driver
     ").newInstance();
   Connection conn=DriverManager.
        getConnection("jdbc:
        postgresql://localhost:5432/
        DentISt",database_username,
        database_password);
 // conn.setAutoCommit(false);
   CallableStatement calstat=conn.
        prepareCall("{call
        delete_userrole(?,?)}");
     calstat.setInt(1, user_id);
     calstat.setInt(2, role_id);

     ResultSet rs = calstat.
          executeQuery();



    conn.close();
    calstat.close();
    System.out.println("Successful
         call for DELETE function");
}



// NEW FOR MULTI-ROLES

   //lists of roles of a user
    public  ArrayList<Role>
         getListRolesofUser(int user_id)
          throws Exception{

   ArrayList<Role> roleLists= new
        ArrayList<Role>();

   Class.forName("org.postgresql.
        Driver").newInstance();
     Connection conn=DriverManager.
        getConnection("jdbc:
        postgresql://localhost:5432/
        DentISt",database_username,
        database_password);

      CallableStatement calstat=conn.
           prepareCall("{call
           listrolesofuser(?)}");
       //calstat.
            registerOutParameter(10,
             java.sql.Types.INTEGER)
            ;
        calstat.setInt(1,user_id);

        ResultSet rs= calstat.
             executeQuery();
```

```
            while(rs.next()){

             Role role= new Role();
                    RoleMapper rolemap=
                        new RoleMapper
                        ();
                   role= rolemap.mapRow(
                       rs, 5);


               roleLists.add(role);
            }

            conn.close();
            calstat.close();
            System.out.println("
                 Function has been
                 called.");

            return roleLists;
        }
    }
```

Listing 43: addSectionRole

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<title>Insert title here</title>

<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>

<script>

$(document).ready(function(){

 $('#roleButton').click(function(){
  if($(this).val()=="Add Roles"){
   $("#roleAppear").show();
   $("#roleButton").val('Cancel');
  }
  else if($(this).val()=="Cancel"){
   $("#roleAppear").hide();
   $("#roleButton").val('Add Roles');
  }
 });
 $('#addRoleSubmit').click(function(){
  $('#addroleform').submit();
 });

 $('#showUserList').click(function(){
  if($(this).val()=="View Users"){
   $("#showUserDiv").show();
   $("#showUserList").val('Cancel');
  }
  else if($(this).val()=="Cancel"){
   $("#showUserDiv").hide();
   $("#showUserList").val('View Users');
  }
 });

 $('#deleteSubmit').click(function(){
  deleteValidate();
 });

});



function deleteValidate(){
  var r=confirm("Are you sure to delete
       this role in this section?");
  if (r==true)
   $('#deleteForm').submit();

}

</script>

</head>
<body>



<div class="tab-body">

<c:url var="addRoleURL" value="/forms/
     dentist/simpleForm/addSectionRole?
     section_id=${section.section_id}" />
<c:url var="editSectionURL" value="/forms
     /dentist/simpleForm/editSectionForm?
     section_id=${section.section_id}" />

<ul class = "tabs primary">
    <li class = "active"><a href = "${
         addRoleURL}" class = "active">
         Apply Role</a></li>
```

```
    <li><a href = "${editSectionURL}" >
        Edit Section</a></li>

</ul>

<c:url var="addRoleURL" value="/forms/
    dentist/simpleForm/addSectionRole"
    />

<form:form action="${addRoleURL}"
    commandName="roleSection" >
<form:hidden path="section_id"/>

<h2>Apply Roles to Section</h2>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
     span>
</c:if>
<div class="clear"></div>
</div>
  <form:select path="role_name">
       <form:option value="${selected}"
           >Choose</form:option>
       <c:forEach var="allRole" items="$
           {allRoles}">
       <c:if test="${allRole != selected
           }">
              <c:set var="
                  conditionVariable"
                   value ="YES"/>

         <c:forEach items="${
             currentRoleLists}" var="
             currentRoleList" varStatus
             ="loop">


      <c:if test='${currentRoleList.
          role_name == allRole}'>
                        <c:set var="
                            conditionVariable
                            " value
                            ="NO"
                            />
      </c:if>

              </c:forEach>

          <c:if test="${
              conditionVariable eq '
              YES'}">
           <form:option value="${
               allRole}">${allRole}</
               form:option>
          </c:if>
          </c:if>
          </c:forEach>
        </form:select>
       <input type="submit" value="Apply Role
           " class="classname"/>

 </form:form>
<br/>

<c:url var="deleteRoleURL" value="/forms/
    dentist/simpleForm/deleteRoleSection
    .html" />

<form:form action="${deleteRoleURL}"
    method="POST" commandName="
    deleteFunction" id="deleteForm">
<input type="button" value="Delete Roles"
     id="deleteSubmit" class="classname
    "/>
<br/>
<br/>
<form:hidden path="key_id"/>

<table class="list-table">
<thead>
<tr class="head-list-table">
<td>Role Name</td>
<td>Role Description</td>
</tr>
</thead>

<tbody>

  <c:forEach items="${currentRoleLists}"
      var="currentRoleList" varStatus="
      loop">

<c:choose>
     <c:when test='${(loop.index)%2 eq
         0}'>
<c:set var="rowColor" value="even"
         scope="page"/>
     </c:when>
     <c:otherwise>
       <c:set var="rowColor" value="odd"
           scope="page"/>
     </c:otherwise>
   </c:choose>

<tr class="${rowColor}">

 <td>
```

```
        <form:checkbox path="deleteObject
            " value="${currentRoleList.
            role_id}"/> ${
            currentRoleList.role_name}
    </td>
 <td>
        ${currentRoleList.
            role_description}
    </td>

 </tr>
  </c:forEach>
<c:if test=" ${empty currentRoleLists}">
<tr><td colspan="2">No roles</td>
</tr>
</c:if>
</tbody>
</table>
</form:form>
<br/>
<input type="button" id="showUserList"
     value="View Users" class="classname
     "/>
<div style="display:none" id="showUserDiv
     ">
<table class="list-table">

<tr class="head-list-table">
<td>List of Users</td>
</tr>
  <c:forEach items="${currentUserLists}"
      var="currentUserList" varStatus="
      loop">
     <c:forEach items="${currentUserList
         }" var="currentUser" varStatus
         ="loop">

<c:choose>
     <c:when test='${(loop.index)%2 eq
         0}'>
      <c:set var="rowColor" value="even"
          scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
          scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">


 <td>
        ${currentUser.fname_user} ${
            currentUser.minit_user}  ${
            currentUser.lname_user}
    </td>

 </tr>
    </c:forEach>
  </c:forEach>
</table>
</div>


</div>
</body>
</html>
```

Listing 44: AddUserRole

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<title>Add User Role</title>
<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>
<script>

$(document).ready(function(){
 $('#deleteSubmit').click(function(){
  deleteValidate();
 });

});



function deleteValidate(){
  var r=confirm("Are you sure to delete
       this role?");
  if (r==true)
   $('#deleteForm').submit();
```

```
}
</script>
</head>
<body>
<div class="tab-body">
<c:url var="addUserURL" value="/forms/
    dentist/simpleForm/addUserRole?
    user_id=${user.user_id}" />
<c:url var="editUserURL" value="/forms/
    dentist/simpleForm/editUserForm?
    user_id=${user.user_id}" />
<c:url var="deleteRoleURL" value="/forms/
    dentist/simpleForm/deleteUserRole.
    html" />

<ul class = "tabs primary">

    <li class = "active"><a href = "${
        addUserURL}" class = "active">
        Apply Role</a></li>

    <li ><a href = "${editUserURL}" >Edit
        User</a></li>

</ul>

<h2>Apply Role to User</h2>
<h3>${user.fname_user}  ${user.minit_user
    }  ${user.lname_user}</h3>

<c:if test="${not empty errorMessage}">
 <span style="color:red">${errorMessage
     }</span>
</c:if>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
     span>
</c:if>
<div class="clear"></div>
</div>

<form:form action="UserRoleConfirm.html"
    commandName="roleuser" >
     <form:hidden path="user_id"/>

  <form:select path="role_name">
       <form:option value="${selected}"
           >Choose</form:option>
       <c:forEach var="allRole" items="$
           {allRoles}">
       <c:if test="${allRole != selected
           }">
            <c:set var="
                conditionVariable"
                 value ="YES"/>
            <c:forEach items="${
                currentRoleLists}" var="
                currentRoleList" varStatus
                ="loop">


        <c:if test='${currentRoleList.
            role_name == allRole}'>
                    <c:set var="
                        conditionVariable
                        " value
                        ="NO"
                        />
        </c:if>

            </c:forEach>

        <c:if test="${
            conditionVariable eq '
            YES'}">
         <form:option value="${
             allRole}">${allRole}</
             form:option>
        </c:if>

       </c:if>
       </c:forEach>
    </form:select>
   <td><input type="submit" value="Apply
       Role" class="classname"/></td>

</form:form>


<br/>
<br/>

<form:form action="${deleteRoleURL}"
    method="POST" commandName="
    deleteFunction" id="deleteForm">
<form:hidden path="key_id"/>

<table class="list-table">
<thead>
<tr class="head-list-table">
<td>Role Name</td>
<td>Role Description</td>
</tr>
</thead>

<tbody>

  <c:forEach items="${currentRoleLists}"
      var="currentRoleList" varStatus="
      loop">
```

```
<c:choose>
     <c:when test='${(loop.index)%2 eq
          0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">



 <td>
        <form:checkbox path="deleteObject
             " value="${currentRoleList.
             role_id}"/> ${
             currentRoleList.role_name}
    </td>
 <td>
            ${currentRoleList.
                 role_description}
    </td>

 </tr>
  </c:forEach>
<c:if test=" ${empty currentRoleLists}">
<tr><td colspan="2">No roles</td>
</tr>
</c:if>
</tbody>
</table>
</form:form>
 <br/>
 <br/>
 <input type="button" value="Delete Roles
      " id="deleteSubmit" class="
      classname"/>

  </div>

</body>
</html>
```

Listing 45: auditTrail

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>


<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript" src="<c:
     url value="/showresult.js" />" >
</script>
<script type="text/javascript" src="<c:
     url value="/calendar.js" />" >
</script>
<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>
<script>

$(document).ready(function(){

 $('#searchSubmit').click(function(){
  auditValidate();
 });
});

function auditValidate(){
 var action= document.getElementById('
      action').value;
 var auditConfirm="";


 //check for missing input
 if(action=="")
  auditConfirm+="Action Selection is
       required before searching. Pls
       choose an action.";

 if(auditConfirm==""){
   $('#searchForm').submit();
 }
 else
  alert(auditConfirm);


}

</script>
</head>
```

```
<body>
<c:url var="searchURL" value="/forms/
    dentist/admin/viewSearchAudit.html"
    />

<div class="list-body-audit">
<div class="sub-body-title">View Audit
    Trail</div>
<form:form action="${searchURL}" method='
    GET' modelAttribute="auditTrail" id
    ="searchForm">

<div class="searchBox">
<table>
<tr>
<td>Search input :</td>
<td><form:input path="name" size="30"/>
    </td>

<td><input type="text" name="datefrom" id
    ="datefrom" size="10" onClick="
    showCalendar(this)" />
    <a href="javascript:NewCal('datefrom
        ','ddmmyyyy')">
                                    <img
                                        src
                                        ="
                                        $
                                        {
                                        pageContext
                                        .
                                        request
                                        .
                                        contextPath
                                        }/
                                        images
                                        /
                                        cal
                                        .
                                        gif
                                        "

                                        width
                                        ="16"

                                        height
                                        ="16"

                                        border
                                        ="0"
                                        alt
                                        ="
                                        Pick
                                        a
                                        date
                                        "></
                                        a
                                        >
                                    </td>
    - <input type="text" name="dateto"
        id="dateto" size="10" onClick
        ="showCalendar(this)" /><a href
        ="javascript:NewCal('dateto','
        ddmmyyyy')">
                                    <img
                                        src
                                        ="
                                        $
                                        {
                                        pageContext
                                        .
                                        request
                                        .
                                        contextPath
                                        }/
                                        images
                                        /
                                        cal
                                        .
                                        gif
                                        "

                                        width
                                        ="16"

                                        height
                                        ="16"

                                        border
                                        ="0"

                                        alt
                                        ="
                                        Pick
                                        a
                                        date
                                        "></
                                        a
                                        >
                                    </td>
    <td> (dd/MM/yyyy)</td>
    alt</tr>
    =" <tr>
    Pick<td></td>
    <td> <form:select path="action">
    a       <c:forEach var="action" items="${
```

```
            actions}">
        <c:if test="${action != selected
            }">
            <form:option value="${action
                }">${action}</form:
                option>
        </c:if>
        </c:forEach>
    </form:select>
</td>
<td>
<form:select path="action_form">
        <c:forEach var="dentistTable"
            items="${dentistTables}">
        <c:if test="${dentistTable !=
            selected}">
            <form:option value="${
                dentistTable}">${
                dentistTable}</form:
                option>
        </c:if>
        </c:forEach>
    </form:select>
</td>
<td>
    <input type="button" value="Search"
        id="searchSubmit" class="
        classname"/>
</td>
</tr>
</table>
</div>
</form:form>

<div class="clear"></div>
<hr>
<div class="clear"></div>

<table class="list-table" width="90%">
<thead>
<tr class="head-list-table">
<td width="30%">Name :</td>
<td width="10%"> Action :</td>
<td width="10%">Form :</td>
<td width="30%">Action Performed to :</td
    >
<td width="10%">Action Date :</td>
</tr>
</thead>
<tbody>
  <c:forEach items="${allAuditList}" var
      ="audit" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq
         0}'>
      <c:set var="rowColor" value="even"
          scope="page"/>
     </c:when>
     <c:otherwise>
       <c:set var="rowColor" value="odd"
          scope="page"/>
     </c:otherwise>
  </c:choose>


<tr class="${rowColor}">


 <td width="30%">
  ${audit.name}
    </td>
 <td width="10%">
          ${audit.action}
    </td>
     <td width="10%">
          ${audit.action_form}
    </td>
    <td width="30%">
          ${audit.action_performed}
    </td>

    <td width="10%">
          ${audit.action_date}
    </td>
 </tr>
  </c:forEach>
<c:if test="${empty allAuditList}">

      <tr>
      <td colspan="6" align="center">
          Currently No Audit</td>
      </tr>
</c:if>
</tbody>
</table>
</div>
</body>
</html>
```

Listing 46: changeEmailAdmin

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Users</title>
<script>
$(document).ready(function(){
 $('#emailSubmit').click(function(){
  emailValidate();
 });


});


function emailValidate(){

 //fields
 var smtp_from= document.getElementById('
     smtp_from').value;
 var smtp_subj= document.getElementById('
     smtp_subj').value;
 var smtp_message= document.
     getElementById('smtp_message').
     value;


 var emailConfirm="";

 //check for missing input
 if(smtp_from=="" || smtp_subj=="" ||
     smtp_message=="")
  emailConfirm+="Required information is
      incomplete. Please check if all
      information needed is filled.\n";

 if(emailConfirm==""){
   $('#emailform').submit();
 }
 else
  alert(emailConfirm);
}

</script>
</head>
<body>
<c:url var="emailAddress" value="/forms/
    dentist/admin/changeEmailAdmin.html"
     />
<c:url var="cancelURL" value="/forms/
    dentist/login/index.html" />

<div class="form-body">
<div class="sub-body-title">Change Email
    Address details</div>
<div class="clear"></div>

<c:if test="${not empty success}">
 <span style="color:green">${success}</
     span>
</c:if>
<div class="clear"></div>
<form:form action="${emailAddress}"
    commandName="emailAddress" id="
    emailform">
<table class="form-table" >

<tr>

<td>FROM :</td>

<td><form:input path="smtp_from" size
    ="50"/>
</td>
</tr>

<tr>

<td>SUBJECT :</td>

<td><form:input path="smtp_subj" size
    ="50"/></td>

</tr>


<tr>

<td>MESSAGE :</td>

<td><form:textarea path="smtp_message"
    rows="5" cols="30" /></td>

</tr>


<tr>

<td><input type="button" value="Change
    Email" id="emailSubmit" class="
    classname"/></td>
<td><input type="button" value="Cancel"
    onClick="window.location='${
    cancelURL}'" class="classname"/></td
```

```
    >
</tr>
</table>
</form:form>
```

Listing 47: changePassword

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Users</title>
<script>
$(document).ready(function(){

 $('#passwordSubmit').click(function(){
  passwordValids();
 });


});


function passwordValids(){

 //fields

 var password= document.getElementById('
     password').value;
 var repassword= document.getElementById
     ('repassword').value;


 var passwordConfirm="";

 //validate if password entered is
     correct
 if(password!=repassword)
  passwordConfirm+="Password didn't match
      . Re-enter password.\n";


 //check for missing input
 if(password=="" || repassword=="")
  passwordConfirm+="Required information
      is incomplete. Please check if all
       information needed is filled.\n";


 if(passwordConfirm==""){
  var r=confirm("Are you sure to change
      the password?");
  if (r==true)
   $('#passwordform').submit();
 }
 else
  alert(passwordConfirm);
}

</script>
</head>
<body>

<c:url var="passwordURL" value="/forms/
    dentist/login/changePassword" />
<c:url var="cancelURL" value="/forms/
    dentist/login/index.html" />


<div class="form-body">
<div class="sub-body-title">Change
    Password</div>
<h4><c:out value="${user.fname_user}" />
    <c:out value="${user.minit_user}" />
     <c:out value="${user.lname_user}"
    /></h4>
<h5>Role: ${sessionScope.sessionUserRole
    }</h5>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
     span>
</c:if>
<div class="clear"></div>
</div>
</div>

</body>
</html>
```

```
<div class="form-body">



<form:form action="${passwordURL}"
     commandName="user" id="passwordform
     ">
<form:hidden path="user_id" />
<form:hidden path="fname_user" />
<form:hidden path="minit_user" />
<form:hidden path="lname_user" />

<table class="form-table" >

<tr>

<td>Password : </td>

<td><form:password path="password" size
     ="30"/>
<span id="passwordValidate" style="
     display:none; color:red">Password
     must be atleast 6-8 characters with
     atleast 1 number</span></td>

</tr>

<tr>

<td>Confirm Password :</td>

<td><input type="password" id="repassword
     " size="30"/></td>

</tr>

<tr>

<td><input type="button" value="Change
     Password" id="passwordSubmit" class
     ="classname"/></td>
<td><input type="button" value="Cancel"
     onClick="window.location='${
     cancelURL}'" class="classname"/></td
     >
</tr>
</table>
</form:form>
</div>
</body>
</html>
```

Listing 48: changeSecret

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Users</title>
<script>
$(document).ready(function(){

 $('#secretSubmit').click(function(){
  secretValidate();
 });


});

function confirmValidation(){
 var r=confirm("Are you sure to change
      the password?");
 if (r==true)
  $('#deleteForm').submit();

}


function secretValidate(){

 //fields

 var secret_question= document.
      getElementById('secret_question').
      value;
 var secret_answer= document.
      getElementById('secret_answer').
      value;


 var secretConfirm="";
```

```
 //check for missing input
 if(secret_question=="" || secret_answer
      =="")
  secretConfirm+="Required information is
       incomplete. Please check if all
      information needed is filled.\n";

 if(secretConfirm==""){
  var r=confirm("Are you sure to change
       the secret question and answer?");
  if (r==true)
   $('#secretform').submit();
 }
 else
  alert(secretConfirm);
}

</script>
</head>
<body>

<c:url var="secretURL" value="/forms/
     dentist/login/changeSecret" />
<c:url var="cancelURL" value="/forms/
     dentist/login/index.html" />


<div class="form-body">
<div class="sub-body-title">Change Secret
      Question and Answer</div>
<h4><c:out value="${user.fname_user}" />
     <c:out value="${user.minit_user}" />
      <c:out value="${user.lname_user}"
     /></h4>
<h5>Role: ${sessionScope.sessionUserRole
     }</h5>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
      span>
</c:if>
<div class="clear"></div>
</div>


<div class="form-body">


<form:form action="${secretURL}"
     commandName="user" id="secretform">
<form:hidden path="user_id" />
<form:hidden path="fname_user" />
<form:hidden path="minit_user" />
<form:hidden path="lname_user" />

<table class="form-table" >
<tr>

<td>Secret Question : </td>

<td><form:select path="secret_question">
        <form:option value="${selected}"
             >Choose</form:option>
        <c:forEach var="secret_question"
             items="${secret_questions}">
        <c:if test="${secret_question !=
             selected}">
            <form:option value="${
                 secret_question}">${
                 secret_question}</form:
                 option>
        </c:if>
        </c:forEach>
    </form:select>
</td>
</tr>

<tr>

<td>Secret Answer :</td>

<td><form:input path="secret_answer" size
     ="30"/></td>

</tr>



<tr>

<td><input type="button" value="Change
     Answer" id="secretSubmit" class="
     classname"/></td>
<td><input type="button" value="Cancel"
     onClick="window.location='${
     cancelURL}'" class="classname"/></td
     >
</tr>

</table>
</form:form>
</div>
</body>
</html>
```

Listing 49: editRoleForm

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Roles</title>
<script>

$(document).ready(function(){
 $('#roleSubmit').click(function(){
  roleValidate();
 });

 $('#role_name').bind('change', function
     (){
  var role_name = $(this).val();
  var allRolenames= '${allRolenames}'.
      split(',');
  isRoleExist(allRolenames,role_name);

 });



});


function isRoleExist(allRolenames,
    role_name){
 var roleBool=true;
 for(i=0;i<allRolenames.length;i++){
  if(role_name==allRolenames[i]){
   $("#roleExist").show();
   roleBool=false;
  }
 }
 if(roleBool)
  $("#roleExist").hide();

}

function roleValidate(){

 //fields
 var role_name= document.getElementById('
     role_name').value;
 var role_description= document.
     getElementById('role_description').
     value;
 var database_role= document.
     getElementById('database_role').
     value;
 var roleConfirm="";

 //check for missing input
 if(role_name=="" || role_description==""
     || database_role=="")
  roleConfirm+="Required information is
      incomplete. Please check if all
      information needed is filled.\n";

 if(roleConfirm==""){
   $('#roleform').submit();
 }
 else
  alert(roleConfirm);
}



</script>




</head>
<body>

<c:url var="cancelURL" value="/forms/
    dentist/login/index.html" />
<div class="tab-body">

<c:url var="editRoleURL" value="/forms/
    dentist/simpleForm/editRoleForm?
    role_id=${role.role_id}" />
<c:url var="viewUserRoleURL" value="/
    forms/dentist/simpleForm/
    viewUserRole?role_id=${role.role_id
    }" />

<ul class = "tabs primary">
```

```
    <li><a href = "${viewUserRoleURL}">
         View Users</a></li>
    <li class = "active"><a href = "${
         editRoleURL}" class = "active">
         Edit Role</a></li>

</ul>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
      span>
</c:if>
<div class="clear"></div>
</div>
<c:url var="saveUrl" value="/forms/
     dentist/simpleForm/editRoleForm" />

<form:form action="${saveUrl}" id="
     roleform" modelAttribute="role"
     method="POST">
<form:hidden path="role_id" />

<table class="form-table" >

<tr>

<td>Role Name :</td>

<td><form:input path="role_name" size
     ="30"/>
<span id="roleExist" style="display:none;
      color:red">Role already exists!</
     span></td>

</tr>
```

```
<tr>
<td>Database Role Name :</td>
<td><form:input path="database_role" size
     ="50"/></td>
</tr>
<tr>

<tr>

<td>Role Description :</td>

<td><form:textarea path="role_description
     " rows="5" cols="50" /></td>

</tr>


<tr>
<td><input type="button" value="Edit Role
     " id="roleSubmit" class="classname
     "/></td>
<td><input type="button" value="Cancel"
     onClick="window.location='${
     cancelURL}'" class="classname"/>

</td>

</tr>

</table>

</form:form>

</div>
</body>
</html>
```

Listing 50: editSectionForm

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">

<html>

<head>
```

```
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Sections</title>
<script>

$(document).ready(function(){
 $('#sectionSubmit').click(function(){
  sectionValidate();
 });

 $('#section_name').bind('change',
      function(){
```

```
  var section_name = $(this).val();
  var allSectionnames= '${allSectionnames
       }'.split(',');
  isSectionExist(allSectionnames,
       section_name);

 });
 $('#roleButton').click(function(){
  if($(this).val()=="Add Roles"){
   $("#roleAppear").show();
   $("#roleButton").val('Cancel');
  }
  else if($(this).val()=="Cancel"){
   $("#roleAppear").hide();
   $("#roleButton").val('Add Roles');
  }
 });
 $('#addRoleSubmit').click(function(){
  $('#addroleform').submit();
 });

 $('#showUserList').click(function(){
  if($(this).val()=="View Users"){
   $("#showUserDiv").show();
   $("#showUserList").val('Cancel');
  }
  else if($(this).val()=="Cancel"){
   $("#showUserDiv").hide();
   $("#showUserList").val('View Users');
  }
 });

 $('#deleteSubmit').click(function(){
  deleteValidate();
 });


});



function deleteValidate(){
  var r=confirm("Are you sure to delete
       this role in this section?");
  if (r==true)
   $('#deleteForm').submit();

}



function isSectionExist(allSectionnames,
     section_name){
 var sectionBool=true;
 for(i=0;i<allSectionnames;i++){
  if(section_name==allSectionnames[i]){
   $("#sectionExist").show();
    sectionBool=false;
   }
  }
  if(roleBool)
   $("#sectionExist").hide();

}

function sectionValidate(){

 //fields
 var section_name= document.
      getElementById('section_name').
      value;
 var section_description= document.
      getElementById('section_description
      ').value;
 var sectionConfirm="";

 //check for missing input
 if(section_name=="" ||
      section_description=="")
  sectionConfirm+="Required information
       is incomplete. Please check if all
        information needed is filled.\n";

 if(sectionConfirm==""){
   $('#sectionform').submit();
 }
 else
  alert(sectionConfirm);
}


</script>



</head>

<body>

<c:url var="cancelURL" value="/forms/
     dentist/login/index.html" />
<div class="tab-body">

<c:url var="addRoleURL" value="/forms/
     dentist/simpleForm/addSectionRole?
     section_id=${section.section_id}" />
<c:url var="editSectionURL" value="/forms
     /dentist/simpleForm/editSectionForm?
     section_id=${section.section_id}" />

<ul class = "tabs primary">
```

```
    <li><a href = "${addRoleURL}">Apply
        Role</a></li>

    <li class = "active"><a href = "${
        editSectionURL}" class = "active
        ">Edit Section</a></li>

</ul>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
     span>
</c:if>
<div class="clear"></div>
</div>


<c:url var="saveUrl" value="/forms/
    dentist/simpleForm/editSectionForm"
    />

<form:form action="${saveUrl}" id="
    sectionform" modelAttribute="section
    " method="POST">
<form:hidden path="section_id" />

<table class="form-table" >

<tr>

<td>Section Name :</td>

<td><form:input path="section_name" />
<span id="sectionExist" style="display:
    none; color:red">Section already
    exists!</span></td>

</tr>

<tr>

<td>Section Description :</td>

<td><form:textarea path="
    section_description" rows="5" cols
    ="30" /></td>

</tr>


<tr>

<td><input type="submit" value="Edit
    Section" id="sectionSubmit" class="
    classname"/></td>

<td><input type="button" value="Cancel"
    onClick="window.location='${
    cancelURL}'" class="classname"/></td
    >

</tr>

</table>

</form:form>


</div>

</body>
</html>
```

Listing 51: editUserForm

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Users</title>
<script>

$(document).ready(function(){
 $('#userSubmit').click(function(){
  userValidate();
 });

 $('#addUserRole').click(function(){
  $("#userform").attr("action", "
```

```
        AddUserRole.html");
  $('#userform').submit();

 });

 $('#username').bind('change', function()
      {
  var username = $(this).val();
  var allUsername= '${allUsernames}'.
       split(',');
  //alert(username);
  isUsernameExist(allUsername,username);

 });



});


function isUsernameExist(allUsername,
     username){
 var userBool=true;
 for(i=0;i<allUsername.length;i++){
  if(username==allUsername[i]){
   $("#userExist").show();
   userBool=false;
  }
 }
 if(userBool)
  $("#userExist").hide();

}

function userValidate(){

 //fields
 var fname_user= document.getElementById
      ('fname_user').value;
 var minit_user= document.getElementById
      ('minit_user').value;
 var lname_user= document.getElementById
      ('lname_user').value;
 var username= document.getElementById('
      username').value;
 var password= document.getElementById('
      password').value;
 var repassword= document.getElementById
      ('repassword').value;
 var email= document.getElementById('
      email').value;
 var secret_answer= document.
      getElementById('secret_answer').
      value;

 var userConfirm="";
 //validate if password entered is
      correct
 if(password!=repassword)
  userConfirm+="Password didn't match. Re
       -enter password.\n";

 if(!isValidEmailAddress(email))
  userConfirm+="Email address is invalid.
       Re-enter email address.\n";

 //check for missing input
 if(fname_user=="" || minit_user=="" ||
      lname_user=="" || username=="" ||
      password=="" || secret_answer=="")
  userConfirm+="Required information is
       incomplete. Please check if all
       information needed is filled.\n";

 if(userConfirm==""){
  $('#userform').submit();
 }
 else
  alert(userConfirm);
}

//validate if input is really an emailAdd
function isValidEmailAddress(email) {
    var pattern = new RegExp(/^(("[\w-+\s
         ]+")|([\w-+]+(?:\.[\w-+]+)*)
         |("[\w-+\s]+")([\w-+]+(?:\.[\w
         -+]+)*))(@((?:[\w-+]+\.)*\w[\w
         -+]{0,66})\.([a-z]{2,6}(?:\.[a-z
         ]{2})?)$)|(@
         \[?((25[0-5]\.|2[0-4][\d]\.|1[\d
         ]{2}\.|[\d]{1,2}\.))
         ((25[0-5]|2[0-4][\d]|1[\d]{2}|[\
         d]{1,2})\.){2}(25[0-5]|2[0-4][\d
         ]|1[\d]{2}|[\d]{1,2})\]?$)/i);
    return pattern.test(email);
}


</script>




</head>

<body>
<c:url var="cancelURL" value="/forms/
     dentist/login/index.html" />
<div class="tab-body">
```

```
<c:url var="addUserURL" value="/forms/
     dentist/simpleForm/addUserRole?
     user_id=${user.user_id}" />
<c:url var="editUserURL" value="/forms/
     dentist/simpleForm/editUserForm?
     user_id=${user.user_id}" />

<ul class = "tabs primary">

    <li><a href = "${addUserURL}">Apply
         Role</a></li>

    <li class = "active"><a href = "${
         editUserURL}" class = "active">
         Edit User</a></li>

</ul>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
      span>
</c:if>
<div class="clear"></div>
</div>
<c:url var="saveUrl" value="/forms/
     dentist/simpleForm/editUserForm" />

<form:form action="${saveUrl}" id="
     userform" modelAttribute="user"
     method="POST">
<form:hidden path="user_id" />
<table class="form-table" >

<tr>

<td>Firstname :</td>
<td><form:input path="fname_user" size
     ="30"/></td>

</tr>


<tr>

<td>MI :</td>

<td><form:input path="minit_user" size
     ="10"/></td>

</tr>

<tr>

<td>Lastname :</td>

<td><form:input path="lname_user" size
     ="30"/></td>

</tr>

<tr>

<td>Username :</td>

<td><form:input path="username" size
     ="30"/>
<span id="userExist" style="display:none;
      color:red">Username already exists
     !</span></td>

</tr>

<tr>

<td>Password : </td>

<td><form:password path="password" size
     ="30"/>
<span id="passwordValidate" style="
     display:none; color:red">Password
     must be atleast 6-8 characters with
     atleast 1 number</span></td>

</tr>

<tr>

<td>Confirm Password :</td>

<td><input type="password" id="repassword
     " size="30"/></td>


</tr>

<tr>

<td>Email :</td>

<td><form:input path="email" size="40"
     /></td>

</tr>

<tr>

<td>Secret Question : </td>

<td><form:select path="secret_question">
        <form:option value="${selected}"
             >Choose</form:option>
        <c:forEach var="secret_question"
             items="${secret_questions}">
```

```
        <c:if test="${secret_question !=
             selected}">
            <form:option value="${
                secret_question}">${
                secret_question}</form:
                option>
        </c:if>
        </c:forEach>
    </form:select>
</td>
</tr>

<tr>

<td>Secret Answer :</td>

<td><form:input path="secret_answer" size
    ="30"/></td>
</tr>

<tr>

<td><input type="button" value="Edit User
    " id="userSubmit" class="classname
    "/></td>
<td><input type="button" value="Cancel"
    onClick="window.location='${
    cancelURL}'" class="classname"/></td
    >
</tr>

</table>

</form:form>
</div>
</body>
</html>
```

Listing 52: forgotPasswordUN

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@taglib uri="http://java.sun.com/
        jsp/jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<script>
$(document).ready(function(){
 $('#usernameSubmit').click(function(){

  var username = $("#checkUN").val();
  $('#passwordForm').submit();

 });

});

</script>
</head>

<body>
<c:url var="passwordURL" value="/forms/
    dentist/login/forgotPasswordUN.html"
     />

<form:form action="${passwordURL}" method
    ='GET' modelAttribute="user" id="
    passwordForm">
<table class="password-table">
<tr class="sub-menu-title">
<td colspan="2">Forgout your Password?</
    td>
</tr>
<tr>
<td colspan="2"><c:if test="${not empty
    errorMsg}">
 <span style="color:red">${errorMsg}</
     span>
</c:if></td>
</tr>
<tr>
<td>Username:</td>
<td><form:input path="username" /></td>
<td><input type="button" id="
    usernameSubmit" value="NEXT" class="
    classname"/></td>
</tr>
</table>
</form:form>
<c:if test="${empty errorMsg}">
<c:choose>
    <c:when test="${not empty applicable
        }">
```

```
<c:url var="retrievalURL" value="/forms/
    dentist/login/forgotPasswordRetrieve
    .html" />

<form:form action="${retrievalURL}" id="
    userform" modelAttribute="user"
    method="POST">
  <form:hidden path="user_id" />
    <form:hidden path="username" />
    <form:hidden path="email" />


  <table class="password-table">
  <tr class="sub-menu-title">
   <td colspan="2">Answer the question
       to reset password.</td>
  </tr>
  <tr>
   <td>Secret Question</td>
   <td><c:out value="${user.
       secret_question}" /></td>
   <form:hidden path="secret_question"
       />
  </tr>
  <tr>
   <td>Secret Answer</td>
   <td><form:input path="secret_answer"
       size="30"/></td>
  </tr>
  <tr>
  <td><input type="submit" id="
      passwordSubmit" value="Submit"
      class="classname"/></td>
  </tr>
  </table>

</form:form>
  </c:when>
  <c:otherwise>

  </c:otherwise>
</c:choose>
</c:if>
</body>
</html>
```

Listing 53: header

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@ taglib prefix="fn" uri="http://java.
    sun.com/jsp/jstl/functions" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Insert title here</title>
      <link rel="stylesheet" type="text
          /css" href="${pageContext.
          request.contextPath}/theme.
          css" />
      <script src="//ajax.googleapis.
          com/ajax/libs/jquery/1.8.3/
          jquery.min.js" >
      </script>
 <script type="text/javascript">
      $(document).ready(function () {
          $('#menu li').hover(
          function () {
              //show its submenu
              $('ul', this).slideDown
                  (100);
          },
          function () {
              //hide its submenu
              $('ul', this).slideUp
                  (100);
          }
      );
      });

  </script>
</head>
<body>
<c:url var="homeURL" value="/forms/
    dentist/login/index.html" />
<c:url var="userURL" value="/forms/
    dentist/simpleForm/userForm.html" />
<c:url var="roleURL" value="/forms/
    dentist/simpleForm/roleForm.html" />
<c:url var="sectionURL" value="/forms/
    dentist/simpleForm/sectionForm.html"
    />
<c:url var="loginURL" value="/forms/
    dentist/login/loginForm.html" />
<c:url var="logoutURL" value="/forms/
    dentist/login/logout.html" />
<c:url var="profileURL" value="/forms/
    dentist/login/myProfile.html?user_id
    =${sessionScope.sessionUserId}" />
```

```
<c:url var="configurationURL" value="/
    forms/dentist/admin/changeEmailAdmin
    .html" />

<c:url var="dentalURL" value="/forms/
    dentist/simpleForm/viewdentalchart.
    html?patient_id=2" />
<c:url var="viewdentalURL" value="/forms/
    dentist/simpleForm/viewdentalchart.
    html?patient_id=1" />

<c:url var="viewQueryURL" value="/forms/
    dentist/search/viewQueryPatients.
    html" />
<c:url var="viewStatURL" value="/forms/
    dentist/search/statisticsform.html"
    />



<c:url var="addressURL" value="/forms/
    dentist/admin/changeEmailAdmin.html"
     />
<c:url var="auditURL" value="/forms/
    dentist/admin/auditTrailView.html"
    />



<div id="header">
<div id="logo-box">
<div id="logo-img-box">
<img src="${pageContext.request.
    contextPath}/images/upcd.png" alt="
    anotherimage" width="60" height="74"
     />
</div>
<div id="logo-text"><h1>DentISt</h1></div
    >
</div>
<div class="menu-header" >


<ul id="menu">
<li><a></a></li>
<c:if test="${not empty sessionScope.
    sessionUser}">

 <li><a href="${homeURL}" class="menu-
     divider">Home</a>
 </li>
 <c:if test="${fn:containsIgnoreCase(fn:
     join(sessionScope.currentRoleList
     ,',',),'system')}">
 <li><a href="#">Admin</a>
  <ul>
   <li><a href="${userURL}">User</a></li>
   <li><a href="${roleURL}">Role</a></li>
   <li><a href="${sectionURL}">Section</a
       ></li>
   <li><a href="${addressURL}">Email
       Settings</a></li>
   <li><a href="${auditURL}">Audit Trail
       </a></li>


  </ul>
</li>
<li><a href="#">Added</a>
<ul>
   <li><a href="${dentalURL}">Dental
       Chart</a></li>
   <li><a href="${viewdentalURL}">View DC
       </a></li>



  </ul>
</li>
</c:if>
   <c:if test="${fn:containsIgnoreCase(fn
       :join(sessionScope.
       currentRoleList,',','workflow')
       }">

<li><a href="#">Workflow</a>
 <ul>
   <li><a href="${workflowDesigner}"
       class="menu-divider">Designer</a
       ></li>
 </ul>
</li>
</c:if>
<c:if test="${fn:containsIgnoreCase(fn:
    join(sessionScope.currentRoleList
    ,',',),'faculty') || fn:
    containsIgnoreCase(fn:join(
    sessionScope.currentRoleList,',',),'
    student')}">

<li><a href="#">Patient</a>
 <ul>
   <li><a href="${ODSection}" class="menu
       -divider">New OD</a></li>
 </ul>
</li>
</c:if>
 <li><a href="#">Added</a>
<ul>
   <li><a href="${viewQueryURL}">Query
       Patients</a></li>
   <li><a href="${viewStatURL}">
       Statistics</a></li>



  </ul>
```

```
 </li>
 </c:if>

</ul>

</div>
<div id="login-bar">
<span>
<c:choose>
    <c:when test="${empty sessionScope.
        sessionUser}">
   Not logged in | <a href="${loginURL}">
        Log in</a>
    </c:when>
    <c:otherwise>
   Currently logged in as ${sessionScope.
        sessionUser} | <a href="${
        logoutURL}">Log out</a>
   | <a href="${profileURL}">My Profile</
        a>
 </c:otherwise>
</c:choose>
</span>
</div>
</div>
</body>
</html>
```

Listing 54: index

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@ taglib prefix="fn" uri="http://java.
    sun.com/jsp/jstl/functions" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">

<html>

<head>

<link rel="stylesheet" type="text/css"
    href="${pageContext.request.
    contextPath}/css/theme.css" />



<script type="text/javascript">
function noBack(){window.history.forward
    ();}
noBack();
window.onload=noBack;
window.onpageshow=function(evt){if(evt.
    persisted)noBack();}
window.onunload=function(){void(0);}
</script>

<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<title>DentISt 3.0</title>

</head>

<body>

<div class="login-main">
<img src="${pageContext.request.
    contextPath}/images/upcd-main.png"
    alt="upcd-main" />
<div class="clear"></div>
<br>
<span>Welcome to DentISt, ${sessionScope.
    sessionUser}</span>
<br/>
 <c:forEach var="privilege" items="${
     privilege}">
   <div class="privilegeDiv">
        <a href="${privilege.url}"> ${
            privilege.value}</a>
        </div>
         <br/>
       </c:forEach>


</div>

</body>

</html>
```

Listing 55: layout

```
<%@ taglib uri="http://tiles.apache.org/
    tags-tiles" prefix="tiles"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=UTF-8">
<title><tiles:insertAttribute name="title
```

```
    " ignore="true" /></title>
    </head>
<body>
<div class="layout">


        <div id="layout-header"><tiles:
            insertAttribute name="header
            " />
        </div>

        <div id="layout-menu"><tiles:
            insertAttribute name="menu"
            /></div>
        <div id="layout-body"><tiles:
            insertAttribute name="body
            "/></div>

        <div><tiles:insertAttribute name
            ="footer" /></div>

</div>
</body>
</html>
```

Listing 56: loginForm

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Login</title>

<link rel="stylesheet" type="text/css"
    href="${pageContext.request.
    contextPath}/css/theme.css" />


</head>
<body>
<c:if test="${not empty sessionScope.
    sessionUser}">
<c:redirect url="/forms/dentist/login/
    index.html" />
</c:if>

<div class="login-main">
<div class="login-img">
<img src="${pageContext.request.
    contextPath}/images/upcd-main.png"
    alt="upcd-main" />
</div>
<div class="clear"></div>
<br>
<span class="login-welcome">Welcome to
    DentISt. Please login to proceed.</
    span>
</div>

<div class="login-form">
<c:if test="${not empty error}">
 <span style="color:red">${error}</span>
</c:if>
<c:url var="outputURL" value="/forms/
    dentist/login/loginOutput.html" />

<form:form action="${outputURL}"
    commandName="login">
<table class="login-table">
<tr>
<td>Username :</td>
<td><form:input path="username" size
    ="30"/></td>
</tr>
<tr>
<td>Password :</td>
<td><form:password path="password" size
    ="30"/>
</td>
</tr>

<tr>

<td></td>

<td><input type="submit" value="Log in"
    class="classname"/></td>

</tr>
<tr>
<td></td>
<c:url var="passwordURL" value="/forms/
    dentist/login/forgotPassword.html"
    />

<td><a href="${passwordURL}">I forgot my
    password</a></td>
</tr>
</table>
</form:form>
</div>
</body>
</html>
```

Listing 57: myProfile

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Users</title>
<script>

$(document).ready(function(){
 $('#userSubmit').click(function(){
  userValidate();
 });




});


function userValidate(){

 //fields

 var email= document.getElementById('
     email').value;

 var userConfirm="";

 if(!isValidEmailAddress(email))
  userConfirm+="Email address is invalid.
      Re-enter email address.\n";

 //check for missing input
 if(email=="")
  userConfirm+="Required information is
      incomplete. Please check if all
      information needed is filled.\n";

 if(userConfirm==""){
  $('#userform').submit();
 }
 else
  alert(userConfirm);
}

//validate if input is really an emailAdd
function isValidEmailAddress(email) {
    var pattern = new RegExp(/^(("[\w-+\s
        ]+")|([\w-+]+(?:\.[\w-+]+)*)
        |("[\w-+\s]+")([\w-+]+(?:\.[\w
        -+]+)*))(@((?:[\w-+]+\.)*\w[\w
        -+]{0,66})\.([a-z]{2,6}(?:\.[a-z
        ]{2})?)$)|(@
        \[?((25[0-5]\.|2[0-4][\d]\.|1[\d
        ]{2}\.|[\d]{1,2}\.))
        ((25[0-5]|2[0-4][\d]|1[\d]{2}|[\
        d]{1,2})\.){2}(25[0-5]|2[0-4][\d
        ]|1[\d]{2}|[\d]{1,2})\]?$)/i);
    return pattern.test(email);
}

</script>




</head>

<body>
<c:url var="cancelURL" value="/forms/
    dentist/login/index.html" />
<div class="form-body">
<div class="sub-body-title">Profile
    General Information</div>
<h4><c:out value="${user.fname_user}" />
    <c:out value="${user.minit_user}" />
     <c:out value="${user.lname_user}"
    /></h4>
<h5>Role:   <c:forEach items="${
    sessionScope.currentRoleList}" var="
    currentRole" varStatus="loop">
  ${currentRole},

</c:forEach></h5>
<c:if test="${not empty success}">
 <span style="color:green">${success}</
     span>
</c:if>
<div class="clear"></div>
</div>
```

```
<div class="form-body">


<c:url var="saveUrl" value="/forms/
    dentist/login/myProfile" />

<form:form action="${saveUrl}" id="
    userform" modelAttribute="user"
    method="POST">
<form:hidden path="user_id" />
<form:hidden path="fname_user" />
<form:hidden path="lname_user" />
<form:hidden path="minit_user" />
<form:hidden path="username" />
<form:hidden path="password" />
<form:hidden path="secret_question" />
<form:hidden path="secret_answer" />

<table class="form-table" >

<tr>

<td>Firstname :</td>
<td><c:out value="${user.fname_user}"
    /></td>

</tr>


<tr>

<td>MI :</td>

<td><c:out value="${user.minit_user}"
    /></td>

</tr>

<tr>

<td>Lastname :</td>

<td><c:out value="${user.lname_user}"
    /></td>

</tr>

<tr>

<td>Username :</td>

<td><c:out value="${user.username}" />
<span id="userExist" style="display:none;
     color:red">Username already exists
    !</span></td>

</tr>


<tr>

<td>Email :</td>

<td><form:input path="email" size="40"
    /></td>

</tr>


<tr>

<td><input type="button" value="Update
    Profile" id="userSubmit" class="
    classname"/></td>
<td><input type="button" value="Cancel"
    onClick="window.location='${
    cancelURL}'" class="classname"/></td
    >
</tr>

</table>

</form:form>
</div>
</body>
</html>
```

Listing 58: queryPatients

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript" src="<c:
    url value="/showresult.js" />" >
</script>
</head>
<body>
<c:url var="homeURL" value="/forms/
```

```
      dentist/login/index.html" />
<c:url var="searchURL" value="/forms/
      dentist/searchResult/viewResults.
      html" />

<div class="searchDiv">

<h2>Query Patients</h2>

<form name="searchform" action="${
      searchURL}" method="POST" onSubmit="
      return checkForm2()">
 <b>
  <table class="boxHeader">
 <tr><td align="left" width="80%">
 Demographics
 </td></tr>
 </table>
 </b>
 <div class="box">
  <table>
   <tr>
    <td>Age</td>
    <td><input type="text" name="agefrom"
          id="agefrom">- <input type="
          text" name="ageto" id="ageto"></
          td>
   </tr>
   <tr>
    <td>Sex</td>
    <td>
     <input type="radio" name="sex" id="
          sex" value="male"/>M
     <input type="radio" name="sex" id="
          sex" value="female"/>F
     <input type="radio" name="sex" id="
          sex" value="all"/>Both

    </td>
   </tr>
   <tr>
    <td>City</td>
    <td><input type="text" name="city" id
         ="city"></td>
   </tr>
   <tr>
    <td>Occupation</td>
    <td><input type="text" name="
          occupation" id="occupation"></td
          >
   </tr>
  </table>
 </div>
 <br/>
 <br/>
<b >
 <table class="boxHeader">
 <col width="50%"> <col width="50%">
 <tr><td>Other Conditions (Using
      Demographics)</td>
 <td align="right"><input type="checkbox
      " id="or3" name="condd" value="or3
      " onClick="checkOrAnd3('or')"
      checked>Or |
 <input type="checkbox" id="and3" name="
      condd" value=and3 onClick="
      checkOrAnd3('and')" >And
 </td></tr>
 </table>
</b>
<br/>

<b >
 <table class="boxHeader">
 <col width="20%"> <col width="80%">
 <tr><td>Sections</td>
 <td align="right">
 <input type="checkbox" id="selectall3"
      onclick="selectAll3();">Select All
       </td></tr>
 </table>
</b>
<div class="box">
 <table><col width="250"><col width
      ="250"><col width="250">
  <tr>
   <td><b>Operative Dentistry</b></td>
   <td><b>Oral Medicine </b></td>
   <td><b>Prosthodontics</b></td>
  </tr>
  <tr>
   <td><input type="checkbox" id="s1"
        name="perio" value="PERIO" />
        Periodontics</td>
   <td><input type="checkbox" id="s2"
        name="rpd" value="RPD" />
        Removable Prosthodontics</td>
   <td><input type="checkbox" id="s3"
        name="ortho" value="Ortho" />
        Orthodontics</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="s4"
        name="os" value="OS" />Oral
        Surgery</td>
   <td><input type="checkbox" id="s5"
        name="fpd" value="FPD" />Fixed
        Partial Prosthodontics</td>
   <td><input type="checkbox" id="s6"
        name="pedo" value="PEDO" />
        Pedodontics</td>
```

```
  </tr>
  <tr>
   <td><input type="checkbox" id="s7"
        name="endo" value="ENDO" />
        Endodontics</td>
   <td><input type="checkbox" id="s8"
        name="cd" value="CD" />Complete
        Denture</td>
   <td><input type="checkbox" id="s9"
        name="resto" value="Resto" />
        Restorative Dentistry</td>
  </tr>
 </table>
 <br/>
</div>
<br/>
<br/>

<b >
 <table class="boxHeader">
 <col width="20%"> <col width="80%">
 <tr><td>Dental Condition</td>
 <td align="right"><input type="checkbox
      " id="or1" name="or1" value="yes"
      checked>Or |
 <input type="checkbox" id="and1" name="
      and1" value="yes">And |
 <input type="checkbox" id="selectall1"
      onclick="selectAll1();">Select All
       </td></tr>
 </table>
</b>
<div class="box">
 <table><col width="250"><col width
      ="250"><col width="250"><col width
      ="250"><col width="250">
  <tr>
   <td><input type="checkbox" id="dc1"
        name="caries" value="yes">
        Caries</td>
   <td><input type="checkbox" id="dc2"
        name="extrusion" value="yes">
        Extrusion</td>
   <td><input type="checkbox" id="dc3"
        name="completedenture" value="
        yes"> Complete Denture</td>
   <td><input type="checkbox" id="dc4"
        name="impacted" value="yes">
        Impacted</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="dc6"
        name="recurrentcaries" value="
        yes"> Recurrent Caries</td>
   <td><input type="checkbox" id="dc7"
        name="intrusion" value="yes">
        Intrusion</td>
  <td><input type="checkbox" id="dc8"
        name="singledenture" value="yes
        "> Single Denture</td>
  <td><input type="checkbox" id="dc22"
        name="missing" value="yes">
        Missing</td>
 </tr>
 <tr>
  <td><input type="checkbox" id="dc6"
        name="restoration" value="yes">
        Restoration</td>
  <td><input type="checkbox" id="dc12"
        name="mesialdrift" value="yes">
        Mesial Drifting Rotation</td>
  <td><input type="checkbox" id="dc13"
        name="removablepartial" value="
        yes"> Removable Partial Denture
        </td>
  <td><input type="checkbox" id="dc14"
        name="acryliccrown" value="yes
        "> Acrylic Crown</td>

 </tr>
 <tr>
  <td><input type="checkbox" id="dc5"
        name="porcelainfused" value="yes
        "> Porcelain Fused To Metal</td>
  <td><input type="checkbox" id="dc17"
        name="distaldrift" value="yes">
        Distal Drifting Rotation</td>
  <td><input type="checkbox" id="dc18"
        name="pitandfissure" value="yes
        "> Pit and Fissure Sealants</td>
  <td><input type="checkbox" id="dc19"
        name="metalcrown" value="yes">
        Metal Crown</td>
 </tr>
 <tr>
  <td><input type="checkbox" id="dc21"
        name="rotation" value="yes">
        Rotation</td>
  <td><input type="checkbox" id="dc26"
        name="rootcanal" value="yes">
        Root Canal Treatment</td>
  <td><input type="checkbox" id="dc9"
        name="postcorecrown" value="yes
        "> Post Core Crown</td>
 </tr>
 <tr>
  <td><input type="checkbox" id="dc25"
        name="extracted" value="yes">
        Extracted</td>
  <td><input type="checkbox" id="dc27"
        name="unerupted" value="yes">
        Unerupted</td>
```

```
   <td><input type="checkbox" id="dc23"
        name="porcelaincrown" value="
       yes"> Porcelain Crown</td>
  </tr>
 </table>
 <br/>
</div>
<br/>
<br/>


<b >
 <table class="boxHeader">
 <col width="20%"> <col width="80%">
 <tr><td>Services Needed</td>
 <td align="right"><input type="checkbox
      " id="or2" name="or2" value="yes"
      checked>Or |
 <input type="checkbox" id="and2" name="
      and2" value="yes">And |
 <input type="checkbox" id="selectall2"
      onclick="selectAll2();">Select All
       </td></tr>
 </table>
</b>
<div class="box">
 <table><col width="250"><col width
      ="250"><col width="300">
  <tr>
   <td><u>Periodontics</u></td>
  </tr>
  <tr>
   <td colspan="3"><input type="checkbox
       " id="sn1" name="periodontics"
       value="yes"> Management of
       Periodontal Disease</td>
  </tr>

 <!--_ROW_2_-->

  <tr>
   <td><br/><u>Operative Dentistry </u
       ></td>
   <td><br/><u> Surgery </u></td>
   <td><br/><u>Emergency Treatment</u></
       td>
  </tr>
  <tr>
   <td><input type="checkbox" id="sn2"
       name="class1" value="yes"> Class
        I</td>
   <td><input type="checkbox" id="sn3"
       name="extraction" value="yes">
       Extraction</td>
   <td><input type="checkbox" id="sn4"
       name="pulpsedation" value="yes">
        Pulp Sedation</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="sn5"
       name="class2" value="yes"> Class
        II</td>
   <td><input type="checkbox" id="sn6"
       name="odontectomy" value="yes">
       Odontectomy</td>
   <td><input type="checkbox" id="sn7"
       name="crownrecementation" value
       ="yes"> Recementation of Crowns
       </td>
  </tr>
  <tr>
   <td><input type="checkbox" id="sn8"
       name="class3" value="yes"> Class
        III</td>
   <td><input type="checkbox" id="sn9"
       name="specialcase" value="yes">
       Special Case</td>
   <td><input type="checkbox" id="sn10"
       name="tempfillingservice" value
       ="yes"> Temporary Fillings</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="sn11"
       name="class4" value="yes"> Class
        IV</td>
   <td><input type="checkbox" id="sn12"
       name="pedodontics" value="yes">
       Pedodontics </td>
   <td><input type="checkbox" id="sn13"
       name="acuteinfections" value="
       yes"> Management of acute
       infections</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="sn14"
       name="class5" value="yes"> Class
        V</td>
   <td><input type="checkbox" id="sn15"
       name="orthodontics" value="yes">
        Orthodontics</td>
   <td><input type="checkbox" id="sn16"
       name="traumaticinjuries" value="
       yes"> Management of Temporary
       Injuries</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="sn17"
       name="onlay" value="yes"> Onlay
       </td>
  </tr>

 <!--_ROW_3_-->
```

```
    <tr>
     <td><br/><u>Fixed Partial Denture</u
          ></td>
     <td><br/><u>Prosthodontics</u><br/>
     <td><br/><u>Endodontics</u></td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn18"
          name="laminated" value="yes">
          Laminated</td>
     <td><input type="checkbox" id="sn19"
          name="completedentservice" value
          ="yes"> Complete Denture</td>
     <td><input type="checkbox" id="sn20"
          name="anterior" value="yes">
          Anterior</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn21"
          name="singlecrown" value="yes">
          Single Crown</td>
     <td><input type="checkbox" id="sn22"
          name="singledentservice" value="
          yes"> Single Denture</td>
     <td><input type="checkbox" id="sn23"
          name="posterior" value="yes">
          Posterior</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn24"
          name="bridge" value="yes">
          Bridge</td>
     <td><input type="checkbox" id="sn25"
          name="removablepartialservice"
          value="yes"> Removable Partial
          Denture</td>
    </tr>
   </table>
   <br/>
  </div>

  <br/>
  <input type="submit" value="Search"
       class="classname">
  <input type="button" name="clear" value
       ="Clear Form" onclick="clearForm()
       ;" class="classname"/>
  <a href="${homeURL}"> Go Back </a> <br/>
       <br/>
 </form>
 </div>
 </body>
 </html>
```

Listing 59: roleForm

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Roles</title>

<script>

$(document).ready(function(){
 $('#roleSubmit').click(function(){
  roleValidate();
 });

 $('#role_name').bind('change', function
      (){
  var role_name = $(this).val();
  var allRolenames= '${allRolenames}'.
       split(',');
  //alert(role_name);
  isRoleExist(allRolenames,role_name);

 });



});



function isRoleExist(allRolenames,
     role_name){
 var roleBool=true;
 for(i=0;i<allRolenames.length;i++){
```

```
  if(role_name==allRolenames[i]){
   $("#roleExist").show();
   roleBool=false;
  }
 }
 if(roleBool)
  $("#roleExist").hide();

}

function roleValidate(){

 //fields
 var role_name= document.getElementById('
      role_name').value;
 var role_description= document.
      getElementById('role_description').
      value;
 var database_role= document.
      getElementById('database_role').
      value;
 var roleConfirm="";

 //check for missing input
 if(role_name=="" || role_description==""
       || database_role=="")
  roleConfirm+="Required information is
       incomplete. Please check if all
       information needed is filled.\n";

 if(roleConfirm==""){
   $('#roleform').submit();
 }
 else
  alert(roleConfirm);
}



</script>


</head>

<body>
<c:url var="cancelURL" value="/forms/
     dentist/login/index.html" />

<div class="form-body">
```

```
<div class="sub-body-title">Add Role</div
     >

<form:form action="roleoutput.html"
     commandName="role" id="roleform">

<table class="form-table">

<tr>

<td>Role Name :</td>

<td><form:input path="role_name" size
     ="50"/>
<span id="roleExist" style="display:none;
      color:red">Role already exists!</
     span></td>

</tr>
<tr>
<td>Database Role Name :</td>
<td><form:input path="database_role" size
     ="50"/></td>
</tr>
<tr>

<td>Role Description :</td>

<td><form:textarea path="role_description
     " rows="5" cols="30" /></td>

</tr>

</table>
<div class="submit-buttons">
<input type="button" value="Submit" id="
     roleSubmit" class="classname"/>
<input type="button" value="Cancel"
     onClick="window.location='${
     cancelURL}'" class="classname"/>

</div>
</form:form>
</div>

</body>

</html>
```

Listing 60: searchResults

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@taglib uri="http://java.sun.com/
         jsp/jstl/core" prefix="c" %>
    <%@ taglib prefix="fn" uri="http://
         java.sun.com/jsp/jstl/functions"
          %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
```

```
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript" src="<c:
    url value="/showresult.js" />" >
</script>



</head>
<body>
<div class="searchDiv">
<c:url var="viewQueryURL" value="/forms/
    dentist/search/viewQueryPatients.
    html" />

<a href="${viewQueryURL}"> Go Back </a>

<h2>SearchResults</h2>

<c:set var="condd" value="${condd}"/>

<c:set var="agefrom" value="${agefrom}"/>
<c:set var="ageto" value="${ageto}"/>
<c:set var="agecheck" value="${agecheck
    }"/>
<c:set var="sexcheck" value="${sexcheck
    }"/>
<c:set var="sex" value="${sex}"/>
<c:set var="city" value="${city}"/>
<c:set var="citycheck" value="${citycheck
    }"/>
<c:set var="occupation" value="${
    occupation}"/>
<c:set var="occupationcheck" value="${
    occupationcheck}"/>
<c:set var="caries" value="${caries}"/>
<c:set var="recurrentcaries" value="${
    recurrentcaries}"/>

<c:set var="periodontics" value="${
    periodontics}"/>
<c:set var="class1" value="${class1}"/>
<c:set var="class2" value="${class2}"/>
<c:set var="class3" value="${class3}"/>
<c:set var="class4" value="${class4}"/>
<c:set var="class5" value="${class5}"/>
<c:set var="onlay" value="${onlay}"/>

<c:set var="extraction" value="${
    extraction}"/>
<c:set var="odontectomy" value="${
    odontectomy}"/>
<c:set var="specialcase" value="${
    specialcase}"/>
<c:set var="pedodontics" value="${
    pedodontics}"/>
<c:set var="orthodontics" value="${
    orthodontics}"/>

<c:set var="pulpsedation" value="${
    pulpsedation}"/>
<c:set var="crownrecementation" value="${
    crownrecementation}"/>
<c:set var="tempfillingservice" value="${
    tempfillingservice}"/>
<c:set var="acuteinfections" value="${
    acuteinfections}"/>
<c:set var="traumaticinjuries" value="${
    traumaticinjuries}"/>

<c:set var="laminated" value="${laminated
    }"/>
<c:set var="singlecrown" value="${
    singlecrown}"/>
<c:set var="bridgeservice" value="${
    bridge}"/>

<c:set var="anterior" value="${anterior
    }"/>
<c:set var="posterior" value="${posterior
    }"/>

<c:set var="completedentservice" value="$
    {completedentservice}"/>
<c:set var="singledentservice" value="${
    singledentservice}"/>
<c:set var="removablepartialservice"
    value="${removablepartialservice}"/>

<br/>



<c:set var="agecount" value="0"/>
<br/><br/>
<c:if test="${(occupation!='' || city!=''
     || ageto!='' || agefrom!=''|| not
    empty sex) }">

<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
 Patients
 </td></tr>
 </table>
 </b>
<div class="box">
```

```
<c:if test="${agefrom != '' || ageto !=
    ''}">
 <b>| Age: ${agefrom} - ${ageto} </b>
</c:if>
<c:if test="${sex != ''}">
 <b>| Gender: ${sex} </b>
</c:if>
<c:if test="${city != ''}">
 <b>| City: '${city}' </b>
</c:if>
<c:if test="${occupation != ''}">
 <b>| Occupation: '${occupation}' </b>
</c:if>
<br/>
<br/>
<table id = "sample" cellspacing="2">
 <col width="25%"><col width="50%"><col
     width="25%">
 <tr> <td><b> Id </b></td> <td><center><
     b> Name </b> </center></td> <td><
     center><b> Age </b> </center></td>
      </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="item" items="${
     agegrouppatients}" varStatus="
     status">
  <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter % 3 ==  1}">
   <tr><td>
   <a href="">${item}</a></td>
   <c:set var="agecount" value="${
       agecount + 1}"/>
  </c:if>
  <c:if test="${counter % 3 == 2 ||
      counter % 3 == 0}">
   <td align="center"> ${item} </td>
   <c:if test="${counter % 3 == 0}">
    <td></td></tr>
   </c:if>
  </c:if>
 </c:forEach>
 <c:if test="${agecount != 0 }">
  <tr> <td colspan="2"></td><td align="
      center" colspan="2"><font size =
      "1">${agecount} patient(s) found.
       </font></td> </tr>
 </c:if>
 <c:if test="${agecount == 0 }">
  <tr> <td colspan="3"></td><td align="
      left"><font size = "1">No results
       found.</font></td> </tr>
 </c:if>
</table>
</div>
<br/>
<br/>
</c:if>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
 Other Conditions using Demographic
 <c:if test="${condd == 'or3' }">(OR)</c:
     if>
 <c:if test="${condd == 'and3' }">(AND)</
     c:if>
 </td></tr>
 </table>
 </b>

<c:if test="${perio == 'PERIO' || os == '
    OS' || endo == 'ENDO' || rpd == 'RPD
    ' || fpd == 'FPD' ||
 cd == 'CD' || ortho == 'Ortho' || pedo
     == 'PEDO' || resto == 'Resto'}">
<br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
 Patients
 </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
 <col width="150"><col width="50"><col
     width="250">
 <tr> <td> </td> <td><b> Id </b></td> <td
     ><center><b> Name </b> </center></
     td> <td></td></tr>
  <c:if test="${perio == 'PERIO'}">
   <c:set var="periocount" value="0"/>
   <tr> <td colspan="1">  <b>
       Periodontics </b> </td> </tr>
   <c:set var="counter" value="0"/>
   <c:forEach var="thisid" items="${
       getPatientOrAnd}">
   <c:set var="counter" value="${counter
       +1}"/>
    <c:if test="${counter % 3 == 1}">
     <c:set var="flag" value="0"/>
     <c:forEach var="consultation" items
         ="${theConsultations}">
      <c:if test="${fn:containsIgnoreCase
          (consultation.consultto, perio
          )}">
       <c:if test="${consultation.
           patientid == thisid}">
        <tr><td></td><td><a href="">${
            consultation.patientid}</a
            ></td>
        <c:set var="flag" value="1"/>
```

```
      <c:set var="periocount" value="${
           periocount + 1}"/>
     </c:if>
    </c:if>
   </c:forEach>
  </c:if>
  <c:if test="${counter % 3 == 2 &&
       flag == 1}">
   <td align="center">${thisid}</td>
  </c:if>
 </c:forEach>
 <c:if test="${periocount != 0 }">
  <tr> <td colspan="4"> </td><td align
       ="right"> <font size = "1">${
       periocount} patient(s) found </
       font> </td> </tr>
 </c:if>
 <c:if test="${periocount == 0 }">
  <tr><td> </td> <td colspan="4" align
       ="right"> <font size = "1"> No
       results found</font> </td></tr>
 </c:if>
</c:if>
<c:if test="${os == 'OS'}">
 <c:set var="oscount" value="0"/>
 <tr> <td colspan="1">  <b> Oral
      Surgery </b> </td> </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="thisid" items="${
      getPatientOrAnd}">
 <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter % 3 == 1}">
   <c:set var="flag" value="0"/>
   <c:forEach var="consultation" items
        ="${theConsultations}">
    <c:if test="${fn:containsIgnoreCase
         (consultation.consultto, os)
         }">
     <c:if test="${consultation.
          patientid == thisid}">
      <tr><td></td><td><a href="#">${
           consultation.patientid}</a
           ></td>
      <c:set var="flag" value="1"/>
      <c:set var="oscount" value="${
           oscount + 1}"/>
     </c:if>
    </c:if>
   </c:forEach>
  </c:if>
  <c:if test="${counter % 3 == 2 &&
       flag == 1}">
   <td align="center">${thisid}</td>
  </c:if>
 </c:forEach>
 <c:if test="${oscount != 0 }">
  <tr> <td colspan="4"> </td><td align
       ="right"> <font size = "1">${
       oscount} patient(s) found </font
       > </td> </tr>
 </c:if>
 <c:if test="${oscount == 0 }">
  <tr><td> </td> <td colspan="4" align
       ="right"> <font size = "1"> No
       results found</font> </td></tr>
 </c:if>
</c:if>
<c:if test="${endo == 'ENDO'}">
 <c:set var="endocount" value="0"/>
 <tr> <td colspan="1">  <b> Endodontics
       </b> </td> </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="thisid" items="${
      getPatientOrAnd}">
 <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter % 3 == 1}">
   <c:set var="flag" value="0"/>
   <c:forEach var="consultation" items
        ="${theConsultations}">
    <c:if test="${fn:containsIgnoreCase
         (consultation.consultto, endo)
         }">
     <c:if test="${consultation.
          patientid == thisid}">
      <tr><td></td><td><a href="#">${
           consultation.patientid}</a
           ></td>
      <c:set var="flag" value="1"/>
      <c:set var="endocount" value="${
           endocount + 1}"/>
     </c:if>
    </c:if>
   </c:forEach>
  </c:if>
  <c:if test="${counter % 3 == 2 &&
       flag == 1}">
   <td align="center">${thisid}</td>
  </c:if>
 </c:forEach>
 <c:if test="${endocount != 0 }">
  <tr> <td colspan="4"> </td><td align
       ="right"> <font size = "1">${
       endocount} patient(s) found </
       font> </td> </tr>
 </c:if>
 <c:if test="${endocount == 0 }">
  <tr><td> </td> <td colspan="4" align
       ="right"> <font size = "1"> No
       results found</font> </td></tr>
 </c:if>
```

```
</c:if>
<c:if test="${rpd == 'RPD'}">
 <c:set var="rpdcount" value="0"/>
 <tr> <td colspan="1"> <b> Removable
      Prosthodontics </b> </td> </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="thisid" items="${
      getPatientOrAnd}">
 <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter % 3 == 1}">
   <c:set var="flag" value="0"/>
   <c:forEach var="consultation" items
        ="${theConsultations}">
    <c:if test="${fn:containsIgnoreCase
         (consultation.consultto, rpd)
         }">
     <c:if test="${consultation.
          patientid == thisid}">
      <tr><td></td><td><a href="#">${
           consultation.patientid}</a
           ></td>
      <c:set var="flag" value="1"/>
      <c:set var="rpdcount" value="${
           rpdount + 1}"/>
     </c:if>
    </c:if>
   </c:forEach>
  </c:if>
  <c:if test="${counter % 3 == 2 &&
       flag == 1}">
   <td align="center">${thisid}</td>
  </c:if>
 </c:forEach>
 <c:if test="${rpdcount != 0 }">
  <tr> <td colspan="4"> </td><td align
       ="right"> <font size = "1">${
       rpdcount} patient(s) found </
       font> </td> </tr>
 </c:if>
 <c:if test="${rpdcount == 0 }">
  <tr><td> </td> <td colspan="4" align
       ="right"> <font size = "1"> No
       results found</font> </td></tr>
 </c:if>
</c:if>
<c:if test="${fpd == 'FPD'}">
 <c:set var="fpdcount" value="0"/>
 <tr> <td colspan="1"> <b> Fixed
      Partial Prosthodontics </b> </td>
       </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="thisid" items="${
      getPatientOrAnd}">
 <c:set var="counter" value="${counter
      +1}"/>
 <c:if test="${counter % 3 == 1}">
  <c:set var="flag" value="0"/>
  <c:forEach var="consultation" items
       ="${theConsultations}">
   <c:if test="${fn:containsIgnoreCase
        (consultation.consultto, fpd)
        }">
    <c:if test="${consultation.
         patientid == thisid}">
     <tr><td></td><td><a href="#">${
          consultation.patientid}</a
          ></td>
     <c:set var="flag" value="1"/>
     <c:set var="fpdcount" value="${
          fpdcount + 1}"/>
    </c:if>
   </c:if>
  </c:forEach>
 </c:if>
 <c:if test="${counter % 3 == 2 &&
      flag == 1}">
  <td align="center">${thisid}</td>
 </c:if>
</c:forEach>
<c:if test="${fpdcount != 0 }">
 <tr> <td colspan="4"> </td><td align
      ="right"> <font size = "1">${
      fpdcount} patient(s) found </
      font> </td> </tr>
</c:if>
<c:if test="${fpdcount == 0 }">
 <tr><td> </td> <td colspan="4" align
      ="right"> <font size = "1"> No
      results found</font> </td></tr>
</c:if>
</c:if>
<c:if test="${cd == 'CD'}">
 <c:set var="cdcount" value="0"/>
 <tr> <td colspan="1"> <b> Complete
      Denture</b> </td> </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="thisid" items="${
      getPatientOrAnd}">
 <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter % 3 == 1}">
   <c:set var="flag" value="0"/>
   <c:forEach var="consultation" items
        ="${theConsultations}">
    <c:if test="${fn:containsIgnoreCase
         (consultation.consultto, cd)
         }">
     <c:if test="${consultation.
          patientid == thisid}">
      <tr><td></td><td><a href="#">${
           consultation.patientid}</a
```

```
            ></td>
          <c:set var="flag" value="1"/>
          <c:set var="cdcount" value="${
              cdcount + 1}"/>
         </c:if>
        </c:if>
       </c:forEach>
      </c:if>
      <c:if test="${counter % 3 == 2 &&
          flag == 1}">
       <td align="center">${thisid}</td>
      </c:if>
     </c:forEach>
     <c:if test="${cdcount != 0 }">
      <tr> <td colspan="4"> </td><td align
          ="right"> <font size = "1">${
          cdcount} patient(s) found </font
          > </td> </tr>
     </c:if>
     <c:if test="${cdcount == 0 }">
      <tr><td> </td> <td colspan="4" align
          ="right"> <font size = "1"> No
          results found</font> </td></tr>
     </c:if>
    </c:if>
    <c:if test="${ortho == 'Ortho'}">
     <c:set var="orthocount" value="0"/>
     <tr> <td colspan="1"> <b>
          Orthodontics </b> </td> </tr>
     <c:set var="counter" value="0"/>
     <c:forEach var="thisid" items="${
          getPatientOrAnd}">
     <c:set var="counter" value="${counter
          +1}"/>
      <c:if test="${counter % 3 == 1}">
       <c:set var="flag" value="0"/>
       <c:forEach var="consultation" items
            ="${theConsultations}">
        <c:if test="${fn:containsIgnoreCase
            (consultation.consultto, ortho
            )}">
         <c:if test="${consultation.
             patientid == thisid}">
          <tr><td></td><td><a href="#">${
              consultation.patientid}</a
              ></td>
          <c:set var="flag" value="1"/>
          <c:set var="orthocount" value="${
              orthocount + 1}"/>
         </c:if>
        </c:if>
       </c:forEach>
      </c:if>
      <c:if test="${counter % 3 == 2 &&
          flag == 1}">
       <td align="center">${thisid}</td>
      </c:if>
     </c:forEach>
     <c:if test="${orthocount != 0 }">
      <tr> <td colspan="4"> </td><td align
          ="right"> <font size = "1">${
          orthocount} patient(s) found </
          font> </td> </tr>
     </c:if>
     <c:if test="${orthocount == 0 }">
      <tr><td> </td> <td colspan="4" align
          ="right"> <font size = "1"> No
          results found</font> </td></tr>
     </c:if>
    </c:if>
    <c:if test="${pedo == 'PEDO'}">
     <c:set var="pedocount" value="0"/>
     <tr> <td colspan="1"> <b> Pedodontics
          </b> </td> </tr>
     <c:set var="counter" value="0"/>
     <c:forEach var="thisid" items="${
          getPatientOrAnd}">
     <c:set var="counter" value="${counter
          +1}"/>
      <c:if test="${counter % 3 == 1}">
       <c:set var="flag" value="0"/>
       <c:forEach var="consultation" items
            ="${theConsultations}">
        <c:if test="${fn:containsIgnoreCase
            (consultation.consultto, pedo)
            }">
         <c:if test="${consultation.
             patientid == thisid}">
          <tr><td></td><td><a href="#">${
              consultation.patientid}</a
              ></td>
          <c:set var="flag" value="1"/>
          <c:set var="pedocount" value="${
              pedocount + 1}"/>
         </c:if>
        </c:if>
       </c:forEach>
      </c:if>
      <c:if test="${counter % 3 == 2 &&
          flag == 1}">
       <td align="center">${thisid}</td>
      </c:if>
     </c:forEach>
     <c:if test="${pedocount != 0 }">
      <tr> <td colspan="4"> </td><td align
          ="right"> <font size = "1">${
          pedocount} patient(s) found </
          font> </td> </tr>
     </c:if>
     <c:if test="${pedocount == 0 }">
      <tr><td> </td> <td colspan="4" align
          ="right"> <font size = "1"> No
```

```
        results found</font> </td></tr>
   </c:if>
  </c:if>
  <c:if test="${resto == 'Resto'}">
   <c:set var="restocount" value="0"/>
   <tr> <td colspan="1"> <b> Restorative
        Dentistry </b> </td> </tr>
   <c:set var="counter" value="0"/>
   <c:forEach var="thisid" items="${
       getPatientOrAnd}">
   <c:set var="counter" value="${counter
       +1}"/>
    <c:if test="${counter % 3 == 1}">
     <c:set var="flag" value="0"/>
     <c:forEach var="consultation" items
         ="${theConsultations}">
      <c:if test="${fn:containsIgnoreCase
          (consultation.consultto, resto
          )}">
       <c:if test="${consultation.
           patientid == thisid}">
        <tr><td></td><td><a href="#">${
            consultation.patientid}</a
            ></td>
        <c:set var="flag" value="1"/>
        <c:set var="restocount" value="${
            restocount + 1}"/>
       </c:if>
      </c:if>
     </c:forEach>
    </c:if>
    <c:if test="${counter % 3 == 2 &&
        flag == 1}">
     <td align="center">${thisid}</td>
    </c:if>
   </c:forEach>
   <c:if test="${restocount != 0 }">
    <tr> <td colspan="4"> </td><td align
        ="right"> <font size = "1">${
        restocount} patient(s) found </
        font> </td> </tr>
   </c:if>
   <c:if test="${restocount == 0 }">
    <tr><td> </td> <td colspan="4" align
        ="right"> <font size = "1"> No
        results found</font> </td></tr>
   </c:if>
  </c:if>
 </table>
</div>
</c:if>

<c:if test="${(caries == 'yes' ||
     recurrentcaries == 'yes' || amalgam
     == 'yes' || composite == 'yes' ||
     glassionomer == 'yes' ||
     tempfilling == 'yes' || extrusion == '
         yes' || intrusion == 'yes' ||
         mesialdrift == 'yes' || distaldrift
          == 'yes' ||
     completedenture == 'yes' ||
         singledenture == 'yes' ||
         removablepartial == 'yes' ||
         rotation == 'yes' || postcorecrown
        == 'yes' ||
     rootcanal == 'yes' || pitandfissure == '
         yes' || extracted == 'yes' ||
         missing == 'yes' || unerupted == '
         yes' ||
     impacted == 'yes' || porcelainfused == '
         yes' || acryliccrown == 'yes' ||
         metalcrown == 'yes' ||
         porcelaincrown == 'yes' ||
     restorable == 'yes' || nonrestorable ==
         'yes') && and1 == 'yes'}">
<br/> <br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
 Dental Condition Queries (AND)
 </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
 <col width="150"><col width="50"><col
     width="250"><col width="100">
 <tr> <td> </td> <td><b> Id </b></td> <td
     ><center><b> Name </b> </center></
     td></tr>

 <c:set var="counter" value="${0}"/>
 <c:set var="realcount" value="0"/>
 <tr> <td colspan="1"> <b> Results </b>
     </td> </tr>
 <c:forEach var="item" items="${results
     }">
  <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter % 3 ==  1}">
   <tr><td> </td><td>
   <c:set var="realcount" value="${
       realcount+1}"/>
   <a href="">${item} </a></td>
  </c:if>
  <c:if test="${counter % 3 == 2}">
   <td align="center"> ${item} </td><td
       ></td>
   </tr>
  </c:if>
 </c:forEach>
 <c:if test="${counter != 0 }">
```

```
  <tr> <td colspan="3"> </td><td align="
       right"> <font size = "1">${
       realcount} patient(s) found </font
       > </td> </tr>
 </c:if>
 <c:if test="${counter == 0 }">
  <tr><td> </td> <td colspan="3" align="
       right"> <font size = "1"> No
       results found</font> </td></tr>
 </c:if>
 </table>
</div>
</c:if>



<c:if test="${(caries == 'yes' ||
     recurrentcaries == 'yes' ||
     restoration=='yes' || extrusion == '
     yes' || intrusion == 'yes' ||
     mesialdrift == 'yes' || distaldrift
     == 'yes' ||
  completedenture == 'yes' ||
      singledenture == 'yes' ||
      removablepartial == 'yes' ||
      rotation == 'yes' || postcorecrown
      == 'yes' ||
  rootcanal == 'yes' || pitandfissure == '
      yes' || extracted == 'yes' ||
      missing == 'yes' || unerupted == '
      yes' ||
  impacted == 'yes' || porcelainfused == '
      yes' || acryliccrown == 'yes' ||
      metalcrown == 'yes' ||
      porcelaincrown == 'yes') && or1 ==
      'yes'}">


<br/> <br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
Dental Condition Queries (OR)
 </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
 <col width="150"><col width="50"><col
      width="250"><col width="150">
 <tr> <td> </td> <td><b> Id </b></td> <td
      ><center><b> Name </b> </center></
      td> <td><b> Tooth Number(s) </b> </
      td> </tr>
 <c:if test="${caries == 'yes'}">
  <tr> <td colspan="1">  <b> Caries </b>
       </td> </tr>
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       cariespatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="">${
         item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
 </c:if>

<c:if test="${recurrentcaries == 'yes
     '}">
 <tr> <td colspan="1"> <b> Recurrent
      Caries </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      recurrentcariespatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
```

```
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
 </c:if>

<c:if test="${restoration == 'yes'}">
  <tr> <td colspan="1"> <b> restoration
       </b> </td> </tr>
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       restorationpatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="">${
         item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
 </c:if>



 <c:if test="${extrusion == 'yes'}">
  <tr> <td colspan="1"> <b> Extrusion </b
       > </td> </tr>
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       extrusionpatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="../../
         patientDashboard.form?patientId=
         ${item}">${item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
 </c:if>

<c:if test="${intrusion == 'yes'}">
 <tr> <td colspan="1"><b> Intrusion </b>
        </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      intrusionpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
```

```
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${mesialdrift == 'yes'}">
 <tr> <td colspan="1"><b> Mesial Drift
      </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      mesialdriftpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${distaldrift == 'yes'}">
 <tr> <td colspan="1"><b> Distal Drift
      </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      distaldriftpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${completedenture == 'yes
     '}">
 <tr> <td colspan="1"><b> Complete
      Denture </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      completedentpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
```

```
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${singledenture == 'yes'}">
 <tr> <td colspan="1"><b> Single Denture
       </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      singledentpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${removablepartial == 'yes
     '}">
 <tr> <td colspan="1"><b> Removable
      Partial Denture </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      removablepartialdentpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${rotation == 'yes'}">
 <tr> <td colspan="1"><b> Rotation </b>
      </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      rotationpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
```

```
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${postcorecrown == 'yes'}">
 <tr> <td colspan="1"><b> Post Core
      Crown </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      postcorecrownpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${rootcanal == 'yes'}">
 <tr> <td colspan="1"><b> Root Canal </b
      > </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      rootcanalpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${pitandfissure == 'yes'}">
 <tr> <td colspan="1"><b> Pit and
      Fissure </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      pitandfissurepatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
```

```
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${extracted == 'yes'}">
 <tr> <td colspan="1"><b> Extracted </b>
       </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      extractedpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${missing == 'yes'}">
 <tr> <td colspan="1"><b> Missing </b>
      </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      missingpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${unerupted == 'yes'}">
 <tr> <td colspan="1"><b> Unerupted </b>
       </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      uneruptedpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
```

```
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${impacted == 'yes'}">
 <tr> <td colspan="1"><b> Impacted </b>
      </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      impactedpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${porcelainfused == 'yes'}">
 <tr> <td colspan="1"><b> Porcelain
      Fused </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      porcelainfusedpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${acryliccrown == 'yes'}">
 <tr> <td colspan="1"><b> Acrylic Crown
      </b> </td> </tr>
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      acryliccrownpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
```

```
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
</c:if>

<c:if test="${metalcrown == 'yes'}">
  <tr> <td colspan="1"><b> Metal Crown </
       b> </td> </tr>
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       metalcrownpatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="../../
         patientDashboard.form?patientId=
         ${item}">${item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
</c:if>

<c:if test="${porcelaincrown == 'yes'}">
  <tr> <td colspan="1"><b> Porcelain
       Crown </b> </td> </tr>
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       porcelaincrownpatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="../../
         patientDashboard.form?patientId=
         ${item}">${item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
</c:if>


 </table>
</div>
</c:if>


<c:if test="${(periodontics == 'yes' ||
     class1 == 'yes' || class2 == 'yes'
     || class3 == 'yes' || class4 == 'yes
     ' ||
  class5 == 'yes' || onlay == 'yes' ||
       extraction == 'yes' || odontectomy
       == 'yes' || specialcase == 'yes' ||
  pulpsedation == 'yes' ||
       tempfillingservice == 'yes' ||
       laminated == 'yes' || singlecrown
       == 'yes' || bridgeservice == 'yes'
       ||
  anterior == 'yes' || posterior == 'yes'
       || pedodontics == 'yes' ||
       orthodontics == 'yes' ||
       acuteinfections == 'yes' ||
  traumaticinjuries == 'yes' ||
       completedentservice == 'yes' ||
       singledentservice == 'yes' ||
       removablepartialservice == 'yes')
  && and2 == 'yes'}">
```

```
<br/> <br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
Needed Services Queries (AND)
 </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
 <col width="150"><col width="50"><col
      width="250"><col width="100">
 <tr> <td> </td> <td><b> Id </b></td> <td
      ><center><b> Name </b> </center></
      td></tr>

 <c:set var="counter" value="${0}"/>
 <c:set var="realcount" value="${0}"/>
 <tr> <td colspan="1"> <b> Results </b>
      </td> </tr>
 <c:forEach var="item" items="${results2
      }">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter % 3 ==  1}">
   <tr><td> </td><td>
   <c:set var="realcount" value="${
        realcount+1}"/>
   <a href="">${item} </a></td>
  </c:if>
  <c:if test="${counter % 3 == 2}">
   <td align="center"> ${item} </td><td
        ></td>
   </tr>
  </c:if>
 </c:forEach>
 <c:if test="${counter != 0 }">
  <tr> <td colspan="3"> </td><td align="
       right"> <font size = "1">${
       realcount} patient(s) found </font
       > </td> </tr>
 </c:if>
 <c:if test="${counter == 0 }">
  <tr><td> </td> <td colspan="3" align="
       right"> <font size = "1"> No
       results found</font> </td></tr>
 </c:if>
 </table>
</div>
</c:if>


<c:if test="${(periodontics == 'yes' ||
     class1 == 'yes' || class2 == 'yes'
     || class3 == 'yes' || class4 == 'yes
     ' ||
     class5 == 'yes' || onlay == 'yes' ||
          extraction == 'yes' || odontectomy
          == 'yes' || specialcase == 'yes' ||
     pulpsedation == 'yes' ||
          tempfillingservice == 'yes' ||
          laminated == 'yes' || singlecrown
          == 'yes' || bridgeservice == 'yes'
          ||
     anterior == 'yes' || posterior == 'yes'
          || pedodontics == 'yes' ||
          orthodontics == 'yes' ||
          acuteinfections == 'yes' ||
     traumaticinjuries == 'yes' ||
          completedentservice == 'yes' ||
          singledentservice == 'yes' ||
          removablepartialservice == 'yes')
     && or2 == 'yes'}">
<br/> <br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
Needed Services Queries (OR)
 </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2"> <
      col width="150"><col width="50"><
      col width="250"><col width="150">
 <tr> <td> </td> <td><b> Id </b></td> <td
      ><center><b> Name </b> </center></
      td> <td><b> Tooth Number(s) </b> </
      td> </tr>
 <c:if test="${periodontics == 'yes'}">
  <c:set var="periodonticscount" value
       ="0"/>
  <tr> <td colspan="1"><b> Management of
       Periodontal Disease  </b> </td> </
       tr>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       periodonticspatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="">${
         item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
```

```
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${class1 == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Class 1 </b>
      </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      class1patients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${class2 == 'yes'}">
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <tr> <td colspan="1"><b> Class 2 </b>
      </td> </tr>
 <c:forEach var="item" items="${
      class2patients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0}">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0}">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${class3 == 'yes'}">
 <c:set var="count" value="0"/>
 <c:set var="realcount" value="0"/>
 <tr> <td colspan="1"><b> Class 3 </b>
      </td> </tr>
 <c:forEach var="item" items="${
      class3patients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0}">
  <tr> <td colspan="4"> </td><td align="
```

```
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0}">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
</c:if>

<c:if test="${class4 == 'yes'}">
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <tr> <td colspan="1"><b> Class 4 </b>
       </td> </tr>
  <c:forEach var="item" items="${
       class4patients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="">${
         item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center"> ${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0}">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0}">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
</c:if>

<c:if test="${class5 == 'yes'}">
  <c:set var="count" value="0"/>
  <c:set var="realcount" value="0"/>
  <tr> <td colspan="1"><b> Class 5 </b>
       </td> </tr>
  <c:forEach var="item" items="${
       class5patients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="">${
         item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0}">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
  <c:if test="${count == 0}">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1"> No results
         found. </font></td> </tr>
  </c:if>
</c:if>

<c:if test="${onlay == 'yes'}">
  <c:set var="count" value="0"/>
  <tr> <td colspan="1"><b> Onlay </b> </
       td> </tr>
  <c:set var="realcount" value="0"/>
  <c:forEach var="item" items="${
       onlaypatients}">
   <c:set var="count" value="${count +
        1}"/>
   <c:if test="${count % 3 == 1}">
    <c:set var="realcount" value="${
         realcount+1}"/>
    <tr> <td> </td> <td> <a href="">${
         item}</a></td>
   </c:if>
   <c:if test="${count % 3 == 2}">
    <td align="center"> ${item} </td>
   </c:if>
   <c:if test="${count % 3 == 0}">
    <td align="center">${item} </td><td
         ></td><tr>
   </c:if>
  </c:forEach>
  <c:if test="${count != 0 }">
   <tr> <td colspan="4"> </td><td align="
        right"><font size="1">${realcount
        } patient(s) found. </font></td>
        </tr>
  </c:if>
```

```
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${extraction == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Extraction </b
      > </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      extractionpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${odontectomy == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Odontectomy </
      b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      odontectomypatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${specialcase == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Special Case
      </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      specialcasepatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
```

```
 </c:if>
</c:if>

<c:if test="${pulpsedation == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Pulp Sedation
      </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      pulpsedationpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="">${
        item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${crownrecementation == 'yes
     '}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Crown
      Recementation </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      crownrecementationpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${tempfillingservice == 'yes
     '}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Temporary
      Filling </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      tempfillingpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
```

```
 </c:if>
</c:if>

<c:if test="${laminated == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Laminated </b>
      </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
     laminatedpatients}">
  <c:set var="count" value="${count +
      1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
       realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
       patientDashboard.form?patientId=
       ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
       ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1">${realcount
      } patient(s) found. </font></td>
      </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1"> No results
       found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${singlecrown == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Single Crown
     </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
     singlecrownpatients}">
  <c:set var="count" value="${count +
      1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
       realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
       patientDashboard.form?patientId=
       ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
       ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1">${realcount
      } patient(s) found. </font></td>
      </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1"> No results
       found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${bridgeservice == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Bridge Service
      </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
     bridgeservicepatients}">
  <c:set var="count" value="${count +
      1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
       realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
       patientDashboard.form?patientId=
       ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
       ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1">${realcount
      } patient(s) found. </font></td>
      </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1"> No results
       found. </font></td> </tr>
 </c:if>
```

```
</c:if>

<c:if test="${anterior == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Anterior </b>
      </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      anteriorpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${posterior == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Posterior </b>
      </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      posteriorpatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${pedodontics == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Pedodontics </
      b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      pedodonticspatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>
```

```
<c:if test="${orthodontics == 'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Orthodontics
      </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      orthodonticspatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${acuteinfections == 'yes
     '}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Acute
      Infections </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      acuteinfectionspatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${traumaticinjuries == 'yes
     '}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Traumatic
      Injuries </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
      traumaticinjuriespatients}">
  <c:set var="count" value="${count +
       1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
        realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
        patientDashboard.form?patientId=
        ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
        ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1">${realcount
       } patient(s) found. </font></td>
       </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
       right"><font size="1"> No results
        found. </font></td> </tr>
 </c:if>
```

```
</c:if>

<c:if test="${completedentservice == '
    yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Complete
     Denture </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
     completedenturepatients}">
  <c:set var="count" value="${count +
      1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
       realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
       patientDashboard.form?patientId=
       ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
       ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1">${realcount
      } patient(s) found. </font></td>
      </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1"> No results
       found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${singledentservice == 'yes
    '}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Single Denture
      </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
     singledenturepatients}">
  <c:set var="count" value="${count +
      1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
       realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
       patientDashboard.form?patientId=
       ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
       ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1">${realcount
      } patient(s) found. </font></td>
      </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1"> No results
       found. </font></td> </tr>
 </c:if>
</c:if>

<c:if test="${removablepartialservice ==
    'yes'}">
 <c:set var="count" value="0"/>
 <tr> <td colspan="1"><b> Removable
     Partial Denture </b> </td> </tr>
 <c:set var="realcount" value="0"/>
 <c:forEach var="item" items="${
     removablepartialpatients}">
  <c:set var="count" value="${count +
      1}"/>
  <c:if test="${count % 3 == 1}">
   <c:set var="realcount" value="${
       realcount+1}"/>
   <tr> <td> </td> <td> <a href="../../
       patientDashboard.form?patientId=
       ${item}">${item}</a></td>
  </c:if>
  <c:if test="${count % 3 == 2}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${count % 3 == 0}">
   <td align="center">${item} </td><td
       ></td><tr>
  </c:if>
 </c:forEach>
 <c:if test="${count != 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1">${realcount
      } patient(s) found. </font></td>
      </tr>
 </c:if>
 <c:if test="${count == 0 }">
  <tr> <td colspan="4"> </td><td align="
      right"><font size="1"> No results
```

```
        found. </font></td> </tr>
  </c:if>
 </c:if>
 </table>
</div>
<br/><br/>
```

Listing 61: sectionForm

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Manage Section</title>


<script>

$(document).ready(function(){
 $('#sectionSubmit').click(function(){
  sectionValidate();
 });

 $('#section_name').bind('change',
      function(){
  var section_name = $(this).val();
  var allSectionnames= '${allSectionnames
       }'.split(',');
  //alert(role_name);
  isSectionExist(allSectionnames,
       section_name);

 });


});



function isSectionExist(allSectionnames,
    section_name){
 var sectionBool=true;
 for(i=0;i<allSectionnames.length;i++){
  if(section_name==allSectionnames[i]){
   $("#sectionExist").show();
   roleBool=false;
  }
 }
 if(sectionBool)
  $("#sectionExist").hide();

}

function sectionValidate(){

 //fields
 var section_name= document.
      getElementById('section_name').
      value;
 var section_description= document.
      getElementById('section_description
      ').value;
 var sectionConfirm="";

 //check for missing input
 if(role_name=="" || role_description
     =="")
  sectionConfirm+="Required information
       is incomplete. Please check if all
        information needed is filled.\n";

 if(sectionConfirm==""){
   $('#sectionform').submit();
 }
 else
  alert(sectionConfirm);
}



</script>


</head>

<body>
<c:url var="cancelURL" value="/forms/
    dentist/login/index.html" />

<div class="form-body">
<div class="sub-body-title">Add Section</
    div>
</c:if>
<a href="${viewQueryURL}"> Go Back </a>
<br/>
</div>
</body>
</html>
```

```
<form:form action="sectionoutput.html"
     commandName="section" id="
     sectionform">

<table class="form-table">

<tr>

<td>Section Name :</td>

<td><form:input path="section_name" />
<span id="sectionExist" style="display:
     none; color:red">Section already
     exists!</span></td>

</tr>

<tr>

<td>Section Description :</td>

<td><form:textarea path="
     section_description" rows="5" cols
     ="30" /></td>

</tr>


<tr>


<td><input type="submit" value="Submit"
     id="sectionSubmit" class="classname
     "/></td>
<td>
<input type="button" value="Cancel"
     onClick="window.location='${
     cancelURL}'" class="classname"/></td
     >

</tr>

</table>

</form:form>
</div>
</body>

</html>
```

Listing 62: statisticsform

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@ taglib prefix="fn" uri="http://java.
     sun.com/jsp/jstl/functions" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript" src="<c:
     url value="/showresult.js" />" >
</script>
<script type="text/javascript" src="<c:
     url value="/calendar.js" />" >
</script>

</head>
<body>
<c:url var="homeURL" value="/forms/
     dentist/login/index.html" />

<div class="searchDiv">

<h2>Statistics</h2>
<br/>
<c:url var="statURL" value="/forms/
     dentist/statistics/statisticsResults
     .html" />

<a href="${homeURL}"> Go Back </a> <br/><
     br/>
<form name="searchform" action="${statURL
     }" method="POST" onSubmit="return
     checkForm()">
<b>
  <table class="boxHeader">
 <tr><td align="left" width="80%">
 Range
 </td></tr>
 </table>
 </b>
  <div class="box">
  <table>
   <tr>
    <td> Specify range of date: </td>
    <td><input type="text" name="datefrom
         " id="datefrom" size="10"
         onClick="showCalendar(this)" />
    <a href="javascript:NewCal('datefrom
```

```
','ddmmyyyy')">                                                    /
                                <img                               cal
                                    src                            .
                                    ="                             gif
                                    $                              "
                                    {
                                    pageContext                    width
                                    .                              ="16"
                                    request
                                    .                              height
                                    contextPath                    ="16"
                                    }/
                                    images                         border
                                    /                              ="0"
                                    cal
                                    .                              alt
                                    gif                            ="
                                    "                              Pick

                                    width                          a
                                    ="16"
                                                                   date
                                    height                         "></
                                    ="16"                          a
                                                                   >
                                    border
                                    ="0"                                (dd/
                                                                        MM
                                    alt                                 /
                                    ="                                  yyyy
                                    Pick                                )
                                                                        </
                                    a                                   td
                                                                        >
                                    date
                                    "></  </tr>
                                    a     <tr>
                                    >       <td>Specify age:</td>
                                            <td><input type="text" name="agefrom"
- <input type="text" name="dateto"                id="agefrom"> - <input type="
    id="dateto" size="10"  onClick              text" name="ageto" id="ageto">
    ="showCalendar(this)" /><a href             </td>
    ="javascript:NewCal('dateto','          </tr>
    ddmmyyyy')">                            <tr>
                                <img          <td>City:</td>
                                    src       <td><input type="text" name="city" id
                                    ="              ="city"></td>
                                    $       </tr>
                                    {     </table>
                                    pageContext
                                    .     <br/>
                                    request
                                    .
                                    contextPath
                                    }/    <table class="boxHeader">
                                    images<col width="50%"> <col width="50%">
```

```
 <tr><td>Other Conditions (Using
      Demographics)</td>
 <td align="right"><input type="checkbox
      " id="or3" name="condd" value="or3
      " onClick="checkOrAnd3('or')"
      checked>Or |
 <input type="checkbox" id="and3" name="
      condd" value=and3 onClick="
      checkOrAnd3('and')" >And
 </td></tr>
 </table>
</b>
<br/>

 <b >
 <table class="boxHeader">
 <col width="20%"> <col width="80%">
 <tr><td>Sections</td>
 <td align="right">
 <input type="checkbox" id="selectall3"
      onclick="selectAll3();">Select All
       </td></tr>
 </table>
</b>
<div class="box">
 <table><col width="250"><col width
      ="250"><col width="250">
  <tr>
   <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',','),'operative
        ') || fn:containsIgnoreCase(fn:
        join(sessionScope.
        currentRoleList,',','),'
        administrator')}">
   <td><b>Operative Dentistry</b></td>
   </c:if>
   <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',','),'medicine')
         || fn:containsIgnoreCase(fn:
        join(sessionScope.
        currentRoleList,',','),'
        administrator')}">
   <td><b>Oral Medicine </b></td>
   </c:if>
   <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',','),'
        prosthodontics') || fn:
        containsIgnoreCase(fn:join(
        sessionScope.currentRoleList
        ,',','),'administrator')}">
   <td><b>Prosthodontics</b></td>
   </c:if>
  </tr>
<tr>
 <c:if test="${fn:containsIgnoreCase(
      fn:join(sessionScope.
      currentRoleList,',','),'operative
      ') || fn:containsIgnoreCase(fn:
      join(sessionScope.
      currentRoleList,',','),'
      administrator')}">
 <td><input type="checkbox" id="s1"
      name="perio" value="PERIO" />
      Periodontics</td>
 </c:if>
 <c:if test="${fn:containsIgnoreCase(
      fn:join(sessionScope.
      currentRoleList,',','),'medicine')
       || fn:containsIgnoreCase(fn:
      join(sessionScope.
      currentRoleList,',','),'
      administrator')}">
 <td><input type="checkbox" id="s2"
      name="rpd" value="RPD" />
      Removable Prosthodontics</td>
 </c:if>
 <c:if test="${fn:containsIgnoreCase(
      fn:join(sessionScope.
      currentRoleList,',','),'
      prosthodontics') || fn:
      containsIgnoreCase(fn:join(
      sessionScope.currentRoleList
      ,',','),'administrator')}">
 <td><input type="checkbox" id="s3"
      name="ortho" value="Ortho" />
      Orthodontics</td>
 </c:if>
</tr>
<tr>
 <c:if test="${fn:containsIgnoreCase(
      fn:join(sessionScope.
      currentRoleList,',','),'operative
      ') || fn:containsIgnoreCase(fn:
      join(sessionScope.
      currentRoleList,',','),'
      administrator')}">
 <td><input type="checkbox" id="s4"
      name="os" value="OS" />Oral
      Surgery</td>
 </c:if>
 <c:if test="${fn:containsIgnoreCase(
      fn:join(sessionScope.
      currentRoleList,',','),'medicine')
       || fn:containsIgnoreCase(fn:
      join(sessionScope.
      currentRoleList,',','),'
      administrator')}">
 <td><input type="checkbox" id="s5"
      name="fpd" value="FPD" />Fixed
```

```
        Partial Prosthodontics</td>
    </c:if>
    <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',',),'
        prosthodontics') || fn:
        containsIgnoreCase(fn:join(
        sessionScope.currentRoleList
        ,',',),'administrator')}">
    <td><input type="checkbox" id="s6"
        name="pedo" value="PEDO" />
        Pedodontics</td>
    </c:if>
   </tr>
   <tr>
    <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',',),'operative
        ') || fn:containsIgnoreCase(fn:
        join(sessionScope.
        currentRoleList,',',),'
        administrator')}">
    <td><input type="checkbox" id="s7"
        name="endo" value="ENDO" />
        Endodontics</td>
    </c:if>
    <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',',),'medicine')
         || fn:containsIgnoreCase(fn:
        join(sessionScope.
        currentRoleList,',',),'
        administrator')}">
    <td><input type="checkbox" id="s8"
        name="cd" value="CD" />Complete
        Denture</td>
    </c:if>
    <c:if test="${fn:containsIgnoreCase(
        fn:join(sessionScope.
        currentRoleList,',',),'
        prosthodontics') || fn:
        containsIgnoreCase(fn:join(
        sessionScope.currentRoleList
        ,',',),'administrator')}">
    <td><input type="checkbox" id="s9"
        name="resto" value="Resto" />
        Restorative Dentistry</td>
    </c:if>
   </tr>
  </table>
  <br/>
 </div>
 <br/>
 <br/>

 <b >
  <table class="boxHeader">
  <col width="20%"> <col width="80%">
  <tr><td>Dental Condition</td>
  <td align="right"><input type="checkbox
       " id="or1" name="or1" value="yes"
       checked>Or |
  <input type="checkbox" id="and1" name="
       and1" value="yes">And |
  <input type="checkbox" id="selectall1"
       onclick="selectAll1();">Select All
        </td></tr>
  </table></b>
 <div class="box">
  <table><col width="250"><col width
       ="250"><col width="250"><col width
       ="250"><col width="250">
   <tr>
    <td><input type="checkbox" id="dc1"
        name="caries" value="yes">
        Caries</td>
    <td><input type="checkbox" id="dc2"
        name="extrusion" value="yes">
        Extrusion</td>
    <td><input type="checkbox" id="dc3"
        name="completedenture" value="
        yes"> Complete Denture</td>
    <td><input type="checkbox" id="dc4"
        name="impacted" value="yes">
        Impacted</td>
    <td><input type="checkbox" id="dc5"
        name="porcelainfused" value="yes
        "> Porcelain Fused To Metal</td>
   </tr>
   <tr>
    <td><input type="checkbox" id="dc6"
        name="recurrentcaries" value="
        yes"> Recurrent Caries</td>
    <td><input type="checkbox" id="dc7"
        name="intrusion" value="yes">
        Intrusion</td>
    <td><input type="checkbox" id="dc8"
        name="singledenture" value="yes
        "> Single Denture</td>
    <td><input type="checkbox" id="dc9"
        name="postcorecrown" value="yes
        "> Post Core Crown</td>
   </tr>
   <tr>
    <td><input type="checkbox" id="dc11"
         name="restoration" value="yes">
         Restoration</td>
    <td><input type="checkbox" id="dc12"
         name="mesialdrift" value="yes">
         Mesial Drifting Rotation</td>
    <td><input type="checkbox" id="dc13"
         name="removablepartial" value="
```

```
        yes"> Removable Partial Denture
        </td>
   <td><input type="checkbox" id="dc14"
        name="acryliccrown" value="yes
       "> Acrylic Crown</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="dc17"
        name="distaldrift" value="yes">
        Distal Drifting Rotation</td>
   <td><input type="checkbox" id="dc18"
        name="pitandfissure" value="yes
       "> Pit and Fissure Sealants</td>
   <td><input type="checkbox" id="dc19"
        name="metalcrown" value="yes">
       Metal Crown</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="dc21"
        name="rotation" value="yes">
       Rotation</td>
   <td><input type="checkbox" id="dc22"
        name="missing" value="yes">
       Missing</td>
   <td><input type="checkbox" id="dc23"
        name="porcelaincrown" value="
       yes"> Porcelain Crown</td>
  </tr>
  <tr>
   <td><input type="checkbox" id="dc25"
        name="extracted" value="yes">
       Extracted</td>
   <td><input type="checkbox" id="dc26"
        name="rootcanal" value="yes">
       Root Canal Treatment</td>
   <td><input type="checkbox" id="dc27"
        name="unerupted" value="yes">
       Unerupted</td>
  </tr>
 </table>
 <br/>
</div>
<br/>
<br/>

<div class="box">
 <b >
  <table class="boxHeader">
  <col width="20%"> <col width="80%">
  <tr><td>Services Needed</td>
  <td align="right"><input type="
       checkbox" id="or2" name="or2"
       value="yes" checked>Or |
  <input type="checkbox" id="and2" name
       ="and2" value="yes">And |
  <input type="checkbox" id="selectall2"
        onclick="selectAll2();">Select
       All </td></tr>
 </table>
</b>
<table><col width="250"><col width
     ="250"><col width="300">
 <tr>
  <td><u>Periodontics</u></td>
 </tr>
 <tr>
  <td colspan="3"><input type="checkbox
       " id="sn1" name="periodontics"
       value="yes"> Management of
       Periodontal Disease</td>
 </tr>

<!--ROW_2-->

 <tr>
  <td><br/><u>Operative Dentistry </u
       ></td>
  <td><br/><u> Surgery </u></td>
  <td><br/><u>Emergency Treatment</u></
       td>
 </tr>
 <tr>
  <td><input type="checkbox" id="sn2"
       name="class1" value="yes"> Class
        I</td>
  <td><input type="checkbox" id="sn3"
       name="extraction" value="yes">
       Extraction</td>
  <td><input type="checkbox" id="sn4"
       name="pulpsedation" value="yes">
        Pulp Sedation</td>
 </tr>
 <tr>
  <td><input type="checkbox" id="sn5"
       name="class2" value="yes"> Class
        II</td>
  <td><input type="checkbox" id="sn6"
       name="odontectomy" value="yes">
       Odontectomy</td>
  <td><input type="checkbox" id="sn7"
       name="crownrecementation" value
       ="yes"> Recementation of Crowns
       </td>
 </tr>
 <tr>
  <td><input type="checkbox" id="sn8"
       name="class3" value="yes"> Class
        III</td>
  <td><input type="checkbox" id="sn9"
       name="specialcase" value="yes">
       Special Case</td>
  <td><input type="checkbox" id="sn10"
```

```
        name="tempfillingservice" value
        ="yes"> Temporary Fillings</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn11"
        name="class4" value="yes"> Class
         IV</td>
     <td><input type="checkbox" id="sn12"
        name="pedodontics" value="yes">
        Pedodontics </td>
     <td><input type="checkbox" id="sn13"
        name="acuteinfections" value="
        yes"> Management of acute
        infections</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn14"
        name="class5" value="yes"> Class
         V</td>
     <td><input type="checkbox" id="sn15"
        name="orthodontics" value="yes">
         Orthodontics</td>
     <td><input type="checkbox" id="sn16"
        name="traumaticinjuries" value="
        yes"> Management of Temporary
        Injuries</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn17"
        name="onlay" value="yes"> Onlay
        </td>
    </tr>

<!--ROW-3-->

    <tr>
     <td><br/><u>Fixed Partial Denture</u
        ></td>
     <td><br/><u>Prosthodontics</u><br/>
     <td><br/><u>Endodontics</u></td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn18"
        name="laminated" value="yes">
        Laminated</td>
     <td><input type="checkbox" id="sn19"
        name=completedentservice" value
        ="yes"> Complete Denture</td>
     <td><input type="checkbox" id="sn20"
        name="anterior" value="yes">
        Anterior</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn21"
        name="singlecrown" value="yes">
        Single Crown</td>
     <td><input type="checkbox" id="sn22"
        name="singledentservice" value="
        yes"> Single Denture</td>
     <td><input type="checkbox" id="sn23"
        name="posterior" value="yes">
        Posterior</td>
    </tr>
    <tr>
     <td><input type="checkbox" id="sn24"
        name="bridge" value="yes">
        Bridge</td>
     <td><input type="checkbox" id="sn25"
        name="removablepartialservice"
        value="yes"> Removable Partial
        Denture</td>
    </tr>
   </table>
   <br/>
  </div>

  <br/>
  <input type="submit" value="Generate
      Report" class="classname">
  <input type="button" name="clear" value
      ="Clear Form" onclick="clearForm()
      ;" class="classname"/>
  <a href="${homeURL}"> Go Back </a> <br
      /><br/>
 </form>

 </div>
 </body>
 </html>
```

Listing 63: statisticsResults

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@taglib uri="http://java.sun.com/
        jsp/jstl/core" prefix="c" %>
       <%@ taglib prefix="fn" uri="http
           ://java.sun.com/jsp/jstl/
           functions" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>

<meta http-equiv="Content-Type" content="
```

```
    text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript" src="<c:
    url value="/showresult.js" />" >
</script>

</head>
<body>
<div class="searchDiv">

<c:url var="viewQueryURL" value="/forms/
    dentist/search/statisticsform.html"
    />

<a href="${viewQueryURL}"> Go Back </a>

<h2>Statistics</h2>

<c:set var="condd" value="${condd}"/>

<c:set var="datefrom" value="${datefrom
    }"/>
<c:set var="dateto" value="${dateto}"/>
<c:set var="datecheck" value="${datecheck
    }"/>
<c:set var="agefrom" value="${agefrom}"/>
<c:set var="ageto" value="${ageto}"/>
<c:set var="agecheck" value="${agecheck
    }"/>
<c:set var="sexcheck" value="${sexcheck
    }"/>
<c:set var="sex" value="${sex}"/>
<c:set var="occupation" value="${
    occupation}"/>
<c:set var="city" value="${city}"/>
<c:set var="occupationcheck" value="${
    occupationcheck}"/>
<c:set var="citycheck" value="${citycheck
    }"/>
<c:set var="caries" value="${caries}"/>
<c:set var="recurrentcaries" value="${
    recurrentcaries}"/>
<c:set var="restoration" value="${
    restoration}"/>

<c:set var="periodontics" value="${
    periodontics}"/>
<c:set var="class1" value="${class1}"/>
<c:set var="class2" value="${class2}"/>
<c:set var="class3" value="${class3}"/>
<c:set var="class4" value="${class4}"/>
<c:set var="class5" value="${class5}"/>
<c:set var="onlay" value="${onlay}"/>

<c:set var="extraction" value="${
    extraction}"/>
<c:set var="odontectomy" value="${
    odontectomy}"/>
<c:set var="specialcase" value="${
    specialcase}"/>
<c:set var="pedodontics" value="${
    pedodontics}"/>
<c:set var="orthodontics" value="${
    orthodontics}"/>

<c:set var="pulpsedation" value="${
    pulpsedation}"/>
<c:set var="crownrecementation" value="${
    crownrecementation}"/>
<c:set var="tempfillingservice" value="${
    tempfillingservice}"/>
<c:set var="acuteinfections" value="${
    acuteinfections}"/>
<c:set var="traumaticinjuries" value="${
    traumaticinjuries}"/>

<c:set var="laminated" value="${laminated
    }"/>
<c:set var="singlecrown" value="${
    singlecrown}"/>
<c:set var="bridgeservice" value="${
    bridge}"/>

<c:set var="anterior" value="${anterior
    }"/>
<c:set var="posterior" value="${posterior
    }"/>

<c:set var="completedentservice" value="$
    {completedentservice}"/>
<c:set var="singledentservice" value="${
    singledentservice}"/>
<c:set var="removablepartialservice"
    value="${removablepartialservice}"/>

<c:set var="agecount" value="0"/>
<c:set var="sexMcount" value="0"/>
<c:set var="sexFcount" value="0"/>
<br/>
<c:if test="${city!='' || (agefrom != 0
    && ageto != 100) || condd=='and3'}">

<b>
  <table class="boxHeader">
 <tr><td align="left" width="80%">
 Patients registered from ${datefrom} to
     ${dateto}

 </td></tr>
 </table>
```

```
 </b>
<div class="box">
 <table id = "sample" cellspacing="2"> <
      col width="200"><col width="75"><
      col width="75">
  <c:set var="counter" value="0"/>
  <c:forEach var="item" items="${totals}"
       varStatus="status">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr> <td> Total # of Females </td> <
        td align="center"> ${item} </td>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${item}% </td> </
        tr>
   </c:if>
   <c:if test="${counter == 3}">
    <tr> <td> Total # of Males </td> <td
        align="center"> ${item} </td>
   </c:if>
  </c:forEach>
</table>

<c:if test="${agefrom != 0 && ageto !=
     100}">
 <br/>
 <b>AGE GROUP (${agefrom} - ${ageto})</b
     >
 <table id = "sample" cellspacing="2">
      <col width="200"><col width="75"><
      col width="75">
  <c:set var="counter" value="0"/>
  <c:forEach var="item" items="${
       agegrouppatients}" varStatus="
       status">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr> <td> # of Females </td> <td
        align="center"> ${item} </td>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${item}% </td>
        </tr>
   </c:if>
   <c:if test="${counter == 3}">
    <tr> <td> # of Males </td> <td align
        ="center"> ${item} </td>
   </c:if>
  </c:forEach>
 </table>
</c:if>

 <c:if test="${city != ''}">
 <br/>
 <b> City: '${city}' </b>
 <table id = "sample" cellspacing="2">
      <col width="200"><col width="75"><
      col width="75">
  <c:set var="counter" value="0"/>
  <c:forEach var="item" items="${
       citygrouppatients}" varStatus="
       status">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr> <td> # of Females </td> <td
        align="center"> ${item} </td>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${item}% </td>
        </tr>
   </c:if>
   <c:if test="${counter == 3}">
    <tr> <td> # of Males </td> <td align
        ="center"> ${item} </td>
   </c:if>
  </c:forEach>
 </table>
</c:if>
</div>
 </c:if>
 <br/>
 <br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
 Other Conditions from ${datefrom} to ${
     dateto}
 <c:if test="${condd == 'or3' }">(OR)</c:
     if>
 <c:if test="${condd == 'and3' }">(AND)</
     c:if>
 </td></tr>
 </table>
 </b>


<c:if test="${perio == 'PERIO' || os == '
     OS' || endo == 'ENDO' || rpd == 'RPD
     ' || fpd == 'FPD' ||
  cd == 'CD' || ortho == 'Ortho' || pedo
      == 'PEDO' || resto == 'Resto'}">
<br/>
<b >
<table class="boxHeader">
 <tr><td align="left" width="80%">
```

```
 Sections
 </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
  <col width="200"><col width="75"><col
       width="75"><col width="75"><col
       width="75"><col width="75"><col
       width="100"><col width="100"><col
       width="100">

  <tr><td><b>Condition</b></td><td align
       ="center"><b># of Cases</b></td><
       td align="center"><b>Females</b></
       td><td align="center"><b>% Females
       </b></td><td align="center"><b>
       Males</b></td><td align="center"><
       b>% Males</b></td><td align="
       center"><b>% Females (over total
       females)</b></td><td align="center
       "><b>% Males (over total males)</b
       ></td><td align="center"><b>%
       Cases (over total patients)</b></
       td></tr>
 <c:set var="total" value="0"/>
  <c:if test="${perio == 'PERIO'}">
    <c:set var="counter" value="0"/>
   <c:forEach var="periocountt" items="${
        periocount}">
    <c:set var="counter" value="${counter
         +1}"/>
    <c:if test="${counter == 1}">
     <tr><td>Class 1</td><td align="
          center"> ${periocountt}</td>
     <c:set var="total" value="${total+
          periocountt}"/>
    </c:if>
    <c:if test="${counter == 2 || counter
          == 4}">
     <td align="center"> ${periocountt}
          </td>
    </c:if>
    <c:if test="${counter == 3 || counter
          > 4}">
     <td align="center"> ${periocountt}%
          </td>
    </c:if>
   </c:forEach>
   </tr>
  </c:if>
  <c:if test="${os == 'OS'}">
     <c:set var="counter" value="0"/>
   <c:forEach var="oscountt" items="${
        oscount}">
    <c:set var="counter" value="${counter
         +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Oral Surgery</td><td align="
         center"> ${oscountt}</td>
    <c:set var="total" value="${total+
         oscountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
         == 4}">
    <td align="center"> ${oscountt} </td
         >
   </c:if>
   <c:if test="${counter == 3 || counter
         > 4}">
    <td align="center"> ${oscountt}% </
         td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${endo == 'ENDO'}">
    <c:set var="counter" value="0"/>
  <c:forEach var="endocountt" items="${
       endocount}">
   <c:set var="counter" value="${counter
        +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Endodontics</td><td align="
         center"> ${endocountt}</td>
    <c:set var="total" value="${total+
         endocountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
         == 4}">
    <td align="center"> ${endocountt} </
         td>
   </c:if>
   <c:if test="${counter == 3 || counter
         > 4}">
    <td align="center"> ${endocountt}%
         </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${fpd == 'FPD'}">
     <c:set var="counter" value="0"/>
  <c:forEach var="fpdcountt" items="${
       fpdcount}">
   <c:set var="counter" value="${counter
        +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Fixed Partial Prosthodontics
         </td><td align="center"> ${
         fpdcountt}</td>
    <c:set var="total" value="${total+
```

```
        fpdcountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${fpdcountt} </
        td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${fpdcountt}% </
        td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${cd == 'CD'}">
    <c:set var="counter" value="0"/>
  <c:forEach var="cdcountt" items="${
      cdcount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Complete Denture</td><td
        align="center"> ${cdcountt}</td
        >
    <c:set var="total" value="${total+
        cdcountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${fpdcountt} </
        td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${cdcountt}% </
        td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${ortho == 'Ortho'}">
    <c:set var="counter" value="0"/>
  <c:forEach var="cdcountt" items="${
      cdcount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Orthodontics</td><td align="
        center"> ${cdcountt}</td>
    <c:set var="total" value="${total+
        cdcountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${fpdcountt} </
        td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${cdcountt}% </
        td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${resto == 'Resto'}">
    <c:set var="counter" value="0"/>
  <c:forEach var="restocountt" items="${
      restocount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Restorative Dentistry</td><
        td align="center"> ${
        restocountt}</td>
    <c:set var="total" value="${total+
        restocountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${fpdcountt} </
        td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${restocountt}%
        </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${pedo == 'PEDO'}">
    <c:set var="counter" value="0"/>
  <c:forEach var="pedocountt" items="${
      pedocount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Pedodontics</td><td align="
        center"> ${pedocountt}</td>
    <c:set var="total" value="${total+
        pedocountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${pedocountt} </
        td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${pedocountt}%
```

```
            </td>
        </c:if>
       </c:forEach>
       </tr>
      </c:if>

      <c:if test="${rpd == 'RPD'}">
         <c:set var="counter" value="0"/>
       <c:forEach var="rpdcountt" items="${
            rpdcount}">
        <c:set var="counter" value="${counter
             +1}"/>
        <c:if test="${counter == 1}">
         <tr><td>Restorative Dentistry</td><
              td align="center"> ${rpdcountt
              }</td>
         <c:set var="total" value="${total+
              rpdcountt}"/>
        </c:if>
        <c:if test="${counter == 2 || counter
             == 4}">
         <td align="center"> ${rpdcountt} </
              td>
        </c:if>
        <c:if test="${counter == 3 || counter
             > 4}">
         <td align="center"> ${rpdcountt}% </
              td>
        </c:if>
       </c:forEach>
       </tr>
      </c:if>
    <tr><td><br/></td></tr>
      <tr><td>Total # of Cases</td> <td align
           ="center"> <b> ${total} </b> </td
           ></tr>
      </table>
      </div>
    </c:if>



    <c:if test="${(caries == 'yes' ||
         recurrentcaries == 'yes' ||
         restoration == 'yes' ||
     extrusion == 'yes' || intrusion == 'yes'
           || mesialdrift == 'yes' ||
          distaldrift == 'yes' ||
     completedenture == 'yes' ||
          singledenture == 'yes' ||
          removablepartial == 'yes' ||
          rotation == 'yes' || postcorecrown
         == 'yes' ||
     rootcanal == 'yes' || pitandfissure == '
          yes' || extracted == 'yes' ||
          missing == 'yes' || unerupted == '
          yes' ||
     impacted == 'yes' || porcelainfused == '
          yes' || acryliccrown == 'yes' ||
          metalcrown == 'yes' ||
          porcelaincrown == 'yes') && and1 ==
           'yes'}">


    <br/><br/>
     <b>
      <table class="boxHeader">
     <tr><td align="left" width="80%">
    Dental Condition (AND) </td></tr>
     </table>
     </b>
    <div class="box">
     <table id = "sample" cellspacing="2">
     <col width="200"><col width="75"><col
          width="75"><col width="75"><col
          width="75"><col width="75"><col
          width="100"><col width="100"><col
          width="100">
     <c:set var="total" value="0"/>
     <tr><td><b> </b></td><td align="center
          "><b># of Cases</b></td><td align="
          center"><b>Females</b></td><td
          align="center"><b>% Females</b></td
          ><td align="center"><b>Males</b></
          td><td align="center"><b>% Males</b
          ></td><td align="center"><b>%
          Females (over total females)</b></
          td><td align="center"><b>% Males (
          over total males)</b></td><td align
          ="center"><b>% Cases (over total
          patients)</b></td></tr>


     <tr> <td colspan="7">  <b>
      Results
      </b> </td> </tr>
     <c:set var="counter" value="0"/>
     <c:forEach var="item" items="${results
          }">
      <c:set var="counter" value="${counter
           +1}"/>
      <c:if test="${counter == 1}">
       <tr><td>   </td><td align="center"> ${
            item}</td>
      </c:if>
      <c:if test="${counter == 2 || counter
           == 4}">
       <td align="center"> ${item} </td>
      </c:if>
      <c:if test="${counter == 3 || counter >
           4}">
       <td align="center"> ${item}% </td>
      </c:if>
     </c:forEach>
```

```
 </table>
 </div>
</c:if>

<c:if test="${(caries == 'yes' ||
     recurrentcaries == 'yes' ||
     restoration == 'yes' ||
 extrusion == 'yes' || intrusion == 'yes'
      || mesialdrift == 'yes' ||
      distaldrift == 'yes' ||
 completedenture == 'yes' ||
      singledenture == 'yes' ||
      removablepartial == 'yes' ||
      rotation == 'yes' || postcorecrown
      == 'yes' ||
 rootcanal == 'yes' || pitandfissure == '
      yes' || extracted == 'yes' ||
      missing == 'yes' || unerupted == '
      yes' ||
 impacted == 'yes' || porcelainfused == '
      yes' || acryliccrown == 'yes' ||
      metalcrown == 'yes' ||
      porcelaincrown == 'yes') && or1 ==
      'yes'}">

<br/><br/>
   <b>
  <table class="boxHeader">
 <tr><td align="left" width="80%">
Dental Condition (OR) </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
  <col width="200"><col width="75"><col
       width="75"><col width="75"><col
       width="75"><col width="75"><col
       width="100"><col width="100"><col
       width="100">
  <c:set var="total" value="0"/>
  <tr><td><b>Condition</b></td><td align
       ="center"><b># of Cases</b></td><
       td align="center"><b>Females</b></
       td><td align="center"><b>% Females
       </b></td><td align="center"><b>
       Males</b></td><td align="center"><
       b>% Males</b></td><td align="
       center"><b>% Females (over total
       females)</b></td><td align="center
       "><b>% Males (over total males)</b
       ></td><td align="center"><b>%
       Cases (over total patients)</b></
       td></tr>
  <c:if test="${caries == 'yes'}">
   <c:set var="counter" value="0"/>
   <c:forEach var="cariescountt" items="$
       {cariescount}">
   <c:set var="counter" value="${counter
       +1}"/>

   <c:if test="${counter == 1}">
    <tr><td>Caries</td><td align="center
       "> ${cariescountt}</td>
    <c:set var="total" value="${total+
       cariescountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${cariescountt}
       </td>
   </c:if>
   <c:if test="${counter == 3 || counter
       > 4}">
    <td align="center"> ${cariescountt}%
       </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${recurrentcaries == 'yes
      '}">
  <c:set var="counter" value="0"/>
  <c:forEach var="recurrentcariescountt"
       items="${recurrentcariescount}">
   <c:set var="counter" value="${counter
       +1}"/>

   <c:if test="${counter == 1}">
    <tr><td>Recurrent Caries</td><td
       align="center"> ${
       recurrentcariescountt}</td>
    <c:set var="total" value="${total+
       recurrentcariescountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${
       recurrentcariescountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
       > 4}">
    <td align="center"> ${
       recurrentcariescountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
  <c:if test="${restoration == 'yes'}">
  <c:set var="counter" value="0"/>
  <c:forEach var="restorationcountt"
       items="${restorationcount}">
   <c:set var="counter" value="${counter
```

```
      +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>restoration</td><td align="
        center"> ${restorationcountt}</
        td>
   <c:set var="total" value="${total+
        restorationcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        restorationcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        restorationcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>

<c:if test="${extrusion == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="extrusioncountt" items
      ="${extrusioncount}">
  <c:set var="counter" value="${counter
       +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Extrusion</td><td align="
        center"> ${extrusioncountt}</td
        >
   <c:set var="total" value="${total+
        extrusioncountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        extrusioncountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        extrusioncountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${intrusion == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="intrusioncountt" items
      ="${intrusioncount}">

  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Intrusion</td><td align="
        center"> ${intrusioncountt}</td
        >
   <c:set var="total" value="${total+
        intrusioncountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        intrusioncountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        intrusioncountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${mesialdrift == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="mesialdriftcountt"
      items="${mesialdriftcount}">
  <c:set var="counter" value="${counter
       +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Mesial Drift</td><td align="
        center"> ${mesialdriftcountt}</
        td>
   <c:set var="total" value="${total+
        mesialdriftcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        mesialdriftcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        mesialdriftcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${distaldrift == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="distaldriftcountt"
      items="${distaldriftcount}">
  <c:if test="${counter == 1}">
   <tr><td>Distal Drift</td><td align="
        center"> ${distaldriftcountt}</
        td>
```

```
    <c:set var="total" value="${total+
        distaldriftcountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        distaldriftcountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        distaldriftcountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${completedenture == 'yes
     '}">
  <c:set var="counter" value="0"/>
  <c:forEach var="completedenturecountt"
       items="${completedenturecount}">
   <c:set var="counter" value="${counter
       +1}"/>

   <c:if test="${counter == 1}">
    <tr><td>Complete Denture</td><td
        align="center"> ${
        completedenturecountt}</td>
    <c:set var="total" value="${total+
        completedenturecountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        completedenturecountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        completedenturecountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${singledenture == 'yes'}">
  <c:set var="counter" value="0"/>
  <c:forEach var="singledenturecountt"
      items="${singledenturecount}">
   <c:set var="counter" value="${counter
       +1}"/>

   <c:if test="${counter == 1}">
    <tr><td>Single Denture</td><td align
        ="center"> ${
        singledenturecountt}</td>
    <c:set var="total" value="${total+
        singledenturecountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        singledenturecountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        singledenturecountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${removablepartial == 'yes
     '}">
  <c:set var="counter" value="0"/>
  <c:forEach var="removablepartialcountt
      " items="${removablepartialcount
      }">
   <c:set var="counter" value="${counter
       +1}"/>
    <c:set var="total" value="${total+
        removablepartialcountt}"/>

   <c:if test="${counter == 1}">
    <tr><td>Removable Partial Denture</
        td><td align="center"> ${
        removablepartialcountt}</td>
    <c:set var="total" value="${total+
        removablepartialcountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        removablepartialcountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        removablepartialcountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${rotation == 'yes'}">
  <c:set var="counter" value="0"/>
  <c:forEach var="rotationcountt" items
      ="${rotationcount}">
   <c:set var="counter" value="${counter
       +1}"/>

   <c:if test="${counter == 1}">
    <tr><td>Rotation</td><td align="
        center"> ${rotationcountt}</td>
```

```
   <c:set var="total" value="${total+
        rotationcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${rotationcountt
        } </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${rotationcountt
        }% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${postcorecrown == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="postcorecrowncountt"
      items="${postcorecrowncount}">
  <c:set var="counter" value="${counter
       +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Post Core Crown</td><td
        align="center"> ${
        postcorecrowncountt}</td>
   <c:set var="total" value="${total+
        postcorecrowncountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        postcorecrowncountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        postcorecrowncountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${rootcanal == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="rootcanalcountt" items
      ="${rootcanalcount}">
  <c:set var="counter" value="${counter
       +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Root Canal</td><td align="
        center"> ${rootcanalcountt}</td
        >
   <c:set var="total" value="${total+
        rootcanalcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        rootcanalcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        rootcanalcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${pitandfissure == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="pitandfissurecountt"
      items="${pitandfissurecount}">
  <c:set var="counter" value="${counter
       +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Pit and Fissure</td><td
        align="center"> ${
        pitandfissurecountt}</td>
   <c:set var="total" value="${total+
        pitandfissurecountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        pitandfissurecountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        pitandfissurecountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${extracted == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="extractedcountt" items
      ="${extractedcount}">
  <c:set var="counter" value="${counter
       +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Extracted</td><td align="
        center"> ${extractedcountt}</td
        >
   <c:set var="total" value="${total+
        extractedcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
```

```
        == 4}">
    <td align="center"> ${
        extractedcountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        extractedcountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
</c:if>
<c:if test="${missing == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="missingcountt" items="
     ${missingcount}">
  <c:set var="counter" value="${counter
      +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Missing</td><td align="
        center"> ${missingcountt}</td>
   <c:set var="total" value="${total+
        missingcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${missingcountt}
        </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${missingcountt
        }% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${unerupted == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="uneruptedcountt" items
     ="${uneruptedcount}">
  <c:set var="counter" value="${counter
      +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Unerupted</td><td align="
        center"> ${uneruptedcountt}</td
        >
   <c:set var="total" value="${total+
        uneruptedcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        uneruptedcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${
        uneruptedcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${impacted == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="impactedcountt" items
     ="${impactedcount}">
  <c:set var="counter" value="${counter
      +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Impacted</td><td align="
        center"> ${impactedcountt}</td>
   <c:set var="total" value="${total+
        impactedcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${impactedcountt
        } </td>
  </c:if>
  <c:if test="${counter == 3 || counter
        > 4}">
   <td align="center"> ${impactedcountt
        }% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${porcelainfused == 'yes
    '}">
 <c:set var="counter" value="0"/>
 <c:forEach var="porcelainfusedcountt"
     items="${porcelainfusedcount}">
  <c:set var="counter" value="${counter
      +1}"/>

  <c:if test="${counter == 1}">
   <tr><td>Porcelain Fused</td><td
        align="center"> ${
        porcelainfusedcountt}</td>
   <c:set var="total" value="${total+
        pocelainfusedcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
        == 4}">
   <td align="center"> ${
        porcelainfusedcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
```

```
        > 4}">
    <td align="center"> ${
        porcelainfusedcountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${acryliccrown == 'yes'}">
  <c:set var="counter" value="0"/>
  <c:forEach var="acryliccrowncountt"
      items="${acryliccrowncount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Acrylic Crown</td><td align
        ="center"> ${acryliccrowncountt
        }</td>
    <c:set var="total" value="${total+
        acryliccrowncountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${
        acryliccrowncountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
       > 4}">
    <td align="center"> ${
        acryliccrowncountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${metalcrown == 'yes'}">
  <c:set var="counter" value="0"/>
  <c:forEach var="metalcrowncountt"
      items="${metalcrowncount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Metal Crown</td><td align="
        center"> ${metalcrowncountt}</
        td>
    <c:set var="total" value="${total+
        metalcrowncountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${
        metalcrowncountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
       > 4}">
    <td align="center"> ${
        metalcrowncountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${porcelaincrown == 'yes
     '}">
  <c:set var="counter" value="0"/>
  <c:forEach var="porcelaincrowncountt"
      items="${porcelaincrowncount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Porcelain Crown</td><td
        align="center"> ${
        porcelaincrowncountt}</td>
    <c:set var="total" value="${total+
        pocelaincrowncountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
       == 4}">
    <td align="center"> ${
        porcelaincrowncountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
       > 4}">
    <td align="center"> ${
        porcelaincrowncountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>


 <tr><td><br/></td></tr>
 <tr><td>Total # of Cases</td> <td align
     ="center"> <b> ${total} </b> </td
     ></tr>
 <tr><td><br/></td></tr>
 </table>
 </div>
</c:if>

<c:if test="${(periodontics == 'yes' ||
    class1 == 'yes' || class2 == 'yes'
    || class3 == 'yes' || class4 == 'yes
    ' ||
 class5 == 'yes' || onlay == 'yes' ||
     extraction == 'yes' || odontectomy
    == 'yes' || specialcase == 'yes' ||
 pulpsedation == 'yes' ||
     tempfillingservice == 'yes' ||
     laminated == 'yes' || singlecrown
    == 'yes' || bridgeservice == 'yes'
    ||
 anterior == 'yes' || posterior == 'yes'
     || pedodontics == 'yes' ||
     orthodontics == 'yes' ||
```

```
      acuteinfections == 'yes' ||
 traumaticinjuries == 'yes' ||
      completedentservice == 'yes' ||
      singledentservice == 'yes' ||
      removablepartialservice == 'yes')
 && and2 == 'yes'}">
<br/> <br/>
 <b>
  <table class="boxHeader">
 <tr><td align="left" width="80%">
Services Needed (AND) </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
 <col width="200"><col width="75"><col
      width="75"><col width="75"><col
      width="75"><col width="75"><col
      width="100"><col width="100"><col
      width="100">
 <c:set var="total" value="0"/>
 <tr><td><b> </b></td><td align="center
     "><b># of Cases</b></td><td align="
      center"><b>Females</b></td><td
      align="center"><b>% Females</b></td
      ><td align="center"><b>Males</b></
      td><td align="center"><b>% Males</b
      ></td><td align="center"><b>%
      Females (over total females)</b></
      td><td align="center"><b>% Males (
      over total males)</b></td><td align
      ="center"><b>% Cases (over total
      patients)</b></td></tr>

 <tr> <td colspan="7">  <b>
  Results
  </b> </td> </tr>
 <c:set var="counter" value="0"/>
 <c:forEach var="item" items="${results2
     }">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>  </td><td align="center"> ${
       item}</td>
  </c:if>
  <c:if test="${counter == 2 || counter
      == 4}">
   <td align="center"> ${item} </td>
  </c:if>
  <c:if test="${counter == 3 || counter >
       4}">
   <td align="center"> ${item}% </td>
  </c:if>
 </c:forEach>
 </table>
 </div>
 </c:if>


<c:if test="${(periodontics == 'yes' ||
     class1 == 'yes' || class2 == 'yes'
     || class3 == 'yes' || class4 == 'yes
     ' ||
  class5 == 'yes' || onlay == 'yes' ||
      extraction == 'yes' || odontectomy
      == 'yes' || specialcase == 'yes' ||
  pulpsedation == 'yes' ||
      tempfillingservice == 'yes' ||
      laminated == 'yes' || singlecrown
      == 'yes' || bridgeservice == 'yes'
      ||
  anterior == 'yes' || posterior == 'yes'
      || pedodontics == 'yes' ||
      orthodontics == 'yes' ||
      acuteinfections == 'yes' ||
  traumaticinjuries == 'yes' ||
      completedentservice == 'yes' ||
      singledentservice == 'yes' ||
      removablepartialservice == 'yes')
 && or2 == 'yes'}">

<br/><br/>
  <b>
  <table class="boxHeader">
 <tr><td align="left" width="80%">
Services Needed (OR) </td></tr>
 </table>
 </b>
<div class="box">
 <table id = "sample" cellspacing="2">
  <col width="200"><col width="75"><col
      width="75"><col width="75"><col
      width="75"><col width="75"><col
      width="100"><col width="100"><col
      width="100">
  <tr><td><b>Condition</b></td><td align
      ="center"><b># of Cases</b></td><
      td align="center"><b>Females</b></
      td><td align="center"><b>% Females
      </b></td><td align="center"><b>
      Males</b></td><td align="center"><
      b>% Males</b></td><td align="
      center"><b>% Females (over total
      females)</b></td><td align="center
      "><b>% Males (over total males)</b
      ></td><td align="center"><b>%
      Cases (over total patients)</b></
      td></tr>
  <c:set var="total" value="0"/>
```

```
<c:if test="${class1 == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="class1countt" items="$
      {class1count}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Class 1</td><td align="
        center"> ${class1countt}</td>
   <c:set var="total" value="${total+
        class1countt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${class1countt}
        </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${class1countt}%
        </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${class2 == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="class2countt" items="$
      {class2count}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Class 2</td><td align="
        center"> ${class2countt}</td>
   <c:set var="total" value="${total+
        class2countt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${class2countt}
        </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${class2countt}%
        </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${class3 == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="class3countt" items="$
      {class3count}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Class 3</td><td align="
        center"> ${class3countt}</td>
   <c:set var="total" value="${total+
        class3countt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${class3countt}
        </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${class3countt}%
        </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${class4 == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="class4countt" items="$
      {class4count}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Class 4</td><td align="
        center"> ${class4countt}</td>
   <c:set var="total" value="${total+
        class4countt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${class4countt}
        </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${class4countt}%
        </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${class5 == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="class5countt" items="$
      {class5count}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Class 5</td><td align="
        center"> ${class5countt}</td>
   <c:set var="total" value="${total+
        class5countt}"/>
  </c:if>
```

```
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${class5countt}
       </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${class5countt}%
        </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${onlay == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="onlaycountt" items="${
     onlaycount}">
  <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Onlay</td><td align="center
       "> ${onlaycountt}</td>
   <c:set var="total" value="${total+
       onlaycountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${onlaycountt}
       </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${onlaycountt}%
       </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${extraction == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="extractioncountt"
     items="${extractioncount}">
  <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Extraction</td><td align="
       center"> ${extractioncountt}</
       td>
   <c:set var="total" value="${total+
       extractioncountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
       extractioncountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
       extractioncountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${odontectomy == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="odontectomycountt"
     items="${odontectomycount}">
  <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Odontectomy</td><td align="
       center"> ${odontectomycountt}</
       td>
   <c:set var="total" value="${total+
       odontectomycountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
       odontectomycountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
       odontectomycountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${specialcase == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="specialcasecountt"
     items="${specialcasecount}">
  <c:set var="counter" value="${counter
      +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Special Case</td><td align="
       center"> ${specialcasecountt}</
       td>
   <c:set var="total" value="${total+
       specialcasecountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
       specialcasecountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
       specialcasecountt}% </td>
```

```
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${pedodontics == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="pedodonticscountt"
      items="${pedodonticscount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Pedodontics</td><td align="
        center"> ${pedodonticscountt}</
        td>
   <c:set var="total" value="${total+
        pedodonticscountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        pedodonticscountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        pedodonticscountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${orthodontics == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="orthodonticscountt"
      items="${orthodonticscount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Orthodontics</td><td align="
        center"> ${orthodonticscountt
        }</td>
   <c:set var="total" value="${total+
        orthodonticscountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        orthodonticscountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        orthodonticscountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${pulpsedation == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="pulpsedationcountt"
      items="${pulpsedationcount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Pulp Sedation</td><td align
        ="center"> ${pulpsedationcountt
        }</td>
   <c:set var="total" value="${total+
        pulpsedationcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        pulpsedationcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        pulpsedationcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${crownrecementation == '
     yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="
      crownrecementationcountt" items="
      ${crownrecementationcount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Crown Recementation</td><td
        align="center"> ${
        crownrecementationcountt}</td>
   <c:set var="total" value="${total+
        crownrecementationcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        crownrecementationcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        crownrecementationcountt}% </td
        >
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${tempfillingservice == '
```

```
    yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="
      tempfillingservicecountt" items="
      ${tempfillingservicecount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Temporary Filling</td><td
        align="center"> ${
        tempfillingservicecountt}</td>
   <c:set var="total" value="${total+
        tempfillingservicecountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        tempfillingservicecountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        tempfillingservicecountt}% </td
        >
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${acuteinfections == 'yes
     '}">
 <c:set var="counter" value="0"/>
 <c:forEach var="acuteinfectionscountt"
       items="${acuteinfectionscount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Acute Infections</td><td
        align="center"> ${
        acuteinfectionscountt}</td>
   <c:set var="total" value="${total+
        acuteinfectionscountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        acuteinfectionscountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        acuteinfectionscountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${traumaticinjuries == 'yes
     '}">
 <c:set var="counter" value="0"/>
 <c:forEach var="
      traumaticinjuriescountt" items="$
      {traumaticinjuriescount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Traumatic Injuries</td><td
        align="center"> ${
        traumaticinjuriescountt}</td>
   <c:set var="total" value="${total+
        traumaticinjuriescountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        traumaticinjuriescountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        traumaticinjuriescountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${laminated == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="laminatedcountt" items
      ="${laminatedcount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Laminated</td><td align="
        center"> ${laminatedcountt}</td
        >
   <c:set var="total" value="${total+
        laminatedcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        laminatedcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        laminatedcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${singlecrown == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="singlecrowncountt"
```

```
      items="${singlecrowncount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Single Crown</td><td align="
        center"> ${singlecrowncountt}</
        td>
   <c:set var="total" value="${total+
        singlecrowncountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        singlecrowncountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        singlecrowncountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${bridgeservice == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="bridgeservicecountt"
      items="${bridgeservicecount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Bridge Service</td><td align
        ="center"> ${
        bridgeservicecountt}</td>
   <c:set var="total" value="${total+
        bridgeservicecountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        bridgeservicecountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        bridgeservicecountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${anterior == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="anteriorcountt" items
      ="${anteriorcount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Anterior</td><td align="
        center"> ${anteriorcountt}</td>
   <c:set var="total" value="${total+
        anteriorcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${anteriorcountt
        } </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${anteriorcountt
        }% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${posterior == 'yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="posteriorcountt" items
      ="${posteriorcount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Posterior</td><td align="
        center"> ${posteriorcountt}</td
        >
   <c:set var="total" value="${total+
        posteriorcountt}"/>
  </c:if>
  <c:if test="${counter == 2 || counter
       == 4}">
   <td align="center"> ${
        posteriorcountt} </td>
  </c:if>
  <c:if test="${counter == 3 || counter
       > 4}">
   <td align="center"> ${
        posteriorcountt}% </td>
  </c:if>
 </c:forEach>
 </tr>
</c:if>
<c:if test="${completedentservice == '
     yes'}">
 <c:set var="counter" value="0"/>
 <c:forEach var="
      completedentservicecountt" items
      ="${completedentservicecount}">
  <c:set var="counter" value="${counter
       +1}"/>
  <c:if test="${counter == 1}">
   <tr><td>Complete Denture</td><td
        align="center"> ${
        completedentservicecountt}</td>
```

```
    <c:set var="total" value="${total+
        completedentservicecountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        completedentservicecountt} </td
        >
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        completedentservicecountt}% </
        td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${singledentservice == 'yes
     '}">
  <c:set var="counter" value="0"/>
  <c:forEach var="
      singledentservicecountt" items="$
      {singledentservicecount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Single Denture</td><td align
        ="center"> ${
        singledentservicecountt}</td>
    <c:set var="total" value="${total+
        singledentservicecountt}"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        singledentservicecountt} </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        singledentservicecountt}% </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <c:if test="${removablepartialservice
     == 'yes'}">
  <c:set var="counter" value="0"/>
  <c:forEach var="
      removablepartialservicecountt"
      items="${
      removablepartialservicecount}">
   <c:set var="counter" value="${counter
       +1}"/>
   <c:if test="${counter == 1}">
    <tr><td>Removable Partial Denture</
        td><td align="center"> ${
        removablepartialservicecountt
        }</td>
    <c:set var="total" value="${total+
        removablepartialservicecountt
        }"/>
   </c:if>
   <c:if test="${counter == 2 || counter
        == 4}">
    <td align="center"> ${
        removablepartialservicecountt}
        </td>
   </c:if>
   <c:if test="${counter == 3 || counter
        > 4}">
    <td align="center"> ${
        removablepartialservicecountt}%
         </td>
   </c:if>
  </c:forEach>
  </tr>
 </c:if>
 <tr><td><br/></td></tr>
 <tr><td>Total # of Cases</td> <td align
     ="center"> <b> ${total} </b> </td
     ></tr>
  </table>
 </div>
</c:if>

<br/>
<a href="${viewQueryURL}"> Go Back </a>

<br/>
</div>
</body>
</html>
```

Listing 64: userForm

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
```

```
<html>

<head>

<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">

<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>

<title>Manage Users</title>
<script>

$(document).ready(function(){
 $('#userSubmit').click(function(){
  userValidate();
 });

 $('#username').bind('change', function()
     {
  var username = $(this).val();
  var allUsername= '${allUsernames}'.
      split(',');
  //alert(username);
  isUsernameExist(allUsername,username);

 });


});


function isUsernameExist(allUsername,
    username){
 var userBool=true;
 for(i=0;i<allUsername.length;i++){
  if(username==allUsername[i]){
   $("#userExist").show();
   userBool=false;
  }
 }
 if(userBool)
  $("#userExist").hide();

}

function userValidate(){

 //fields
 var fname_user= document.getElementById
     ('fname_user').value;
 var minit_user= document.getElementById
     ('minit_user').value;
 var lname_user= document.getElementById
     ('lname_user').value;
 var username= document.getElementById('
     username').value;
 var password= document.getElementById('
     password').value;
 var repassword= document.getElementById
     ('repassword').value;
 var email= document.getElementById('
     email').value;
 var secret_answer= document.
     getElementById('secret_answer').
     value;

 var userConfirm="";

 //validate if password entered is
     correct
 if(password!=repassword)
  userConfirm+="Password didn't match. Re
      -enter password.\n";

 if(!isValidEmailAddress(email))
  userConfirm+="Email address is invalid.
       Re-enter email address.\n";

 //check for missing input
 if(fname_user=="" || lname_user=="" ||
     username=="" || password=="" ||
     secret_answer=="")
  userConfirm+="Required information is
      incomplete. Please check if all
      information needed is filled.\n";

 if(userConfirm==""){
   $('#userform').submit();
 }
 else
  alert(userConfirm);
}

//validate if input is really an emailAdd
function isValidEmailAddress(email) {
    var pattern = new RegExp(/^(("[\w-+\s
        ]+")|([\w-+]+(?:\.[\w-+]+)*)
        |("[\w-+\s]+")([\w-+]+(?:\.[\w
        -+]+)*))(@((?:[\w-+]+\.)*\w[\w
        -+]{0,66})\.([a-z]{2,6}(?:\.[a-z
        ]{2})?)$)|(@
        \[?((25[0-5]\.|2[0-4][\d]\.|1[\d
        ]{2}\.|[\d]{1,2}\.))
        ((25[0-5]|2[0-4][\d]|1[\d]{2}|[\
        d]{1,2})\.){2}(25[0-5]|2[0-4][\d
        ]|1[\d]{2}|[\d]{1,2})\]?$)/i);
    return pattern.test(email);
}
```

```
</script>



</head>

<body>
<c:url var="cancelURL" value="/forms/
    dentist/login/index.html" />

<div class="form-body">
<div class="sub-body-title">Add User</div
    >

<form:form action="useroutput.html"
    commandName="user" id="userform">

<table class="form-table">

<tr>

<td>Firstname :</td>

<td><form:input path="fname_user" size
    ="30"/></td>

</tr>



<tr>

<td>MI :</td>

<td><form:input path="minit_user" size
    ="10"/></td>

</tr>

<tr>

<td>Lastname :</td>

<td><form:input path="lname_user" size
    ="30"/></td>

</tr>

<tr>

<td>Username :</td>

<td><form:input path="username" size
    ="30"/>
<span class=validateTextbox id="userExist
    ">Username already exists!</span></
    td>

</tr>

<tr>

<td>Password :</td>

<td><form:password path="password" size
    ="30"/></td>

</tr>

<tr>

<td>Confirm Password :</td>

<td><input type="password" id="repassword
    " size="30"/></td>


</tr>

<tr>

<td>Email :</td>

<td><form:input path="email" size="40"
    /></td>

</tr>

<tr>

<td>Secret Question : </td>

<td>  <form:select path="secret_question
    ">
        <form:option value="${selected}"
            >Choose</form:option>
        <c:forEach var="secret_question"
            items="${secret_questions}">
        <c:if test="${secret_question !=
            selected}">
            <form:option value="${
                secret_question}">${
                secret_question}</form:
                option>
        </c:if>
        </c:forEach>
    </form:select>

</td>
</tr>

<tr>
```

```
<td>Secret Answer :</td>

<td><form:input path="secret_answer" size
    ="30"/></td>

</tr>

</table>


<div class="submit-buttons">
 <input type="button" value="Add User" id
     ="userSubmit" class="classname"/>
<input type="button" value="Cancel"
    onClick="window.location='${
    cancelURL}'" class="classname"/>

</div>
</form:form>
</div>
</body>

</html>
```

Listing 65: viewAllRoles

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>
<title>View Roles</title>
<script>

$(document).ready(function(){
 $('#deleteSubmit').click(function(){
  deleteValidate();
 });

});

function deleteValidate(){
  var r=confirm("Some users are dependent
       to this role. Role for these
      users will be reset to none. Are
      you sure to delete this role?");
  if (r==true)
   $('#deleteForm').submit();

}

</script>
</head>
<body>
<div class="list-body">
<div class="sub-body-title">View Role
    List</div>
<form:form action="deleteRoles" method="
    POST" commandName="deleteFunction"
    id="deleteForm">
<div class="searchBox">
<h3>Delete Roles: </h3>
<input type="button" value="Delete Role"
    id="deleteSubmit" class="classname
    "/>
</div>
<div class="clear"></div>
<hr>
<div class="clear"></div>

<table class="list-table" width="90%">
<thead>
<tr class="head-list-table">
<td width="40%">Name :</td>
<td width="25%">Database Name :</td>
<td width="25%"> Created Date :</td>
</tr>
</thead>
<tbody>

  <c:forEach items="${roleLists}" var="
      roleList" varStatus="loop">
<c:choose>
     <c:when test='${(loop.index)%2 eq
         0}'>
      <c:set var="rowColor" value="even"
          scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
          scope="page"/>
    </c:otherwise>
  </c:choose>
```

```
<tr class="${rowColor}">


 <td width="40%">
        <form:checkbox path="deleteObject
             " value="${roleList.role_id
             }"/> <a href="editRoleForm.
             html?role_id=${roleList.
             role_id}"> ${roleList.
             role_name} </a>
    </td>
 <td width="25%">
          ${roleList.database_role}
    </td>
    <td width="25%">
                ${roleList.created_date}
    </td>

 </tr>
  </c:forEach>
<c:if test=" ${empty roleLists}">
<tr><td colspan="3">No roles</td></tr>
</c:if>

</tbody>
</table>
</form:form>
</div>
</body>
</html>
```

Listing 66: viewAllSections

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
     pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>
<title>View Sections</title>
<script>

$(document).ready(function(){
 $('#deleteSubmit').click(function(){
  deleteValidate();
 });

});

function deleteValidate(){
 var r=confirm("Some roles are part of
      this section. Are you sure to
      delete this role?");
  if (r==true)
   $('#deleteForm').submit();
}

</script>
</head>
<body>
<div class="list-body">
<div class="sub-body-title">View Users
     List</div>
<form:form action="deleteSections" method
     ="POST" commandName="deleteFunction"
      id="deleteForm">
<div class="searchBox">
<h3>Delete Sections: </h3>
<input type="button" value="Delete
     Section" id="deleteSubmit" class="
     classname"/>
</div>
<div class="clear"></div>
<hr>
<div class="clear"></div>

<table class="list-table" width="90%">
<thead>
<tr class="head-list-table">
<td  width="40%">Name :</td>
<td  width="25%">Created By :</td>
<td  width="25%">Created Date :</td>
</tr>
</thead>
<tbody>

  <c:forEach items="${sectionLists}" var
       ="sectionList" varStatus="loop">
 <c:choose>
     <c:when test='${(loop.index)%2 eq
          0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
```

```
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">



 <td width="40%">
       <form:checkbox path="deleteObject
            " value="${sectionList.
            section_id}"/> <a href="
            editSectionForm.html?
            section_id=${sectionList.
            section_id}"> ${sectionList.
            section_name} </a>
    </td>
 <td width="25%">
            ${sectionList.created_by}
    </td>
    <td width="25%">
            ${sectionList.created_date}
    </td>
 </tr>
  </c:forEach>
<c:if test="${empty sectionLists}">

      <tr>
      <td colspan="3" align="center">
           Currently No Section</td>
      </tr>
</c:if>

</tbody>
</table>
</form:form>
</div>
</body>
</html>
```

Listing 67: viewAllUsers

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
     springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<script src="//ajax.googleapis.com/ajax/
     libs/jquery/1.8.3/jquery.min.js" >
</script>
<script>

$(document).ready(function(){
 $('#deleteSubmit').click(function(){
  deleteValidate();
 });

 $('#searchSubmit').click(function(){

  $('#searchForm').submit();
 });
});



function deleteValidate(){
  var r=confirm("Are you sure to delete
       this user?");
  if (r==true)
   $('#deleteForm').submit();

}
</script>
<title>View Users</title>
</head>
<body>
<c:url var="searchURL" value="/forms/
     dentist/simpleForm/
     viewSpecificedUsers.html" />

<div class="list-body">
<div class="sub-body-title">View Users
     List</div>
<form:form action="${searchURL}" method='
     GET' modelAttribute="user" id="
     searchForm">

<div class="searchBox">
<h3>Search Users: </h3>
<form:input path="username" />
<input type="button" value="Search" id="
     searchSubmit" class="classname"/>
```

```
</div>
</form:form>
<form:form action="deleteUsers" method="
    POST" commandName="deleteFunction"
    id="deleteForm">
<div class="deleteSearchBox">
<h3>Delete Users: </h3>
<input type="button" value="Delete User"
    id="deleteSubmit" class="classname
    "/>
</div>
<div class="clear"></div>
<hr>
<div class="clear"></div>

<table class="list-table"  width="90%">
<thead>
<tr class="head-list-table">
<td  width="40%">Name :</td>
<td  width="25%">Username :</td>
<td  width="25%">Created Date :</td>
</tr>
</thead>
<tbody>
  <c:forEach items="${userLists}" var="
      userList" varStatus="loop">
 <c:choose>
    <c:when test='${(loop.index)%2 eq
        0}'>
      <c:set var="rowColor" value="even"
          scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
          scope="page"/>
    </c:otherwise>
  </c:choose>


<tr class="${rowColor}">


 <td  width="40%">
       <form:checkbox path="deleteObject
           " value="${userList.user_id
           }"/> <a href="editUserForm.
           html?user_id=${userList.
           user_id}"> ${userList.
           fname_user}  ${userList.
           minit_user}  ${userList.
           lname_user} </a>
    </td>
 <td  width="25%">
        ${userList.username}
    </td>
    <td  width="25%">
        ${userList.created_date}
    </td>
 </tr>
  </c:forEach>
<c:if test=" ${empty userLists}">
<tr><td colspan="3">No users</td></tr>
</c:if>

</tbody>
</table>

</form:form>
</div>
</body>
</html>
```

Listing 68: viewFacultySection

```
<%@ page language="java" contentType="
    text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
    jstl/core" prefix="c" %>
<%@taglib prefix="form" uri="http://www.
    springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN" "http://www.
    w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
    text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script src="//ajax.googleapis.com/ajax/
    libs/jquery/1.8.3/jquery.min.js" >
</script>
<script>

$(document).ready(function(){

 $('#searchSubmit').click(function(){

  $('#searchForm').submit();

 });
});

</script>
</head>
<body>
<div class="list-body">
```

```
<div class="sub-body-title">View Users
    List</div>

<h3>${sessionScope.sessionName}</h3>
<b>Username:</b> ${sessionScope.
    sessionUser}
<br/>
<br/>
<div class="clear"></div>
<table>
<tr>
<td><b>Search:</b></td>
<td><b>Roles:</b></td>
<td><b>Sections:</b></td>
</tr>
<tr>
<c:url var="searchURL" value="/forms/
    dentist/simpleForm/
    viewSpecificFacultySection.html" />

<td><form:form action="${searchURL}"
    method='GET' modelAttribute="
    userRoleSection" id="searchForm">


<form:input path="user_name" size="30"/>

<form:select path="section_id">


          <c:forEach items="${
              currentSectionLists}" var
              ="currentSectionList"
              varStatus="loop">
          <c:if test="${
              currentSectionList.
              section_id != selected}">
        <form:option value="${
            currentSectionList.
            section_id}" >${
            currentSectionList.
            section_name}</form:option>
      </c:if>
        </c:forEach>



    </form:select>
  <br/>

<form:select path="user_role">
          <form:option value="all" >ALL
              </form:option>
           <form:option value="student"
               >Student Clinician</form
               :option>
        <form:option value="faculty" >
            Faculty Clinician</form:
            option>

    </form:select>
    <input type="button" value="Search"
        id="searchSubmit" class="
        classname"/>



</form:form>
</td>
<td>
  <c:forEach items="${sessionScope.
      currentRoleList}" var="currentRole
      " varStatus="loop">
  ${currentRole}
  <br/>
</c:forEach>
</td>
<td> <c:forEach items="${
    currentSectionLists}" var="
    currentSectionList" varStatus="loop
    ">

${currentSectionList.section_name}
<br/>
        </c:forEach></td>
</tr>
</table>
<hr/>

<table class="list-table" width="78%">
<thead>
<tr class="head-list-table">
<td width="26%">Name :</td>
<td width="26%">Username :</td>
<td width="26%">Clinician Role :</td>
</tr>
</thead>
<tbody>

  <c:forEach items="${currentLists}" var
      ="currentPick" varStatus="loop">
 <c:choose>
    <c:when test='${(loop.index)%2 eq
        0}'>
      <c:set var="rowColor" value="even"
          scope="page"/>
    </c:when>
    <c:otherwise>
      <c:set var="rowColor" value="odd"
          scope="page"/>
    </c:otherwise>
  </c:choose>
```

```
<tr class="${rowColor}">




 <td width="26%">
  ${currentPick.full_name}
    </td>
 <td width="26%">
   ${currentPick.user_name}
     </td>
     <td width="26%">
      ${currentPick.user_role}
     </td>
    </tr>
     </c:forEach>

</tbody>
</table>
</div>
</body>
</html>
```

Listing 69: viewUserRole

```
<%@ page language="java" contentType="
     text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/
     jstl/core" prefix="c" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
     4.01 Transitional//EN" "http://www.
     w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="
     text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div class="tab-body">

<c:url var="editRoleURL" value="/forms/
     dentist/simpleForm/editRoleForm?
     role_id=${role.role_id}" />
<c:url var="viewUserRoleURL" value="/
     forms/dentist/simpleForm/
     viewUserRole?role_id=${role.role_id
     }" />

<ul class = "tabs primary">
    <li class = "active"><a href = "${
         viewUserRoleURL}" class = "
         active">View Users</a></li>
    <li><a href = "${editRoleURL}" >Edit
         Role</a></li>

</ul>
<table class="list-table">
<tr class="head-list-table">
<td colspan="3">List of Users</td>
</tr>
  <c:forEach items="${currentUserLists}"
       var="currentUser" varStatus="loop
       ">

<c:choose>
     <c:when test='${(loop.index)%2 eq
          0}'>
      <c:set var="rowColor" value="even"
           scope="page"/>
     </c:when>
     <c:otherwise>
      <c:set var="rowColor" value="odd"
           scope="page"/>
     </c:otherwise>
  </c:choose>


<tr class="${rowColor}">


 <td>
       ${currentUser.fname_user} ${
            currentUser.minit_user}  ${
            currentUser.lname_user}
    </td>
 <td>${currentUser.username}</td>
 <td>${currentUser.created_date}</td>
 </tr>
  </c:forEach>
</table>
</div>
</body>
</html>
```

Listing 70: AddUserRole

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE tiles-definitions PUBLIC
        "-//Apache Software Foundation//
             DTD Tiles Configuration 2.0//
             EN"
        "http://tiles.apache.org/dtds/
```

```
            tiles-config_2_0.dtd">
<tiles-definitions>
    <definition name="base.definition"
        template="/WEB-INF/views/layout.
            jsp">
        <put-attribute name="css" value="
            theme.css" />
        <put-attribute name="title" value
            ="DentISt" />
        <put-attribute name="header"
            value="/WEB-INF/views/header
            .jsp" />
        <put-attribute name="menu" value=
            "" />
        <put-attribute name="body" value=
            "" />
        <put-attribute name="footer"
            value="/WEB-INF/views/footer
            .jsp" />
    </definition>

    <definition name="loginForm" extends=
        "base.definition">
        <put-attribute name="body" value=
            "/WEB-INF/views/loginForm.
            jsp" />
    </definition>


  <definition name="index" extends="base.
      definition">
        <put-attribute name="body" value=
            "/WEB-INF/views/index.jsp" /
            >
    </definition>

    <definition name="Error" extends="
        base.definition">
        <put-attribute name="body" value=
            "/WEB-INF/views/Error.jsp" /
            >
    </definition>

    <!-- Profile USE CASE -->

  <definition name="myProfile" extends="
      base.definition">
      <put-attribute name="menu" value="/
          WEB-INF/views/profileMenu.jsp"
          />
        <put-attribute name="body" value=
            "/WEB-INF/views/myProfile.
            jsp" />
    </definition>

    <definition name="changePassword"
        extends="base.definition">
      <put-attribute name="menu" value="/
          WEB-INF/views/profileMenu.jsp"
          />
        <put-attribute name="body" value=
            "/WEB-INF/views/
            changePassword.jsp" />
    </definition>

    <definition name="changeSecret"
        extends="base.definition">
      <put-attribute name="menu" value="/
          WEB-INF/views/profileMenu.jsp"
          />
        <put-attribute name="body" value=
            "/WEB-INF/views/changeSecret
            .jsp" />
    </definition>


<!-- USER USE CASE -->

    <definition name="userForm" extends="
        base.definition">
        <put-attribute name="menu" value="/
            WEB-INF/views/userMenu.jsp" />
        <put-attribute name="body" value=
            "/WEB-INF/views/userForm.jsp
            " />
    </definition>

    <definition name="useroutput" extends
        ="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/userMenu.jsp
            " />
        <put-attribute name="body" value=
            "/WEB-INF/views/useroutput.
            jsp" />
    </definition>

    <definition name="viewAllUsers"
        extends="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/userMenu.jsp
            " />
        <put-attribute name="body" value=
            "/WEB-INF/views/viewAllUsers
            .jsp" />
    </definition>

    <definition name="editUserForm"
        extends="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/userMenu.jsp
            " />
```

```
    <put-attribute name="body" value=
        "/WEB-INF/views/editUserForm
        .jsp" />
</definition>

 <definition name="AddUserRole"
      extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/userMenu.jsp
        " />
    <put-attribute name="body" value=
        "/WEB-INF/views/AddUserRole.
        jsp" />
</definition>

 <definition name="UserRoleConfirm"
      extends="base.definition">
     <put-attribute name="menu" value
         ="/WEB-INF/views/userMenu.
         jsp" />
     <put-attribute name="body" value
         ="/WEB-INF/views/
         UserRoleConfirm.jsp" />
</definition>

<!-- Role USE CASE -->
  <definition name="roleForm" extends
      ="base.definition">
  <put-attribute name="menu" value="/
      WEB-INF/views/roleMenu.jsp" />
    <put-attribute name="body" value=
        "/WEB-INF/views/roleForm.jsp
        " />
</definition>

<definition name="roleoutput" extends
     ="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/roleMenu.jsp
        " />
    <put-attribute name="body" value=
        "/WEB-INF/views/roleoutput.
        jsp" />
</definition>

<definition name="viewAllRoles"
     extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/roleMenu.jsp
        " />
    <put-attribute name="body" value=
        "/WEB-INF/views/viewAllRoles
        .jsp" />
</definition>

 <definition name="editRoleForm"
      extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/roleMenu.jsp
        " />
    <put-attribute name="body" value=
        "/WEB-INF/views/editRoleForm
        .jsp" />
</definition>

  <definition name="addPrivilegeRole"
       extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/roleMenu.jsp
        " />
    <put-attribute name="body" value=
        "/WEB-INF/views/
        addPrivilegeRole.jsp" />
</definition>

   <definition name="viewUserRole"
       extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/roleMenu.jsp
        " />
    <put-attribute name="body" value=
        "/WEB-INF/views/viewUserRole
        .jsp" />
</definition>

    <!-- SECTION USE CASE -->
  <definition name="sectionForm"
       extends="base.definition">
  <put-attribute name="menu" value="/
      WEB-INF/views/sectionMenu.jsp"
       />
    <put-attribute name="body" value=
        "/WEB-INF/views/sectionForm.
        jsp" />
</definition>

<definition name="sectionoutput"
     extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/sectionMenu.
        jsp" />
    <put-attribute name="body" value=
        "/WEB-INF/views/
        sectionoutput.jsp" />
</definition>

<definition name="viewAllSections"
     extends="base.definition">
    <put-attribute name="menu" value=
        "/WEB-INF/views/sectionMenu.
        jsp" />
    <put-attribute name="body" value=
```

```
            "/WEB-INF/views/
            viewAllSections.jsp" />
    </definition>

     <definition name="editSectionForm"
          extends="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/sectionMenu.
            jsp" />
        <put-attribute name="body" value=
            "/WEB-INF/views/
            editSectionForm.jsp" />
    </definition>

      <definition name="addSectionRole"
           extends="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/sectionMenu.
            jsp" />
        <put-attribute name="body" value=
            "/WEB-INF/views/
            addSectionRole.jsp" />
    </definition>

     <!-- Login USE CASE -->
      <definition name="forgotPasswordUN"
           extends="base.definition">
        <put-attribute name="body" value=
            "/WEB-INF/views/
            forgotPasswordUN.jsp" />
    </definition>
    <definition name="success" extends="
         base.definition">
        <put-attribute name="body" value=
            "/WEB-INF/views/success.jsp"
             />
    </definition>

     <!-- Admin USE CASE -->
     <definition name="changeEmailAdmin"
          extends="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/
            configurationMenu.jsp" />
          <put-attribute name="body"
              value="/WEB-INF/views/
              changeEmailAdmin.jsp" />
    </definition>

     <definition name="auditTrail"
          extends="base.definition">
            <put-attribute name="body"
                value="/WEB-INF/views/
                auditTrail.jsp" />
        </definition>
    <definition name="dentalchart" extends
        ="base.definition">
            <put-attribute name="body"
                value="/WEB-INF/views/
                dentalchart.jsp" />
    </definition>

    <!-- OTHERS USE CASE -->

    <definition name="viewFacultySection"
          extends="base.definition">
        <put-attribute name="menu" value=
            "/WEB-INF/views/facultyMenu.
            jsp" />
        <put-attribute name="body" value=
            "/WEB-INF/views/
            viewFacultySection.jsp" />
    </definition>
       <definition name="queryPatients"
            extends="base.definition">
            <put-attribute name="body"
                value="/WEB-INF/views/
                queryPatients.jsp" />
    </definition>
    <definition name="searchResults"
         extends="base.definition">
            <put-attribute name="body"
                value="/WEB-INF/views/
                searchResults.jsp" />
    </definition>
    <definition name="statisticsform"
         extends="base.definition">
            <put-attribute name="body"
                value="/WEB-INF/views/
                statisticsform.jsp" />
    </definition>

    <definition name="statisticsResults"
         extends="base.definition">
            <put-attribute name="body"
                value="/WEB-INF/views/
                statisticsResults.jsp" />
    </definition>


</tiles-definitions>
```

Listing 71: dispatcher-servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.
    org/schema/beans"
```

```
xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance"

xmlns:p="http://www.springframework.org/
    schema/p"

xmlns:context="http://www.springframework
    .org/schema/context"

xmlns:mvc="http://www.springframework.org
    /schema/mvc"

xsi:schemaLocation="

http://www.springframework.org/schema/
    beans

http://www.springframework.org/schema/
    beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/
    context

http://www.springframework.org/schema/
    context/spring-context-3.0.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc
    /spring-mvc-3.0.xsd

">


 <bean class="org.springframework.web.
     servlet.mvc.support.
     ControllerClassNameHandlerMapping"
     >
   <property name="caseSensitive" value="
       true" />
   <property name="pathPrefix" value="/
       dentist" />
   </bean>
 <bean class="com.dentist.version.three.
     controller.SimpleFormController" />
 <bean class="com.dentist.version.three.
     controller.LoginController" />
 <bean class="com.dentist.version.three.
     controller.AdminController" />
 <bean class="com.dentist.version.three.
     controller.SearchController" />
 <bean class="com.dentist.version.three.
     controller.SearchResultController" /
     >
 <bean class="com.dentist.version.three.
     controller.StatisticsController" />


 <bean id="tilesConfigurer"
 class="org.springframework.web.servlet.
     view.tiles2.TilesConfigurer"
 p:definitions="/WEB-INF/tiles.xml">
 </bean>


 <bean id="tilesResolver" class="org.
     springframework.web.servlet.view.
     UrlBasedViewResolver">
     <property name="viewClass" value="
         org.springframework.web.servlet
         .view.tiles2.TilesView"/>

 </bean>

 </beans>
```

XII. Acknowledgement

When I first shifted in comsci, ang hirap magadjust nun. No block. No barkada. New people. New professors. New subjects to tackle on. Sometimes I wonder and contemplated back then if I had the right decision na magshift but at the end I decided to stay at itry kng ano man ang mapapala ko dito. Now after all those years masasabi ko April 2 is one of the best moments in my college life and I don't regret shifting ng course and FINALLY graduating. :)

Marami dapat ithanks hahaha. Pero syempre first things first, thank you Lord for never abandoning me. All those sweat and tears, hindi sayang. If it wasn't from my will at determination nakuha ko from listening to mass and prayers, I wouldn't even be here writing this. So for that thank you.

Syempre I cant forget to thank my adviser, Sir Chua! : Medyo stressful and pressuring pero if not for that, I don't think makakagraduate ako lolol. Thank po for the push and help to finish the SP. Thank you din po for trusting me and Jamie na matatapos namin to. Sobrang thank you po tlga Sir ang laki ng utang naming sa inyo hahaha *bows down*

To Dean Medina and Doc Charlie thank you for always accommodating us despite your busy scheds. Though we feel like traitors na former dent students kami, lmao I love my SP topic! Cause no matter what Dentistry is still close to my heart esp the people in it : Kahit ba isang taon lang ako sa dent, it will always be a special moment for me.

Speaking of Dent, makakalimutan ko ba ang mga dent friends ko dito? Mehehehe To Jorge who has constantly worried and cheering for us para matapos to and my guy bestfriend, to Yanie who I vent out my feelings when I feel pressured or just my worries on real life and cyber life, to Lili, Yves, Lyka, Yvette, Lei, Xtian and Lyka miss ko na kayo lahat! Thank you sa mga times together in college life. Meeting you people and having friend on a new chapter in life was one of the best. Masasabi ko I went through a lot with these people

na nandiyan sakin to listen when im down or sa mga kalokohan. Pag kayo kasama ko I feel like being a kid again. I love you guys forever and I hope magkikita tayo na BUO again soon.

Syempre andiyan din yng 2 biochemistry students na sobrang love na love ko ayiee. To Jerlene and Eloisa thank you din sa mga katawanan and mga kpop spazz na meron tayo. Lakas tawa pag tayo apat magkakasama and I feel so happy na I can show the dorky side of me with you guys. Miss ko na kayong dalawa. Thank you for the encouragement ang pagsasama. Isa kayo sa mga tao na nakatulong for me to adjust being without my block. Im so thankful. Thank you.

And now *drum roll* for the comsci peeps! Sa mga 09 who don't left us out kahit hindi kami blockmates. Though im your ate, I had fun with all the kulitan and laughs esp with the HI groups- Ven, Janelle, Allen, Pebbe, Maan and Rachelle. All those overnights para sa mga 127-128 projects hihi sana maulit muli.! You even include us sa batch id niyo hihi that was my first and last so thank you :

Sa mga 08 naman though late ako naging close sa inyo, happy ako to have gone close with the same age as me. Sa mga 08 na kasabayan ko sina Christine, Lalay, Dan and Arvin, kaya to! Hahaha thank you for the company and the talks, esp kayna Xtin and Lalay na Runningman-mates hihi ill miss those days when we're just gonna hang out in 141 and watch a runningman epi and laugh out loud without caring sa surrounding. Thank you to you girls! Ill miss you!

Also syempre how could I forgot yng mga close 08 friends ko? Sa mga kalokohan, lakad, "fire exit", outing, Im thankful to have been part of that. Cause I didn't have a block when I left, its nice to have people in my age to have those kalokohan moments with. Shout out to Joa, Ate Naji, Chessel, Patrick, Louie, Mik and Ian! I want another lakad after this I miss you guys! And yes we're ALL going to graduate no matter what! FIGHTING!

To my family thank you for all the support and help you have done for

me para lang dito. To Ate Nika who give me rides to go to school pag wala akong tulog, to Kuya Viboy who will always make me laugh to brighten up the mood, to my baby brother-na hindi na baby bro Ian, na despite being 17 is still malambing and caring sakin. And ofcourse to my one and only adorable niece na walang ginawa kung hindi makulit lol pero despite that, she makes me smile everyday such an angel. Im really thankful to have such cool siblings who im so close with na parang barkada na. I love hanging out with you guys and just laugh and enjoy things together. Love you ate, kuya, ian and pia!

Ofcourse my mom and dad, words cannot explain how thankful I am to have such supportive and kind-hearted parents. They will always sleep late as well just to help me in thesis, printing out my papers and doing the foldering of 100 pages per folder. My mom who spends so much para mapakain ang panel ng masarap na food and listen to my rants, cries, my dad who'll never fails to give and spoil me with whatever I want and encourage me to finish this. When I first decided to shift to comsci, they never once complained and supported me. Even when I was at the verge of giving up and shifting to another school bec of the pressure, they didn't scold and yet supported me on whatever decisions I make. Kahit ba nadelayed ako, dad told me its ok with him rather than me being pressured and stress if push it. Kay mama na I will cry to when I feel down and always hold my hand to keep me safe when im scared. I love my parents. They have given me everything I wanted and supported me always, what can I ask for more. I worked hard and studied hard bec of them. It's the least thing I can do to show them na appreciate everything they have done for me in my 22 years of life. MA AND PA THIS IS FOR YOU. GRADUATE NA AKO. THANK YOU PO SA LAHAT AND I LOVE YOU BOTH SO MUCH FOREVER.

And lastly hinuli ko to kasi this person has helped me a lot. Hindi lang dahil sa thesis pero for the whole 4 years of stay in comsci. She's the friend that would always put up with my crankiness and craziness and still accept

me for who I am- JAMIE thank you. We started out as just 'blockmates' since di tayo barkada we weren't close but that all change when we shifted to comsci. All I can say is I don't think I can survive those 4 years without her. Siya yng nagencourage sakin paggusto ko na magiveup. Siya yng nakikinig sakin sa mga petty rants ko about kpop. Siya yng kausap ko sa mga chismis na nakakuha naman. Siya yng nandiyan lagi pagumiiyak ako. Siya yun lagi andiyan to help me in projects and all nighters. Siya yun lagi nasa bahay dhil angthesis kami(LOL). Basta si Jamie ang masasabi kong my bestfriend in college. Through thick and thin she was always there. Thank you tlga friend. Lahat ng tears, lack of sleep and rants natin, nabawi na din. We made it and im super happy that you are with me ¡3 thank you friend and I love you! You'll always be that special friend who ill never ever forget and cherish. Time flies but friendship always remain and I know ours will always.

People come and go in life either to give you strength or give you a lesson to never give up BUT both are still blessing bec they made you what you are now. Ive learned and experienced a lot in my college life. There are ups and downs and those people who left me now but im still thankful. College is memorable for me and despite all the craziness of being an Iskolar ng Bayan- im pretty proud to say I graduated as one an survived the best school in Philippines. Im happy with my life and im happy on what I have achieved. Thank you and HENGSHO.