ABSTRACT

Systems mature. Like people, they grow bigger, gain more mass, and learn new skills. However, as systems mature, the complexity and its size tend to grow too. The system suddenly becomes a tangled mess, full of duplicate and redundant code and prone to bugs. This is where refactoring can help.

Refactoring is a process that involves altering the internal structure of the program or system to make it cleaner and more intuitive. It is a change made to the structure of the system to make it easier to understand and cheaper to modify. However it is important to note that refactoring does not aim to modify the observable behaviour of a system; in fact it is the opposite as it makes changes to the internal structure while making sure that the functionalities of the system stay the same. In addition, refactoring is also helpful in finding bugs because bugs in the code can easily be spotted by clarifying the structure of the program.

One candidate of such process is the Virus Host Interaction Lexicon system. Composed of seven modules, it has interlocking components whose code base has become so large that bugs and duplication of code become inevitable. Two of these modules, in particular the Virho References and Virho Hotspots, contain several bugs that prevent it from being useful. The refactoring of the Virho References module and Virho Hotspots module addressed these problems, while giving it the opportunity to be expanded and modified in the future versions because of its more modular and more manageable code.

keywords: *virholex, virho references, virho hotspots, refactoring*

Table of Contents

I. INTRODUCTION

Background of the Study

There are many different ways to write a program. Like an artwork or an essay, each program reflects the style and abilities of its creator. Every programmer uses his or her own approach in dealing with problems that his or her program intends to solve. In fact, different programmers can employ different techniques, patterns or methodologies to solve a similar puzzle. While there may exist guidelines in how programmers are supposed to write their codes, the uniqueness of each program can be seen in the way the programmer wrote the program's source code. Because of this, it is believed that no two programs are entirely similar.

However, as a program or a system matures, there can also be a corresponding increase in the size of its code base. As more functions are added or revised, codes tend to become unwieldy and too complex, to the point that there is a "code smell", a symptom of deep problem in the source code [1]. Thus, the need for refactoring arises.

Refactoring can be defined as "a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior" [1]. It usually involves altering the internal structure of the program or system to make it cleaner and more intuitive. In addition, refactoring is also helpful in finding bugs because bugs in the code can easily be spotted by clarifying the structure of the program. Moreover, since source code will be read and modified more frequently than it will be written, refactoring can be used to keep code readable and modifiable [1].

VirHoLex (**Vir**us-**Ho**st Interaction **Lex**icon) is a system formulated by a research/study group composed of three academic institutions in the Philippines (University of the Philippines Manila, De La Salle University, Mapua Institute of Technology) to set up a **C**ommunity **Or**iented **I**nformation (CORI) platform for virologists around the world. This initiative was born from the perspective of U. Reichl (Director, Max-Planck-Institute for the Dynamics of the Complex Technical Systems, Magdeburg, Germany), J. Haas (LMU & Edinburgh University Medical Schools), J. Rädler (LMU Faculty of Physics) and the author for the postdoctoral research of J. Bantang (on leave from the UPD National Institute of Physics). The project was inspired by EUCLIS (EUCLOCK Information System), another CORI system developed for the global community of chronobiologists. VirHoLEx will provide information about three prevalent viruses in the country: Dengue, Influenza A, and Herpes Viruses [2].

Seven modules comprise the VirHoLex system, according to the specification of the system [3]. The User Interface Module handles ways of accessing information about the system (e.g. per module, per virus). It also provides the general layout of menus and links to the other six components of VirHoLex. Interfaces such as hotspot features are also managed in this module. The Registered User Services Module grants and delimits access for varying types of users in the different modules of VirHoLex. The Information Services Module considers information affiliated with other system modules (i.e. Images, Hotspot User Interface feature). It also manages queries performed in the databases internal to VirHoLex and from selected external databases. The Experiments Module stores and manages laboratory experiments, experiments descriptions, metadata and data files. It also includes tools for visualization of some data types. The Images Module serves as repository for images with their associated metadata for and from the users of VirHoLex. In this module, the users may upload/download/search images that may be closely coupled with other modules in VirHoLex such as Virho Experiments, Virho Models, etc. The Models Module is a repository of summary descriptions of relevant models/modelling studies. Lastly, the References Module stores bibliographic entries or references of virologists, particularly from Endnote files, which they can easily share among themselves [2].

However, Virholex still has bugs in some of its modules, including the images module, the references module, the hotspots module and the experiments module. Bugs in the other modules that have not yet been identified can also exist. More so, the references module and the experiments module are either lacking in features or is not consistent with the original specifications of the system.

These shortcomings can be obstructions in the deployment of the Virholex system. Thus, refactoring and rewriting of the said modules are needed.

Statement of the Problem

General

Software bugs exist in the different modules of the first version of the Virus-Host Interaction Lexicon system. These modules include, but are not limited to, the hotspots module, images module, references module, and the experiments module.

Specific

Different software bugs in the modules of the first version of the Virus-Host Interaction Lexicon exist. These modules are the hotspots module, images module, references module, and the experiments module. However, bugs from the other modules can also exist. These bugs can be an obstacle in the use of the system by virologists. More so, the experiments and the references module lack the proper documentation to facilitate possible further developments to the system.

Objectives of the Study

The aim of this study is to be able to refactor and rewrite some of the modules of the Virus-Host Interaction Lexicon system without introducing new bugs, as well as complete the proper documentations.

The References Module's original objectives are as follows:

1. Allow *collection coordinator* to:
    a. view/browse/search bibliographic entries
    b. download documents
    c. export entries to EndNoteXML files
    d. add/edit bibliographic entries
    e. delete bibliographic entries
    f. manage references collection user
2. Allow *collection contributor* to:
    a. view/browse/search bibliographic entries
    b. download documents

c. export entries to EndNoteXML files

d. add/edit bibliographic entries

3. Allow *restricted user* to:

    a. view/browse/search bibliographic entries

    b. download documents

    c. export entries to EndNoteXML files

4. Allow *registered user* to:

    a. view/browse/search bibliographic entries

Below is the tabular form of the capabilities of each user level per functionalities in the Virho Reference Module.

REFERENCE ACCESS LEVEL		**PRIVILEGES**					
		View/Browse/Search Bibliographic Entries	**Download Documents**	**Export Entries To EndNoteXML Files**	**Add/Edit Bibliographic Entries**	**Delete Bibliographic Entries**	**Manage References Collection User**
4	*Collection coordinator*	☑	☑	☑	☑	☑	☑
3	*Collection Contributor*	☑	☑	☑	☑	✖	✖
2	*Restricted User*	☑	☑	☑	✖	✖	✖
1	*Registered User*	☑	✖	✖	✖	✖	✖

Table 1 Summarized capabilities of each user level for Reference Module

The Image Hotspots feature's original objectives are as follows:

1. Allow *hotspot manager* to:

    a. view hotspot diagram and basic virus information

    b. upload new Virho Hotspot diagram

    c. add/edit/delete hotspots and basic information

    d. manage references collection user

2. Allow *other users* to:

a. view hotspot diagram and basic virus information

Below is the tabular form of the capabilities of each user level per functionalities in the Virho Hotspots Module.

Hotspot Info Access Level		**Privileges**			
		View Hotspot Diagram and Basic Virus Information	**Upload new Virho Hotspot Diagram**	**Add/Edit/Delete Hotspot and Basic Information**	**Manage References Collection User**
4	*Hotspot Manager*	☑	☑	☑	☑
0-3		☑	✖	✖	✖

Table 2 Summarized capabilities of each user level for Hotspots Module

On top of the system's original objectives, this study has the following additional objectives:

1. Identify the bugs in the system
   a. References Module
   b. Image Hotspots feature
2. Refactor the source code of the system
   a. Identify the code smells
      i. Duplicated code
      ii. Long method
      iii. Large class
      iv. Long parameter list
      v. Divergent change
      vi. Shotgun surgery
      vii. Feature envy
      viii. Data clumps
      ix. Primitive obsession
      x. Switch statements

    xi. Parallel inheritance hierarchy
    xii. Lazy class
    xiii. Speculative generality
    xiv. Temporary field
    xv. Message chains
    xvi. Middle man
    xvii. Inappropriate intimacy
    xviii. Alternative classes with different interfaces
    xix. Incomplete library class
    xx. Data class
    xxi. Refuse bequest
    xxii. Comments

b. Refactor the code using refactoring techniques

    i. Collapse hierarchy
    ii. Encapsulate field
    iii. Extract method
    iv. Extract class
    v. Extract subclass
    vi. Replace parameter with method
    vii. Pull up method
    viii. Pull up field
    ix. Push down method
    x. Push down field

xi. Move method

xii. Move field

xiii. Rename method

xiv. Rename field

xv. Replace temp with query

c. Perform tests

i. Unit testing

ii. Integration testing

iii. System testing

d. Document the refactorings done

3. Complete the documentation of the images and references modules

Significance of the Study

Refactoring

Elimination of bugs in the system and improvement of the modules in line with its original design can help in the faster adoption and wider use of the Virus-Host Interaction Lexicon system for virologists. More so, the completion of the proper documentations for the system can lead to the evolution of the VirHoLex system.

Image Hotspots

Information about almost everything constantly updates. Even the information about virus is regularly restructured. Since such trend is observable, using the Image Hotspot feature can help virologists to regularly update the information in the virus diagram as well as the basic virus information [2].

References Module

Reference citation and bibliographies are integral parts of an academic work. Not only do they give credence to a theory or claim, they also provide a way for readers to verify the information presented. More so, reference citation gives courtesy and credit to original authors of different academic works. With this, the References module of the Virus-Host Interaction Lexicon system will provide a way for the users to record, organize and share their reference lists.

Scope and Limitations

This study is only concerned with the refactoring of the images and references modules. It will also cover the refactoring of the other modules of the system related to the previously mentioned two modules. It will not add new modules to the system nor will it add new functionalities not in line with the original design of the system.

The References module will only act as a reference manager for the users of the VirHoLex system. It will allow the users to share, add and organize their reference lists. However, it will not be responsible for providing other users the referenced material unless uploaded by the original uploader.

For the Image Hotspot feature, its functionalities can only be used by the Hotspot Manager. Furthermore, the Image Hotspot feature does not directly modify the location of the hotspots on the diagram; it merely manages the information contained on these stages. In order to revise the location or title of the stages, a third party software that edits SVG files is needed. The system will not provide any direct download of such software but can provide a link to a website [2].

In addition, the browsers of the users are expected to be able to handle SVG files in order to properly display the diagram. Hotspot managers are assumed to be knowledgeable of any software that handles SVGs in order for them to upload revised diagrams.

II. REVIEW OF RELATED LITERATURE

Refactoring

Programmers know that software code is read and modified more frequently than it is written; thus there is a need to keep the code readable and modifiable. One way to ensure the readability and modifiability of software code is through refactoring [1]. This technique has gained wide acceptance that Brant and Roberts, in their plenary talk, presented the idea of refactoring as an "essential tool for handling software evolution" [4]. However, one of the major obstacles in refactoring is discerning where and when to refactor. In his book, Fowler stated that refactorings are based on human intuition and that "no set of metrics rivals informed human intuition" [1].

However, in the paper *Metrics Based Refactorings*, the researchers showed that metrics can help to identify particular anomalies for certain types of refactorings. They believed that tool support is necessary to assist the human intuition in a very efficient and effective way. More so, they argued that software visualisation based on static structure analysis and metrics is a key issue for this task. Thus, they presented a generic approach to generate visualisations supporting the developer to identify candidates for refactoring [5]. However, due to the premise that the developer has to be the last authority in identifying and applying refactorings, their work focused on providing decision support.

An alternative way to help in the identification of anomalies is the use of program invariants. The paper *Automated Support for Program Refactoring using Invariants* used an invariant detection tool and an invariant pattern matcher to automatically detect candidate refactorings. A particular pattern of invariants at a program point indicates the applicability of a specific refactoring [6]. The paper showed that the use of program invariants can be a complementary approach to tools that help to automate refactoring itself.

Another point that Brant and Roberts mentioned in their plenary talk is the problem of integrating refactoring in software lifecycles. They argued that development methods do not necessarily support software evolution and by extension refactoring. It thus created a need for integration of refactoring into these different methodologies and techniques [4].

UML or the Unified Modelling Language is a design language used to model system behaviour and structure. Its syntax is precisely defined by a metamodel, and various structural and dynamic views exist. However, one of the problems faced by designers is that it is often hard to measure the actual impact of modifications on the various design views, as well as on the implementation code. In the paper *Refactoring UML Models*, the researchers showed that "refactorings can be defined in such a way that their behaviour-preserving properties are guaranteed, based on the OCL constraints at the meta-model level" [7].

Extreme Programming (XP) is a lightweight development process which focuses on unit testing. "If there is a technique at the heart of extreme programming (XP), it is unit testing." [8] As part of their programming activity, XP developers write and maintain unit tests continually. These tests are automated, written in the same programming language as the production code, considered an explicit part of the code, and put under revision control. The downside of having many tests, however, is that changes in functionality will typically involve changes in the test code as well. Thus, ensuring readability and modifiability is important in the test code. In the paper *Refactoring Test Code*, the researchers identified different test smells to help developers identify weak spots in their test code, provide a solution to these problems, and give a "set of specific test refactorings to help developers make improvements to their codes in a systematic way" [9].

Aspect-orientation is a new programming paradigm that increases the modularity of software. It provides means to encapsulate concerns which cannot be modularized using traditional programming techniques. These concerns are called crosscutting concerns and examples of such are tracing, concurrency control or transaction management. However, since this a different programming paradigm, refactoring tools for object-oriented base system cannot be used. The tool needed should be aspect aware. Thus, an Eclipse plug in tool that supports AspectJ (an aspect-oriented extension for Java) aware refactorings in that IDE was developed. It consists of three collaborating parts: coding wizard, transformation and code generation, and condition checking [10].

References

References and citations are standard parts in a scientific paper or book. It is a form of intellectually honesty, and gives courtesy and credit to original authors of different academic works. More so, these acts as source for theories or claims that are used in the text, giving credence to the work. Many reference management software exist today, from web-based systems with online storage space, to plugs ins for desktop tools programs or web browsers and to stand alone desktop programs.

BibTex is a tool for automating list of references for a particular work. It takes care of automating tedious tasks such as sorting bibliography entries either alphabetically or as they appear in the text. Each entry is formatted according to the bibliography style chosen by the user. In addition, citations in the continuous text are also formatted automatically [11]. This is done by inserting commands in the said text.

One feature of BibTex is the ability to incorporate user styles into the program. A style is basically a file that tells BibTex how to format the entries that will go in the reference list. The language used for these files has ten commands that manipulate the language's objects: constants, variables, functions, the stack and the entry list [12].

While BibTex is a useful tool, it has a simple and bare format that does not allow complex queries and manipulations. This can be a problem for online collections that contain huge amounts of references. Thus, BibTeXML was developed. BibTeXML is an XML environment for representing and structuring BibTex bibliographies, which make management of bibliographical data easier, and to build an online database which allows upload and download of bibliographic entries [13]. This database will provide complex queries and data navigation which help the user to fetch the required references.

MLBibTex is another implementation of BibTex. It stands for Multilingual BibTex, and was initiated because of the interest in multilingual processors nowadays. It allows its users to specify *language changes*, which is a string of characters that expresses conformity to other typographical conventions and can be used to hyphenate foreign language; and *language switches*, which are used for information about what must be put, possibly in another language, and for details that can be given in a particular language but can be omitted if no translation is available [14].

Another type of reference manager is Connotea. It is a free online reference management and social bookmarking service for scientists created by Nature Publishing Group. Its main feature is the auto-discovery and the ability to import bibliographic information for any article or book that is added in the system. It has an online storage of references and bookmarks, with the capability to share to friends and colleagues as well as to other users of the service. This allows Connotea to make recommendations to the users using sophisticated collaborative filtering algorithms. It uses a simple, non-hierarchal flat file system where data can be viewed from the perspective of tags, or users, or links [15].

EndNote is another popular reference management software program among biomedical and healthcare professionals [16]. It is used to manage references, insert citations into manuscripts, and format bibliographies. EndNote uses a library that is essentially an electronic database containing various types of references, such as journal articles, books, magazine articles, figures and tables. It consists of various reference fields such as Author, Title, Year URL and Publication Date. A library can include files such as images, PDFs or Excel spreadsheet associated with references.

Image Hotspots

SVG is a language that was developed by the World Wide Consortium. SVG is defined to be a two-dimensional vector graphics for both strong information and distribution of information on the Web [2].

SVG formats do not reduce quality even when scaled; this is because SVG files are stored as a collection of instruction rather than a set of dots as compared to raster formats like JPEG, GIF or PNG. Besides the resolution advantage, SVG files are commonly small in size and web pages can easily load them. SVG files are also easily integrated with other scripting languages that further enhance the quality and the flexibility of the vector graphic. SVG files are also used in animations and interactive effects [2].

Since SVG is a two-dimensional graphics format, a Cartesian plane coordinate system is implemented. This allows developers to use SVG formats for mapping purposes with both static and interactive features [2].

SVG behaves like a normal image but with more features. Such feature makes it more flexible compared to traditional image formats. "SVG allows translations, rotations, scaling, skewing and matrix transformations. All transformations may be combined and nested. SVG allows the definition or creation of viewpoints either per link or per script." [2] SVG can also handle scripting. Such scripting adds interactivity within the SVG file. These interactivities include events, hyperlinks, animations, and other special interactivity elements [2].

III. THEORETICAL FRAMEWORK

Refactoring

Refactoring is defined as a “change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour” [1]. It a technique used to clean up code in an efficient and controlled manner. While it is usually just a small change to the software, one refactoring can also involve others; thus it becomes a series of refactoring. This is done to improve the structure of software. It is like tidying up the code, ensuring that the code retains its shape. With this, refactoring helps in the preservation of the structure of the software, and slows down the possible decays in its source code.

The refactoring process is done as a series of activities. The list is as follows [17]:

1. Identify where the software should be refactored (code smells) .
2. Determine which refactoring(s) should be applied to the identified places.
3. Guarantee that the applied refactoring preserves behavior.
4. Apply the refactoring.
5. Assess the effect of the refactoring on quality characteristics of the software (e.g., complexity, understandability, maintainability) or the process (e.g., productivity, cost, effort).
6. Maintain the consistency between the refactored program code and other software artifacts (such as documentation, design documents, requirements specifications, tests, etc.).

A. Code Smells

Code smells is a term coined by Kent Beck and Martin Fowler to describe "certain structures in the code that suggest the possibility of refactoring." [1] These are the common symptoms in software programs that possibly indicate a deeper problem. Programmers use code smells as heuristics to indicate when to refactor, and what specific refactoring techniques to use. Thus, a code smell is a driver for refactoring.

However, it is important to note that determining whether a specific part of a program is a code smell or not is often a subjective judgement. It is up to the programmer to develop his own sense of "how many instance variables are too many instance variables and how many lines of code in a method are too many lines." [1]

Below is a list of some of the code smells and their definitions, taken from Fowler's book [1]:

- Duplicated code – same code structure in more than one place
- Long method – long procedures; methods with lots of parameters and temporary variables
- Large class – class is trying to do too much; class has too many instance variables
- Long parameter list – lots of parameters to be passed
- Divergent change – one class is commonly changed in different ways for different reasons
- Shotgun surgery – a change in the class results to a lot of little changes to a lot of different classes
- Feature envy – a method that seems more interested in a class other than the one it actually is in

- Data clumps – bunches of data that hang around together
- Primitive obsession – storing more information on primitives rather than making a class
- Switch statements – same switch statement scattered about a program in different places
- Parallel inheritance hierarchy – for every subclass of one class, there is also subclass for another
- Lazy class – class that isn't doing enough that may be remnants of refactoring processes
- Speculative generality – only users of a method or class are test cases
- Temporary field – an object in which an instance variable is set only in certain circumstances
- Message chains – a client asks one object for another object, which the client then asks for yet another object, which the client then asks for yet another object, and so on
- Middle man – most of the methods of a class are delegated to another class
- Inappropriate intimacy – classes become far too intimate and spend too much time delving in each other’s' private parts
- Alternative classes with different interfaces – methods that do the same thing but have different signatures for what they do
- Incomplete library class – libraries lacking methods
- Data class – classes that have fields, getting and setting methods for the fields, and nothing else; dumb data holders

- Refuse bequest – most data inherited by subclasses are not wanted or needed
- Comments – comments are only there because code is bad

B. *Refactoring Techniques*

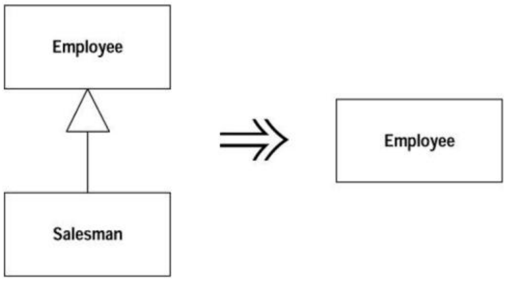
Refactoring technique is a term to describe the series of refactoring steps that needs to be done given a specific set of circumstances. There are many refactoring techniques available, currently indexed by Martin Fowler in his book [1] and in his website [18]. Some of these can be used to allow for more abstraction, others for breaking code apart into more logical pieces, or for improving names and location of code.

Below are some examples of refactoring techniques taken from Fowler's book [1]:

COLLAPSE HIERARCHY

Problem: A superclass and subclass are not very different.

Solution: Merge them together.



Before and after for ***collapse hierarchy*** *technique*

ENCAPSULATE FIELD

Problem: There is a public field.

Solution: Make it private and provide accessors.

```
public String _name
```

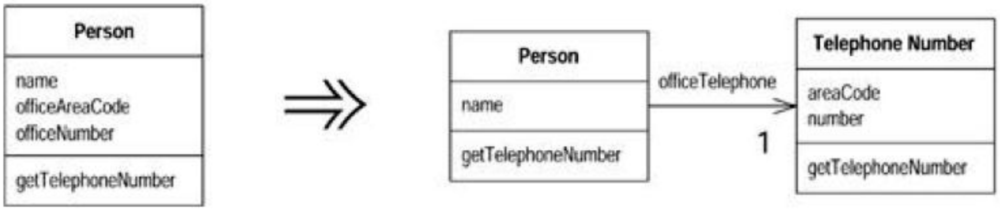
```
private String _name;
public String getName() {return _name;}
public void setName(String arg) {_name =
arg;}
```

*Before and after for **encapsulate field** technique*

EXTRACT CLASS

Problem: You have one class doing work that should be done by two.

Solution: Create a new class and move the relevant fields and methods from the old class into the new class.



*Before and after for **extract class** technique*

EXTRACT METHOD

Problem: You have a code fragment that can be grouped together.

Solution: Turn the fragment into a method whose name explains the purpose of the method.

```
void printOwing() {

     Enumeration e = _orders.elements();
     double outstanding = 0.0;

     printBanner();

     // calculate outstanding
     while (e.hasMoreElements()) {
         Order each = (Order)
e.nextElement();
         outstanding += each.getAmount();
     }

     //print details
     System.out.println ("name:" +
_name);
     System.out.println ("amount" +
outstanding);
}
```

```
void printOwing() {

     Enumeration e = _orders.elements();
     double outstanding = 0.0;

     printBanner();

     // calculate outstanding
     while (e.hasMoreElements()) {
         Order each = (Order)
e.nextElement();
         outstanding += each.getAmount();
     }

     printDetails(outstanding);
}

void printDetails (double outstanding) {
    System.out.println ("name:" + _name);
    System.out.println ("amount" +
outstanding);
}
```

Before and after for ***extract method*** *technique*

Replace Parameter With Method

Problem: An object invokes a method, and then passes the result as a parameter for a method. The receiver can also invoke this method.

Solution: Remove the parameter and let the receiver invoke the method.

```
int basePrice = _quantity * _itemPrice;
discountLevel = getDiscountLevel();

double finalPrice =
discountedPrice(basePrice,
discountLevel);
```

```
int basePrice = _quantity * _itemPrice;

double finalPrice =
discountedPrice(basePrice);
```

Before and after for ***replace parameter with method*** *technique*

Pull Up Method

Problem: You have methods with identical results on subclasses.

Solution: Move them to the superclass.

```
  public class SuperClass{
       void methodA() {
           //do something
       }

       void methodB() {
           //do something else
       }
   }

   public class SubClass extends
SuperClass {
       void methodC() {
           //do something
       }
   }
```

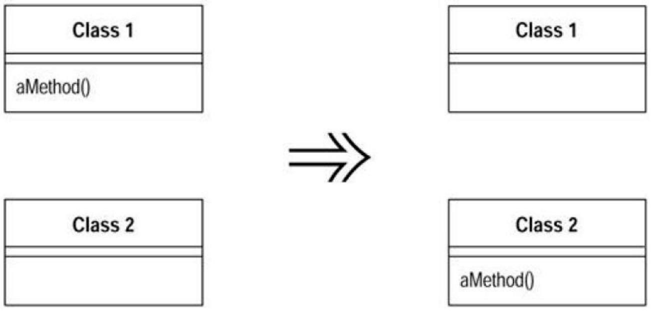
```
   public class SuperClass{
       void methodA() {
           //do something
       }
   }

   public class SubClass extends
SuperClass{
       void methodB() {
           //do something
       }

       void methodC() {
           //do something else
       }
   }
```

Before and after for ***push down*** *technique*

Move Method

Problem: A method is, or will be, using or used by more features of another class than the class on which it is defined.

Solution: Create a new method with a similar body in the class it uses most. Either turn the old method into a simple delegation, or remove it altogether.

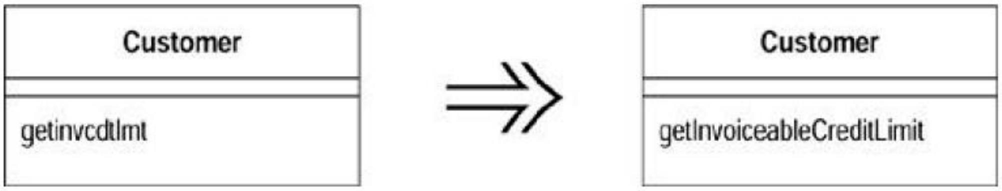


Before and after for ***move method*** *technique*

Rename Method

Problem: The name of a method does not reveal its purpose.

Solution: Change the name of the method.



Before and after for ***rename method*** *technique*

<u>REPLACE TEMP WITH QUERY</u>

Problem: You are using a temporary variable to hold the result of an expression.

Solution: Extract the expression into a method. Replace all references to the temp with the expression. The new method can then be used in other methods.

```
double basePrice = _quantity *
_itemPrice;

if (basePrice > 1000)
   return basePrice * 0.95;
else
   return basePrice * 0.98;
```

```
if (basePrice() > 1000)
   return basePrice() * 0.95;
else
   return basePrice() * 0.98;

...

double basePrice() {
   return _quantity * _itemPrice;
}
```

Before and after for ***replace temp with query*** *technique*

The given examples are usually used in the refactoring of object-oriented programming such as the Java.

C. Unit Testing

Unit testing is a method for “modular testing of a programs’ functional behavior” [19]. A program is decomposed into units which are collections of functions, and the units are independently tested. The test is done by generating inputs for a single entry function. This is a practical approach to increasing the correctness and quality of software.

D. Performance Optimization vs. Refactoring

The purpose of refactoring is to make the software easier to understand and modify. A programmer can introduce many changes in software that make little or no change in the observable behavior, but only changes made to make the software easier to understand are refactorings.

Performance optimization, on the other hand, is the "process of modifying a software system to make some aspect of it work more efficiently or use fewer resources" [20]. Software may be optimized so that it executes more rapidly, or is capable of operating with less memory needs or other resources, or draw less power. Like refactoring, performance optimization does not usually change the behavior of a component; it only alters the internal structure [1]. However, optimization often makes the code harder to read and understand as well as and add code that is used only to improve the performance. This usually complicates software, making it harder to maintain and debug.

References

A. EndNote

EndNote is one of the most popular reference management software program among biomedical and healthcare professionals. Used to manage references, insert citations into manuscripts, and format bibliographies, EndNote uses a library that is an electronic database containing various types of references, such as journal articles, books, magazine articles, figures and tables. In addition library can include files such as images, PDFs or Excel spreadsheet associated with references [16].

B. References vs. Reference List/Bibliography vs. Citation

A reference is a short description or note that contains information about the source. Simply put, a reference is the "address" of the source. References enable the reader to access and verify the original source of information. A citation or in-text citation meanwhile is a link to the reference in the body of the manuscript in a short form [16].

A reference list is a numbered or alphabetically sorted list of references that are actually cited in the text of the manuscript as endnotes or footnotes. Bibliography is a term typically used to indicate a comprehensive list of all the resources the author has consulted during the course of the research. It may include resources in addition to those cited in the text. Note that the terms bibliography and reference list are often used interchangeable in common practice [16].

Image Hotspots

A. Raster Graphics vs. Vector Graphics

Today, there are two kinds of computer graphics available. One is called the raster graphics that are composed of pixels while the other is called vector that is composed of paths. Raster uses a grid of pixels that has varying color and shade. Vector graphics, on the other hand, use mathematical relationships between points and paths [2].

Raster graphics tend to be pixelated when scaled and produces rough edges. Raster graphics are just plain images that can only be embedded on a webpage. Creating hotspots over raster graphics is easy to do but produces less interactivity and less effects. Moreover, these hotspots will be written within the HTML file and thus can be seen when the source is looked at.

Vector graphics also support scripting that can create hotspots. Hotspots can then be more flexible as to either a text or a region within the vector image. Since it uses scripting, the underlying code will not be shown when the source is looked at [2].

B. Scalable Vector Graphics

Scalable Vector Graphics is a platform for describing two dimensional vector graphics, both static and dynamic. The SGV specification is an open standard developed by the World Wide Web Consortium (W3C). It is composed of an XML-based file format and a programming API for graphical applications. Key features include shapes, text and embedded raster graphics, with many different painting styles [21]. It supports scripting through languages such as ECMAScript and has comprehensive support for animation.

SVG images can interact with users in many ways. One is through linking, where SVG images can contain hyperlink to other documents. More so, any part of an SVG image can be made to trigger events using scripting. These events can either be changes in focus, mouse clicks, scrolling or zooming the image and other pointer, keyboard and document events. Event handlers may start, stop or alter animations and trigger any other scripts in response to these events.

Animation with SVG documents can be done using the built-in animation elements, or by manipulating the Document Object Model (DOM) using ECMAScript. Animations in SVG can be continuous, they can loop and repeat and they can respond to user events, as mentioned above.

IV. DESIGN AND IMPLEMENTATION

An integral part of the study is the documentation of the refactorings done the different parts of the system. The format will be as follows:

Problem	The problem identified based on the code smell found. This may include a short description as well as other related information.
Method	The refactoring technique used to fix the problem.
Code	The lines of code wherein the problem is identified. Note that these lines may have been altered by previous refactorings, and thus are not necessarily the original lines of code found in the first version of the system.
Modified Code	The new code after the refactoring technique is applied.

Entity Relationship Diagram

The Entity Relationship Diagram (ERD) for the Virho References module is shown below. From the diagram, it can be seen that the database tables relevant to the module are `document`, `reference` and `user_prev`. The tables that are the primary scope of Virho References module are the `document` and the `reference` tables.

The relationships of the tables are as follows: A particular reference, stored in the `reference` table, can have zero or one supporting document. Conversely, a particular document can only be included in one reference. The relationship of the two entities is noted through the `reference_id` in the `document` table. A user, whose privileges for a particular module of a particular project are stored in the `user_prev` table, can create many reference entries. On the other hand, a reference entry can only have a particular user as its creator. Their relationship is noted through the `prev_id` in the reference table.

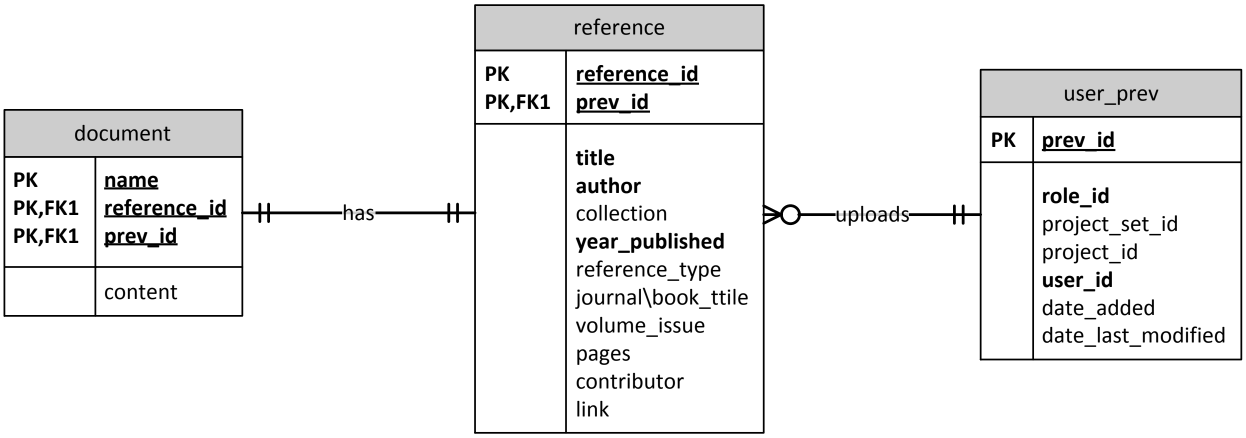


Figure 1 Entity Relationship Diagram

The Entity Relationship Diagram (ERD) for the Virho Image Hotspots is shown below. From the diagram, it can be seen that the database tables relevant to the feature are `virus` and `virhohotspot`. The table that is the primary scope of Virho Image Hotspots is the `virhohotspot` table.

The relationships of the tables are as follows: A particular virus, stored in the virus table, has exactly one hotspot diagram. Conversely, a particular hotspot diagram can only have one kind of virus assigned to it. The relationship of the two entities is noted through the `virus` entry in the `virhohotspot` table.

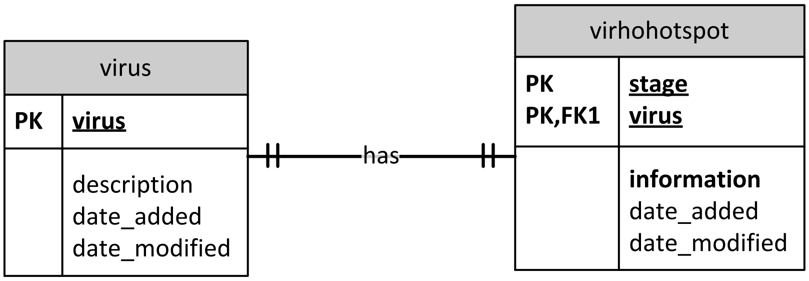


Figure 2 Entity Relationship Diagram

Data Dictionary

The tables below are the detailed description of the tables that are the primary scope of the Virho References module. The reference table contains information about each reference entry, while the `document` table contains information about the document file associated with each entry.

Attribute	Attribute type	Description
reference_id	integer	primary key of the reference item cited
title	varchar(50)	title of the reference item cited
author	varchar(50)	author of the reference item cited
collection	varchar(30)	collection where the entry is included
year_published	varchar(4)	year the reference item cited was published
reference_type	varchar(20)	type of reference item
journal\book_title	varchar(50)	title of the journal or book where the reference item was lifted
volume_issue	varchar(5)	volume issue of the journal or book where the reference item was lifted
pages	varchar(5)	pages of the journal or book where the reference item was lifted
contributor	varchar(50)	contributor of the journal or book where the reference item was lifted
link	varchar(50)	hyperlink of the reference item cited
prev_id	integer	foreign key from the `user_prev` table

Table 3 Detailed description of the reference table

Attribute	Attribute type	Description
name	varchar(50)	name of the document file of the reference item/primary key
content	text	content of the document file of the reference item

reference_id	integer	foreign key from the `document` table

Table 4 Detailed description of the document table

The tables below are the detailed description of the table that is the primary scope of the Virho Hotspots. The `virhohotspot` table contains information about each virus image diagram.

Attribute	Attribute type	Description
stage	varchar(100)	name of the infection stage/primary key
virus	varchar(100)	foreign key from the `virus` table
information	text	information displayed for that stage
date_added	timestamp	date when the information was added
date_modified	timestamp	date when the information was modified

Table 5 Detailed description of the virhohotspot table

Use Case Diagrams

Virho References Module

The use case diagrams below illustrate the functionalities of the Virho References Module available to each user level. Only registered Virholex users have access to images. Level 1 (Registered user), level 2 (Restricted user), level 3 (Collection contributor) and level 4 (Collection coordinator) users as defined by the Registered Users Services are those considered as registered Virholex users. Unregistered users (level 0), thus, will not be able to access the references.

Level 2 (Restricted user) to level 4 (Collection coordinator) users can download the document file of the reference entry, but only level 3 (Collection coordinator) and level 4 users can edit and add bibliographic entries.

The figures below are the use case diagrams together with its corresponding use case descriptions and sequence diagrams of the Virho Reference module taken from the VirHoLex Functional Specification for Release 1.0 [3].

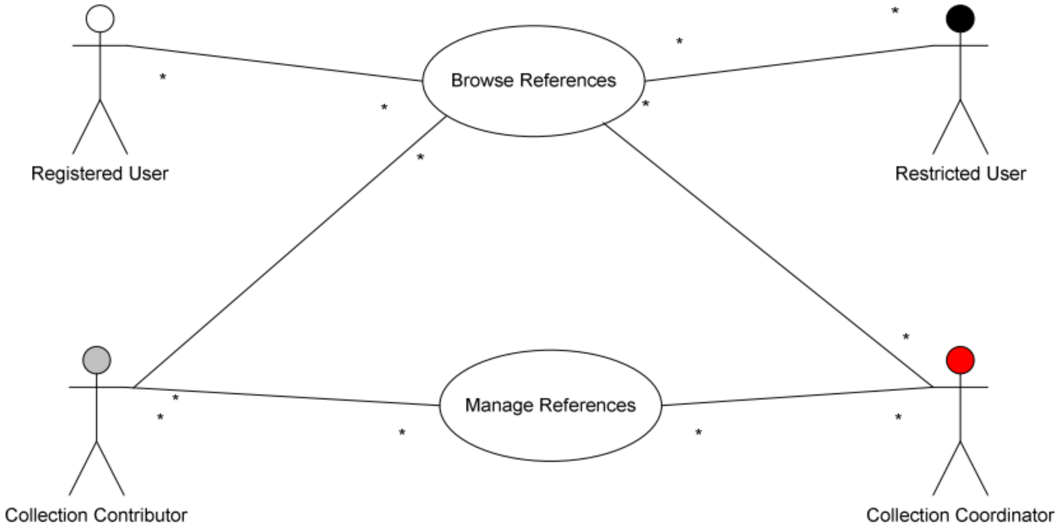


Figure 3 Virho References Use Case Diagram

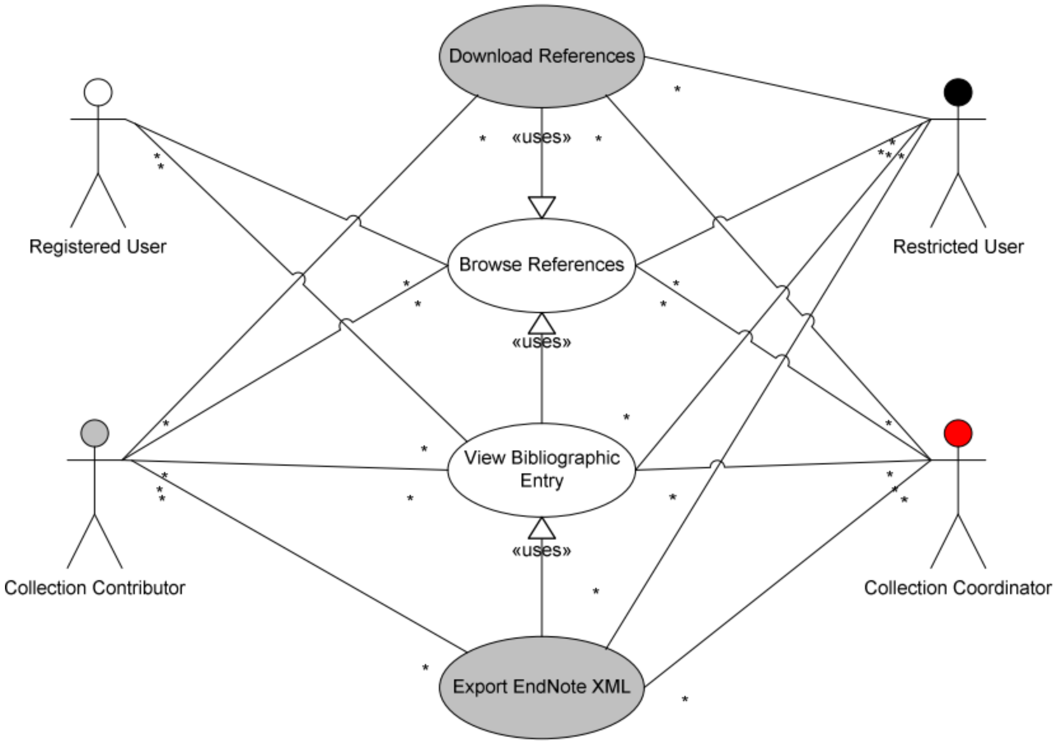


Figure 4 Browse References Use Case Diagram

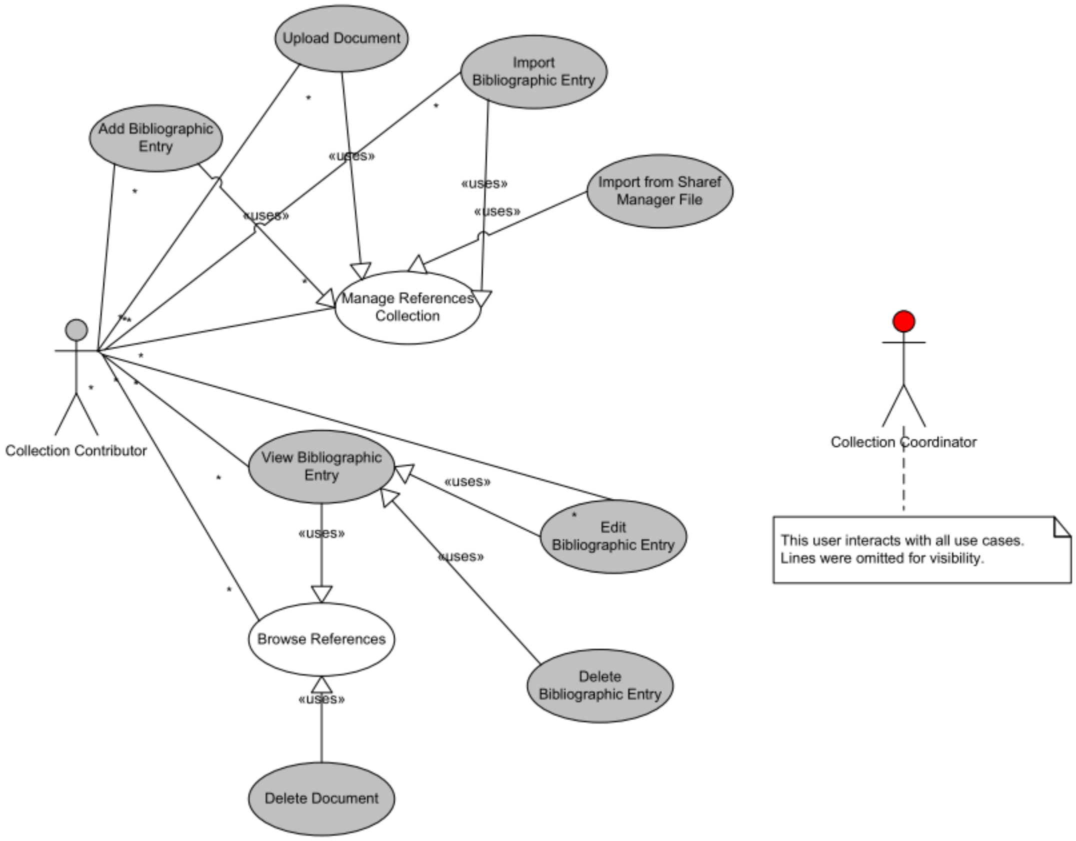
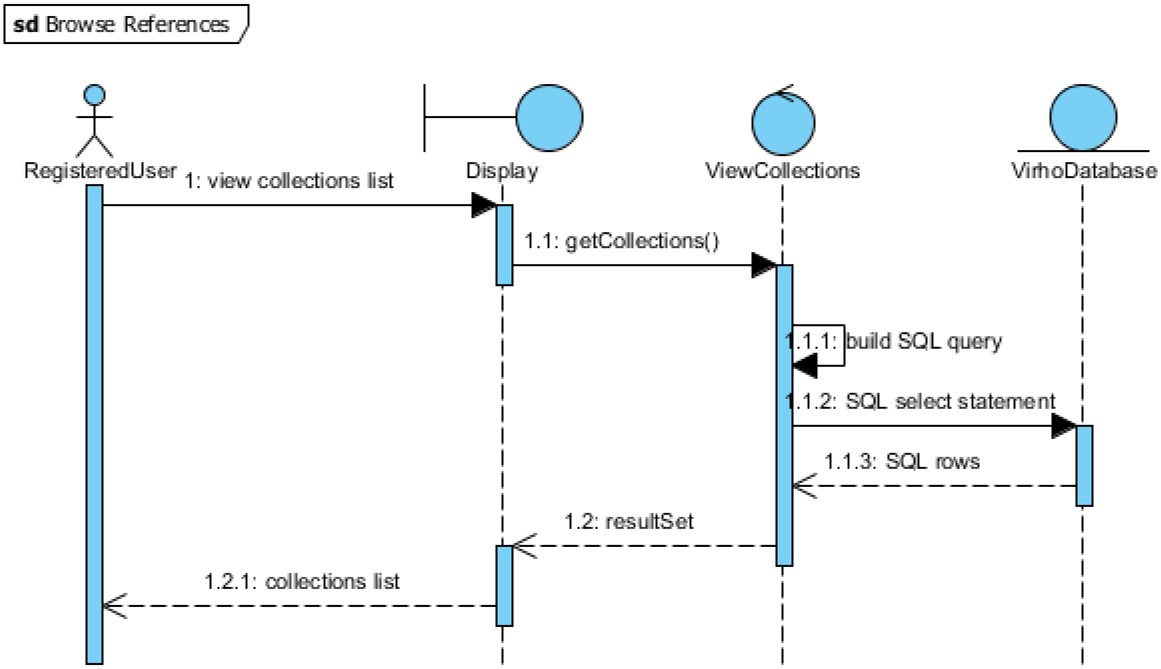


Figure 5 Manage References Use Case Diagram

<table>
<tr><td><b>USE CASE NAME:</b></td><td colspan="2">Browse References</td></tr>
<tr><td><b>SCENARIO:</b></td><td colspan="2">View references list</td></tr>
<tr><td><b>TRIGGERING EVENT:</b></td><td colspan="2">Registered Virholex user clicks the Virho References link</td></tr>
<tr><td><b>BRIEF DESCRIPTION:</b></td><td colspan="2">Registered Virholex user goes to the Virho References module. The system then displays a list of stored references.</td></tr>
<tr><td><b>ACTORS:</b></td><td colspan="2">Registered User, Restricted User, Collection Contributor, Collection Coordinator</td></tr>
<tr><td><b>RELATED USE CASES:</b></td><td colspan="2">None</td></tr>
<tr><td><b>PRECONDITIONS:</b></td><td colspan="2">User must be a registered Virholex user</td></tr>
<tr><td><b>POST CONDITIONS:</b></td><td colspan="2">A list of references is displayed</td></tr>
<tr><td rowspan="2"><b>FLOW OF EVENTS:</b></td><td><b>Actor</b></td><td><b>System</b></td></tr>
<tr><td>1. Registered Virholex user clicks on the Virho References link</td><td>1.1. Redirect to the Virholex References page
1.2. Display list of available references grouped by</td></tr>
</table>

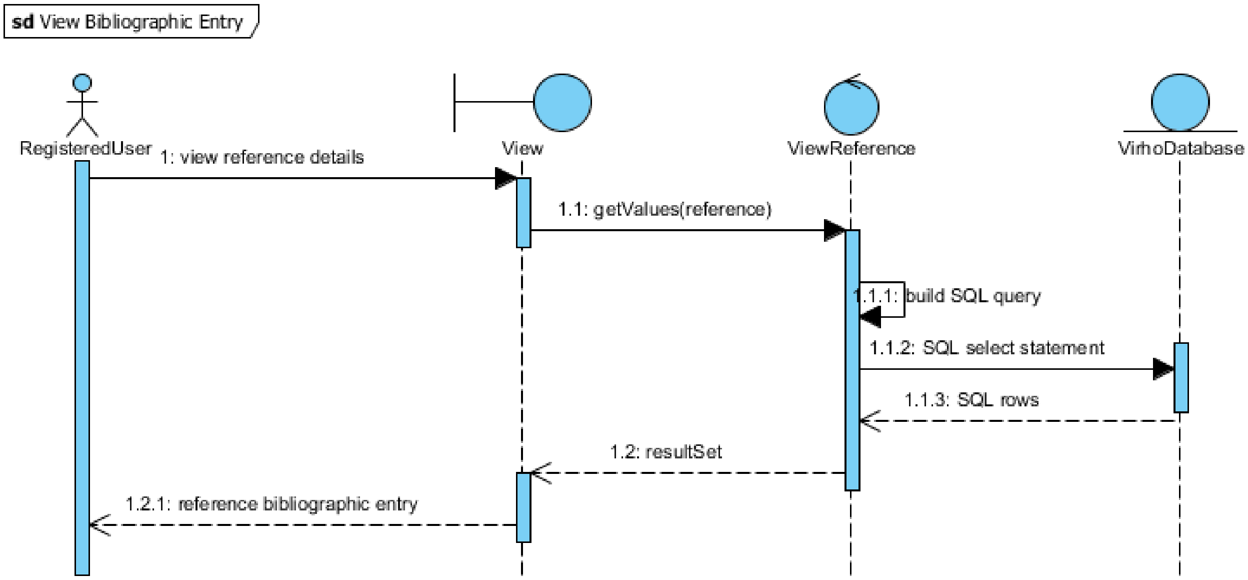
		collection
EXCEPTION CONDITIONS:	1.1 If the user is not a registered Virholex user, then the system pauses this use-case	

Table 6 Detailed description of Browse References Use Case



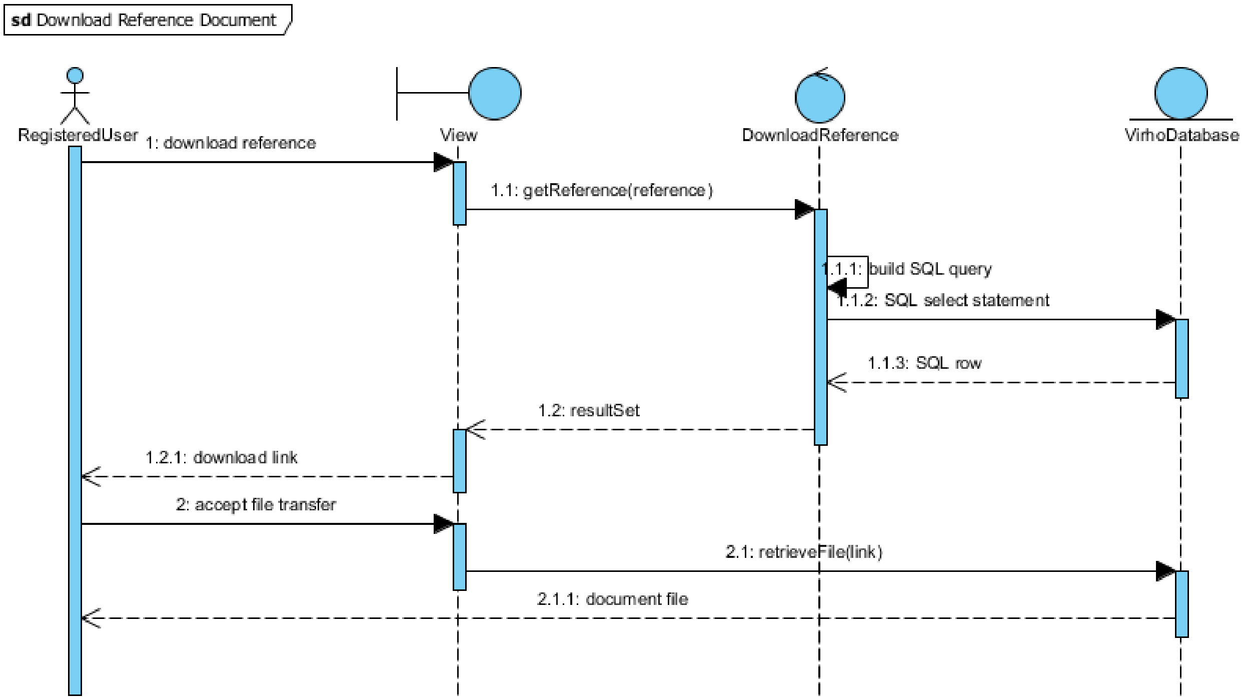
USE CASE NAME:	View Bibliographic Entry	
SCENARIO:	View bibliographic entry of a reference	
TRIGGERING EVENT:	Registered Virholex user clicks the link of a reference entry	
BRIEF DESCRIPTION:	Registered Virholex user clicks the link of a reference entry. The system then displays the bibliographical details of that particular reference entry.	
ACTORS:	Registered User, Restricted User, Collection Contributor, Collection Coordinator	
RELATED USE CASES:	Includes: *Browse References*	
PRECONDITIONS:	User must be a registered Virholex user	
POST CONDITIONS:	The bibliographic entries of the reference are displayed	
FLOW OF EVENTS:	**Actor**	**System**
	1. Registered Virholex user goes to the page of a particular reference (see *Browse References*)	1.1. Display the bibliographic entries of the reference
EXCEPTION CONDITIONS:	None	

Table 7 Detailed description of View Bibliographic Entry Use Case


sd View Bibliographic Entry
RegisteredUser
View
ViewReference
VirhoDatabase
1: view reference details
1.1: getValues(reference)
1.1.1: build SQL query
1.1.2: SQL select statement
1.1.3: SQL rows
1.2: resultSet
1.2.1: reference bibliographic entry

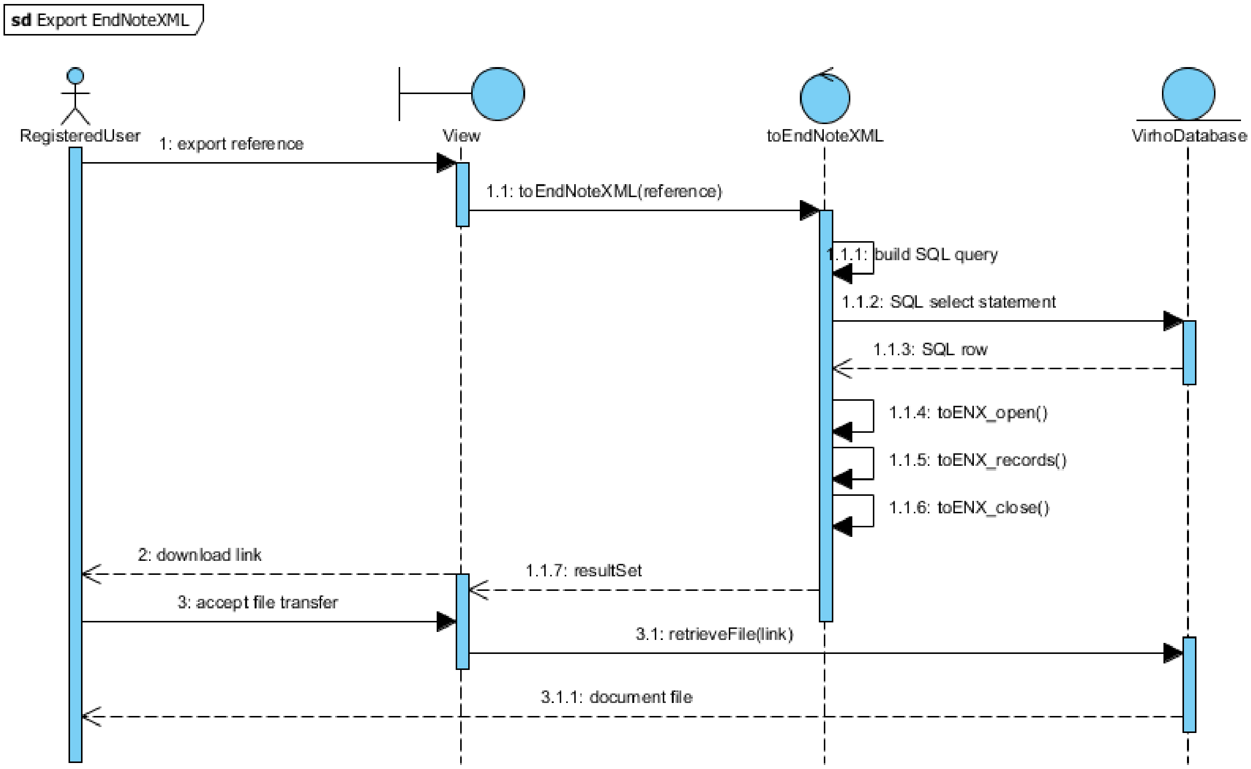
<table>
<tr><td><b>USE CASE NAME:</b></td><td colspan="2">Download reference document</td></tr>
<tr><td><b>SCENARIO:</b></td><td colspan="2">Download the actual document of a reference</td></tr>
<tr><td><b>TRIGGERING EVENT:</b></td><td colspan="2">Registered Virholex user clicks the Download link of a reference entry</td></tr>
<tr><td><b>BRIEF DESCRIPTION:</b></td><td colspan="2">Registered Virholex user clicks the download link of a reference entry. The system then sends to the user the appropriate document file.</td></tr>
<tr><td><b>ACTORS:</b></td><td colspan="2">Restricted User, Collection Contributor, Collection Coordinator</td></tr>
<tr><td><b>RELATED USE CASES:</b></td><td colspan="2">Includes: <i>View Bibliographic Entry</i></td></tr>
<tr><td><b>PRECONDITIONS:</b></td><td colspan="2">User must be at least a level 2 Virholex user</td></tr>
<tr><td><b>POST CONDITIONS:</b></td><td colspan="2">The document file of the reference entry is provided</td></tr>
<tr><td rowspan="2"><b>FLOW OF EVENTS:</b></td><td><b>Actor</b></td><td><b>System</b></td></tr>
<tr><td>1. Registered Virholex user clicks the download link in the page of the reference entry (see <i>View Bibliographic Entry</i>)
2. Registered Virholex user accepts the file transfer</td><td>1.1. The system provides a link for the user to download the file from
2.1. The system sends the appropriate document file and copies the file to the local drive of the user</td></tr>
<tr><td><b>EXCEPTION CONDITIONS:</b></td><td colspan="2">1.1 If the user is a Registered user, the download link will be inactive
2.1 If the download fails, the user will be notified</td></tr>
</table>

Table 8 Detailed description of Download Reference Document Use Case



<table>
<tr><td><b>USE CASE NAME:</b></td><td colspan="2">Export EndNoteXML/BibTex</td></tr>
<tr><td><b>SCENARIO:</b></td><td colspan="2">Export the EndNoteXML/BibTex file</td></tr>
<tr><td><b>TRIGGERING EVENT:</b></td><td colspan="2">Registered Virholex user marks the checkbox of the reference entry and clicks the Export link</td></tr>
<tr><td><b>BRIEF DESCRIPTION:</b></td><td colspan="2">Registered Virholex user clicks the export link of a reference entry. The system then sends to the user the appropriate XML file.</td></tr>
<tr><td><b>ACTORS:</b></td><td colspan="2">Restricted User, Collection Contributor, Collection Coordinator</td></tr>
<tr><td><b>RELATED USE CASES:</b></td><td colspan="2">Includes: <i>View Bibliographic Entry</i></td></tr>
<tr><td><b>PRECONDITIONS:</b></td><td colspan="2">User must be at least a level 2 Virholex user</td></tr>
<tr><td><b>POST CONDITIONS:</b></td><td colspan="2">The document file of the reference entry is provided</td></tr>
<tr><td rowspan="2"><b>FLOW OF EVENTS:</b></td><td><b>Actor</b></td><td><b>System</b></td></tr>
<tr><td>1. Registered Virholex user clicks the export link in the page of the reference entry (see <i>View Bibliographic Entry</i>)
2. Registered Virholex user accepts the file transfer</td><td>1.1. The system provides a link for the user to download the XML file from
2.1. The system sends the appropriate XML file and copies the file to the local drive of the user</td></tr>
<tr><td><b>EXCEPTION CONDITIONS:</b></td><td colspan="2">1.1 If the user is a Registered user, the export link will be inactive
2.1 If the download fails, the user will be notified</td></tr>
</table>

Table 9 Detailed description of Export EndNoteXML/BibTex Use Case



Virho Hotspots

The use case diagrams below illustrate the functionalities of the Virho Hotspots feature available to each user level. Registered and unregistered Virholex users both have access to the image hotspots. However, only level 4 (Collection coordinator) users as defined by the Registered Users Services can upload new hotspot diagrams, and add, delete, and edit hotspot information.

The figure below is the use case diagram together with the use case description and sequence diagrams of the Virho Image Hotspots feature taken from the VirHoLex Functional Specification for Release 1.0 [3].

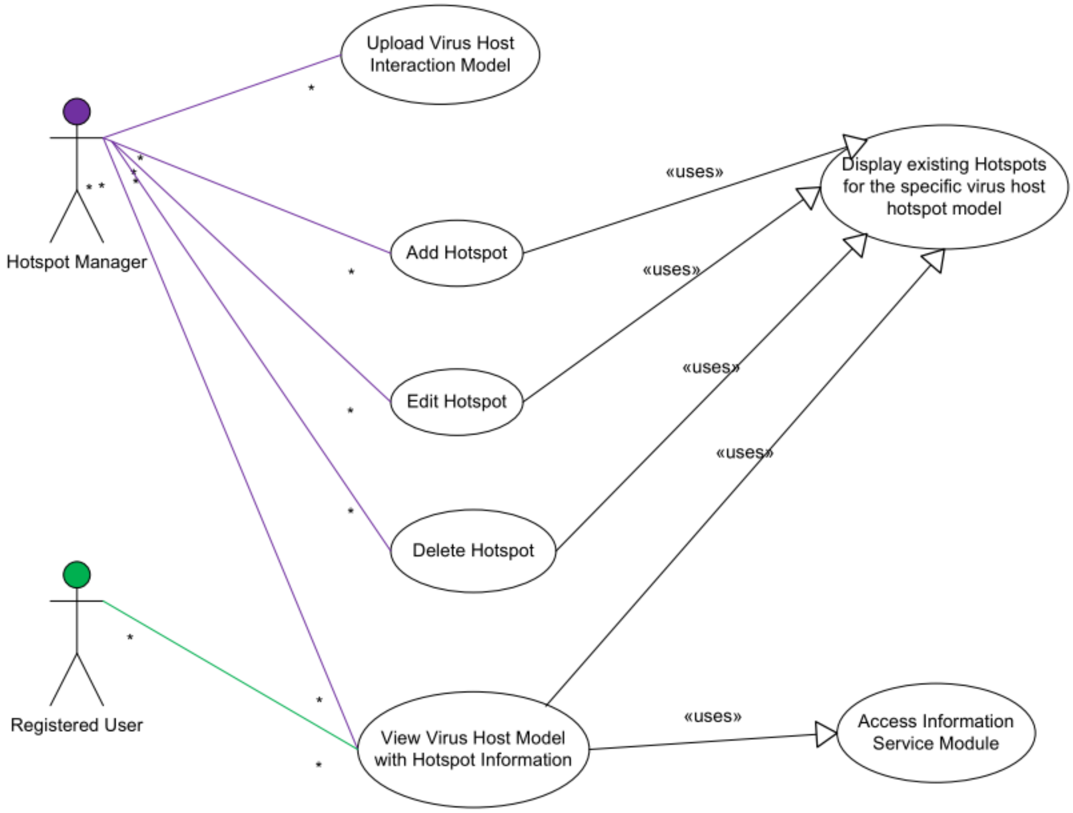
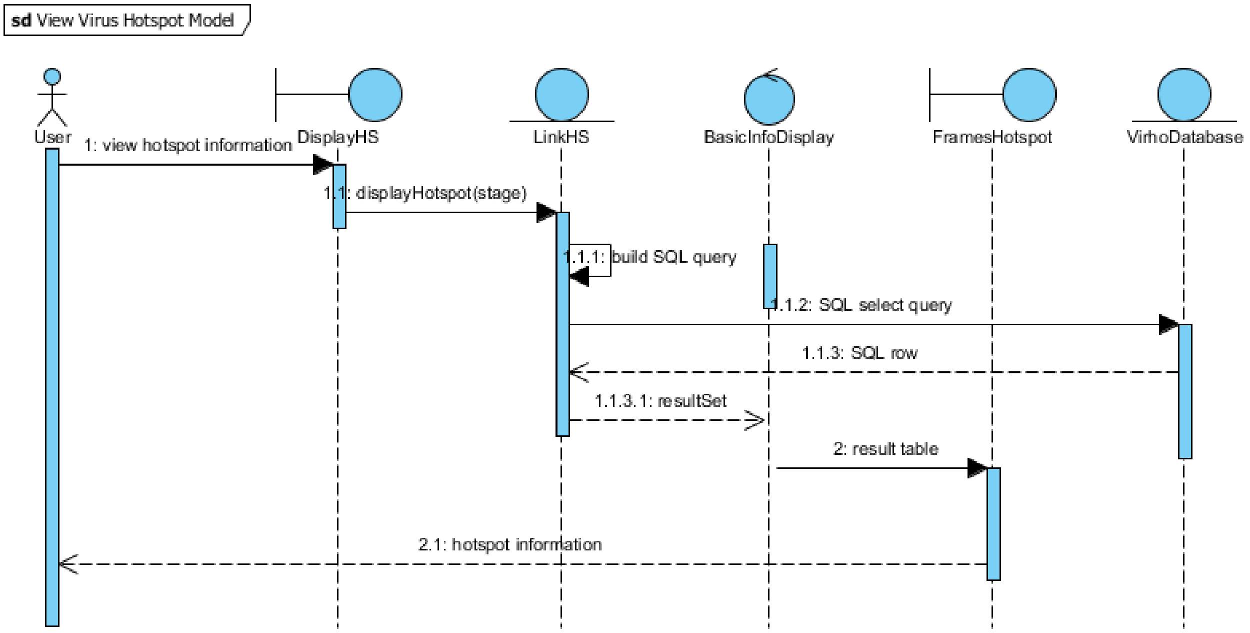

Upload Virus Host Interaction Model
Hotspot Manager
Add Hotspot
Edit Hotspot
Delete Hotspot
Display existing Hotspots for the specific virus host hotspot model
«uses»
«uses»
«uses»
«uses»
Registered User
View Virus Host Model with Hotspot Information
«uses»
Access Information Service Module
*

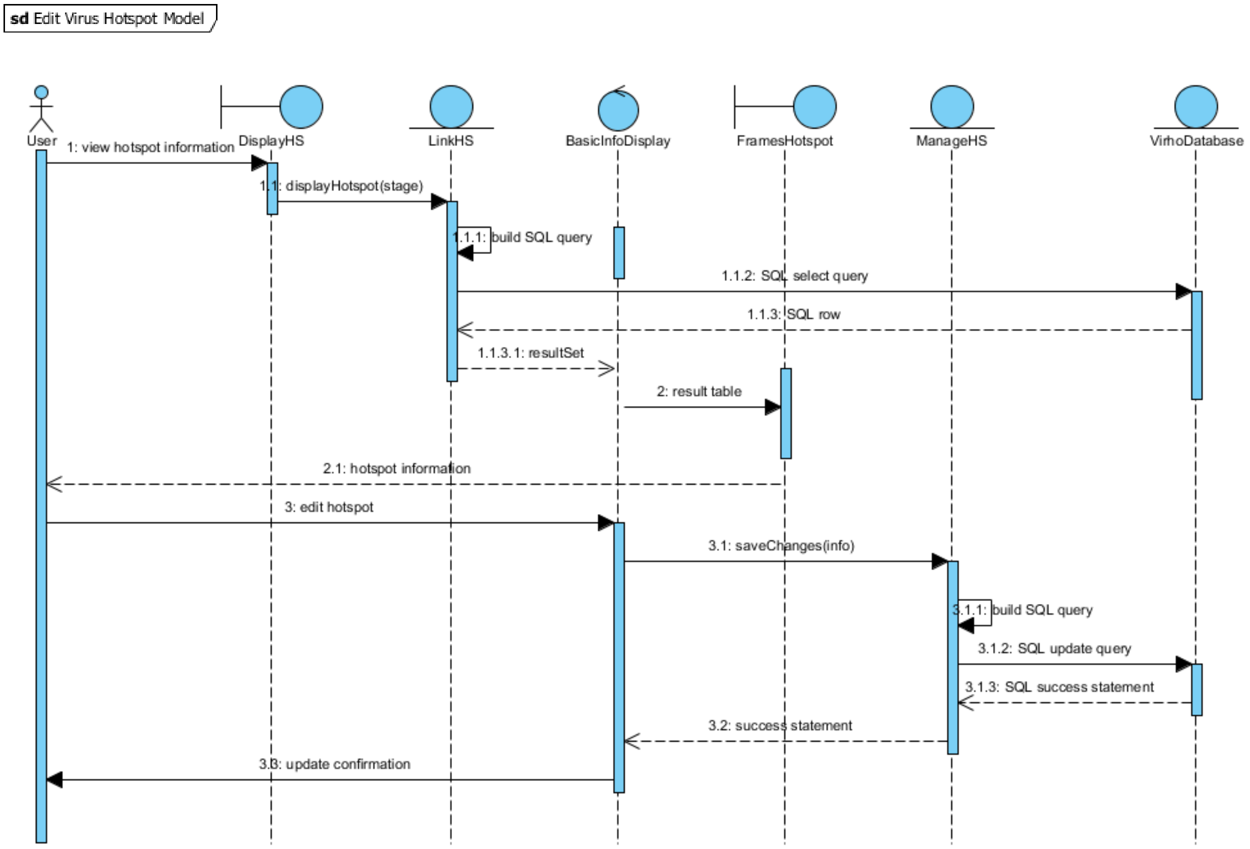
Figure 6 Virho Hotspots Use Case Diagram

USE CASE NAME:	Manage Hotspot	
SCENARIO:	Manage hotspot information of the Virus-Host Interaction model	
TRIGGERING EVENT:	User clicks on the Edit Hotspot button	
BRIEF DESCRIPTION:	When the user clicks on the Update Hotspot Button, he/she will be redirected to the Virho Hotspot Manager Tool page and will be able to Add, Edit or Delete hotspot and related information for each infection step.	
ACTORS:	Hotspot Manager	
RELATED USE CASES:	None	
PRECONDITIONS:	User must be a Hotspot Manager	
POST CONDITIONS:	Update hotspot database Update Virus Host model display	
FLOW OF EVENTS:	**Actor**	**System**
	1. Registered Virholex Hotspot Manager logs on to the system 2. Registered Virholex Hotspot Manager chooses the desired virus	1.1. Redirect to the Virholex home page 2.1. Display the model of the virus host together with the corresponding information

	3. Registered Virholex Hotspot Manager clicks on the edit hotspot button 4. Registered Virholex Hotspot Manager performs desired operations	3.1. Redirect to a page displaying the hotspot model details in editable mode 4.1. Update the hotspot database 4.2. Updates Virus-Host Interaction Main Page
EXCEPTION CONDITIONS:	1.1 If the user is not a Hotspot Manager, then the system pauses this use-case 2.2 If the model of the virus host is not available, ask the user to upload the model instead.	

Table 10 Detailed description of Manage Hotspot Use Case





Technical Architecture

The following will be used in the development of VirHolex:

- *Database*: MySQL
- *Web server*: Apache Tomcat
- *Programming language*: Java Servlet and Java Server Pages (JSP)
- *IDE*: Web Tools Project (WTP) by the Eclipse Foundation

V. RESULTS

Upon testing, several bugs were found in the initial version of the Virus-Host Interaction Lexicon system. While there were several modules that have errors, this study will focus on two modules – the Hotspots and References modules. The References module has a few minor bugs, though several major errors also exist. These bugs hamper the usability of the said module.

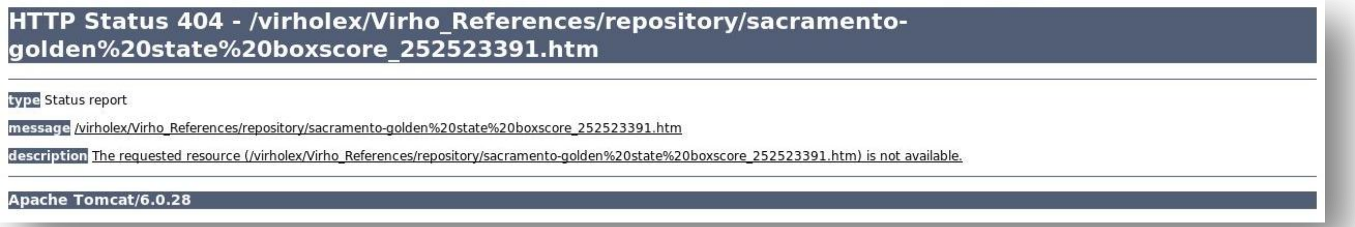
The Hotspot module meanwhile contains some of the biggest bug. A major bug is the inability to add and delete hotspots for a virus diagram. This severely limits the functionality of the module. Another shortcoming of the system is in its use of a generic image for all virus diagrams. This results in an inaccurate illustration of the viruses stored in the system database. Aside from refactoring the code, fixing these bugs is one of the main focus of this study.

Below is the complete listing of the bugs and errors found in the Virho References and Virho Hotspots module together with their descriptions.

Virho References

1. Missing file for reference entry/File Not Found error

   A user cannot download the file related to reference entry that is supposed to be stored in the server. This can be attributed to the fact that the system fails to save the said file when it is uploaded by a user.



HTTP Status 404 - /virholex/Virho_References/repository/sacramento-golden%20state%20boxscore_252523391.htm

type Status report

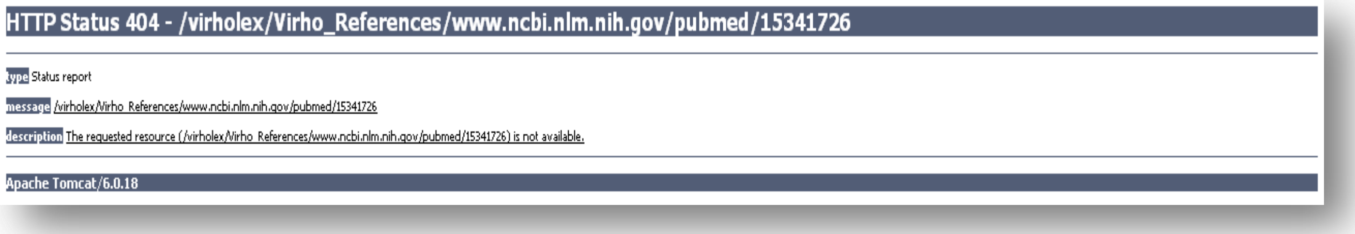
message /virholex/Virho_References/repository/sacramento-golden%20state%20boxscore_252523391.htm

description The requested resource (/virholex/Virho_References/repository/sacramento-golden%20state%20boxscore_252523391.htm) is not available.

Apache Tomcat/6.0.28

2. Wrong hyperlink address for a reference entry

   If a reference entry has a hyperlink for an external webpage, the system prepends its own web address to the link address. This results in a wrong hyperlink address which, when clicked, would give a Page Not Found error.



HTTP Status 404 - /virholex/Virho_References/www.ncbi.nlm.nih.gov/pubmed/15341726

type Status report

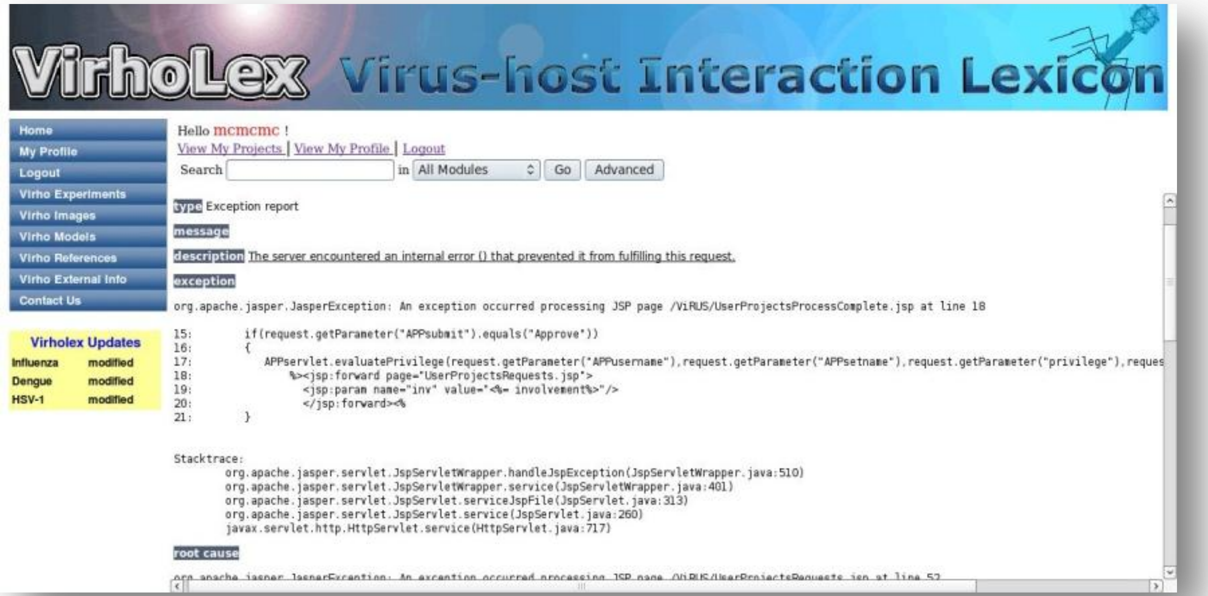
message /virholex/Virho_References/www.ncbi.nlm.nih.gov/pubmed/15341726

description The requested resource (/virholex/Virho_References/www.ncbi.nlm.nih.gov/pubmed/15341726) is not available.

Apache Tomcat/6.0.18

3. Membership approval

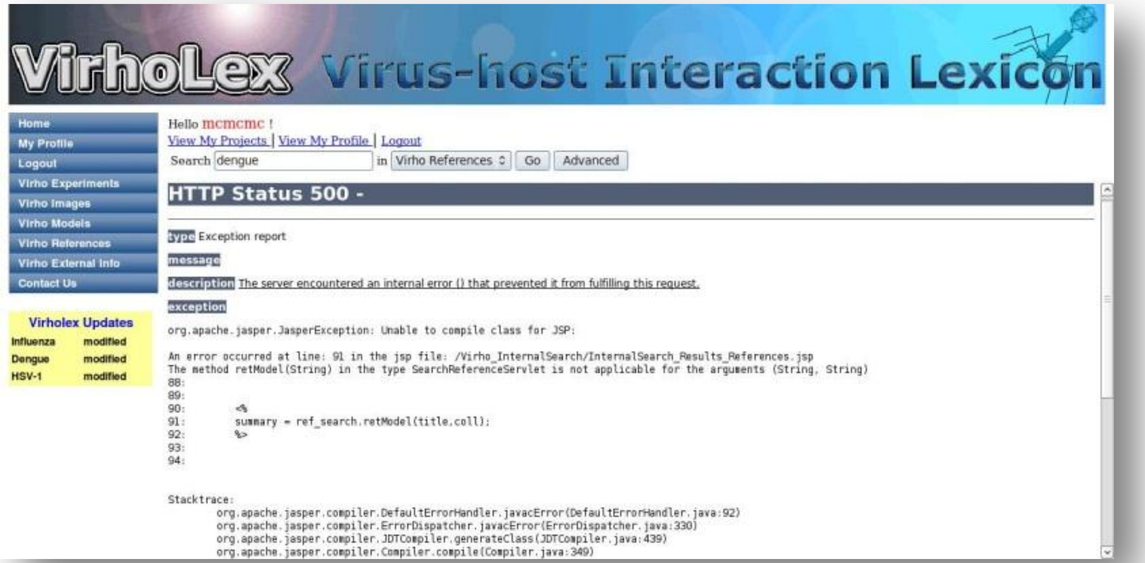
   This bug is specifically for Collection Coordinator accounts. When a Collection Coordinator (*Level 4*) attempts to approve a membership request to a particular collection, the system throws an Exception. This bug appears randomly even during normal usage of the system.



```
type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

org.apache.jasper.JasperException: An exception occurred processing JSP page /ViRUS/UserProjectsProcessComplete.jsp at line 18

15:         if(request.getParameter("APPsubmit").equals("Approve"))
16:         {
17:            APPservlet.evaluatePrivilege(request.getParameter("APPusername"),request.getParameter("APPsetname"),request.getParameter("privilege"),reques
18:                %><jsp:forward page="UserProjectsRequests.jsp">
19:                   <jsp:param name="inv" value="<%= involvement%>"/>
20:                   </jsp:forward><%
21:         }


Stacktrace:
        org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:510)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:401)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:313)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:260)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:717)

root cause

org.apache.jasper.JasperException: An exception occurred processing JSP page /ViRUS/UserProjectsRequests.jsp at line 52
```

4. Internal Search for Virho Reference

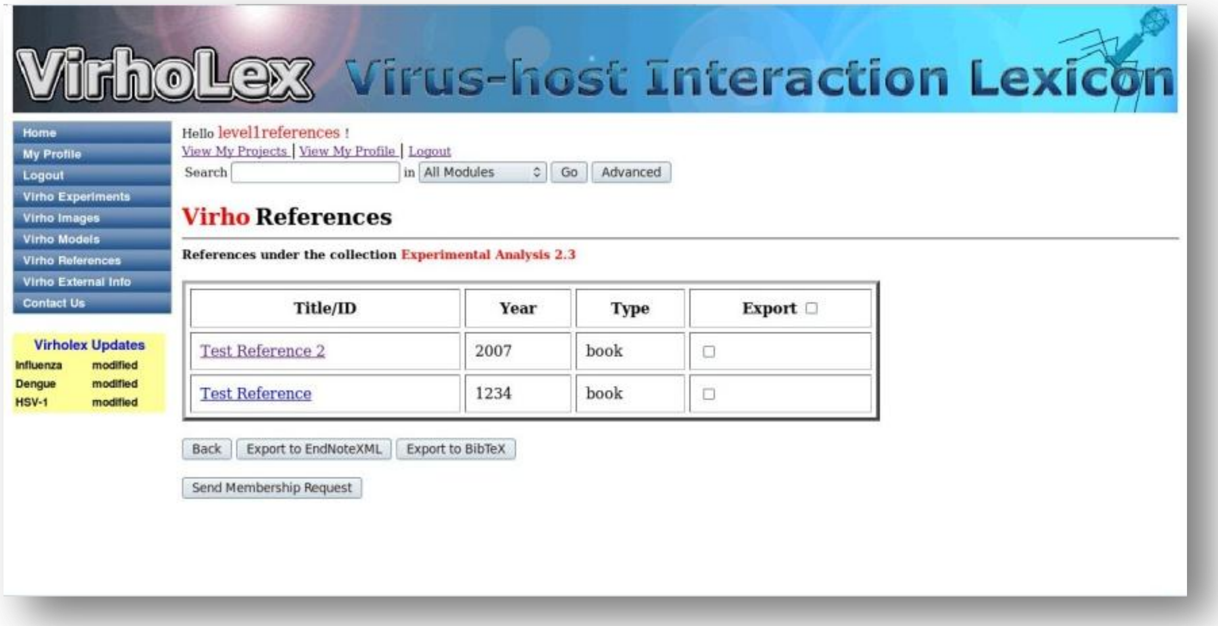
When a user searches a key word that is contained in any collection or reference detail, the system throws an Exception. This does not happen when the key word is not found in any collection information or reference entry stored.



```
HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: 91 in the jsp file: /Virho_InternalSearch/InternalSearch_Results_References.jsp
The method retModel(String) in the type SearchReferenceServlet is not applicable for the arguments (String, String)
88:
89:
90:         <%
91:         summary = ref_search.retModel(title,coll);
92:         %>
93:
94:


Stacktrace:
        org.apache.jasper.compiler.DefaultErrorHandler.javacError(DefaultErrorHandler.java:92)
        org.apache.jasper.compiler.ErrorDispatcher.javacError(ErrorDispatcher.java:330)
        org.apache.jasper.compiler.JDTCompiler.generateClass(JDTCompiler.java:439)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:349)
```

5. Registered User can perform actions reserved for Restricted User and higher level users

   A Registered User (*Level 1*) can download the file related to the reference and can export bibliographic entries to *EndNoteXML* format or *Bibtext* format. These actions are reserved for Restricted User (*Level 2*), Collection Contributor (*Level 3*), and Collection Coordinator (*Level 4*) accounts only.



Hello level1references !

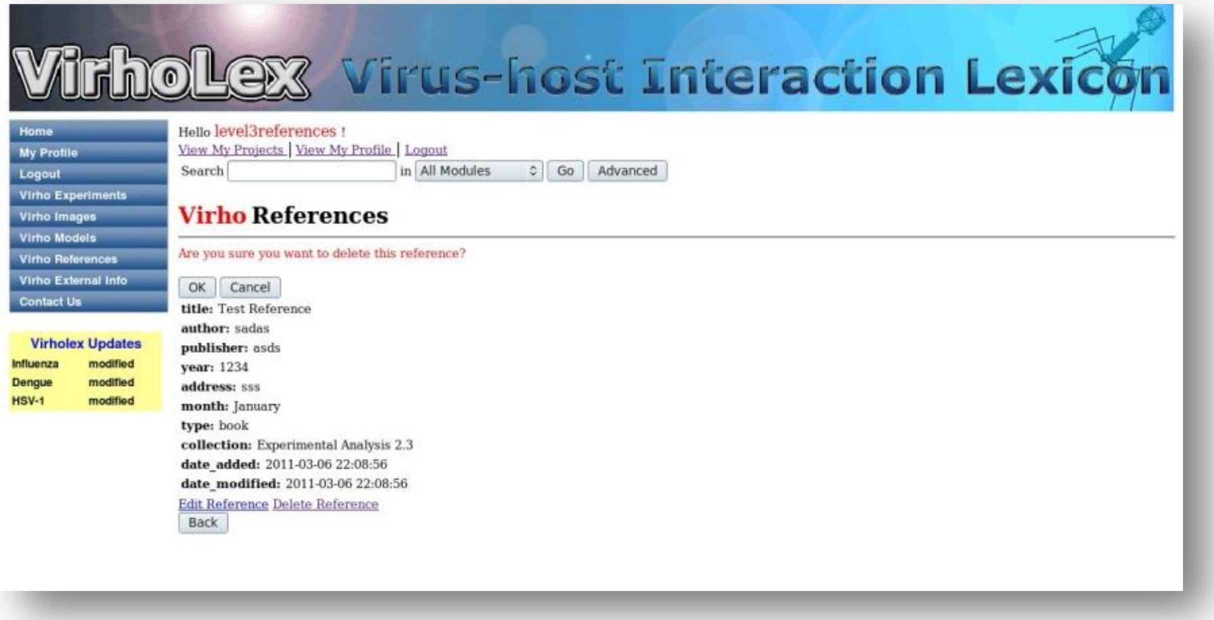
Virho References

References under the collection Experimental Analysis 2.3

Title/ID	Year	Type	Export □
Test Reference 2	2007	book	□
Test Reference	1234	book	□

6. Collection Contributor can delete reference entries

   A Collection Contributor (*Level 3*) can delete an entry even though this privilege is reserved for the Collection Coordinator (*Level 4*).



Hello level3references !

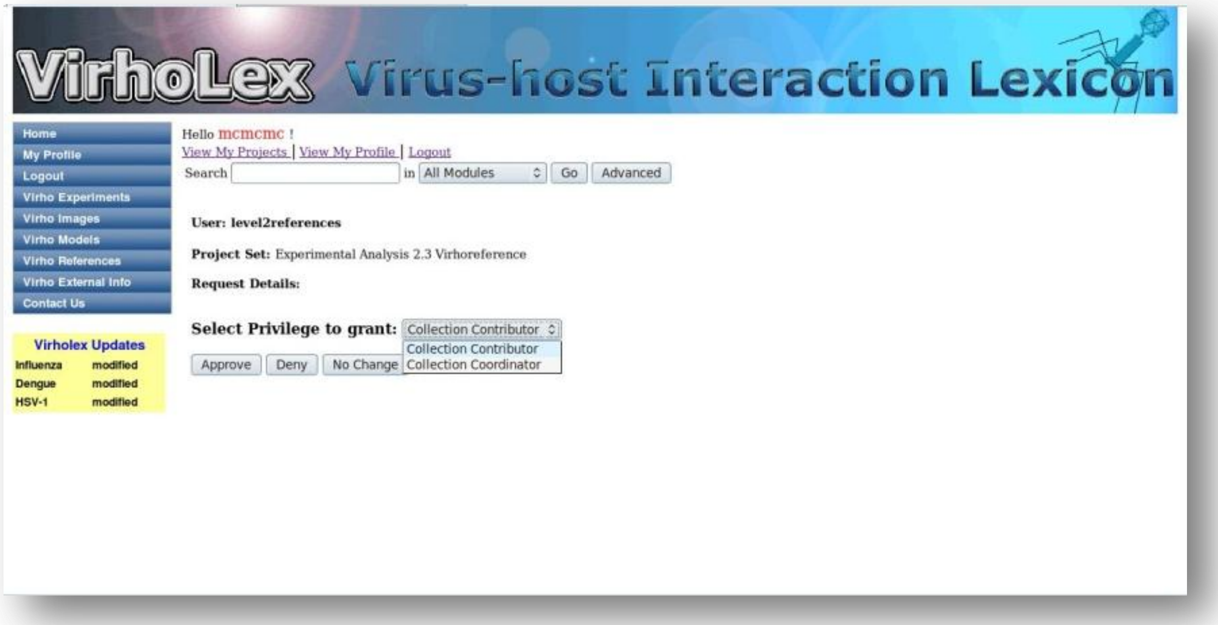
Virho References

Are you sure you want to delete this reference?

title: Test Reference
author: sadas
publisher: asds
year: 1234
address: sss
month: January
type: book
collection: Experimental Analysis 2.3
date_added: 2011-03-06 22:08:56
date_modified: 2011-03-06 22:08:56

7. No Restricted User

The module does not recognize the Restricted User (*Level 2*) role. As a result, a Collection Coordinator cannot assign the said role to a member user.



Hello mcmcmc !

User: level2references

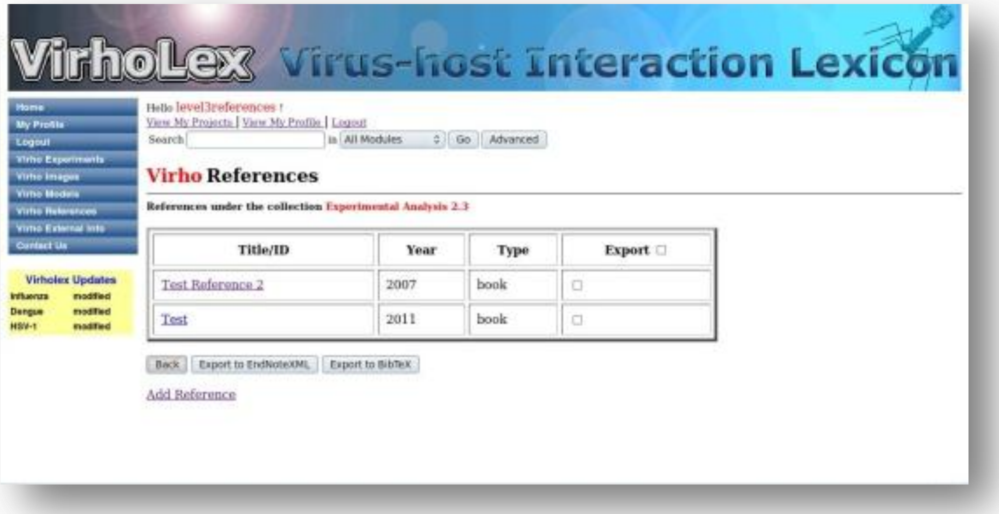
Project Set: Experimental Analysis 2.3 Virhoreference

Request Details:

Select Privilege to grant: Collection Contributor

8. Navigation errors

The *Back*, *Ok*, and *Cancel* buttons in the collection listing page, reference details page and edit membership pages bring the user back and forth between two pages instead of going back to the previous pages. This is a minor issue.

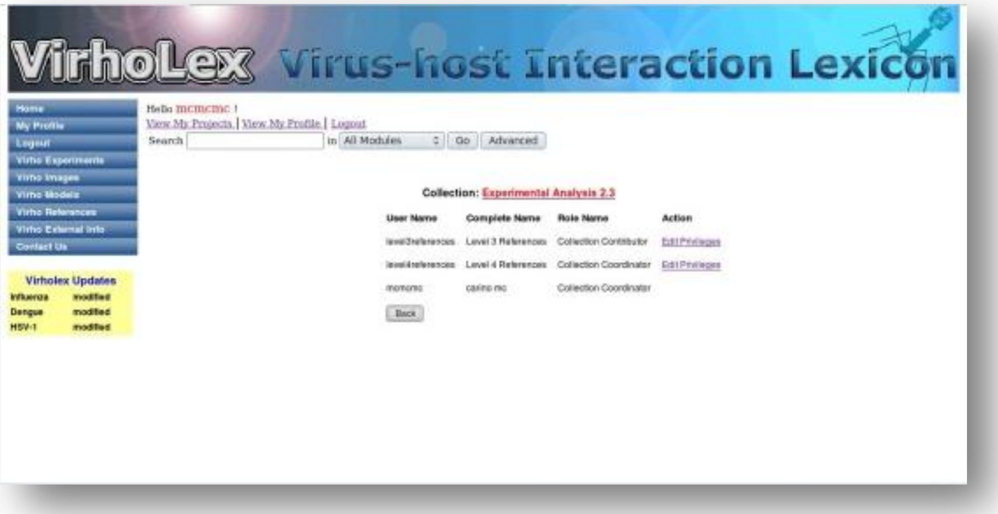


Hello level3references !

Virho References

References under the collection Experimental Analysis 2.3

Title/ID	Year	Type	Export □
Test Reference 2	2007	book	□
Test	2011	book	□



Hello mcmcmc !

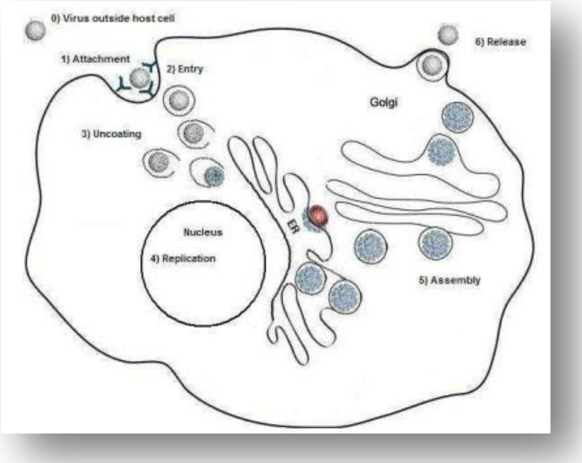
Collection: Experimental Analysis 2.3

User Name	Complete Name	Role Name	Action
level3references	Level 3 References	Collection Contributor	Edit Privileges
level4references	Level 4 References	Collection Coordinator	Edit Privileges
mcmcmc	carino mc	Collection Coordinator	

Virho Hotspots

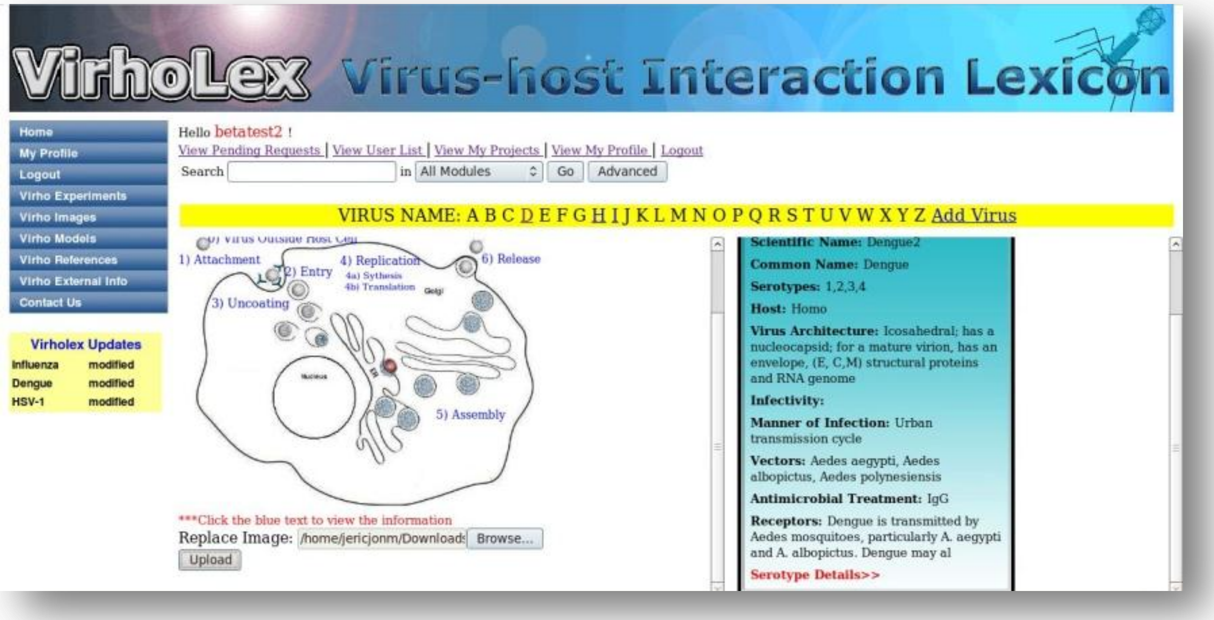
1. Virus diagrams cannot be changed

The system uses the same diagram for all viruses, regardless whether it is accurate or not. More so, the hotspots in the diagram are hard coded into the diagram.



2. Can’t upload a new diagram for a virus

   Since the system uses the same diagram for all viruses, the user cannot upload a new diagram for a particular virus. While the system does not throw errors upon completion of upload, the system still displays the old diagram instead of the new one.



***Click the blue text to view the information

Replace Image: /home/jericjonm/Download

Scientific Name: Dengue2

Common Name: Dengue

Serotypes: 1,2,3,4

Host: Homo

Virus Architecture: Icosahedral; has a nucleocapsid; for a mature virion, has an envelope, (E, C,M) structural proteins and RNA genome

Infectivity:

Manner of Infection: Urban transmission cycle

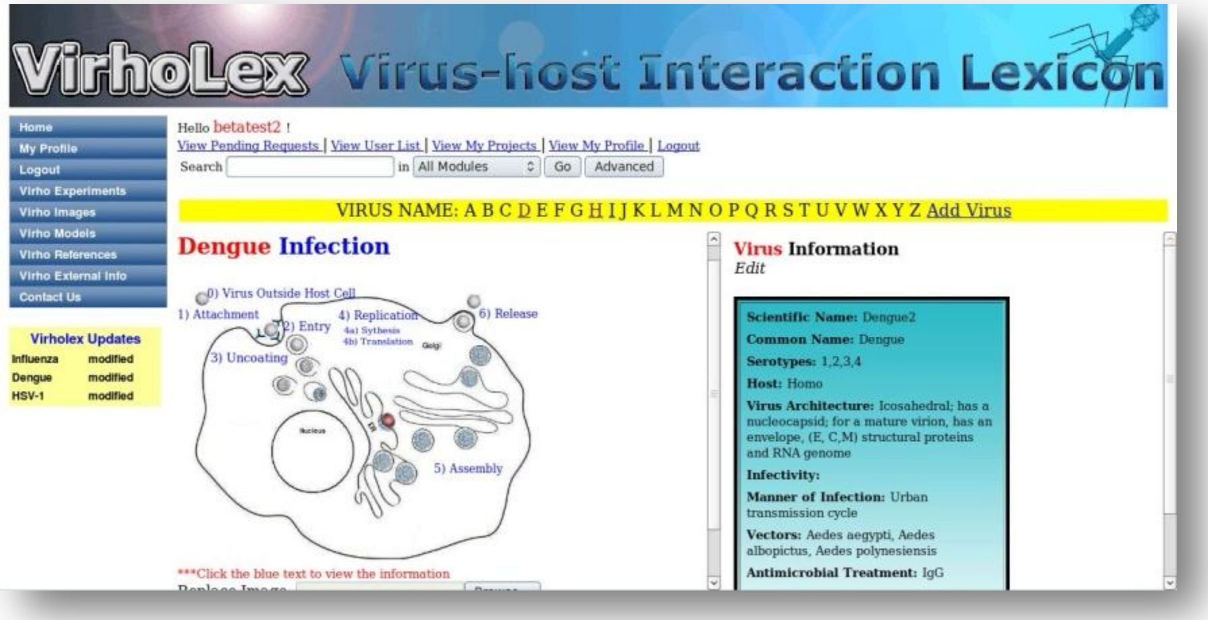
Vectors: Aedes aegypti, Aedes albopictus, Aedes polynesiensis

Antimicrobial Treatment: IgG

Receptors: Dengue is transmitted by Aedes mosquitoes, particularly A. aegypti and A. albopictus. Dengue may al

3. Add hotspot, edit hotspot and delete hotspot are not functional

   While the user can edit virus hotspot information, he/she cannot add, edit or delete a hotspot in the said diagram.



***Click the blue text to view the information

Scientific Name: Dengue2

Common Name: Dengue

Serotypes: 1,2,3,4

Host: Homo

Virus Architecture: Icosahedral; has a nucleocapsid; for a mature virion, has an envelope, (E, C,M) structural proteins and RNA genome

Infectivity:

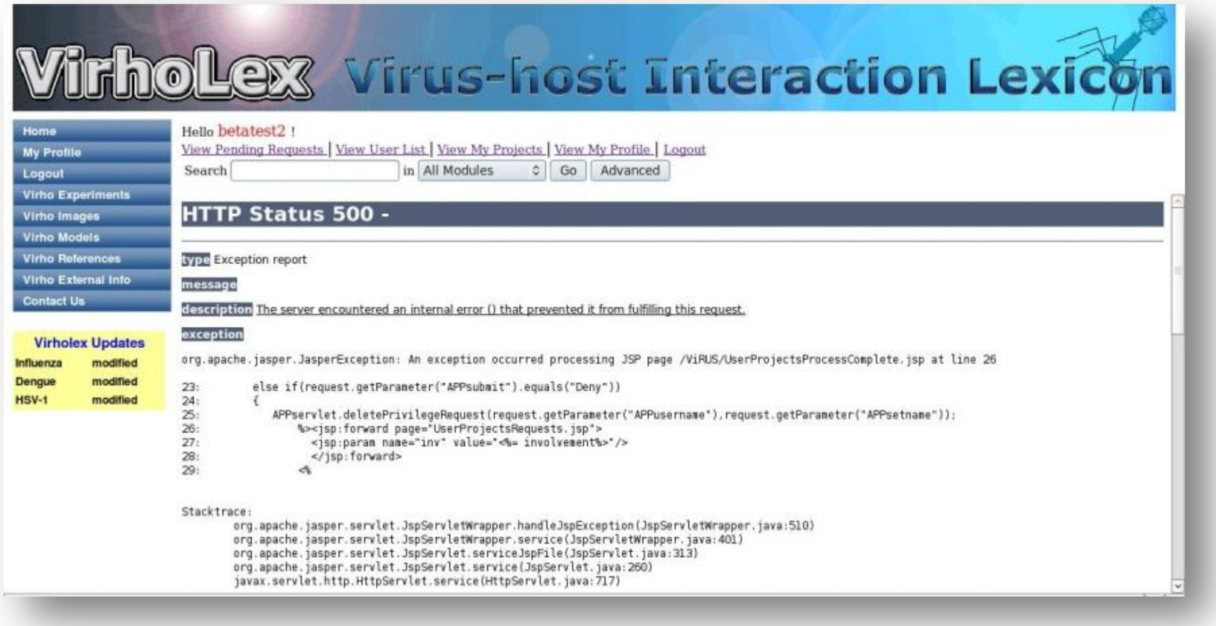
Manner of Infection: Urban transmission cycle

Vectors: Aedes aegypti, Aedes albopictus, Aedes polynesiensis

Antimicrobial Treatment: IgG

4. Membership approval

This bug is specifically for Hotspot Manager accounts. When a Hotspot Manager (*Level 4*) attempts to approve a membership request to a particular virus set, the system throws an Exception. This bug appears randomly even during normal usage of the system.



HTTP Status 500 -

type Exception report

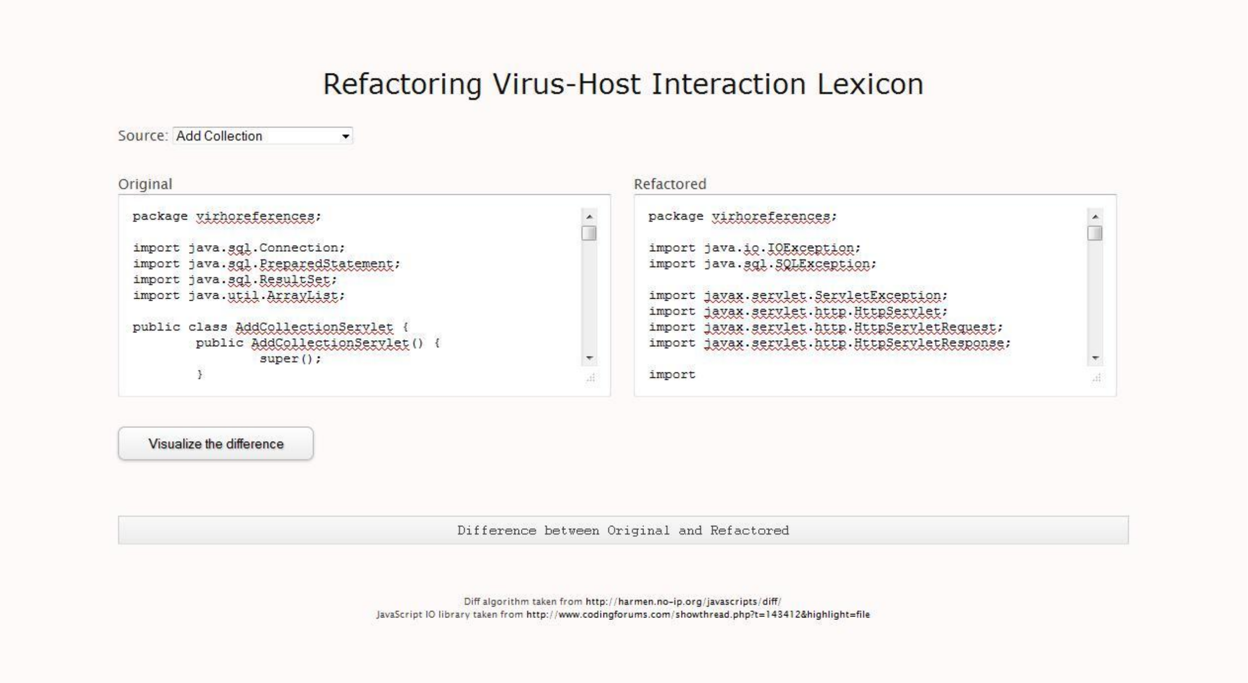
message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

```
org.apache.jasper.JasperException: An exception occurred processing JSP page /ViRUS/UserProjectsProcessComplete.jsp at line 26

23:         else if(request.getParameter("APPsubmit").equals("Deny"))
24:         {
25:            APPservlet.deletePrivilegeRequest(request.getParameter("APPusername"),request.getParameter("APPsetname"));
26:                %><jsp:forward page="UserProjectsRequests.jsp">
27:                  <jsp:param name="inv" value="<%= involvement%>"/>
28:                  </jsp:forward>
29:                <%


Stacktrace:
        org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:510)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:401)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:313)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:260)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
```

To help compare the original code from the refactored code, a simple web application was created. It shows the two codes side by side in a page and loads these in a visual diff algorithm similar to the diff program used in UNIX systems.



Refactoring Virus-Host Interaction Lexicon

Source: Add Collection

Original

```
package virhoreferences;

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

public class AddCollectionServlet {
        public AddCollectionServlet() {
                super();
        }
```

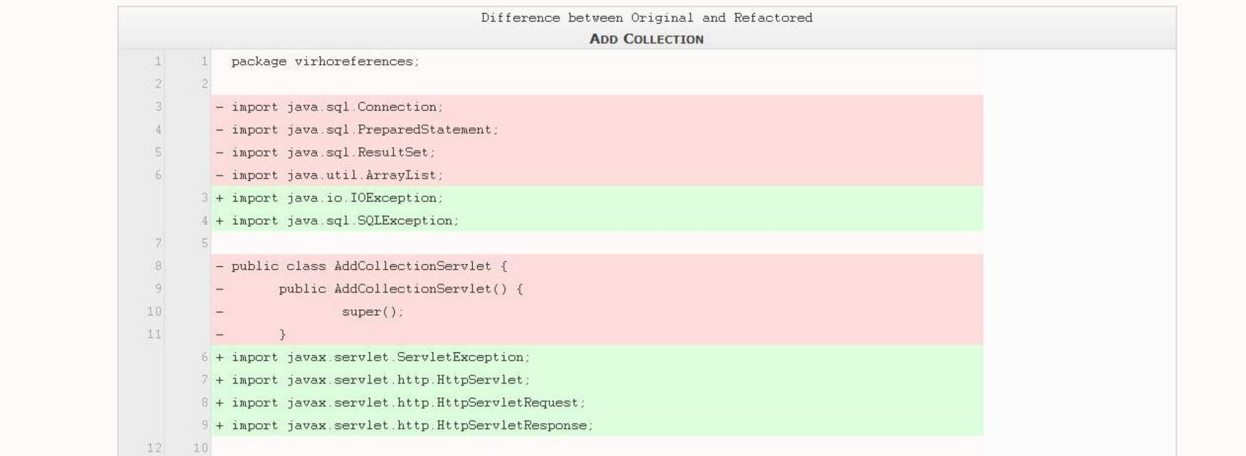
Refactored

```
package virhoreferences;

import java.io.IOException;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import
```

Difference between Original and Refactored

Diff algorithm taken from http://harmen.no-ip.org/javascripts/diff/
JavaScript IO library taken from http://www.codingforums.com/showthread.php?t=143412&highlight=file

The resulting table shows the comparison of the two codes, with those colored *pink* as the lines removed from the original code, and those colored *green* as the lines of code added to the refactored code.

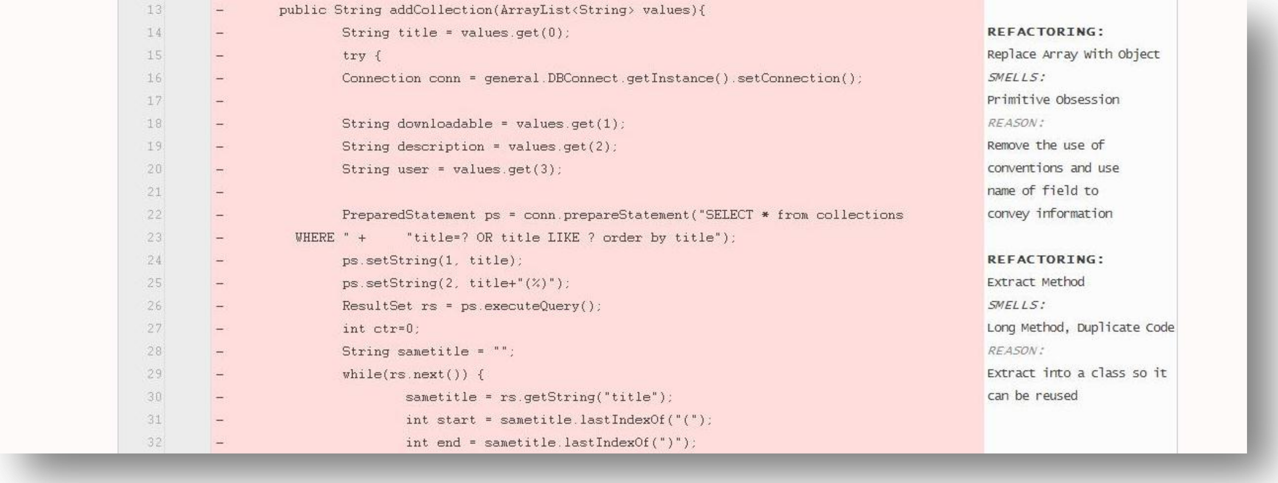


Difference between Original and Refactored

ADD COLLECTION

```
  package virhoreferences;

- import java.sql.Connection;
- import java.sql.PreparedStatement;
- import java.sql.ResultSet;
- import java.util.ArrayList;
+ import java.io.IOException;
+ import java.sql.SQLException;

- public class AddCollectionServlet {
-       public AddCollectionServlet() {
-               super();
-       }
+ import javax.servlet.ServletException;
+ import javax.servlet.http.HttpServlet;
+ import javax.servlet.http.HttpServletRequest;
+ import javax.servlet.http.HttpServletResponse;

```

The refactoring technique, code smell found and the reason for refactoring is also added beside lines where the said technique was applied.



```
-       public String addCollection(ArrayList<String> values){
-               String title = values.get(0);
-               try {
-               Connection conn = general.DBConnect.getInstance().setConnection();
-
-               String downloadable = values.get(1);
-               String description = values.get(2);
-               String user = values.get(3);
-
-               PreparedStatement ps = conn.prepareStatement("SELECT * from collections
-         WHERE " +      "title=? OR title LIKE ? order by title");
-               ps.setString(1, title);
-               ps.setString(2, title+"(%)");
-               ResultSet rs = ps.executeQuery();
-               int ctr=0;
-               String sametitle = "";
-               while(rs.next()) {
-                       sametitle = rs.getString("title");
-                       int start = sametitle.lastIndexOf("(");
-                       int end = sametitle.lastIndexOf(")");
```

REFACTORING:
Replace Array with Object
SMELLS:
Primitive Obsession
REASON:
Remove the use of conventions and use name of field to convey information

REFACTORING:
Extract Method
SMELLS:
Long Method, Duplicate Code
REASON:
Extract into a class so it can be reused

The refactorings done were able to fix the bugs in Virho References and Virho Hotspots module. However, for the Virho Hotspots module, the accepted image type for the virus diagram was changed from an SVG file to any of the more common image file type (*jpeg*, *gif*, *png*, *bmp*, *and the like)*.

The following tables show the refactorings done for each functionality as well as their corresponding code smell.

Add Reference

while (itr.hasNext()) {
FileItem item = (FileItem) itr.next();
if (item.isFormField()){
String name = item.getFieldName();
String value = item.getString();
 	
if(name.equals("collection"))
collection = value;
 	
else if(name.equals("down"))
download = value;
 	
else if(name.equals("filename")) {
 	
if (pathname.equals(""))
filename = value;
}
 	
else if(name.equals("type"))
type = value;
 	
else if(name.equals("author"))
author = value;
 	
…
…
…
 	
else if(name.equals("year"))
year = value;
	HashMap<String, String> formContents = new HashMap<String, String>();
	while (itr.hasNext()) {
	FileItem item = itr.next();
	if (item.isFormField()) {
	String name = item.getFieldName();
	String value = item.getString();
	
	if (!name.equalsIgnoreCase("submit"))
	formContents.put(name, value);
	
	if(name.equalsIgnoreCase("collection"))
	collection = value;
	
	}else { //image upload
	if(item.getSize() > 0) {
	String imageRoot = File.separator +
	"Virho_References" + File.separator + "repository";
	int extension = item.getName().lastIndexOf(".");
	String filename = item.getName().substring(0, extension) + "_"
	+ Math.abs(Math.random()) +
	item.getName().substring(extension);
	String pathname =
	request.getSession().getServletContext().getRealPath(imageRoot)
	+ File.separator + filename;
	formContents.put("path", filename);
	File uploadedFile = new File(pathname);
	item.write(uploadedFile);
		Decompose Conditional	Switch Statement, Long Method

String no = "no";
address="Virho_References/ViewReferenceDetails.jsp?ref="
+ title + "&coll=" + collection + "&down=" + download +
"&back=" + no;
	}
	}
	}
	
	return ReferencesDB.updateDB(formContents,
	ReferencesDB.REFERENCE_ADD);		

conn = general.DBConnect.getInstance().setConnection();
s = conn.createStatement ();
s.execute ("INSERT into coll_reference (collection, path,
title, type, author, editor, publisher, year, chapter, pages,
booktitle, school, institution, note, volume, series, address,
edition, " + "month, subtype, organization, number,
howpublished, link,journal, date_added, date_modified) VALUES "
+ "('"+ collection +"', '"   + filename + "','" + title +
"','"+type +"', '"+author +"', '"+editor+"', '"+publisher+"' ,
'"+year+"', '"+chapter+"', '"+pages+"', '"+booktitle+"', '"
+school+"', '"+institution+"', '"+note+ "','" +volume+ "','"
+series+ "','" +addres+ "','" +edition+"','" +month+ "','"
+stype+ "','" +org+ "','" +number+ "','"+howpublished+ "','"
+link+ "','" +journal+ "', NOW(), NOW())");
	result =
	ReferencesDB.viewDB(request.getParameter("coll"),
	request.getParameter("ref")
	…
	…
	…
	
	protected static String updateDB(HashMap<String,
	String> form, int action) throws SQLException,
	FileNotFoundException {
	switch(action) {
	case REFERENCE_ADD: return addToDB(form);
	case REFERENCE_EDIT: return editDB(form);
	default: return null;
	}
	}
		Hide Delegate	Message Chain

filename = "";
pathname = item.getName();
Random generator = new Random();
int randnum = Math.abs(generator.nextInt());
 	
String reg = "[.*]";
String replacingtext = "";
Pattern pattern = Pattern.compile(reg);
Matcher matcher = pattern.matcher(pathname);
StringBuffer buffer = new StringBuffer();
while (matcher.find()) {
matcher.appendReplacement(buffer, replacingtext);
}
 	
int IndexOf = pathname.indexOf(".");
String domainname = pathname.substring(IndexOf);
 	
pathname = buffer.toString();
int LastIndexOf = pathname.lastIndexOf("\\");
filename = pathname.substring(LastIndexOf+1);
filename = filename + "_" + randnum + domainname;
 	
pathname =
request.getSession().getServletContext().getRealPath(imageRoot)
+File.separator + filename;
File savedfile = new File(pathname);
item.write(savedfile);
	String imageRoot = File.separator +
	"Virho_References" + File.separator + "repository";
	int extension = item.getName().lastIndexOf(".");
	String filename = item.getName().substring(0,
	extension) + "_"
	+ Math.abs(Math.random()) +
	item.getName().substring(extension);
	String pathname =
	request.getSession().getServletContext().getRealPath(
	imageRoot)
	+ File.separator + filename;
	formContents.put("path", filename);
	File uploadedFile = new File(pathname);
	item.write(uploadedFile);
		Extract Method	Long Method

Add Collection

String title = values.get(0);
…
…
…
 	
String downloadable = values.get(1);
String description = values.get(2);
String user = values.get(3);
	public class Collection {
	private String title;
	private String id;
	private String description;
	private String downloadable;
	private String user;
	//Parameter for edit action
	private String oldTitle;
	…
	…
	…
	}
		Replace Array With Object	Primitive Obsession

PreparedStatement ps = conn.prepareStatement("SELECT * from
collections
WHERE " + "title=? OR title LIKE ? order by title");
ps.setString(1, title);
ps.setString(2, title+"(%)");
ResultSet rs = ps.executeQuery();
int ctr=0;
String sametitle = "";
while(rs.next()) {
sametitle = rs.getString("title");
int start = sametitle.lastIndexOf("(");
int end = sametitle.lastIndexOf(")");
try{
if(sametitle.equals(title)){
ctr++;
}
else if(Integer.valueOf(sametitle.substring(start+1, end))>0){
ctr = Math.max(ctr,Integer.valueOf(sametitle.substring(start+1,
end))+1);
}
	private static String generateTitle(String title,
	String collection) throws SQLException {
	PreparedStatement ps = conn.prepareStatement("SELECT
	title FROM " +
	"(SELECT title, collection FROM coll_reference coll
	WHERE title LIKE ? OR title=? ORDER BY title) " +
	"AS result WHERE collection =?");
	ps.setString(1, title + "(%)");
	ps.setString(2, title);
	ps.setString(3, collection);
	ResultSet rs = ps.executeQuery();
	
	int ctr = 0;
	while(rs.next()) {
	String fromDatabase = rs.getString("title");
	int start = fromDatabase.lastIndexOf("(");
	int end = fromDatabase.lastIndexOf(")");
	
	…
	…
	…
	
	rs.close();
	ps.close();
	return (ctr > 0) ? title + "(" + ctr + ")" : title;
	}
		Extract Method	Long Method, Duplicate Code

ps = conn.prepareStatement("INSERT INTO project (project_name,
description, author, date_added, date_modified)
VALUES(?,?,?,NOW(),NOW()) ");
	CollectionDB.updateDB(collection,
	CollectionDB.COLLECTION_ADD)
		Hide Delegate	Message Chain

ps.setString (1, title + " VirhoReference");
ps.setString (2, description);
ps.setString (3, user);
ps.executeUpdate ();
ps.close ();
…
…
…
	…
	…
	…
	
	protected static String updateDB(Collection collection, int action) throws SQLException {
	switch(action) {
	case COLLECTION_ADD: return addToDB(collection);
	case COLLECTION_DELETE: return deleteFromDB(collection);
	case COLLECTION_EDIT: return editDB(collection);
	default: return "failed";
	//TODO Default case
	}
	}		

Delete Collection

Connection conn = general.DBConnect.getInstance().setConnection();
 	
PreparedStatement ps = conn.prepareStatement ("DELETE FROM coll_reference WHERE collection=?");
ps.setString(1, title);
ps.executeUpdate ();
ps.close();
 	
ps = conn.prepareStatement ("DELETE FROM collections WHERE title=?");
ps.setString(1, title);
ps.executeUpdate ();
ps.close();
…
…
…
	CollectionDB.updateDB(collection, CollectionDB.COLLECTION_DELETE)
	
	…
	…
	…
	
	protected static String updateDB(Collection collection, int action) throws SQLException {
	switch(action) {
	case COLLECTION_ADD: return addToDB(collection);
	case COLLECTION_DELETE: return deleteFromDB(collection);
	case COLLECTION_EDIT: return editDB(collection);
	default: return "failed";
	//TODO Default case
	}
	}
		Hide Delegate	Message Chain

Delete Reference

conn = general.DBConnect.getInstance().setConnection();
File file = new File(path);
file.delete();
Statement s = conn.createStatement ();
s.execute ("DELETE FROM coll_reference WHERE id='" + id + "'");
Statement st = conn.createStatement ();
st.execute ("UPDATE models SET evidence='' WHERE evidence='"+id+"' ");
PreparedStatement ps = conn.prepareStatement ("UPDATE image SET reference_id=0 WHERE reference_id=?");
ps.setInt(1, id);
ps.executeUpdate();
ps.close ();
s.close();
st.close();
 	
conn.close();
	ReferencesDB.updateDB(ref, coll, ReferencesDB.REFERENCE_DELETE)
	…
	…
	…
	
	protected static String updateDB(String reference, String collection, int action) throws SQLException, FileNotFoundException {
	if (action == REFERENCE_DELETE)
	return deleteFromDB(reference, collection);
	
	return null;
	}
		Hide Delegate	Message Chain

Edit Collection

String downloadable = values.get(1);
String description = values.get(2);
String oldtitle = values.get(4);
	Collection collection = new Collection.Builder(title).oldTitle(oldTitle).downloadable(downloadable).description(description).build();	Replace Array With Object	Primitive Obsession

PreparedStatement ps;
if(!oldtitle.equals(newtitle)){
ps = conn.prepareStatement("SELECT * from collections WHERE " + "title=? OR title LIKE ? order by title");
ps.setString(1, newtitle);
ps.setString(2, newtitle+"(%)");
ResultSet rs = ps.executeQuery();
int ctr=0;
String sametitle = "";
while(rs.next()) {
sametitle = rs.getString("title");
int start = sametitle.lastIndexOf("(");
int end = sametitle.lastIndexOf(")");
try {
if(sametitle.equals(newtitle)){
ctr++;
}
	private static String generateTitle(String title, String collection) throws SQLException {
	PreparedStatement ps = conn.prepareStatement("SELECT title FROM " +
	"(SELECT title, collection FROM coll_reference coll WHERE title LIKE ? OR title=? ORDER BY title) " +
	"AS result WHERE collection =?");
	ps.setString(1, title + "(%)");
	ps.setString(2, title);
	ps.setString(3, collection);
	ResultSet rs = ps.executeQuery();
	
	int ctr = 0;
	while(rs.next()) {
	String fromDatabase = rs.getString("title");
	int start = fromDatabase.lastIndexOf("(");
	int end = fromDatabase.lastIndexOf(")");
	
	…
	…
	…
	
	rs.close();
	ps.close();
	return (ctr > 0) ? title + "(" + ctr + ")" : title;
	}
		Extract Method	Long Method, Duplicate Code

ps = conn.prepareStatement ("UPDATE collections SET title=?, description=?, downloadable=?" + " WHERE title=?");
ps.setString(1, newtitle);
	CollectionDB.updateDB(collection,CollectionDB.COLLECTION_EDIT)
	…
		Hide Delegate	Inappropriate Intimacy

ps.setString(2, description);
ps.setString(3, downloadable);
ps.setString(4, oldtitle);
ps.executeUpdate();
ps.close ();
…
…
…
	…
	…
	protected static String updateDB(Collection collection, int action) throws SQLException {
	switch(action) {
	case COLLECTION_ADD: return addToDB(collection);
	case COLLECTION_DELETE: return deleteFromDB(collection);
	case COLLECTION_EDIT: return editDB(collection);
	default: return "failed";
	//TODO Default case
	}
	}		

Edit Reference

String collection="", download="", filename="", title="", type="", author="", editor="", publisher="", year="", chapter="", pages="";
String booktitle="", school="", institution="", note="", volume="", series="", addres="", edition="", month="";
String stype="", org="", number="", howpublished="", link="", old_id="", journal="";
	while (itr.hasNext()) {
	FileItem item = itr.next();
	String name = item.getFieldName();
	String value = item.getString();
	if (item.isFormField()) {
	if (!name.equalsIgnoreCase("submit"))
	formContents.put(name, value);
	
	if(name.equalsIgnoreCase("collection"))
	collection = value;
		Extract Method	Long Method

while (itr.hasNext()) {
FileItem item = (FileItem) itr.next();
if (item.isFormField()){
String name = item.getFieldName();
String value = item.getString();
 	
if(name.equals("collection"))
collection = value;
 	
else if(name.equals("down"))
download = value;
 	
else if(name.equals("filename")) {
 	
if (pathname.equals(""))
filename = value;
}
 	
else if(name.equals("type"))
type = value;
 	
else if(name.equals("author"))
author = value;
 	
…
…
…
 	
else if(name.equals("year"))
year = value;
 	
String no = "no";
address="Virho_References/ViewReferenceDetails.jsp?ref="
+ title + "&coll=" + collection + "&down=" + download +
"&back=" + no;
	HashMap<String, String> formContents = new HashMap<String, String>();
	while (itr.hasNext()) {
	FileItem item = itr.next();
	if (item.isFormField()) {
	String name = item.getFieldName();
	String value = item.getString();
	
	if (!name.equalsIgnoreCase("submit"))
	formContents.put(name, value);
	
	if(name.equalsIgnoreCase("collection"))
	collection = value;
	
	}else { //image upload
	if(item.getSize() > 0) {
	String imageRoot = File.separator +
	"Virho_References" + File.separator + "repository";
	int extension = item.getName().lastIndexOf(".");
	String filename = item.getName().substring(0, extension) + "_"
	+ Math.abs(Math.random()) +
	item.getName().substring(extension);
	String pathname =
	request.getSession().getServletContext().getRealPath(imageRoot)
	+ File.separator + filename;
	formContents.put("path", filename);
	File uploadedFile = new File(pathname);
	item.write(uploadedFile);
	
	}
	}
	}
	
	return ReferencesDB.updateDB(formContents,
	ReferencesDB.REFERENCE_ADD);
		Decompose Conditional	Switch Statement, Long Method

filename = "";
pathname = item.getName();
Random generator = new Random();
int randnum = Math.abs(generator.nextInt());
 	
String reg = "[.*]";
String replacingtext = "";
Pattern pattern = Pattern.compile(reg);
Matcher matcher = pattern.matcher(pathname);
StringBuffer buffer = new StringBuffer();
while (matcher.find()) {
matcher.appendReplacement(buffer, replacingtext);
}
 	
int IndexOf = pathname.indexOf(".");
String domainname = pathname.substring(IndexOf);
 	
pathname = buffer.toString();
int LastIndexOf = pathname.lastIndexOf("\\");
filename = pathname.substring(LastIndexOf+1);
filename = filename + "_" + randnum + domainname;
 	
pathname =
request.getSession().getServletContext().getRealPath(imageRoot)
+File.separator + filename;
File savedfile = new File(pathname);
item.write(savedfile);
	String imageRoot = File.separator +
	"Virho_References" + File.separator + "repository";
	int extension = item.getName().lastIndexOf(".");
	String filename = item.getName().substring(0, extension) + "_"
	+ Math.abs(Math.random()) +
	item.getName().substring(extension);
	String pathname =
	request.getSession().getServletContext().getRealPath(imageRoot)
	+ File.separator + filename;
	formContents.put("path", filename);
	File uploadedFile = new File(pathname);
	item.write(uploadedFile);
		Extract Method	Long Method

if (item.isFormField()){
String name = item.getFieldName();
String value = item.getString();
	HashMap<String, String> formContents = new HashMap<String, String>();
	while (itr.hasNext()) {
		Decompose Conditional	Switch Statement, Long Method

if(name.equals("old_id"))
old_id = value;
else if(name.equals("title"))
title = value;
else if(name.equals("collection"))
collection = value;
else if(name.equals("down"))
download = value;
else if(name.equals("filename")) {
if (pathname.equals(""))
filename = value;
}
else if(name.equals("type"))
type = value;
else if(name.equals("author"))
author = value;
else if(name.equals("editor"))
editor = value;
else if(name.equals("publisher"))
publisher = value;
else if(name.equals("year"))
year = value;
else if(name.equals("chapter"))
chapter = value;
else if(name.equals("pages"))
pages = value;
else if(name.equals("booktitle"))
booktitle = value;
else if(name.equals("school"))
school = value;
else if(name.equals("institution"))
institution = value;
else if(name.equals("note"))
note = value;
else if(name.equals("volume"))
volume = value;
else if(name.equals("series"))
series = value;
else if(name.equals("addres"))
addres = value;
else if(name.equals("edition"))
edition = value;
else if(name.equals("month"))
month = value;
else if(name.equals("stype"))
stype = value;
else if(name.equals("org"))
org = value;
else if(name.equals("number"))
number = value;
else if(name.equals("howpublished"))
howpublished = value;
else if(name.equals("link"))
link = value;
else if(name.equals("journal"))
journal = value;
}
	FileItem item = itr.next();
	String name = item.getFieldName();
	String value = item.getString();
	if (item.isFormField()) {
	if (!name.equalsIgnoreCase("submit"))
	formContents.put(name, value);
	
	if(name.equalsIgnoreCase("collection"))
	collection = value;
	
	}		

Privilege Process

Connection conn =
general.DBConnect.getInstance().setConnection();
Statement s = conn.createStatement ();
s.execute ("UPDATE user_prev SET username='" + username + "', role_name='"+ role +"', involvement='" + project +"' WHERE prev_id='" + id + "'");
s.close();
conn.close();
	if(request.getParameter("submit").equalsIgnoreCase("Remove Privilege")) {
	return UserListDB.updateUserPrivileges(user, title, privilege, UserListDB.USER_PRIV_DELETE);
	} else {
	return UserListDB.updateUserPrivileges(user, title, privilege, UserListDB.USER_PRIV_EDIT);
	}
		Inappropriate Intimacy	Message Chain

View Collections

Connection conn =
general.DBConnect.getInstance().setConnection();
Statement s = conn.createStatement ();
s.executeQuery ("SELECT * from collections");
ResultSet rs = s.getResultSet ();
 	
while(rs.next()) {
values.add(rs.getString("title"));
values.add(rs.getString("description"));
values.add(rs.getString("downloadable"));
master.add(values);
values = new ArrayList();
}
 	
conn.close();
	if((request.getParameter("index").trim().equalsIgnoreCase("null")) ||
	request.getParameter("index").trim().isEmpty())
	result = CollectionDB.viewDB();
	else
	result =
	CollectionDB.viewDB(request.getParameter("index").trim().charAt(0));
	
	…
	…
	…
	
	protected static ArrayList<Collection> viewDB(char index) throws SQLException {
	Statement s = conn.createStatement ();
	s.executeQuery ("SELECT * from collections WHERE title LIKE '" + index + "%'");
	return (formatRS(s.getResultSet ()));		

	}
	
	protected static ArrayList<Collection> viewDB(String title) throws SQLException {
	Statement s = conn.createStatement ();
	s.executeQuery ("SELECT * from collections WHERE title='" + title + "'");
	return (formatRS(s.getResultSet ()));
	}		

String title = values.get(0);
…
…
…
 	
String downloadable = values.get(1);
String description = values.get(2);
String user = values.get(3);
	public class Collection {
	private String title;
	private String id;
	private String description;
	private String downloadable;
	private String user;
	//Parameter for edit action
	private String oldTitle;
	…
	…
	…
	}
		Replace Array With Object	Primitive Obsession

View Collection Users

Connection conn = general.DBConnect.getInstance().setConnection();
Statement s = conn.createStatement ();
s.executeQuery ("SELECT DISTINCT username from user_prev where involvement= '"+collection+"'");
ResultSet rs = s.getResultSet ();
 	
while(rs.next()) {
values.add(rs.getString("username"));
}
s.close();
conn.close();
…
…
…
	UserListDB.getUserList(title)
	…
	…
	…
	protected static ArrayList<HashMap<String, String>> getUserList(String title) throws SQLException {
	ArrayList<HashMap<String, String>> users = getUserNames(title);
	return getCompleteNames(users);
	}
		Hide Delegate	Message Chain

View References

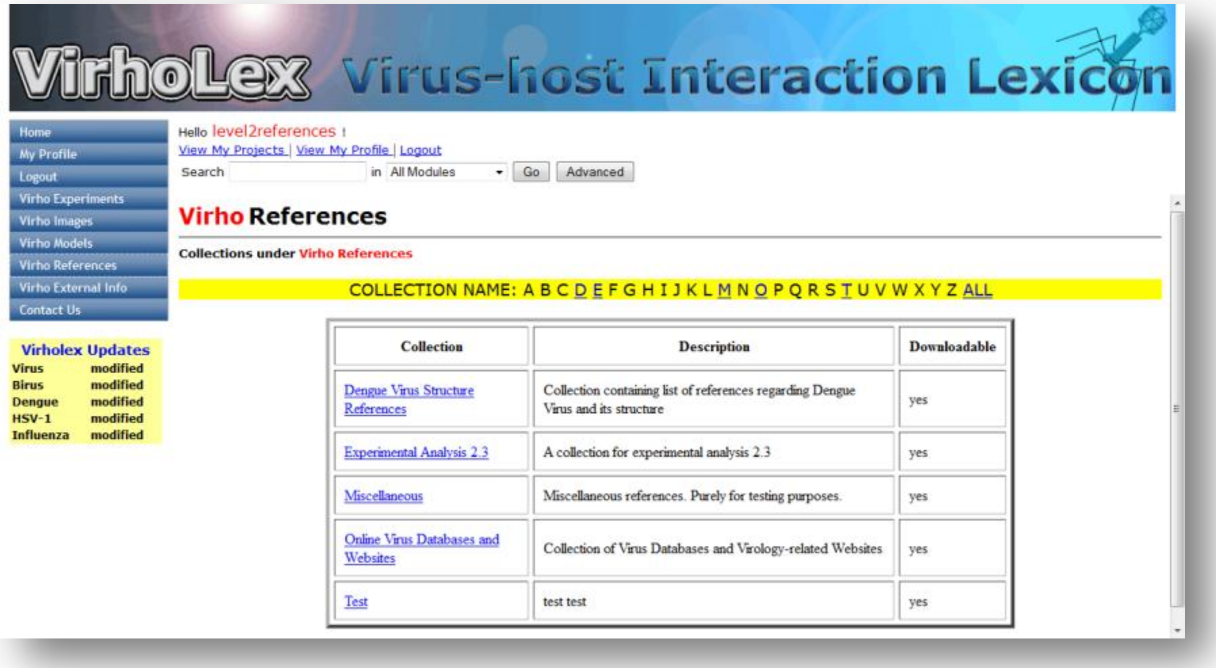
Connection conn = general.DBConnect.getInstance().setConnection();
Statement s = conn.createStatement ();
s.executeQuery ("SELECT * from coll_reference where title='"+ref+"'AND collection='"+coll+"' ");
ResultSet rs = s.getResultSet ();
	if((request.getParameter("ref") == null) || request.getParameter("ref").trim().isEmpty()) {
	result = ReferencesDB.viewDB(request.getParameter("coll").trim(), null);
	} else {
	result = ReferencesDB.viewDB(request.getParameter("coll").trim(), request.getParameter("ref").trim());
		Hide Delegate	Inappropriate Intimacy

values.add(rs.getString("author"));
values.add(rs.getString("year"));
values.add(rs.getString("type"));
values.add(rs.getString("volume"));
values.add(rs.getString("issue"));
values.add(rs.getString("pages"));
values.add(rs.getString("jb_title"));
values.add(rs.getString("short_t"));
values.add(rs.getString("keyword"));
values.add(rs.getString("link"));
values.add(rs.getString("path"));
values.add(rs.getInt("id"));
	ResultSet rs = s.getResultSet ();
	ResultSetMetaData rsmd = rs.getMetaData();
	
	while(rs.next()) {
	HashMap<String, String> current = new HashMap<String, String>();
	for(int i = 1; i <= rsmd.getColumnCount(); i++) {
	String columnName = rsmd.getColumnName(i);
	current.put(columnName, rs.getString(columnName));
	}
	
	references.add(current);
		Replace Array With Object	Primitive Obsession

Several new classes were also created to complement the refactorings done. These added classes helped in ensuring that the functionalities of the system remained the same even after the refactoring process. These new classes include the collection object, the database access for collections and references, and the add, edit and delete hotspots classes, to name a few.

Currently, the Virho References and Virho Hotspots module are completely functional and are faithful to their original specifications. The following screenshots show some of these functionalities.

For the Virho References module, the image below shows the listing of the collections in the module. It can be accessed by clicking the `Virho References` link in the left hand side of the page. Note that the image is for Registered Users to Collection Contributors.



Hello level2references !

View My Projects | View My Profile | Logout

Virholex Updates	
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

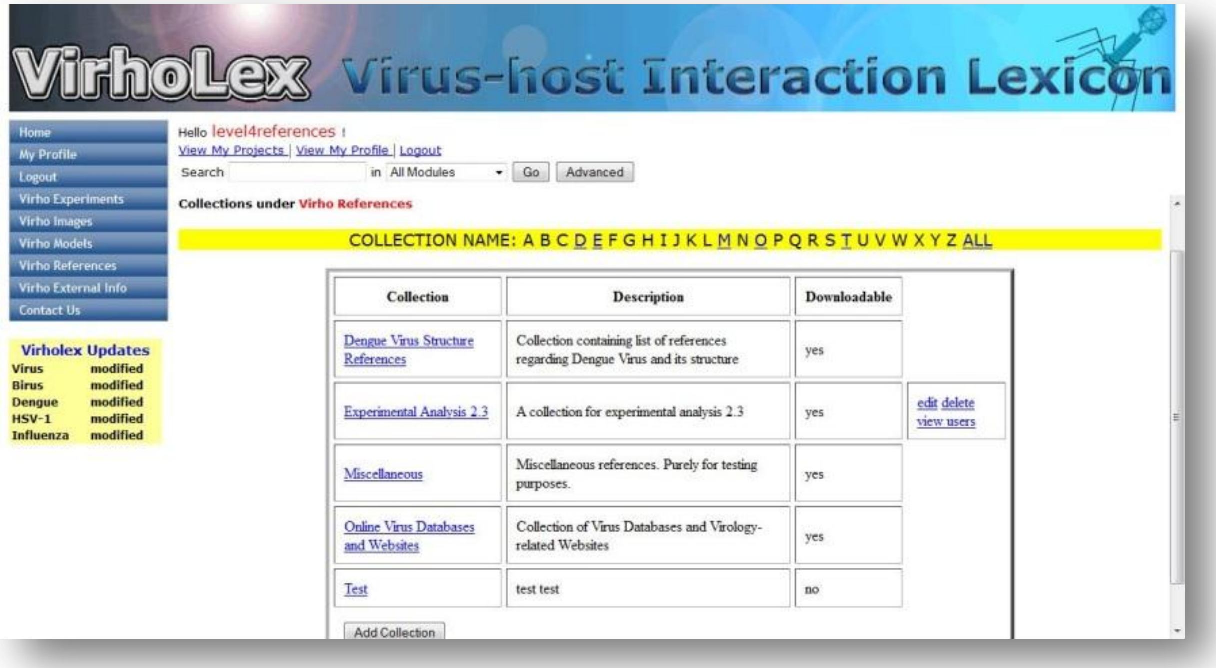
Virho References

Collections under Virho References

COLLECTION NAME: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ALL

Collection	Description	Downloadable
Dengue Virus Structure References	Collection containing list of references regarding Dengue Virus and its structure	yes
Experimental Analysis 2.3	A collection for experimental analysis 2.3	yes
Miscellaneous	Miscellaneous references. Purely for testing purposes.	yes
Online Virus Databases and Websites	Collection of Virus Databases and Virology-related Websites	yes
Test	test test	yes

This is the listing for a Collection Contributor.



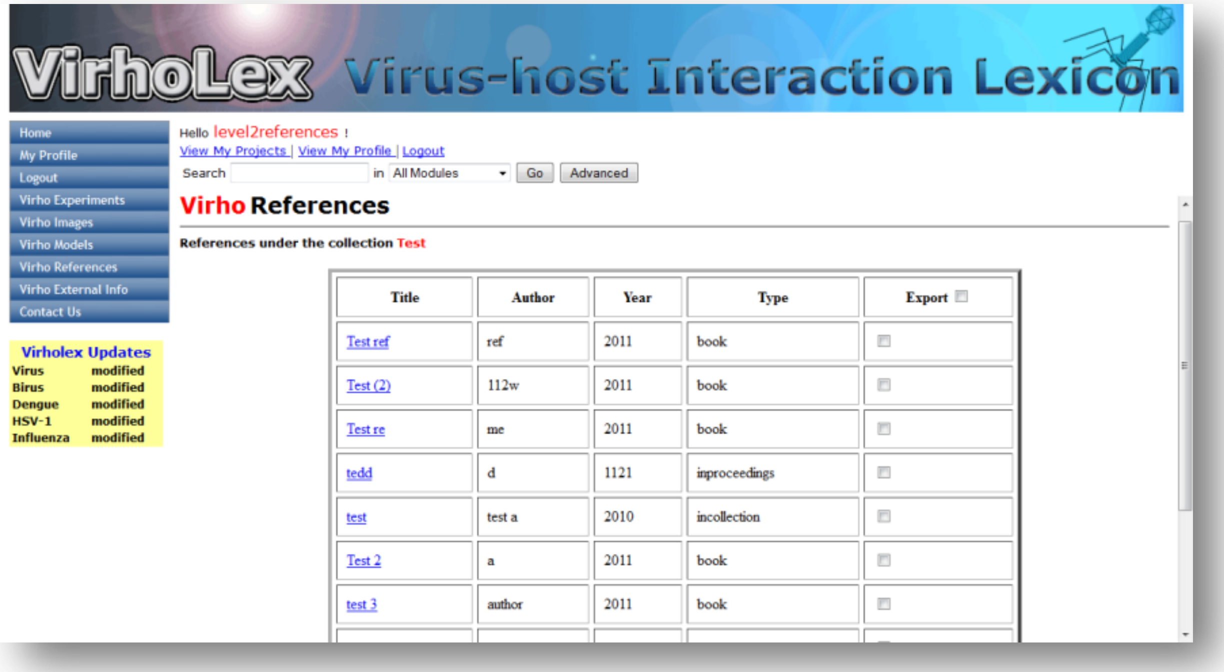
Hello level4references !
View My Projects | View My Profile | Logout

Collections under Virho References

COLLECTION NAME: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ALL

Collection	Description	Downloadable	
Dengue Virus Structure References	Collection containing list of references regarding Dengue Virus and its structure	yes	
Experimental Analysis 2.3	A collection for experimental analysis 2.3	yes	edit delete view users
Miscellaneous	Miscellaneous references. Purely for testing purposes.	yes	
Online Virus Databases and Websites	Collection of Virus Databases and Virology-related Websites	yes	
Test	test test	no	

Below is the view collection contents functionality for Register Users until Collection Contributors. It can be accessed by clicking the name of the collection.



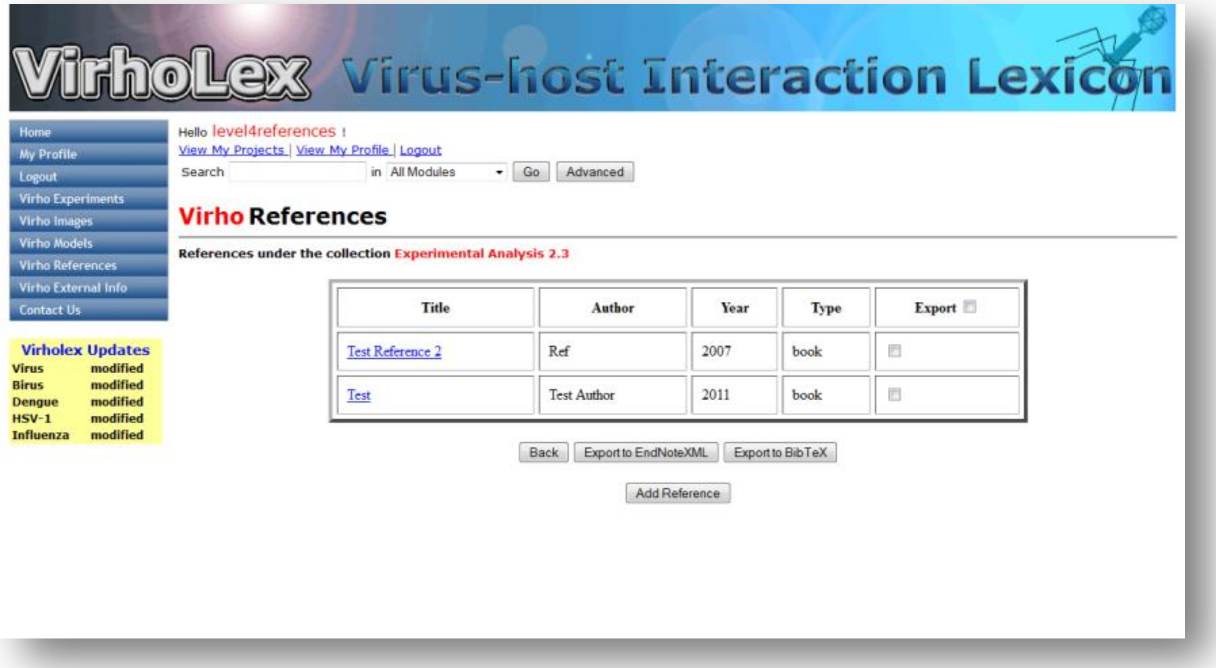
Hello level2references !
View My Projects | View My Profile | Logout

Virho References

References under the collection Test

Title	Author	Year	Type	Export □
Test ref	ref	2011	book	□
Test (2)	112w	2011	book	□
Test re	me	2011	book	□
tedd	d	1121	inproceedings	□
test	test a	2010	incollection	□
Test 2	a	2011	book	□
test 3	author	2011	book	□

The image below shows the same functionality but for a Collection Coordinator.



Hello level4references !

Virholex Updates

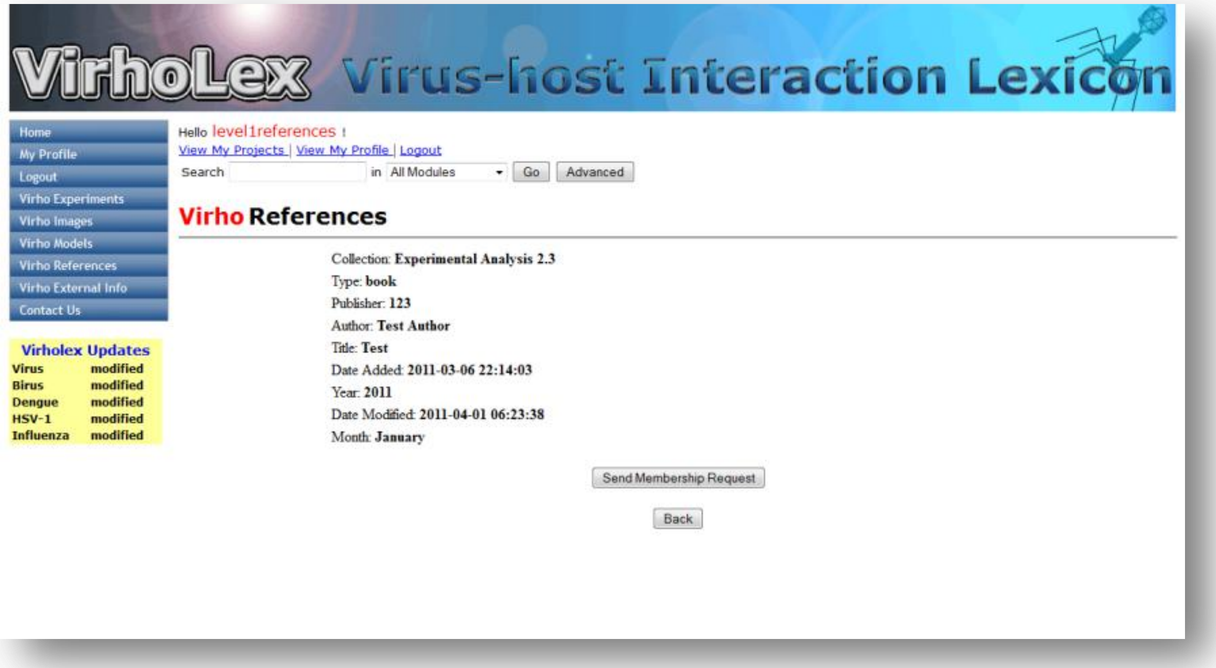
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Virho References

References under the collection Experimental Analysis 2.3

Title	Author	Year	Type	Export □
Test Reference 2	Ref	2007	book	□
Test	Test Author	2011	book	□

If a user wants to view the reference entry, the user will need to click on the name of the desired entry and the system will then display the appropriate details. Below is the view for a Registered User.



Hello level1references !

View My Projects | View My Profile | Logout

Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Virho References

Collection: **Experimental Analysis 2.3**

Type: **book**

Publisher: **123**

Author: **Test Author**

Title: **Test**

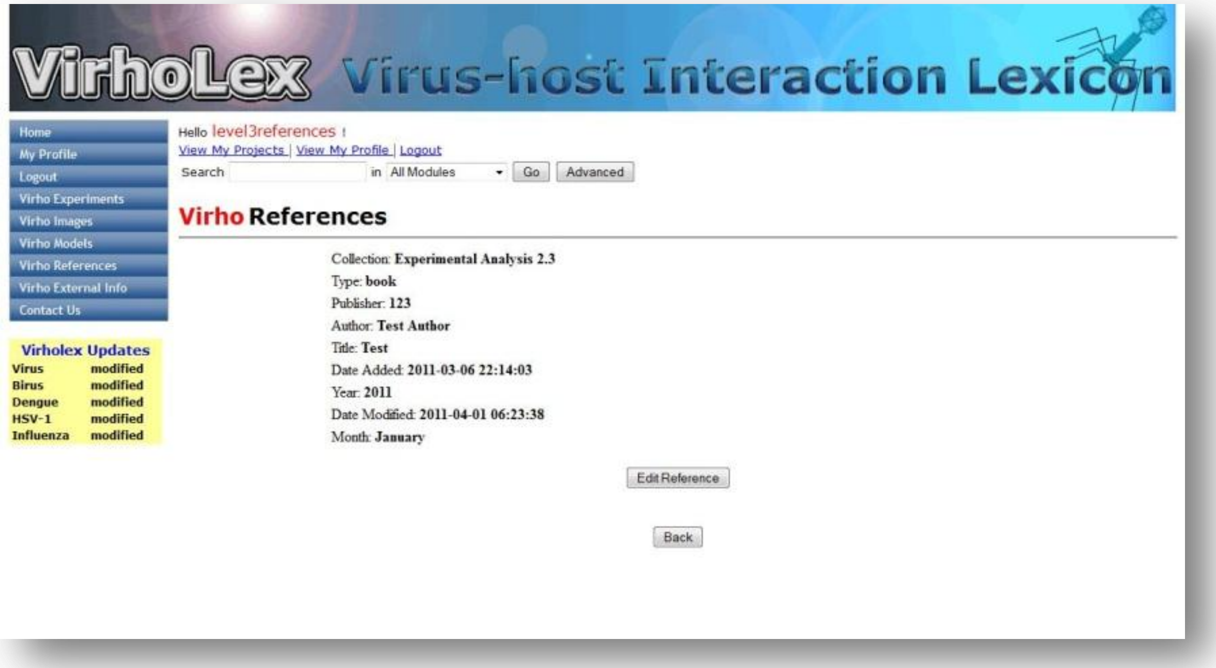
Date Added: **2011-03-06 22:14:03**

Year: **2011**

Date Modified: **2011-04-01 06:23:38**

Month: **January**

This view is for a Collection Contributor, while the next image is for the Collection Coordinator.



Hello level3references !

View My Projects | View My Profile | Logout

Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Virho References

Collection: **Experimental Analysis 2.3**

Type: **book**

Publisher: **123**

Author: **Test Author**

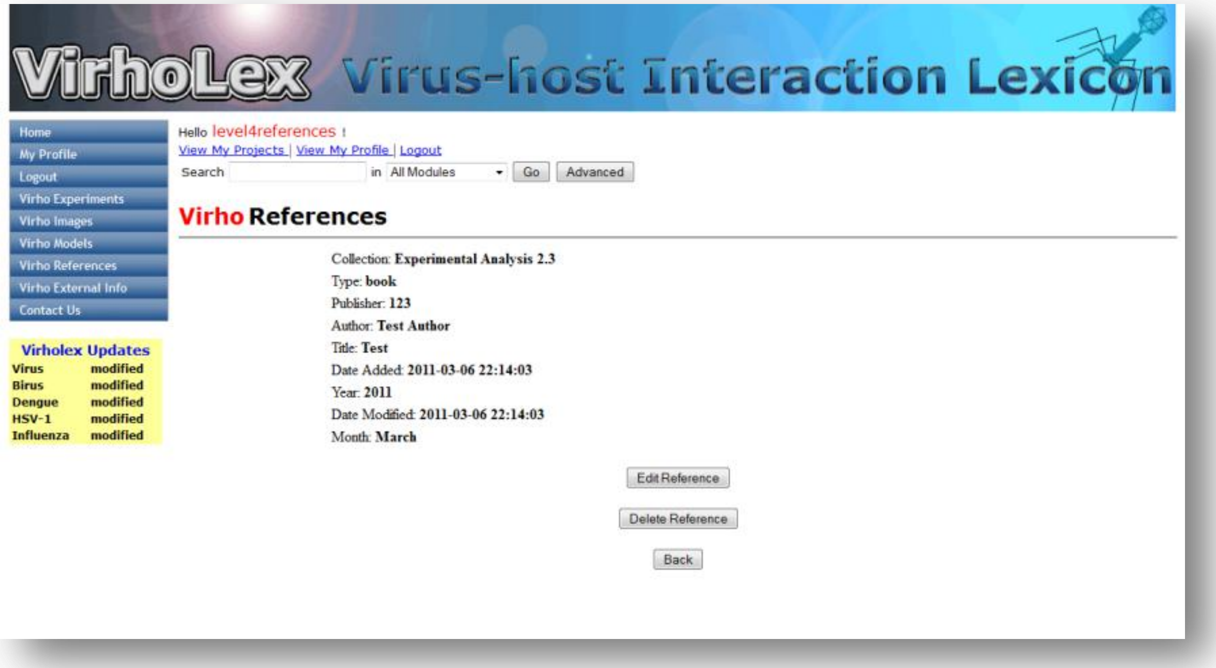
Title: **Test**

Date Added: **2011-03-06 22:14:03**

Year: **2011**

Date Modified: **2011-04-01 06:23:38**

Month: **January**



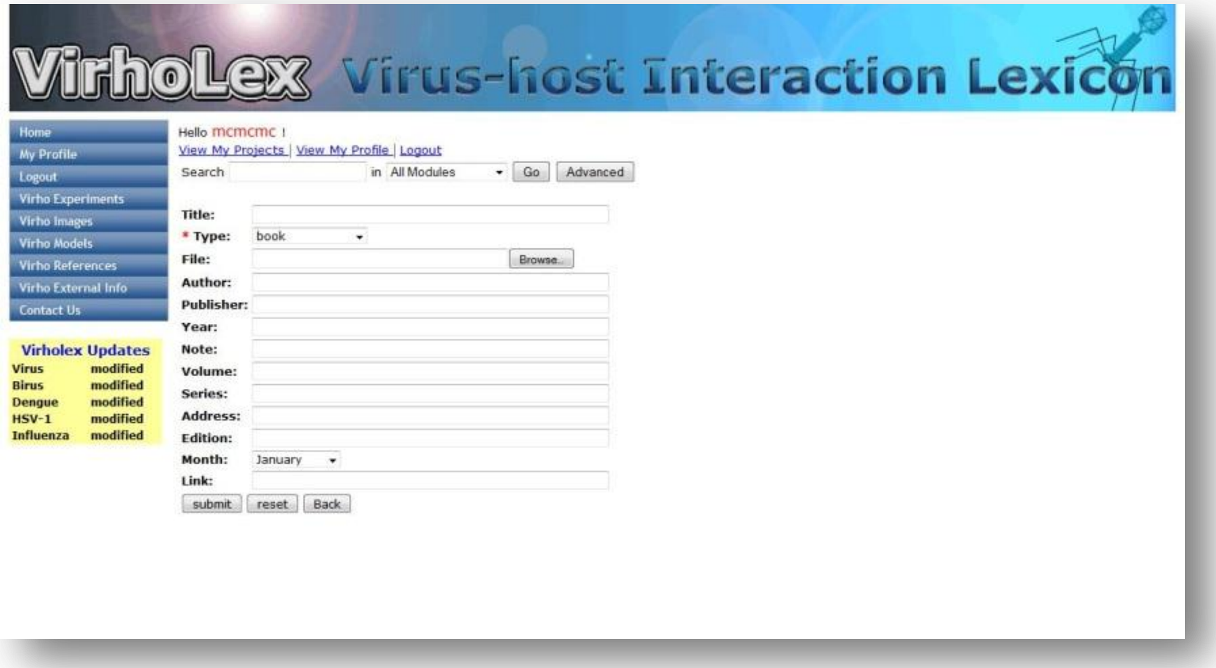
Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Virho References

Collection: **Experimental Analysis 2.3**
Type: **book**
Publisher: **123**
Author: **Test Author**
Title: **Test**
Date Added: **2011-03-06 22:14:03**
Year: **2011**
Date Modified: **2011-03-06 22:14:03**
Month: **March**

To add a reference to the collection, the user must click on the `Add Reference` button found at the bottom of the page. Note that this functionality is for Collection Coordinator and Collection Contributor account types only.



Hello mcmcmc !

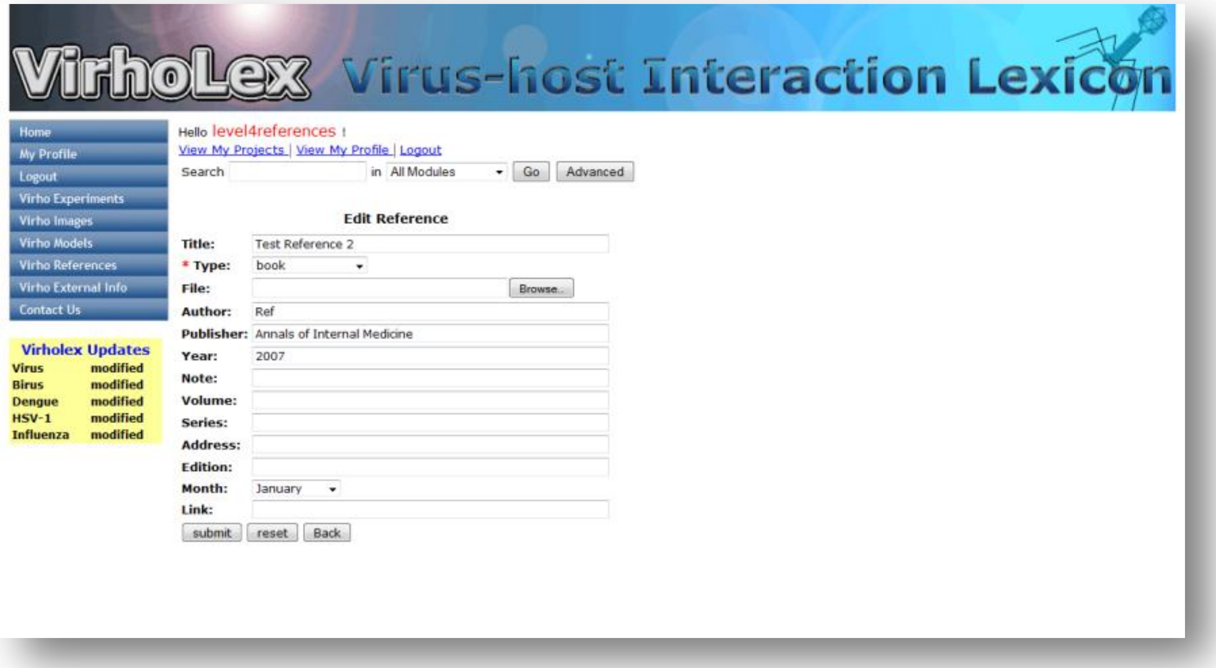
Search in All Modules

Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Title:
* Type: book
File:
Author:
Publisher:
Year:
Note:
Volume:
Series:
Address:
Edition:
Month: January
Link:

A Collection Contributor or Collection Coordinator can edit the reference entry by clicking the `Edit Reference` button.



Hello level4references !

Search in All Modules

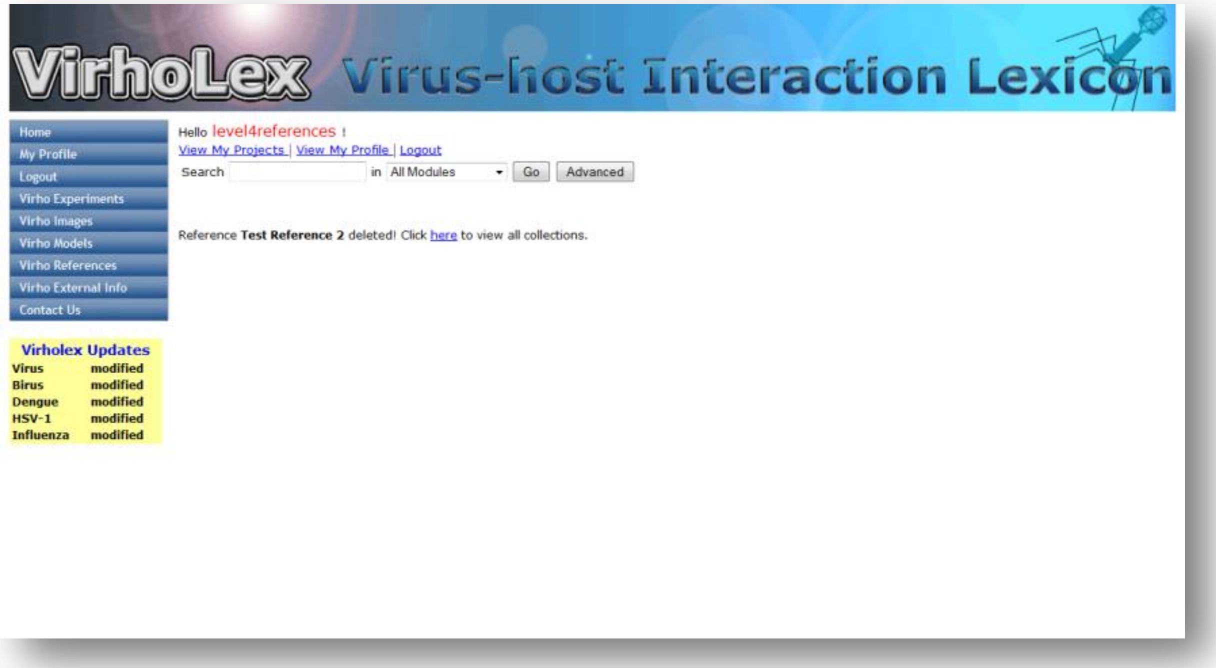
Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Edit Reference

Title: Test Reference 2
* Type: book
File:
Author: Ref
Publisher: Annals of Internal Medicine
Year: 2007
Note:
Volume:
Series:
Address:
Edition:
Month: January
Link:

To delete an entry, the user must click the `Delete` Reference button. Note that this is for Collection Coordinators only. Upon clicking, the following page will appear.



Hello level4references !

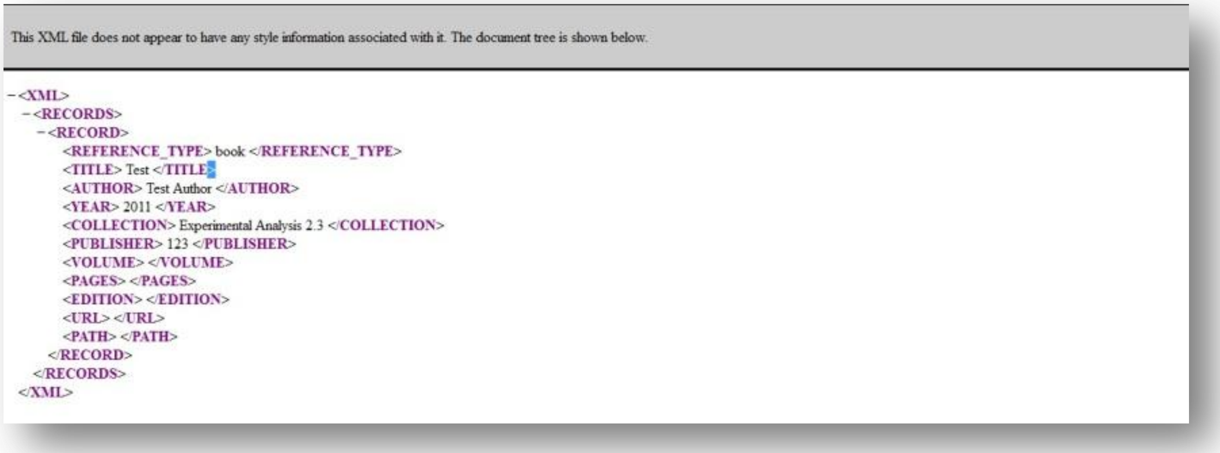
View My Projects | View My Profile | Logout

Reference **Test Reference 2** deleted! Click here to view all collections.

Virholex Updates

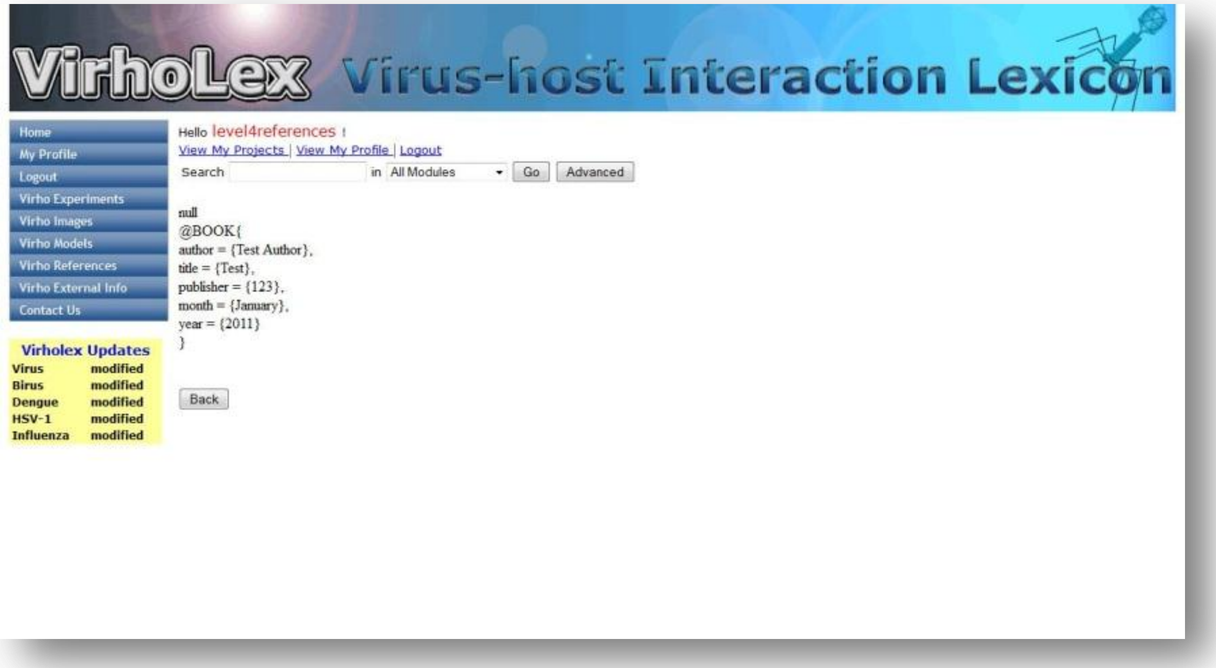
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

A user can also export the reference entries and collection entries to *BibTex* format or to *EndNoteXML* format. Below are example outputs of the said functionality.



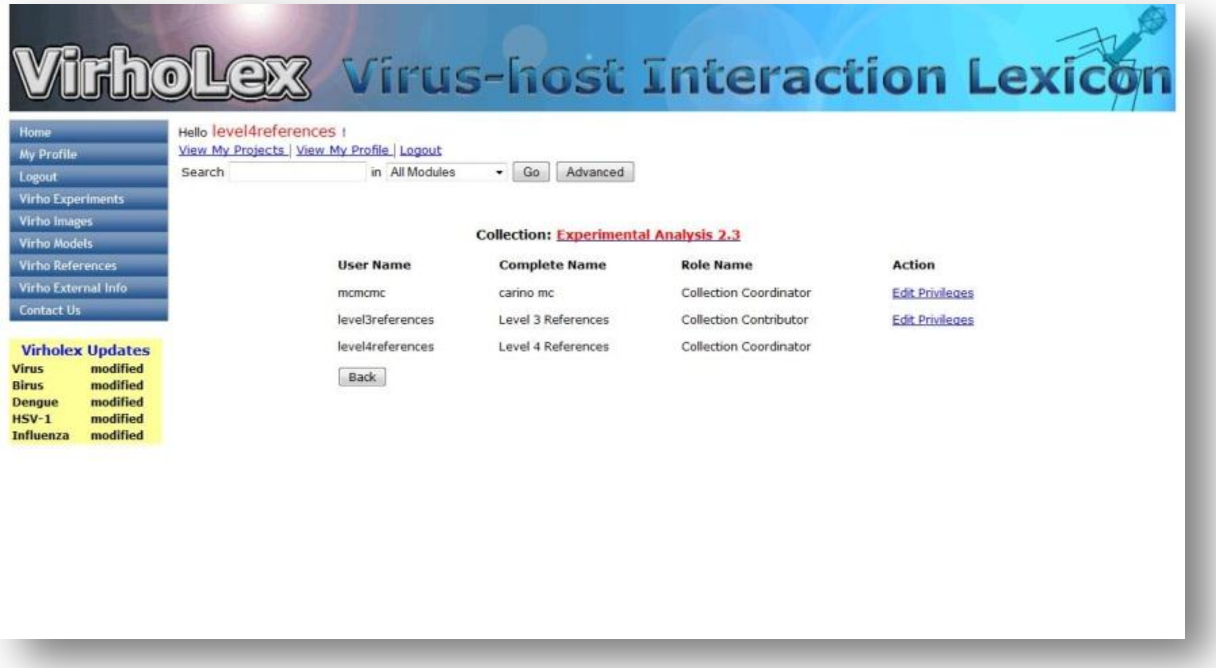
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
-<XML>
  -<RECORDS>
    -<RECORD>
       <REFERENCE_TYPE> book </REFERENCE_TYPE>
       <TITLE> Test </TITLE>
       <AUTHOR> Test Author </AUTHOR>
       <YEAR> 2011 </YEAR>
       <COLLECTION> Experimental Analysis 2.3 </COLLECTION>
       <PUBLISHER> 123 </PUBLISHER>
       <VOLUME> </VOLUME>
       <PAGES> </PAGES>
       <EDITION> </EDITION>
       <URL> </URL>
       <PATH> </PATH>
     </RECORD>
   </RECORDS>
 </XML>
```



```
null
@BOOK{
author = {Test Author},
title = {Test},
publisher = {123},
month = {January},
year = {2011}
}
```

A Collection Coordinator can also view the users of a particular collection and edit their privileges accordingly.



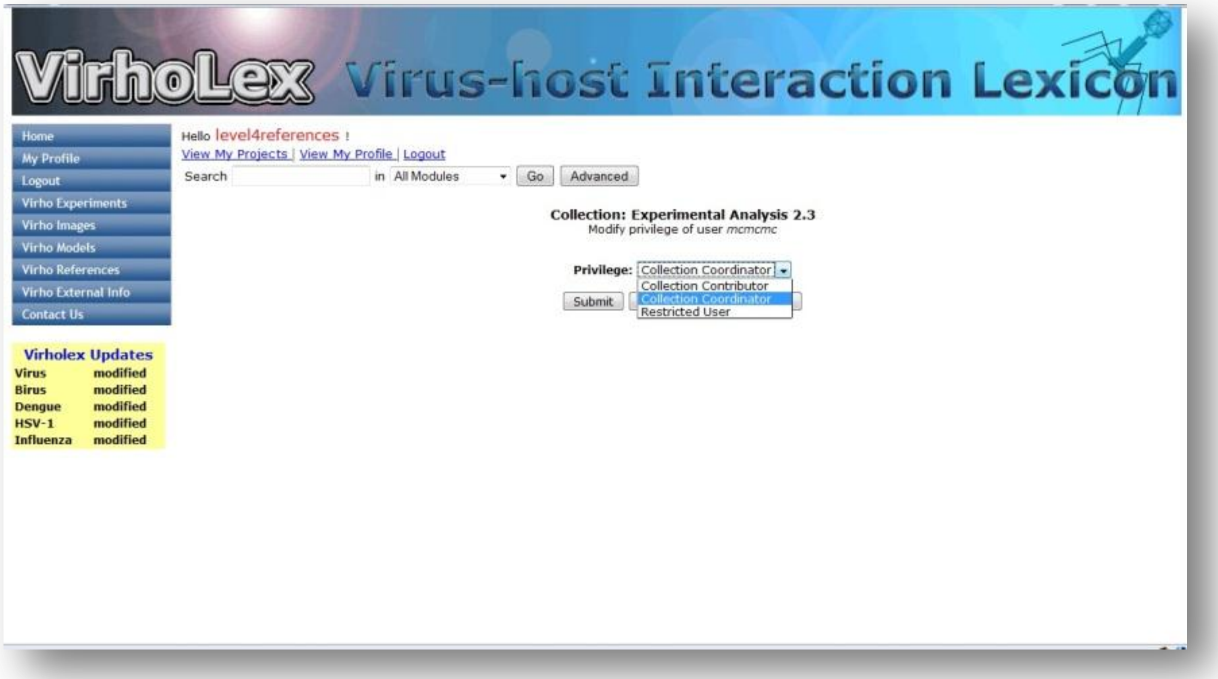
Hello level4references !

Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Collection: Experimental Analysis 2.3

User Name	Complete Name	Role Name	Action
mcmcmc	carino mc	Collection Coordinator	Edit Privileges
level3references	Level 3 References	Collection Contributor	Edit Privileges
level4references	Level 4 References	Collection Coordinator	



Hello level4references !

Virholex Updates

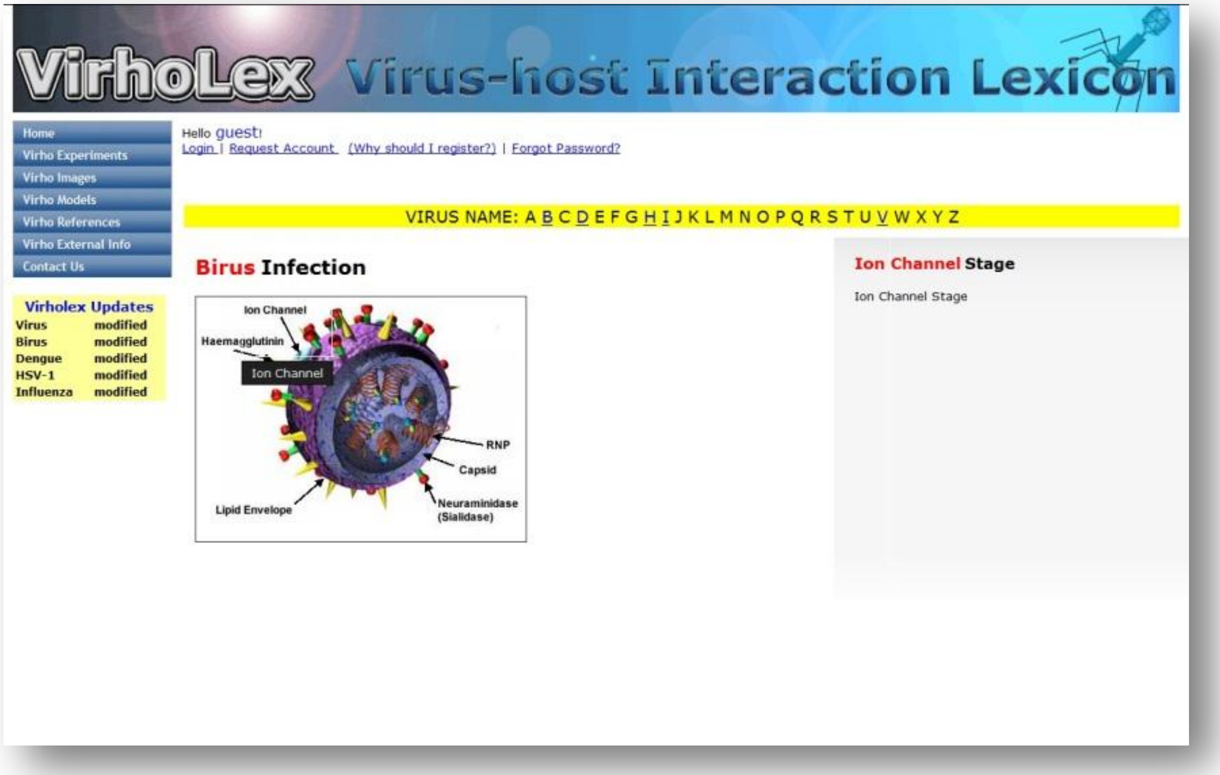
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Collection: Experimental Analysis 2.3

Modify privilege of user *mcmcmc*

Privilege: Collection Coordinator

For the Virho Hotspots module, the image below shows the virus diagram with the mouse pointer hovering atop one hotspot. Note that this is for a non-Hotspot Manager user.



Hello guest!
Login | Request Account (Why should I register?) | Forgot Password?

VIRUS NAME: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Virholex Updates

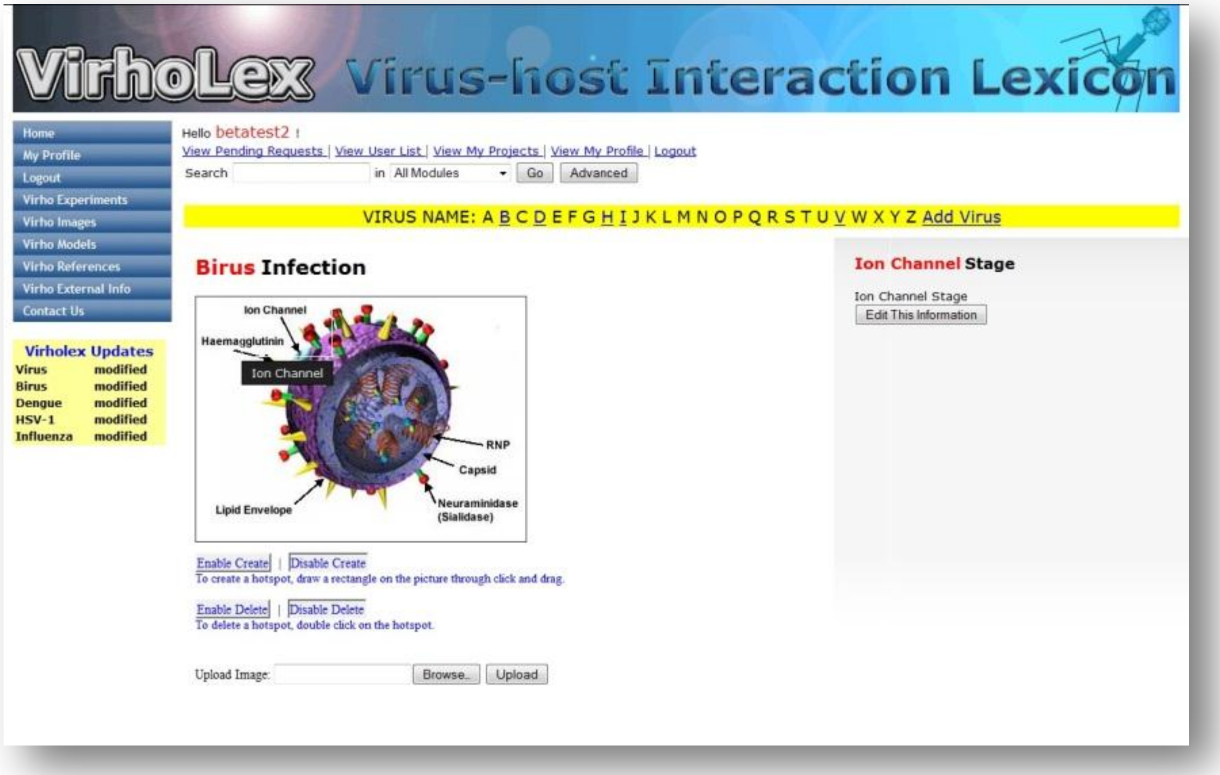
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Birus Infection

Ion Channel Stage

Ion Channel Stage

Meanwhile, this is the view for a Hotspot Manager account.


VirhoLex Virus-host Interaction Lexicon
Home
My Profile
Logout
Virho Experiments
Virho Images
Virho Models
Virho References
Virho External Info
Contact Us
View Pending Requests | View User List | View My Projects | View My Profile | Logout
Search
in All Modules
Go
Advanced
VIRUS NAME: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Add Virus
Ion Channel
Haemagglutinin
Ion Channel
RNP
Capsid
Lipid Envelope
Neuraminidase (Sialidase)
Enable Create | Disable Create
Enable Delete | Disable Delete
Upload Image:
Browse..
Upload
Edit This Information

Hello betatest2 !

Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Birus Infection

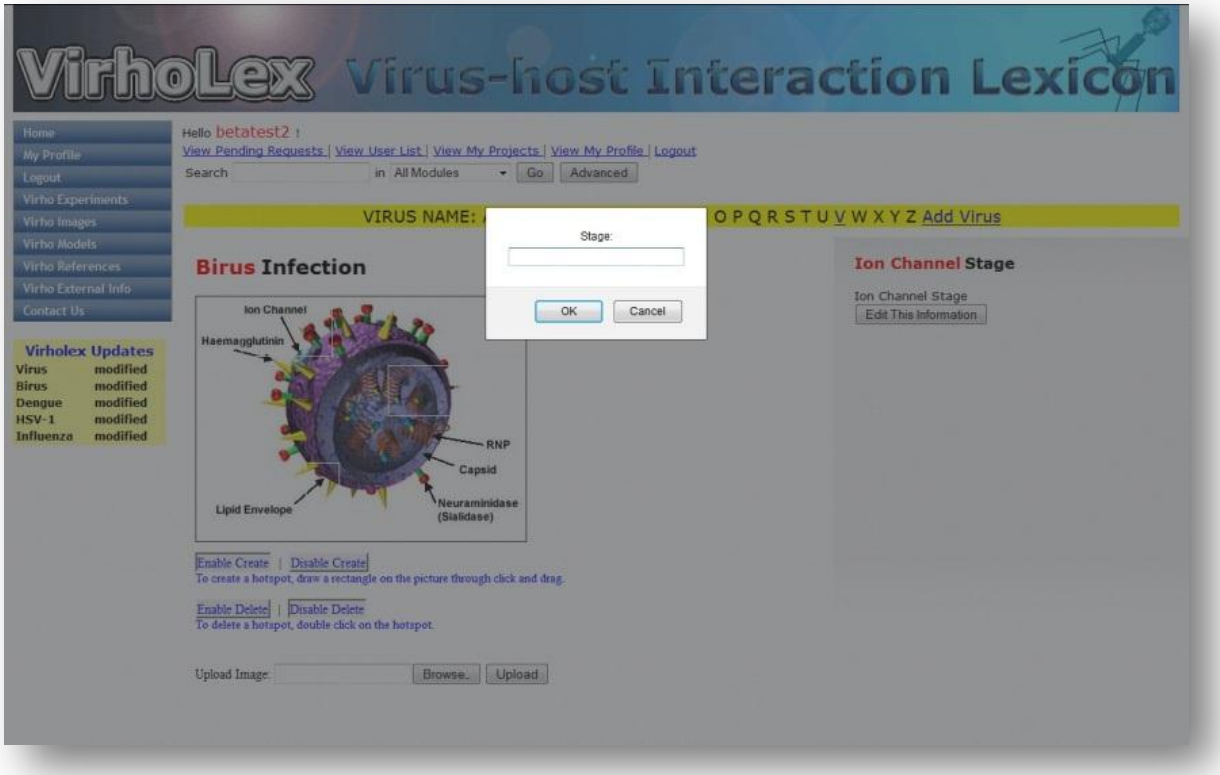
To create a hotspot, draw a rectangle on the picture through click and drag.

To delete a hotspot, double click on the hotspot.

Ion Channel Stage

Ion Channel Stage

A hotspot is added by clicking `Enable Create` button and drawing a rectangle in the desired hotspot through click and drag. When the mouse click is released, an alert box asking for the *Stage* name will be shown like the one below.



VirhoLex Virus-host Interaction Lexicon
Home
My Profile
Logout
Virho Experiments
Virho Images
Virho Models
Virho References
Virho External Info
Contact Us
View Pending Requests | View User List | View My Projects | View My Profile | Logout
Search
in All Modules
Go
Advanced
VIRUS NAME:
O P Q R S T U V W X Y Z Add Virus
Ion Channel
Haemagglutinin
RNP
Capsid
Neuraminidase (Sialidase)
Lipid Envelope
OK
Cancel
Edit This Information
Enable Create | Disable Create
Enable Delete | Disable Delete
Browse..
Upload

Hello betatest2 !

Stage:

Birus Infection

Ion Channel Stage

Ion Channel Stage

Virholex Updates

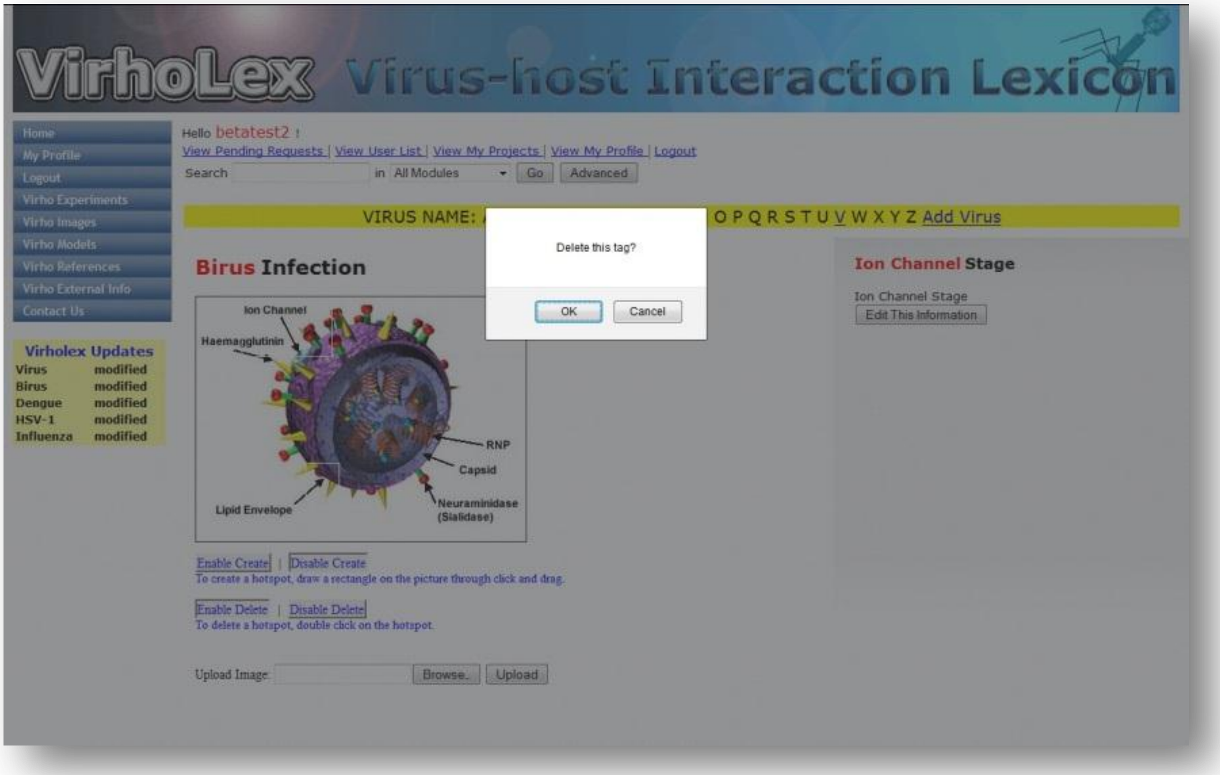
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

To create a hotspot, draw a rectangle on the picture through click and drag.

To delete a hotspot, double click on the hotspot.

Upload Image:

To delete a hotspot, the user must enable the delete functionality by clicking the `Enable Delete` button and double clicking the hotspot to be deleted. An alert would then appear asking for confirmation, like the one below.



Delete this tag?

Virholex Updates

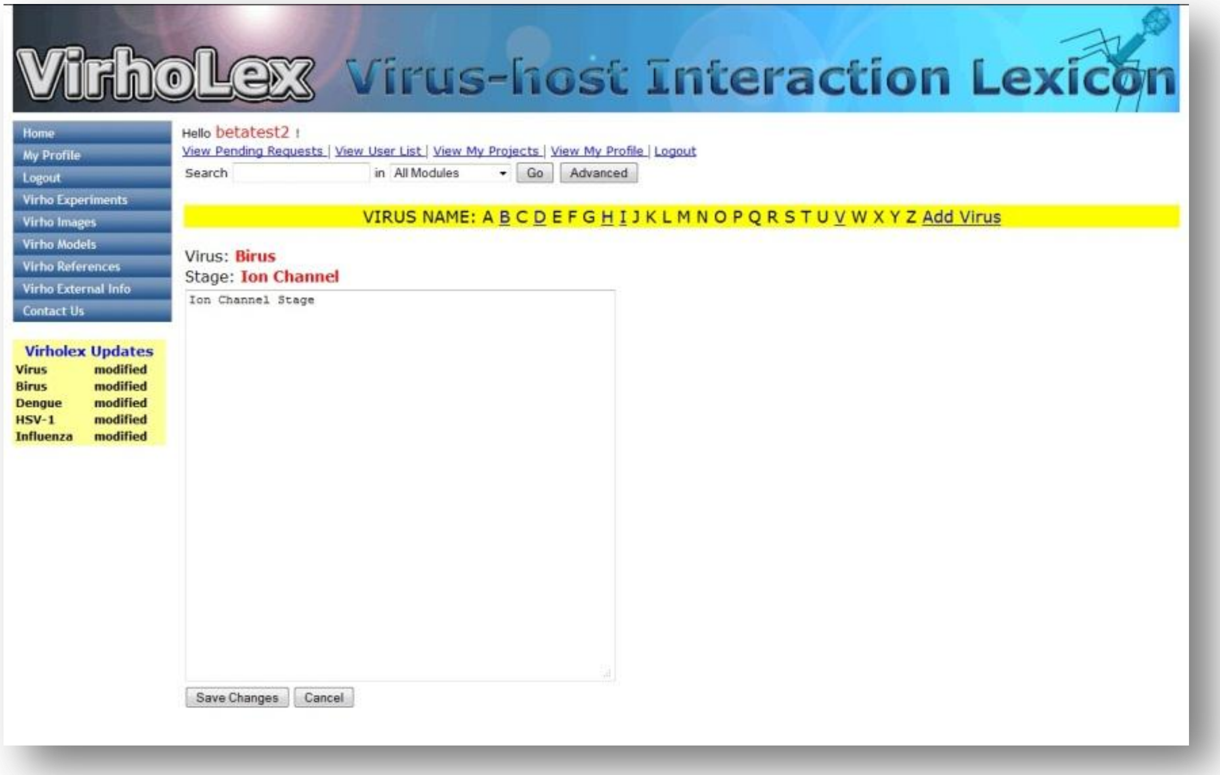
Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Ion Channel Stage

To create a hotspot, draw a rectangle on the picture through click and drag.

To delete a hotspot, double click on the hotspot.

A Hotspot Manager can also edit certain hotspot information by placing the mouse pointer atop the desired hotspot and clicking the `Edit This Information` button on the right hand side of the screen. The user would then be taken to a page with a form to be filled out.


VirhoLex Virus-host Interaction Lexicon
Home
My Profile
Logout
Virho Experiments
Virho Images
Virho Models
Virho References
Virho External Info
Contact Us
Hello betatest2 !
View Pending Requests | View User List | View My Projects | View My Profile | Logout
Search
in All Modules
Go
Advanced
VIRUS NAME: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Add Virus
Save Changes
Cancel

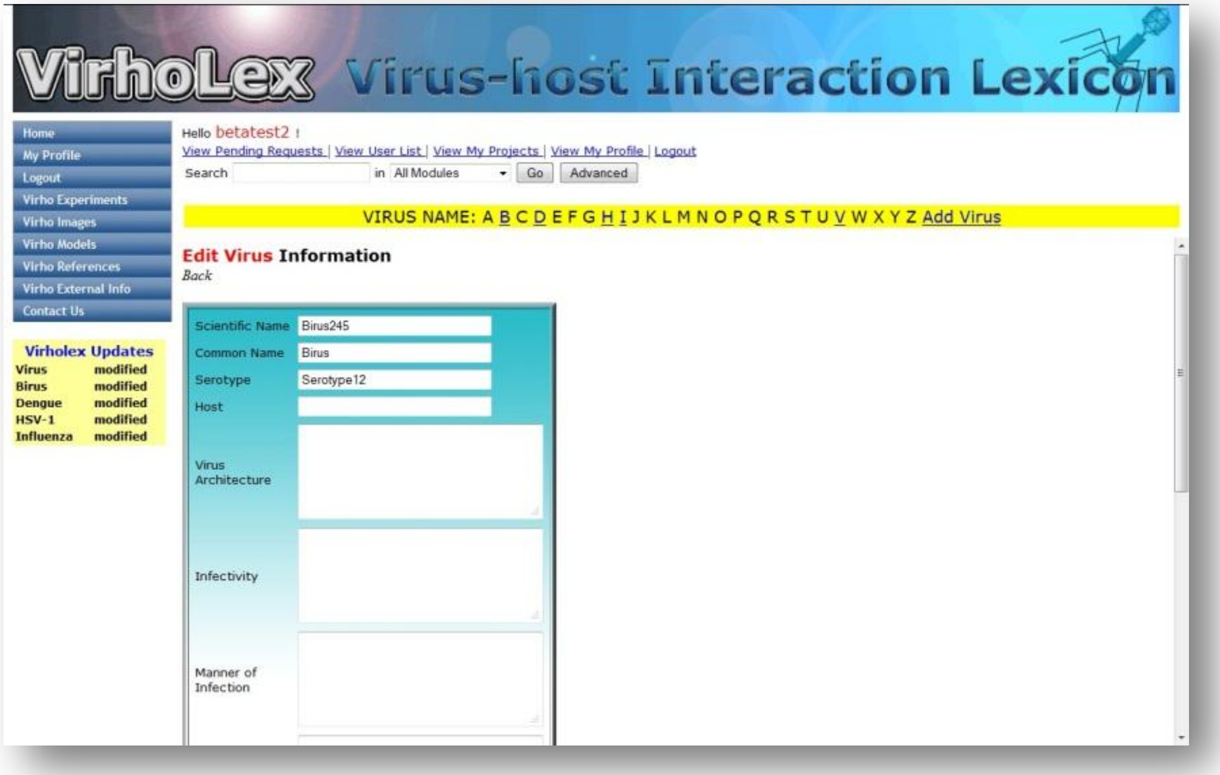
Virholex Updates

Virus	modified
Birus	modified
Dengue	modified
HSV-1	modified
Influenza	modified

Virus: Birus
Stage: Ion Channel

```
Ion Channel Stage
```

A Hotspot Manager can also edit the virus basic information. This can be done by clicking the `Edit` link in the right hand side of the screen as soon as the page and the virus diagram are loaded. The user would then be taken to a page with a form to be filled out.


VirhoLex Virus-host Interaction Lexicon
Home
My Profile
Logout
Virho Experiments
Virho Images
Virho Models
Virho References
Virho External Info
Contact Us
Virholex Updates
Virus modified
Birus modified
Dengue modified
HSV-1 modified
Influenza modified
Hello betatest2 !
View Pending Requests | View User List | View My Projects | View My Profile | Logout
Search in All Modules Go Advanced
VIRUS NAME: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Add Virus

Edit Virus Information

Back

Scientific Name	Birus245
Common Name	Birus
Serotype	Serotype12
Host	
Virus Architecture	
Infectivity	
Manner of Infection	

VI. DISCUSSION

VirHoLex References module is responsible for collecting and classifying the reference entries based on their type as well as the project it belongs to. It gives users an avenue to view pertinent information about a particular entry and possibly download a file related to the reference provided by the uploader. It also allows users to view hyperlinks leading to other websites that can be useful in their current research. More so, the module allows users to easily export bibliographic entries in EndNoteXML or BibTeX for easier storage of reference details.

VirHoLex Hotspots module allow users to create hotspots in a given virus diagram and attach pertinent information about that particular part of the image. Through this tagging system, user can easily share information regarding a particular stage of a virus which might help them and other researchers in their research.

However, in order to be fully useful, both modules had to be rid of all errors first. More so, with the refactorings done, the modules can be easily modified and extended in the future by a new development team. While no immediate benefits will be felt by the end user, the ability to add new functionalities without development taking too long would surely be welcomed by these users.

VII. CONCLUSION

Refactoring is a process whose results are not really geared towards the user. Instead, refactoring is actually a process undertaken by developers for other developers. Its aim is to make code as readable and as easy to understand as possible. While it is a seemingly insurmountable task if the system in question is large, the process has matured enough throughout the years that it is able to cope up with these challenges. Listings of the most common refactoring techniques as well as code smells are available not only through books but also through the Internet.

The Virus Host Interaction Lexicon system is an example of this. VirHoLex is a complex system composed of interlocking parts and business logic, whose code base became so large that bugs and duplication of code became inevitable. The refactoring of the Virho References module and Virho Hotspots module addressed these problems, while giving it the opportunity to be expanded and modified in the future versions because of its more modular and more manageable code.

While it is only limited to two modules, it is a great step towards making the system a useful tool in helping virologists around the world with their researches.

VIII. RECOMMENDATION

To further ensure the successful use of the system by researchers, it is recommended that the other modules be refactored as well. While there are no bugs currently known in these other modules, refactoring the code would make it easier to extend and add functionalities in the future.

More so, it is also recommended that a framework be used for the next iteration of the system. A framework would greatly reduce the number of redundant and duplicate code in the system, and help make sure that the system is in tip-top shape. It would also help cut the development time shorter because it will help developers focus on more pressing issues in the development rather spend valuable resources solving problems that can be taken cared by the frameworks instead.

IX. REFERENCES

[1] M. Fowler and K. Beck, *Refactoring: improving the design of existing code*. Addison-Wesley, 1999.

[2] B. Elepano, “ViRUS: Virho Registered Users Services and Image Hotspots,” University of the Philippines, Manila, 2008.

[3] Virholex Team, “Virus-Host Interaction Lexicon Functional Specifications for Release 1.0,” 19-Feb-2009.

[4] J. Brant and D. Roberts, “Refactoring techniques and tools,” Mar-1999.

[5] F. Simon, F. Steinbr\ückner, and C. Lewerentz, “Metrics based refactoring,” in *csmr*, p. 30, 2001.

[6] Y. Kataoka, D. Notkin, M. D. Ernst, and W. G. Griswold, “Automated support for program refactoring using invariants,” in *icsm*, p. 736, 2001.

[7] G. Sunyé, D. Pollet, Y. Le Traon, and J. M. Jézéquel, “Refactoring UML models,” *«UML» 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, pp. 134–148, 2001.

[8] K. Beck, “Embracing Change with Extreme Programming,” *Computer*, vol. 32, no. 10, pp. 70-77, 1999.

[9] A. Van Deursen, L. Moonen, A. van den Bergh, and G. Kok, “Refactoring test code,” in *Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2001)*, pp. 92–95, 2001.

[10] S. Hanenberg, C. Oberschulte, and R. Unland, “Refactoring of aspect-oriented software,” in *4th Annual International Conference on Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World (Net. ObjectDays)*, 2004.

[11] J. Fenn, “Managing citations and your bibliography with bibtex,” *The PracTEX Journal,(4)*, 2006.

[12] O. Patashnik, *Designing BIBTEX styles*. February, 1988.

[13] L. Previtali, B. Lurati, and E. Wilde, “BibTEXML: An XML representation of BibTEX,” in *Poster Proceedings of the Tenth International World Wide Web Conference*, pp. 1090–1091, 2001.

[14] J. Hufflen, “mlbibtex: a New Implementation of bibtex.”

[15] B. Lund, T. Hammond, M. Flack, and T. Hannay, “Social Bookmarking Tools (II),” *D-Lib Magazine*, vol. 11, no. 04, 2005.

[16] A. Agrawal, *Endnote 1 - 2 - 3 Easy!: Reference Management for the Professional*. Springer, 2009.

[17] T. Mens and T. Tourwé, “A survey of software refactoring,” *IEEE Transactions on software engineering*, vol. 30, no. 2, pp. 126–139, 2004.

[18] M. Fowler, “Refactoring Home.” [Online]. Available: http://www.refactoring.com/. [Accessed: 19-Oct-2010].

[19] K. Sen, D. Marinov, and G. Agha, “CUTE: A concolic unit testing engine for C,” in *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pp. 263–272, 2005.

[20] R. Sedgewick, *Algorithms*. 1984.

[21] “Scalable Vector Graphics (SVG) 1.1 (Second Edition).” [Online]. Available: http://www.w3.org/TR/SVG11/. [Accessed: 10-Oct-2010].