SWE Group 10 - Sprint 1

Andy Hine, Kate Watkins, Assia Hardiman, Alex Wilhelm, Wei Xian Low, Dewi Kharismawati

Group Meetings

Date: 10/27/2016: 2:00-3:15pm
Location: W0013 Lafferre
Attendance: All

Date: 10/30/2016: 7:00-9:00pm
Location: W2003 Lafferre
Attendance: All

Date: 10/31/2016: 2.30-4.30
Location: W2003 Lafferre
Attendance: All

Overview of Sprint Requirements:

The main goals and work distributions of this week's sprint are as follows:

1. General
    * Sprint documentation - Everyone
    * Complete Setup of Deployment Environment - Wei Xian
    * Organize GitHub Repo - Wei Xian

2. Database
    * Finalize ERD - Andy
    * Database Creation SQL - Andy
    * Implement DB, seed data for development Dewi

3. User Interface
    * Complete design of the user interface (all screens) - Alex
    * Complete design of the information architecture - Alex

4. Testing and Documentation
    * Build tests for data insertion, updating, and deletion - Kate, Assia
    * Regression Testing - Kate, Assia

Additional Documentation for Individual Tasks

Database Blueprint & Foundation - Andy Hine

The first step in this sprint was to finalize the system ERD. We will be using a MongoDB no-SQL database to implement this system, but a complete ERD helps everyone understand the complex relationships that are going on within the manifests and between the users. I completed this ERD and added it to our complete documentation. Along with the ERD, I created a mock SQL database to further visualize content and variety of our data. These documents were pushed into the repository on 10/30/2016.

Deployment - Wei Xian Low

Our web app for this project was created and deployed on Amazon Web Services (AWS), running on Ubuntu 14.04. A LAMP (Linux, Apache2, mySQL (Replaced with mongoDB), pHp) stack is installed on the server with mongoDB being used instead of mySQL.

Apache is then configured with the proper SSL certificate, forcing all connection to the server to be under HTTPS, ensuring a secure connection is maintained. The SSL certificate is then automatically renewed every day using a cron job, to ensure the SSL is up to date.

The URL to connect to the database is https://cs4320.weixianlow.me, there is also a testing subdomain (https://test.cs4320.weixianlow.me) to provide developers a place to test out the system without affecting the main working program.

A 'Hello World' page is then generated on the server and it's currently available on the URL.

Process connecting to server:

1. Obtain .pem file from Instance Owner
2. Convert .pem to .ppk file (depending on application you would use)
3. Every connection to the server either through SSH/Putty or FileZilla will require the .ppk for authentication (No password will be needed).
4. Connect to server with user account ubuntu.

Testing - Kate & Assia

Our initial testing is only done for the database. The test cases will provide a fail/success message upon operation. We are testing for different cases for insert, delete, and update operations. Our user testing includes gathering a small group of volunteers to test the database, and provide feedback.Additionally, every Sunday, we will do integration testing. Our future we will test uploading, downloading, browsing, and searching.

Testing

Database Implementation - Dewi

Our group utilized MongoDB for Database management purpose. After all the environments set, I created database for this project called SWE. SWE database consists of all collections corresponds to our ERD, such as users, manifest, dates, distribution, file, researchObject, creator, bibliographicCitation, and researchObject_creators.

We utilize the manifest.json file to taking care of the manifest, dates, distribution, file, researchObject, creator, bibliographicCitation, and researchObject_creators collections. We also created user collection separately because the json file does not include the user's information for loin and logout.

Two dummy data are created in each collection.

```
>monggo

//create the database name SWE
>use SWE

//create the collections
>db.createCollection("manifest")
>db.createCollection("user")

//inserting the documents
>db.manifest.insert({data inside the manifest.json})
>db.manifest.insert({data inside the manifest2.json})
>db.user.insert({manually insert user data})
>db.user.insert({manually insert user data})

//To show the documents
>db.manifest.find().pretty()
>db.user.find().pretty()
```

User Interface - Alex

The complete system was simplified down into 3 pages. The search page, the view/edit page, and the create/update page. The slash means that they are multi-purpose. In the search page, the user can search through the list using a manifest search and a keyword search. The manifest search allows for you to search everything while the keyword search allows you to only search through metadata like the creator and name of the manifest. Also, from the this page you can navigate to the other two pages. If you navigate towards the create/upload page you can either type up a new manifest or upload an existing manifest to the page. And if you go to the view/edit page you can you can see everything in the manifest and if you have permission you can edit it.