

OCDX Research Project

Full Documentation
28 November, 2016

Group 10:

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

GitHub: https://github.com/weixianlow/SWE_Group10

Drive: <https://drive.google.com/file/d/0Bzc5cB9bdfSUdXJXa3NvNEFBSDA/view?usp=sharing>

Deployment: <http://cs4320.weixianlow.me>

Table of Contents

• Requirements & Design Tasks List	2
• Task Assignments	2
• Requirements Analysis	3
○ User Requirements	3
○ System Requirements	4
○ Use Cases	4
• System Design	13
○ Screen Views	13
○ Class List	17
○ Table List	18
○ ERD	19
• Sprint 1 Documentation	23
• Sprint 2 Documentation	35
• Sprint 3 Documentation	51
○ Group Meetings	51
○ Task List.	52
○ Login, Logout & Sessions	53
○ Deployment	58
○ Temporal UI Logic	66
○ Use Case Analysis	67
○ User Documentation Outline.	68
• Sprint 4 Documentation	69
○ Group Meetings	69
○ Task List	69
○ Addressing Sprint 3 Feedback	70
○ UI	70
○ Upload/Create Manifest	71
○ List View/View Manifest	73
○ Use Case Analysis	74
○ Testing Done in Application	75
○ User Documentation	85
○ Deployment	85
• Change Log.	86
• Glossary	87

Requirements & Design

Requirements & Design Tasks List

- Requirements Analysis
 - User Requirements
 - System Requirements
 - Use Cases:
 - Browse Manifest
 - Upload Dataset
 - Contribute to Existing Dataset
 - Download Info
 - Generate Upload Manifest
 - Save
 - Search on Manifest
- System Design
 - Screen Views
 - Class List
 - Table List
 - ERD
 - UML
 - Activity Diagram
 - Sequence Diagram
 - Use Case Diagram

Task Assignments

- Andy
 - Primary Author: Download Info use case, table list, ERD
 - Secondary Reviewer: Upload Dataset use case, system requirements
- Alex
 - Primary Author: Contribute to Existing Dataset use case, Screen Views

- Secondary Reviewer: Browse Manifest use case, generate upload manifest use case
- Wei Xian
 - Primary Author: Browse Manifest use case, Search on Manifest use case, Activity Diagram
 - Secondary Reviewer: Save use case
- Assia
 - Primary Author: Generate Upload Manifest use case, User Requirements, Use Case Diagram
 - Secondary Reviewer: Search on Manifest use case
- Dewi
 - Primary Author: Save use case, Sequence Diagram
 - Secondary Reviewer: Contribute to Existing Dataset use case
- Kate
 - Primary Author: Upload Dataset use case, System Requirements, Class List
 - Secondary Reviewer: Download Info use case

Requirements Analysis

Expected Users

We begin by identifying four expected users for this system:

- Students - Read only, not affiliated with the research
- Researchers - Collecting data
- Data Scientists - Analyzing data/make predictions
- System Admins - Maintain the system

User Requirements

A typical user expects the following from this system:

- To be able to browse through a collection of research data that was compiled and formatted as per the OCDX specification.
- As a researcher, to be able to upload their own collections of data formatted as per the OCDX specification.
- As a researcher or a student, to be able to download collections of data to their local machines in order to further their own research, or for educational value.
- A Student should be able to view the data in a visually appealing format.
- Researches should be able modify data sets.

- Researchers and Data Scientists can make edits to manifests

System Requirements

- Database:
 - For this project there is a need for a database in which we will use MySQL or SQL
 - Database security is a must, through hash functions and encryption we will protect the data that is uploaded to our system.
- Servers:
 - This product will need to have servers for hosting, and possibly in multiple locations.
 - Backup servers to account for potential failure. Losing all of this data and cross-research could be detrimental.
- Web Service:
 - In this project a web service will be needed, and for this we will use Microsoft Azure, or Amazon Web Service.
- Internet Connection:
 - For a user to use this product, they will need to be connected to the internet (wireless or ethernet).
- Deployment:
 - For building and deployment we will use Microsoft Azure or Amazon Web Service.
- Operating Systems:
 - We we deploy for MAC OS, and Windows. Maybe be able to go mobile?

Use Cases

For each of the use cases detailed below, one or more of the expected users will take part in an activity relative to the OCDX process.

Browse Manifest

When the user searches by keyword, relevant manifests and corresponding SNC files are displayed with brief descriptions on what the data and files hold. This use case will activate whenever a user chooses to search or browse the manifest available in the website.

Functional Requirements and Non-Functional Constraints

- "Date Uploaded" will be differentiating factors for multiple sets of SNC files for the same data set.
- A search entry provided by the user if the user decides to search for a manifest.

- A search on Manifest returns a collection of appropriate results, in user's choice of list or table view.
- Search results can be filtered or sorted alphabetically or by length, keyword etc.

Non-Functional Constraints

- Minimum of one manifest available for browsing.
- Search by Manifest returns a maximum of ten results per page.

Technical Requirements

- Providing user a list of manifest available.

Primary Actors

- Researchers
- Students
- Data Scientists
- System Admins

Pre Conditions

- User has provided a search entry or clicked the button to show and browse all manifest.

Main Success Scenario

- A list of manifest is listed on the website for the user and the corresponding datasets and manifest pertaining to the manifest shown.

Failed End Condition

- No list of manifest is available for the user to browse.

Trigger

- User clicks on browse manifest or provided a search entry to the search bar and hit search.

Dependent Use Cases

- Search Manifest

Upload Data Set

A researcher with a data set will come to our site to contribute their data set. They will also have the option to upload any SNC files (scripts, notebooks, and config files).

Functional Requirements

- Form to upload/link to dataset
- Form to upload scripts (if provided by the user)
- Docker config file (if provided) [could be generated with use case "Generate Config Settings"]

- There is going to be levels of authorizations; admins, researchers/data scientist, and students.
- For admins, there should be special administration functionality, such as who can upload/link data sets, security, maintenance, and monitoring of what is being uploaded, ect.
- An error message needs to be displayed if not uploaded correctly.
- A success message needs to be displayed if uploaded correctly.

Non-Functional Constraints

- There needs to be a maximum file size so that a user cannot upload a big file.
- Uploading data set needs to have acceptable performance(good run time), so that it doesn't take seven years to upload a set.
- Placing to upload/link to data set needs to be usable.
- Data sets that are uploaded need to have data integrity.
- Uploads need to have some form security so that data sets do not get corrupted during the upload process.
- There needs to be an error message if upload time runs out.

Technical Requirements

- Back end version control for the scripts
- Disk space for storage of data (possibly large data sets)
- Working, public web server

Actors

- Researchers
- System Admin
- Students
- Data scientists

Pre Conditions

- One primary actor wants to upload and share data.

Main Success Scenario

- Datasets and SNC files have been successfully contributed to the site. A successful outcome would be when a data set is uploaded completely and correctly.

Failed End Condition

- A researcher with a data set will come to our site to contribute their data set. They will also have the option to upload any SNC files (scripts, notebooks, and config files). A non successful outcome would be if during the data set upload, something got corrupted, or failed to upload during the upload process.

Triggers

- User clicks "Submit Data Set Button on the OCDX.IO website."

Contribute to Existing Dataset

A user wishes to add SNC files for a dataset already uploaded to the server. The user will provide a link to the dataset that will be found on the server using its SHA1 code.

Functional Requirements

- Place to upload JSON file
- Docker config files (if provided) or link to "Generate Config Settings".
- Files are scanned prior to upload to avoid file corruption or viruses.

Non-functional Requirements

- Ability to upload up to 3 JSON files at one time.
- Database security ensured by detecting and removing potential SQL injections.
- Limit on file size for large dataset uploads.

Technical Requirements

- Back end version control for scripts
- Working, public web server.

Primary Actors

- Researchers
- Students
- Data scientists
- System admins

Pre Conditions

- User wished to upload SNC files.
- A request was sent from the save

Main Success Scenario

- Dataset has been found on server and SNC files have been successfully uploaded to site and connected to the dataset's manifest.

Failed End Condition

- User is unable to upload SNC files or dataset not found on server.

Trigger

- User clicks "Contribute to Existing Database"
- User saves

Dependent Use Case

- Save

Download Info

When a user has found a dataset or SNC files they are interested in, they can opt to download it onto their local machine. They can choose to download just the dataset or the SNC files with the dataset.

Functional Requirements and Non-Functional Constraints

- A button on the page that the user clicks to prompt the system to start the download process.
- A form or pop-up where the user can select whether they want to download only the dataset or the SNC files with the dataset.
- Some type of form/progress bar that keeps the user updated on the status of their download.

Non-Functional Constraints

- If the download is proceeding exceptionally slower than expected, the user should be prompted with an option to retry or cancel the download.

Technical Requirements

- Copying files onto user's local machine.

Primary Actors

- Researchers
- Students
- Data Scientists
- System Admins

Pre Conditions

- A user has browsed for and selected a dataset.

Main Success Scenario

- A copy of the files is on the user's local machine.

Failed End Condition

- The download fails and the user does not have a copy of the files.
 - The user should be notified when this occurs.

Trigger

- User clicks "Download" on manifest's browsing page.

Dependent Use Cases

- Search Manifest
- Browse Manifest

Generate Upload Manifest

After a user has uploaded the dataset and any SNC (Scripts, Notes & Configuration) files, they will have one of two options. If a complete OCDX Manifest is already available to the user, the file can be uploaded directly. Otherwise, the user will manually fill out a manifest form with the necessary specifications. Once they do this, they will also have the ability to search for the manifest on the web app. Information of the manifest should be described in detail, from basic author information to tags. In both cases, the manifest must be complete before the dataset and SNC files are saved.

Functional Requirements

- For this function to work, the users must have already uploaded their datasets and SNC files into the website. From there, the requirements are as follows:
- Buttons to "Generate Manifest" and "Upload Manifest"
- Include input fields for all manifest specifications for "Generate Manifest"
- Dataset and any SNC files has been contributed.

Non-functional Constraints

- Maximum file size for uploaded Manifest.
- Only one Manifest is generated or uploaded per dataset.

Technical Requirements

- Database for storing manifests and populating entries in the table to catalog events.

Primary Actors

- Researchers
- Students
- Data scientists
- System admins

Pre Conditions

- A primary actor has shared data and any SNC files.

Main Success Scenario

- Manifest, data, and any SNC files are uploaded to server and are saved and searchable.

Failed End Condition

- User does not have a complete manifest.

Trigger

- User clicks "Create Manifest" or "Upload Manifest" button on site after submitting dataset.

Dependent Use Cases

- Upload Data Set

Save

Users will be able to save their work, either created, edited, updated, or manipulated SNC (Scripts, Notes & Configuration) files, before closing the program. If they choose to save it, the user will be directed to "Contribute to Existing Dataset" where they will be able to choose where to save in the database. If they click Don't Save, none of their work will be saved.

Functional Requirements

- Button to "Save": save by clicking the save button in the program
 - The user will be directed to "Contribute to Existing Dataset" page
- Popup to "Save" if user tries to exit without saving their changes
 - If user decides to click the "Save" button on the popup, they directed to the Contribute to Existing Dataset like the regular save
 - If user choose "Don't Save" exit the program normally
 - If user click cancel, it canceling the popup and the exit action

Non-functional Constraints

- If the popup does not get any response for 10 minutes, the program will cancel the exit action.

Technical Requirements

- Redirected to "Contribute to Existing Dataset" to save the changes that made by the users

Actors

- Researchers
- Students
- Data Scientists
- System Admins

Pre Conditions

- User has opened data or SNC files in the program
- User has made any changes to original SNC files or has created new SNC files

Main Success Scenario

- User is able to save their additions and is directed to create a new project linked to an existing dataset.

Failed End Condition

- The user is not able to save their additions or is not directed to "Contribute to Existing Dataset" when wanting to save.

Trigger

- Click "Save" button
- Exit the program without saving

Dependent Use Cases

- Search Manifest
- Browse Manifest

Search on Manifest

A user will choose to search the site's database for datasets and corresponding SNC files using keywords. The manifest will be searched for these keywords.

Functional Requirements

- Input text field for keywords
- Users have input a search entry or keyword into the search input text field.
- Users have input valid search entry keyword.
- Search keywords on creator, date, content etc.

Non-Functional Constraints

- Search by one keyword at a time.
- Search results display ten per page.

Technical Requirements

- Program to search and select manifest for keywords.

Primary Actors

- Researchers
- Students
- Data scientists
- System admins

Pre Conditions

- A user wants to search the site for data.
- User is logged in to the web application.

Main Success Scenario

- Manifests on the server have been searched for the keyword(s) and SNC files and datasets connected to the manifests with the keywords are displayed to the user.

Failed End Condition

- No manifests found with given keywords.

Trigger

- User clicks "Search Manifests"

Dependent Use Cases

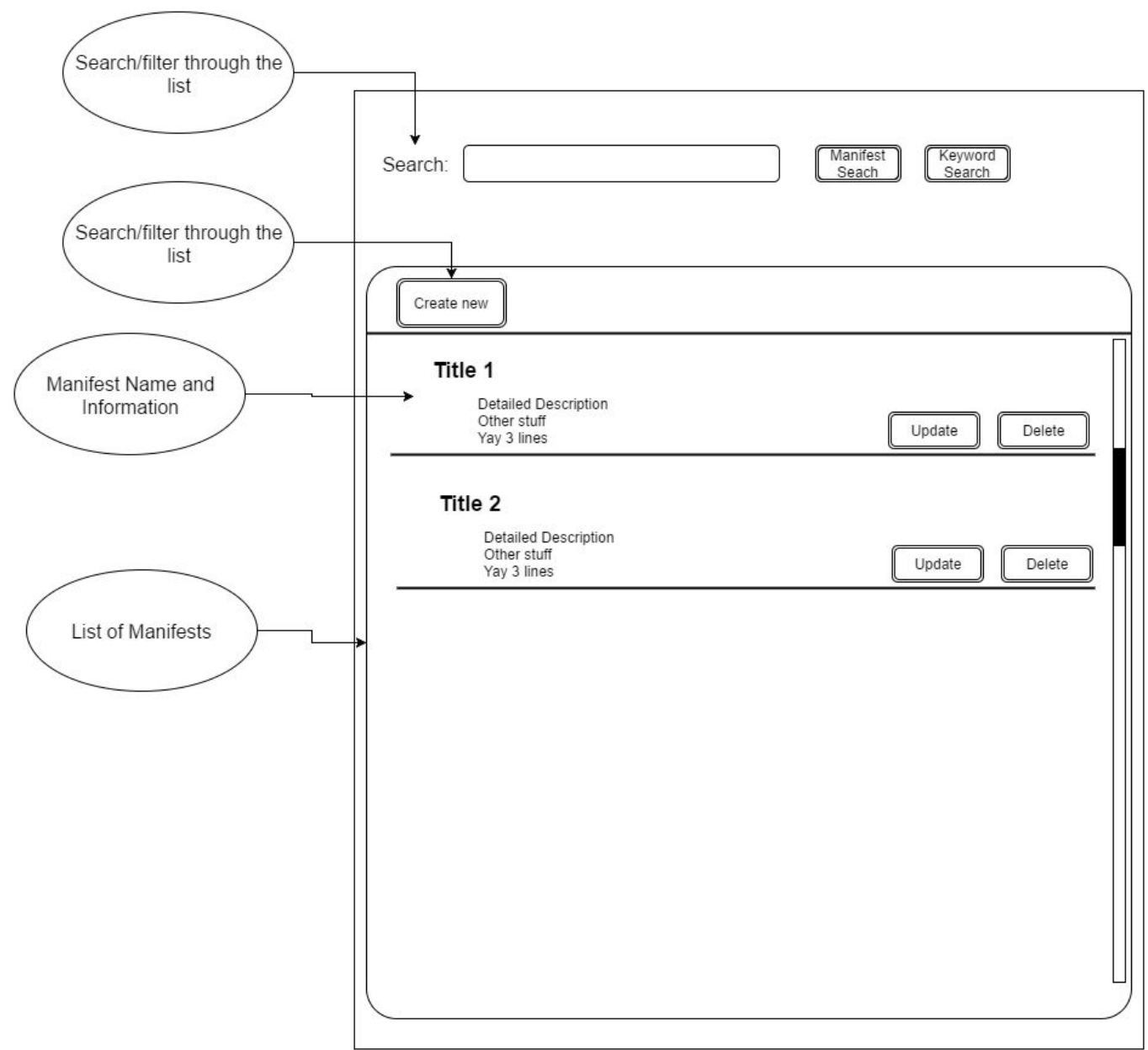
- Upload Data Set
- Generate Upload Manifest
- Contribute to Existing Dataset

System Design

Screen Views

Browse Manifest

Summary: This page is to locate the manifest using a search function or by brute force and scrolling through the entire list. This will link or do all of the functionality of this page.



Manifest Search: Searches everything. This includes all metadata fields and the values in the data

Keyword Search: Searches the metadata. So this includes title, id, creator, date created.

Create New: This creates a new manifest and navigates you to the create/upload manifest

Update: This navigates you to the view/edit page so that you can make changes with the correct permissions

Delete: This will delete the manifest with the correct permission

Upload/Create Manifest

Summary: To upload, the user will choose a file and that will update the details section. If they want to create, all of the fields will be editable and the user can write in the information they want.

Upload a file and it will auto-generate details

File Picker Choose File

Details

Manifest Name: Manifest 1
Author: Some Guy
Date: Today

```
{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to create markup languages
such
as DocBook.",
            "GlossSeeAlso": [
              "GML",
              "XML"
            ]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}
```

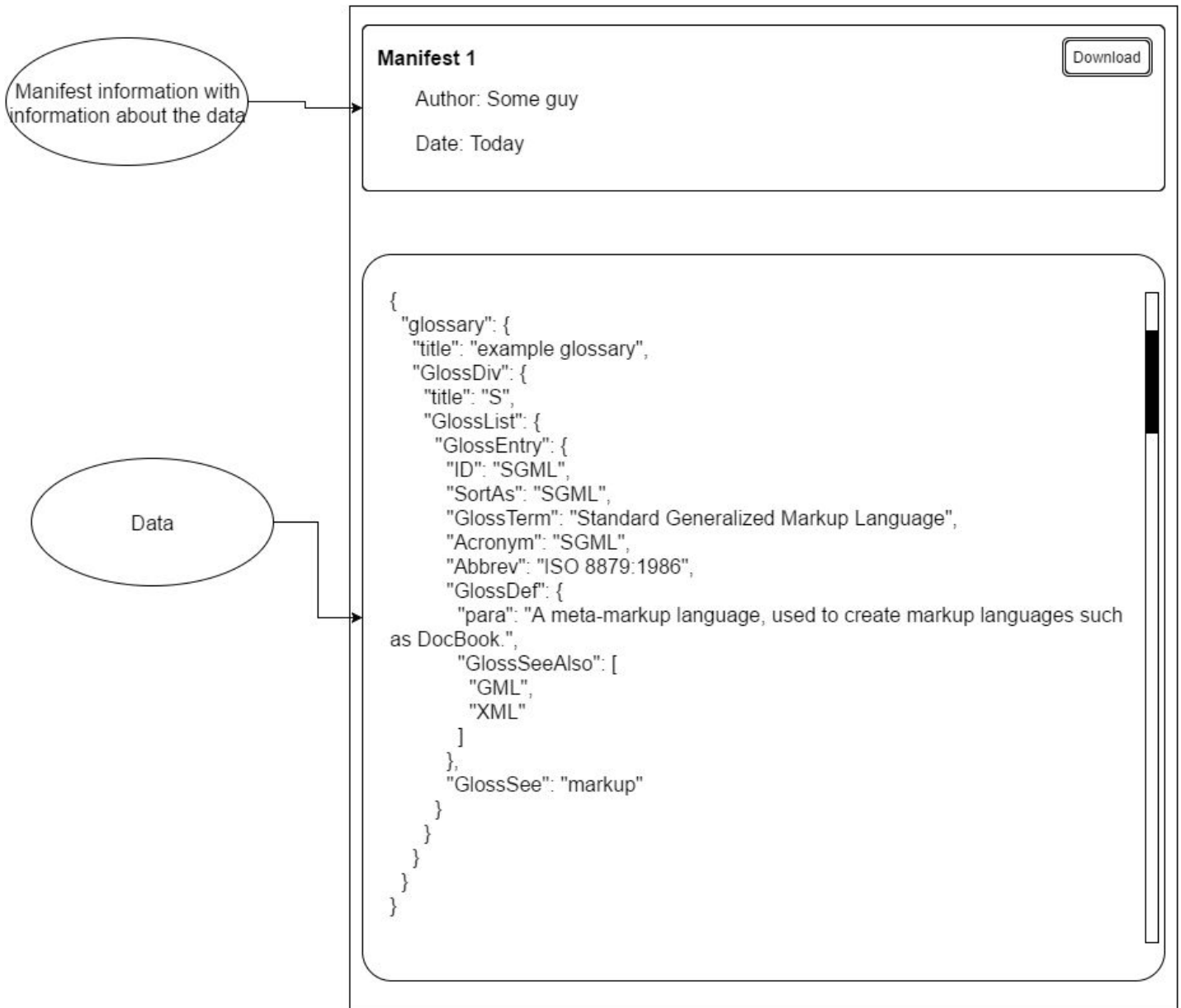
This field will auto generate, but still be editable

The user will enter this data or have it auto-loaded by the file picker

Choose file: This will open up a file picker with the restriction to json files

View/Edit Manifest

Summary: This will display all of the information that is in a specific manifest. With the right permissions, the user can edit this file.



Download: This will download the json file onto your computer.

Update: This will send a request to the server with the changes.

Class List

- Users
 - Users()
 - This function will have a list of users. This list of users will be grabbed from a database. This function can also grab data and details about the user, such as if the user is a researcher, student, data scientist, or system admin.
 - userLevel()
 - There could another function under class user which handles different levels of administration privileges.
 - login()
 - There will also be a function that will check user login information, and handle the situation where a user needs to register an account.
- Manifest upload
 - uploadManifest()
 - There will be a function just for manifest upload, which will include an upload button that will be an onClick event. This function will grab manifest data.
 - displayDataUpload()
 - After fetching data, there is going to be another function that displays the data that was just uploaded. Also, this is where the functionality of displaying and error or success message.
- Manifest download
 - downloadManifest()
 - This function will use the database, and download new data to the database. There will be a button that will be onClick event.
 - displayDataDownload()
 - After downloading data, there is going to be another function that displays the data just downloaded. Also, this is where the functionality of displaying and error or success message.
- Manifest search
 - search()
 - This function will use the database. There will be a text field that allows a user to search through the site's database for datasets and corresponding SNC files using keywords. The manifest will be searched for these keywords. The function will take these keywords or phrases and respond with relatable/significant data.
 - displaySearch()
 - This function simply just displays searched keywords/phrases.

- Contribute to existing
 - contribToExisting()
 - A user wishes to add SNC files for a dataset already uploaded to the server. The user will provide a link to the dataset that will be found on the server using its SHA1 code. This code will require talking to the database of files. This function will also talk to the users part of the database. This is because, certain users will only have certain rights to edit certain material. For example, researchers can upload and edit their own work, but can't edit another researcher's work. Researches may have the option to request edits on other researchers work. Students would not have any editing powers, and system admins will have all editing powers.

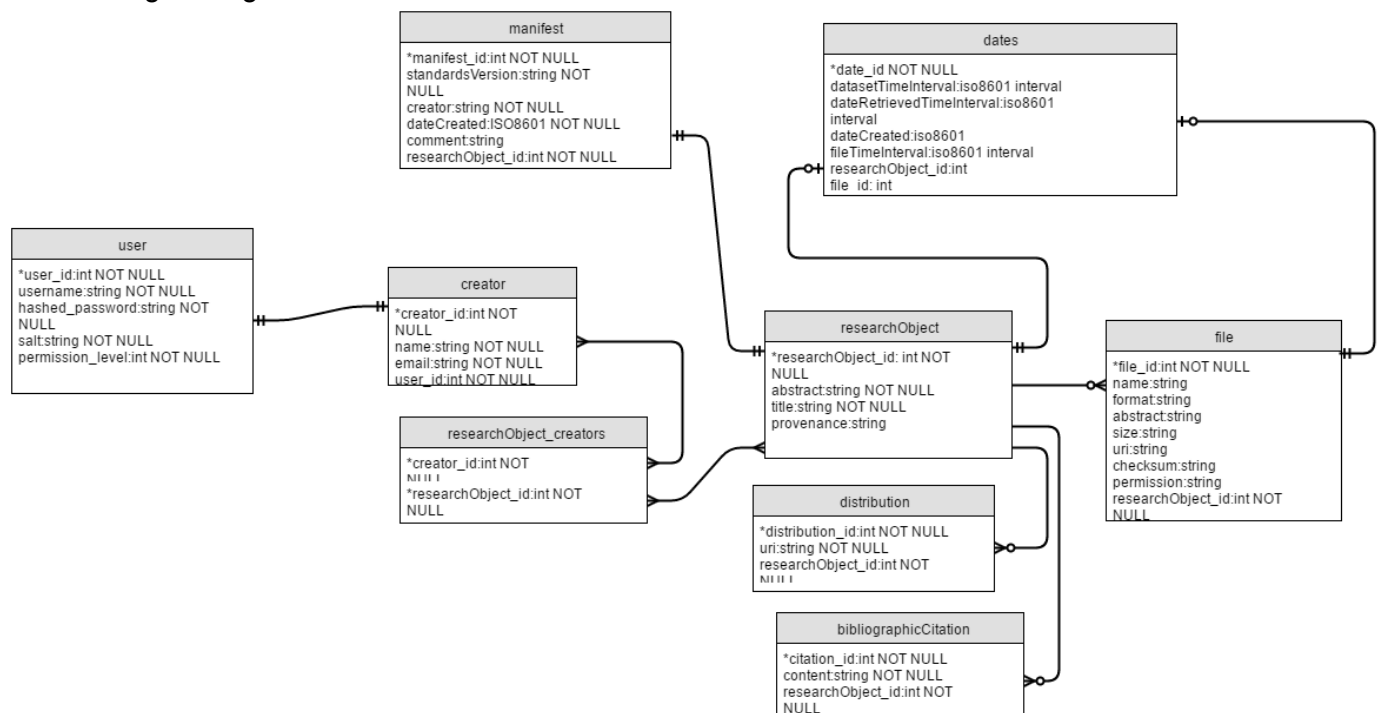
Table List

- Manifest
 - abstract:string, title:string, arovenance:string, bibliogrpahicCiations:string
- Creator
 - name:string, email:string
- Date
 - datasetTimeInterval:time
 - dateRetrievedTimeInterval:time
 - dateCreated:datetime
- Distribution
 - Uri:string
 - Comment:string
- File
 - Name:string
 - Format:string
 - Date->fileTimeInterval:time
 - Date->dateRetrievedInterval:time
 - Date->dateCreated:datetime
 - Abstract:string
 - Size:int
 - Uri:string
 - Checksum:string
 - Permission:string
- Manifest Relations: Every manifest has one or more than one creator, one date, one distribution, and one or more than one file.

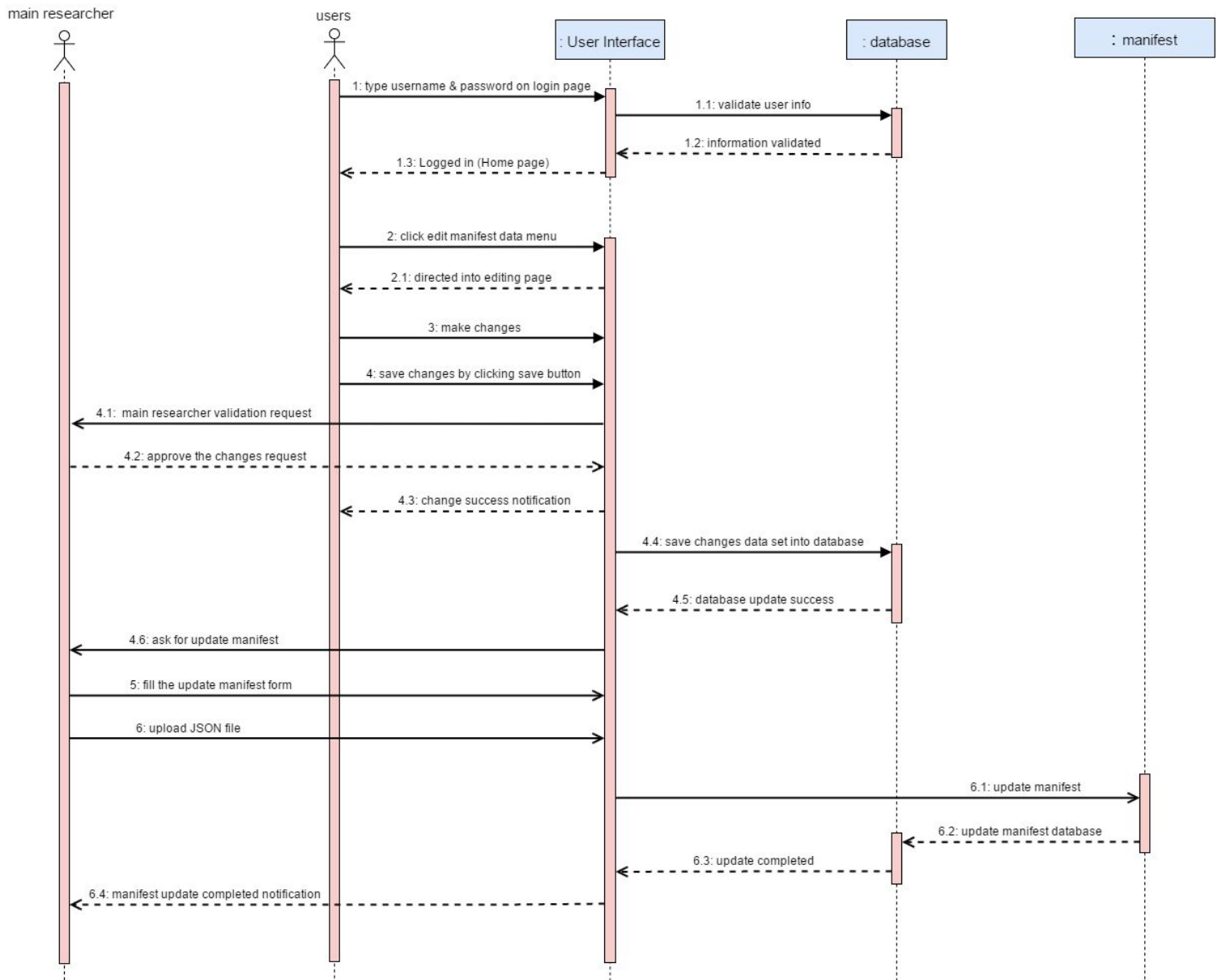
- User
 - userID:string, password:string, permissionLevel:int
- Researcher
 - Manifest:manifest
 - PermissionLevel:int = 2 (level two permission - edit and upload access)
 - Relation: Researcher is a User (one to one), researcher will have one or more than one manifests attributed to him/her.
- Student
 - permissionLevel:int = 1 (read and download access only)
 - Relation: Student is a User (one to one)
- SysAdmin
 - permissionLevel:int = 3 (full access to front and back-end)
 - Relation: SysAdmin is a User(one to one)

ERD

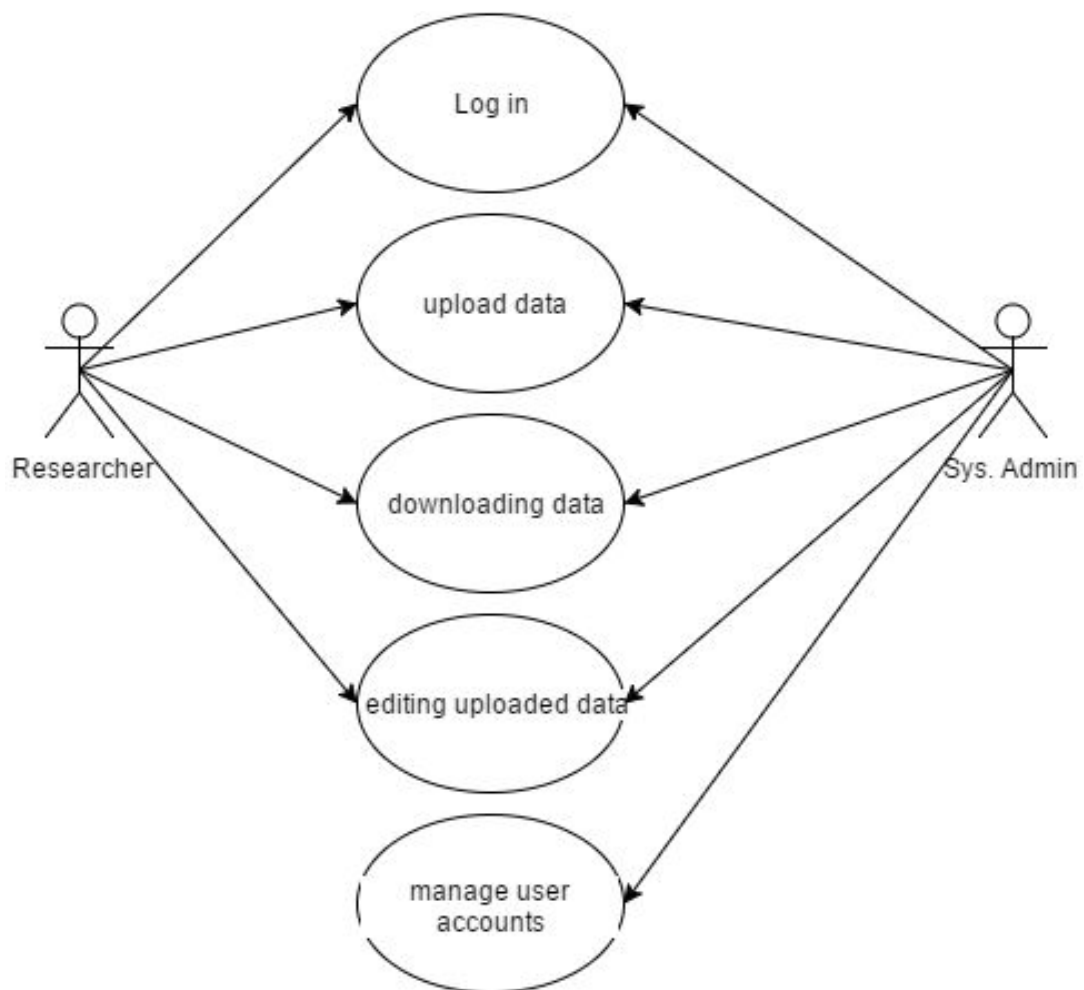
Larger image available [here](#)



Sequence Diagram: Edit a Dataset



Use Case Diagram: Working with Datasets



Sprint 1

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

- * Unit Testing - Kate, Assia
- * Regression Testing - Kate, Assia
- * Integration Testing - Kate, Assia
- * User Acceptance Testing - Kate, Assia

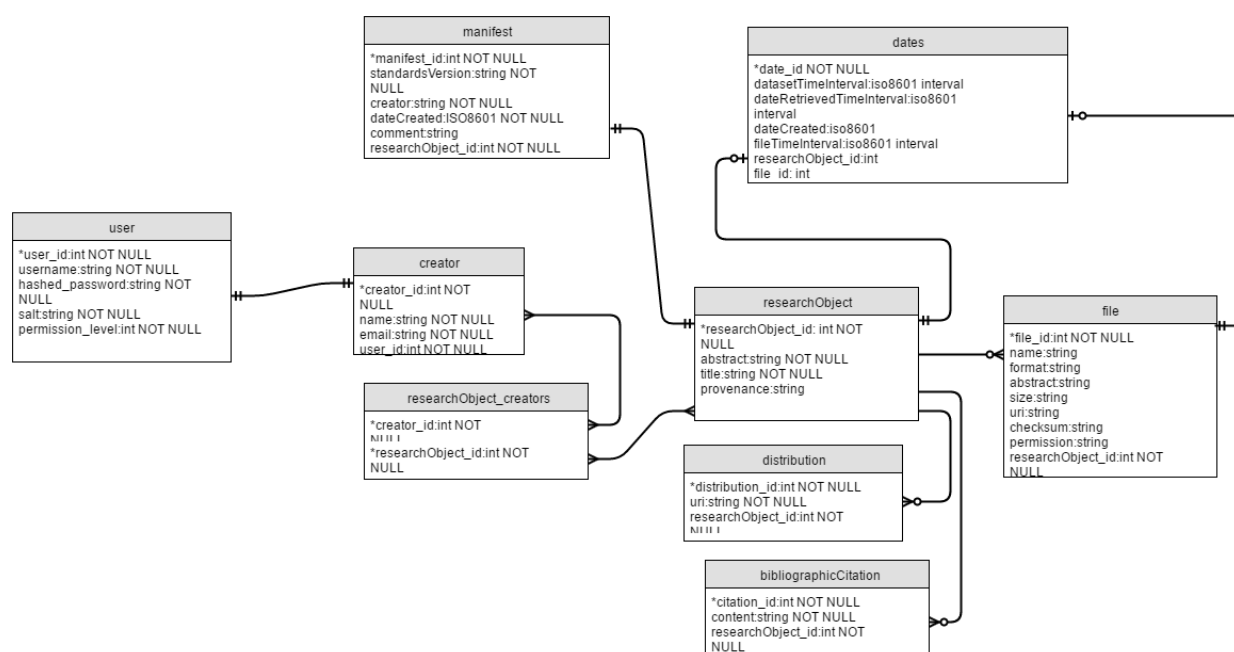
Additional Documentation for Individual Tasks

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, and it can also be found below. 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.

Larger viewable link:

<https://drive.google.com/file/d/0Bzc5cB9bdfSUWWUzNURiTEFPYW8/view?usp=sharing>



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 Documentation

All test cases will provide a failed or success message upon operation

Unit test:

Insertion:

- Test case 1:
 - Check to see if user inserted correct type of data (date created, comments, title, etc.).
 - Can't put a string type into a int field (title of research into a dateCreated).
 - User is prompted to re-enter data if incorrect data is entered, before any processing occurs.
- Test case 2:
 - Check to see if user filled out all required data (no blank input fields when data is required).
- Test case 3:

- Check to see if user has correct permissions (only researchers and system admins can upload new data into the system).
- Test case 4:
 - Check to see if data was successfully inserted (backend testing).

Deletion

- Test case 1:
 - Check to see if user has correct permissions. Only the researcher of the project can delete items from their own project.
- Test case 2:
 - Check to see if data was deleted correctly and fully in database (backend testing).

Updating

- Test case 1:
 - Check to see if user updated correct type of data(date created, comments, title, etc.).
 - Can't put a string type into a int field (title of research into a dateCreated).
- Test case 2:
 - Check to see if user has correct permission(researcher can update).
- Test case 3:
 - Check if data was deleted(backend testing).

Login/logout

- We will have code the checks for a user credentials(username/password)
 - Present a success message on correct validation
 - User has to be in database.
- Check to see is user is in database
- User upon first time to website will need to register themselves
 - If user tries to login without being registered, present error message.
- Can assign themselves as an admin of the project, and can invite students/data scientist

User privilege (There will be several levels of privileges)

- System Admin-level 1 in database
 - has all privileges; reading, writing, updating, deleting, inserting, user account management, manipulating UI elements, download/upload
 - Test privileges whenever this admin tries to use these operations.
- Researcher-level 2 in database
 - has all privileges; reading, deleting, updating, inserting, writing, upload/download
 - Test privileges whenever this researcher tries to use these operations.
- Student-level 3 in database
 - has all privileges; reading, download
 - Test privileges whenever this student tries to use these operations.
- Data scientist-4 in database

- has all privileges; reading, download
- Test privileges whenever this data scientist tries to use these operations.

User Acceptance testing

- Give product to Goggin's and TA's, and have them determine if the product meets all the specific requirements (basically, is it acceptable).
 - Specific requirements:
 - Upload/download
 - Contribute to existing
 - Save
 - Browse/search
- Have them provide feedback.

Integration Testing

- **Integration testing** is the phase in software **testing** in which individual software modules are combined and tested as a group. It occurs after unit **testing** and before validation **testing**.
- We will using the third approach, sometimes referred to as the umbrella approach, requires testing along functional data and control-flow paths. First, the inputs for functions are integrated in the bottom-up pattern discussed above. The outputs for each function are then integrated in the top-down manner. The primary advantage of this approach is the degree of support for early release of limited functionality. It also helps minimize the need for stubs and drivers. The potential weaknesses of this approach are significant, however, in that it can be less systematic than the other two approaches, leading to the need for more regression testing.
- The group will present their code and explain the changes they made.
- We will have code reviews if necessary.
- The group will then discuss if the project is ready for integration.
- Once the group agrees everything is good to go, the project leader will merge the project.
- If any conflicts happen here, they should be fixed.

Regression Testing

- **Regression testing** is a type of software **testing** that verifies that software previously developed and tested still performs correctly even after it was changed or interfaced with other software. Changes may include software enhancements, patches, configuration changes, etc.
- In computer security, a **sandbox** is a security mechanism for separating running programs. It is often used to execute untested or untrusted programs or code, possibly from unverified or untrusted third parties, suppliers, users or websites, without risking harm to the host machine or operating system.
- Regression testing will include integration testing (explained above), along with unit tests.

- Regression testing will be used so that is a bug in the application, we can easily fix it.
- A possible product we can use is called MircoFocus: microfocus.com/product
- To prevent conflicts, we developed a sandbox, so all testing occurs here.
 - Our testing occurs on a site which is a sub domain of the main site.
 - Once completed testing, we will integrate it with the main site.
- Our main site is what clients see, the sub domain site is what we use and see.

Verification and validation

- Regression Testing – **Verification**
- Integration Testing – **Verification**
- User Acceptance – **Verification**
- Unit Test
- Updating – **Validation and Verification**
- Insertion – **Validation and Verification**
- Deleting – **Validation**
- Login/logout-**Verification**
- User privilege (There will be several levels of privileges) - **Validation**

Future testing:

- Upload/download
- Browsing/searching

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 login 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

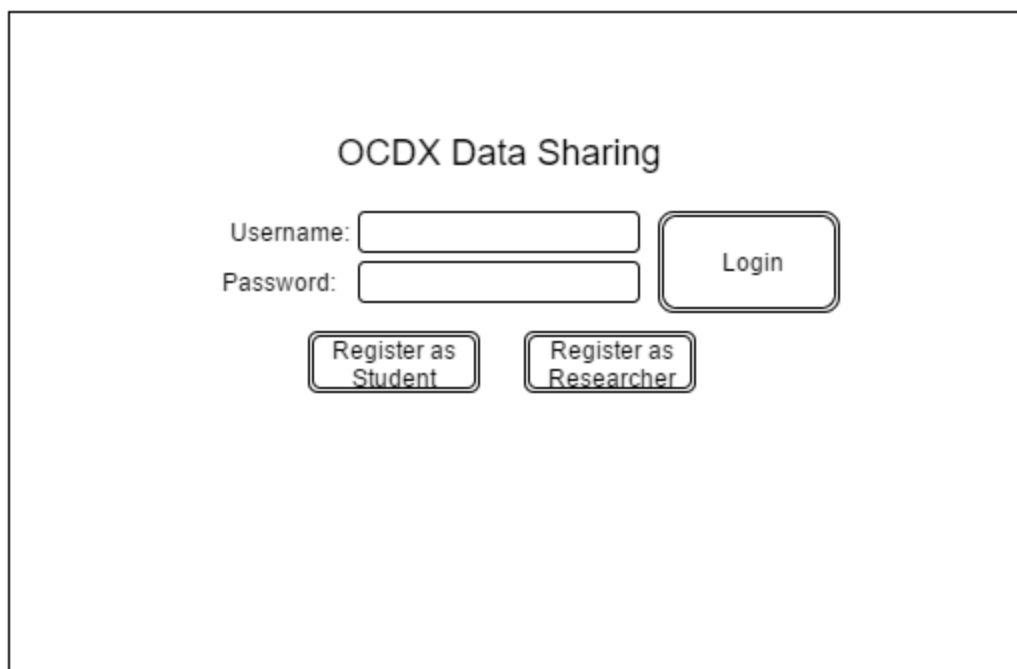
The complete system was simplified down into 3 pages. The search page, the view/edit page, and the create/update page. The slash means that they are multi-purpose. In the search page, the user can search through the list using a manifest search and a keyword search. The manifest search allows for you 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 see everything in the manifest and if you have permission you can edit it.

Screen views will be found on the next pages.

Screen Views

Login Screen

Summary: The first screen a user sees when visiting the page. They may either enter in their credentials and log in, or redirect their page to register.



The image shows a login screen for 'OCDX Data Sharing'. It features a title 'OCDX Data Sharing' at the top. Below the title, there are two input fields: 'Username:' and 'Password:'. To the right of the 'Username:' field is a 'Login' button. Below the 'Password:' field are two buttons: 'Register as Student' and 'Register as Researcher'.

Button: Login: Sends username and password info to the server via php POST. If user credentials are correct, they are taken to the Browse Manifest/Home page. If the username does not exist or the password is incorrect, appropriate feedback is displayed on the page.

Button: Register as Student: Sends user to the Register page, where they may register a new account with read and download privileges on the web server.

Button: Register as Researcher: Sends user to the Register page, where they may register a new account with read, write, edit, and download privileges provided they can provide a correct access code.

Register

Summary: Both register buttons on the login screen redirect to this page, where the user can create a new account. If the user wishes to register as a researcher, an extra form for 'access code' is required. Those who wish to create a research account must apply for one of these codes, as not just anyone should be able to submit their 'research'.

The diagram shows a registration form titled "New User Account". It contains the following fields and elements:

- First Name**: Text input field.
- Last Name**: Text input field.
- Email Address**: Text input field.
- Username**: Text input field.
- Password**: Text input field with a note: "password must be 6-24 characters in length".
- Access Code**: Text input field, indicated by an arrow from a note on the left.
- Create Account**: Button.
- Cancel**: Button.

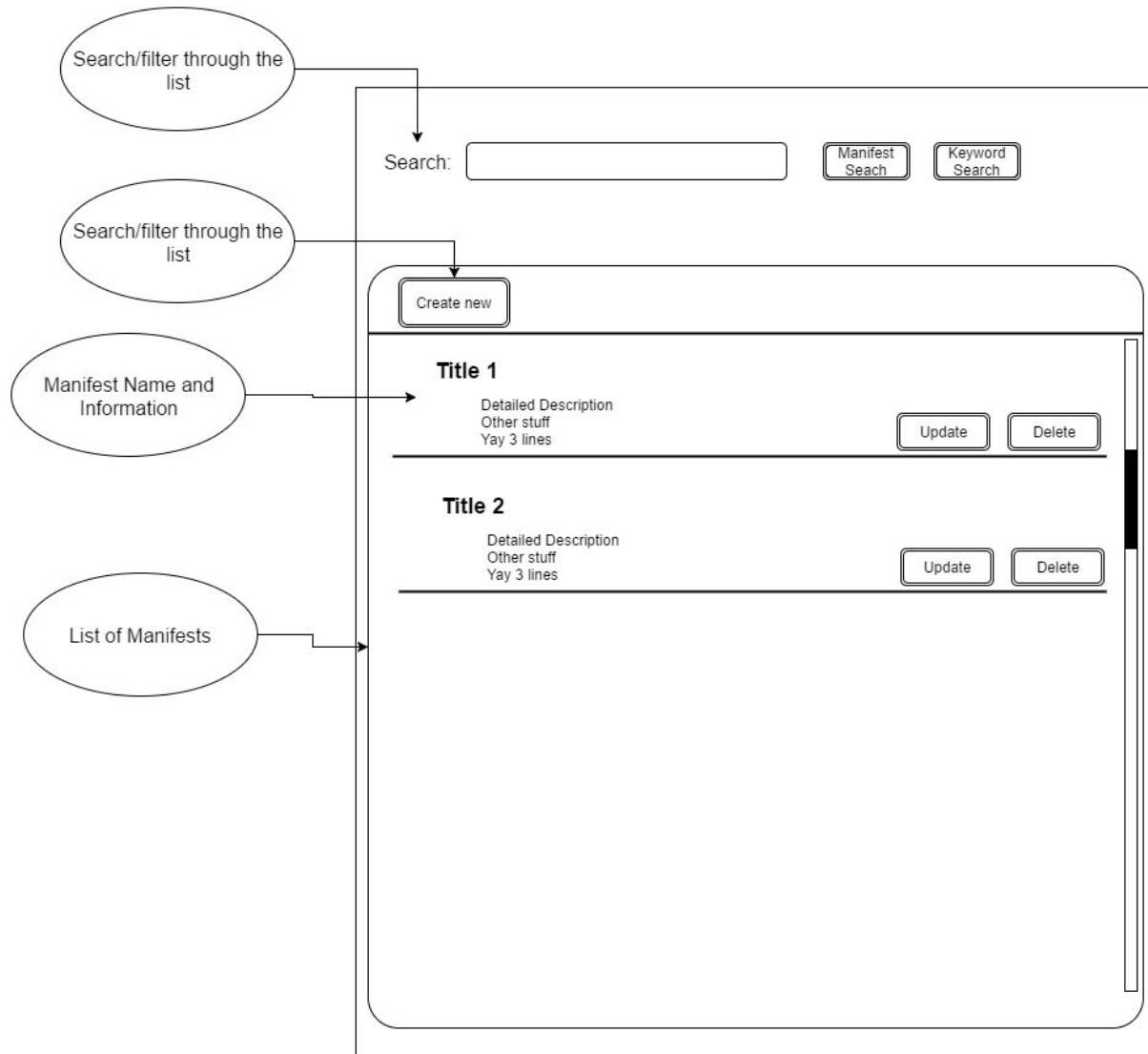
A note on the left side of the form states: "'Access Code' form only visible to those who wish to register as a Researcher. In this case, an access code is required to register, as research accounts are invite-only". An arrow points from this note to the "Access Code" input field.

Button: Create Account: Attempts to send the user's input into the user database as a new collection item. If any elements of the user's input are invalid, the data is not sent and appropriate feedback is displayed by the offending form(s). If the input is valid, the data is added as a new collection item and the user is redirected to the login page.

Button: Cancel: Sends the user back to the login screen without addressing any form input or changing any data.

Browse Manifest/Home page

Summary: This page is to locate the manifest using a search function or by brute force and



scrolling through the entire list. This will link or do all of the functionality of this page.

Manifest Search: Searches everything. This includes all metadata fields and the values in the data

Keyword Search: Searches the metadata. So this includes title, id, creator, date created.

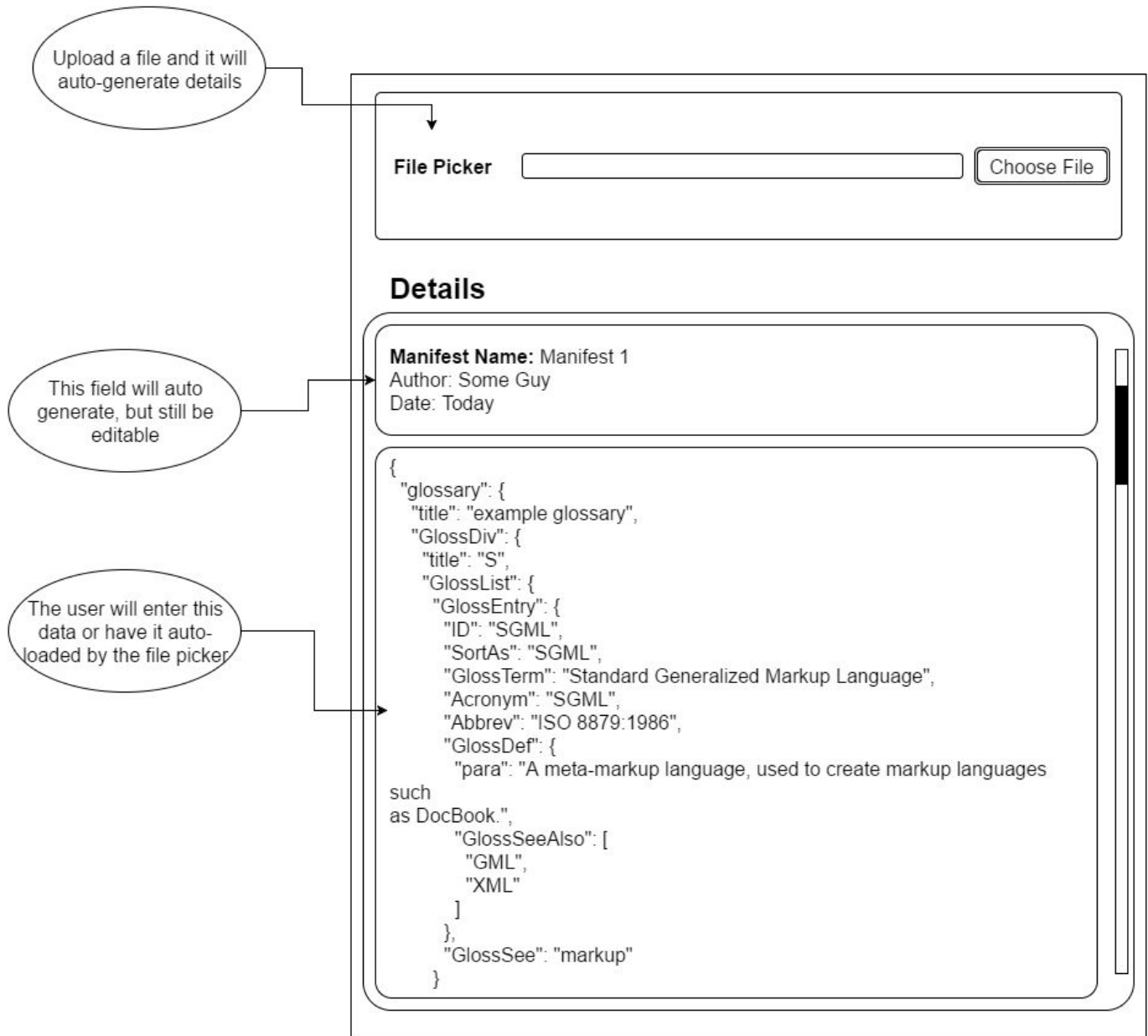
Create New: This creates a new manifest and navigates you to the create/upload manifest

Update: This navigates you to the view/edit page so that you can make changes with the correct permissions

Delete: This will delete the manifest with the correct permission

Upload/Create Manifest

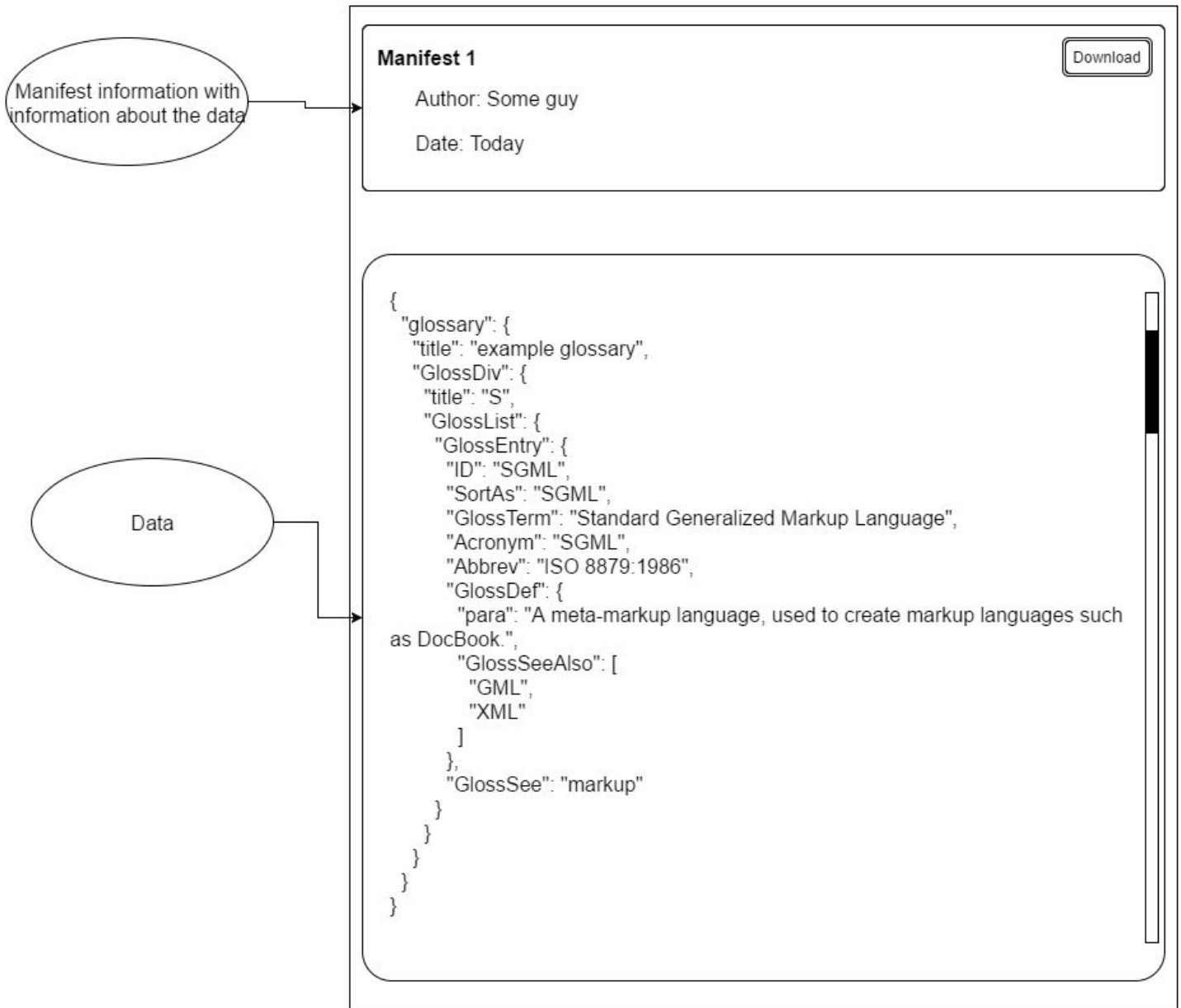
Summary: To upload, the user will choose a file and that will update the details section. If they want to create, all of the fields will be editable and the user can write in the information they want.



Choose file: This will open up a file picker with the restriction to json files

View/Edit Manifest

Summary: This will display all of the information that is in a specific manifest. With the right permissions, the user can edit this file.



Download: This will download the json file onto your computer.

Update: This will send a request to the server with the changes.

Sprint 2

Group Meetings

Meeting 1:

Date: 11/6/2016

Time: 2:00pm - 6:00pm

Attendance: Andy, Wei Xian, Dewi, Kate

Discussion:

Meeting 2:

Date: 11/10/2016

Time: 2:00pm - 5pm

Attendance: Andy, Wei Xian, Dewi, Kate, Alex

Discussion:

Sprint 2 Tasks List

- Sprint Documentation
 - Incorporate feedback from Sprint 1 into an improved document for resubmission
 - Add ERD, Screen Views to Sp1 documentation, develop the documentation fully so that readers do not have to search around the repository or other documents to find our designs
 - Add screen views and UI documentation for Login/Register/Home pages
 - Fix our branching situation in GitHub
 - Improve Testing documentation based on feedback
 - Use feedback as a benchmark for Sprint 2 documentation, to ensure a better product this week
 - Incorporate feedback from Requirements & Design Documentation
 - Indicate individual assignments in document

- Database
 - Write all insert, update, and delete queries for our application
 - Create New Manifest
 - Edit Manifest Contents
 - Delete Manifest
 - Edit Data (Creator)
 - Request Edit Data (non-author creator)
 - Search by Title/Author/Keyword
 - Add new user (registration)
 - Upload queries as separate files into a DML directory in our repo
 - Create dummy collection for Users
- User Interface
 - Began UI Elements
 - Implement UI functionality so that our buttons and forms are able to communicate with the database
 - Stub-calls for all interactive UI elements
- Other
 - Management of users/roles
 - User account dummy collection
 - Prepare UI to interface with users of various access privileges

Task Assignments

- Andy
 - Address feedback for Requirements & Design documentation and make appropriate edits
 - Address feedback for Sprint 1 documentation: Add diagrams and links that were missing from final doc
 - Describe management of roles & user accounts within our system
- Wei Xian
 - Changed branch name due to typo error at previous branch.
 - Perform testing on server to ensure that php is working with mongoDB modules.

- Created test scripts to perform db collection creations, inserting data, and finding data.
 - Php scripts example taken from tutorialspoint.com.
- Copied initial html files to test site on the server to check if UI works.
- Continue working on automatic deployment from github utilizing webhooks PUSH.
- Helped Dewi in learning mongoDB queries and designing queries to work with required JSON format.
- Kate
 - Address feedback for Sprint 1 documentation: Expand and improve testing documentation based on feedback
- Dewi
 - Together with Wei Xian created CRUD queries for database implementation
 - Implement all queries and ensure everything works fine
 - Database implementation documentation in Sprint 2
 - Upload manifest.json and queries.txt in DML GitHub repo
- Assia
 - Address feedback for Sprint 1 documentation: Expand and improve testing documentation based on feedback
- Alex
 - Begin developing UI elements (low-functionality)

Database - Query Implementation

In this database exploration, we are working with JSONArray data. The array is called manifests and for now we have 2 indexes. Below is the complete JSON file with dummy data that we dumped into our MongoDB.

JSON

```
{
  "manifests": [
    {
      "standardsVersion": "v0.0.1",
      "id": "1111201601",
      "creator": "Sean",
      "dateCreated": "11/11/2016",
      "researchObject": {
        "title": "Malware Security",
        "creators": [
```

```

{
  "name": "David Saws",
  "email": "saws.david@missouri.edu"
},
{
  "name": "Vincent Bros",
  "email": "bros.vincent@missouri.edu"
}
],
"dates": {
  "dateCreated": "2016-30-10"
},
"provenance": {
  "narrative": "full details in paper"
},
"bibliographicCitations": [
  "Susan, B. M. (2009). Malware and Virus Analysis. Journal of Studies in International Technology, 7(4), 379-403. doi:10.1177/102831543532257120",
  "Tricia, K. A. (2013). Information Security System. Journal of Research in Computer Science, 5, 131-126. doi:10.1177/147524090606558954325342"
],
"distributions": [
  {
    "uri": "https://datahub.io/dataset/malware_analysis",
    "comment": "Folder with 15 findings data"
  }
],
"files": [
  {
    "name": "2015ResearchDataCollection.json.txt",
    "format": "json",
    "abstract": "Contains data collection for the research analysis",
    "size": "78 K",
    "uri": "https://datahub.io/dataset/malware_analysis/2015ResearchDataCollection.json",
    "permissions": "Open Database License 2.0",
    "dates": {
      "dateCreated": "2015-10-10"
    }
  }
]

```

```

    }
  },
  {
    "name": "2015ResearchAnalysis.tsv.txt",
    "format": "tsv",
    "abstract": "Contains the result of research analysis",
    "size": "256 K",

"uri": "https://datahub.io/dataset/malware_analysis/2015ResearchAnalysis.json",
    "permissions": "Open Database License 1.0",
    "dates":
      {
        "dateCreated": "2015-10-11"
      }
  }
]
}
},
{
  "standardsVersion": "v0.1.2",
  "id": "1112201601",
  "creator": "Goggins",
  "dateCreated": "11/12/2016",
  "researchObject": {
    "title": "Big Data Security",
    "creators": [
      {
        "name": "Collin Tankard",
        "email": "tankard.collin@missouri.edu"
      },
      {
        "name": "Jillian William",
        "email": "william.jillian@missouri.edu"
      }
    ],
    "dates": {
      "dateCreated": "2016-10-10"
    },
    "provenance": {

```



```

    "narrative":"full details in paper"
  },
  "bibliographicCitations":[
    "Trice, A. G. (2003). Security in Big Data. Journal of Studies in International Technology, 7(4), 379-403. doi:10.1177/1028315303257120",
    "Andrade, M. S. (2006, July 19). The use of Big Data in System Security. Journal of Research in Computer Science, 5, 131-126. doi:10.1177/1475240906065589"
  ],
  "distributions":[
    {
      "uri":"https://datahub.io/dataset/bigdata_analysis_security",
      "comment":"Folder with 15 data files"
    }
  ],
  "files":[
    {
      "name":"2015ResearchDataCollection.json.txt",
      "format":"json",
      "abstract":"Contains data collection for the research analysis",
      "size":"78 K",

      "uri":"https://datahub.io/dataset/bigdata_analysis_security/2015ResearchDataCollection.json",
      "permissions":"Open Database License 2.0",
      "dates":
        {
          "dateCreated":"2015-10-10"
        }
    },
    {
      "name":"2015ResearchAnalysis.tsv.txt",
      "format":"tsv",
      "abstract":"Contains insults gleaned from messages sent to the LKML mailing list by Linus Torvalds during the year 2013",
      "size":"256 K",

      "uri":"https://datahub.io/dataset/bigdata_analysis_security/2015ResearchAnalysis.tsv.txt",
    },

```



```

    "writeErrors" : [ ],
    "writeConcernErrors" : [ ],
    "nInserted" : 2,
    "nUpserted" : 0,
    "nMatched" : 0,
    "nModified" : 0,
    "nRemoved" : 0,
    "upserted" : [ ]

```

}) the most important part, when we get the idea right, each data of the manifests indexes are having “_id” that is generated by the machine.

However, when we include the first braces and the array name, they only notify us with

```
WriteResult({ "nInserted" : 1 })
```

that does not represent what we want and it will not gave us the right result for searching, updating, and removing.

Further in this sprint, we are making sure the functionality of the json array data collections inside mongoDB by searching, updating, and removing the manifest.

//find specific manifest by creator for search. In the future we also able to do search by keywords, or author. The template for searching

```
>db.manifest.find({"creator":"insert creator's name here"}).pretty()
```

//what we actually did

```
>db.manifest.find({"creator":"Sean"}).pretty()
```

Method pretty() only functioned to display readable output. The queries above resulting on displaying all the information in the manifests[0] since creator Sean information is there.

//edit manifest content by update query, we decided to treat id as the primary key here since the machine distinctly generated it. In the future, we will use _id as the parameter to update. The template query for edit is as follow

```
>db.manifest.update({"_id": ObjectId("insert ID number here")},{ $set:{whatever user wants to update}})
```

//what we actually did

```
>db.manifest.update({"_id":
ObjectId("5825e06e236518c7867540ea")},{ $set:{ "creator": "Dewi Endah" }})
```

Then, mongo will inform the update result by

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

And further we can see the change by

```
>db.manifest.find().pretty() and compare the result with the previous data.
```

```
//delete specific manifest based on the id since id is distinct.
```

```
>db.manifest.remove({"_id": ObjectId("insert ID number here")});
```

```
//what we actually did
```

```
>db.manifest.remove({"_id": ObjectId("5825e06e236518c7867540ea")})
```

Resulting on the deletion of the 2nd index of the manifests array.

This part is user database section, which has function in users' informations and their privileges. This collection is taking part in log in, log out, and session as users' privileges being recorded. We treat username as a primary key, so there will be no same username in this collection.

Insert into the user collection or registering new user into the system.

```
//creating users collection
```

```
>db.createCollection("user")
```

```
//registration or inserting new user to the user collection
```

```
>db.user.insert({manually insert user data})
```

```
//example1 based on the user class
```

```
>db.user.insert({
  username: 'username',
  hashed_password: 'password',
  salt: 'ABC123',
  permission_level: 'is_student',
})
```

```
//what we actually did
```

```
>db.user.insert({
  username: "sean",
  hashed_password: "qwerty0987",
  salt: "ABC123",
  permission_level: "is_creator"
})
```

Hashed password will be automatically generated by the php script in the registration page. Thus, we ensure the security of the user's account.

//find the user informations based on the username. The password will not be shown since it hashed.

```
>db.user.find({"username": "insert username here").pretty()
```

//we did the query below and sean informations is displayed

```
>db.user.find({"username": "sean").pretty()
```

//update in the user collection, in this case we implement the update in permission level.

```
>db.user.update({"key": "value here"} ,{$set:{"key": "new value here"}})
```

//our actual query

```
>db.user.update({"permission_level": "is_creator"} ,{$set:{"permission_level":  
"is_student"}})
```

//remove a user from the collection by this template, in case the user is deactivated the account

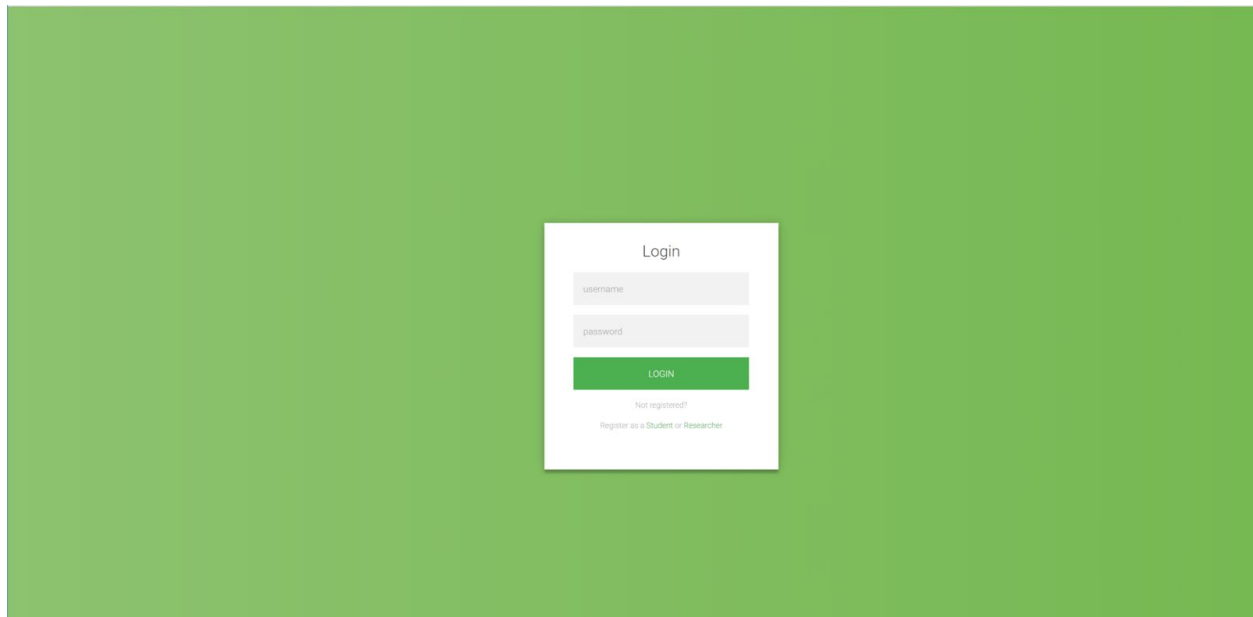
```
>db.user.remove({"username": "insert the username here"})
```

//what we actually did

```
>db.user.remove({"username": "sean"})
```

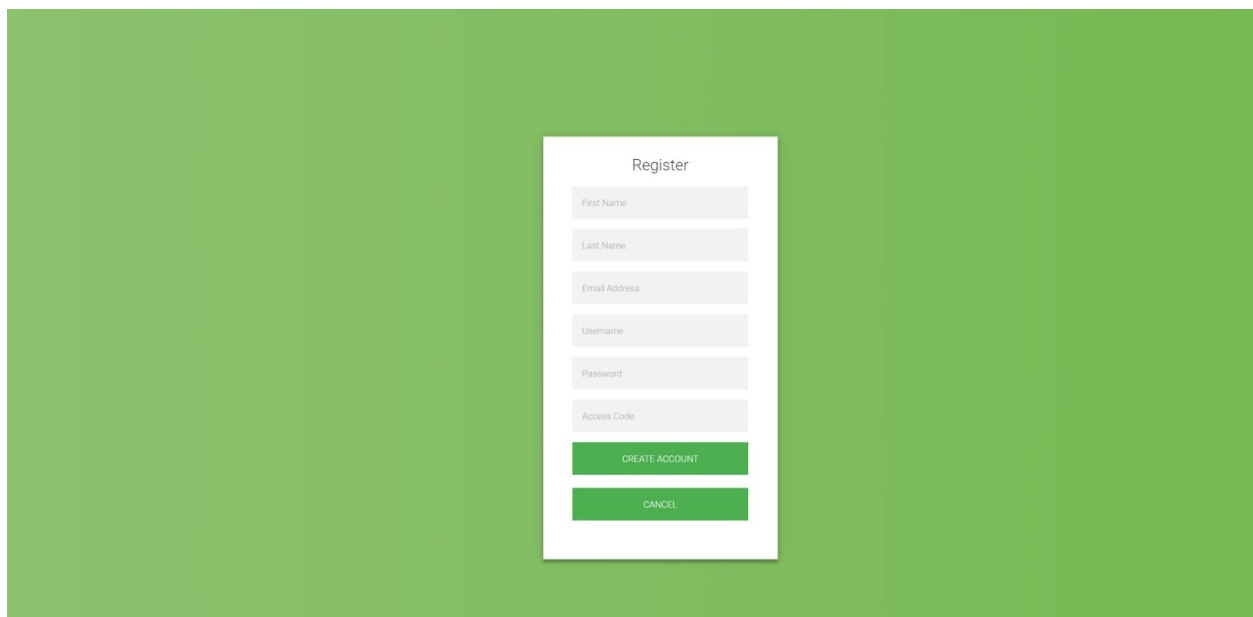
UI

Pages



A login page with a green background. In the center is a white card titled "Login". Inside the card, there are two input fields: "username" and "password". Below these is a green button labeled "LOGIN". At the bottom of the card, there is a link that says "Not registered? Register as a Student or Researcher".

Above is the login page where they can either login or create a new account



A register page with a green background. In the center is a white card titled "Register". Inside the card, there are six input fields: "First Name", "Last Name", "Email Address", "Username", "Password", and "Access Code". Below these is a green button labeled "CREATE ACCOUNT". At the bottom of the card, there is a green button labeled "CANCEL".

Above is the register page so that users can make new accounts. Access code will disappear if registering as a student

Search: Manifest Search Keyword Search

Create New Manifest

List:

Meta data	Update Delete
Meta data	Update Delete
Other Stuff	
Meta data	
Other Stuff	

Above is the Browse manifest where the user can search through a list of manifests and view/update/delete it.

File Picker

Choose File

Details

Manifest Name: kdifd
Author Name: kdifd
Date: kdifd

A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf

Above is the upload/create where the user can add a new manifest

Manifest Name: kdifd
Author Name: kdifd
Date: kdifd

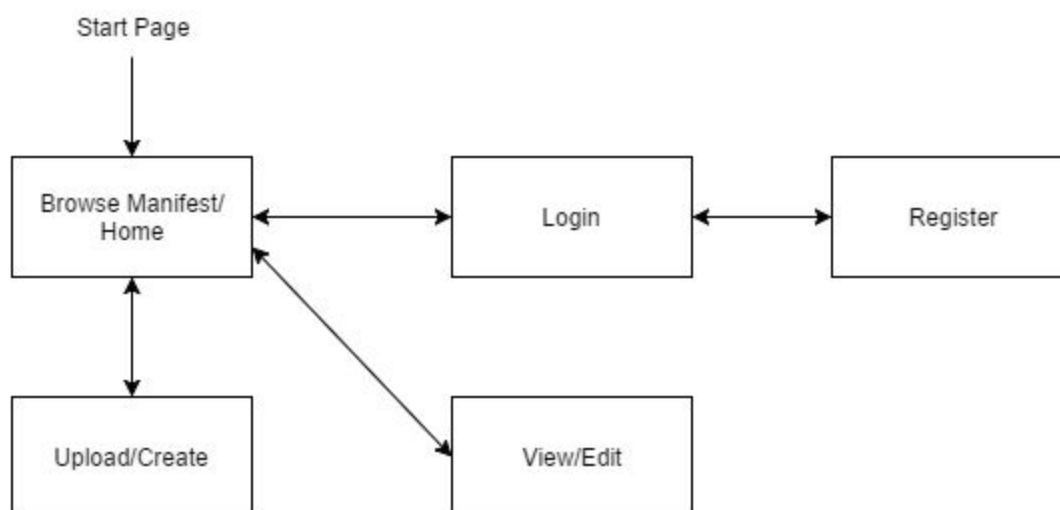
Download
Update

A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf
A lot of fddeeeee
A lot of JSON
A lot of dfd
A lot of fdfd
A lot of sdsf

Above is the view/update where the user can make changes to existing manifests

Group 10

Site Map



Above shows that you start on the browse manifest and can go to almost any page from it

Management of User Accounts/Roles

Overview

This web system will host a variety of users and researchers with differing roles and duties on the site. Different types of user accounts and permission levels are required in order to prevent users from overriding or messing with elements of the site and server that they have no business interacting with. To implement this, each user account will have a permission level attached to it based on what type of account it is. We will keep track of permission levels via php sessions. Our system will deal with 3 primary types of accounts: Reader/Student, Researcher, System Admin.

Guest: Permission level 0

Permission level 0 means the user is not logged in with any account. A guest cannot download, edit or delete any information on the site. They may only navigate a read the data hosted.

Reader/Student: Permission level 1

Permission level 1 means this user will have read and download access to the server only. The user may browse, search for, and view research, and may download

the datasets onto their own machine for future referencing. Users of this permission level may NOT contribute any datasets or manifests to the system, and therefore will be unable to edit anything on the site. Because these users have little to no impact on the system, anyone may make a user account at this level, provided they have an email account. When an account at this level is created, the user is stored in the database with a permission level of 1, which will prevent them from doing any of the forbidden tasks mentioned above.

Researcher: Permission level 2

Permission level 2 means the user has all permissions provided at level 1, as well as permission to upload datasets, edit/remove datasets under their own creation, and request edits to datasets outside of their creation. Users of permission level 2 are the researchers & contributors that wish to share their information on our platform. Because these accounts may have a high impact on the site, and will be uploading files into our system, it is important that we are able to control who can and can not create an account with permission level 2. To do this, we have made access to this server as a researcher 'invite-only'. When attempting to register an account with this level of permission, an access code will be required. These access codes will be known only to the System Admins. In the real world, this would be akin to someone purchasing an access code for an online textbook. When an account is registered with a valid access code, the user is stored with a permission level equal to 2.

System Admin: Permission level 3

Permission level 3 is the highest amount of access possible for this server, available only to those who directly work with the system's backend (us). There is no way to create a user account with permission level 3 within the system, it may only be created manually within the database. Users with permission level have full control over the system and it's data. They may edit and remove data from the site regardless of whether or not they were the original creators, and can edit/remove user account information.

Sprint 3

Group Meetings

Meeting 1:

Date: 11/14/2016

Time: 2:00pm - 5:00pm

Attendance: Andy, WeiXian, Dewi, Kate

Discussion: Revise sprint 2 based on the feedback, make log in, register, upload manifest work

Meeting 2:

Date: 11/15/2016

Time: 2:00pm-3:15pm

Attendance: All

Discussion: Official discussion/distribution of task assignments

Meeting 3:

Date: 11/18/2016

Time: 2:00pm-3:15pm

Attendance: Alex, Andy, Dewi, Kate, WeiXian

Discussion: Ensure all the pages work as expected

Meeting 4:

Date: 11/18/2016

Time: 9:30am - 12:00pm

Attendance: Alex, Andy

Discussion: Worked through issues in accessing manifest data through php, discussed UI and site navigation. Made the site fully navigable.

Sprint 3 Task List

- Deploy to group account (put link in documentation)
- Server oversight (fixing https)
- UI
 - Clean/style UI
 - Finish adding secondary features
 - All buttons functioning
- User Documentation
 - Begin User Documentation (finish by end of sprint 4)
- Testing
 - Edge Cases
 - Explain examples in our server of error checking & testing

Task Assignments

- Deploy to Group Account - Wei Xian
- Restructuring https for proper redirection - Wei Xian
- Begin User Documentation - Assia
- Test Documentation - Kate
 - Edge Cases
 - References to code
 - Explanations, and fail/success cases for each.
- Make sure Table of Contents is correct before submitting - Andy
- UI
 - Make site readily navigable (regardless of completed functionality) - Andy
 - General formatting/prettiness - Alex
 - Finish Login/Logout/Sessions - Andy
 - Finish and clean up user Registration - Andy
 - Upload/Create - Dewi
 - display JSON file upload - Dewi
 - Edit Manifest - Kate
 - Delete Manifest - Kate
 - Search on Manifest - Andy
 - Download to Local Machine - Assia

Login, Logout, and User Sessions

Login

If a user is not logged in, they make click a header link to log in on any page. From there, they are redirect to login.html, where they can enter their username and password. If their username/password combination is a verified match in our database, their session begins and they are redirected to the home page. If their credentials do not match, they are notified that their login failed, and may either navigate back to the login screen or to the site's homepage.

Logout

If a user wishes to end their session, they click a header link to logout on any page. Their session will be destroyed, and they will be redirected back to index.php (the home page).

Sessions

A small change has been made to sessions since the last sprint: To account for guest sessions, where the user is not logged in but can still navigate the page, a permission level of 0 will be given, that restricts their access to read-only, without being able to download datasets and manifests like a student user would. To reflect this change, permission levels have been bumped up by 1. Students now have permission level 1, Researches 2, and SysAdmins 3. This change has been reflected in the code and in the Sprint 2 documentation.

Important user information, such as their username, name, and permission level will be kept track of as the user navigates our site through php sessions. Every page of our main site will have the following chunk of php code:

```
<?php
session_start();
$username = $_SESSION['username'];
$permission_level = $_SESSION['permission'];
$name = $_SESSION['name'];

if($name == ""){
    $permission_level = 0;
    echo 'Welcome Guest! <a href = "/html/login.html">Login</a><br><br> Need an
Account? Register as a <a href = "/html/newUser.php">Student</a> or <a href =
"/html/newResearcher.php">Researcher</a>';
}
}
}
```

```

        echo 'Welcome, ' . $name . '!' <a href = "/html/logout.php/">Log Out</a>;
    }
?>

```

Upon each page's loading, the important session variables are loaded into local variables for the page to use (\$username, \$permission_level, \$name). Immediately after, we check to see if the session is one of a logged-in user, or an empty session by looking at the name. Since all users are required to provide their first name upon registration, a blank \$name field means that there is no user using the session, therefore a 'guest' session begins, and \$permission_level is set to 0. The user is also given a prompt to Log in, or register for a new account if they do not want to be a guest. If the name field is not blank, then the User is given a prompt to log out of their session whenever they choose.

Upload/Create Manifests

This page is redirected from Browse Manifest page. In Upload/Create Manifests only user with permission number 2, which means researcher can have access into the page. The permission number is retrieved from the session in the beginning of the page. Therefore, other permission number will not have access to this page. If somehow they break and successfully enter the page, it has shield that they cannot upload the file because they are not a researcher with permission number 2.

The mechanism of Upload/Create Manifests is the researchers have two options to do the insertion: upload the json file with the specification that we gave or create by typing it directly in the text box that we provided. When the researchers press the submit button, the back-end mechanism is started.

Create Manifests via upload

When the submit button is pressed, the system will check for the permission level. If the permission level is 2, the process will be continued to upload, and refuse otherwise. Uploaded file will temporarily be stored in the temp place generated by the machine. We ensure the file being uploaded by is_uploaded_file() method. Then, the temp file will be moved to the desire place with the same name as the local file with move_uploaded_file() method. In this system, the directory to storing the json will be /var/www/html/test-cs4320/SWE_Group10/public/json. The error checking for uploading include:

file_exists("\$uploads_dir/\$newName") to check whether file already exists in the directory

\$_FILES["ufile"]["size"] > 500000 ensure that the file is less than 500KB

pathinfo(\$newName,PATHINFO_EXTENSION) == 'json' to ensure the file is in json format

For the json file, this system has specific format for the json file. Such as:

```

{
    "standardsVersion": "v0.0.22",
    "id": "1111201601",
    "creator": "Dewi",

```

```
"dateCreated":"11/11/2016",
"researchObject":{
  "title":"Malware Security", ... (cont as desired)
```

} because this format is the only format that work on the next step which is storing these manifests into the database.

When all the requirements meet, the file will be uploaded into the desired directory and the system will give the user notification that the file is uploaded.

After the file successfully uploaded, the system will store the manifests into the database. Here, connection between the page and MongoDB is needed.

First, the system needs to connect to mongodb, in php the syntax work this way

```
$m = new MongoClient();
```

Second, select the desired database, in this case the selected database is named as SWE

```
$db = $m->SWE;
```

Then, select the desired collection where the data need to be inserted, in this case the selected collection is called manifest

```
$collection = $db->manifest;
```

After that, it needs to proceed the file that been uploaded. The mechanism can be done with `file_get_contents(the directory/filename)`. This method will try to get the data content from uploaded file and store it in `fileContent` variable.

```
$fileContent = file_get_contents("$uploads_dir/$newName");
```

Since the system know that the file is in json format, with complicated array inside. It necessary to decode the content. Therefore, the system will realize that the file is in json and will treat it as the json file

```
$json = json_decode($fileContent);
```

The most important part is the insertion of the data into the manifest collection. This is done inserting the json data that been decoded into the collection with query bellow

```
$collection->insert($json);
```

The system will give the users notification whether the insertion was successful or not.

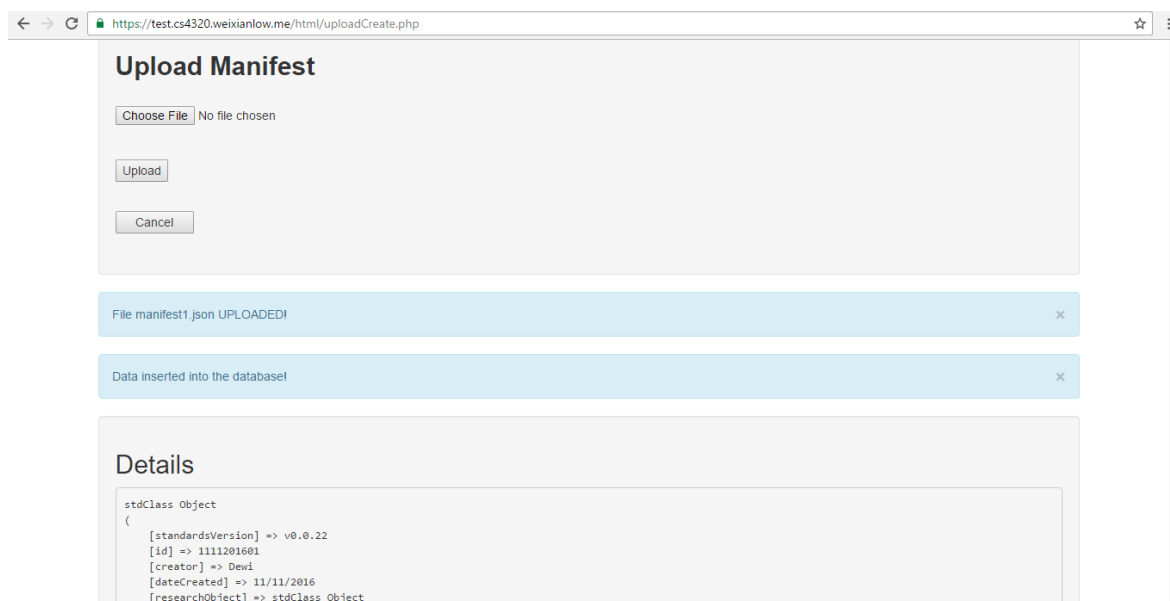
Last, after the upload and insertion process, the system is design to display the uploaded file.

The purpose of this effort is to give the user chance to review what they have been uploaded and check whether they are inserting the right manifest into the database. Further anticipation action will be provided in next sprint. The method to print is by using `print_r` specialty printing the array.

```
echo '<pre>';
```

```
print_r($json);
```

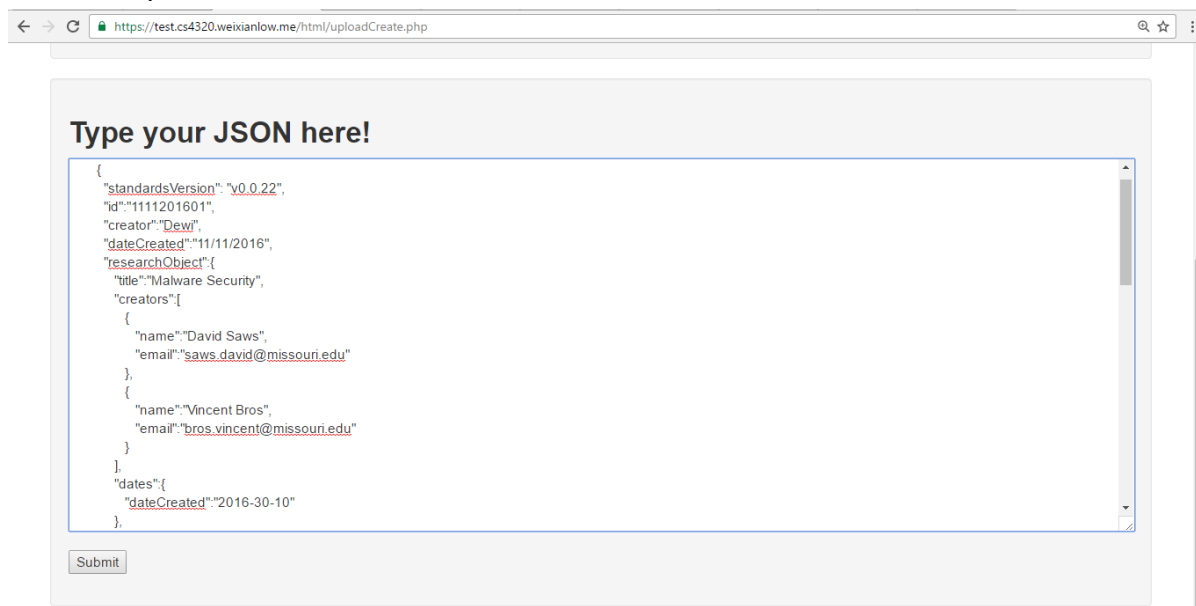
```
echo '</pre> </div>';
```

Above is the end result of uploading new manifest in uploadCreate.php

Create Manifests via TextBox

Another way to create new manifest is by typing the json file manually at the text area that is provided. The format should follow the system requirement (like the image below). For now, the system does not have any error checking or json grammatical error. This stuff will be done further in Sprint 4.



The mechanism for this uploading is similar with file uploads. The only difference is that the system needs to parse the text input and encode it as json format and then insert it into the database with the parse text. The insertion also identical with what the system does in the file upload.

Server Deployment (Wei Xian Low)

Due to the nature of the web application, this web application would require an environment that is flexible enough to handle web request and also able to handle a noSQL system running alongside with it. With that requirement in mind, an EC2 instance powered by Amazon AWS is used in this set up. The EC2 instance has a Ubuntu 14.04 environment being used for our web app and hosting MongoDB.

Dependencies

In this server, a few dependencies needed to be installed on the server to ensure proper functionality is provided. By using the “apt-get” function available in the bash terminal the following modules/functionality is installed in the following order:

1. Apache2

To install the apache2 web server to allow traffic redirection to the proper file path, the following command is used:

```
$ sudo apt-get install apache2
```

2. MongoDB

The first step required to install MongoDB is to add the required repository into the system to let system know where to fetch the required files:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10  
$ echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen'  
$ sudo tee /etc/apt/sources.list.d/mongodb.list
```

After configuring the required repository for the installation, proceed with installing the required files for MongoDB:

```
$ sudo apt-get update  
$ sudo apt-get install mongodb-org mongodb-org-server
```

3. PHP5

PHP5 is required to allow use to execute php scripts to extend the functionality of our web applications, hence the PHP5 files will be installed into the system:

```
$ sudo apt-get install php5 php5-dev libapache2-mod-php5 apache2-threaded-dev php-pear php5-mongo
```

4. MongoDB PHP Module

After installing PHP5, it needs to be configured to properly call MongoDB when mentioned in the script. The Mongo PECL extension needs to be installed:

```
$ sudo pecl install mongo
```

After installing the extension, PHP5 to recognize the extension, navigate to the “php.ini” file to add a line at the end of the file.

```
$ cd /etc/php5/apache2/
```

Then by using the nano editor in bash, navigate to the end of file for the file “php.ini” and add the following line at the bottom in between the quotation marks:

```
“extension=mongo.so”
```


After configuring the php.ini file, apache services is restarted to refresh its configuration

```
$ sudo service apache2 restart
```

5. Testing of Dependencies

After installing all the necessary dependency needed by the web application, a test and checking is needed to be performed to ensure that all installation and configuration is performed correctly.

To test if apache is working correctly, we would need to navigate to the public IP address of the EC2 instance, by navigating it through a web browser this page should appear serving as a landing page:



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all

This page shows that the apache is configured correctly and it's properly redirecting traffic to the proper directories. To test if PHP is working, a simple PHP script is created with the following line

```
<?php phpinfo(); ?>
```

By navigating to the php script on a web browser, this page should appear:

PHP Version 5.5.9-1ubuntu4.20	
	
System	Linux ip-172-31-23-129 3.13.0-100-generic #147-Ubuntu SMP Tue Oct 18 16:48:51 UTC 2016 x86_64
Build Date	Oct 3 2016 13:00:15
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mongo.ini, /etc/php5/apache2/conf.d/20-readline.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API20121212,NTS
PHP Extension Build	API20121212,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled

With this page appearing, scroll down to the mongo section to see if php is configured to work with MongoDB and the proper port number is assigned:

mongo

MongoDB Support	
Version	1.6.14
Streams Support	enabled
SSL Support	enabled
Supported Authentication Mechanisms	
MONGODB-CR	enabled
SCRAM-SHA-1	enabled
MONGODB-X509	enabled
GSSAPI (Kerberos)	disabled
PLAIN	disabled

Directive	Local Value	Master Value
mongo.allow_empty_keys	0	0
mongo.chunk_size	261120	261120
mongo.cmd	\$	\$
mongo.default_host	localhost	localhost
mongo.default_port	27017	27017
mongo.is_master_interval	15	15
mongo.long_as_object	0	0
mongo.native_long	1	1
mongo.ping_interval	5	5

Domains and subdomains

Although Amazon AWS has provides a public IP address and a public DNS address, a shorter and simple domain is more easier to navigate and connect instead of typing in long numbers. Team member Wei Xian Low has provided a subdomain on his domain that will point to the public IP address provided with the EC2 instance. The provided subdomain is: <http://cs4320.weixianlow.me>. An A record is required with the domain's DNS to provide proper redirecting to the server.

With the nature of the code, a different environment to test codes that currently being tested so that untested code would not affect the actual program is created, hence a sub-subdomain is created to allow testing of code here: <http://test.cs4320.weixianlow.me>. In order to create a subdomain, a configuration file for apache2 is needed to be created to work with. A default configuration file could be found at this directory in the server: "/etc/apache2/sites-available/000-default.conf". By duplicating this file and renaming it, the data in the configuration is replaced to fit with the need of the web application. By pointing the DocumentRoot to a new directory other than the default directory for apache, the test code could then be isolated easily. The ServerAddress needs to be changed to test.cs4320.weixianlow.me to let apache understand where to redirect the traffic to their proper directory.

After configuring the necessary configuration, the new configuration file needs to be enabled. This is done through by running the following command in bash to notify apache2 to enable it:

```
$ sudo a2ensite <CONFIGURATION FILE NAME>.conf
```

apache2 is then restarted with the following command in bash for the configuration to be put into effect:

```
$ sudo service apache2 restart
```

By navigating to the URL, the directory serving the subdomain will be properly navigated with a helloworld.html file that has been placed in the directory.

Security

1. SSH/SFTP

Due to the Amazon AWS security protocol, every EC2 instance created is provided a signature permission key to allow password-less access through SSH and SFTP. The permission key is distributed to every team member to ensure equal access to the server. After obtaining the permission key, team member would have to then generate their own permission by using a key generator either through putty or the FTP client of their choice. The generated key is then unique to their computer and is then used to authenticate themselves to the server.

2. HTTPS/SSL Certification

Due to the nature of the web application we are currently developing, which involved the use of storing research data that could be confidential or containing sensitive data, a proper encryption and identity verification is required between the client and the server. SSL certificate is created for all of the domain so that identity is verified during data transfer. With HTTPS implementation, the risk of man in the middle (MITM) attacks are reduced significantly. LetsEncrypt, a free and open source SSL certificate provider that aims to help encrypt and provide SSL certificate to the world for free is chosen to provide us with the necessary SSL certificate. By using a certificate-bot provided from LetsEncrypt, we are able to generate the necessary certificate for all the domains and subdomains we're currently using.

To install the necessary SSL certificate from LetsEncrypt, the necessary installation file for it is downloaded into the server. Proper directory is selected to store the scripts:

```
$ cd /usr/local/sbin/
```

Then the necessary file is downloaded and made executable:

```
$ sudo wget https://dl.eff.org/certbot-auto
```

```
$ sudo chmod a+x /usr/local/sbin/certbot-auto
```

After downloading the necessary file, the script is executed to provide us with a proper SSL certificate. Both the main site and the testing site needs to be configured for SSL certification, the following command is being used:


```
$ certbot-auto --apache -d cs4320.weixianlow.me -d test.cs4320.weixianlow.me
```

The above command uses the downloaded script to automatically generate the required script. Bare in mind, the base domain needs to be the first one on the command and it's followed by the rest of the subdomains separating with a "-d" in the arguments of the command.

During execution of the script, a question is asked to see if user wants a 100% enforcement of HTTPS on the server, that option is selected.

After setting up the necessary SSL certificate, automatic renewal of the SSL certificate is created so fresh certification is available. Crontab is used to automate the renewal processes, the following command is used to edit the crontab configuration:

```
$ sudo crontab -e
```

And the following line is added to the end of the cron configuration file:

```
30 2 * * 1 /usr/local/sbin/certbot-auto renew >> var/log/le-renew.log
```

The above line tells crontab to perform the required command every monday at 2:30am and will output the log to a log file located at /var/log/le-renew.log

The SSL certificate that has been generated from us needs to be tested to ensure it's configured properly, the following site is used to test the generate SSL certificate to ensure it's properly set up:

<https://www.ssllabs.com/ssltest/analyze.html?d=cs4320.weixianlow.me&latest>

<https://www.ssllabs.com/ssltest/analyze.html?d=test.cs4320.weixianlow.me&latest>

3. File/Directory Permission and Ownership

To be able to handle file upload to the server and executing php scripts, all files and directory is to be configured properly with the right permission and ownership in the system.

For file uploads, the directory that's designated to handle all upload needs to be owned by the user www-data:

```
$ sudo chown www-data:www-data <insert file directory here>
```

For php scripts to be executed when needed to only allow reading and executing but restricted from writing:

```
$ sudo chmod 755 <insert file directory here>
```

Automatic Deployment From GitHub

Automatic deployment from GitHub is powered by using GitHub's webhook functionality to automatically calling scripts on the server to run shell commands when a commit or a push is performed to the GitHub Server. To handle the script calling, a link of the script is inserted into the setting of the Repo's setting page at the webhook section.

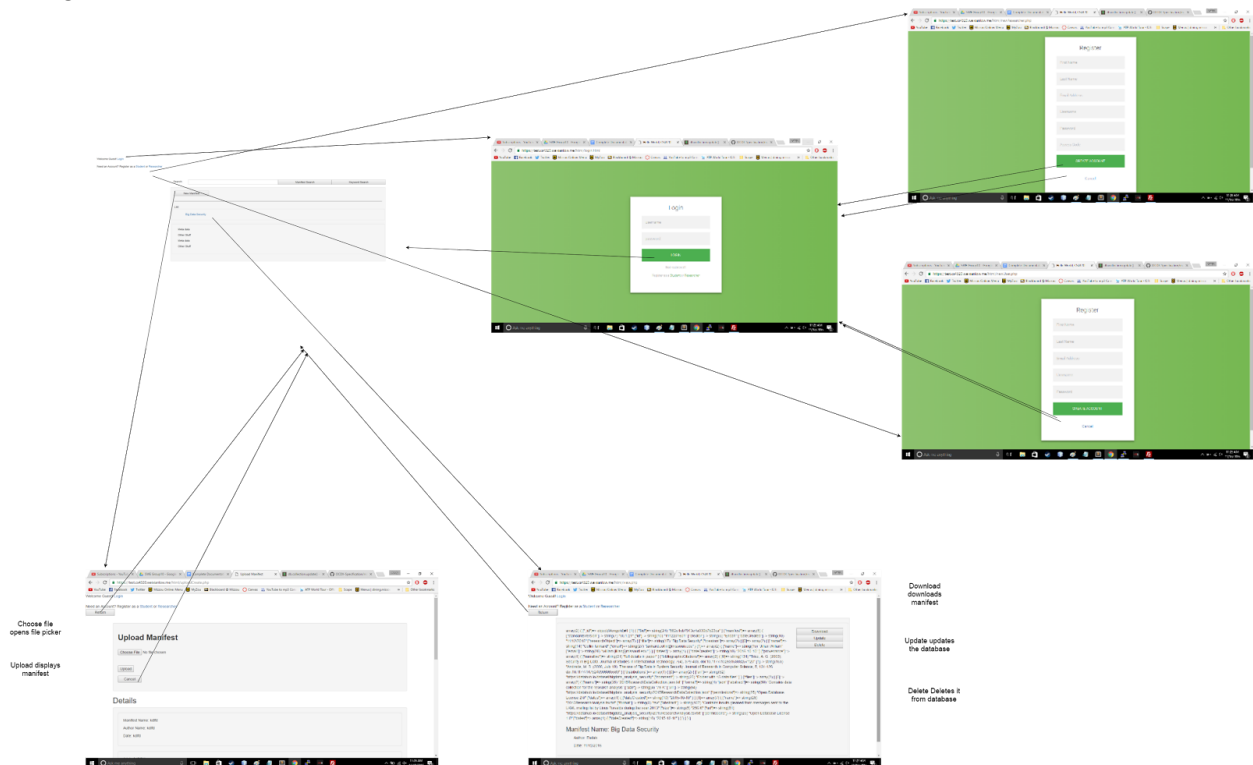
After configuring that, a script is needed to be created to handle command calling. PHP is being used to create the following script:

```
<?php
    exec("sudo git pull downstream master");
?>
```

A crontab scheduled command is also created to perform automatic pulling from the github server is performed everyday at 2am to ensure the latest commit is downloaded to the test site of the server.

User interface - Temporal Logic

Navigation



This is the navigation web that details where each button will navigate to.

Starting on the browse manifest page you can redirect to the login, register user, register researcher, upload manifest and view manifest.

From the login page you can go to register user, register researcher, and go back to browse manifest

From the upload manifest you can only return to browse manifest.

From view manifest you can only return to browse manifest.

All of the pages can redirect to login and register.

Here is a link to see it better

<https://drive.google.com/open?id=0B3y2qQKBQ4O2azFsY0pDTUI5eFU>

Use Case Analysis: Progress & Planning

At this point in the development process, the system is expected to have most, if not all required functionality working smoothly. This part of the document will look at every use case outlined in the requirements analysis, and discuss what still needs to be done. All incomplete functionality will be implemented in Sprint 4, except in the cases where the use case no longer fits into the system.

Browse Manifest - functional, needs polishing

- The index.php page posts links to each manifest view in a list for the user to navigate to. For now, the links display the manifest's title, but it would be useful in the future to include additional data, such as the creators or a short description.

Upload Data Set - functional

- Users can upload .json files directly into the system. They are parsed into the mongo database and are instantly available to browse on the site.

Contribute to Existing Dataset - not functional

- Existing datasets can not yet be edited, as it required the functionality of browsing and viewing manifests to be implemented first, which was done late in the sprint cycle.

Download Info - not functional

- Another use case that is dependent on uploading and viewing manifests. Will be implemented in Sprint 4.

Generate Upload Manifest - functional

- Uploading a .json file automatically generates and inserts the dataset's manifest into the manifest database.

Save - not functional/potential scrap

- Does not necessarily fit in with the system as it is being implemented. Will either be implemented or scrapped through Sprint 4.

Search on Manifest - not functional

- Dependent use case - will be functional upon further polishing of Browse Manifest use case.

Structure of User Documentation

1. Registering
 - a. Student
 - i. Creating account
 - b. Researcher
 - i. Creating account
 - ii. Access code
2. Logging in
3. Browse manifest page
 - a. Manifest search
 - i. Searching everything
 - b. Keyword search
 - i. Searching metadata
 - c. Create new
 - i. Creating new manifests
 - ii. Uploading new manifests
 - d. Update
 - i. View/edit manifest page
 - e. Delete
 - i. Permissions
 1. Deleting manifests

Sprint 4

Group Meetings

Meeting 1:

Date: 11/27/2016

Time: 4:00pm-8:00pm

Attendance: Andy, WeiXian, Dewi, Alex

Discussion: Group work day - restructured & cleaned up the server & github repo.
Developed functionality and discussed Sprint 3 feedback.

Meeting 2:

Date: 11/28/2016

Time: 3:00pm-8:00pm

Attendance: Andy, WeiXian, Dewi, Kate, Assia, Alex

Discussion: Work day, compile individual tasks together and make appropriate fixes.
Submit Sprint 4 Documentation

Sprint 4 Task List

- Elements that still need functionality
 - Download
 - Update
 - Delete
 - Search - Search by title working, need additional search options
 - List view of manifests - does not send correct data to view.php
 - JSON manual upload
 - Register Researcher - Negative feedback if wrong access code used
- UI
 - Create a viewable list on the home page that looks clean and has simple animations
 - Add styling to make everything more readable
 - Create a toolbar at the top
 - Make buttons less bland
- Deployment
 - Deploy full site

- Provide deployment script for automated deployment from Github repo
- User Documentation
 - Readme.MD should have full explanation of application deployment

Task Assignments

- Alex - UI tweaking & cleanup, search operations
- Andy - User Registration & sessions tweaks, implement sprint 3 feedback, general sprint documentation, fix communication between index.php and view.php
- Assia - User Documentation
- Dewi - JSON manual upload, view functionality
- Kate - Testing documentation
- Wei Xian - Deployment instructions & oversight
- All - UI functionality & php scripts as needed

Addressing Sprint 3 Feedback

- Couldn't find credential provided to test out each type of user
 - When registering a new researcher, the access code '123456' is required for the registration's success. This was left out of previous documentation by mistake. In the real world, there would be a vast database of access codes to be activated when a researcher is invited to join the community.
- Since students cannot upload a manifest they should not see the button "New Manifest"
 - Modifications have been made to index.php so that accounts with an access level below two (meaning they are not researchers or system admins) are not able to see the button.

UI Style Tweaks

On every page there is a toolbar that allows the user to redirect to the home page as well as login. Since, it does not have that much functionality it was not made static at the top of the screen. Instead it will disappear if the user scrolls down.

On the main page there is a search bar that searches through the title and author. This searchbar requires no button to submit as it does a real-time search. This allows the user to have constant feedback on whether or not what

they are searching is in the database. Also, it is a lot quicker than a submit button that would load a new page with all of the results.

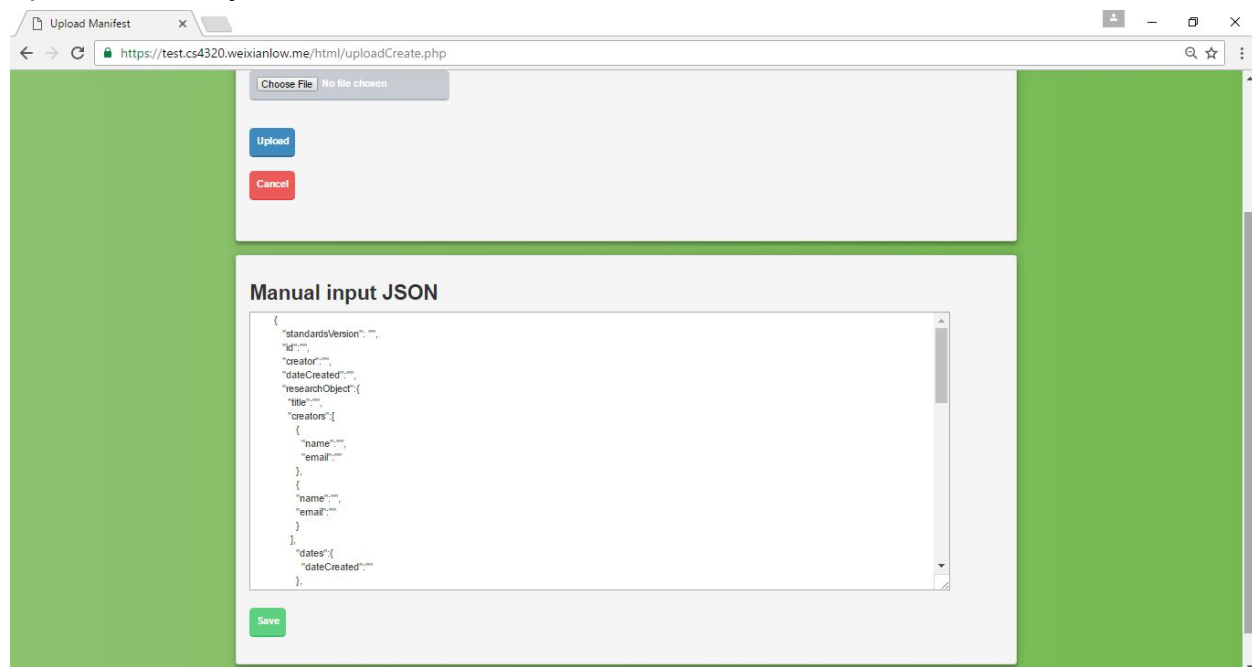
On the create page the buttons are color associated with what they do. So, something like cancel which can be dangerous to use is red. As well as overall aesthetics.

On the view page the buttons are also color associated and the overall aesthetics were improved.

Upload/Create Manifest

The Upload Create Manifests via file upload is fully functioned in Sprint 3. Thus, in Sprint 4, this page is focus on the manual JSON file upload via text area. uploadCreate.php consists of both options. Upper part is for the file upload and the bottom one is for the manual typing. The page will work only on one request at a time because the request is dependent on the button that is being pushed.

Upload manually via text area



The screenshot shows a web browser window titled 'Upload Manifest' with the URL 'https://test.cs4320.weixianlow.me/html/uploadCreate.php'. The page has a green background. The top section is for file upload, with a 'Choose File' button (disabled, showing 'No file chosen'), an 'Upload' button (blue), and a 'Cancel' button (red). The bottom section is titled 'Manual input JSON' and contains a large text area with a JSON template. Below the text area is a green 'Save' button.

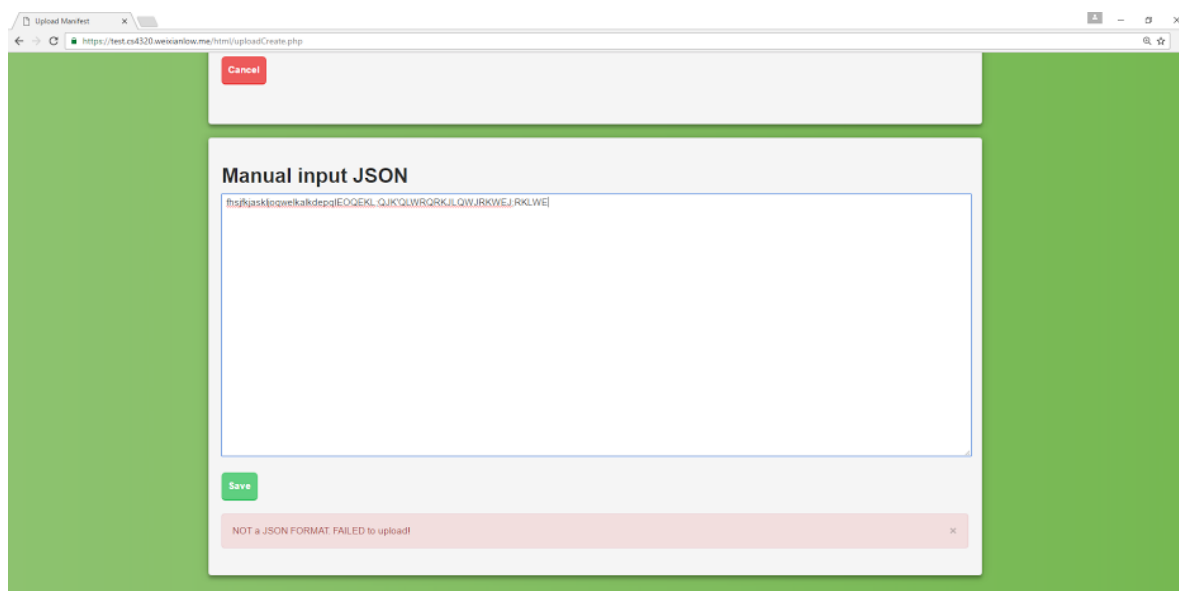
```

{
  "standardsVersion": "",
  "id": "",
  "creator": "",
  "dateCreated": "",
  "researchObject": {
    "title": "",
    "creators": [
      {
        "name": "",
        "email": ""
      },
      {
        "name": "",
        "email": ""
      }
    ],
    "dates": {
      "dateCreated": ""
    }
  }
}

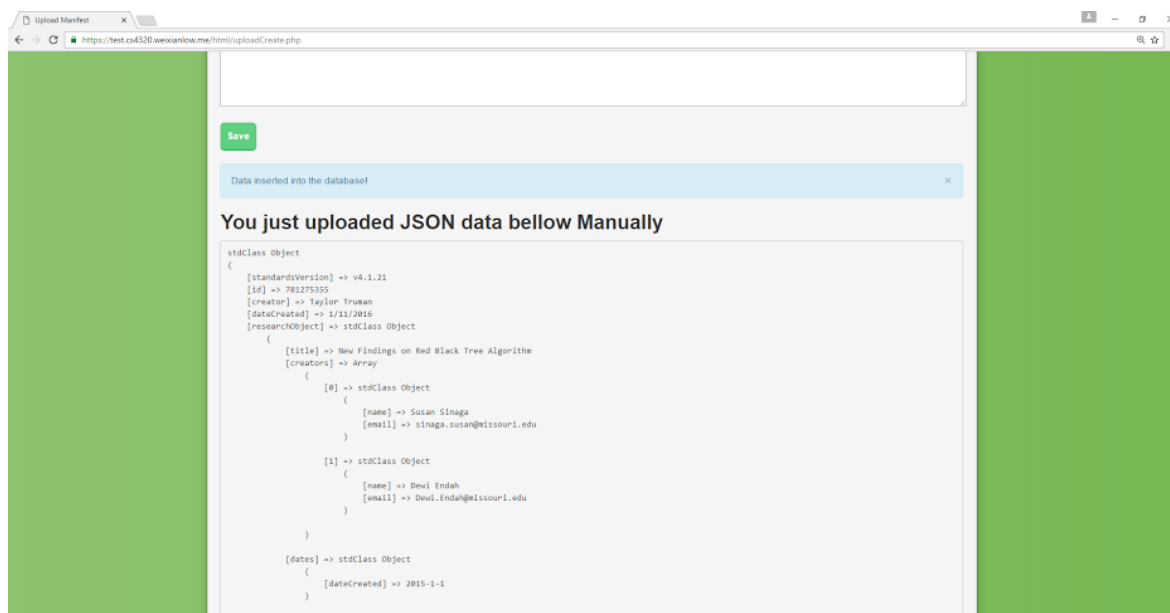
```

The page is providing text area and the save button in the bottom part. In this case, the researcher need to type the manifest data based on the template that is provided by the system. The input must be in valid JSON format. To make sure about the input, the system will validate whether the file typed in the text area is in JSON format or not. When the researcher push the save button, the back-end mechanism will starting to work. First thing the system does is retrieve the user input text from the text area with this code `$textInput = $_POST['textInput'];`

After getting the text, JSON format validation is begin. The only validation step is by decoding the text to the JSON format and check whether error exists during decoding process. The decoding is done with this line of code, `$json = json_decode($textInput);` this resulting the decoded data to be stored in `$json` variable. Then, the system checking whether error exists during the decoding process by this line of code `if(json_last_error() === JSON_ERROR_NONE)`, `json_last_error()` returns the last error occurred during the JSON decoding. Thus, when `json_last_error()` is equals to `JSON_ERROR_NONE` means there is no error occurred during the decoding. Therefore, the manually inputted JSON file is valid JSON format.



Error message because the input text is not in JSON format



Success notification and chance to review the uploaded JSON file

As what shown in the pictures above, when the researcher typed the incorrect JSON file, the system will throw notification that something wrong with the input text. In the other hand, the system will try to connect with the database and insert the new manifest into the manifest collection. Finally, when insertion succeed, the system will give the researcher notification that the insertion into database was succeed and display the inputted JSON text. Here, the researcher has a chance to review the new manifest data. If the researcher did typo or put the wrong information, they can either delete the data or edit it in view.php page.

List View/View Manifest

Previously, there was an error in the communication between index.php and view.php, when the user clicked on a manifest view. No matter which manifest the user wished to view, they were always redirected to a page displaying the last manifest in the database. By restructuring the php scripts that send the user between pages, this was fixed. Previously, sessions were attempted as a way to transfer data (namely the manifest) between pages. This was troublesome, and has been replaced with a POST form. On index.php, each manifest search result is echoed as a form with the title of the manifest as an input button. When the user clicks on the button, the title is sent via POST to view.php. The database is then queried for the manifest with a matching title, and that manifest is displayed in view.php for the user. This change initially broke the search on manifest functionality, but this issue was resolved.

Use Case Analysis: Progress and Planning (Sprint 4)

At this point in the project, full functionality, with a few minor tweaks needed, is expected. While not all of the outlined goals have been met at this point, progress is going well and there is clear improvement on the system design from the last sprint.

Browse Manifest - Functional

- The index.php page posts links to each manifest view in a list for the user to navigate to, as well as the author of the manifest.

Upload Data Set - Functional

- Users can upload .json files directly into the system. They are parsed into the mongo database and are instantly available to browse on the site.

Contribute to Existing Dataset - In Progress

- Existing datasets can not yet be edited, due to backup on some dependent use cases, but they can be deleted. A clear plan is established for implementation of this update. If the user is the researcher responsible for the manifest, view.php will display their manifest back to them as an editable text field. The 'Update' and 'Delete' buttons will be visible for the creator to do as they please with their manifest.

Download Info - In Progress

- Existing Datasets can not yet be downloaded, due to backup on some dependent use cases. Once this use case is implemented, users of permission level above zero (any non-guest user) can click the download button to have the .json file of the manifest downloaded onto their machine. There is an issue with implementing this, as some manifests are contributed manually and have no .json file to go with them. For these cases, a .json file will need to be generated out of the database entry.

Generate Upload Manifest - Functional

- Uploading a .json file automatically generates and inserts the dataset's manifest into the manifest database. The user may also manually generate a manifest, with a JSON standard template available for them to fill in with the necessary information.

Save - Scrapped

- Does not fit into the scope of this project. Any functionality this use case would provide is implemented via 'Contribute to Existing Dataset'

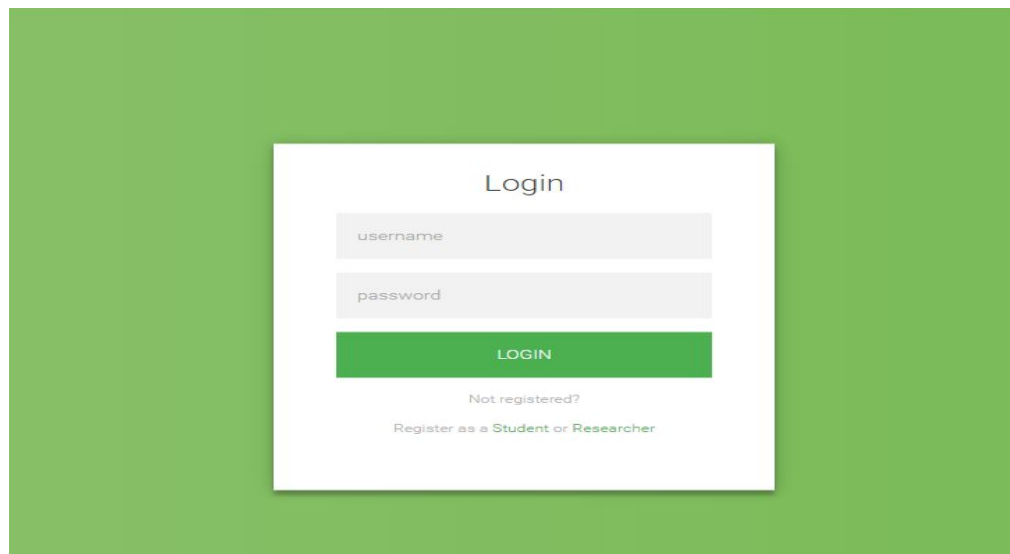
Search on Manifest - Functional

- Users can search the collection of manifests by keywords corresponding to either the title or author of the manifest. As the user types their keywords into the search bar, the list automatically updates and filters out links to manifests that do not fit their criteria.

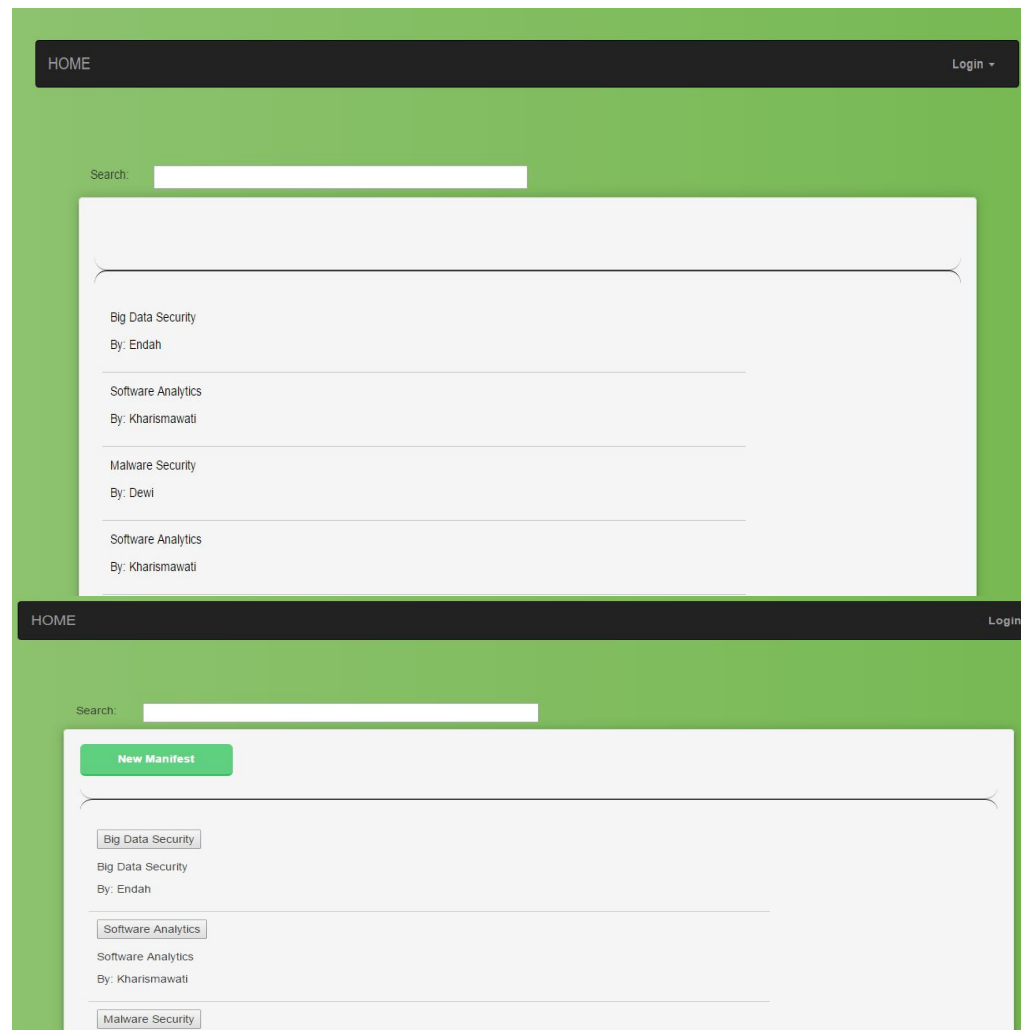
Testing Done in Application

Login/logout:

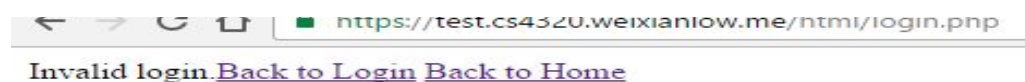
- Logging in with correct credentials
 - A user logs in with a username and password. A user must have previously registered themselves.
 - Successful login will bring the user to the main site. User must have correct username and password. User must have previously registered.



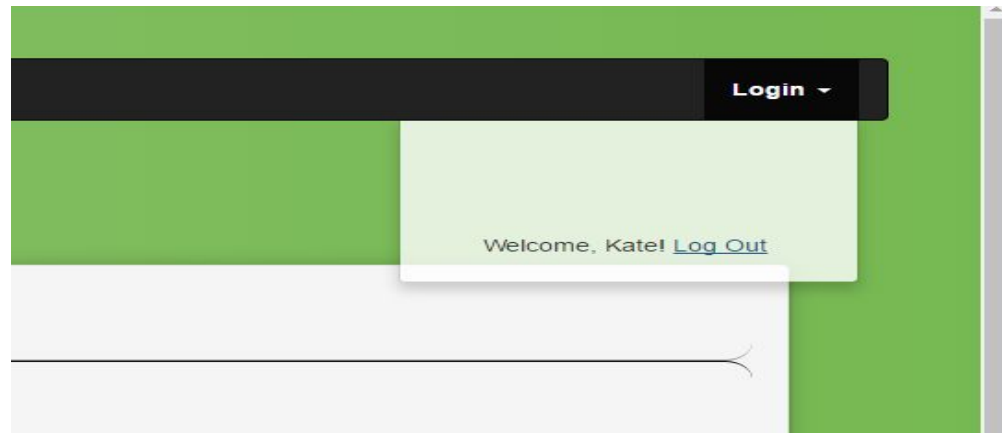
- The below pictures are the home pages for student(first picture) vs. researcher(second picture).



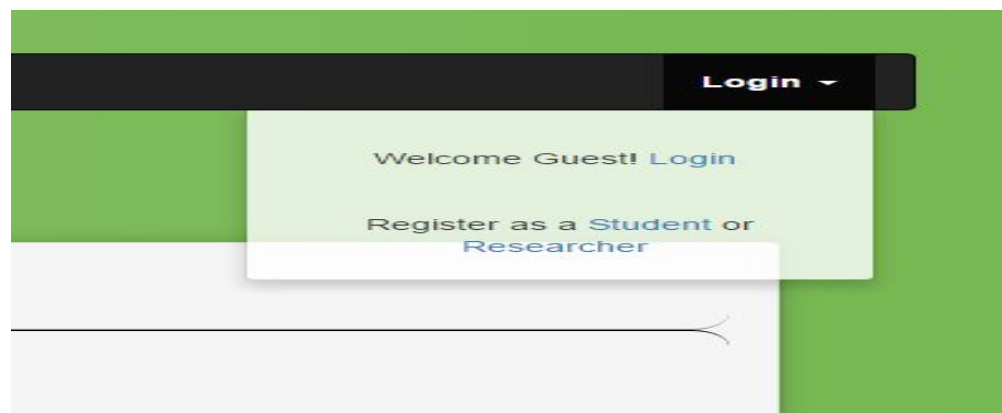
- Failed login will bring the user to a page that says “Invalid login.” The user can then click on the back to login button, and that will redirect the user to the original login page. Or click on back to home to direct back to the main site.



- Successfully logging out
 - When a user successfully logs into the site, they will see their is a logout button and can log out at anytime.



- Upon successfully logging out, the user will be redirected to the home page while logged in as a guest.

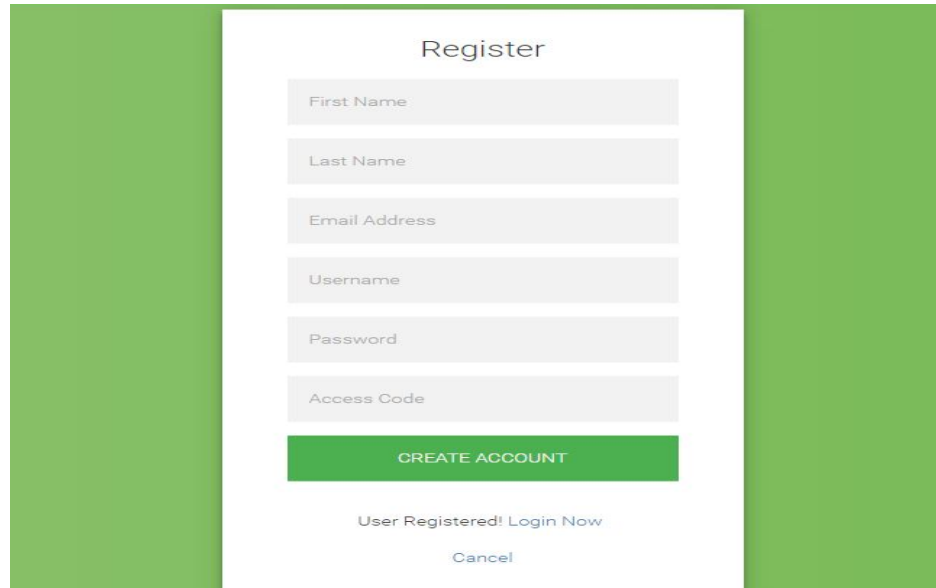


Registration:

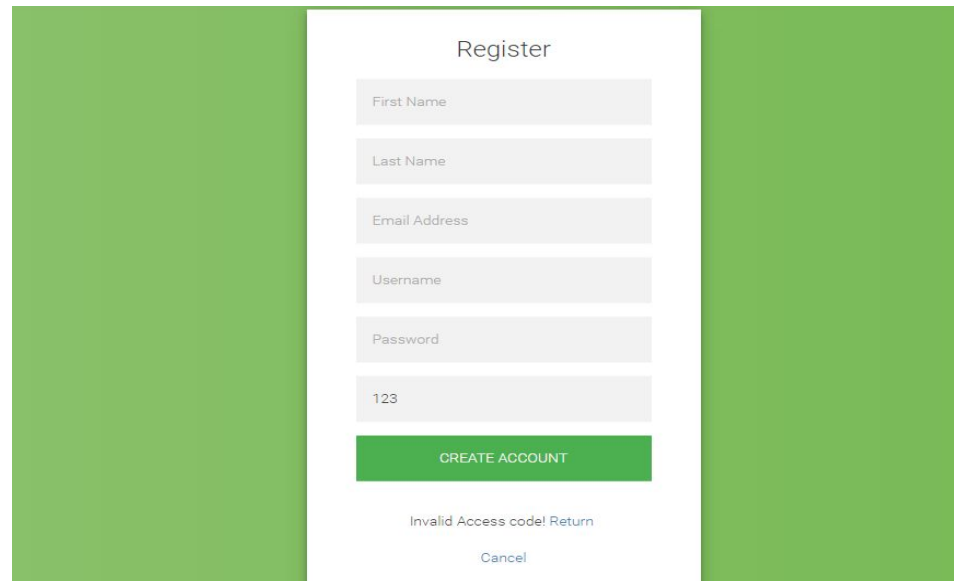
- A user can register as a student or researcher. The registration for researcher requires an access code, first name, last name, email address, username, and

password. A student is only required to have first name, last name, email address, username, and password.

- Successful registration for researcher:
 - A researcher must have a correct access code, as well as having all other text input boxes filled out for a successful register. With all correct information filled out, the user will be redirected to the register page with a “user register” at the bottom of the page.

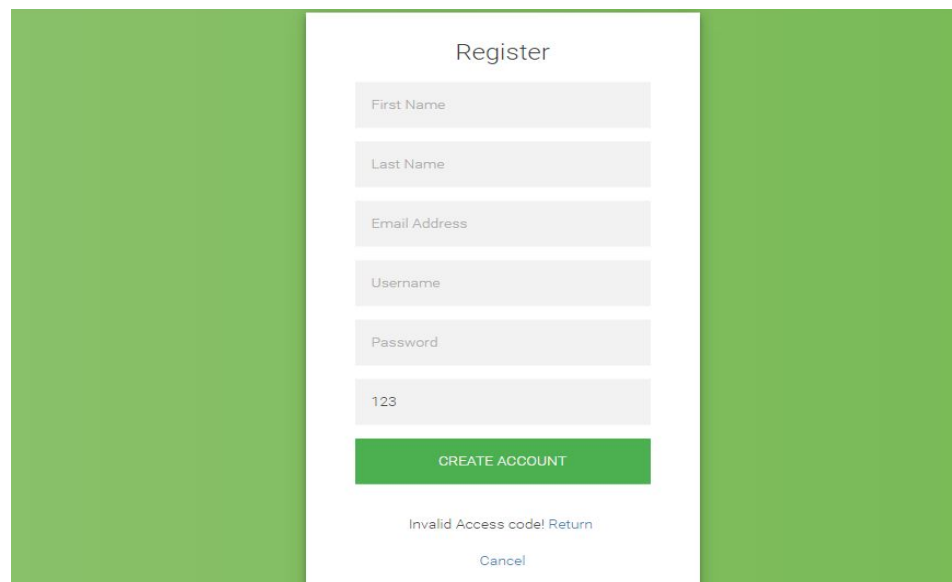
A screenshot of a web form titled "Register". The form is centered on a white background, flanked by two vertical green bars. It contains six text input fields stacked vertically: "First Name", "Last Name", "Email Address", "Username", "Password", and "Access Code". Below these fields is a green button with the text "CREATE ACCOUNT" in white. Underneath the button, the text "User Registered! Login Now" is displayed, followed by a blue "Cancel" link.

- Failed registration for researcher: Each case will refresh the page with “failed registration” message.
 - Failed registration case 1: Researcher does not have access code or invalid access code.



The screenshot shows a 'Register' form with the following fields: First Name, Last Name, Email Address, Username, Password, and a field containing '123'. A green 'CREATE ACCOUNT' button is below the fields. A red error message 'Invalid Access code! Return' is displayed, with a 'Cancel' link below it. The form is centered on a green background.

- Failed registration case 2: Researcher did not fill in all the required text fields.



This screenshot is identical to the one above, showing the 'Register' form with the same fields and the 'Invalid Access code! Return' error message.

- Successful registration for student:
 - Student filled out all required text fields. Successful registration will redirect user to registered page with “User registered” at the bottom of the page.

Register

First Name

Last Name

Email Address

Username

Password

CREATE ACCOUNT

User Registered! [Login Now](#)

[Cancel](#)

Upload/Create and Insertion - manifest:

- When a researcher successfully logs in, they will see that they can create a new manifest. Creating a new manifest means uploading a new JSON file to their account and inserting it into the database. A student cannot see this, they do not have that privilege. A researcher is allowed to give a title and description for the manifest. A title is required, but a description is not.
 - Successful insertion - “Data inserted into the database” and/or File “.json” UPLOADED! message and refreshed page upon successful upload:
 - The researcher uploaded the correct file(JSON file).
 - The researcher uploaded a file that does not exceed file size limit.
 - The researcher uploaded an existing file.
 - File was successfully inserted into database.

The screenshot displays a web interface with a green sidebar. At the top, a green 'Save' button is visible. Below it, a light blue notification bar states 'Data inserted into the database!'. The main content area features the heading 'You just uploaded JSON data bellow Manually'. Underneath, a code editor shows a JSON structure with fields like 'standardsVersion', 'id', 'creator', 'dateCreated', 'researchObject' (containing 'title' and 'creators' array), and 'dates'. Below the code editor, there is a file upload section with a 'Choose File' button (showing 'No file chosen'), an 'Upload' button, and a 'Cancel' button. Two more light blue notification bars are present: 'File package.json UPLOADED!' and 'Data inserted into the database!'. At the bottom, a 'Details' section is partially visible.

- Failed insertion - “You are not a not a researcher, you can not upload manifest!” or “File already exist” or “The file must be in json format” or “FAILED to upload. Try again!” message and refreshed page upon failed upload:
 - If any of the conditions above fail.
 - If no file was uploaded at all(file doesn't exist).
 - If not a researcher.

Upload Manifest

Choose File No file chosen

Upload

Cancel

You are not a researcher, you can not upload manifest!

Upload Manifest

Choose File No file chosen

Upload

Cancel

File already exists!

The file must be in JSON format!

FAILED to upload, Try again!

Updating - manifest:

- When a researcher successfully logs in, will also see an update button, but will only appear if there is already an uploaded manifest. The update button allows a researcher to update any information on a pre-existing manifest. Students will not see this, they do not have that privilege. Can also update the description, and title of manifest, but not required for a successful update.
 - Successful update - “successfully updated” message and refreshed page upon successful update
 - All conditions for insertion still apply for updating.
 - In addition to: database was updated correctly (no new entry was created, & correct fields were updated in correct spots in table).
 - Failed update - “failed to update” message and refreshed page upon failure.
 - Any of the conditions above failed.
 - File does not exist.

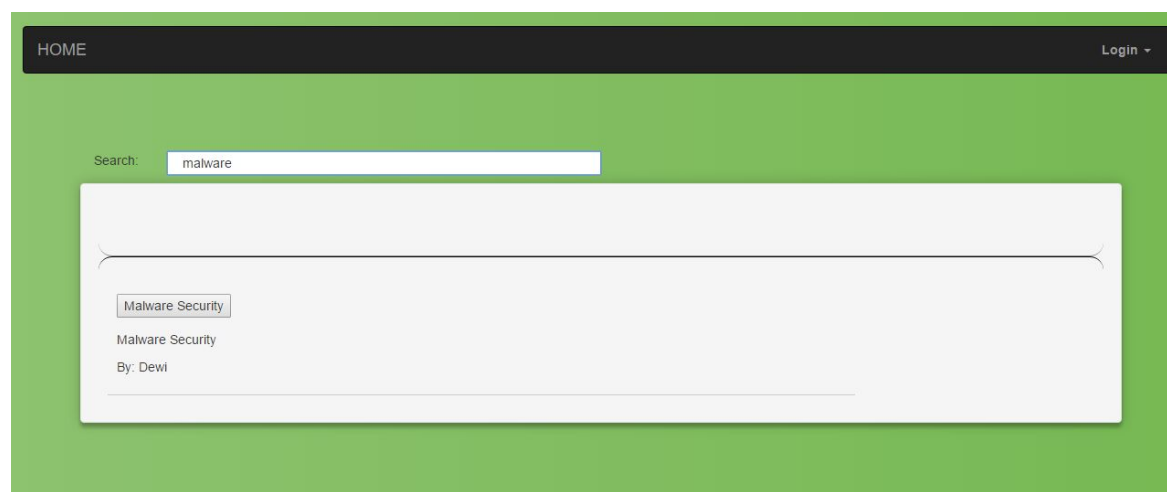
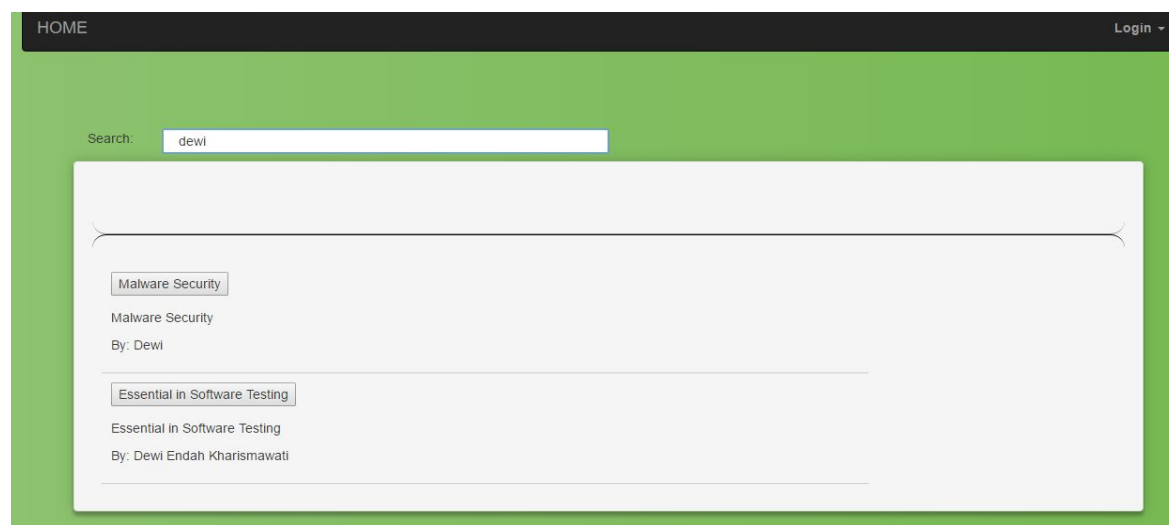
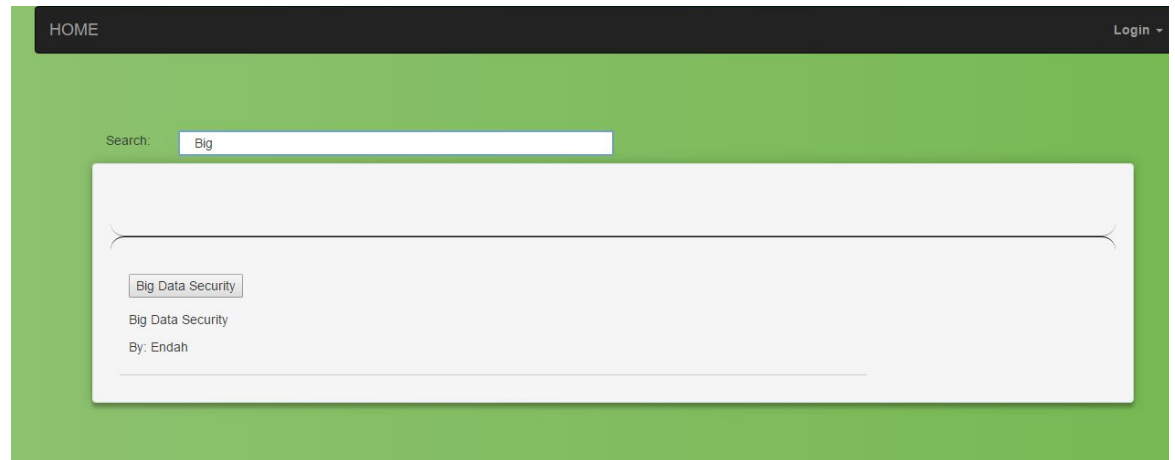
Deletion - manifest:

- When a researcher successfully logs in, will also see a delete button, but will only appear if their is already an uploaded manifest. The delete button will allow a researcher to delete any pre-existing manifest from their account. Students cannot see this, they do not have this privilege.
 - Successful deletion - “successfully deleted” message and refreshed page upon success:
 - Deleting pre-existing manifest.
 - Successfully deleted from account.
 - Deleted correct manifest from database.

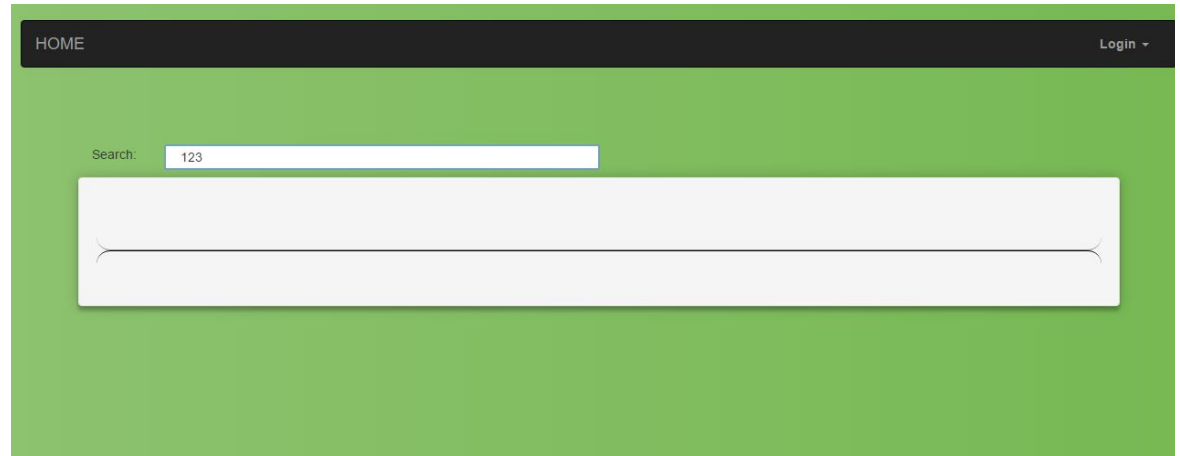
- Failed deletion - “Could not delete” message and refreshed page upon failure:
 - File does not exist.
 - Could not delete from database.
 - Information not deleted from account.

Manifest search:

- Any user who has registered and successfully logged in is able to search manifest. This means researchers and students. Users are able to search manifest with letters/keywords/phrases(title of manifest, description of manifest, date of manifest, ect.). A user is then able to see all manifest obtained by those letters/keywords/phrases.
 - Successful search - Shows all results obtained by what’s written in search bar on page:
 - The letters/keywords/phrases match a manifest name in database.
 - Database returns correct searched results.



- Failed search - Results are not shown on page
 - The manifest does not exist in database.
 - Manifest did not return correct searched results.



Integration/Regression Testing

- There will be no testing for either of these for this sprint, because it will not be of any use for us at the moment, and not necessary.

User Acceptance Testing

- Acceptance testing will be done further down in sprints when we have more of the project finished. The testing will be done by giving it to Professor Goggins, and the TA's and all the required elements will be close to be done.

Update/Upload Manifest

Update/download/delete are still in the process of testing, because they do not have their full functionality at this time.

User Documentation

Download Link:

https://github.com/weixianlow/SWE_Group10/blob/alh3q8_sprint4/UserDocumentation.docx

PDF View:

https://github.com/weixianlow/SWE_Group10/blob/master/UserDocumentation.pdf

Deployment

Full server deployment instructions are available on the README.md on the Github repository, linked below:

https://github.com/weixianlow/SWE_Group10/blob/master/README.md

Change Log

10/18/2016:

Individual work compiled into Version I.

Contributions:

- Alex - First pass screens, Contribute to Existing Dataset use case analysis
- Andy - Download Info use case analysis, first pass table list
- Assia - Generate Upload Manifest use case analysis, use case diagram first pass
- Dewi - Save use case analysis, sequence diagram first pass
- Kate - Upload Dataset use case analysis, class list first pass
- Wei Xian - Browse on Manifest, Search on Manifest use case analysis, full activity diagram first pass

10/30/2016

- Andy - Added System ERD
- Alex - Complete Design of the user interface
- Alex - Complete design of the information architecture

11/6/2016

- Andy - Added task list and task assignments following feedback on the first submission. Updated document will be submitted with Sprint 2

11/10/2016

- Individual Sprint 2 work compiled into document.

11/18/2016

- Individual Sprint 3 work compiled into document.

11/28/2016

- Individual Sprint 4 work compiled into document.

Glossary

OCDX	: Open Community Data Exchange
SNC	: Scripts, Notes, Configurations
DBMS	: Database Management System
AWS	: Amazon Web Services (A web hosting services provider)
Azure	: Microsoft Azure (Web Hosting Services Provider)
mySQL	: Open Sources Relational Database Management System
SQL	: Structured Query Language
Mac OS	: Apple's Operating System
Windows	: Microsoft Operating System
PHP	: Server Side Scripting Language
AJAX	: Asynchronous Javascript and XML
JSON	: Javascript Object Notation
UML	: Unified Modelling Language
Manifest	: A type of inventory/catalog detailing the information related to the datasets and SNC uploaded by a user.
Datasets	: Information and resources pertaining to the research done by a researcher.
ERD	: Entity Relationship Diagram
Use Cases	: List of actions and event, detailing interaction between an actor and a system to achieve a goal.
GitHub	: Web based git repository hosting services
git	: Version Control Software
HTML5	: Hypertext Markup Language v5
Database	: A structured set of data held digitally.
Wiki	: A website that provides collaborative modification of it's content.