MetaDrop

Andrea McGovern, Teddy Ivanov,

Olivia Apperson, Soya Ouk, Xinyi Li

Final Version

12/14/16

Table of Contents

Project Overview

As a team we are going to build an online infrastructure for computational social scientists, social scientist, and citizens that will facilitate and centralize our understanding of online human interaction. We will create a database that will allow metadata specification to store, share, describe, and analyze data sets. This will help solve the problem of not being able to easily access the metadata, and now scientist and citizens can easily access it via the internet.

Necessary Links

Website
http://ec2-35-163-197-29.us-west-2.compute.amazonaws.com/home.html
Repo
https://github.com/TeddyIvanov/SoftwareEngineering-Group3
Markdown (Including Revisions From Sprint 4**[See bottom of markdown for Search Functionality]**):

https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/README.md
Test Cases:
https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/finaltestdocument%20(1).pdf
User Manual:
https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/METADROP-USER-MANUAL.docx%20(1).pdf
Temporal Logic:
https://github.com/TeddyIvanov/SoftwareEngineeringGroup3/blob/master/Sprint_3/TemporalLogicOfUI.pdf

Deployment Instructions:
https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/Deployment.pdf

Requirements Analysis

Actors (Created by: Teddy Ivanov, Reviewed by: Andrea McGovern)**:**

Researcher/User:

- Computational Social Scientist
- Social Scientist
- Citizens

Admin:

- System Administrator

Activities (Created by: Andrea McGovern, Reviewed by: Olivia Apperson):

- Logging in and out of the system
- Uploading metadata files to be stored in the database
- Ability to search and filter through libraries of files
- Ability to download files for use
- Be able to edit metadata files

Use Cases (Created by: Olivia Apperson, Reviewed by: Xinyi Li)**:**

- A Computational Social Scientist, Social Scientist, and Citizens have the ability to...
  - Log in and out of the system
  - Upload a metadata file
  - Edit a metadata file
  - Delete a metadata file
  - Search through metadata files

- Download a file

- A System Admin has the ability to…
  - Log in and out of the system
  - Delete uploaded files/edits
  - Search through files
  - Edit metadata files
  - Add users to the system

User Requirements (Created by: Soya Ouk, Reviewed by: Xinyi Li)**:**

- The System Administrator, Computational Social Scientist, Social Scientist as well as the Citizens should have login credentials.
- The System Administrator and Researchers should each have their own correct permissions.
- The Researchers must submit the correct and compatible file type.
- Admin can upload files, edit files, delete files, browse for files, version control, and add new users to the system.
- Regular users can upload files, edit files, delete files, and browse through the files.

System Requirements (Created by: Olivia Apperson, Reviewed by: Teddy Ivanov)**:**

- Database
  - Used to store information on metadata files and user's credentials.
- Webpage
  - User interface that visualizes the database of metadata and allow user to see what abilities they are performing.
- Internet Access

- Access to the internet will allow the users to visit the webpage, also allow updates to the database.

- Computer
  - Platform in which the webpage can be accessed. Also can be used to make changes to the database for admins.

Functional Requirements (Created by: Xinyi Li, Reviewed by: Soya Ouk)**:**

- Be able to login and logout
- Be able to upload files
- Be able to search in a search bar for JSON files
- Be able to edit files in the browser
- The system should not crash when a user tries to upload a file or edits a file
- If a file is not of the correct types, it should not be uploaded
- If a user does not have the correct credentials or user roles then they should not be able to do those tasks
- Be able to remove files
- Be able to fix changes

Non-Functional Requirements (Created by: Teddy Ivanov, Reviewed by: Soya Ouk)**:**

- This project is a semester project for the software engineering CS 4320 class, so it must be finished by the end of the semester
- The size of the database has to be large enough to hold JSON files (10 Gigabytes of files)

- Be able to handle uploading and downloading in real time (less than one second to upload/download)
- Easy to use User Interface that works on Firefox and Chrome
- Upload JSON files and script files (only acceptable file types)
- File extension checking to make sure it of the correct type (i.e. .JSON)
- Correct username of at least 8 characters and password with at least 8 characters
- If a file is larger than 250MB, then it should not be uploaded

Class List

Classes: (Created by Xinyi Li, Reviewed by Teddy Ivanov)

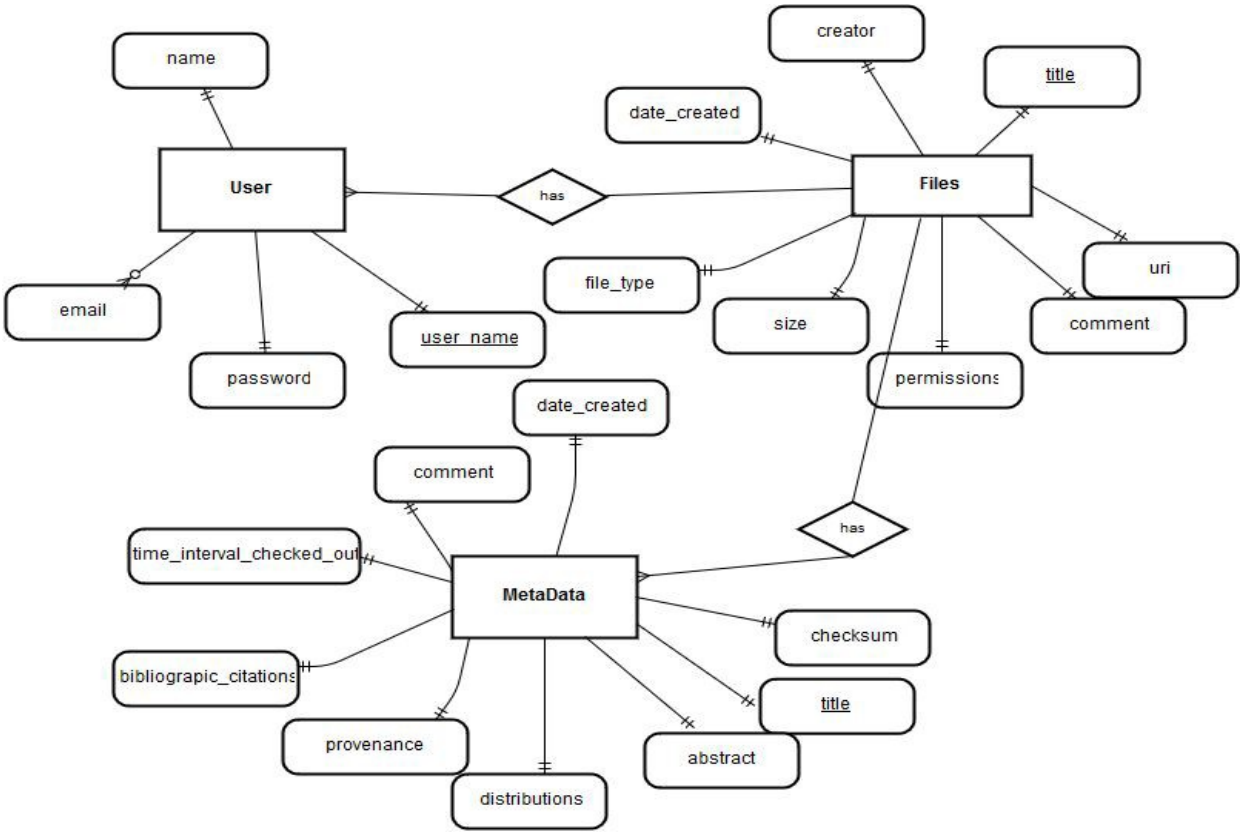
- Researcher/User
  - attributes: name:String, password:String, IsAdmin:Boolean,
  - methods: login(), logout(), uploadFile(), downloadFile(), deleteFile(), editFile()
- Admin
  - attributes: name: String, IsAdmin:Boolean,
  - methods: addUser(), uploadFile(), login(), logout(), downloadFile(), deleteFile(), revertEdit(), editFile(), updateDatabase()
- MetaDataFile
  - attributes: title:String, author:String, date:DateTime
  - methods: display(), delete(), edit(), upload()

Table List

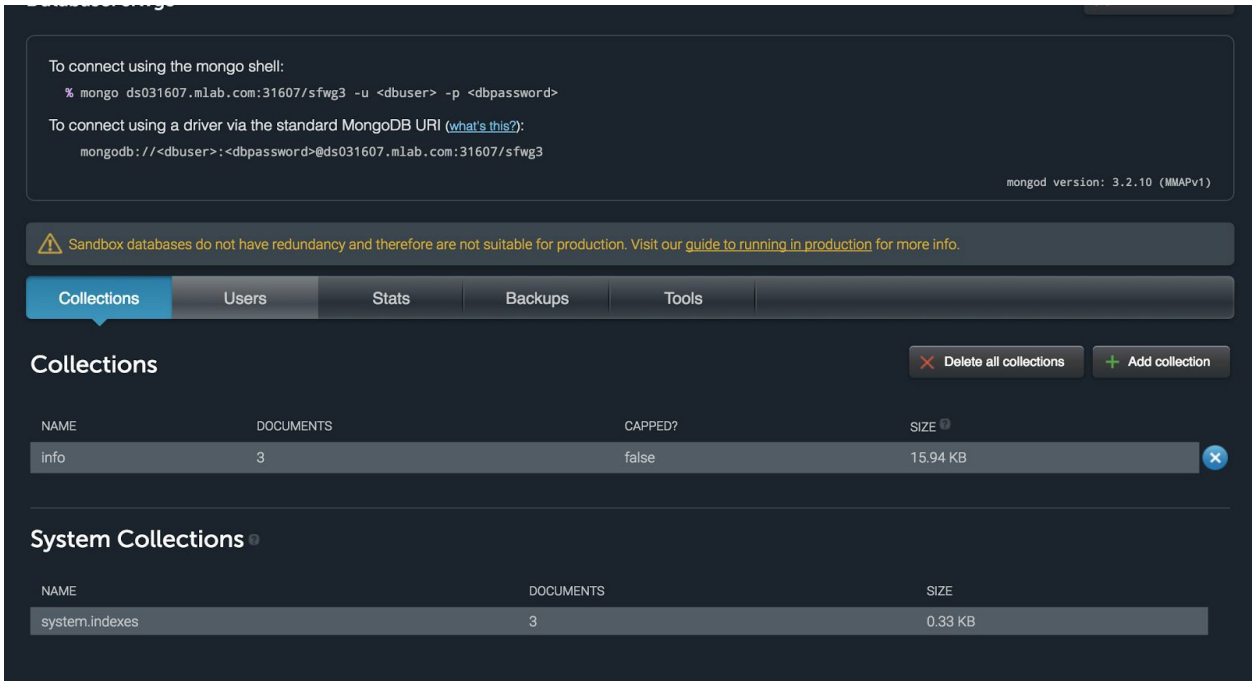
Created by: Olivia Apperson, Reviewed by: Soya Ouk

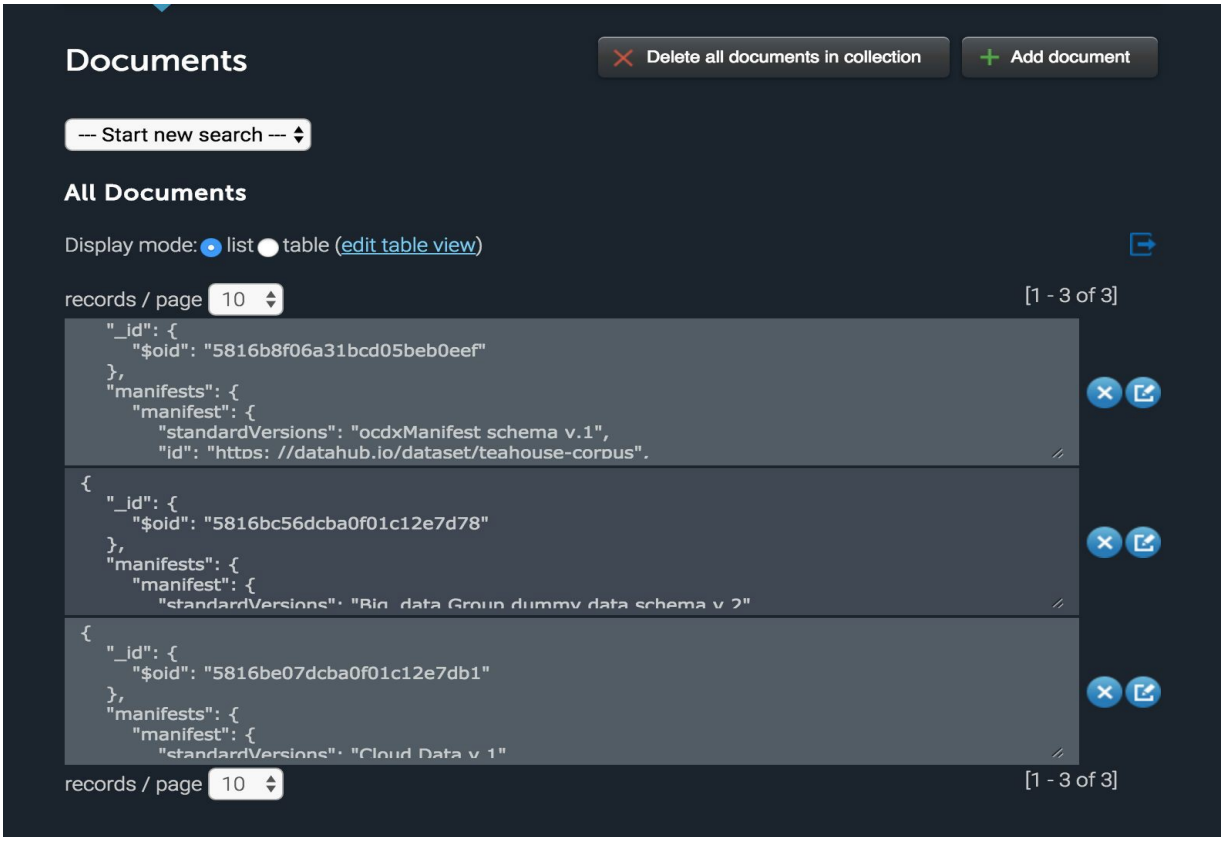
- Table for Files (type, data, name, author, etc.)
- Users (name, username, password, etc.)
- MetaData (comment, checksum, title, etc.)
- Relationship table between Files and Users

ERD

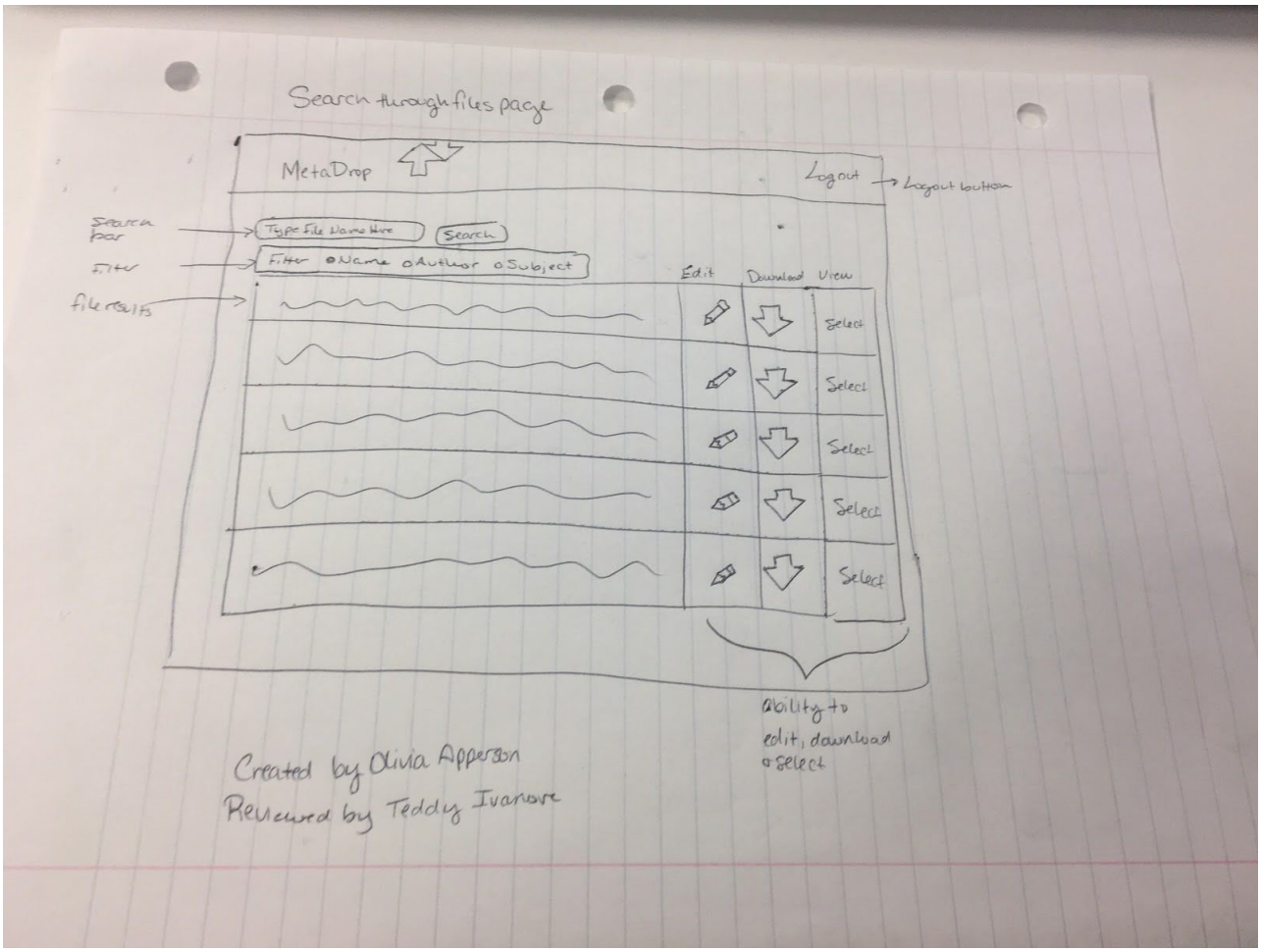

name
creator
title
date_created
User
has
Files
email
password
user_name
file_type
size
permissions
comment
uri
date_created
comment
has
time_interval_checked_out
MetaData
checksum
bibliograpic_citations
title
provenance
abstract
distributions

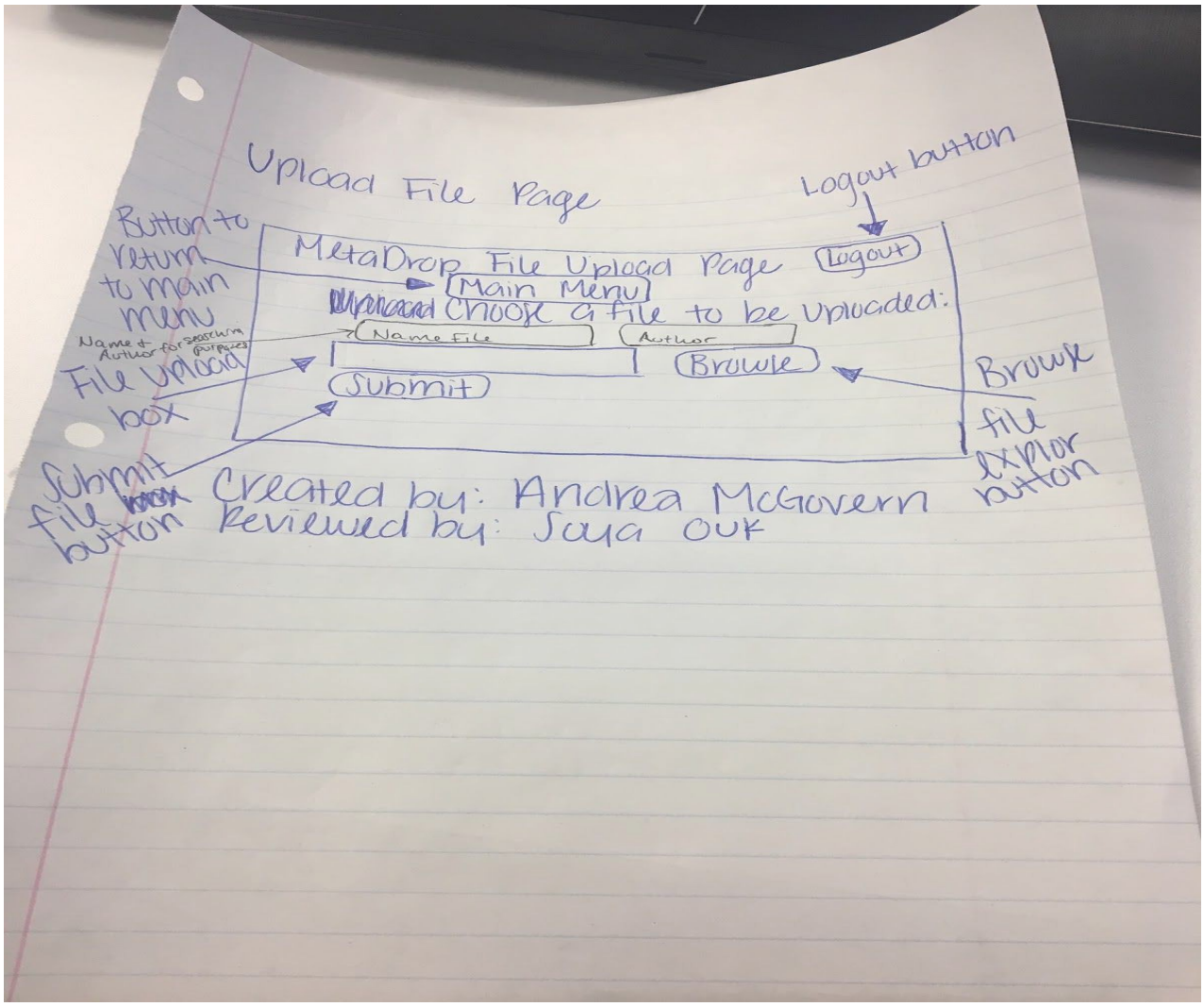
Database Creation/Queries

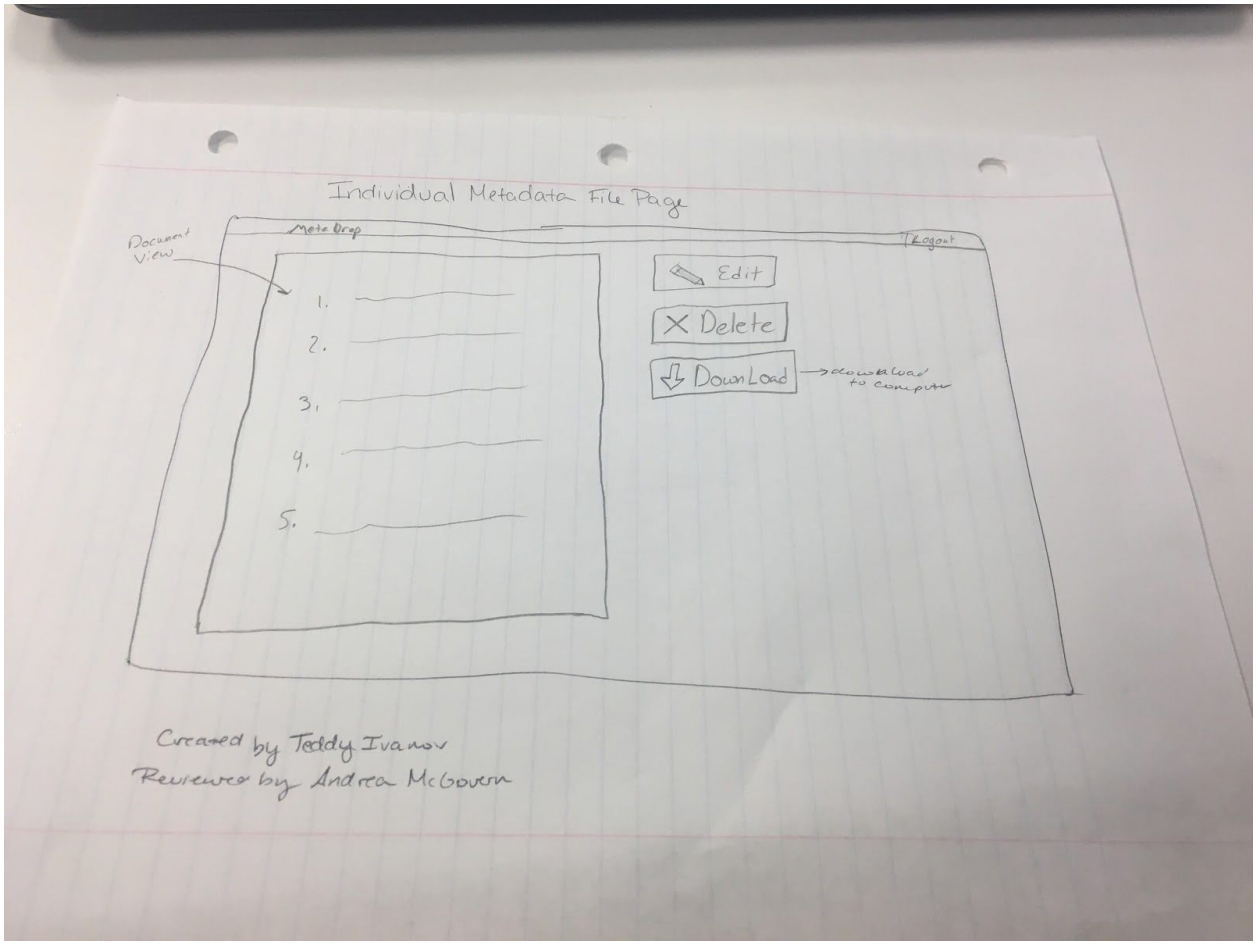


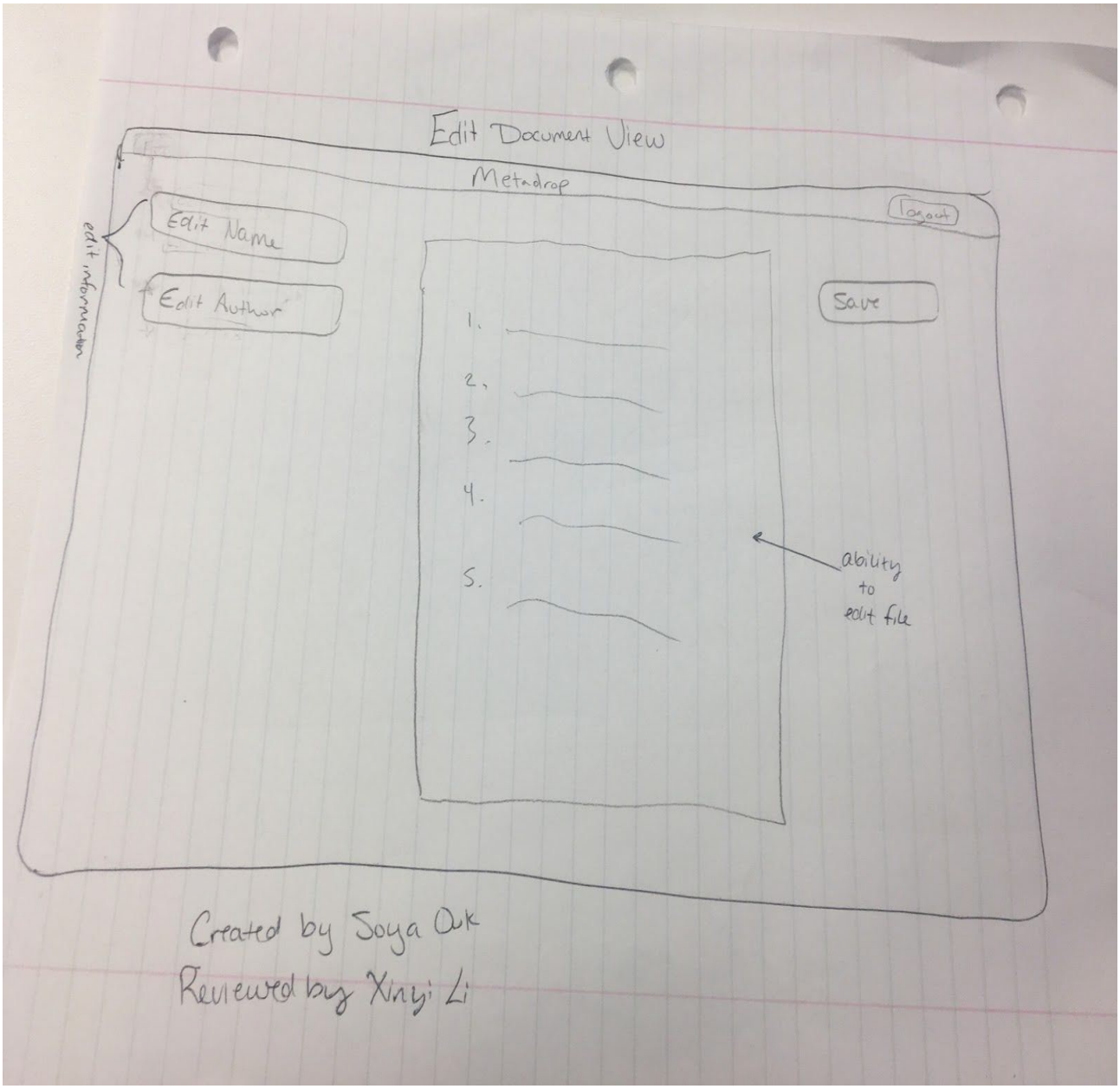


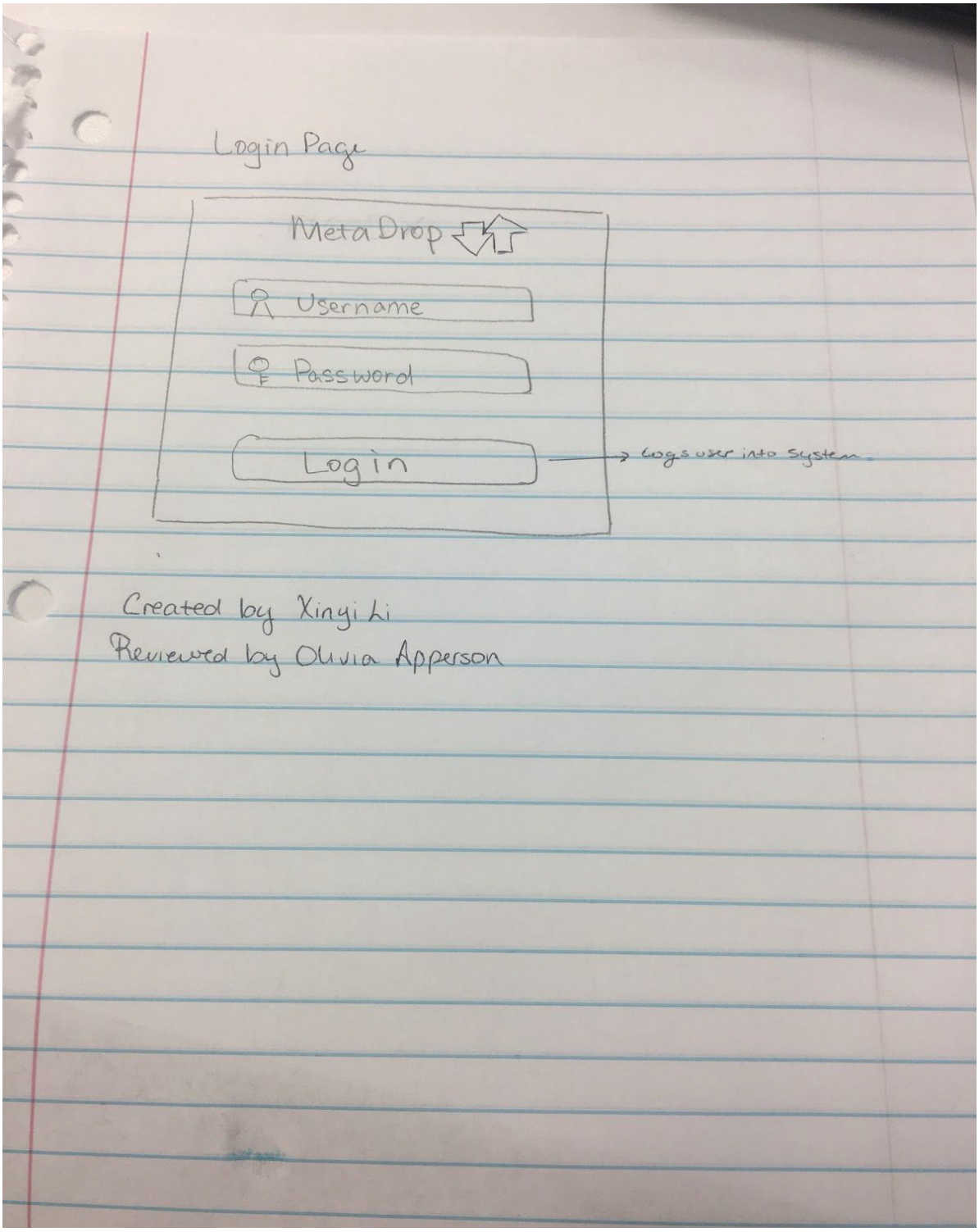
Screen Designs




Upload File Page
Logout button
Button to return to main menu
MetaDrop File Upload Page
Logout
Main Menu
Choose a file to be uploaded:
Name & Author for session purposes
File Upload box
Name File
Author
Browse
Submit
Browse file explor button
Submit file button
Created by: Andrea McGovern
Reviewed by: Saya Ouk


Individual Metadata File Page
Document View
Meta Drop
Logout
1.
2.
3.
4.
5.
Edit
Delete
DownLoad
Created by Teddy Ivanov
Reviewed by Andrea McGovern

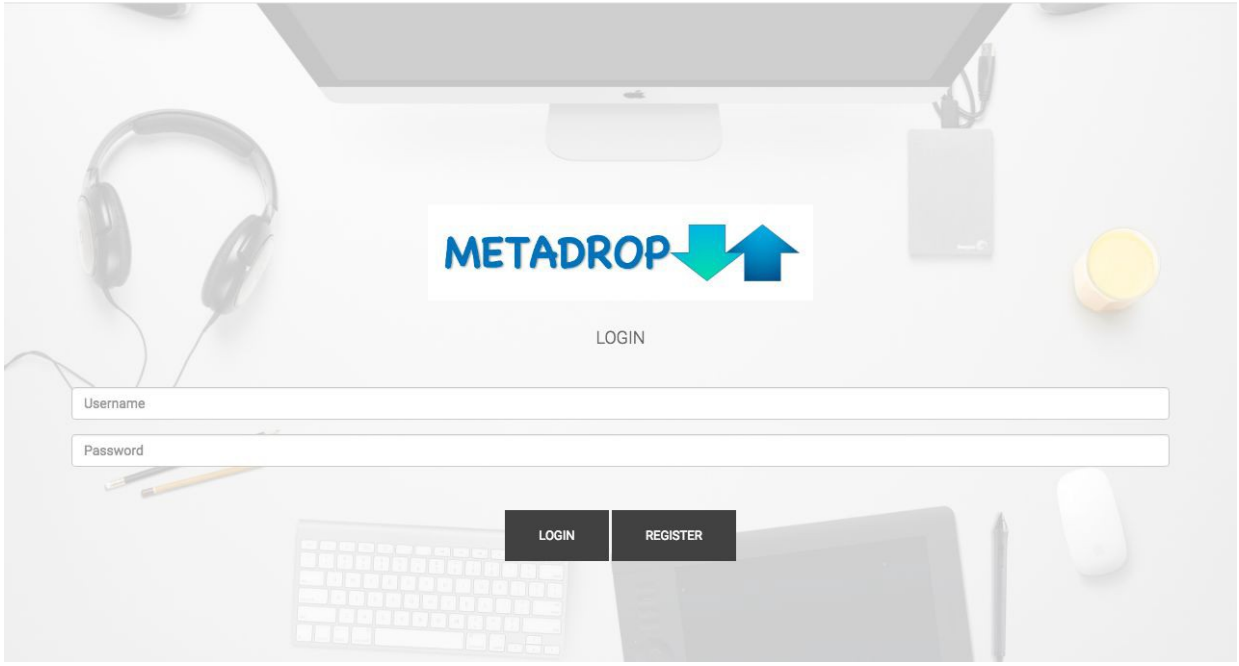

Edit Document View
Metadrop
logout
edit information
Edit Name
Edit Author
Save
1.
2.
3.
4.
5.
ability to edit file
Created by Soya Ouk
Reviewed by Xinyi Li


Login Page
MetaDrop
Username
Password
Login
Logs user into system.
Created by Xingyi Li
Reviewed by Olivia Apperson

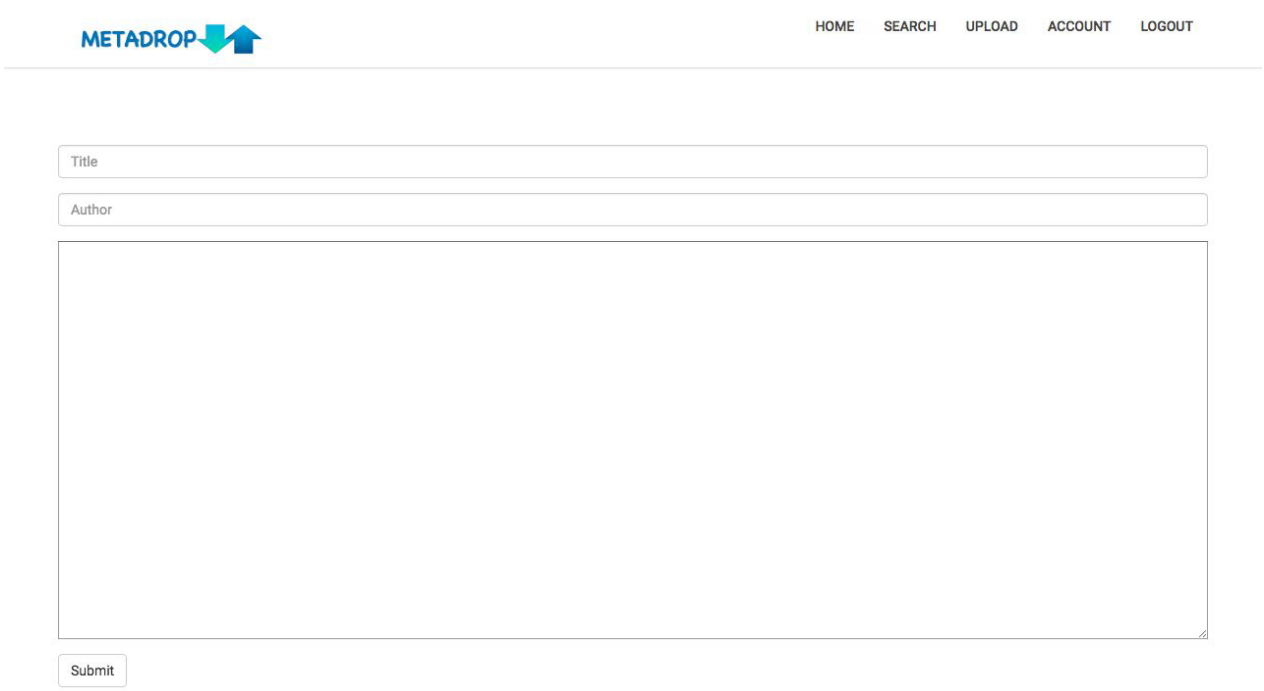
Updated User Interface

Website: http://ec2-35-160-238-84.us-west-2.compute.amazonaws.com/final_project/index.html

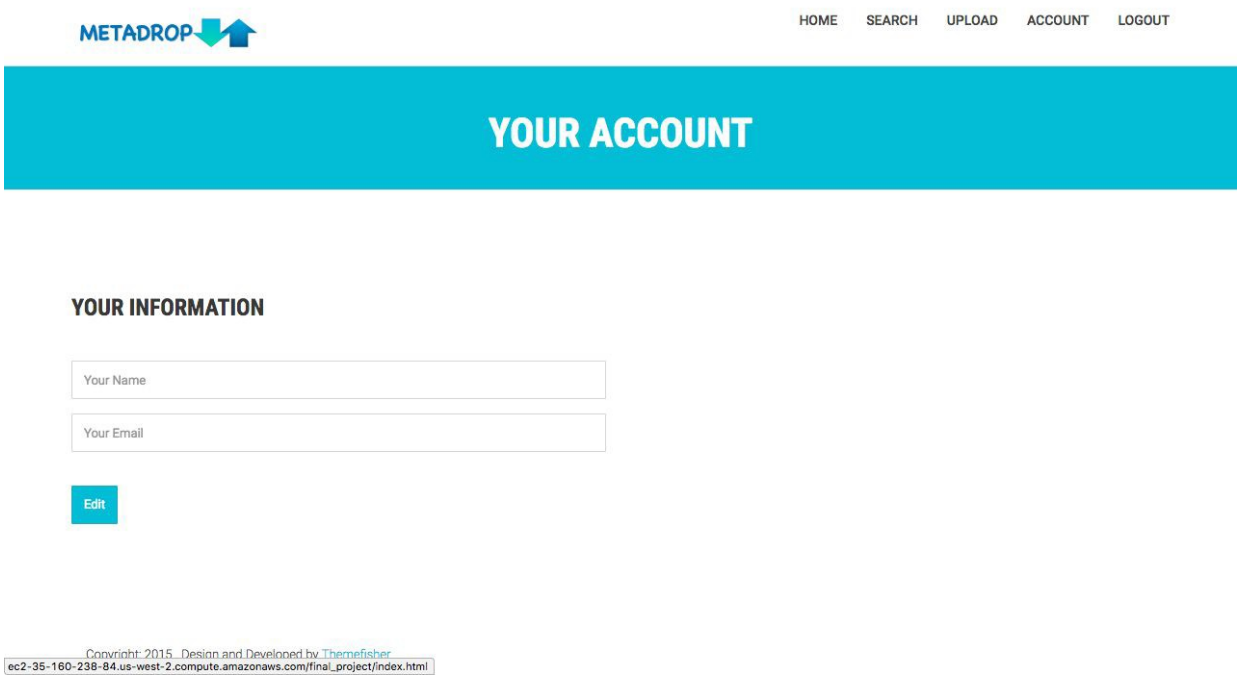
Login Page | Register Page



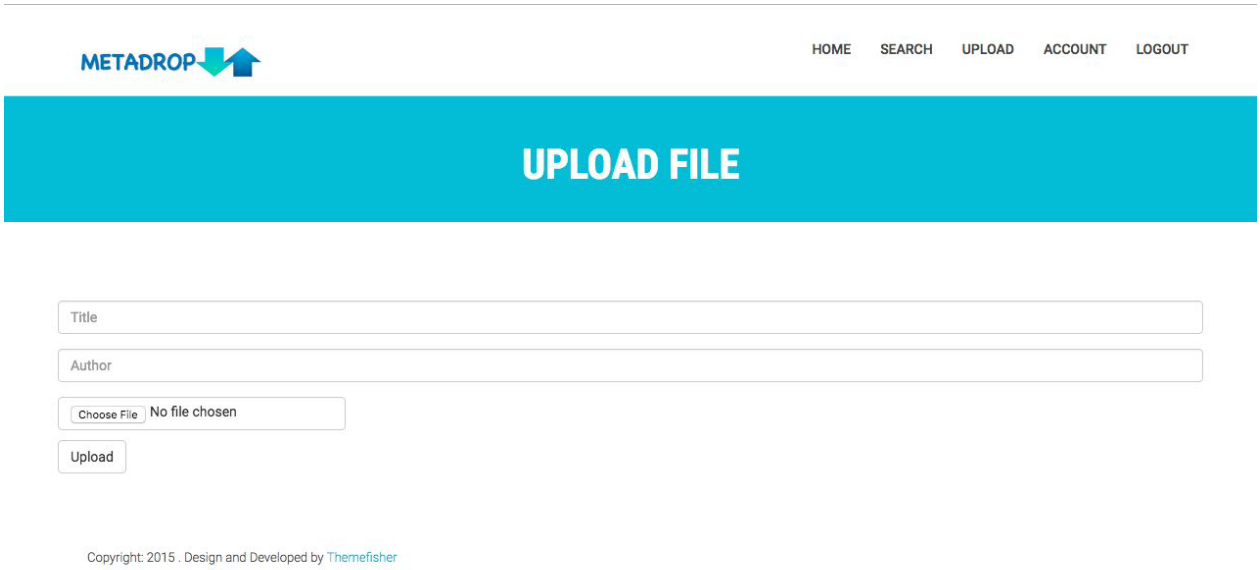
Modify JSON page



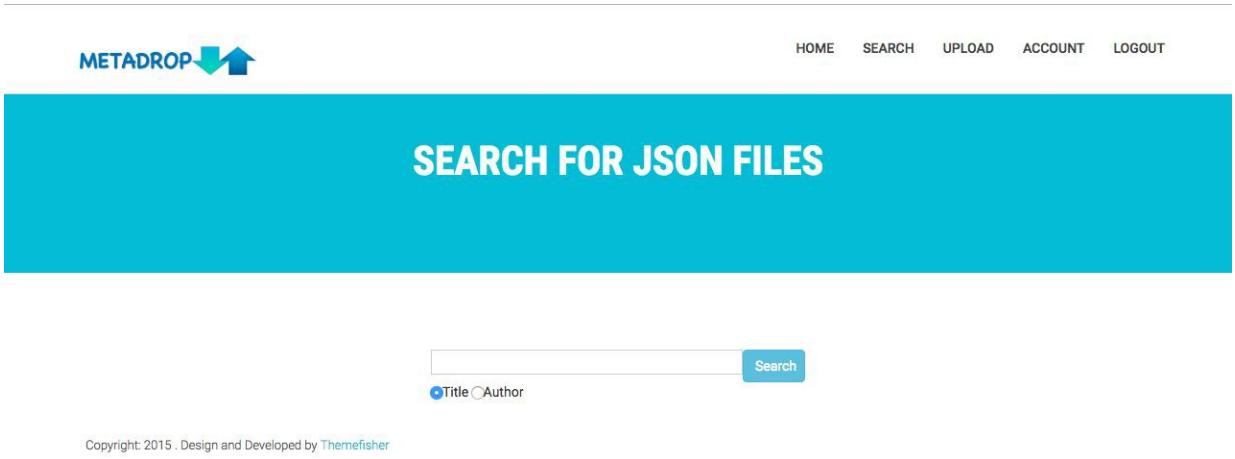
User Account Page



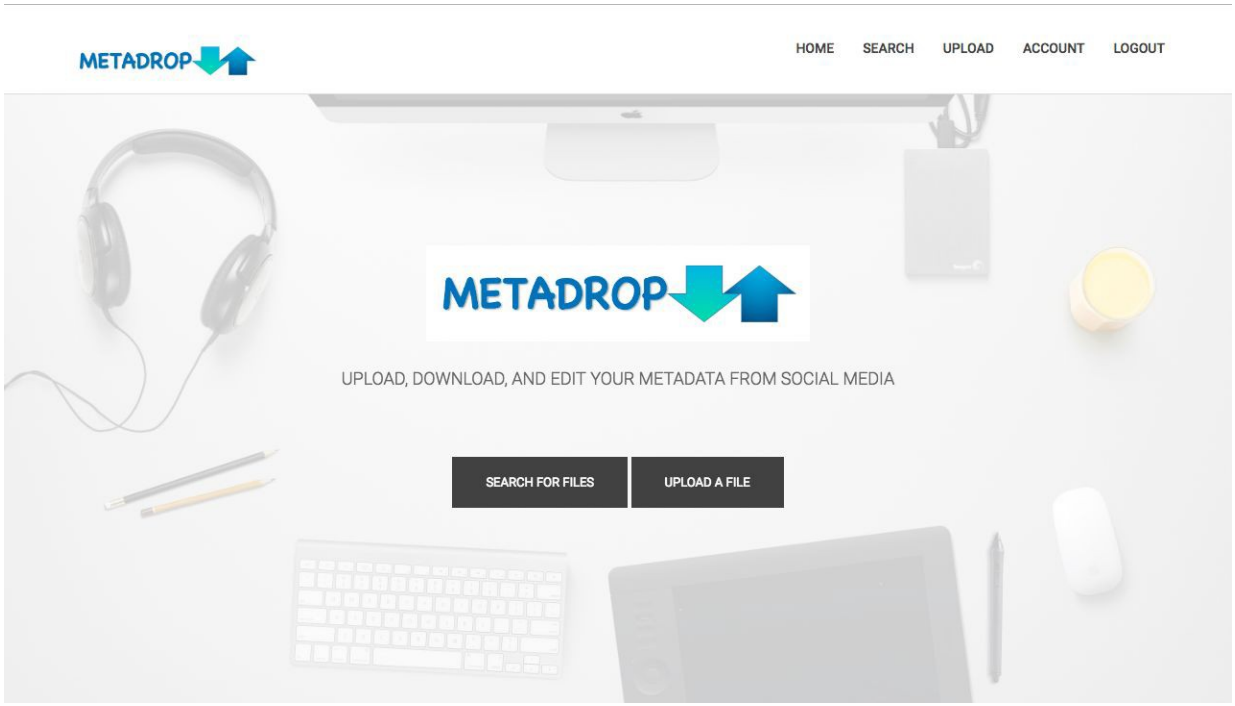
Upload File Page



Search for JSON Files Page



Home Page


METADROP
HOME
SEARCH
UPLOAD
ACCOUNT
LOGOUT
METADROP
UPLOAD, DOWNLOAD, AND EDIT YOUR METADATA FROM SOCIAL MEDIA
SEARCH FOR FILES
UPLOAD A FILE

Sprint 2

General

- Sprint Documentation: https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/xlv4c_Sprint3/Final%20Project%E2%80%94Metadrop-SPRINT-2.pdf
- Lead: Olivia Apperson Co: Xinyi Li
- Students worked to provide accurate and thorough documentation for Sprint 2.

Database

Leads: Teddy Ivanov and Andrea McGovern

- We worked together to create insert, delete, and update commands into the database and use in php scripts. These can be found in the DML folder above.
- We met on Monday from 4-6, Wednesday from 4-6:30, and Thursday from 4-8.
- We had to start over from scratch, and create a new ubuntu instance on AWS becasue php 7 wasnt compatible with Mongodb. It is also easier to debug on a linux server rather than a Mocrosoft one.
- We met with Jeremy during his office hours to set up the new service. He helped us download php 5, mongodb driver, and filezilla.
- We then continued working on our own to download Apache and figured out how to access the root folder with the correct permissions to host all of our files on the instance. Firebase allows you to store JSON files as well, but instead of using php we chose JavaScript. It allows to easily be able to ensure that a user is logged in cross site.
- We met again on Friday from 2 - 8, and decided to change from a MongoDB to a Firebase database, becasuse we thought it would be easier to handle the login. Firebase

is a NoSQL Json database that allows us to host our web app and helps maintain state across the domain.

- Separate Directory found here
- Website can be found here with UI: website

User Interface

1. Stub-calls for all interactive elements:

    - Stub Calls
    - Lead: Soya Ouk Co: Olivia Apperson
    - Stub Call Categories:
        - Registration
        - Files

2. Begin UI Elements: Website can be found here with UI:

    - http://ec2-35-163-197-29.us-west-2.compute.amazonaws.com/index.html
    - Lead: Olivia Apperson Co: Soya Ouk
    - Pages Include:
        - Login
        - Register
        - Edit File
        - Search File
        - Upload File
        - Landing Page
    - Pages were updated to accommodate PHP and connection to the database
    - Register page was deleted and consolidated with Login page

Other

1. Management of users/roles (User Accounts)
    - Lead: Xinyi Li Co: Teddy Ivanov
    - Researcher/User:
        - Logging in and out of the system
        - Uploading metadata files to be stored in the database
        - Search metadata files through key words
        - Download own files for use
        - Edit own metadata files
        - Delete own metadata files
    - Administrators:
        - Logging in and out of the system
        - Edit metadata files by all users
        - Delete metadata files by all users
        - Search files through key words
        - Add/delete users to the system

Testing and Documentation

Sprint 1: https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/RequirementsAnalysis_sprint1.docx.pdf

Sprint 2: https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/Final%20Project—Metadrop-SPRINT-2.pdf

Unfinished Tasks

- Further additions to the database will be needed but current status is on par with Sprint 2

- Further modifications and additions to the UI will be needed but current status is on par with Sprint 2

Log

- Updated Webpages
- Switched to Ubuntu server
- Updated Test Cases
- Updated Documentation
- Added user/admin roles
- Added stub-calls

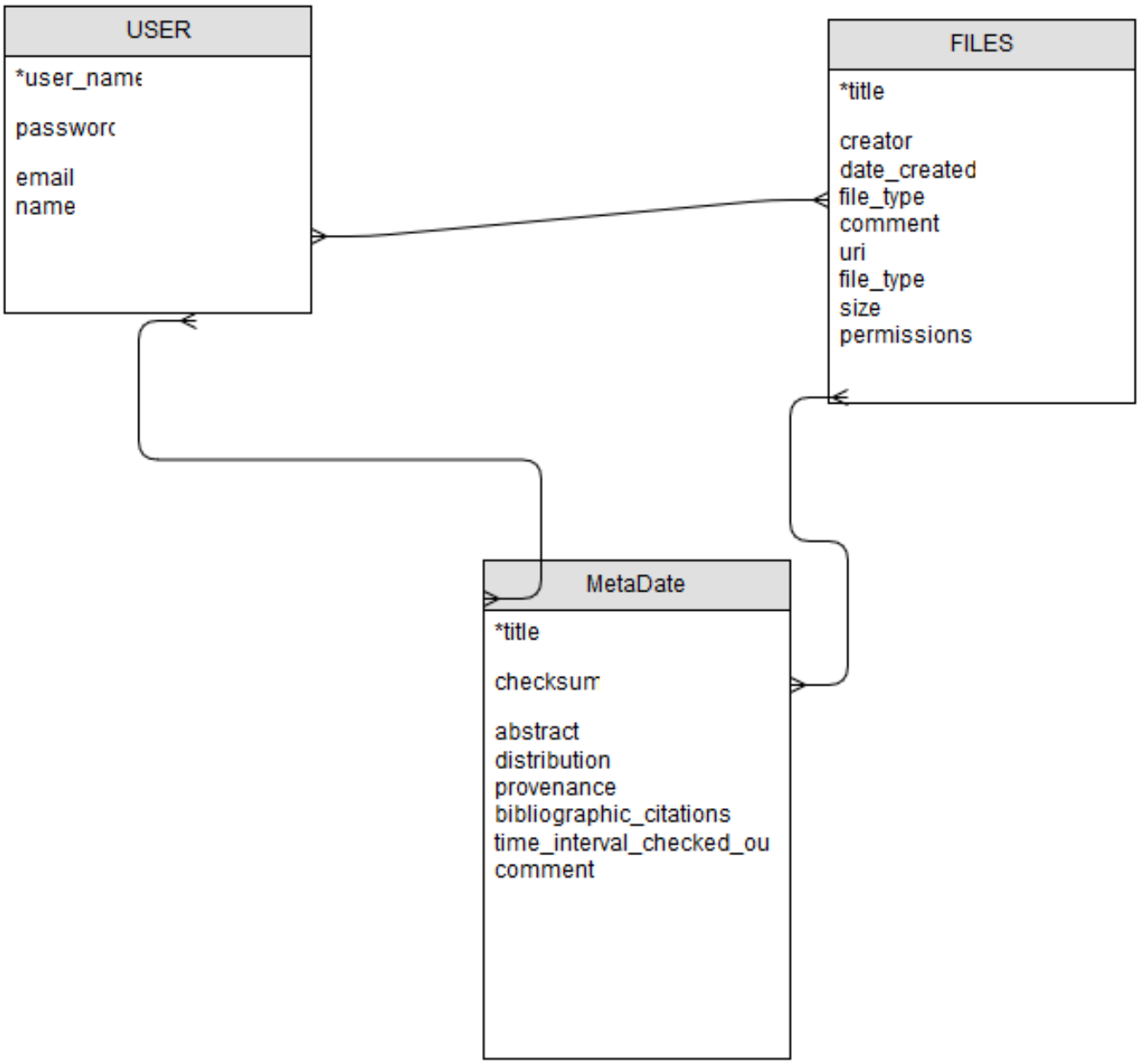
Revisions from Sprint 1

General

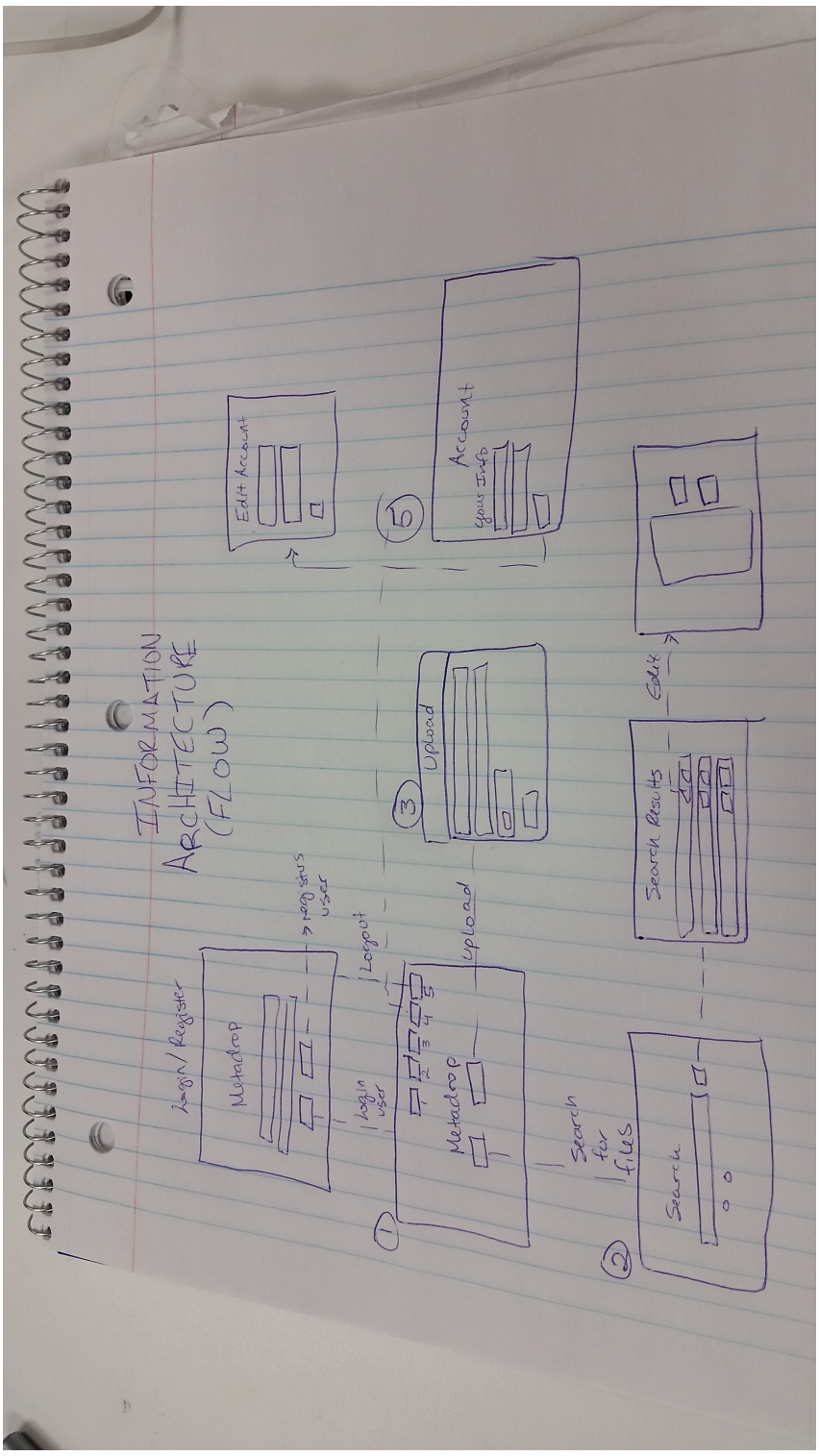
- Link to Github root included in submission:
    - https://github.com/TeddyIvanov/SoftwareEngineering-Group3
- Test cases now included below under testing section

Database/UI

- ERD revised and included below



- Information architecture (flow) is included below



Testing

- Github links with pawprints now available
- Revised Test cases included below:
    - https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/testing_sprint2.pdf

User Acceptance Test Scenarios for documented requirements (overall system of what is acceptable)

Login/Logout

- Users who have proper login credentials should be able to access the A.W.S. and perform certain tasks depending on their access rights.

Uploading/Downloading

- Ensure that Computation Social Scientists, Citizens, and Social Scientists can upload metadata.
- Make sure the System Administrator can upload any file type.
- Make sure that Computation Social Scientists, Citizens, and Social Scientists and the System Administrator can download any file type.

Editing

- Ensure that Computation Social Scientists, Citizens, and Social Scientists can edit and update metadata.
- Make sure the System Administrator can edit and update any file type.

Unit Test Scenarios

Being able to login

- **User provides the correct username and password and is redirected to the home page.**
- **User provides the correct username but incorrect password and is redirected to the login page.**
- **User provides correct password but incorrect username and is redirected to the login page.**
- **User provides incorrect username and password and is redirected to the login page.**
- **User provides a username that is nonexistent and is redirected to the login page.**

Being able to log out

- **Log user out if they're inactive for 10 minutes to avoid the database and/or A.W.S. from crashing or being modified by an unauthorized user.**
- **If user logs out, they cannot redirect back to the previous page or back into the website.**

Being able to upload a file

- **User uploads a file successfully.**
- **User tries to upload a word document, but the system denies it since it only accepts Json file types.**
- **User tries to upload a 300MB Json file, but it exceeds the maximum file size, therefore the system denies the user's request.**
- **User tries to upload an empty file, but the system doesn't accept it since it doesn't contain any content.**
- **User uploads a file with the same name, the system denies it to avoid overwriting.**

- User uploads a file with the same name, the system creates a copy of the same file to avoid overwriting.

Being able to download a file

- User successfully downloads a file.
- User tries to download a file but does not do so successfully.

Managing the Database

- User inserts data into a table in the database with all of the correct fields correctly and successfully.
- User inserts data into a table in the database but didn't spell one of the fields (columns) correctly, therefore the database rejects the request.
- User updates data in a table in the database but didn't spell one of the fields (columns) correctly, therefore the database rejects the request.
- User deletes data from a table in the database but didn't spell one of the fields (columns) correctly, therefore the database rejects the request.
- User deletes data from a table in the database but doesn't specify where it's getting deleted from, therefore the database rejects the request.
- User updates the database but the Administrator didn't approve, therefore the database rejects the request.

REGRESSION TESTING

- Select the tests for regression.
- Group members should present their code or any contribution they made towards the project since the last time we tested.
- The project manager will then run through all the unit testing, acceptance testing, and integration to ensure that the new update/bug fix didn't mess up the most current stable build.
- The project manager should verify that the new update didn't break the old, stable build and begin to merge and update.
- This process must happen on a weekly basis in order to complete each sprint in time.
- Putting everything together; we must perform regression testing on a weekly basis or prior to turning in each sprint to ensure the entire system and software works properly. Anything that gets changed/updated in our system should be tested. Integration testing will be applied if we're updated multiple sections of our system.

Verification tests

- Unit testing
  - Login/Logout
  - Upload/Download files
  - Manage the database
- Regression testing

Validation tests (Login)

- **User Acceptance testing**
- **Integration testing**

Integration Testing

- **We'll use the *Bottom Up Integration* testing method to test each individual component:**
  - **Users can login and logout successfully.**
  - **Users can make updates within the database and have all of the work save successfully.**
  - **Users can insert data into the database successfully.**
  - **Users can delete data from the database successfully.**

Sprint 3

General

Sprint documentation can be found here and within this Sprint 3 Markdown

User Interface

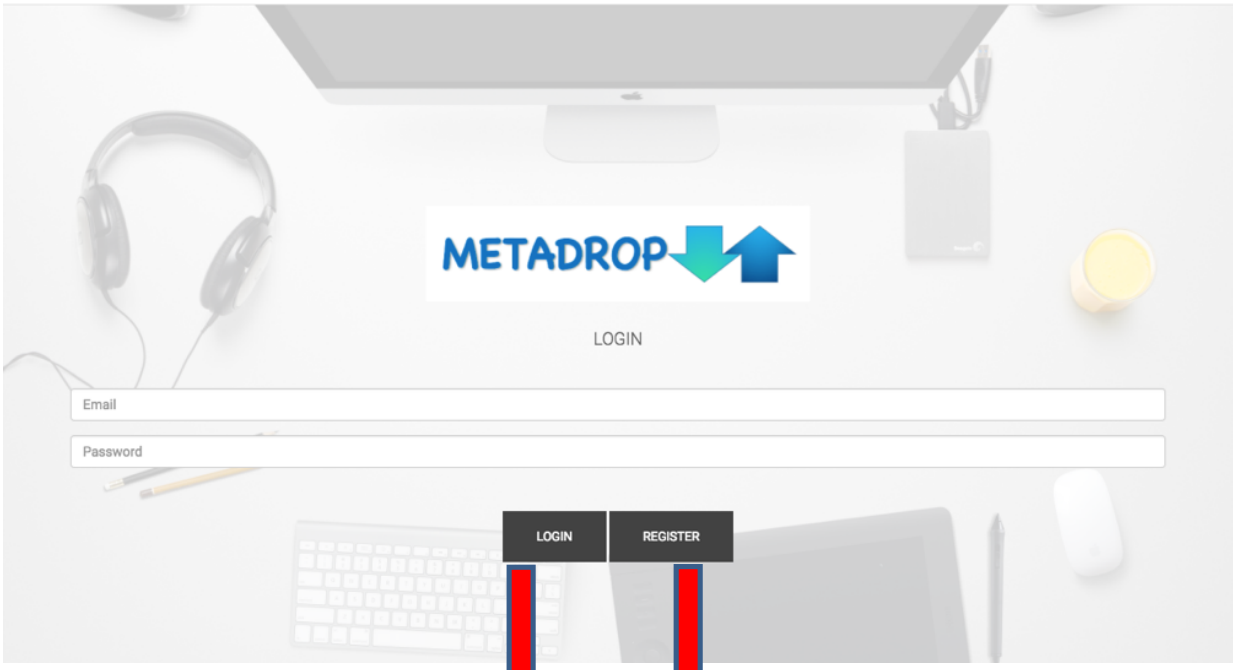
- Pages were updated to correctly connect to database and to check if links were still in working condition
- The following pages were added/edited for additional features:
    - Account.html --Changed fields of user information and correctly linked Edit button
    - EditAccount.html --Added this page to allow user to edit personal information
    - Modify.html --Added a comments section for more user input

Temporal Logic of UI

Doc: https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/oamr6_Sprint3/Temporal%20Logic%20of%20UI.docx

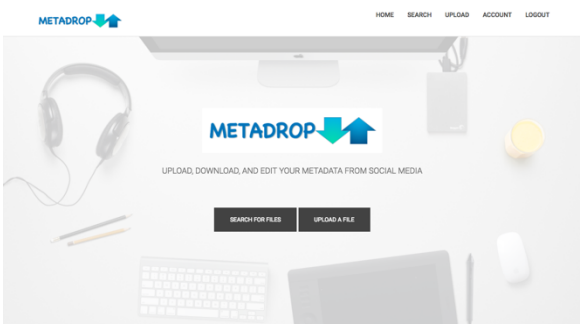
***Full Tutorial of Website can be found in the user manual, located here. The following is the flow and navigation of pages**

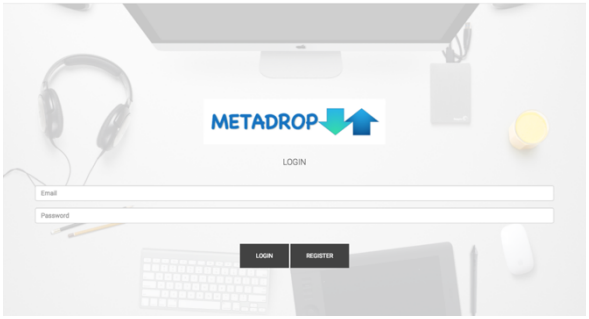
Login/Register Page:

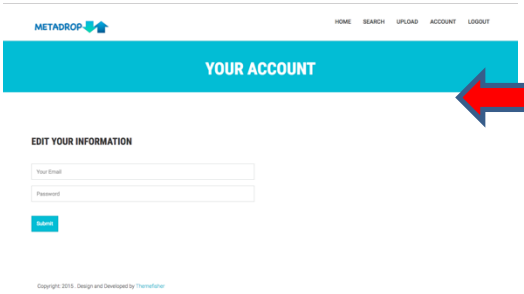


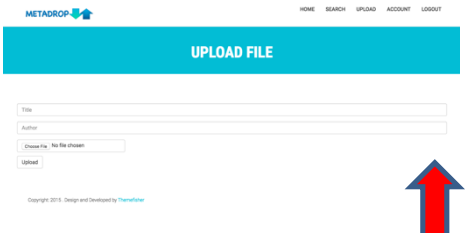
Landing Page

Login/Register Page: Prompted to log in





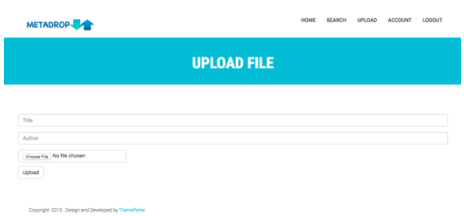


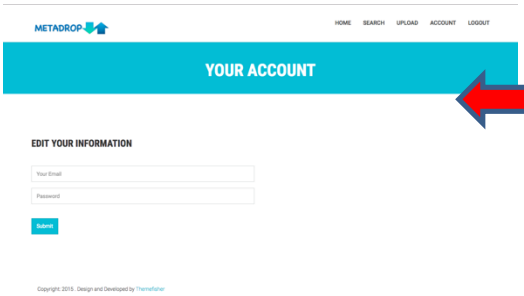


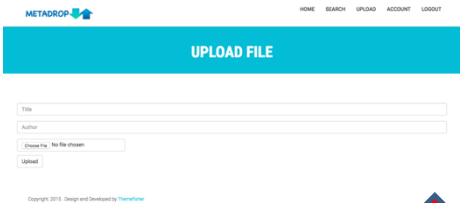
Landing Page

METADROP

HOME SEARCH UPLOAD ACCOUNT LOGOUT



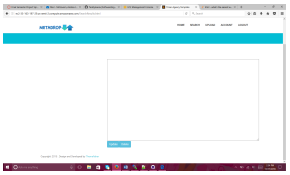




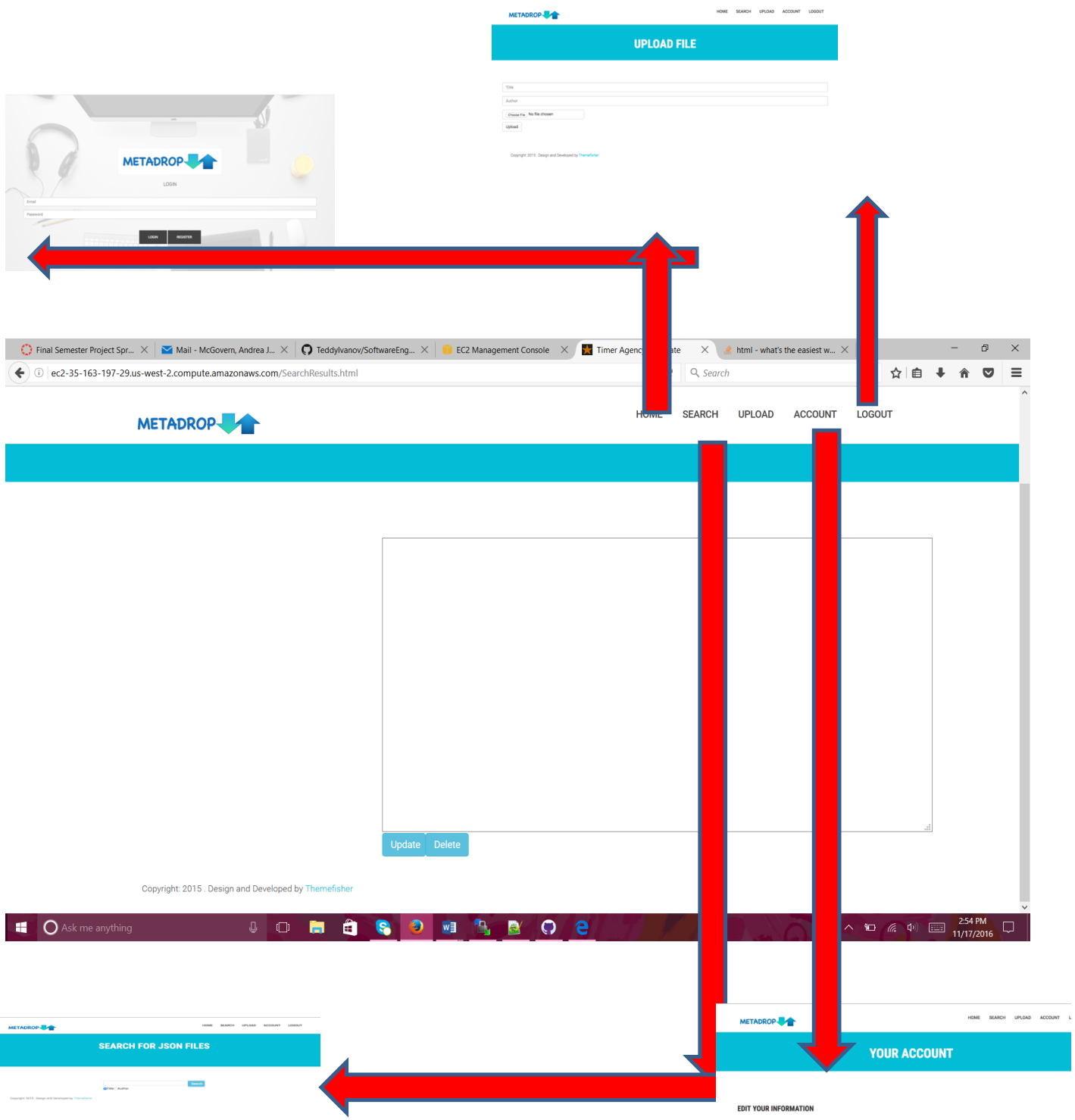
Search for Files Page

Copyright: 2015 . Design and Developed by Themefisher

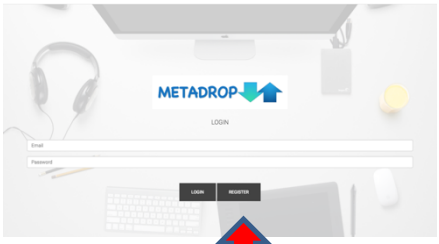
Search Results

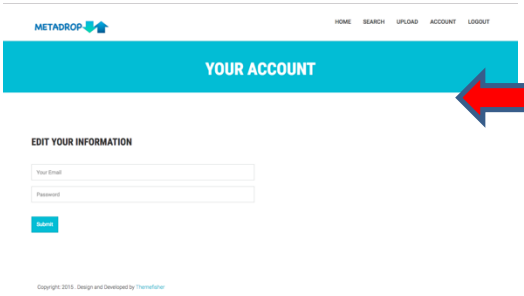


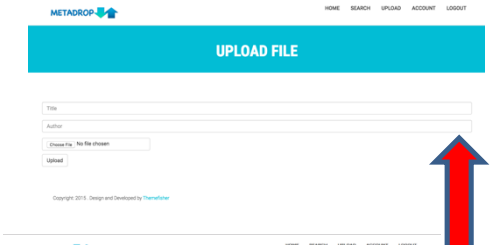
Search Results Page



Both update and delete buttons connect to the database to update
And delete files







View/Edit Page

METADROP HOME SEARCH UPLOAD ACCOUNT LOGOUT

MODIFY FILE

Title

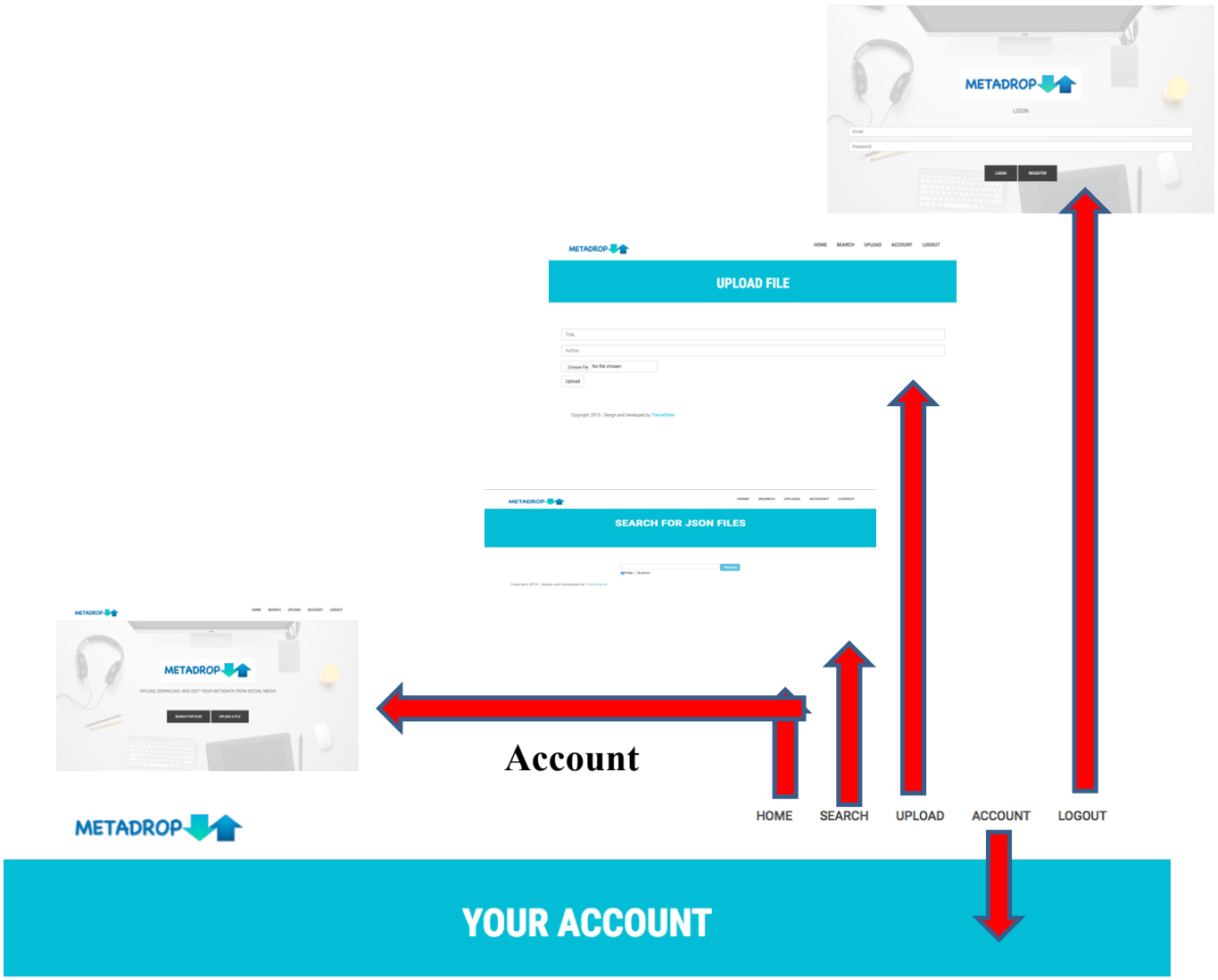
Author

Add Comments:

Submit

<- Search results page

Success Message



YOUR INFORMATION

Your Email

Password

Edit

Copyright: 2015 . Design and Developed by Themefisher

EDIT YOUR INFORMATION

Your Email

Password

Submit

Copyright: 2015 . Design and Developed by Themefisher

Testing

- Description of user tests
- Edge cases

User Documentation

User Documentation was structured and started. The following sections were added:

- Introduction (Olivia Apperson)
- Software and Applications needed (Andrea McGovern)
- Deployment (Andrea McGovern and Teddy Ivanov)
- End User Manual (Olivia Apperson)

Improvements to Sprint 2

8 points were lost in Sprint Documentation

- Sprint 2 added to Requirements and Design Documentation
- Broken links, including Stub Calls and DML, fixed
- Link to Commits added

2 points lost for Testing

- User Acceptance testing not included in Regression testing
- Verification/Validation revised

Version 5 (Sprint 3)

- Deployed all documents to Github
- Fixed broken links in readme, and reorganized files so they are easier to find
- Temporal logic was created, and added to the GitHub

- Switched from MongoDB to a Firebase database, because it is easier to maintain state across the site
- Updated test cases to reflect changes that needed to be made based on feedback from sprint 2
- User documentation was started
- Started working on JavaScript files to be able to download, update, delete, search, and upload files to the database

Sprint 4

Meetings

- **Collaborated over GroupMe application in order to coordinate jobs over Thanksgiving Break. All members attended.**
- **Met 11/27/16 from 1:30pm-6pm to work on and complete tasks for Sprint 4. All members attended.**

General

- **Sprint documentation is included here and was submitted on 11/28/16. It is also included in the Markdown page on Github.**
- **Automated Script- confused on how this should be deployed. Deployment instructions are included in links later in this document.**

User Interface

- **SearchResults.html modified toward full functionality.**
- **Upload JavaScript file cleaned and made more robust for easy use and understanding.**
- **All pages now include scripts to make sure they are accessed only when user is registered and logged in.**

Testing

- **Tests elaborated from Sprint 1 and Sprint 3 can be found here.**

Deployment Instructions

- **User Documentation can be found here in the GitHub.**
- **Readme.MD includes deployment instructions. This can be found here.**

Improvements From Sprint 3

- **User documentation included**
- **Pages require login to be accessed**
- **Test cases improved for integration**

Glossary

Created by: Soya Ouk, Reviewed by: Olivia Apperson

Computer Platform: A system that consists of a hardware device and an operating system that an application, program or process runs upon.

Computational social science: refers to the academic sub-disciplines concerned with computational approaches to the social sciences.

Functional Requirements: Details of services the software must provide.

JSON: A language easy to write and edit by humans and easy to read and parse by machines.

Metadata: a set of data that describes and gives information about other data.

Non-Functional Requirements: Constraints on the functionalities of the software.

OCDX: Open Community Data EXchange, is a metadata specification and robust infrastructure for long term sustainability.

OCDX Manifest: a bill of materials for datasets.

System Requirements: Pre-requisites that often define the operating environment.

User Requirements: Facts and assumptions about the expected outcome of the software implementation; What the software will enable a user to do or not do.

Change Log

Created by: Olivia Apperson, Reviewed by: Andrea McGovern

Version 1

- Project Overview created
- Requirements Analysis created
- Class/Function List created
- Table List created
- Screen Designed Created
- Change Log Created
- Glossary Created

Version 2 (Revised for more points)

- Glossary Updated
- NonFunctional Requirements Updated
- Functional Requirements Updated
- User Requirements Updated

Version 3 (Sprint 1)

- Screen Designs Added
- Website URL added
- Updated ERD added
- Markdown (Wiki) link added
- Database and Storage information

Final Version

- Documentation cleaned up for clarity -- Olivia Apperson and Xinyi Li
- Code finalized and in working condition -- Teddy Ivanov
- Deployment instructions reworked and edited -- Soya Ouk
- Test cases revised and edited -- Andrea McGovern