Final Project—Metadrop

Sprint 2

11/11/16

Teddy Ivanov, Andrea McGovern, Olivia Apperson

Maggie Li, Soya Ouk

Branches-Students created their own branches for this Sprint. Branches are denoted as pawprint_Sprint2.

Meeting-November 7th, 2016

All members attended meeting starting at 4pm and ending 10pm. Met to discuss individual progress and work together to complete other tasks. Discussed Sprint goals to ensure all parts were covered. Also worked on revising Sprint 1 for redo. These tasks determined to complete in Sprint 2 were divided amongst team members and will be explained below.

Tag-Up Meeting-November 10th 2016

All members attended meeting starting at 6pm and ending at 9pm. Finalized changes and updates for Sprint 2. Continued working to improve Sprint 1 content.

General

- Sprint Documentation
    - https://github.com/TeddyIvanov/SoftwareEngineering-Group3/blob/master/README.md
- Lead: Olivia Apperson Co: Maggie 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: Maggie 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

Link to Sprint 1 and 2 is at the top of the README

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

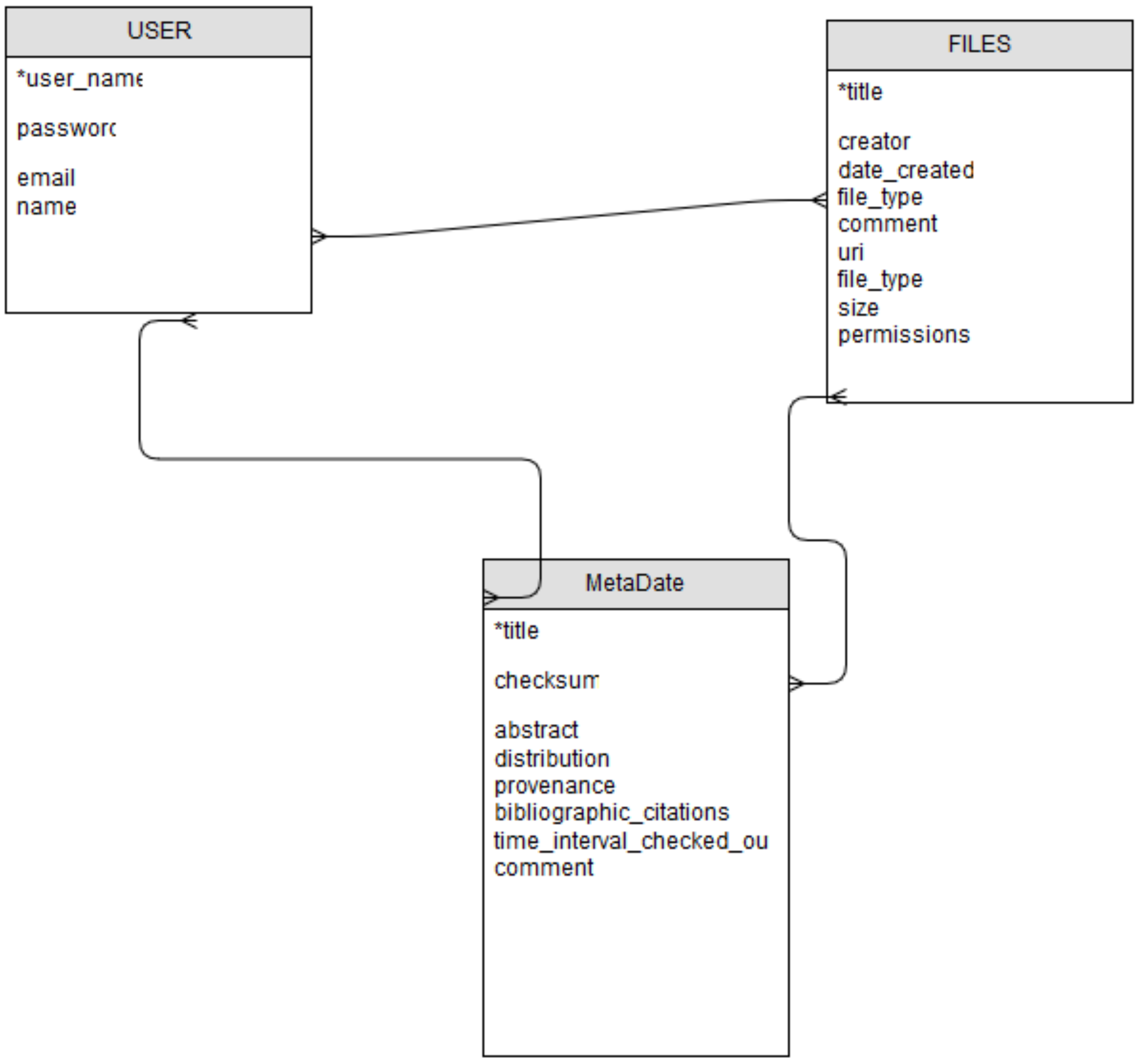
Revisions from Sprint 1

General

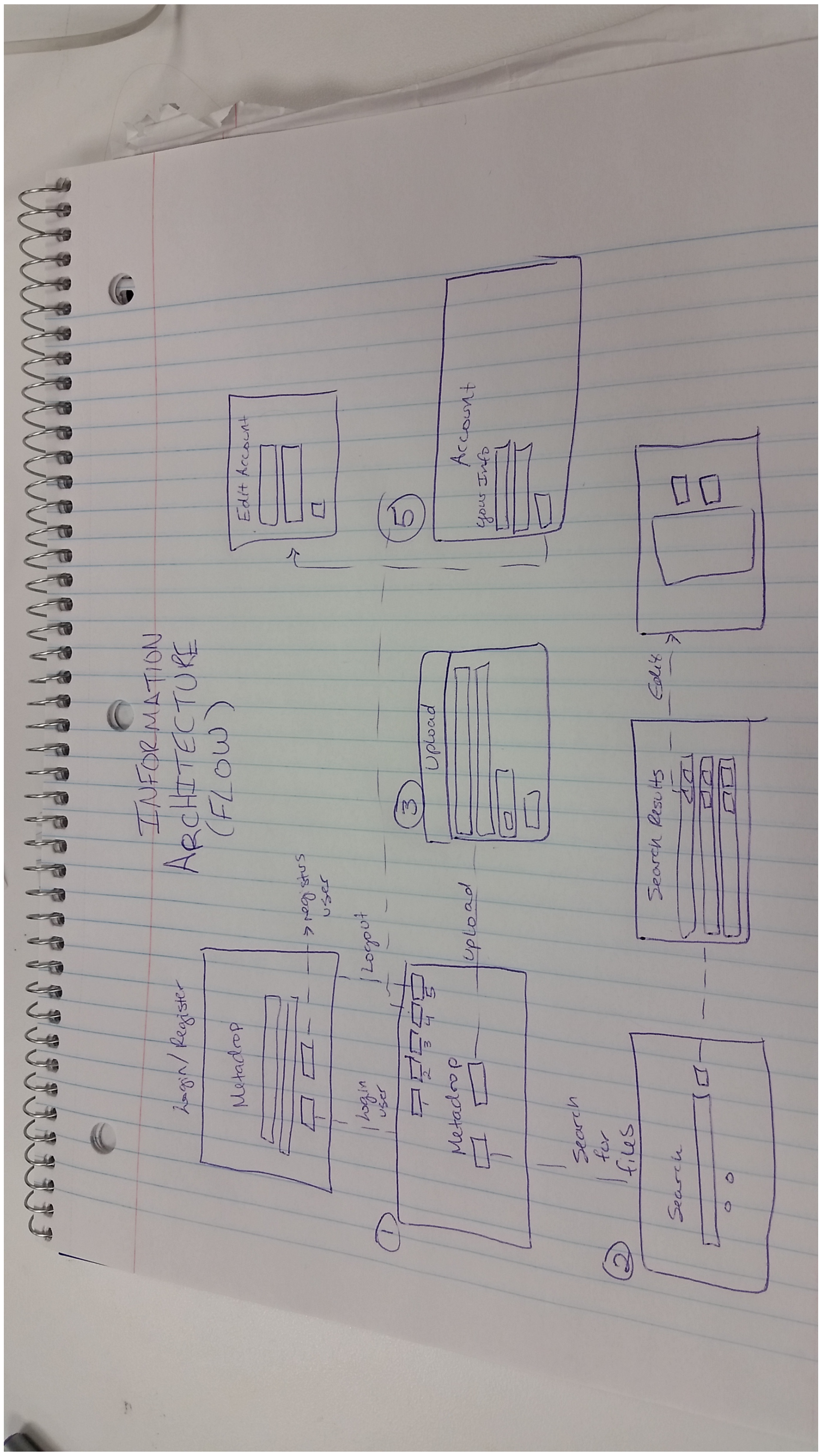
- Link to Github root included in submission:
    - https://github.com/TeddyIvanov/SoftwareEngineering-Group3
- Test cases now included below under testing section

Database/UI

- ERD revised and included below

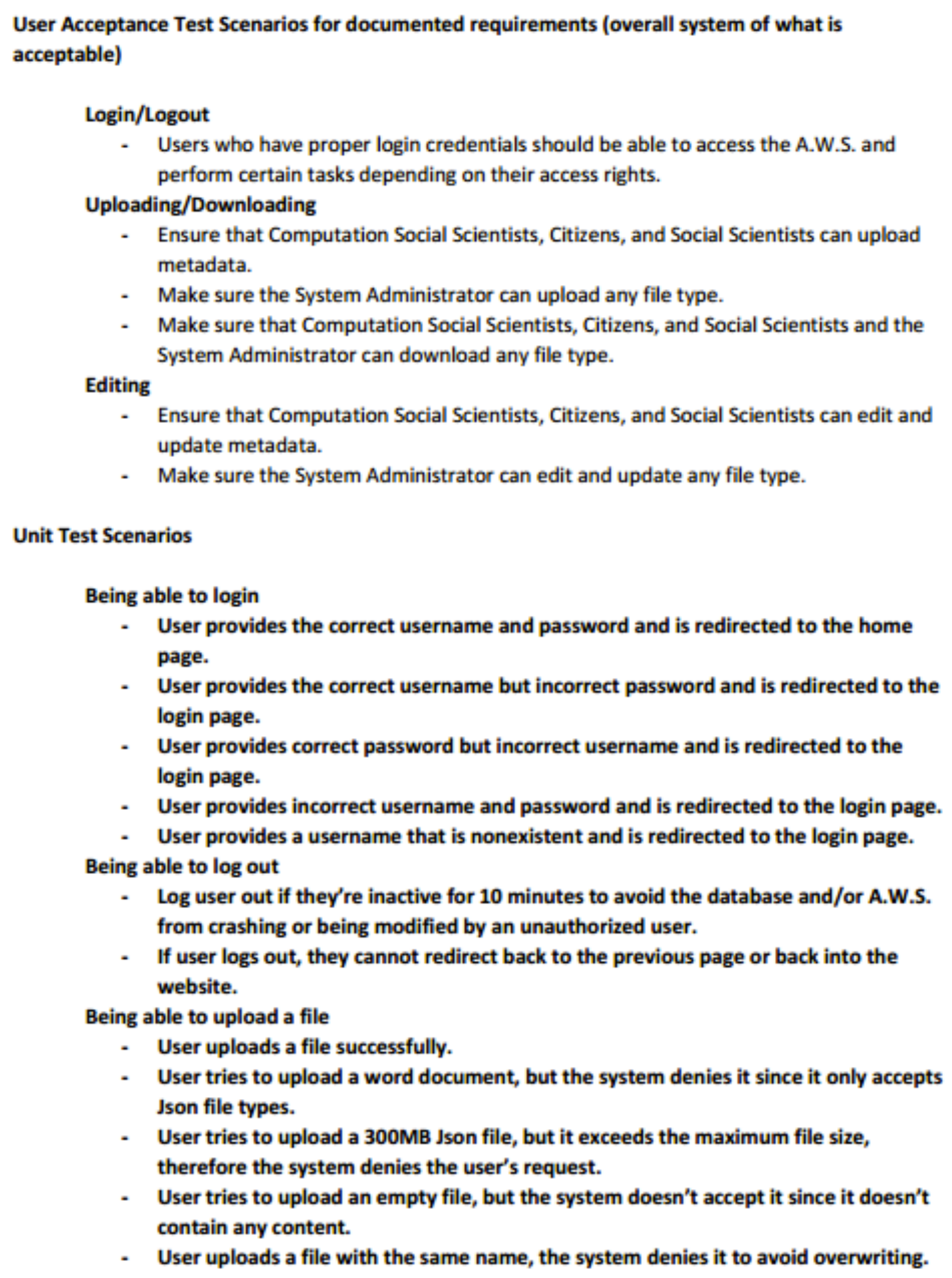


- Information architecture (flow) is included below



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.**