OCDX Research Project

Full Documentation
17 October, 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

Table of Contents

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"

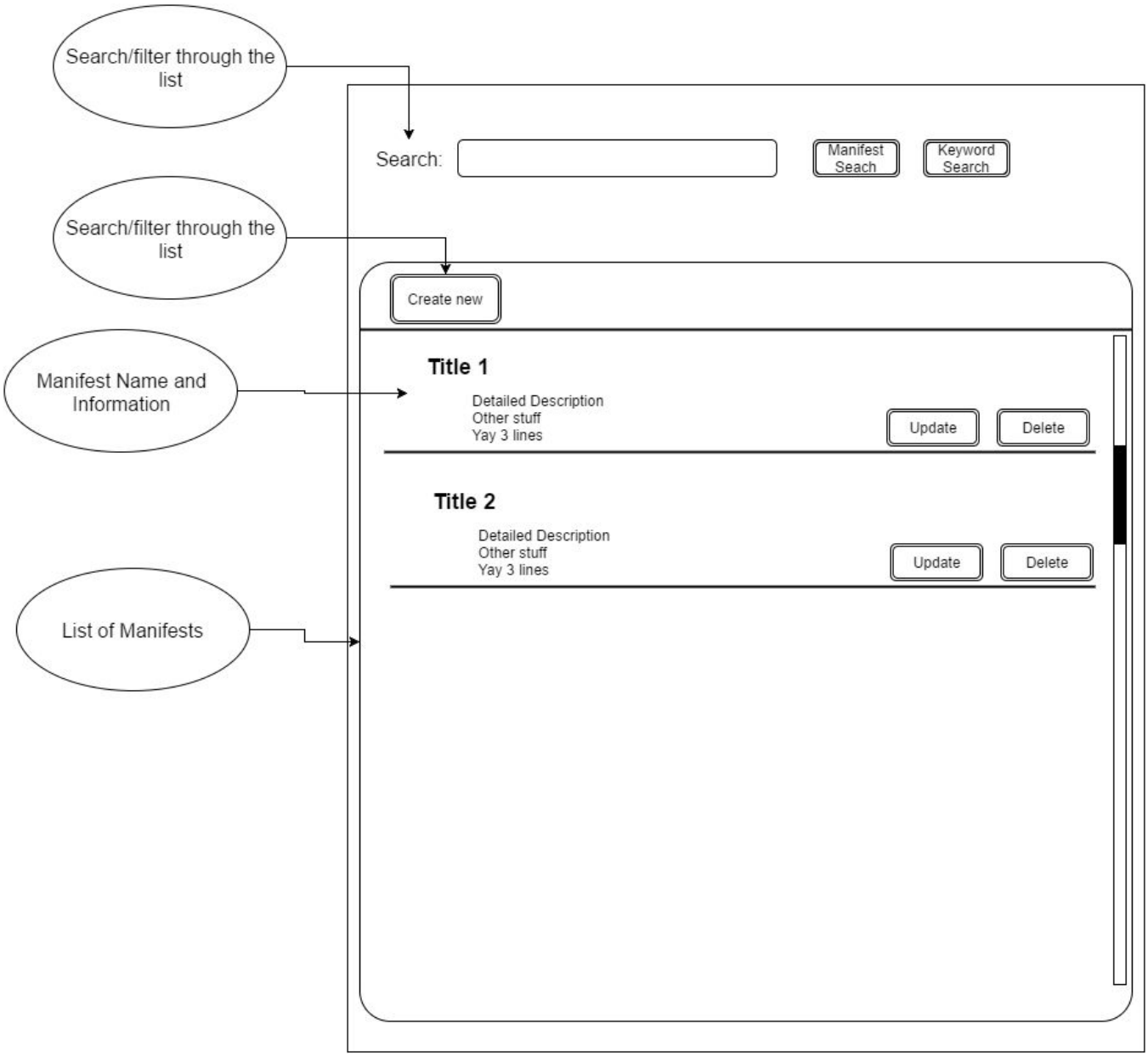
Dependent Use Cases
- Upload Data Set
- Generate Upload Manifest
- Contribute to Existing Dataset

System Design

Screen Views

Browse Manifest

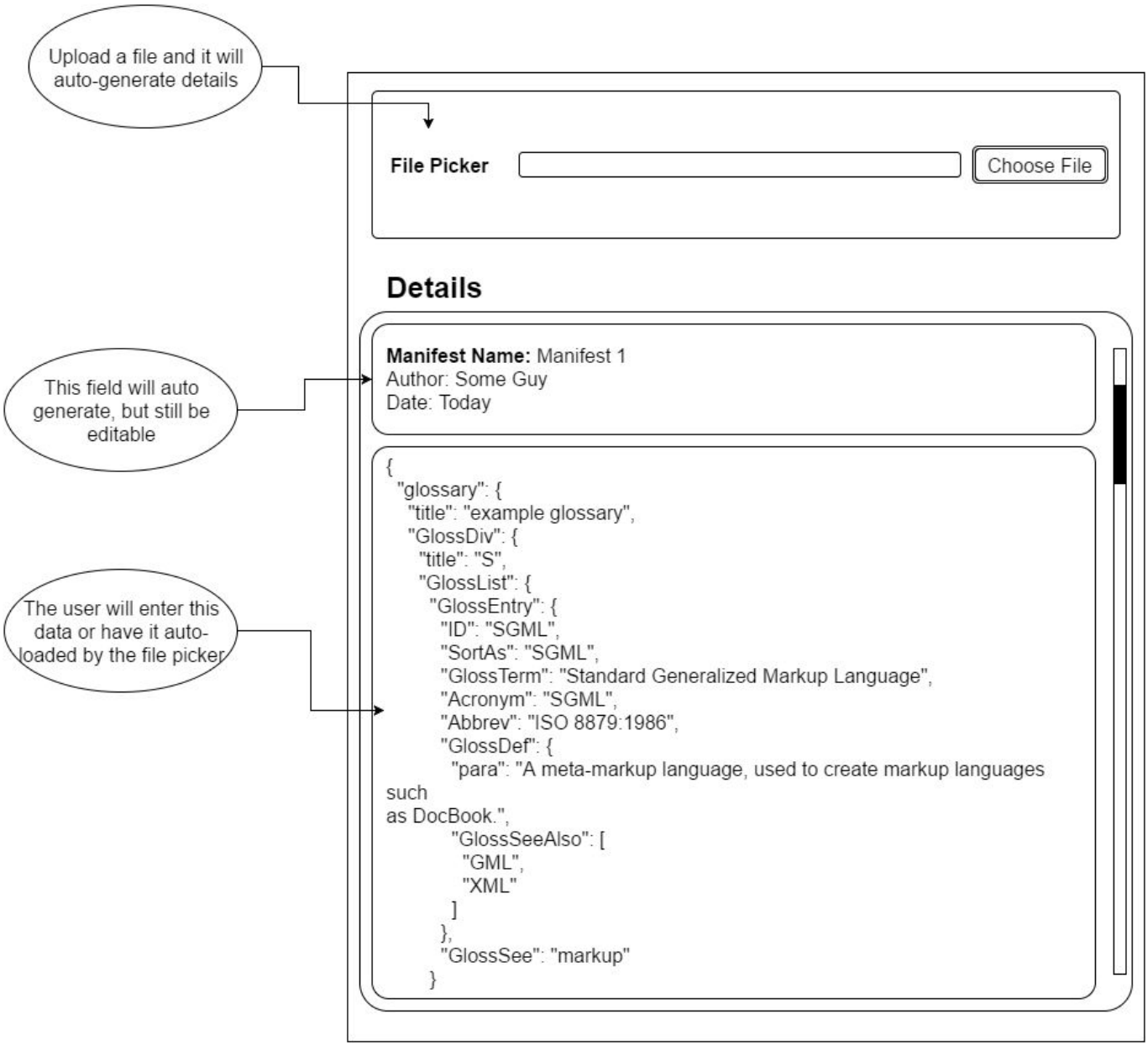
Summary: This page is to locate the manifest using a search function or by brute force and scrolling through the entire list. This will link or do all of the functionality of this page.



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

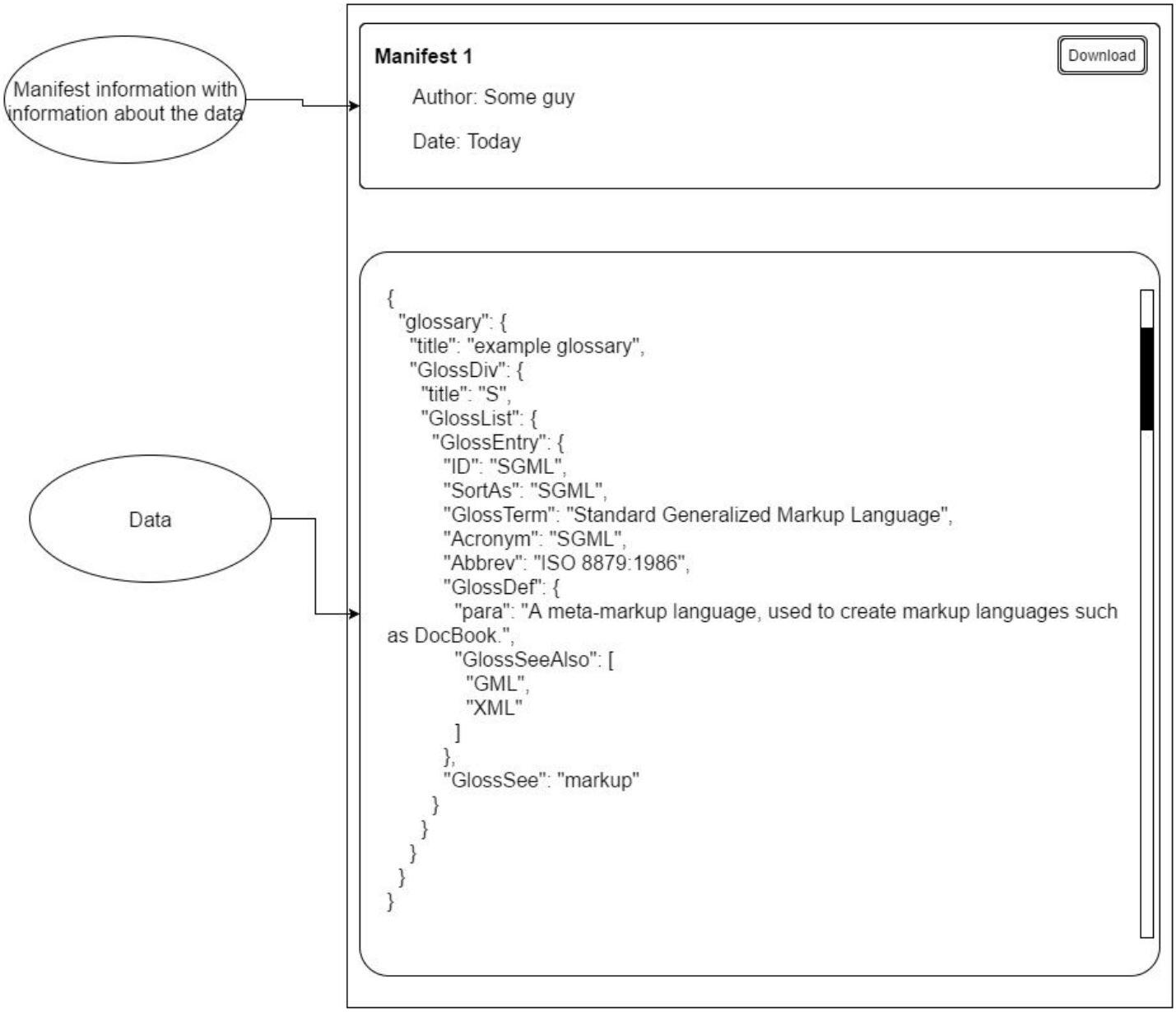
Summary: To upload, the user will choose a file and that will update the details section. If they want to create, all of the fields will be editable and the user can write in the information they want.



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.

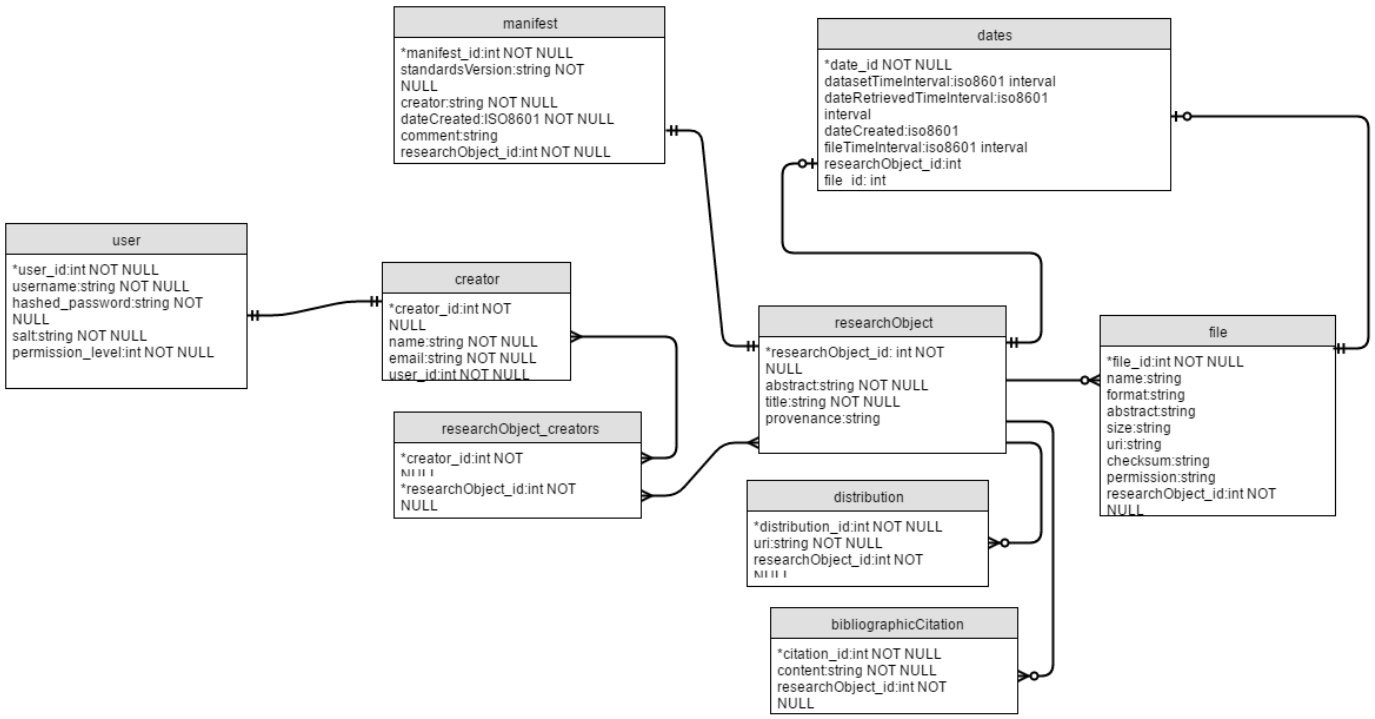
Table List

- 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.

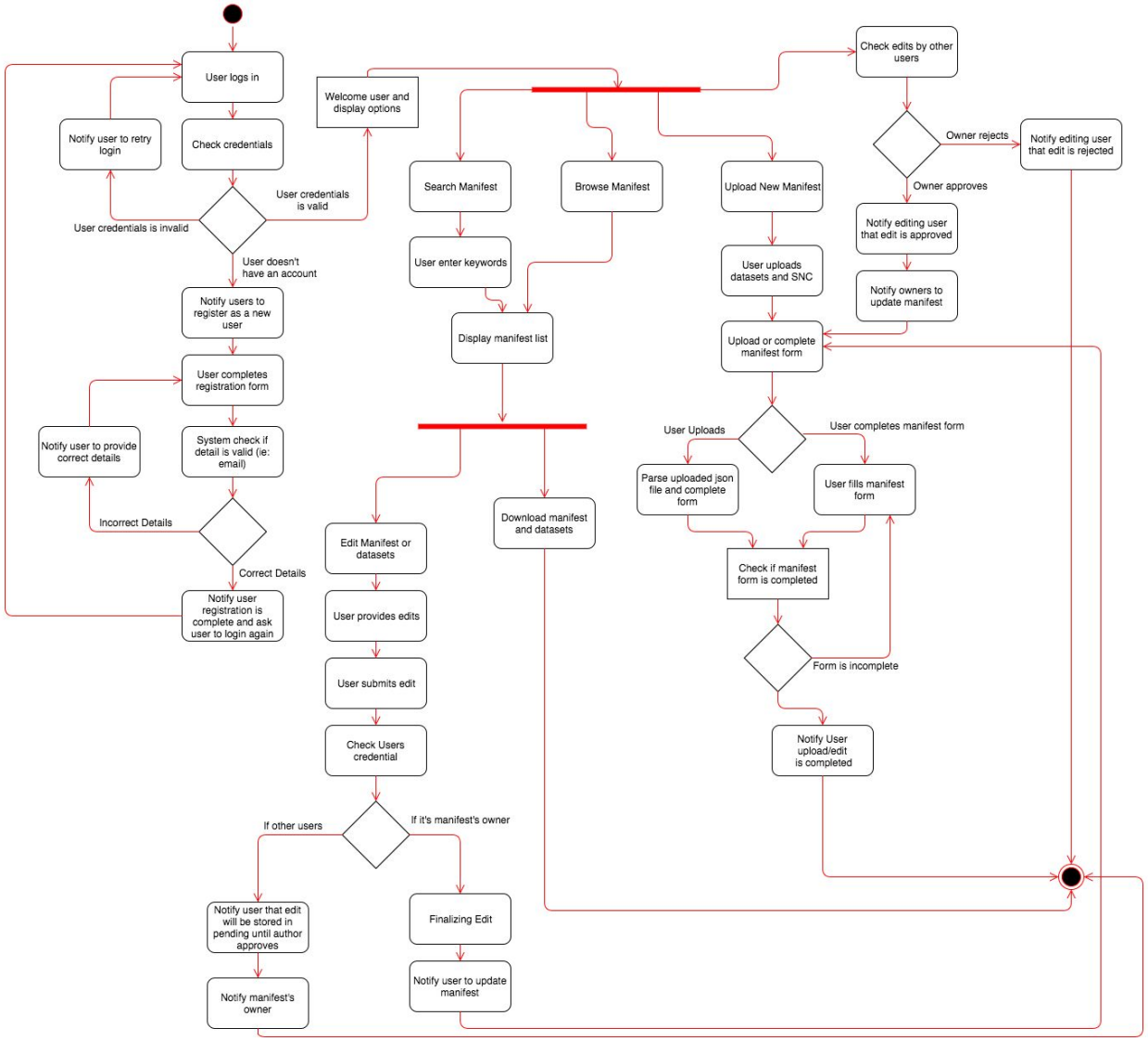
- User
    - userID:string, password:string, permissionLevel:int
- Researcher
    - Manifest:manifest
    - PermissionLevel:int = 2 (level two permission - edit and upload access
    - Relation: Researcher is a User (one to one), researcher will have one or more than one manifests attributed to him/her.
- 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

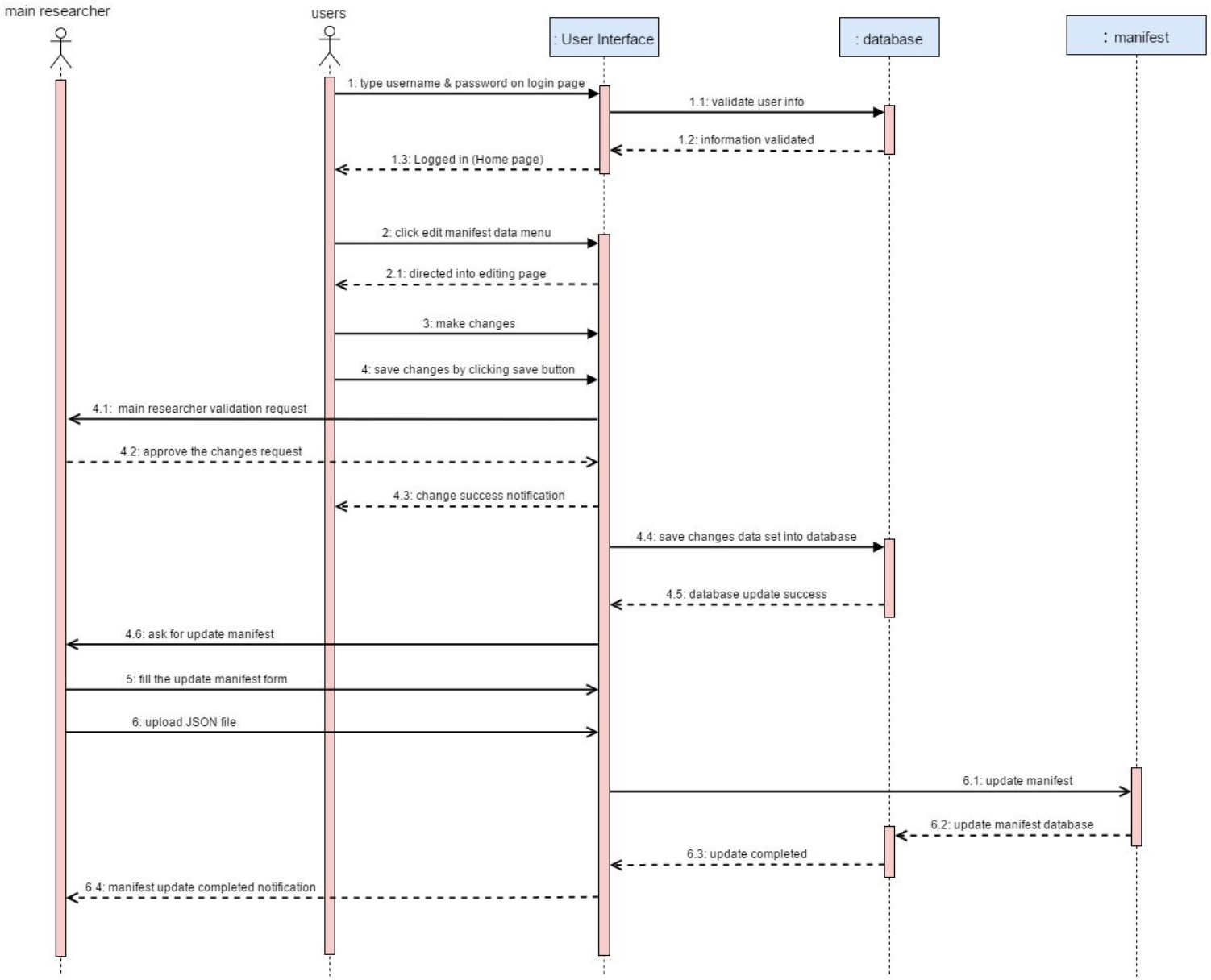
Larger image available here



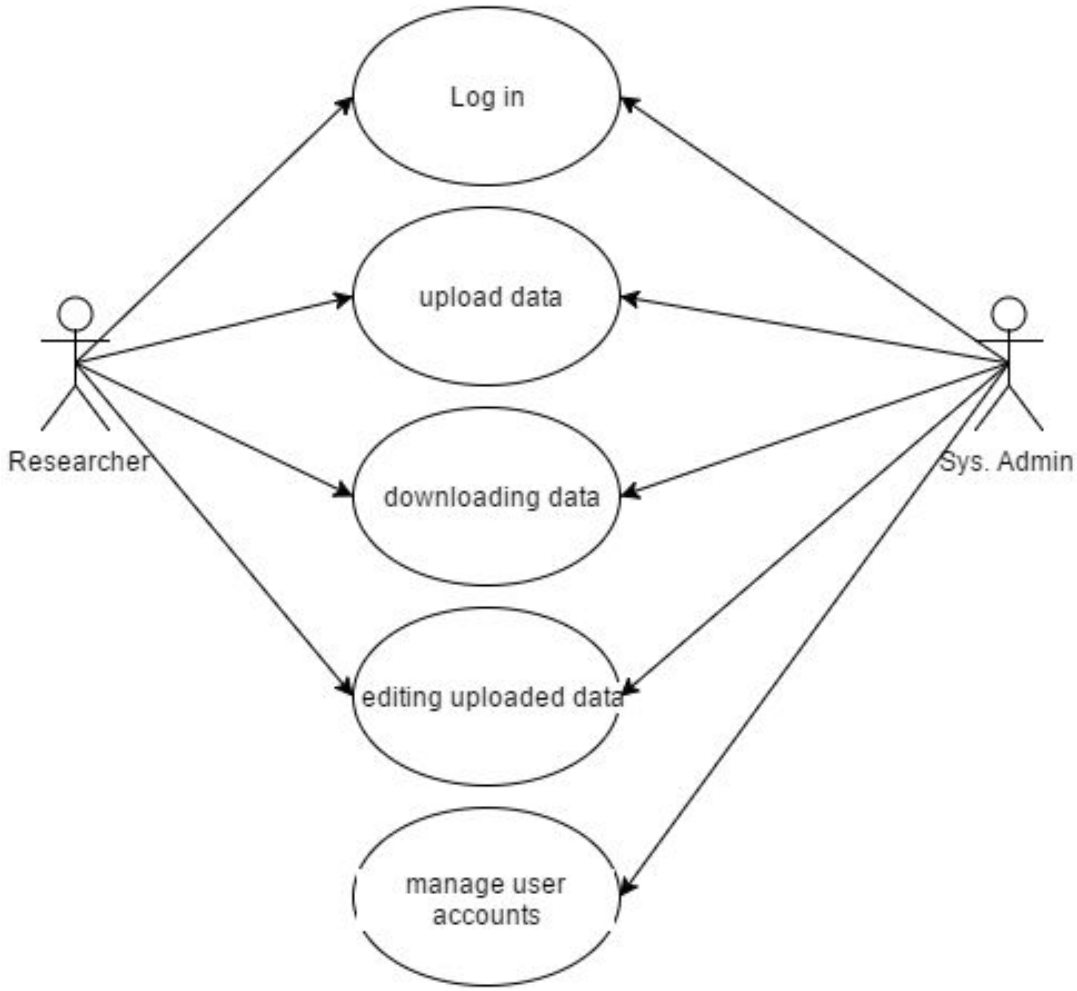
General System Wide Activity Diagram



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

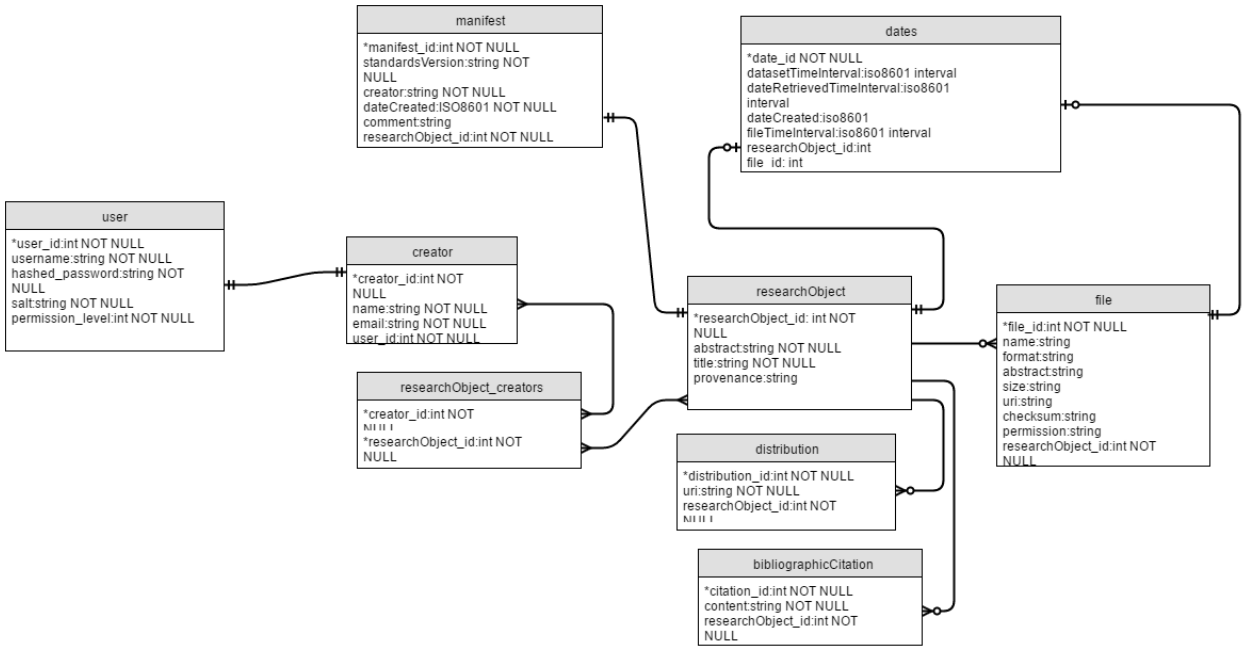
4. Testing and Documentation
    * Unit Testing - Kate, Assia
    * Regression Testing - Kate, Assia
    * Integration Testing - Kate, Assia
    *User Acceptance Testing - Kate, Assia

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.
- Out 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 loin and logout.

Two dummy data are created in each collection.

>monggo

```
//create the database name SWE
>use SWE

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

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

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

User Interface - Alex

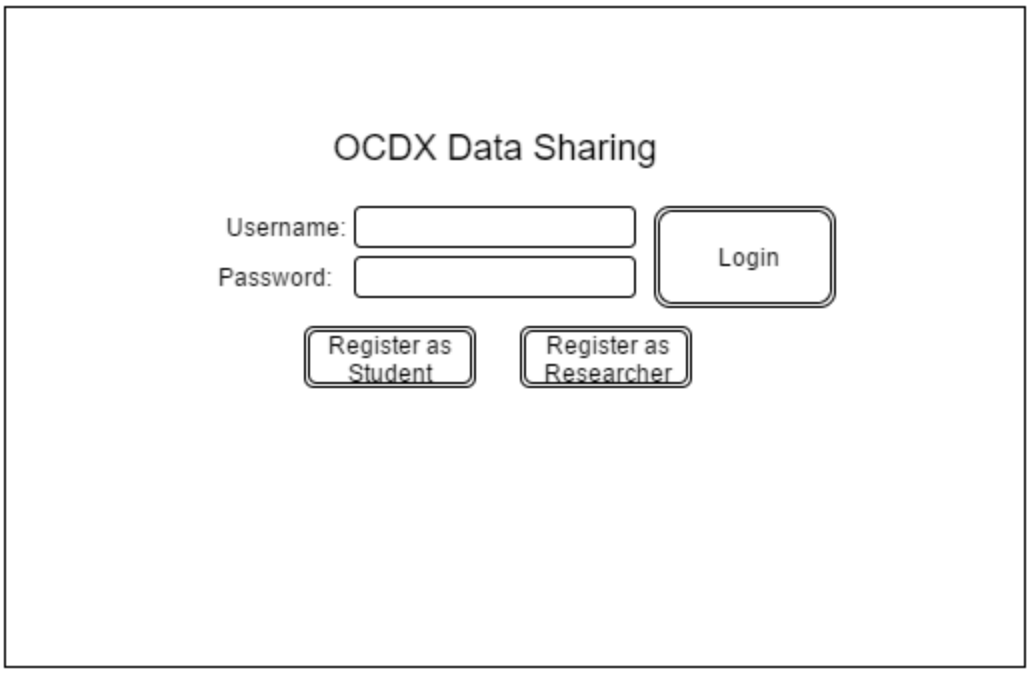
The complete system was simplified down into 3 pages. The search page, the view/edit page, and the create/update page. The slash means that they are multi-purpose. In the search page, the user can search through the list using a manifest search and a keyword search. The manifest search allows for you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

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.



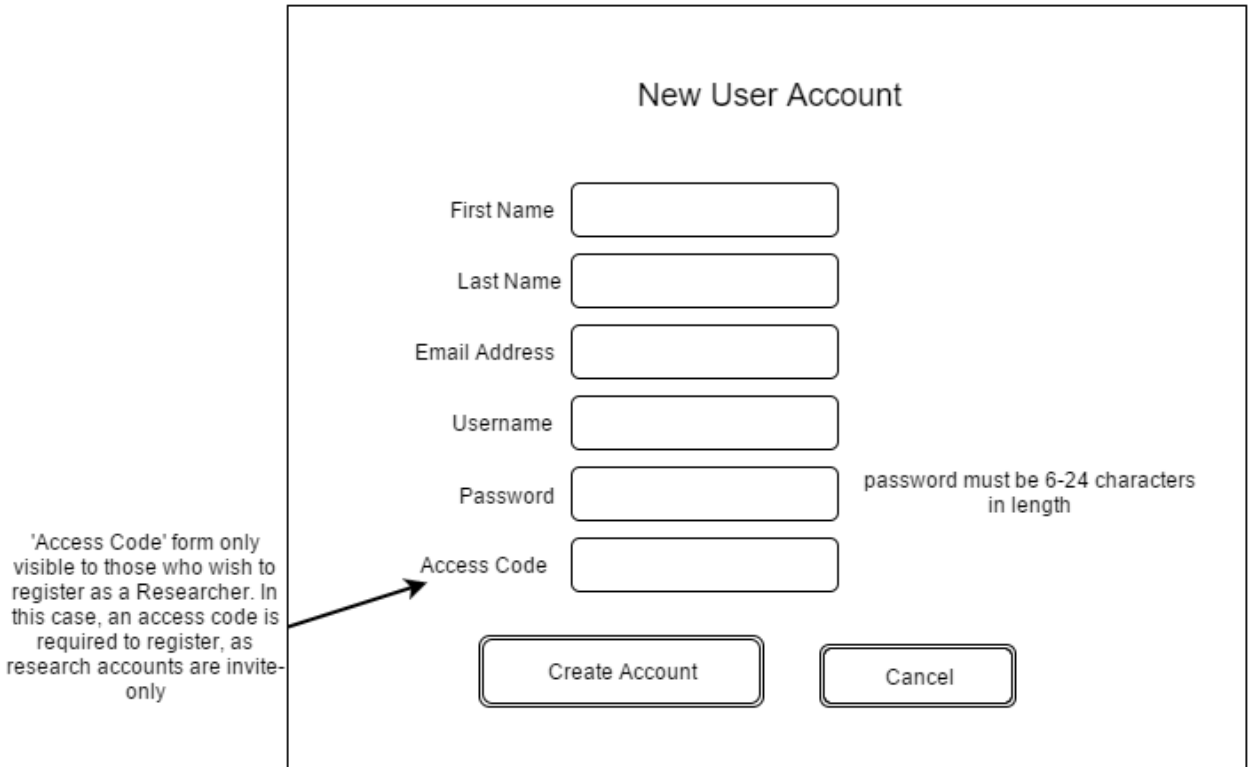
Button: Login: Sends username and password info to the server via php POST. If user credentials are correct, they are taken to the Browse Manifest/Home page. If the username does not exist of the password is incorrect, appropriate feedback is displayed on the page.

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’.

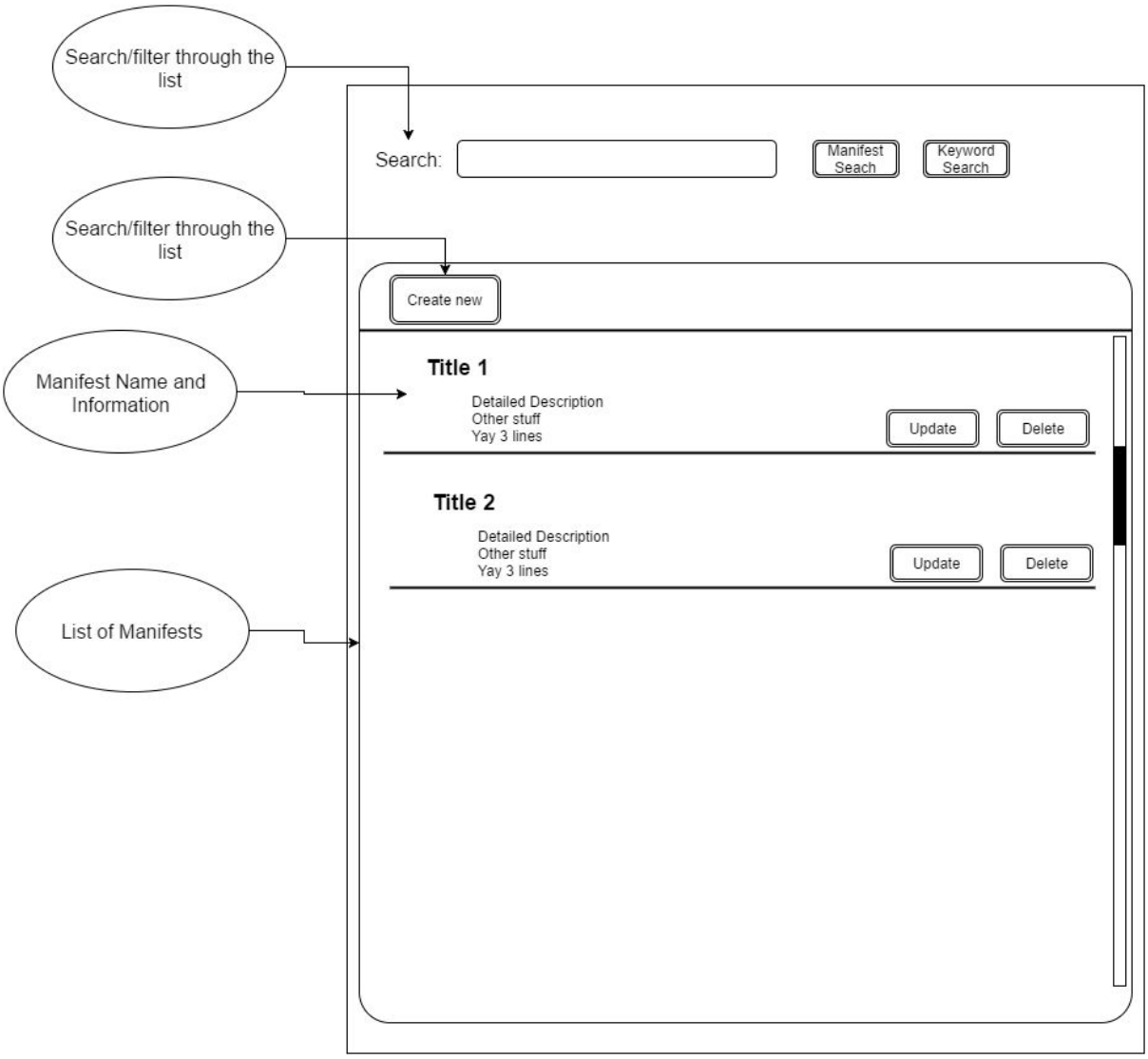


Button: Create Account: Attempts to send the user’s input into the user database as a new collection item. If any elements of the user’s input are invalid, the data is not sent and appropriate feedback is displayed by the offending form(s). If the input is valid, the data is added as a new collection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

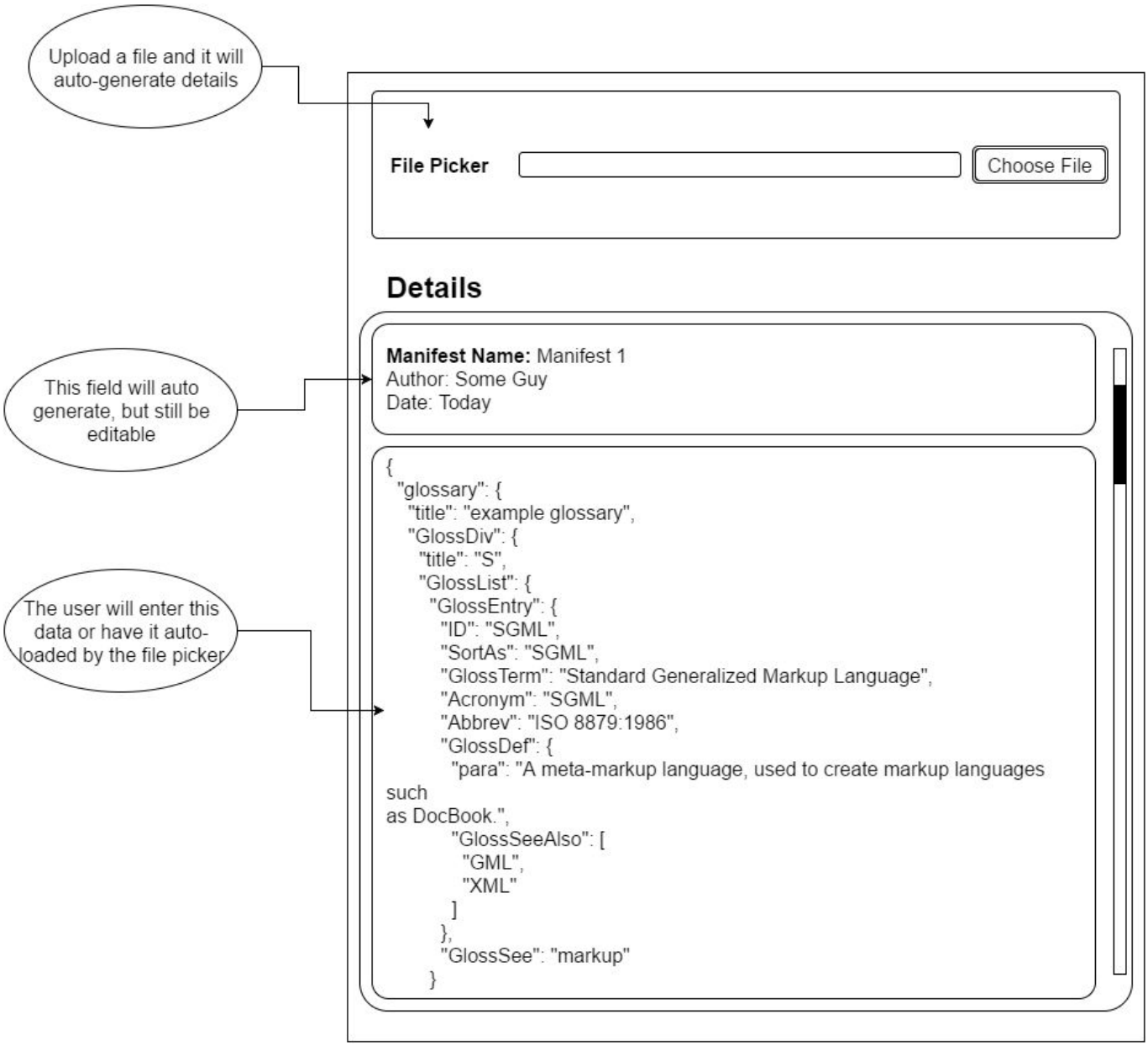


scrolling through the entire list. This will link or do all of the functionality of this page.

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

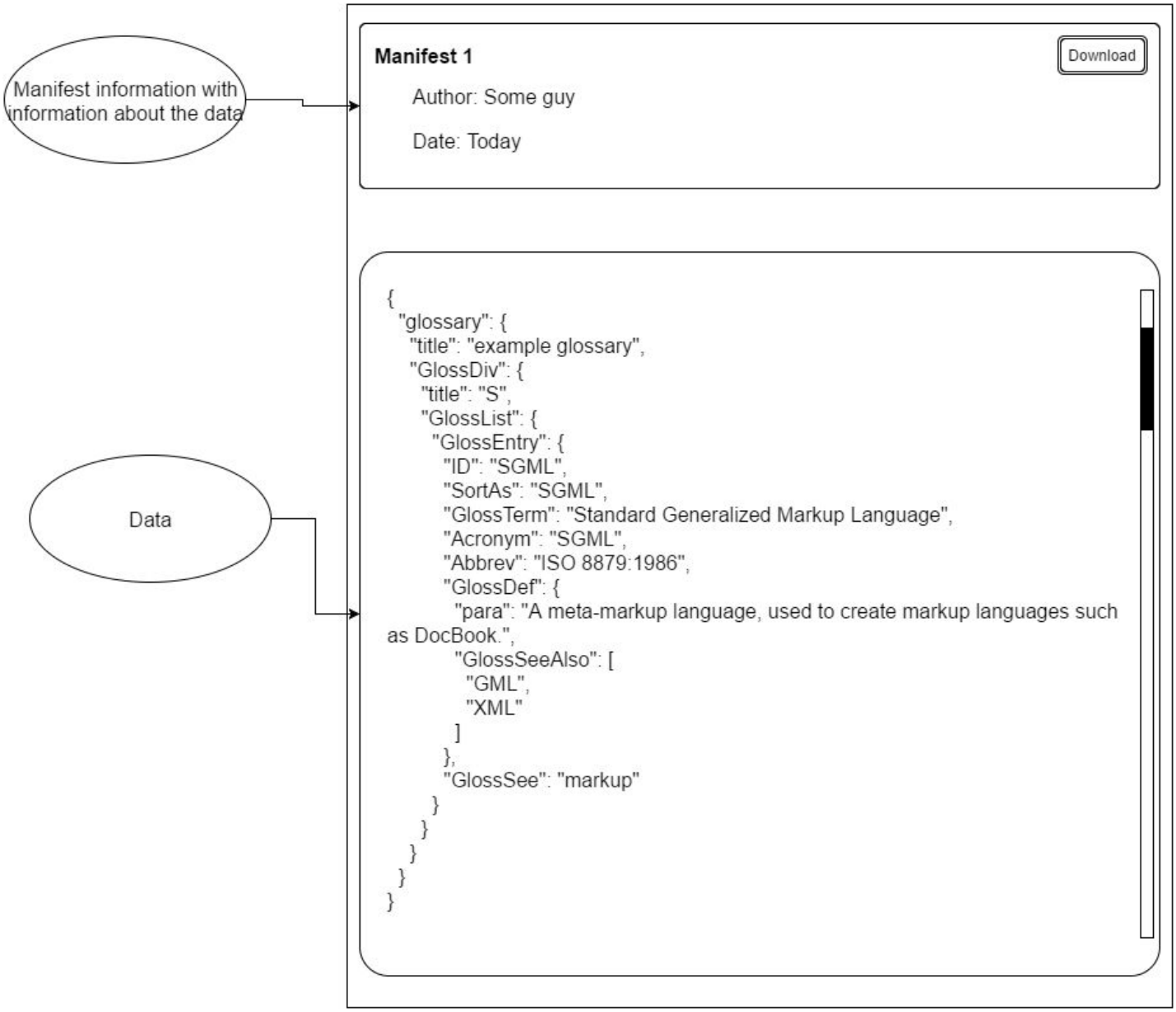
Summary: To upload, the user will choose a file and that will update the details section. If they want to create, all of the fields will be editable and the user can write in the information they want.



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

Meeting 2:

Date: 11/10/2016
Time: 2:00pm - 5pm
Attendance: Andy, Wei Xian, Dewi, Kate, Alex

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
    - Create CRUD queries for database implementation
    -
- 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:10.1177/14752409060558954325342"
        ],
        "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",
"permissions":"Open Database License 1.0",
"dates":
{
"dateCreated":"2015-10-11"
}

```
          }
        ]
      }
    }
  ]
}
```

Database Implementation

In sprint 1, we created the database, collections, and the dummy data in our MongoDB by the queries below:

//create the database name SWE

```
>use SWE
```

//create new manifest, this query actually does not necessary since mongo will generate its own collection if the we insert the data, even though the collection is not there yet

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

inserting the data for creating new manifest (either by dumping from the json file or by form manually input from the user). In this insertion, we explored many things on how to make the JSON file works on MongoDB. The result, we cannot just copy and paste the whole JSON file here. We need to adapt the JSON file based on the mongo environment to make it works, which we do not need the first braces and array name. If we do so, mongo will not understand what we are trying to insert.

//template

```
>db.manifest.insert(data inside the manifest.json)
```

//what we actually did

```
>db.manifest.insert([{
        "standardsVersion": "v0.0.1",
        "id":"1111201601",
        "creator":"Sean",
        "dateCreated":"11/11/2016", …….
         }])
```

When mongo read the file right, they will notice us with

```
BulkWriteResult({
    "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"
        })
```

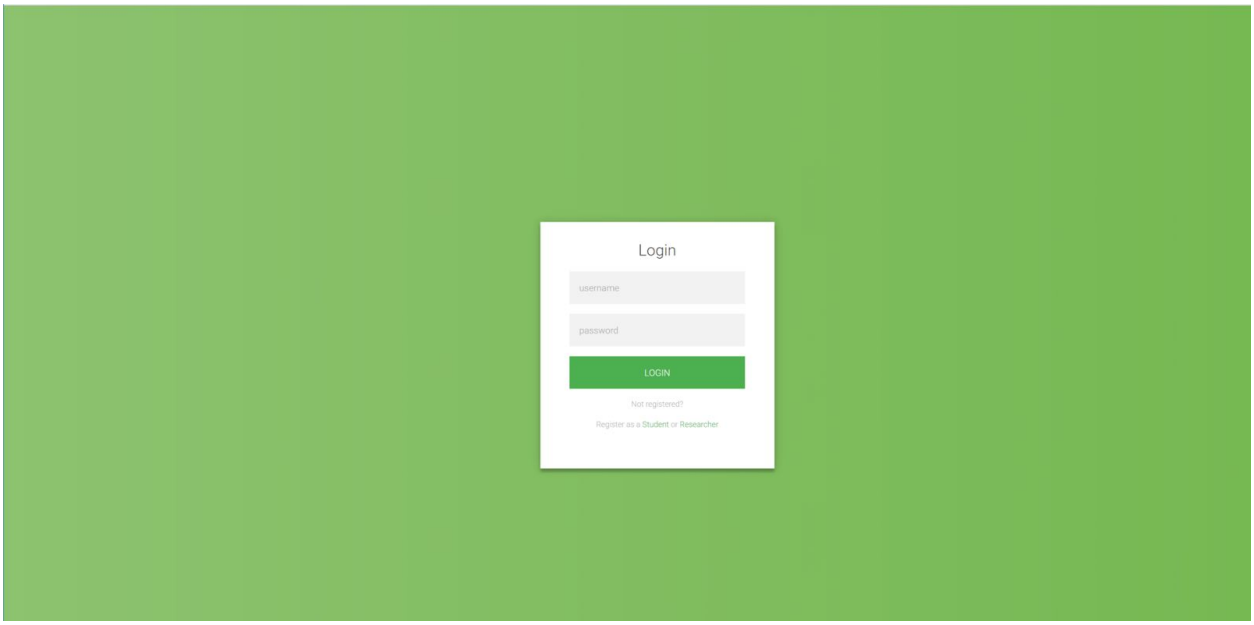
Hashed password will be automatically generated by the php script in the registration page. Thus, we ensure the security of the user's account.

```
//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()
```

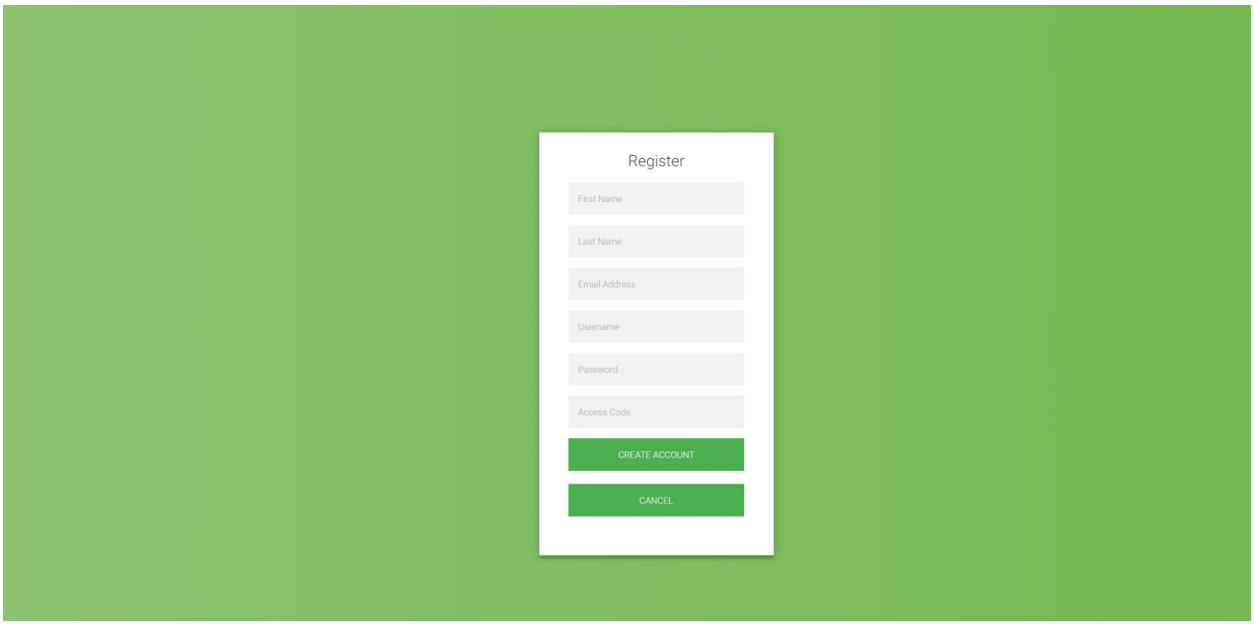
```
//update in the user collection, in this case we implement the update in permission level.
        >db.user.update({"key": "value here"} ,{$set:{"key": "new value here"}})
//our actual query
        >db.user.update({"permission_level": "is_creator"} ,{$set:{"permission_level":
"is_student"}})

//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

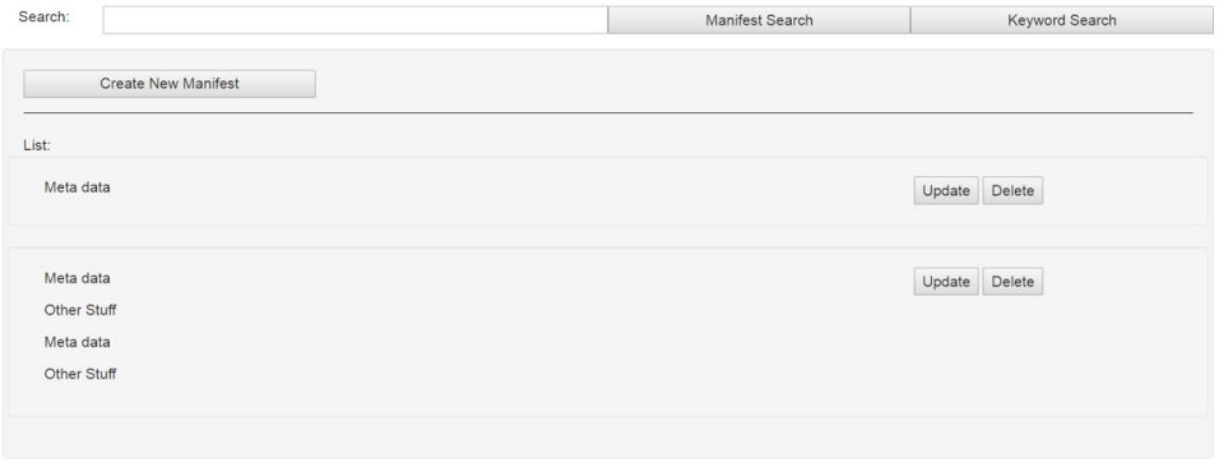
Pages



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

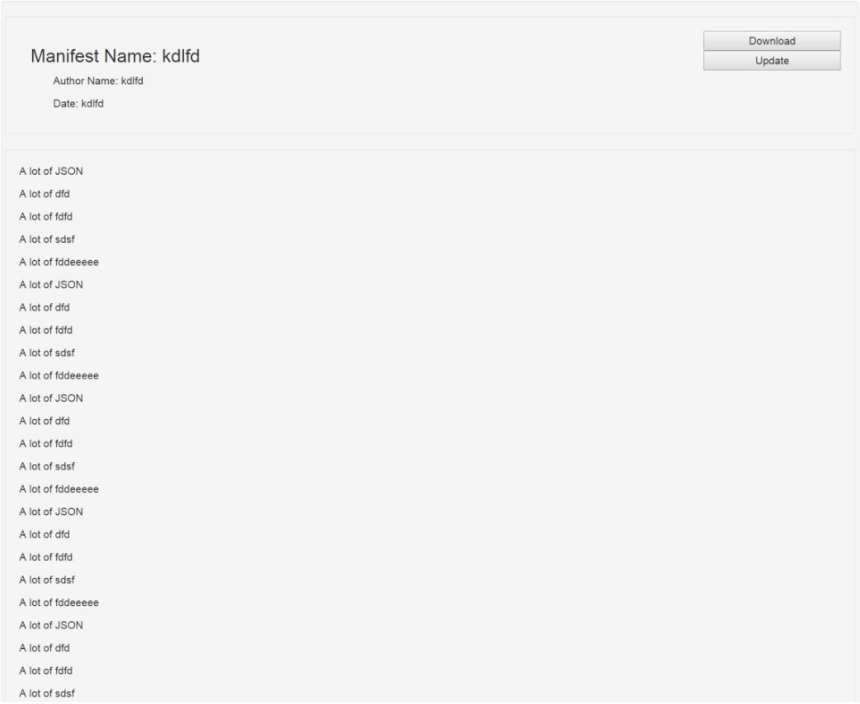


Above is the register page so that users can make new accounts. Access code will disappear if registering as a student



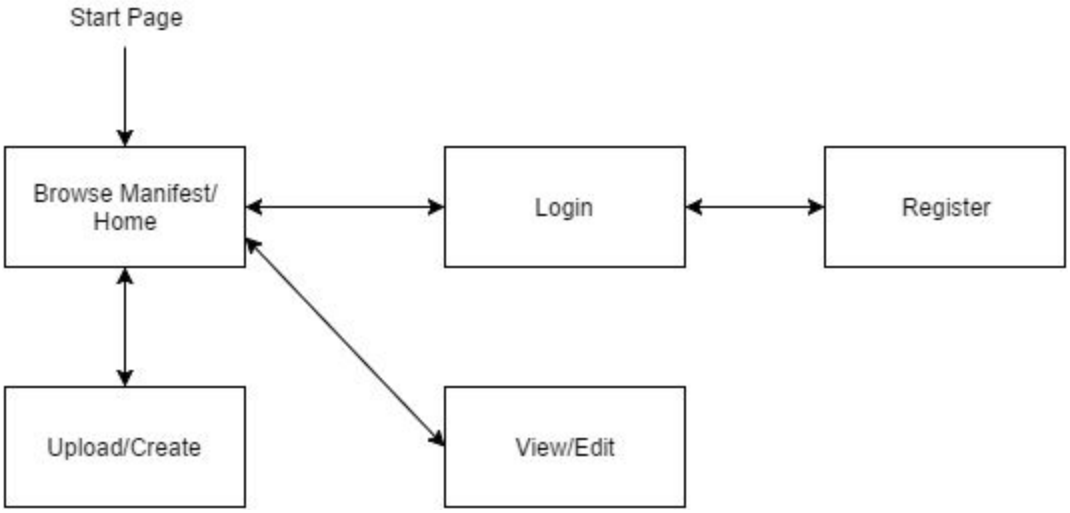
Above is the Browse manifest where the user can search through a list of manifests and view/update/delete it.

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



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

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.

Reader/Student: Permission level 0

Permission level 0 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 0, which will prevent them from doing any of the forbidden tasks mentioned above.

Researcher: Permission level 1

Permission level 1 means the user has all permissions provided at level 0, 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 1 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 1. 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 1.

System Admin: Permission level 2

Permission level 2 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 2 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.

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.

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.