MyOpenJournal Java Rebuild

Jeremy Crafts, John Larimer, Eric Dundore, Dan Sassi
Chris Crawford, Tyler Purdom, Mike Green

Spring 2013
Pennsylvania State University

Table of Contents

1. Abstract

This project mostly involves the redesign and reimplementation of the MyOpenJournal website. MyOpenJournal is meant to be a searchable archive of academic publications, forum for scientific collaboration, and a tool for peer review and research ranking. Some of the objectives of MyOpenJournal are social networking, open access publication, open review capability, ranking system, data storage subsystem and a moderation subsystem. Some of the inefficiencies of publishing scientific journals can be overcome by this system. Printed journals contain valuable research, but have limited access due to fees. MyOpenJournal will be a free and open platform for scientific publishing so that anyone can read or submit papers. Printed papers also go through a very slow review process before they are published, often taking 6-24 months. With MyOpenJournal platform, publishing time is instant, and reviewers can make comments immediately after submission.

Most of the requirements for this system have already been defined in the previous version which is written in PHP in Fall 2012. This project aims to re-implement the system in Java to boost scalability and extendibility in the future. The past implementation used a LAMP (Linux, Apache, MySQL, and PHP) stack. Using a new Java back-end and Java Web-Services will enable future students who have taken CMPSC 221 to make further changes and additions to the project. Since every computer science student at Penn State is required to take CMPSC 221, this will increase the developer base and thus boost productivity. Due to the modularity and object oriented nature of Java, extendibility will be much easier as opposed to the script-like nature of the current implementation in PHP. There will be performance benefits in the form of decreased execution time at a trade off for more lines of code and memory usage. The benefits of the Java implementation outweigh the increase in lines of code and memory usage because of the boosted performance, scalability and expandability.

2. Requirements

2.1 Background

With the previous effort of writing MyOpenJournal in PHP, the system was becoming harder to maintain for new students that were interested in working on the project. This created a need to not only implement changes to the MyOpenJournal system, but also to rewrite the architecture of the system in a language so that future students can maintain and modify the system without learning a new language.

2.2 Environment

The Java rework brought the need to find a new development environment. The idea was brought up to use a CMS (Content Management System) such as Joomla! to design the revamped MyOpenJournal. However, upon further investigation it was discovered that a CMS would be far too restrictive on the features that are to be implemented. This constriction would allow for little to no modification, and essentially only offer a tool to produce a cookie cutter site. The need for a more robust and modifiable environment led to the choice of using the ZK Studio Java framework. ZK allows for user friendly customization of our Java architecture while still allowing for full customization.

The server that MyOpenJournal is deployed on is running Microsoft IIS. ZK Studio produces WAR (Web Application Archive) files which are JAR files used to distribute a collection of JavaServer pages, Java Servlets, and Java classes. Since MyOpenJournal is being rewritten in Java using ZK studio, there needs to be a platform that can deploy these WAR files. Microsoft IIS does not support WAR files, JAR or Java servlets. Apache Tomcat supports WAR and JAR files and thus was chosen to deploy the WAR file that contains the MyOpenJournal Java implementation. See Section 3.4 for more information on how to configure a server running IIS to redirect traffic requesting JavaServer pages to Apache Tomcat.

2.3 Features

Homepage - MyOpenJournal features an informative homepage that lists papers based on three criterion. The first column lists the most popular papers. This popularity is calculated using our ranking system mentioned in section 2.4. The second column lists the newest papers. The papers in this column are organized based on their upload date and time. The third, and last, column lists highest rated papers first. This rating is calculated solely by taking the difference between total upvotes and total downvotes.

Register - MyOpenJournal allows users to register their own account to upload papers and reviews, comment on reviews, and vote on papers, reviews, and comments on reviews. Upon registration, the user will receive a confirmation email. This feature of the website allows a level of filtering on bot users with the intention of malicious use of the website.

Login - MyOpenJournal allows users that have already registered to login to their account and restore preferences that they had previously configured. By logging in, the user is granted access to the features mentioned above in the Register feature.

Edit profile - Once a user has logged in, they are able to edit their profile information. This includes their name, affiliation, a short biography, and a picture. This allows users to feel more integrated with the site and community of users. The edit profile feature also allows users to change their password.

Upload paper - The core of MyOpenJournal is academic and other documents. Once a user is logged into their valid account, they may upload a paper that they have written. On the upload page, they may upload a pdf file of their document and include information such as authors, co-authors, and references.

View paper - Once a paper is uploaded, users must be able to view, rate, and post reviews on the paper. When a user is viewing a paper that has been uploaded, they are able to upvote or downvote the paper, view reviews that have been posted for a paper, and post their own review of the paper.

Upload review - Once a paper is uploaded, a user may have an opinion that they would like to voice. MyOpenJournal allows users to post reviews that link back to a paper. In these reviews, a user may voice their opinions, comments, and criticisms of a paper. To help filter out "flaming" and "troll comments", a review must contain a minimum of 500 words to be submitted. In addition, the only way to make comments on a paper is through a review. There is no feature in MyOpenJournal to directly comment on papers.

View reviews - Once a review is uploaded, other users must be able to view, rate, and comment on the review. When a user is viewing a review of a paper that has been uploaded, they are able to upvote or downvote the review, view comments that have been made on a review, and post their own comments on the review.

Comment on reviews - When a user is viewing a review they may feel the need to voice their opinions on the review. MyOpenJournal allows users to only comment on one aspect of the site, reviews. Users may post comments on reviews, and have their comments rated by other users.

View review comments - MyOpenJournal allows all comments made on reviews to be viewed by other users that have logged in. The comments will be listed in a list based on their rating given by other users. The rating of each comment will also be visible alongside the content of the comment.

Upvote/downvote - MyOpenJournal allows a great deal of community criticism. Users that have logged in may submit a vote on papers, reviews, and comments on reviews. Each user has one vote that they can cast on each unique entity. A user may either "upvote" or "downvote" a paper, review, or comment on a review. An upvote will add one positive point to an entity's rating, while a downvote will add one negative point (essentially remove one positive point) to an entity's rating. With this feature the community may rate higher quality papers with a higher rating, and rate lower quality papers with subsequent lower ratings.

2.4 Ranking

The ranking system that was put in place in the last effort would not work as is with the updated Java architecture. The ranking system that was previously constructed gave a certain weight to every object of the site and calculated a ranking for every individual object. It was deemed that this system would be too taxing on the system and would be beyond the scope of the group to complete along with the other aspects of the project that needed to be completed. Another reason that the ranking system had to be changed is to give a fair chance to all users. If users have ratings and their papers are displayed based on this rating, newer users would never be able to have their papers viewed by others because they would be buried.

The ranking algorithm idea that is going to be implemented in the new version of MyOpenJournal uses up and down votes for users to rate papers and reviews. When a new paper is uploaded, it will appear at the top of the website. Over time, its rating will be determined by how long the paper has been up on the site combined with the net score (up/down votes) that it has received over that time period. This allows all users to have a fair chance at their paper being seen and rated.

Knowing that our rankings would be determined solely on the up-vote count, the down-vote count, and the time since the paper had been uploaded to MyOpenJournal, we set out to make an efficient, simple, and functional ranking system.

Our first few prototype ranking systems calculated the proportion of up-votes to down-votes and then divided this value by some function of the number of days since the paper had been uploaded. The problem with this approach is that the older papers would sometimes have higher priority than the newer papers. Also, as papers got older, their scores would keep decreasing, so even really good but old papers would essentially become invisible after enough time passed.

After reevaluating these prototypes, we made an algorithm that would still find the proportion of up-votes to down-votes, but it would add some constant divided by the number of days to this value. This method fixed the problem of older papers dropping off, since after many days had passed the additional term would essential become 0 (the time this would take would be affected by the size of the constant). We still were not satisfied with this method since we realized that the proportion of up-votes to down-votes could give us highly varying values.

Our current iteration of the ranking system uses the proportion of up-votes to total votes and then adds a constant divided by the number of days. We like this method so far since it can be scaled nicely to a value between 0 and 1. New papers will be shown at the top and (with the constant we have chosen) the regency of the paper won’t be factor past roughly 10 days of its initial upload. One factor our ranking doesn’t take into account is papers with many votes will

have as much rating as papers with few votes, but the same proportion of votes. This may be troublesome, since controversial, well-known papers deserve more attention than obscure ones, but this all depends on how users decide to vote on the papers.

2.5 User Interface

Alongside the redesign of the architecture, the user interface of MyOpenJournal also needed to be updated and refined. Below are the mock ups of the new user interface of MyOpenJournal.

The About Us page describes the nature of the website, and lists details describing the members of the team that developed it.

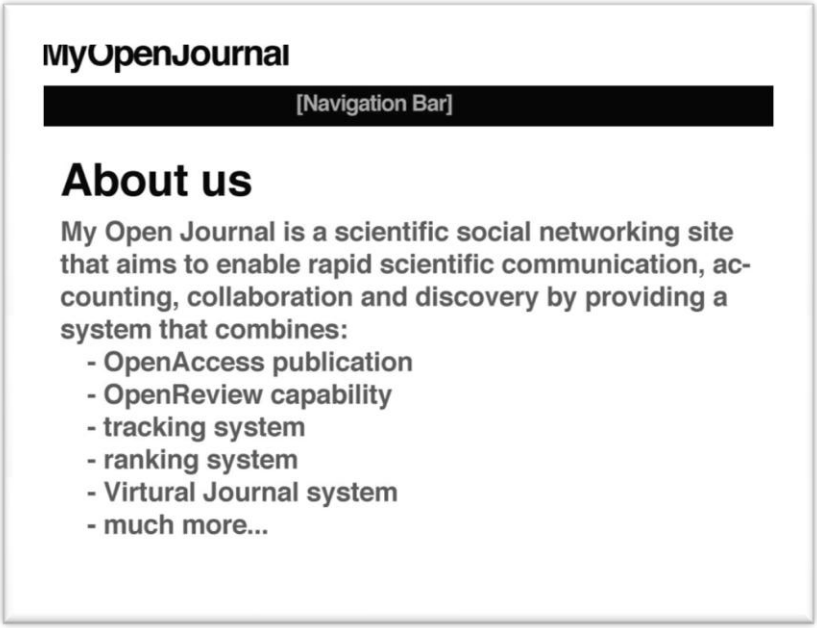

Figure 2.1: "About Us" page for the MyOpenJournal website.

The Submit Review page allows users to rate a paper. In addition, it allows the user to either upload a review that they have previously typed or to write a review directly on the page.

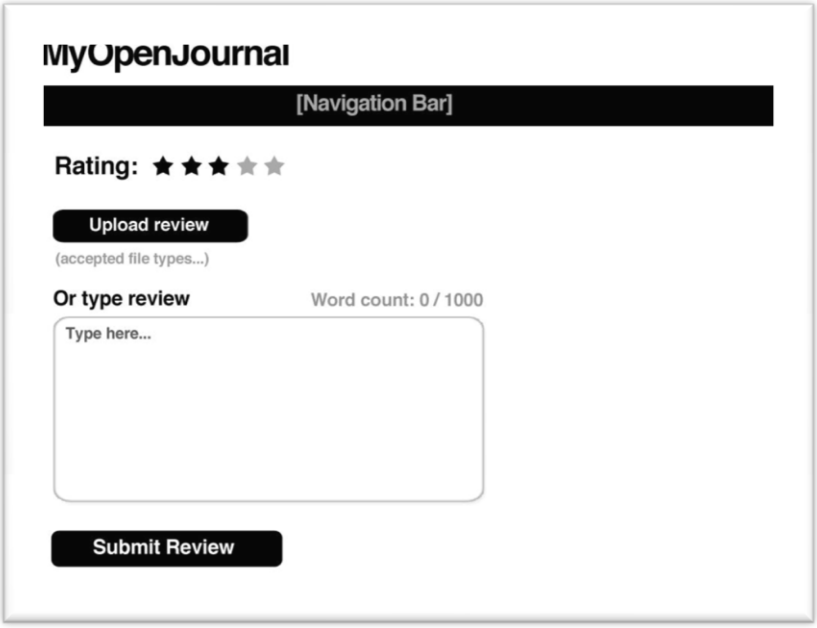

Figure 2.2: "Submit Review" page for the MyOpenJournal website.

The Advanced Search page allows users to search papers by terms, authors, credentials, and multiple categories. Options are included to include reviews, comments, and profiles in the search results.

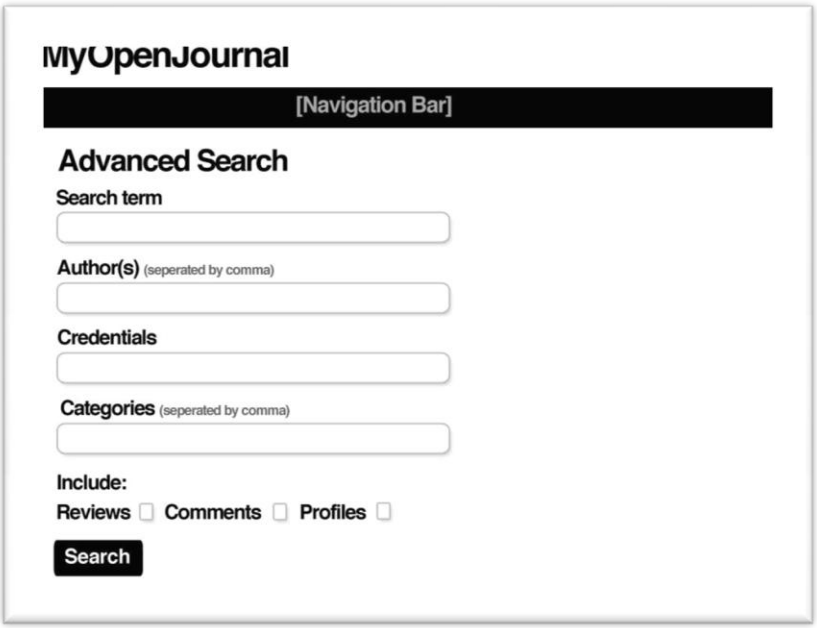

Figure 2.3: "Advanced Search" page for the MyOpenJournal website.

The Contact Us page will provide users with information on how to contact the administration team of MyOpenJournal. A Phone number and email will be provided for the user to report any problem or to request any help.

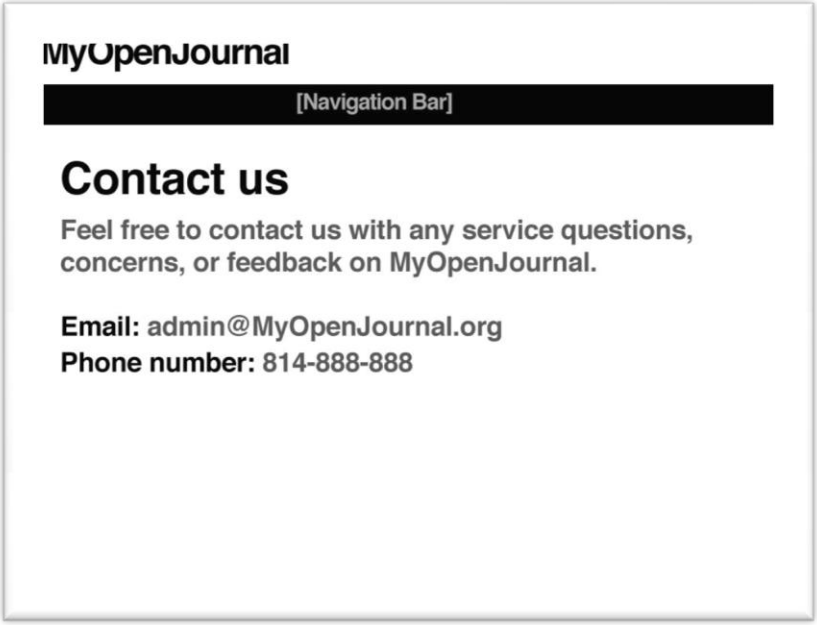

Figure 2.4: "Contact Us" page for the MyOpenJournal website.

The Edit Profile page allows users to input (or edit) different pieces of information to describe themselves. This page also allows users to change their password.

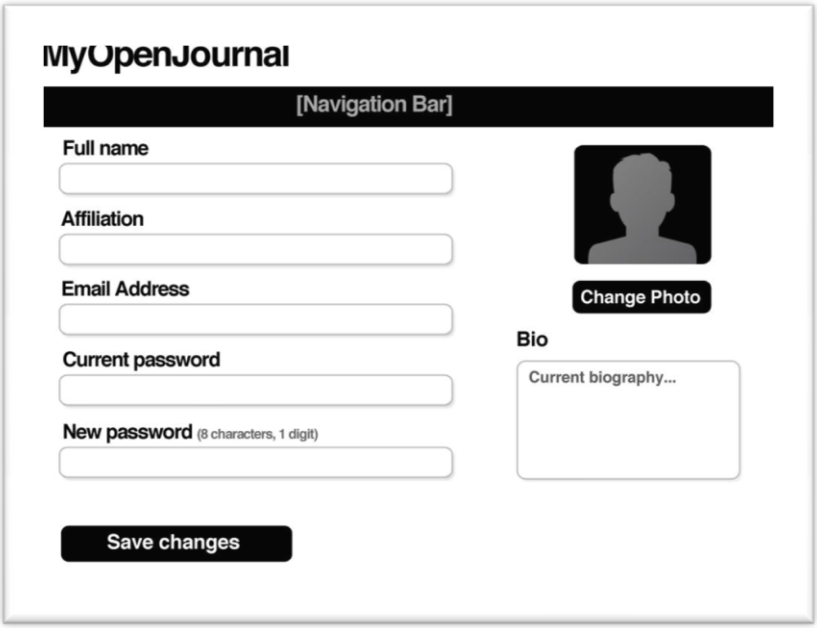

Figure 2.5: "Edit Profile" page for the MyOpenJournal website.

The Home Page of the MyOpenJournal site will list details of the most recent and top ranked papers. A feature that we also added was to allow users to customize their home page. This lets users control what they see when they first log into the website.

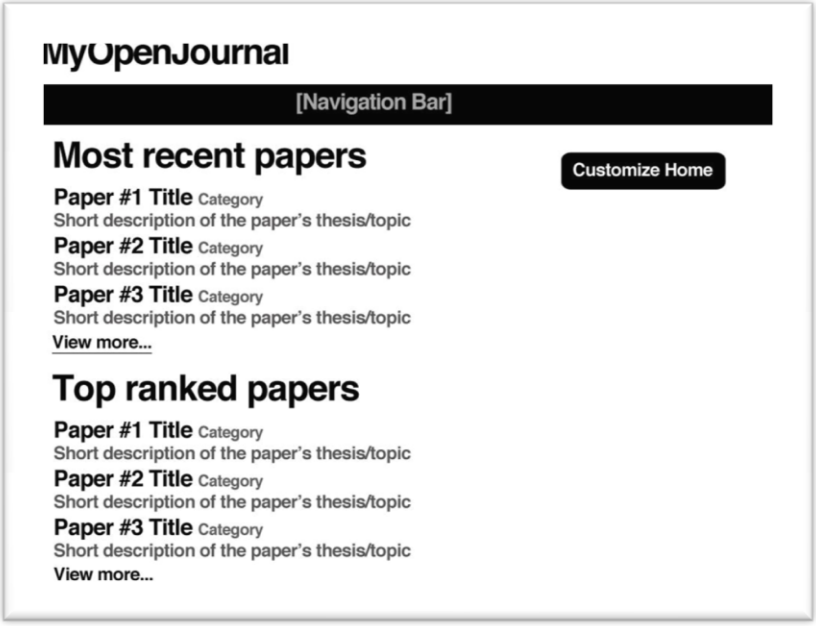

Figure 2.6: Home page of MyOpenJournal website.

The Login page of MyOpenJournal allows users to access their previously registered account. If the user is not already a member, it allows for the user to register for the site as well.

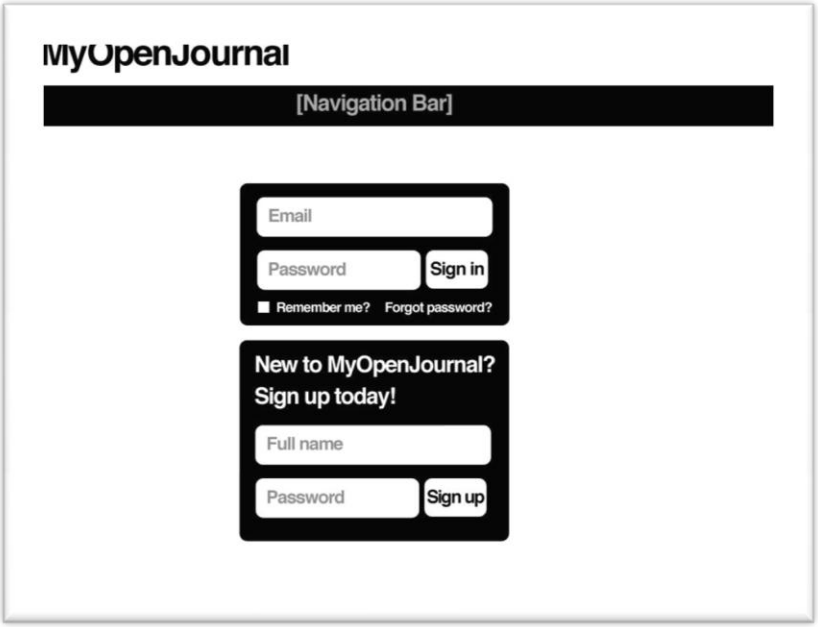

Figure 2.7: "Login" page for the MyOpenJournal website.

The View Paper page of MyOpenJournal allows a user to view the specific details of a paper. Some of these details include the views, author, upvotes, and comments.

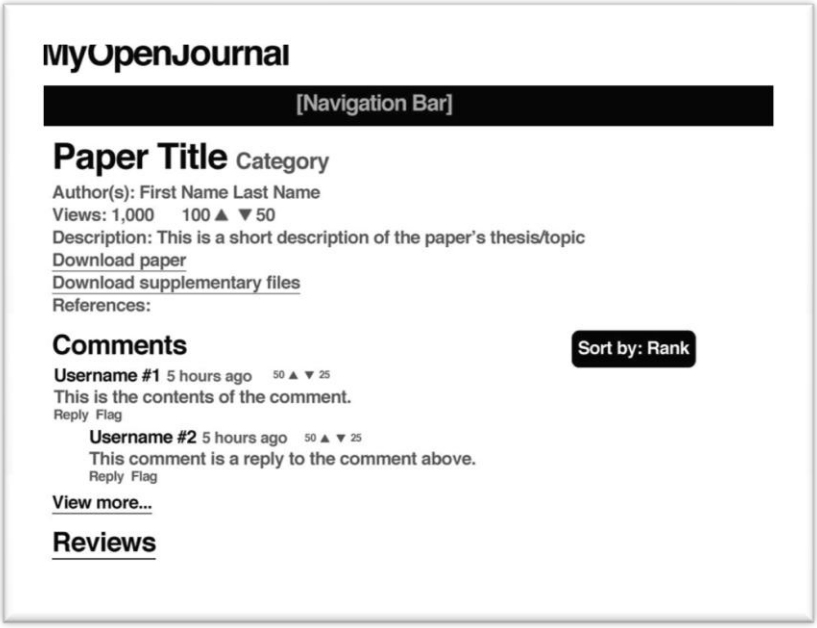

Figure 2.8: "View Paper" page on MyOpenJournal website.

The User Profile page of the MyOpenJournal website displays a user's photograph, name, affiliation, credentials, and the papers that they have uploaded. The papers that are listed will contain the title as well as a short description and the categories that it is listed under.

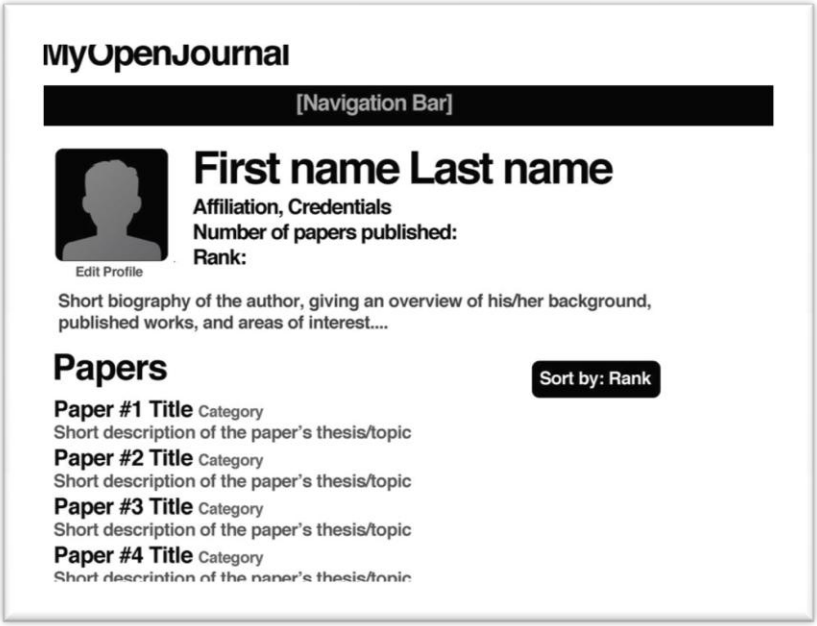

Figure 2.9: "User Profile" page on MyOpenJournal website.

The Register page of MyOpenJournal allows users to register for the website. Full name, username, email address, password, and affiliation are all required fields for registration.

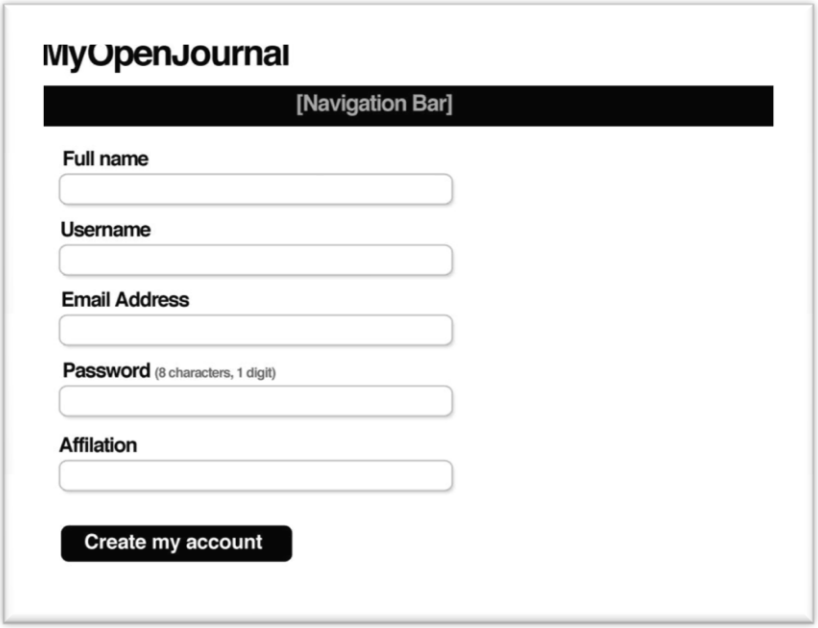

Figure 2.10: "Register" page for the MyOpenJournal website.

The Search page of the MyOpenJournal website allows a user to search for papers that have been uploaded. Unlike the Advanced Search page, this page only allows search by terms.

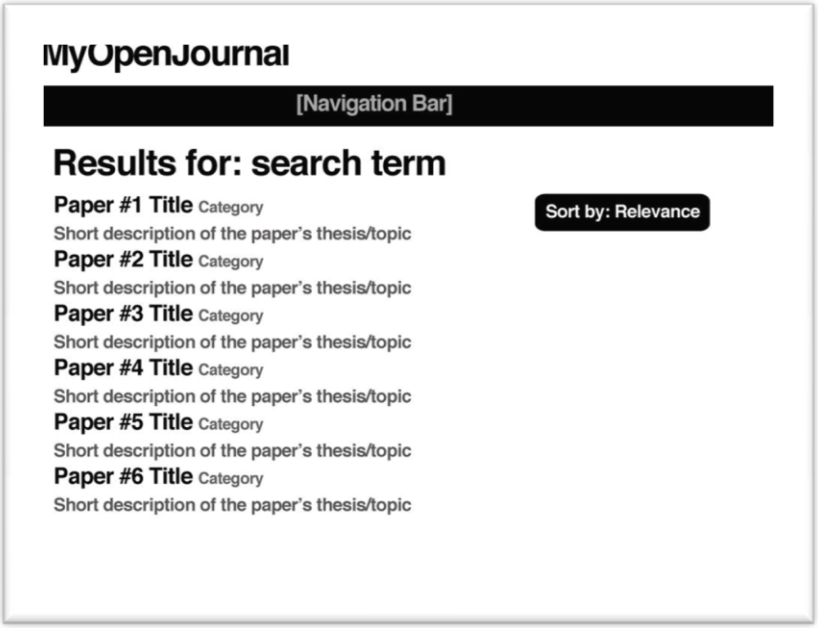

Figure 2.11: "Search" page for the MyOpenJournal website.

The Upload Paper page allows logged in users to upload a paper. The user may enter a title, co-authors, references, category, tags, and description for their paper. The user is then allowed to upload the paper and attachments.

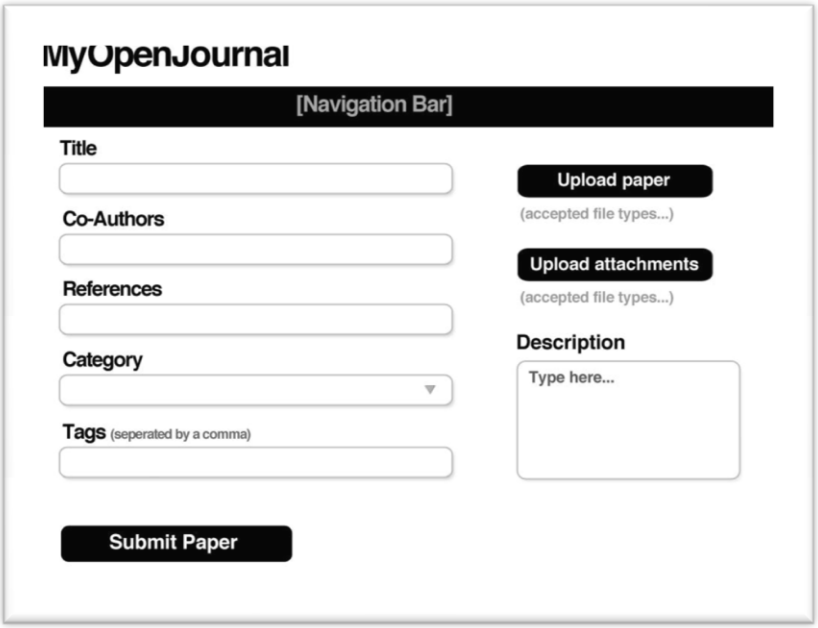

Figure 2.12: "Upload Paper" page for the MyOpenJournal website.

The Reviews page of the website lists reviews that have been written for papers. These reviews can be listed by rank, and comments are listed as well.

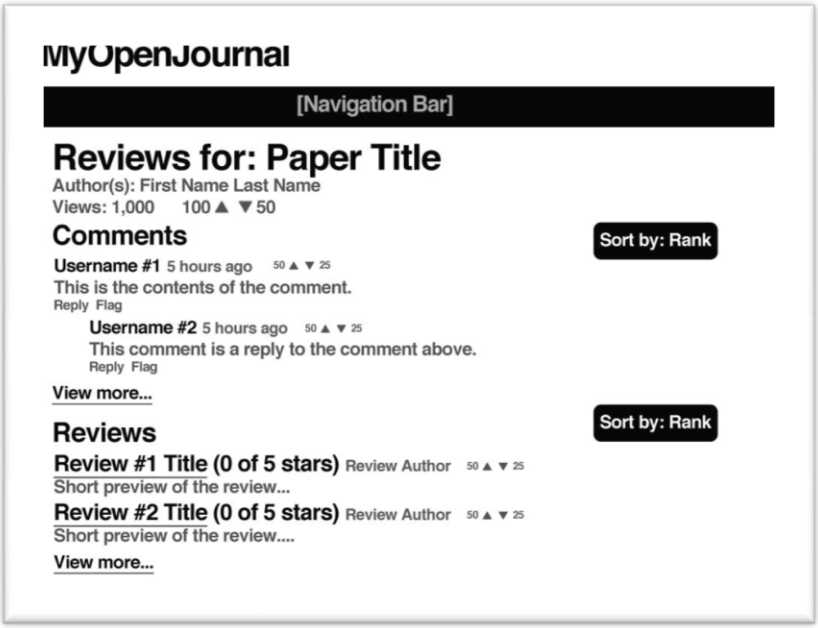

Figure 2.13: "Reviews" page for a paper on the MyOpenJournal website.

3. Technical Specifications

3.1 Development Tools and Languages

The main development tool for this project is the Eclipse IDE. This IDE supports Java and SQL development. ZK is used to develop the user interface and plugs in to Eclipse for easy use. ZK is a framework written in Java that has a lot of interface functionality built in such as text fields and action buttons. It allows for quick and easy interface development in a WYSIWYG (What You See Is What You Get) environment. Windows SQL Server Manager is used to maintain the database and the Database Development Tools plug-in for Eclipse allows for easy integration of the database with our code. The primary languages for this project are Java, SQL, HTML, ZUL, JavaScript, and zkscript.

The server to host the website will be running XAMP, which has an Apache application server and Microsoft SQL database. The project is developed using an AGILE method. Weekly meetings with team members and the customer will ensure continued progress.

Below is the architecture for the ZK Framework. We implicitly use this framework within our project because we have developed with ZK studio.

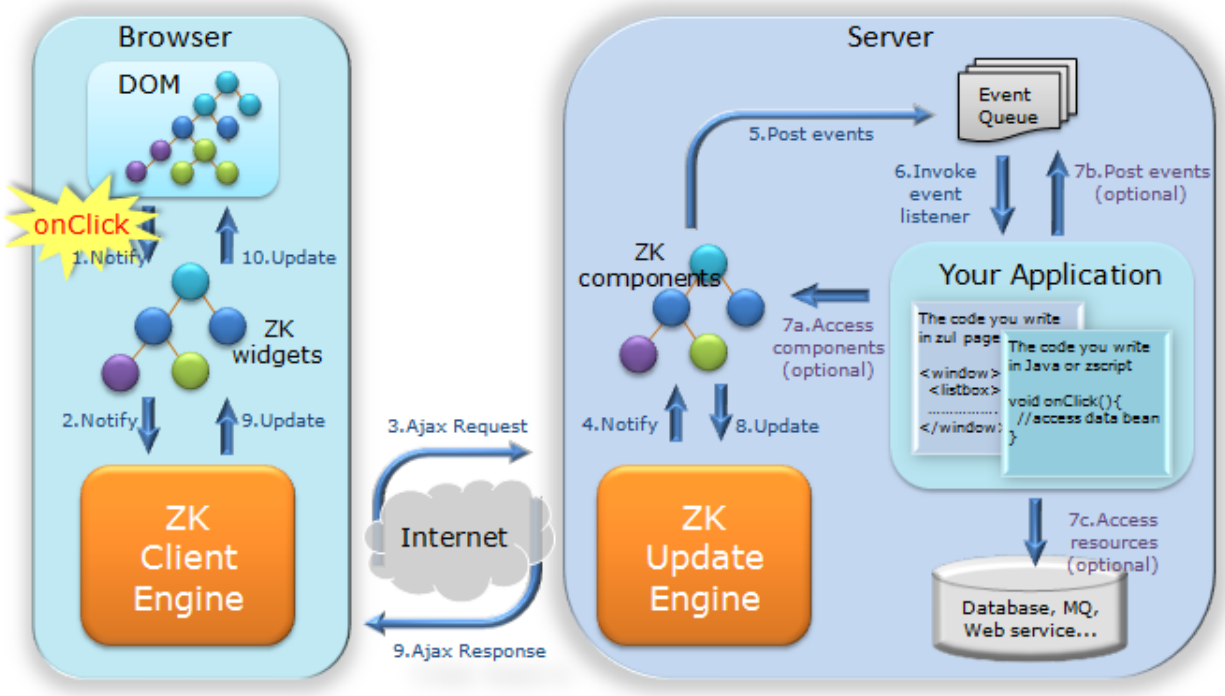

Figure 3.1: ZK Framework Architecture (taken from http://books.zkoss.org/wiki/)

3.1.2 Source Control

For this project, GitHub was used for source control. The repository is located at: https://github.com/thp5016/My-Open-Journal

Follow these steps to import the project into Eclipse from GitHub:

1.) First, make sure to install Eclipse and install the ZK Community Edition Plug-in.
2.) Download the GitHub GUI for Windows or Mac.
3.) In the GitHub GUI, login to your GitHub account and clone the My-Open-Journal repo to a local directory.
4.) Open Eclipse with ZK studio installed, and set your workspace to the workspace in the files you cloned from GitHub on your local drive.
5.) To open the MyOpenJournal project in Eclipse, go to File->Import->General->Existing Projects into Workspace and select the project folder from your local copy of the GitHub repository. For example, if you cloned into "C:/git/My-Open-Journal/workspace/MyOpenJournal", select the "MyOpenJournal" folder under "workspace" as the project folder to import.

If you would like GitHub functionality within Eclipse, download eGit from the Eclipse marketplace.

3.2 Architecture

3.2.1 Application Modularity Diagram

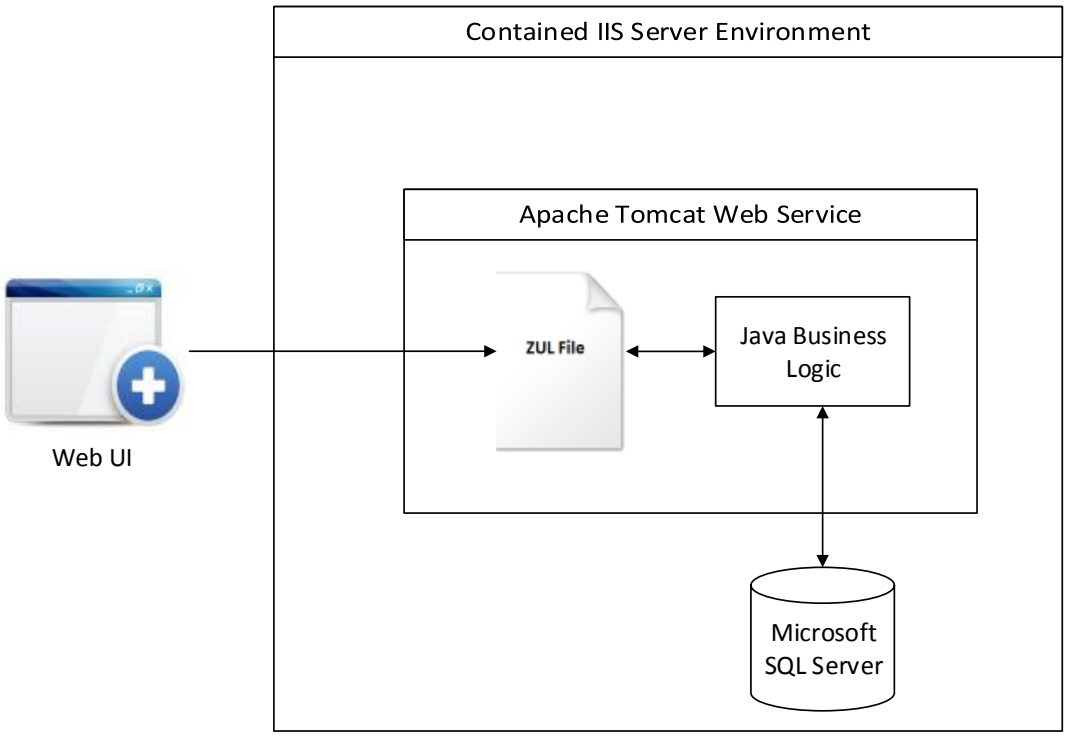

Figure 3.2: Application Modularity Diagram

3.2.2 UML Diagram

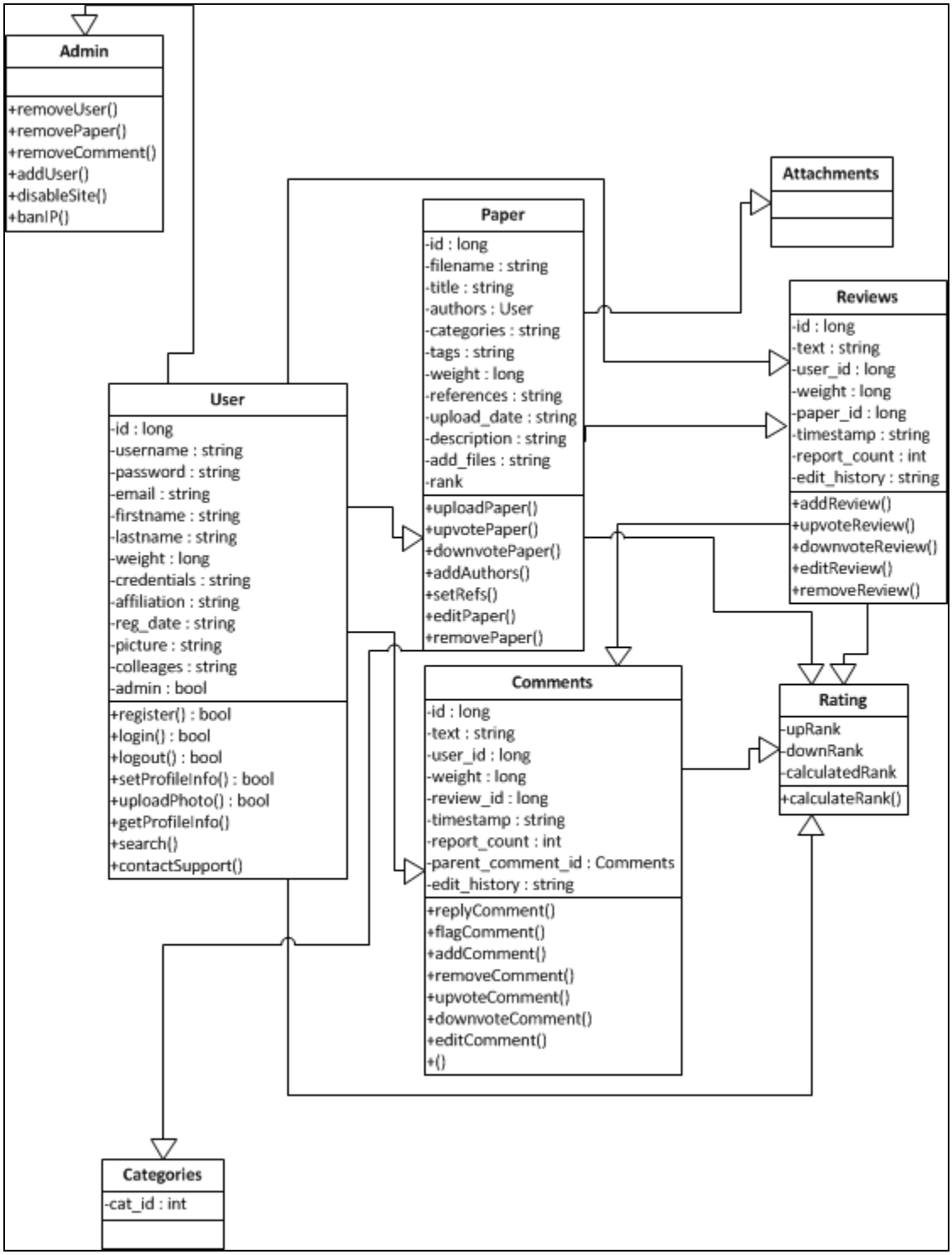

Figure 3.3: UML Diagram with Class relationships

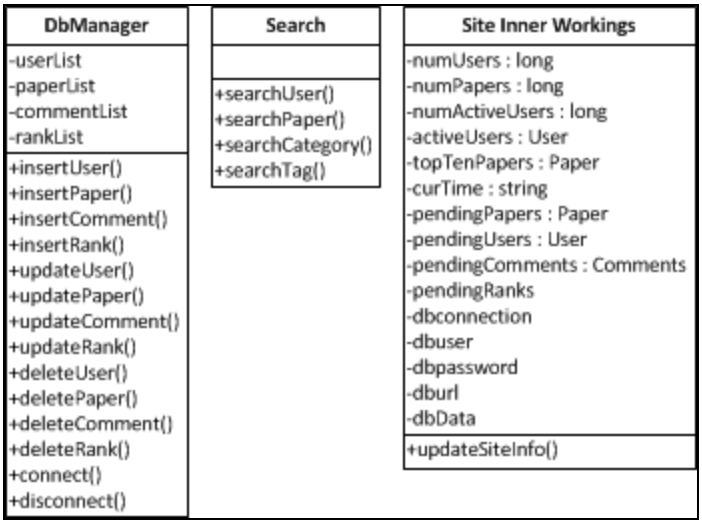

Figure 3.4: UML Diagram Continued

3.2.3 Function Prototypes

DBManager Class

The DBManager.java class provides database functionality for the MyOpenJournal website and contains all the SQL queries to modify the database. Any other class that requires database functionality will call methods from this class to build the queries and access the database.

Prototype	`public boolean InsertUser(String user, String firstName, String lastName, String pass, String admin)`
Pre-Condition	User is a valid username, firstName is a valid first name of user, lastName is a valid last name of user, pass is the user's password, admin is a flag on whether or not the user is an administrator account.
Post-Condition	This function inserts a user into the database by building the insert query from the passed parameters. If it is successful, it returns true. If there is an error, it returns false and prints the error to standard output.

Prototype	`public boolean ChangeProfile(String user, String password, String firstName, String lastName)`
Pre-Condition	User is a valid username, firstName is a valid first name of user, lastName is a valid last name of user, password is the user's password.
Post-Condition	This function edits a user's profile information by building an update statement .

Prototype	`public int GetID(String user)`
Pre-Condition	User is a valid username
Post-Condition	This function returns the userID of a user given the username. If the username is not found, then -1 is returned.

Prototype	`public boolean InsertPaper(int authorID, String title, String path, String description, String date)`
Pre-Condition	authorID is a valid userID, title is the title of the paper, path is the file path of the paper (usually a PDF file), description is the description of the paper, and date is the date of publication.
Post-Condition	This function inserts a paper into the database by building the insert statement from the passed parameters and then running that query through a prepared statement (to prevent against SQL injection) on a DBConnection. It returns true if the paper was successfully inserted into the database, and false if there was an error.

Prototype	`public String GetPaperPath(int id)`
Pre-Condition	id is a valid PaperID in the database.
Post-Condition	This function returns the file path of a paper given its paperID. If the paper does not exist, null is returned.

Prototype	`public List<Data> GetTopPapers()`
Pre-Condition	No pre-conditions.
Post-Condition	This function returns a List object with Data type objects which includes the top ranked papers currently in the database. If no papers are found, null is returned.

Prototype	`public String GetFirstName(String user)`
Pre-Condition	user is a valid username.
Post-Condition	This function returns the first name of a user given a username. If the user is not found, then null is returned.

Prototype	`public String GetLastName(String user)`
Pre-Condition	user is a valid username.
Post-Condition	This function returns the last name of a user given a username. If the user is not found, then null is returned.

Prototype	`List<Data> GetNewPapers()`
Pre-Condition	No pre-conditions.
Post-Condition	This function returns a List object with Data type objects which includes the newest papers uploaded to the database based on the current date. If no papers are found, then null is returned.

Prototype	`public boolean IsValidPassword(String user, String pass)`
Pre-Condition	user is a valid username and pass is the user's password.
Post-Condition	This function takes a hashed password pass and compares it to the hashed password on the database that corresponds to the passed user name. If the user's hashed password on the database matches the passed hashed password, then it returns true. Otherwise, returns false.

Prototype	`public boolean IsValidUser(String user)`
Pre-Condition	user is a valid username
Post-Condition	This function checks the passed username and queries the database to check if that user exists. If user exists, then returns true. Otherwise, returns false.

Prototype	`public boolean InsertComment(int reviewID, int userID, String comment)`
Pre-Condition	reviewID is a valid reviewID, userID is a valid userID, comment is the text for the comment
Post-Condition	This function inserts a comment based on the passed parameters into the database.

Prototype	`public boolean InsertReview(int paperID, int userID, String review)`
Pre-Condition	paperID is a valid paperID, userID is a valid userID, review is the text for the review.
Post-Condition	This function inserts a review based on the passed parameters into the database.

DBConnection Class

The DBConnection.java class provides database connectivity functionality for the MyOpenJournal business logic. No queries are contained within this class, but worker functions to connect and execute queries on a database.

Prototype	`public DBConnection(String server, String database, String user, String password)`
Pre-Condition	server is the IP address of the MSQL server, database is the database name to connect to, user is the username for the database, and password is the password for the database.
Post-Condition	This constructor function creates a new DBConnection object with class member connection of type Connection.

Prototype	`public void Connect(String server, String DBName, String user, String pass)`
Pre-Condition	server is the IP address of the MSQL server, DBName is the name of the database, user is the username for the database, and password is the database password
Post-Condition	This function connects to a Microsoft SQL database using the jdbc driver. An exception is thrown if the connection fails and a message "Failure to load driver!!" is printed to standard output.

Prototype	`public void Execute(String query){`
Pre-Condition	query is a valid MSQL query statement.
Post-Condition	This function executes a MSQL query on the class member connection.

Prototype	`public Connection GetConnection()`
Pre-Condition	No preconditions.
Post-Condition	This function returns the class member connection when a connection is needed to the database.

Prototype	`public void Disconnect()`
Pre-Condition	No preconditions.
Post-Condition	This function closes the class member connection.

SessionManager Class

The SessionManager.java class provides session functionality for users so that user-specific content can be delivered across multiple pages. Sessions as implemented in the ZK library allow users to "login" and "logout" of the website so that they can upload papers, edit their profiles, and make comments/reviews.

Prototype	`public static void setSession(String username, String password)`
Pre-Condition	username is a valid username, and password is the user's password (Salted and Hashed)
Post-Condition	This function sets the current session for the user so that user specific functionality can be delivered across different pages. After the session is set, the user is redirected to the home page.

Prototype	`public static String GetUser()`
Pre-Condition	No preconditions.
Post-Condition	This function gets and returns the username from the current session the user is using. If no session exists, then null is returned.
Prototype	`public static void Logout()`
Pre-Condition	No preconditions.
Post-Condition	This function removes the current session the user is using. If no session exists, then no action is taken.

Prototype	`public boolean checkSession()`
Pre-Condition	No preconditions.
Post-Condition	This function checks to see if a user is logged in (i.e. a session exists on that users computer). If a login session exists, it returns true. If no session exists, returns false.

Prototype	`public void doInit(Page arg0, Map<String, Object> arg1)`
Pre-Condition	arg0 is a Page object, arg1 is a Map with String key and Object value.
Post-Condition	This function overrides the doInit function of the Initiator class that runs each time a zul page is loaded. It checks for a user session, and if one is found, then the navbar for a user is displayed. Otherwise, if there is no session, the general navbar is shown on the page.

Prototype	`public static String saltAndHash(String input)`
Pre-Condition	input is some String.
Post-Condition	This function applies a salt to the input and then runs input through the MessageDigest implementation of MD5 hash function. It returns the salted and hashed value of input as a String.

Data Class

The Data.java class provides a data structure for the upvotes, downvotes, title and id of a paper. It also has get and set functions for these class members. Only the constructor function is shown, the get and set functions are left out of this document because of their implicit nature.

Prototype	`public Data(String paperTitle, String numUpvotes, String numDownvotes, int paperID)`
Pre-Condition	paperTitle is the title of a paper, numUpvotes is the number of up votes for the given paperTitle, numDownvotes is the number of down votes for the given paperTitle, and paperID is the ID (primary key) of the paper.
Post-Condition	This constructor function sets the class members upvotes, downvotes, title, and id to their respective passed parameters.

Home Class

The Home.java class contains functions for use on the home page of the MyOpenJournal website. This class contains no data members.

Prototype	`public static void GoHome()`
Pre-Condition	No preconditions.
Post-Condition	This function redirects the user to the home page (index.zul).

Prototype	`public static void DisplayResult(Grid myGrid, List<Data> data)`
Pre-Condition	myGride is the current Grid object, and data is a List of Data objects.
Post-Condition	This function displays the top papers on the passed myGrid object.

AddComment Class

This class provides functionality to add a comment on the MyOpenJournal website.

Prototype	`public void submitComment()`
Pre-Condition	@Listen for the button click of "Submit Comment"
Post-Condition	This function submits a review by calling the DBManager function insertComment() with the reviewID the comment is being made to, the userid, and the comment.

Prototype	`public void goBack()`
Pre-Condition	No preconditions.
Post-Condition	This function redirects the user back to the review the comment is on.

AddReview Class

This class provides functionality to add a review to a paper on the MyOpenJournal website.

Prototype	`public void submitReview()`
Pre-Condition	@Listen for the button click of "Submit Review"
Post-Condition	This function submits a review by calling the DBManager function insertReview() with the paperID the review is being posted to, the user id that is posting the review, and the review itself.

Prototype	`public void goBack()`
Pre-Condition	No preconditions.
Post-Condition	This function redirects the user back to the paper.

Paper Class

The Paper.java class provides functionality for viewing a paper, upvoting, downvoting, viewing a review and viewing comments.

Prototype	`public void viewPaper()`
Pre-Condition	@Listen for click of "Download Link" button.
Post-Condition	This function redirects the user to the path that the PDF file of the paper is stored.

Prototype	`public void upVote()`
Pre-Condition	@Listen for click of up vote button
Post-Condition	This function inserts an up vote for the paper ID the user is currently viewing.

Prototype	`public void downVote()`
Pre-Condition	@Listen for click of down vote button
Post-Condition	This function inserts a down vote for the paper ID the user is currently viewing.

Prototype	`public void viewReview()`
Pre-Condition	@Listen for click of review link
Post-Condition	This function redirects the user to the review for the paper so that they can comment on it, etc.

Prototype	`public void addReview()`
Pre-Condition	@Listen for click of add review link
Post-Condition	This function redirects the user to a page where they can submit a review for the paper they are currently viewing.

Upload Class

The Upload.java class provides functionality to insert a paper from the upload.zul page on the website. It makes calls to the DBManager class.

Prototype	`public void InsertPaper()`
Pre-Condition	Title, description, and filePath have been successfully entered into the text fields.
Post-Condition	This function displays the path that the user has entered. The function then calls the InsertPaper function of the DBManager class with the parameters being the current date, current user, title, description, and filePath.

Prototype	`public void UploadPaper(UploadEvent event)`
Pre-Condition	Event is a valid UploadEvent object created by the user interacting with the upload button.
Post-Condition	This function uploads the file that the user has provided to the server. If the user has not chosen a file to upload, then "NULL!!" will be printed to the screen. If the copy of the user's file to the server fails, then "fail!!" will be printed to the screen.

Register Class

The Register.java class provides user registration functionality for the website.

Prototype	`public void InsertUser()`
Pre-Condition	The user's entered data for first, last, username, email, and pwd are all valid inputs and have been provided.
Post-Condition	This function calls the saltAndHash function of the SessionManager class with the entered password in order to encrypt the password before passing it. The function then calls the InsertUser function of the DBManager class with the parameters being the username, first name, last name, encrypted password, email, 0 (to trigger not an admin), and the current date. Once the user selects the register button, the user is redirected to the homepage.

Login Class

The Login.java class provides functionality for a user to log into the website.

Prototype	`public void LoginUser()`
Pre-Condition	The user's entered values for username and pwd are valid entries. In addition, the user must have interacted with the login button to call the function.
Post-Condition	The IsValidUser function of the DBManager class is called with the entered username to ensure that the specified username belongs to a registered user. If the user is not registered, then "Invalid User!!" is printed to the screen. If the user is registered, the IsValidPassword function of the DBManager is then called to ensure that the password is correct for the specified username. If the password entered is not valid, then "Incorrect Password!!" is printed to the screen.

Prototype	`public void RegisterUser()`
Pre-Condition	The user has clicked on the Register button.
Post-Condition	The user is redirected to the Register page via the sendRedirect function of the Executions class.

Logout Class

The Logout.java class allows a logged in user to log out of the website.

Prototype	`public void doInit(Page page, Map<String, Object> args)`
Pre-Condition	The user has clicked on the logout button.
Post-Condition	The Logout function of the SessionManager class is called, and the user is redirected to the homepage via the sendRedirect function of the Executions class.

EditProfile Class

This class provides functionality for a user to edit his/her profile on the MyOpenJournal website.

Prototype	`public void onCreate$firstName()`
Pre-Condition	A current user session exits.
Post-Condition	This function sets the firstName text box on the edit profile page to the user's first name based on the active session. It pulls this information from the database based on the username stored in the current session.

Prototype	`public void onCreate$lastName()`
Pre-Condition	A current user session exits.
Post-Condition	This function sets the lastName text box on the edit profile page to the user's last name based on the active session. It pulls this information from the database based on the username stored in the current session.

Prototype	`public void onClick$saveChanges()`
Pre-Condition	A current user session exits.
Post-Condition	This function saves the form data on the edit profile page to the database, updating the password fields through the ChangeProfile() function in the DBManager class.

3.2.4 ER Diagram

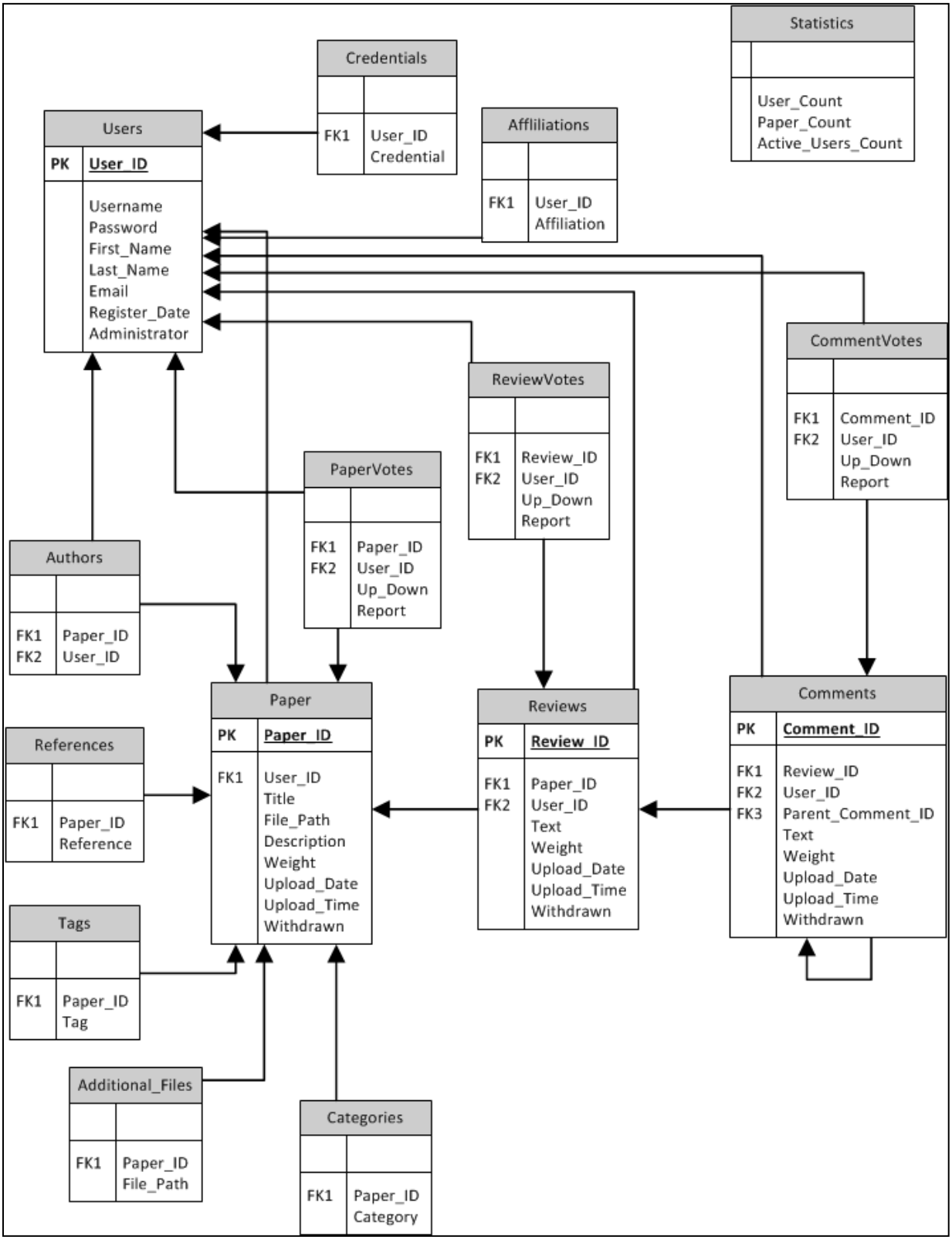

Figure 3.5: Database ER Diagram

3.3 Server Configuration

Microsoft IIS server can be configured to run alongside Apache Tomcat by forwarding traffic using the Application Request Routing extension provided by Microsoft for IIS. (http://www.iis.net/downloads/microsoft/application-request-routing).

Our server configuration is running Apache Tomcat v. 7.x with the Microsoft Application Request Routing extension installed in Microsoft IIS. The instructions to set up this Application Request Routing can be found here: http://www.iisadmin.co.uk/?p=326

4. Standards

Knowing that software is almost never maintained by the original author for its lifetime, and to keep order and readability of the project code, the Java and MSSQL coding standards were used. In order to ensure future endeavors of the MyOpenJournal project are more fluid, these standards were implemented from the beginning of coding.

With Java in mind, all code was developed according to the "Code Conventions for the Java Programming Language." This includes but is not limited to: File Names, File Organization, Indentation, Comments, Declarations, Statements, Naming Conventions, and Other Programming Practices (http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html).

With MSSQL in mind, all code was developed according to common MSSQL convention. These rules include, but are not limited to: Naming Conventions, Data Types, and Join Techniques. With the use of these coding standards, we ensure that future development teams can easily understand code and continue development.

5. Testing

General Setup for Testing

If not done so already, insert this user into the database:

```
Insert into Users (Username, First_Name, Last_Name, Password)
values ('test', 'test', 'test', 'test');
```

5.1 Integration Testing

Test 1: DBManager::IsValidUser()

This tests a function in the DBManager class.

Test class used for this test:

```
package Test;
public class Test extends SelectorComposer<Component> {
     DBManager manager = new DBManager();
     manager.InsertUser("test");
}
```

Setup/Action	Result	Yes/No Comments
1. Create a test class that declares and instantiates a DBManager object in ZK studio 2. Call the DBManager class function as shown above: `IsValidUser("test");`	True	

The pieces of the MyOpenJournal architecture include the zul webpages written in java/html and the MS SQL database. These tests test the integration of these two pieces.

Test 2: Database Insertion of New User

This tests the integration of the zul file displayed in the web browser and the connection with the SQL server to ensure that queries generated on the web page get executed on the SQL server.

Setup/Action	Result	Yes/No Comments
1. Open www.myopenjournal.org/register.zul in a web browser 2. Insert information into each field of the form as follows: First Name: "Insert" Last Name: "Test" Username: inserttest Password: abc123 Email: inserttest@gmail.com 3. Click the "Register" button 4. Run the following query on the SQL server: `SELECT from Users WHERE username='inserttest';`	A row is returned with the information inserted in step 2	

After test, run:

```
DELETE from Users WHERE username='inserttest';
```

Test 3: Database Insertion of New Paper

This tests the integration of the zul file displayed in the web browser and the connection with the SQL server to ensure that queries generated on the web page get executed on the SQL server.

Setup/Action	Result	Yes/No Comments
1. Open www.myopenjournal.org/register.zul in a web browser 2. Insert information into each field of the form as follows: First Name: "test" Last Name: "test" Username: test Password: test Email: test@gmail.com 3. Click the "Register" button 4. Go to http://www.myopenjournal.org/login.zul in a web browser. 5. Insert information into each field of the form as follows: Username: test Password: test 6. Click the "Sign in" button 7. Open www.myopenjournal.org/addpaper.zul in a web browser 8. Insert paper information into the form with title "Test Paper" 9. Click the "Submit" button 10. Run the following query on the SQL server: `SELECT from Papers WHERE Title='Test Paper';`	A row is returned with the title "Test Paper"	

After test, run:

```
DELETE from Papers WHERE Title='Test Paper';
```

Test 4: Database Update of User Information

This tests the integration of the zul file displayed in the web browser and the connection with the SQL server to ensure that queries generated on the web page get executed on the SQL server.

Setup/Action	Result	Yes/No Comments
1. Perform registration and log in as user "test" as prescribed in Test 3 if not already done so. 1. Navigate to http://www.myopenjournal.org/editprofile.zul 2. Modify the information in "Last name" field to : Last name: update 3. Click "Save changes" button 4. Run the following query on the SQL server: `SELECT from Users WHERE Last_Name = 'update';`	A row is returned with the username "test"	

Test 5: Upvote Paper Functionality

This tests the integration of the zul file displayed in the web browser and the connection with the SQL server to ensure that queries generated on the web page get executed on the SQL server.

Setup/Action	Result	Yes/No Comments
1. Perform registration and log in as user "test" as prescribed in Test 3 if not already done so. 2. Open www.myopenjournal.org/addpaper.zul in a web browser 3. Insert paper information into the form with title "Test Paper" 4. Click the "Submit" button 5. Click "OK" in the dialog box 6. Under the category "Newest", click the paper titled "Test Paper" 7. Click the green button with the arrow labeled "up", giving the newly uploaded paper its first upvote 8. Run the following query on the SQL server: `SELECT from Papers WHERE Title='Test Paper';`	A row is returned with the title "Test Paper" and the "Upvotes" field is set to 1.	

After test, run:

```
DELETE from Papers WHERE Title='Test Paper';
```

5.2 Unit Testing

Test 1: TBD

Two function tests (can’t query database)

5.3 System Testing

Test 1: Registering a User

In this test, a user attempts to enter information and register to MyOpenJournal.

Setup/Action	Result	Yes/No Comments
1. Open www.myopenjournal.org/register.zul in a web browser 2. Insert information into each field of the form as follows: First Name: "test" Last Name: "test" Username: test Password: test Email: bjones@gmail.com 3. Click the "Register" button	"test has successfully registered!!" dialog appears on the web page.	
4. Click "OK" button in dialog box	The browser returns to the homepage.	

Test 2: Logging In

This tests the ability of a user registered in the database to successfully login and create a session.

Setup/Action	Result	Yes/No Comments
1. Perform registration of user "test" as prescribed in Test 1 if not already done so. 2. Go to http://www.myopenjournal.org/login.zul in a web browser. 3. Insert information into each field of the form as follows: Username: test Password: test 4. Click the "Sign in" button	"test has successfully logged in!!" dialog appears on the web page.	
5. Click "OK" button in dialog box	The browser returns to the homepage. The link to "Login" in the navigation bar is replaced by a link to "Logout". Links to "Profile" and "Submit Paper" are now in the navigation bar and the link to "Register" is gone.	

Test 3: Edit Profile

This tests the ability of a user to update the profile information entered at the time of registry to MyOpenJournal.

Setup/Action	Result	Yes/No Comments
2. Perform registration of user "test" as prescribed in Test 1 and log in as user "test" as prescribed in Test 2 if not already done so. 3. Navigate to http://www.myopenjournal.org/editprofile.zul 4. Modify the information in "Last name" field to : Last name: newLastName 5. Click the button "Save changes"	"Your profile has been successfully updated" dialog appears on the web page.	
6. Navigate to http://www.myopenjournal.org/editprofile.zul	The "Last name" field is "newLastName".	

Test 4: Logging Out

This tests that a user currently logged in is able to successfully logout and the site returns to guest mode.

Setup/Action	Result	Yes/No Comments
1. Perform registration of user "test" as prescribed in Test 1 and log in as user "test" as prescribed in Test 2 if not already done so. 2. Navigate to http://www.myopenjournal.org/ 3. Click "Logout" in the navigation bar	"You have successfully logged out!!" dialog appears on the web page.	
4. Click "OK" in the dialog box	The browser returns to the homepage, with the link in the navigation bar to "Logout" replaced by a link to "Login". Links to "Submit Paper" and "Profile" are gone and a link to "Register" is now in the navigation bar.	

Test 5: "Newest" Paper Category on Home Page

This test confirms that the "Newest" category on the home page is updated properly after a user uploads a paper

Setup/Action	Result	Yes/No Comments
1. Perform registration of user "test" as prescribed in Test 1 and log in as user "test" as prescribed in Test 2 if not already done so. 2. Open www.myopenjournal.org/addpaper.zul in a web browser 3. Insert paper information into the form with title "Test Paper" and some arbitrary description. 4. Click the "Submit" button 5. Click "OK" in the dialog box. 6. Return to www.myopenjournal.org	A paper titled "Test Paper" appears under the category "Newest".	

5.4 Acceptance Testing

Navigate to Home Page:

Action	Result	Yes / No / Comments
1. Open a web browser.	Browser window appears.	
2. In the navigation bar, type in "www.myopenjournal.org"	The MyOpenJournal home page appears with logo, navigation bar, and 3 paper listing columns.	

Attempt to Login without Registration:

Action	Result	Yes / No / Comments
1. Click the "Login" link on the navigation bar.	The login page appears with prompt.	
2. Attempt to login with invalid credentials.	A message box with "Invalid username or password" should appear.	

Registration:

Action	Result	Yes / No / Comments
1. Click the "Register" button on the login page or on the navbar.	The registration page appears.	
2. Fill out the registration form and use the spaces below of the information you use: First Name: _____ Last Name: _____ Username: _____ Password: _____ Email: _____, then click the "Register" button.	A message box appears with the text "<username> has successfully registered!" appears.	
3. Click "OK" on the pop up message box.	Redirect to the home page.	

Login:

Action	Result	Yes / No / Comments
1. Click the “Login” link on the navigation bar at the top of the page.	The login page appears.	
2. Fill out the login form the username and password you noted above: Username: _____ Password: _____, then click the “Log In” button.	A message box with the dialog “<username> has successfully logged in!!” appears.	
3. Click “OK” on the pop up message box.	Redirect to the home page.	

Edit User Profile.

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page should appear with user logged in.	
2. Click the “Profile” link on the navbar.	The user profile information should appear.	
3. Enter in the current password and a new password: _____. and click “Save Changes”.	A message box with “Your profile has been successfully updated!!” should appear and you will be redirected back to the home page.	
4. Click the “OK” button on the pop-up message box.	Redirect to home page.	
5. Click the “Logout” link on the navbar.	A message box with “You have been successfully logged out!!” should appear.	
6. Click the “OK” button on the pop-up message box.	Redirect to home page.	
7. Click the “Login” link on the navbar.	Login page appears.	
8. Enter in the username as noted above and the new password as noted above.	A message box with “<username> has successfully logged in!!” appears.	
9. Click the “OK” button on the pop-up message box.	Redirect to home page.	

Upload a Paper:

Action	Result	Yes / No / Comments
1. Click the “Submit Paper” link on the navigation bar at the top of the page.	The paper submission page appears.	
2. Fill out the paper submission form and make notes of the information you use: Title: _____ Co-Authors: _____ References: _____ Category: _____ Tags: _____ Description: _____ _____ _____	The text fields are filled without error.	
3. Click the “Upload Paper” button.	A file browser window opens (Explorer on Windows / Finder on Mac / ??? on Linux).	
4. Select a PDF file to upload and make note of the filename below. Filename: _____ Click “OK”.	The file browser window closes, and filename you wrote down is displayed underneath the “Upload Paper” button.	
5. Click the “Submit Paper” button.	A message box with the dialog “You have successfully uploaded a paper!!” appears.	
6. Click the “OK” button on the pop-up message box.	Redirect to home page.	

View a Paper:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page should appear with user logged in.	
2. Click on the title of the paper in the "Newest Papers" column that you uploaded as noted above.	The paper page should appear.	
3. Click the "View This Paper" button.	The PDF file that you uploaded for the paper should appear or download.	
4. If your browser opened the PDF in the same window, then click the "Back" button in your browser. If not, bring the browser window with the paper page back into focus.	Paper page appears.	

Add a Review:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page appears with user logged in.	
2. Navigate to the page of the paper you uploaded as prescribed above.	The paper page appears.	
3. Click the "Add Your Own Review" button.	The add review page should appear.	
4. Click "Back to Paper" button.	Paper page appears.	
5. Repeat Step 3, and then go to step 6.	Submit Review page appears.	
6. Type a review: (at least 500 words) and click "Submit Review"	A message dialog appears with "Your review has been submitted!"	
7. Click the "OK" button on the pop-up message box.	Redirect to home page.	

View a Review:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page appears with user logged in.	
2. Navigate to the page of the paper you uploaded as prescribed above.	The paper page appears.	
3. Click on "View Reviews" button.	Reviews appear for that paper.	
4. Click on the review author (you) that you just submitted.	Your review appears.	

Upvote a Paper:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page appears with user logged in.	
2. Navigate to the page of the paper you uploaded as prescribed above.	The paper page appears.	
3. Click the green “Up” arrow on the paper page.	The page should refresh and the number above the green “Up” arrow should increment by 1.	

Downvote a Paper:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page appears with user logged in.	
2. Navigate to a different paper other than the one you uploaded as noted above.	The paper page appears.	
3. Click the red “Down” arrow on the paper page.	The page should refresh and the number above the red “Down” arrow should increment by 1.	

Upvote a Review:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page appears with user logged in.	
2. Navigate to the review that you submitted above.	The review appears.	
3. Click the green “Up” arrow on the review page.	The page should refresh and the number above the green “Up” arrow should increment by 1.	

Downvote a Review:

Action	Result	Yes / No / Comments
1. Perform the Login test as prescribed above if not already logged in.	The home page appears with user logged in.	
2. Post a review as prescribed in the Add Review Test above to any paper.	Review submitted.	
3. Navigate to the review you just submitted.	The review appears.	
3. Click the red “Down” arrow on the review page.	The page should refresh and the number above the red “Down” arrow should increment by 1.	

6. Future Improvements

Further Ranking

In our ranking system mentioned in section 2.4, we use up-votes, down-votes, and the upload date to calculate a ranking based on the activity that a paper has endured since its upload date. This allows all papers that have been uploaded to have a fair chance at being seen. We felt that this ranking gave users the best chance at obtaining feedback from other users.

In future iterations of MyOpenJournal, additional ranking systems could be implemented. Since our ranking system is solely used for listing papers on the homepage of the website, there are many more possibilities for ranking to be introduced. For instance, user ranking could be introduced as in the PHP version of MyOpenJournal in order to allow users to rate other users. We do not believe that this user rating should affect a paper's visibility when uploaded, because then new users will never have their papers seen over high ranked users. However, this user rating could be used to motivate all users to participate more on MyOpenJournal in order to increase their rating and thus their reputation.

For our iteration, we felt that the ranking mentioned in section 2.4 was adequate. Any further ranking that is introduced can be done in future implementations of MyOpenJournal.

Search

Due to time constraints and unforeseen obstacles, we were unable to implement a search feature into MyOpenJournal. Ideally the user will be able to search by author, tags, references, etc. using a friendly and easy to use user interface. This feature was not a necessary feature in a stable release of MyOpenJournal, so we chose to omit it in order to get more fundamental functions implemented.

DOI/Parsing References

When a user uploads a paper to MyOpenJournal, a program should automatically parse out the references that are listed at the bottom of the paper. If a reference paper is already in the MyOpenJournal system, the system will link the reference to the paper on the site. If a reference paper is not already in the MyOpenJournal system, the system will find the paper on the internet and link to it using a DOI (Digital Object Identifier). This feature was also not implemented due to time constraints in the project, and should be implemented in a future iteration of MyOpenJournal.