

Software Name: Data Exchange Platform

Group 5 Version 4

Kurt Bognar, Zach Bryant, Luke Jehle, Yi
Hou, Hanyu Liang, Kyle Whitney

Date: 11/28/16

Contents

Sprint 1:

| | |
|----|--------------------------|
| 3 | Glossary |
| 4 | Tasks Assignment |
| 7 | Reqs Doc |
| 11 | UI design |
| 15 | Database Design |
| 18 | Class Diagram |
| 19 | Information Architecture |
| 22 | Testing Architecture |

Sprint 2:

| | |
|----|-----------------|
| 24 | Task Assignment |
| 25 | SQL |
| 35 | Stubb calls |
| 36 | Tasks Completed |

Sprint 3:

| | |
|----|--------------------|
| 38 | Task Assignment |
| 39 | Temporal logic |
| 42 | Updated Testing |
| 45 | User Documentation |
| 50 | UI Code |
| 56 | UI Pics |
| 60 | Tasks Completed |

Sprint 4:

| | |
|----|------------|
| 82 | Change Log |
|----|------------|

Glossary

Schema - a unit of breaking down a database into subgroups of tables

Repo (repository) - a place where code and documentation can be stored for sharing and version control

Branch - a version on a repo that can has concurrent development going on

Regression testing - making sure all your tests from previous iterations pass

Integration testing - testing whether your software modules sync

Unit testing - testing whether data is passed between platforms properly

Manifest - information about a dataset and its according scripts

Sprint - a unit of measure of time in Agile project management (see iteration)

Stub call - a generic version of a function to be developed

Iteration - a single version of a software project

Github Repo: <https://github.com/ztbc68/OCDX-Engine-Group5>

Website: <http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com/>

SWE Team 5 Sprint 1 Kickoff Meeting

Kurt Bognar, Zach Bryant, Luke Jehle, Yi Hou, Hanyu Liang, Kyle Whitney attended at the student center 10.25.16 at 7pm.

Tasks

- Review Milestones
- Delegation
- Drill-down tasks needed for each milestone.
- Member show on the wiki Completed by linking the wiki to the GitHub Repo Hash that completed the task.
- During Sprint evaluations, evaluation will be performed for both Team and Individual aspects.
- **Each team member must have the lead (and commit) on one or more software implementation (coding) tasks for each sprint.**
- fork & pull or “everyone commits” strategies allowed

Task Assignment

Kurt **Liang** **Yi** **Zach** **Kyle** **Luke**

| General | Database | User Interface (see Note1) | Testing & Documentation (see Note1) |
|---|---|---|--|
| <ol style="list-style-type: none">1. Sprint Documentation2. Complete setup of Deployment Environment3. Organize GitHub repo | <ol style="list-style-type: none">1. Finalize ERD2. Database creation SQL (Data Definition Language)3. Implement DB, seed data for development. Basically, create a bunch of bogus data so you can run the application. | <ol style="list-style-type: none">1. Complete design of the user interface (all screens so entire UI team)2. Complete design of the information architecture (how the clicking works; what clicks lead to what pages, etc) | <ol style="list-style-type: none">1. Build user acceptance test scenarios for documented requirements on separate Wiki page, linked to all sprints.2. Build unit test scenarios3. Describe regression testing and your regression testing plan.4. Describe how your team will perform integration testing. What needs to be integrated? When?5. Describe which tests are for verification and which tests are for validation |

Completed Tasks

Kurt

- DB creation SQL:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/37516b2300bf39f198f885e4fef9203ce1856ed7>
- Finalize ERD:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/283f6c8e16aeb6237a7a15a8b8e7b940fb321c61>
- Implement DB:
- Sprint Documentation:
- Setup Postgres:

Liang

- Documentation:information architecture:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/dbd761be9112f48629610d86510d2a3ba3788216>

Yi

- Seed data creation:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/87d0b541bae3e5592a3f9d2911ba69ec72a23871>
- Sprint Documentation: Regression testing

Zach

- Testing:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/790df72a5f6581899be96bfba97e413dde1a7e6f>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/4d9363a884b0848be7776e4eaec5073a4db414e>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/5fe185bcaa4526da74aa223cc099057de8046d6e>
- Organize Git:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/ceb0ab6f9f6613b4caf0dde56101d165e1062308>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/552c92e17d60eb33746c21a5243b155bb9af5cfd>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/fcc9a83fd783514e4b880d95361200bb29b7f607>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/6b1fc8318087135eba35f5ec7e4885747c4576dc>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/1f1cd4dd49c17c374cbd46a488be79419f514536>

- Hello World Page:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/5720143d5c26e1653166ebb8a52aabe3783752a7>

Kyle

- Class diagram:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/a2083f867ea60bca32a85f21ea4b7b543123b802>
- User Acceptance Test Scenarios:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/459697d11214d413b0e1c4458517cece78091bbc>
- Integration testing:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/459697d11214d413b0e1c4458517cece78091bbc>

Luke

- Unit testing:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/790df72a5f6581899be96bfba97e413dde1a7e6f>
- How clicking works:
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/dbd761be9112f48629610d86510d2a3ba3788216>

Requirement Analysis

Contents

| | |
|-----|---------------------------------|
| 3 | Glossary |
| 3 | Users and activities |
| 4 | Data |
| 5 | System vs User Requirements and |
| 6 | Functional vs Non-Functional |
| 7-9 | Screen designs |
| 10 | ERD |
| 10 | Wiki Page Link |
| 10 | Group Github Repo Link |
| 12 | Change Log |

Glossary

System requirements: Prerequisites that define the operating environment, architecture, hardware

User requirements: Requirements that specifies what the user expects the software to be able to do

Functional requirements: Requirements defines a function of a [system](#) or its component

Non-functional requirements: Requirements that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors

ERD: a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems

Class Diagram: a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects

Users

1. Researchers - users who actually submit data to the service and do data science.
2. Independent Users - users who simply view the stuff on the sight and post comments but don't actually research
3. Admin

Activities for Each User

1. Researchers can:
 - a. Report other researchers, pages or users to the admin via a button.
 - b. Do CRUD operations on information and profiles they own
 - c. Comment on other people's profiles
 - d. Login to the site
 - e. Request changes to be made on others research
2. Independent Users
 - a. Report other researchers, pages or users to the admin via a button.
 - b. Do RD operations on information and profiles they own
 - c. Comment on other people's profiles
 - d. Login to the site
3. Admin
 - a. Ban users who post inappropriate content
 - b. Do CRUD operations on information and profiles they own
 - c. Login to the site
 - d. Grant permissions to users who have become researchers and admin permissions to researchers
 - e. Accept changes that have been requested on profiles

- a. Comment/delete comments on other people's profiles

Data (all elements require ID and timestamp)

1. Create
 - a. JSON, page_ID
2. Read
 - a. page_ID, time_on_page
3. Update
 - a. page_ID, updated_JSON, archived_JSON
4. Delete
 - a. page_ID,
5. Accept changes
 - a. Acceptor_ID, changer_ID, content
6. Delete comments
 - a. Report_id
7. Permissions
 - a. Id_type
8. Banning
 - a. Report_id
9. Change
 - a. Requestor_id, page_id, content, admin_assigned, researcher_assigned
10. Login
 - a. User_id, password, salt, attempts?
11. Comment
 - a. Content, page_id

12. Report

- b. Reporter_id, reportee_id, reason, page_id, comment_id, admin_assigned

Use Cases (optional)

System vs User Requirements

1. User Reqs

- a. Users are able to upload a file of any size and any type.
- b. Users are able to create a new manifest page.
- c. Users are able make any changes to their manifests.
- d. Users can delete anything they created
- e. Users can comment on other uses' manifest.
- f. Users are able to sign in or login to the system.
- g. Users can search based on variety of fields.
- h. Collaboration and feedback will be built into the system.

2. Sys Reqs

- a. The system will run on a Linux server
- b. The system will have >10gb of memory (~1TB)
- c. The system will require the user to be connected to internet
- d. The system will have a PostgreSQL database
- e. The system must allow multiple users to access the system at once.
- f. The system will be accessible via any browser.

Functional vs Non-Functional

- 1. The functionality of the system will allow the user to

- a. Users can login the system with a username and password. If they don't have usernames and password, they can use sign in page to sign in.
 - b. When users click create button, a new page will be created.
 - c. When users click comment button, they can type their comments and publish to the page.
 - d. When users type keywords and click search, the all related web pages links will be displayed.
 - e. When users click upload button, they can upload files less than 10 GB.
 - f. When users click delete on the page or comments they created or posted, the manifest or comments should be delete.
 - g. Admin has the option to ban any users by changing user's status.
 - h. Users can edit their manifest anytime by clicking edit button.
2. Non-functional requirements of the system are
- a. Response time is less than 3 seconds.
 - b. The app can support maximum of 100000 users online at the same time.
 - c. The app server maximum downtime is less than 40 hours a year. One-time downtime of the app server is less than 2 hours. A backup server is needed.
 - d. The app will be maintained every month.
 - e. The app will be opened in all kinds of operating systems such as windows, linux, macos, not cellphones/tablets
 - f. The app will ensure data security and will implement security measures to prevent the system from cyber attack.

Primary Author: Kurt Bognar

Secondary Authors: Kyle Whitney, Zach Bryant, Yi Hou, Luke Jehle, Hanyu Liang

Screens

Login/Registration Page

Author: Luke Jehle Reviewer: Yi



A screenshot of a web application's login and registration page. The page has a yellow background and is divided into two sections by a vertical line. The left section is titled 'Login' and contains fields for 'Username:' and 'Password:', followed by a 'Login' button. The right section is titled 'Create Account' and contains fields for 'First Name:', 'Last Name:', 'University:', 'Role:', 'Email:', 'Username:', 'Password:', and 'Reenter Password:', followed by a 'Submit' button. Above the 'Login' section, the letters 'N A V B' are visible, and above the 'Create Account' section, the letters 'A R' are visible.

Upload File Page

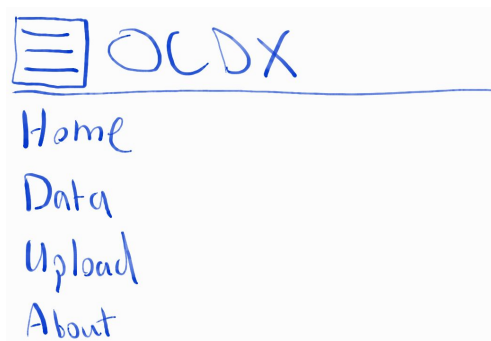
Author: Kyle Whitney Reviewer: Zach Bryant



A screenshot of a web application's 'Upload File' page. The page has a yellow background and a red 'X' icon in the top right corner. The title 'Upload File' is centered at the top. Below the title is a horizontal line. Underneath the line are two buttons: 'Select File' and 'C:/file/fileupload'. At the bottom of the page is a large 'Upload' button.

Nav Bar

Author: Zach Bryant Reviewer: Kyle Whitney

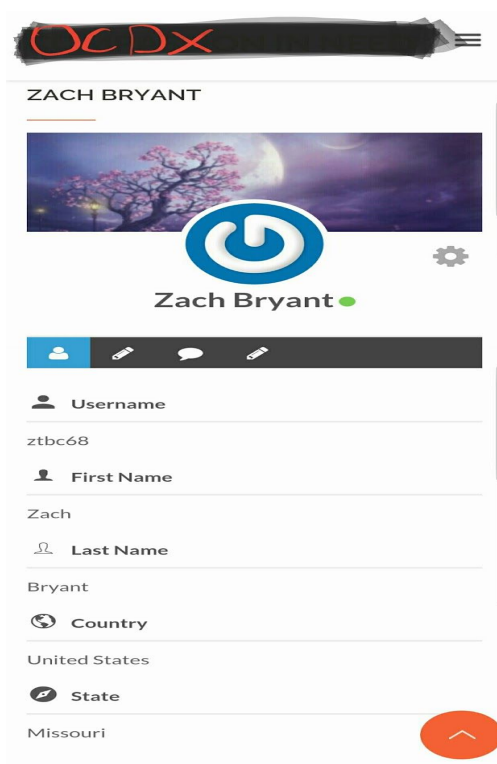


A screenshot of a web application's navigation bar. It features a blue hamburger menu icon on the left, followed by the text 'OCDX' in a blue, handwritten-style font. Below this, there is a horizontal line, and then a list of navigation links: 'Home', 'Data', 'Upload', and 'About', all in a blue, handwritten-style font.

Profile Page

Author: Zach Bryant

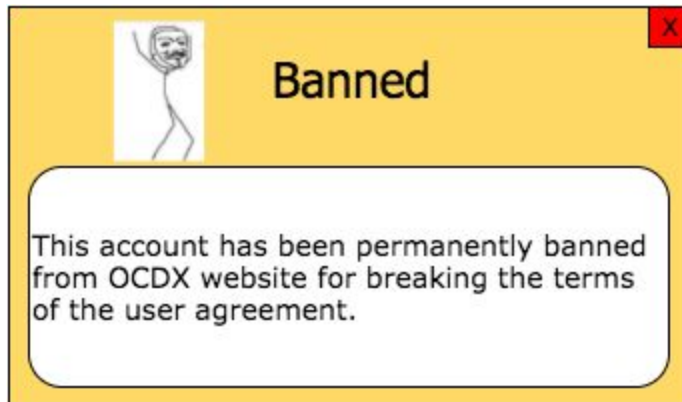
Reviewer: Hanyu Liang



Banned User Page

Author: Kyle Whitney

Reviewer: Luke Jehle



Search Page

Primary author:Hanyu Liang

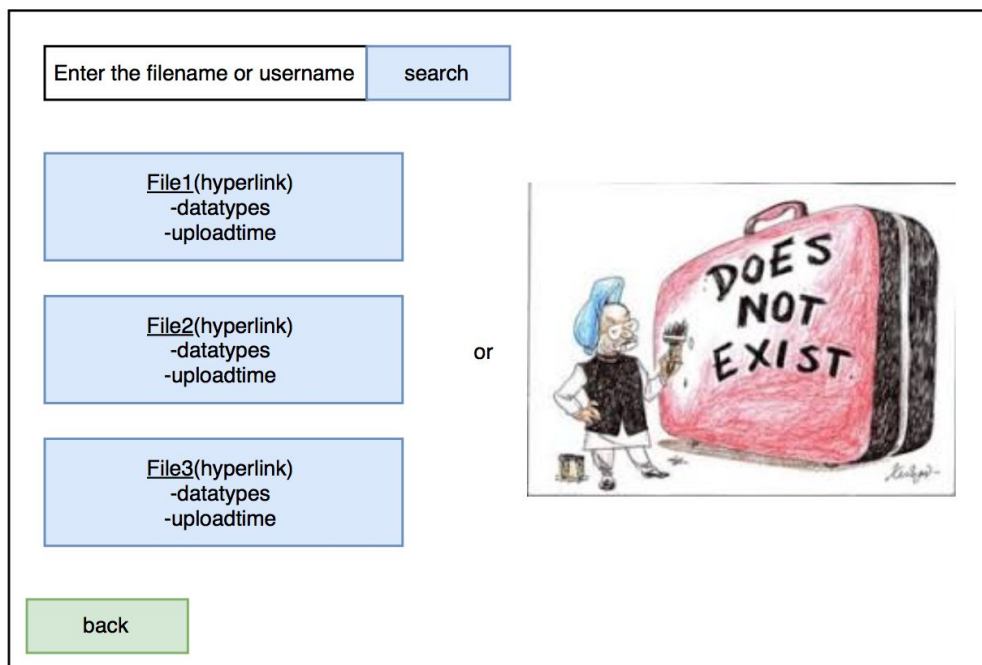
reviewer:Kurt Bognar



Search Return Page

Primary author:Hanyu Liang

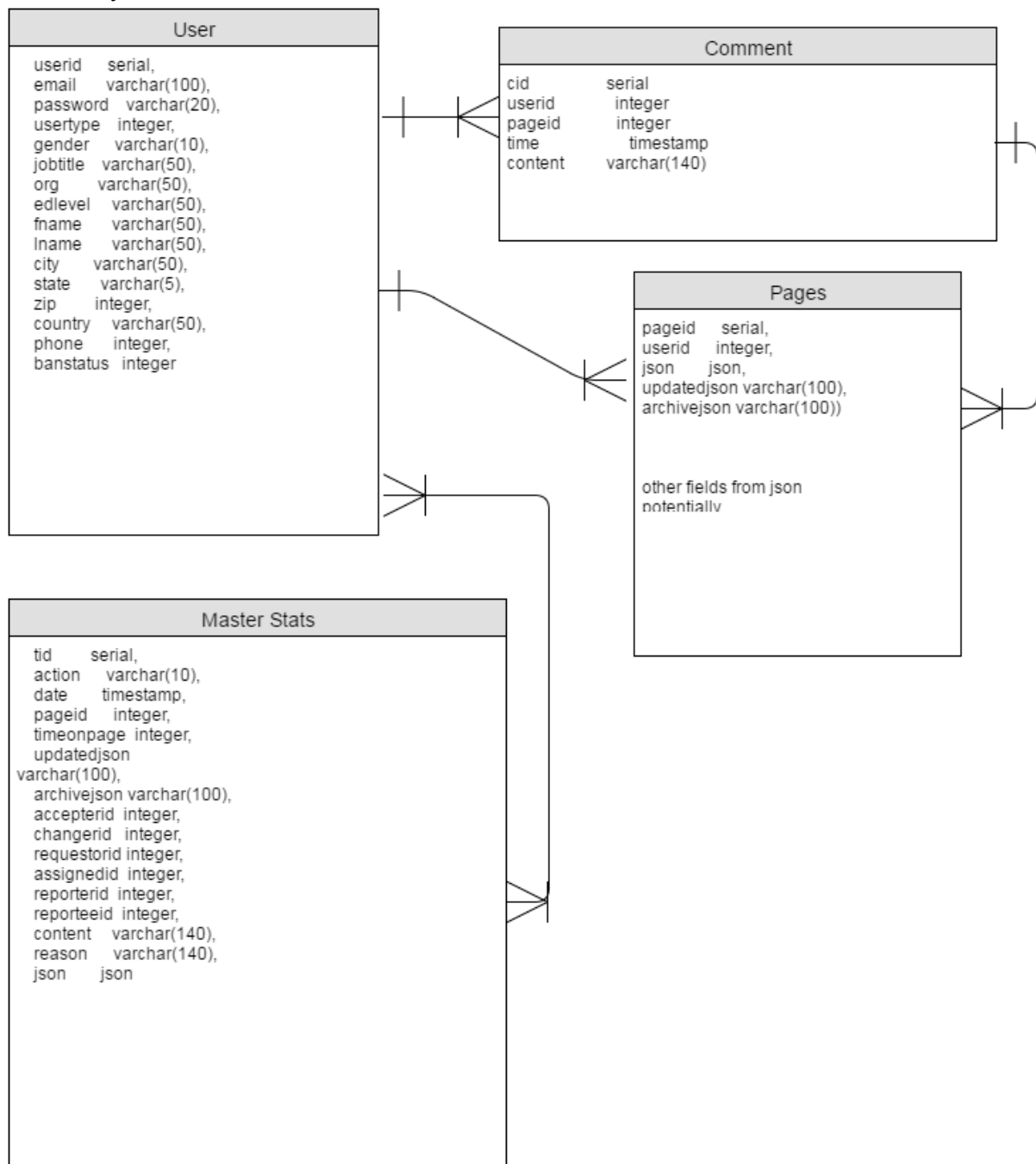
reviewer:Kurt Bognar



ERD

Authors: Kurt Bognar, Yi Hou

Checked by: Luke Jehle



Wiki Page Link

<https://docs.google.com/document/d/1eyouf5pZJNSQW618xA-JStpf18gW9idJlE5P3wCr0ko/edit?usp=sharing>

Group GitHub Repo Link

<https://github.com/ztbc68/OCDX-Engine-Group5>

Change Log

- Updated and finalized ERD
- Added Class diagram
- Updated Reqs document
- Created a main menu mockup
- Created user profile and nav bar mockups
- Organized git: created docs folder and moved relative items
- Created wiki page and organized git
- Added change log
- Added glossary

DML

Authors: Kurt Bognar, Yi Hou

Checked by: Luke Jehle

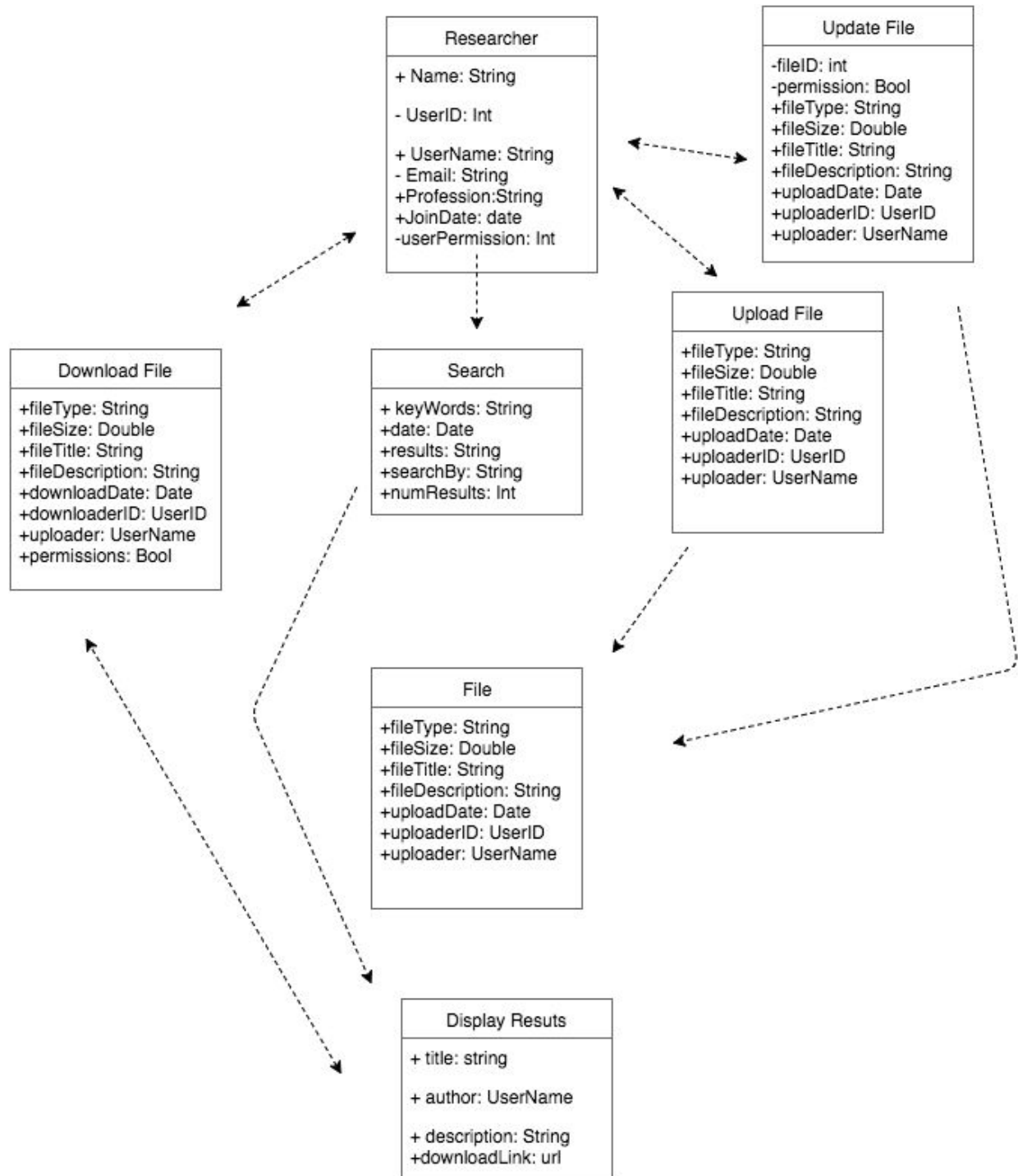
```
drop schema swedb;
create schema swedb;
drop table if exists swedb.pages;
create table swedb.pages(
    pageid      serial,
    userid      integer,
    json        varchar(100),
    updatedjson varchar(100),
    archivejson varchar(100));
drop table if exists swedb.comment;
create table swedb.comment(
    cid          serial,
    userid       integer,
    pageid       integer,
    time         timestamp,
    content      varchar(140));
drop table if exists swedb.users;
create table swedb.users(
    userid      serial,
    email       varchar(100),
    password    varchar(20),
    usertype    varchar(20),
    gender      varchar(10),
    jobtitle    varchar(50),
    org         varchar(50),
    edlevel     varchar(50),
    fname       varchar(50),
    lname       varchar(50),
    city        varchar(50),
    state       varchar(5),
    zip         integer,
    country     varchar(50),
    phone       varchar(20),
    banstatus   integer);
drop table if exists swedb.stats;
create table swedb.stats(
```

```
tid      serial,  
action   varchar(10),  
date     timestamp,  
pageid   integer,  
timeonpage integer,  
updatedjson varchar(100),  
archivejson varchar(100),  
accepterid integer,  
changerid integer,  
requestorid integer,  
assignedid integer,  
reporterid integer,  
reporteeid integer,  
content  varchar(140),  
reason   varchar(140),  
json     varchar(20));
```

```
drop table if exists swedb.intermediary;  
create table swedb.intermediary(  
    tid          integer,  
    userid       integer);
```

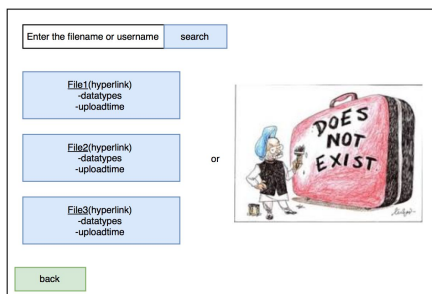
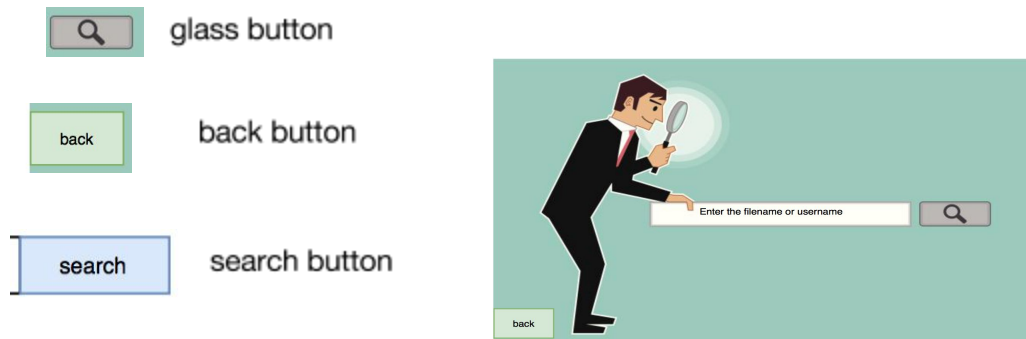
Class Diagram

Created by: Kyle



Information Architecture

About search: In the blank , enter the filename or username that you what to search. And then click the search button(the glass). If you don't want to search anything, you can click back button and it will go to main page. After clicking the button, it will go to search2 page. In the search2 page, the page will shows all file or username about you want to search if the filename or username is exist. Otherwise, the page will show the photo in the right that means you failed search something. If you want to search again, you can enter the filename or username in the text blank. And then click search and it will go to search2 again. If not, you can click back and then the search will end and back to the main page.



about file profile: The file profile shows information about the file.when you click the person button,It shows the personal information of files' owner. when you click the pencil button, it shows report page.you can give a reason and report this file if you hate this file. The third button, the cloud button, is comment button. when you click this button, it will go to comment page. In the comment page, you can look all comments about this file and write yourself comments about this file. When click top-right button, it will go to Nav Bar page.



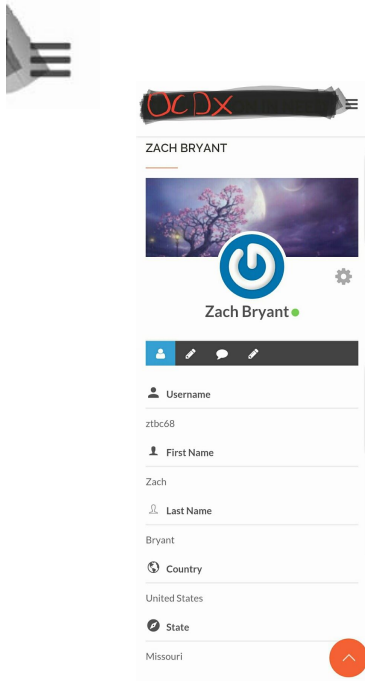
person button



pencil button

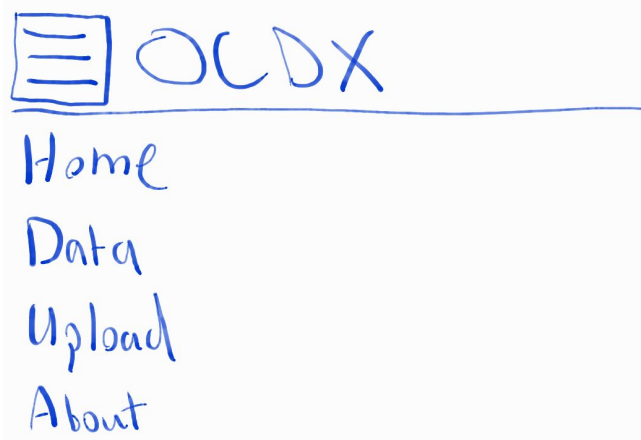


cloud button



About Nav Bar: in the Nav Bar, it has four button: Home, Data, Upload and About. when you click these button, the page will go to other page ,Home page, Data page, Upload page and file profile page.

| | |
|--------|---------------|
| Home | home button |
| Data | data button |
| Upload | upload button |
| About | about button |



How Clicking Will Work

Login Page

After username and password have been entered click login - user will be directed Home Page

- If no username and password is given, error message will be received on click that the user needs an appropriate username and password.
- If incorrect username or password, error message will be received the username and password they entered are incorrect

When creating account, when all information has been entered and submit is clicked, the user will be sent to the login screen again, a success message will pop up saying the user has been registered.

- If not enough info has been entered, error will pop up telling the user more info needs to be provided
- If user has already been registered with information provided by the user, error will pop up saying user has already been registered using that information.

Upload File

- Clicking select file allows the user to browse their local file directory to select a file.
- Clicking upload will upload the file to the database. If no file is selected, or file exceeds the max file size, error message will pop up.

User Profile

Click the edit pencil - Allows user to edit username, first name, last name, country and state.

Click the picture - allows user to edit profile picture

Navbar

Click home - Takes user back to the home page

Click Data - Takes user to library page - this is where users can search and view others research

Click Upload - Takes user to upload data page

Click About - Takes user to the about page where user can find out more about the mission and what happens in this library

Data page

Click Search - clicking search will allow user to search under the keywords entered in the bar. If nothing is entered, the page will refresh.

Click file - Pulls up the file that allows the user to then download the file.

Testing Architecture

Regression Testing

Regression testing is software testing that is to confirm that a recent change to the program or code has not adversely affected the existing functions. This test is done to make sure that new code changes should not have any side effects on the existing functionalities. It ensures that the old code still works when the new code changes are added.

The regression testing can be carried out by using techniques such as retest all, regression test selection and prioritization of test cases. In this case retest all will be selected, because the software is not very complicated and there are only a few user cases. The regression testing will be performed at the end of every sprint. Any adverse effect caused by the code changes will be well documented and fixed.

Unit testing

For unit testing we need to check that inputting and displaying usernames are implemented correctly, uploading, displaying and downloading a file works and inputting and displaying user info are all able to be used and interact correctly with the database. To do this, we will need to input fake usernames to make sure these are shown correctly when users login. With files, we will need to upload fake files and make sure these are uploaded to the correct places, shown correctly on their appropriate web pages, and download correctly for users. For user info, we will need to input fake user info and make sure this data is displayed correctly for a user's profile.

User Acceptance Test Scenarios

Login

The user will enter their userID and password

Success - password and userID will match the userID and password in the database.

Result – User will get a message saying successfully logged in

Failure – password or/and userID don't match userID or password in database.

Result – User will get an error message saying that password and/or userID don't match

File Upload

The user will click the upload file tab on the nav bar, a form will display. User will fill out the form and search for a file to upload.

Success – User filled out all the text boxes and file successfully uploaded

Result – message is displayed saying “file successfully uploaded” bring the user back to the main page.

Failure – The User didn’t fill out one or more of the text boxes or the file wasn’t successfully uploaded.

Result – A error message will display saying that the file wasn’t successfully uploaded and it will highlight fields that weren’t successful.

Search

The user clicks the search bar and types in keywords or userID and clicks the search button.

Success – The keywords are used to search the database and results are displayed by relevance.

Result - The main page is populated with the title and a short description of the relevant files. With a clickable link to see the full file and download link.

Failure – The keywords or userID doesn’t match any files or userID in the database.

Result – The main page displays an error saying “nothing matching your search word”

Download File

The user clicks the download link next to a file title displayed on the main page.

Success – Checks user restrictions and if file is shared with all users.

Result – The file starts to download and shows the files download progress on the browser.

Failure – The user doesn’t have permissions to download or the uploader put restrictions on the file.

Result – An error message will display saying that the file can’t be downloaded.

Adding a Comment

The user clicks on the comment tab under a displayed result and they can add a comment up to 500 characters.

Success – First it checks if the user is logged in then the userPermissions is checked to see if the user is not banned from commenting and the text box is checked to see if the comment is not blank and has less than 500 characters

Result – the comment is added to the bottom of the comments

Failure – The user isn’t signed in or their userPermissions is banned from commenting then it checks to see if the text box is blank or has more than 500 characters.

Result – A message is displayed explaining the error that has occurred. Error 1

“Your not signed in. Please Login” Error 2 “Your profile is banned from commenting”

Error 3 “ The text box is blank” Error 4 “ Your comment is too long. Please use less than 500 characters”

Integration Testing

All of these test will be done during developing and before the final deployment. The main concern of these tests are to make sure that the UI and database are integrated seamlessly and that the user can display and upload any research files, documents and comment on others research. Creating an open source research community making researcher be able to reach out to other researchers with ease.

Login/Signup

To test Signup/Login we will create a few different accounts and test if we are able to login with different computers and on different networks to make sure that any user can login or create an account from anywhere they have internet access. We will also make sure that the logins are secure.

Upload/download files

To test if files can be uploaded we will try to upload many different types of files and make sure that when the user adds a title and description of the file that it is stored with the file. Also we will test different file sizes. To test downloading files we will attempt to download files from different browsers and different computers. We will also make sure that the permissions for a user to download a file work for different scenarios. Not allowed and successful.

Search Feature

To test the search features we will add files to the database and search for them by keywords and by userID. We will also try it on different browsers and computers to ensure that it is accessible by any computer and browser. We will test keyword failures like non-matching keywords or userIDs. Also we will make sure that all the fields are displayed correctly. Last we will make sure that the download button displays next to the displayed search.

Comments Feature

To test the comments feature we will create a comment on a post and test if all the failure scenarios can be achieved. First if a user is signed in before posting a comment. Second is the user banned from commenting. Third is the text box blank. Last is the comment over 500 characters. We will test all these scenarios and make sure that the correct error message is displayed.

NavBar

To test the NavBar we will click on all the tabs making sure that they all navigate to the correct pages and forms. We will also test other UI elements like the profile page by making sure that all the features display and function correctly.

Verification Vs. Validation Tests

Verification tests are for verifying user or sensitive information. For example, verify the email address of a user. Validation tests are for verifying that things in the back end of the system are working. For example, make sure a database query is successful.

Login: <http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com/login/>

Git Hub: <https://github.com/ztbc68/OCDX-Engine-Group5>

SWE Team 5 Sprint 2 Kickoff Meeting

Kurt Bognar, Zach Bryant, Luke Jehle, Yi Hou, Hanyu Liang, Kyle Whitney attended at the student center 11/8/16 at 8pm.

Tasks

- Review Milestones
- Delegation (in color)
- Drill-down tasks needed for each milestone.
- Member show on the wiki Completed by linking the wiki to the GitHub Repo Hash that completed the task.
- During Sprint evaluations, evaluation will be performed for both Team and Individual

Task Assignment

Sprint 2 --- Changes from Requirements and Design Doc FEEDBACK need to be incorporated (TURN IN Req. & Design Commented Doc with Sprint 2)

Kurt Liang Yi Zach Kyle Luke

| General | Database | User Interface | Other | Testing | Documentation |
|-------------------------|---|--|--|---|---------------|
| 1. Sprint Documentation | 1. insert 2. update 3. delete queries (files in REPO under a dir DML) | 2. Stub-calls for all interactive elements of UI 3. Begin UI elements | 1. Management of users / roles (User Accounts) | Link to Sprint 1 information **This Document is where you can get Req & Design Points Back, Plus Sprint 2 Points. | |

SQL

Primary Author: Yi

Reviewer: Kurt Bognar

Insert

```
insert into swedb.users values (  
'0001',  
'abc123@example.com',  
'123456',  
'researcher',  
'male',  
'Assistant Professor',  
'University of Missouri',  
'phd',  
'John',  
'Carter',  
'Columbia',  
'MO',  
'65211',  
'US',  
'5731234567',  
0);
```

```
insert into swedb.pages values (  
00001,  
00001,  
{
```

```
    "manifests": {  
        "manifest": {  
            "standardVersions": "ocdxManifest schema v.1",  
            "id": "https://datahub.io/dataset/teahouse-corpus",  
            "creator": "Kristen Schuster",  
            "dateCreated": "2016 - 20 - 04",  
            "comment": "This is an example OCDX manifest created by Krsiten Schuster",  
            "researchObject": {  
                "title": "Teahouse Corpus",  
                "abstract": "The Teahouse corpus is aset of questions asked at the Wikipedia Teahouse, a  
peer support forum for new Wikipedia editors.This corpus contains data from its first two years of operation.",  
                "dates": {  
                    "date": {  
                        "date": "2012 - 02 - 27",  
                        "label": "start"  
                    }  
                }  
            },  
            "privacyEthics": {  
                "oversight": {  
                    "label": "No assertion"  
                }  
            },  
            "informedConsent": "No assertion",  
            "anonymizedData": {  
                "label": "No assertion"  
            },  
            "privacyConsiderations": "No assertion"  
        },  
        "provenance": {  
            "narrative": "The Teahouse started as an editor engagement initiative and Fellowship project.It was  
launched in February 2012 by a small team working with the Wikimedia Foundation.Our intention was to pilot a new, scalable model  
for teaching Wikipedia newcomers the ropes of editing in a friendly and engaging environment. The ultimate goal of the pilot project  
was to increase the retention of new Wikipedia editors(most of whom give up and leave within their first 24 hours post - registration)  
through early proactive outreach.The project was particularly focused on retaining female newcomers, who are woefully  
underrepresented among the regular contributors to the encyclopedia."
```

```

    },
    "publications": {
        "publication": "No assertion"
    },
    "locations": {
        "location": {
            "url": "",
            "comment": ""
        }
    },
    "files": {
        "file": {
            "name": "teahouse - questions20140223.csv"
        },
        "format": ".csv",
        "abstract": "Metadata for 5,003 questions",
        "size": "No assertion",
        "url": "No assertion",
        "checksum": "No assertion"
    },
    "permissions": "No assertion"
},
"dates": {
    "date": {
        "date": "2014 - 02 - 15"
    },
    "label": "Created"
},
"creators": {
    "creator": {
        "name": "Jonathan Morgan",
        "role": {
            "label": "Other"
        }
    },
    "type": {
        "label": "No assertion"
    },
    "contact": "jmorgan@wikimedia.org"
}
},
'00001_update.json',
'00001_archive.json');

```

```

insert into swedb.comment values (
00001,
00002,
00001,
'5/12/2016 12:27',
'This topic is interesting!');

```

```

insert into swedb.stats values (
00001,
'create',
'1/23/2016 16:36',
00001,
NULL,
'00001_update.json',
'00001_archive.json',
NULL,
NULL,
00001,
00006,
NULL,
NULL,
NULL,

```

```
NULL,  
'00001.json');
```

Delete

```
delete from swedb.users where userid = 00001;  
delete from swedb.comment where cid = 00001;  
delete from swedb.pages where pageid = 00001;  
delete from swedb.stats where tid = 00001;
```

Update

```
update swedb.users  
set password = '654321', jobtitle = 'Professor'  
where userid = '00001';
```

```
update swedb.pages  
set json =  
{
```

```
    "manifests": {  
        "manifest": {  
            "standardVersions": "ocdxManifest schema v.1",  
            "id": "https://datahub.io/dataset/teahouse-corpus",  
            "creator": "Kristen Schuster",  
            "dateCreated": "2016 - 20 - 04",  
            "comment": "This is an example ODCX manifest created by Krsiten Schuster",  
            "researchObject": {  
                "title": "Teahouse Corpus",  
                "abstract": "The Teahouse corpus is aset of questions asked at the Wikipedia Teahouse, a  
peer support forum for new Wikipedia editors.This corpus contains data from its first two years of operation.",  
                "dates": {  
                    "date": {  
                        "date": "2012 - 02 - 27",  
                        "label": "start"  
                    }  
                }  
            },  
            "privacyEthics": {  
                "oversight": {  
                    "label": "No assertion"  
                }  
            },  
            "informedConsent": "No assertion",  
            "anonymizedData": {  
                "label": "No assertion"  
            },  
            "privacyConsiderations": "No assertion"  
        },  
        "provenance": {  
            "narrative": "The Teahouse started as an editor engagement initiative and Fellowship project.It was  
launched in February 2012 by a small team working with the Wikimedia Foundation.Our intention was to pilot a new, scalable model  
for teaching Wikipedia newcomers the ropes of editing in a friendly and engaging environment. The ultimate goal of the pilot project  
was to increase the retention of new Wikipedia editors(most of whom give up and leave within their first 24 hours post - registration)  
through early proactive outreach.The project was particularly focused on retaining female newcomers, who are woefully  
underrepresented among the regular contributors to the encyclopedia."  
        },  
        "publications": {  
            "publication": "No assertion"  
        },  
        "locations": {  
            "location": {  
                "url": "",  
                "comment": ""  
            }  
        },  
        "files": {  
            "file": {  
                "name": "teahouse - questions20140223.csv"  
            }  
        },  
    },  
}
```

```

        "format": ".csv",
        "abstract": "Metadata for 5,003 questions",
        "size": "No assertion",
        "url": "No assertion",
        "checksum": "No assertion"
    },
    "permissions": "No assertion"
},
"dates": {
    "date": {
        "date": "2014 - 02 - 15"
    },
    "label": "Created"
},
"creators": {
    "creator": {
        "name": "Jonathan Morgan",
        "role": {
            "label": "Other"
        }
    },
    "type": {
        "label": "No assertion"
    },
    "contact": "jmorgan@wikimedia.org"
}
},
updatedjson =
'{
    "manifests": {
        "manifest": {
            "standardVersions": "ocdxManifest schema v.1",
            "id": "https://datahub.io/dataset/teahouse-corpus",
            "creator": "Kristen Schuster",
            "dateCreated": "2016 - 20 - 04",
            "comment": "This is an example OCDX manifest created by Krsiten Schuster",
            "researchObject": {
                "title": "Teahouse Corpus (updated)",
                "abstract": "The Teahouse corpus is aset of questions asked at the Wikipedia Teahouse, a
peer support forum for new Wikipedia editors.This corpus contains data from its first two years of operation.",
                "dates": {
                    "date": {
                        "date": "2012 - 02 - 27",
                        "label": "start"
                    }
                }
            },
            "privacyEthics": {
                "oversight": {
                    "label": "No assertion"
                }
            },
            "informedConsent": "No assertion",
            "anonymizedData": {
                "label": "No assertion"
            },
            "privacyConsiderations": "No assertion"
        },
        "provenance": {
            "narrative": "The Teahouse started as an editor engagement initiative and Fellowship project.It was
launched in February 2012 by a small team working with the Wikimedia Foundation.Our intention was to pilot a new, scalable model
for teaching Wikipedia newcomers the ropes of editing in a friendly and engaging environment. The ultimate goal of the pilot project
was to increase the retention of new Wikipedia editors(most of whom give up and leave within their first 24 hours post - registration)
through early proactive outreach.The project was particularly focused on retaining female newcomers, who are woefully
underrepresented among the regular contributors to the encyclopedia."
        },

```

```

    "publications": {
      "publication": "No assertion"
    },
    "locations": {
      "location": {
        "url": "",
        "comment": ""
      }
    },
    "files": {
      "file": {
        "name": "teahouse - questions20140223.csv",
        "format": ".csv",
        "abstract": "Metadata for 5,003 questions",
        "size": "No assertion",
        "url": "No assertion",
        "checksum": "No assertion"
      },
      "permissions": "No assertion"
    },
    "dates": {
      "date": {
        "date": "2014 - 02 - 15"
      },
      "label": "Created"
    },
    "creators": {
      "creator": {
        "name": "Jonathan Morgan",
        "role": {
          "label": "Other"
        }
      },
      "type": {
        "label": "No assertion"
      },
      "contact": "jmorgan@wikimedia.org"
    }
  }
}
where userid = '00001';

```


Seed Data

Author: Kurt

```
insert into swedb.comment
(cid,userid,pageid,time,content) values
(default,2,1,'5-1-2016 12:27','This topic is interesting!'),
(default,4,1,'7-2-2016 09:40','This data is so helpful for my research!'),
(default,4,3,'5-8-2016 11:20','I do not think the data is useful.'),
(default,1,3,'3-2-2016 13:38','How does the author validate the results?'),
(default,3,2,'4-1-2016 15:09','Can we conduct similar research in the future?');
insert into swedb.intermediary
(tid, userid) values
(1,8),
(2, 7),
(3, 6),
(4, 5),
(5, 4),
(6, 3),
(7, 2),
(8, 1);
insert into swedb.pages
(pageid,userid,json,updatedjson,archivejson) values
(default,00001,'00001.json','00001_update.json','00001_archive.json'),
(default,00001,'00002.json','00002_update.json','00002_archive.json'),
(default,00003,'00003.json','00003_update.json','00003_archive.json');
insert into swedb.stats
(tid,action,date,pageid,timeonpage,updatedjson,archivejson,accepterid,changerid,requestorid,assignedid,reporterid,reporteeid,cont
nt,reason,json) values
(default,'create','1-23-2016
16:36',1,NULL,'00001_update.json','00001_archive.json',NULL,NULL,1,6,NULL,NULL,NULL,NULL,'00001.json'),
(default,'create','2-17-2016
10:52',2,NULL,'00002_update.json','00002_archive.json',NULL,NULL,1,7,NULL,NULL,NULL,NULL,'00002.json'),
(default,'create','11-3-2015
21:23',3,NULL,'00003_update.json','00003_archive.json',NULL,NULL,3,6,NULL,NULL,NULL,NULL,'00003.json'),
(default,'comment','5-12-2016 12:27',1,NULL,NULL,NULL,NULL,NULL,2,7,NULL,NULL,NULL,NULL,NULL),
(default,'comment','7-2-2016 9:40',1,NULL,NULL,NULL,NULL,NULL,4,6,NULL,NULL,NULL,NULL,NULL),
(default,'comment','5-8-2016 11:20',3,NULL,NULL,NULL,NULL,NULL,4,7,NULL,NULL,NULL,NULL,NULL),
(default,'comment','3-22-2016 13:38',3,NULL,NULL,NULL,NULL,NULL,1,6,NULL,NULL,NULL,NULL,NULL),
(default,'comment','4-17-2016 15:09',2,NULL,NULL,NULL,NULL,NULL,3,7,NULL,NULL,NULL,NULL,NULL);
insert into swedb.users
(userid,email,password,usertype,gender,jobtitle,org,edlevel,fname,lname,city,state,zip,country,phone,banstatus) values
(default,'abc123@example.com','123456','researcher','male','Assistant Professor','University of
Missouri','phd','John','Carter','Columbia','MO',65211,'US',5731234567,0),
(default,'def456@example.com','584397','researcher','male','Postdoc','University of Michigan','phd','Henry','Brown','Ann
Arbor','MI',48106,'US',7341234567,0),
(default,'ghi789@example.com','gh54893','researcher','female','Professor','University of
California-Berkley','phd','Jake','Musik','Berkley','CA',94701,'US',5101234567,0),
(default,'Abc123@example.com','111111','independent','female','Software Engineer','Facebook','MS','Paige','Loftin','San
Jose','CA',95109,'US',4081234567,0),
(default,'worldpeace@example.com','dh8g8j6','independent','male','Data scientist','LinkedIn','MS','Kevin','Berry','San
Jose','CA',95109,'US',4083421883,1),
(default,'admin1@example.com','Pa55word','admin','male','Software Engineer','University of
Missouri','BS','Gilbert','Young','Columbia','MO',65201,'US',5736543210,0),
(default,'admin2@example.com','pa55word','admin','female','Software Engineer','wikipedia','BS','Jennifer','Nemmers','San
Francisco','CA',94101,'US',4156543210,0);
```

HTML Redirect

Author: Zach

```
<!DOCTYPE html>
<html>
<!--Zach Bryant - Redirect to login page - 11/10/2016-->
<head>
    <meta http-equiv="refresh" content="0;
URL='http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com/login/login_in.php'"/>
```

```
<title>Login</title>
</head>
<body>
</body>
</html>
```

Stub Calls

Example:

```
String printf(string x){return x}
```

Need to do:

```
Login -GET    /login(.:format)    sessions#new  
        POST  /login(.:format)    sessions#create
```

```
Upload - DataFile.save_file(params[:upload])
```

```
Download - send_file("SOURCE", filename: "", type: "")
```

```
Search - Post.search(params[:search]).order("created_at DESC")
```

```
Sign up - :user, :url => {:controller => 'users', :action => 'create'}
```

```
Sign out - delete 'logout' => :destroy
```

```
Comment - Post.comment(params[:comment]).order("created_at  
Comment")
```

```
Report - User.report(params[:username])
```

```
Ban - User.ban(action[:banned])
```

Completed Tasks

Kurt

Sprint Documentation:

Input Seed data:

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/1454ccbc7f11fef1d13dc3200a03473b980a746e>

Change log

Liang

Login in page:

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/957ea18a56aa5cb7f3cd3cd08cdd2f9ddd1bc96>

Website:

http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com/login/login_in.php

Sign up page:

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/262825a9dbab6d9ffa8ac199d9cf7c891759b27f>

Website:

http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com/login/sign_up.php

Yi

Insert

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/38ebe8a156e4ab65dcd93b415141083162a0998>

update

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/24bba0b11851281a5c8ae504dfea7800bae5bc2b>

Delete

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/51226f5b5184338b948761dec292600445f50723>

Zach

Github organization

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/9ebd11abab56508f7d0da7424940a8ac42e7df70>

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/767ba8de310b39a255308f137e0eab1be24bc9f5>

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/dd68e656f69f546347b276c1f2c4602b668b4c68>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/138bdd34751afb5b12e093fd89384a7ec9128ad>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/51d93f5111881e59cc052e6aba14030ab8ad5748>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/7f09cab623a44808535c1fd22f79e9570b607022>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/79c6b60ff0e7c6f3c4cf17c13e33bb1d9f82ce42>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/93c9997b687735ee652820a9cbfc8f175705b74b>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/0e85a7a15e212fe2f17c36157eeaf19c15615aa3>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/32ea4f9ba72f4214a69f96f085a0b2aea656338a>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/cf7436739eb3c3fbb96ef1451ef25487986df193>
<https://github.com/ztbc68/OCDX-Engine-Group5/commit/e18fbda47f60887ad38427900f1827fd6c1aaae2>

HTML redirect page

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/ec92046bbc77b96185aa83670f097bf0fd10a945>

Stub calls (first 3) - page 30 this document

Kyle

Stub calls(last 3) - page 30 this document

Began work on main page of UI

Luke

Stub calls - page 30 this document

Began work on upload page - be7ef2b25683d7f67914a0bbe6aac21a31e5d8f0

Github Repo: <https://github.com/ztbc68/OCDX-Engine-Group5>

Website: <http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com>

SWE Team 5 Sprint 3 Kickoff Meeting

Kurt Bognar, Zach Bryant, Luke Jehle, Yi Hou, Hanyu Liang, Kyle Whitney attended at the student center 11.14.16 at 7pm.

Tasks

- Review Milestones
- Delegation
- Drill-down tasks needed for each milestone.
- Member show on the wiki Completed by linking the wiki to the GitHub Repo Hash that completed the task.
- During Sprint evaluations, evaluation will be performed for both Team and Individual aspects.

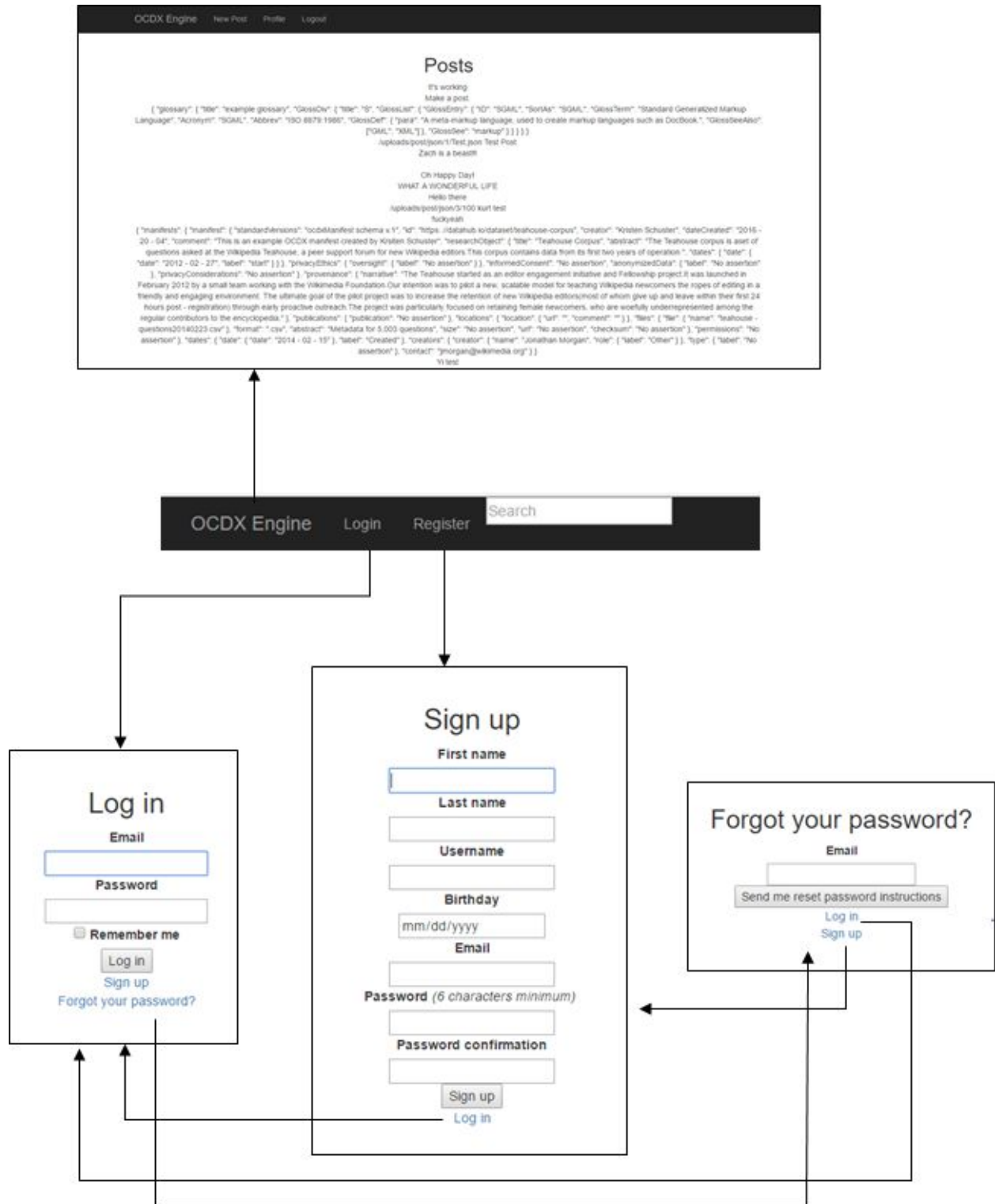
Task Assignment

Kurt **Liang** **Yi** **Zach** **Kyle** **Luke**

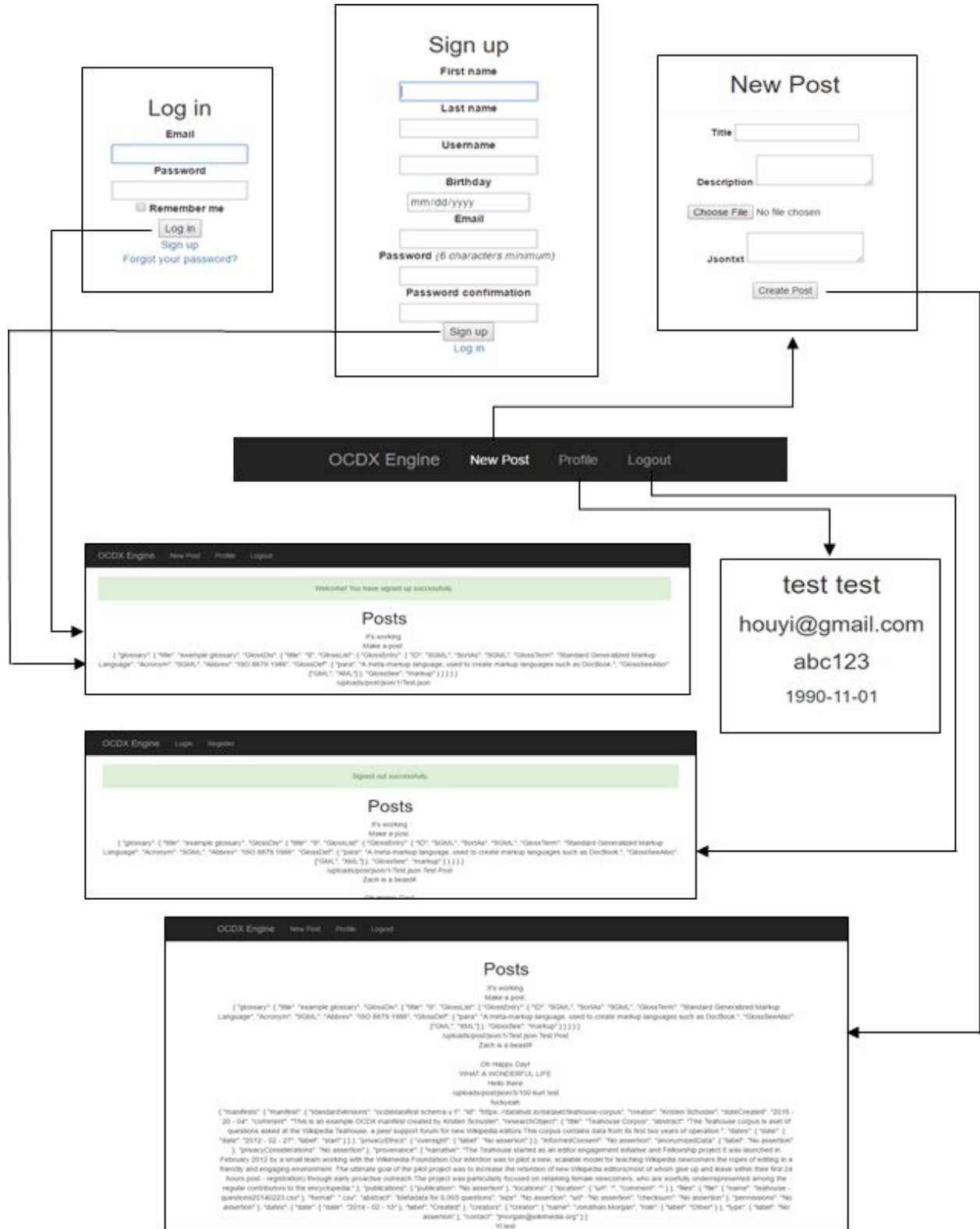
| General | User Interface (UI team) | Other | Testing | User Documentation |
|-------------------------|--|---|--|--|
| 1. Sprint Documentation | 2. Clean/Styling of UI 3. Additional UI elements for New Features (these are all demonstrated on the website, no documentation besides pics) | 4. Temporal logic for application window (clicking) | 5. Think through a description of the unit tests. What would your edge cases be to try and test? | 6. Draft structure of user documentation |

Temporal Logic

Before login or sign in



After login or sign in



Testing Architecture

Regression Testing (verification)

Regression testing is software testing that is to confirm that a recent change to the program or code has not adversely affected the existing functions. This test is done to make sure that new code changes should not have any side effects on the existing functionalities. It ensures that the old code still works when the new code changes are added.

The regression testing can be carried out by using techniques such as retest all, regression test selection and prioritization of test cases. In this case retest all will be selected, because the software is not very complicated and there are only a few user cases. The regression testing will be performed at the end of every sprint. Any adverse effect caused by the code changes will be well documented and fixed.

Integration Testing (Verification)

All of these test will be done during developing and before the final deployment. The main concern of these tests are to make sure that the UI and database are integrated seamlessly and that the user can display and upload any research files, documents and comment on others research. Creating an open source research community making researcher be able to reach out to other researchers with ease.

Login

To test Signup/Login we will create a few different accounts and test if we are able to login with different computers and on different networks to make sure that any user can login or create a account from anywhere they have internet access. We will also make sure that the logins are secure.

The user will enter their userID and password

Success - password and userID will match the userID and password in the database.

Result – User will get a message saying successfully logged in

Failure – password or/and userID don't match userID or password in database.

Result – User will get an error message saying that password and/or userID don't match

File Upload

To test if files are can be uploaded we will try to upload many different types of files and make sure that when the user adds a title and description of the file that it is stored with the file. Also we will test different file sizes. To test downloading files we will attempt to download files from different browsers and different computers. We will also make sure that the permissions for a user to download a file works for different scenarios. Not allowed and successful.

The user will click the upload file tab on the nav bar, a form will display. User will fill out the form and search for a file to upload.

Success – User filled out all the text boxes and file successfully uploaded

Result – message is displayed saying “file successfully uploaded” bring the user back to the main page.

Failure – The User didn’t fill out one or more of the text boxes or the file wasn’t successfully uploaded.

Result – A error message will display saying that the file wasn’t successfully uploaded and it will highlight fields that weren’t successful.

Download File

The user clicks the download link next to a file title displayed on the main page.

Success – Checks user restrictions and if file is shared with all users.

Result – The file starts to download and shows the files download progress on the browser.

Failure – The user doesn’t have permissions to download or the uploader put restrictions on the file.

Result – An error message will display saying that the file can’t be downloaded.

Search

To test the search features we will add files to the database and search for them by keywords and by userID. We will also try it on different browsers and computers to ensure that its accessible by any computer and browser. We will test keywords failures like non matching keywords or userIDs. Also we will make sure that all the fields are displayed correctly. Last we will make sure that the download button displays next to the displayed search.

The user clicks the search bar and types in keywords or userID and clicks the search button.

Success – The keywords are used to search the database and results are displayed by relevance.

Result - The main page is populated with the title and a short description of the relevant files. With a clickable link to see the full file and download link.

Failure – The keywords or userID doesn’t match any files or userID in the database.

Result – The main page displays an error saying “nothing matching your search word”

Adding a Comment

To test the comments feature we will create a comment on a post and test if the all the failure scenarios can be achieved. First if a user is signed in before posting a comment. Second is the user banned from commenting. Third is the text box blank. Last is the comment over 500 characters. We will test all these scenarios and make sure that the correct error message is displayed.

The user clicks on the comment tab under a displayed result and they can add a comment up to 500 characters.

Success – First it checks if the user is logged in then the userPermissions is checked to see if the user is not banned from commenting and the text box is checked to see if the comment is not blank and has less than 500 characters

Result – the comment is added to the bottom of the comments

Failure – The user isn't signed in or their userPermissions is banned from commenting then it checks to see if the text box is blank or has more than 500 characters.

Result – A message is displayed explaining the error that has occurred. Error 1 "Your not signed in. Please Login" Error 2 "Your profile is banned from commenting" Error 3 " The text box is blank" Error 4 " Your comment is too long. Please use less than 500 characters"

NavBar

To Test the NavBar we will click on all the tabs making sure that they all navigate to the correct pages and forms. We will also test other UI elements like the profile page by making sure that all the features display and function correctly.

Unit testing (Verification)

For unit testing, we need to check that inputting and displaying usernames are implemented correctly, uploading, displaying and downloading a file works and inputting and displaying user info are all able to be used and interact correctly with the database. To do this, we will need to input fake usernames to make sure these are shown correctly when users login. With files, we will need to upload fake files and make sure these are uploaded to the correct places, shown correctly on their appropriate web pages, and download correctly for users. For user info, we will need to input fake user info and make sure this data is displayed correctly for a user's profile.

Login/Register

Proper registration adds to the database. Proper login makes a session variable with the proper usertype for permissions purposes (user, admin, researcher). All failures should simply prompt the user to try again and also tell them what they messed up.

Failures: Try to a register a user without any fields, without a username, without a password, and without an email. Try to register with the fields extremely long (more than 500 letters). Try to put letters in number fields. Try to login with a good username and wrong password, wrong username and good password, and wrong username and wrong password.

Upload

Proper upload should yield success dialog and also a database entry. Failures should yield a failure dialog and tell the user their problem. Try uploading an empty file. Try uploading a 10 gb file. Try uploading a different (.jpg) file. Try uploading with long (1000) title and description fields.

User Acceptance Test Scenarios (Validation)

UAT is done by the stakeholders. The end user, a TA or someone in the lab, will use the site using the documentation and then give feedback from them. This will be one of the last things we do. Test data will be in the system and also a test case to practice with will be given to the user. They will be asked to browse/search through the test data,

comment, download a file, and upload the given test case. They will then navigate the pages freely and give feedback on the flow of site.

Verification Vs. Validation Tests

Validation: – Does the software meet the customer's expectations? Is this the right piece of software for the requested job

Our validation testing will be demoing the software in before the final grading with the TA's so we can make sure that we have done what is asked(UAT).

Verification: – Does the software meet the stated functional and non-functional requirements? Does it meet the initial specs?

In order to do verification tests, we need to make sure that users can login/out to the system. We also need to make sure that they can upload files and JSON text. We also need to make sure that these "posts" are editable and that they can be searched through and commented on. (All testing besides UAT)

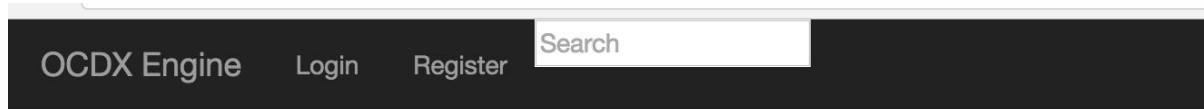
User Documentation

Description:

a metadata specification (instantiated as an OCDX manifest) and infrastructure for storing, describing, analyzing, and sharing datasets for making sense of online human interactions. This work will enable scientists and citizens to answer fundamental questions about our society as it exists on the Internet.

Before login in:

Navigation bar1:



1.Choose the page you want to go in here through clicking appropriate button.

“OCDX Engine” -> post page.

“Login”-> login-in page.

“Register”-> sign-up page.

2.Direct enter the information that you want to search into search space,and click search button.

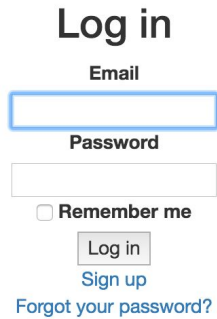
It will go to search page and show information that you want to search in the search page.

OCDX page:

Posts

we will post information that we want to tell to persons in this page.

Login_in page:



The login form is titled "Log in". It contains two input fields: "Email" and "Password". Below the "Password" field is a checkbox labeled "Remember me". At the bottom of the form are three buttons: "Log in", "Sign up", and "Forgot your password?".

Log in

Email

Password

☐ Remember me

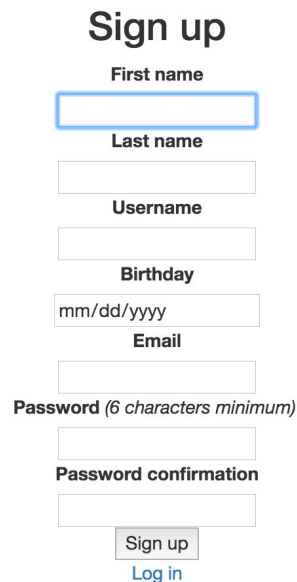
Log in

Sign up

[Forgot your password?](#)

1. Enter your personal email and password, check it correct. And click login in button.
(if you are lazy and do not want to enter the e-mail and password again, click "Remember me")
2. If you forget password, click "forgot your password?". The password will automatically be sent to your e-mail.
3. If you do not register before, click "sign up". You will go to register page and can create your account in the register page.

Register page:



The sign up form is titled "Sign up". It contains several input fields: "First name", "Last name", "Username", "Birthday" (with a placeholder "mm/dd/yyyy"), "Email", "Password (6 characters minimum)", and "Password confirmation". At the bottom are two buttons: "Sign up" and "Log in".

Sign up

First name

Last name

Username

Birthday

mm/dd/yyyy

Email

Password (6 characters minimum)

Password confirmation

Sign up

[Log in](#)

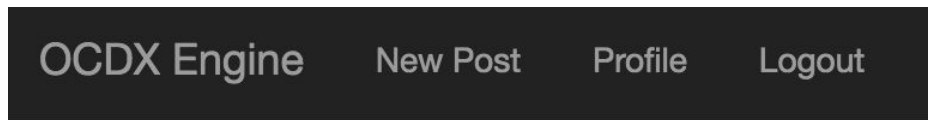
Create your account in this page.

1. enter your First name and Last name.
2. enter your username (This is the your name that let others look in this system.)

- ps: hope the username is not same as others,so create a unique username.
- 3.enter your birthday.
 - 4.enter your e-mail: this is your account when you login in.
 - 5.enter your password:at least 6 characters.
 - 6.enter your password again: must be same as password you first enter.
 - 7.check the information you enter. If think is correct, click “sign up” button.
 - 8.when finish your register and want to enter this system as soon as possible,click “login in” button and go to login-in page.

After login in:

Navigation bar2:



Choose the page you want to go in here through clicking appropriate button.

“OCDX Engine” -> post page.

“New Post”->NewPost page.

“Profile”->profile page.

“Loginout”->Logout page.

OCDX page:

Posts

we will post information that we want to tell to persons in this page.(same as OCDX page before login in)

NewPost page:

New Post

Title

Description

Choose File

No file chosen

Json txt

Create Post

This page is for your new post.

1. enter the title.
2. enter description of your new post.
3. can choose file you want to upload if you want to do.
4. enter Json txt.
5. check the information that you enter and click "Create Post" button, your new post will show in the OCXD page.

Profile page:

abc abc
aaa@bbb
lhy
1994-08-13

This page will show your all personal information. (like example above).

Login out page:

Signed out successfully.

Posts

When click “Login out “ button, it will go to OCDX page before login in. And it will tell you signed out successfully. If you want to do something, login in again. If not,you can close the system.

UI Code (Done with Ruby on Rails)

Home Page(Template for other pages as well)

Author: Zach Bryant, Kyle Whitney and Luke Jehle

File: OCDX_Engine002/app/views/layouts/application.html.erb

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>OCDX Engine</title>
  <%= stylesheet_link_tag "application", media: "all", "data-turbolinks-track" => true %>
  <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  <%= csrf_meta_tags %>

</head>
<body>

  <!-- Navigation -->
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
      <!-- Brand and toggle get grouped for better mobile display -->
      <div class="navbar-header">

        <%= link_to "OCDX Engine", root_url, class: "navbar-brand" %>

      </div>
      <!-- Collect the nav links, forms, and other content for toggling -->
      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav">
          <%- if current_user %>
            <li>
              <%= link_to "New Post", new_post_path %>
            </li>
            <li>
              <%= link_to "Profile", user_path(current_user.id), method: :get %>
            </li>
            <li>
              <%= link_to "Logout", destroy_user_session_path, method: :delete, confirm: "Are
you sure?" %>
            </li>
            <li>
              <%= form_tag search_project_path do %>

                <input class="navbar-form navbar-right search full-width" id="searchbar" name="query"
placeholder="Search" type="text" autocomplete="off">

              </div>
            </li>
          </%->
        </ul>
      </div>
    </div>
  </nav>

</body>
</html>
```

```

        <% end %>
        </li>

        <% else %>
        <li>

            <%= link_to "Login", new_user_session_path %>

        </li>
        <li>

            <%= link_to "Register", new_user_registration_path %>

        </li>
        <% end %>
        <li>

            <%= form_tag search_project_path do %>

                <input class="navbar-form navbar-right search full-width" id="searchbar" name="query"
placeholder="Search" type="text" autocomplete="off">

            <% end %>
        </li>

    </ul>
</div>
<!-- /.navbar-collapse -->
</div>
<!-- /.container -->
</nav>

<!-- Page Content -->
<div class="container">

    <div class="row">
        <div class="col-lg-12 text-center">
            <% flash.each do |a, b| %>
                <div class="alert alert-success"> <%= b %> </div>
            <% end %>
            <%= yield %>
        </div>
    </div>
    <!-- /.row -->

</div>
<!-- /.container -->

<style>body{padding-top:70px;}</style>

</body>

```

```
</html>
```

New Post

Author: Zach Bryant, Luke Jehle

File: OCDX_Engine002/app/views/posts/new.html.erb

```
<h1>New Post</h1>
```

```
<hr>
```

```
<div align="center"><%= form_for (@post) do |f| %>
  <div class="field">
    <%= f.label :title %>
    <%= f.text_field :title %>
  </div>
  <br>
  <div class="field">
    <%= f.label :description %>
    <%= f.text_area :description %>
  </div>
  <br>
  <div class="field">
    <%= f.file_field :json %>
    <%= f.hidden_field :json_cache %>
  </div>
  <br>
  <div class="field">
    <%= f.label :jsonTxt %>
    <%= f.text_area :jsonTxt %>
  </div>
  <br>

  <div class="actions">
    <%= f.submit %>
  </div>
<% end %></div>
```

Posts Index

Author: Zach Bryant

File: OCDX_Engine002/app/views/posts/index.html.erb

```
<h1>Posts</h1>
```

```
<% @posts.each do |x| %>
```

```
  <p>Title: <%= x.title %>
```

```
  </p>
```

```
  <p>Description: <%= x.description %>
```

```

    </p>
    <p>JSON Text: <%= x.jsonTxt %>
  </p>
  <p>JSON File: <%= x.json %>
</p>
<hr>
<% end %>

```

Show Post Page

Author: Zach Bryant

File: OCDX_Engine002/app/views/posts/show.html.erb

```

<h1><%= @post.title %>
<h2><%= @post.description %></h2>
<h2><%= @post.jsonTxt %></h2>
<h3><%= @post.json %></h3>

```

User Profile Page

Author: Zach Bryant

File: OCDX_Engine002/app/views/users/show.html.erb

```

<h1><%= @user.first_name %> <%= @user.last_name %></h1>
<h2><%= @user.email %></h2>
<h2><%= @user.username %></h2>
<h3><%= @user.birthday %></h3>

```

```

<% @user_posts.each do |post| %>
  <%= link_to post.title, post_path(post) %>
<% end %>

```

User Sign-up

Author: Zach Bryant

File: OCDX_Engine002/app/views/devise/registrations/new.html.erb

```

<h2>Sign up</h2>

<%= form_for(resource, as: resource_name, url: registration_path(resource_name)) do |f| %>
  <%= devise_error_messages! %>
  <div class="field">
    <%= f.label :first_name %><br />
    <%= f.text_field :first_name, autofocus: true %>
  </div>

  <div class="field">
    <%= f.label :last_name %><br />
    <%= f.text_field :last_name, autofocus: true %>
  </div>

```

```

<div class="field">
  <%= f.label :username %><br />
  <%= f.text_field :username, autofocus: true %>
</div>

```

```

<div class="field">
  <%= f.label :birthday %><br />
  <%= f.date_field :birthday, autofocus: true %>
</div>

```

```

<div class="field">
  <%= f.label :email %><br />
  <%= f.email_field :email, autofocus: true %>
</div>

```

```

<div class="field">
  <%= f.label :password %>
  <% if @minimum_password_length %>
  <em>(<%= @minimum_password_length %> characters minimum)</em>
  <% end %><br />
  <%= f.password_field :password, autocomplete: "off" %>
</div>

```

```

<div class="field">
  <%= f.label :password_confirmation %><br />
  <%= f.password_field :password_confirmation, autocomplete: "off" %>
</div>

```

```

<div class="actions">
  <%= f.submit "Sign up" %>
</div>
<% end %>

```

```

<%= render "devise/shared/links" %>

```

User Login

Author: Zach Bryant

File: OCDX_Engine002/app/views/devise/sessions/new.html.erb

```

<h2>Log in</h2>

```

```

<%= form_for(resource, as: resource_name, url: session_path(resource_name)) do |f| %>
  <div class="field">
    <%= f.label :email %><br />
    <%= f.email_field :email, autofocus: true %>

```

```
</div>
```

```
<div class="field">
```

```
<%= f.label :password %><br />
```

```
<%= f.password_field :password, autocomplete: "off" %>
```

```
</div>
```

```
<% if devise_mapping.rememberable? -%>
```

```
<div class="field">
```

```
<%= f.check_box :remember_me %>
```

```
<%= f.label :remember_me %>
```

```
</div>
```

```
<% end -%>
```

```
<div class="actions">
```

```
<%= f.submit "Log in" %>
```

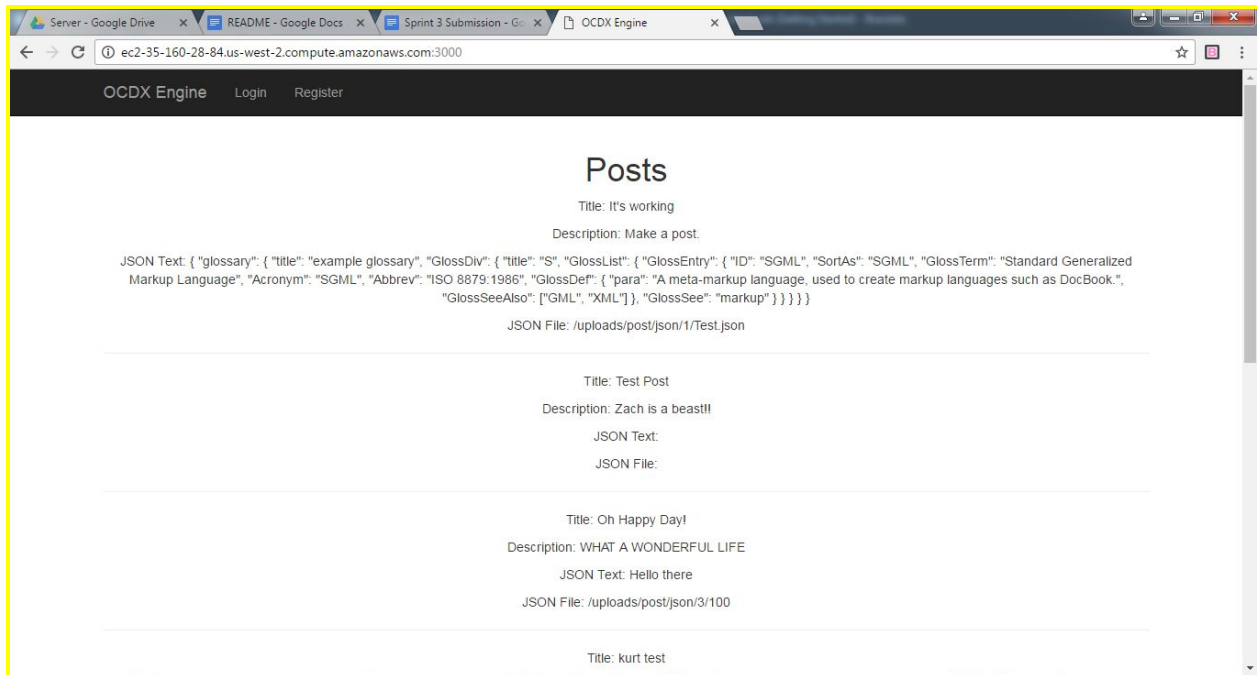
```
</div>
```

```
<% end %>
```

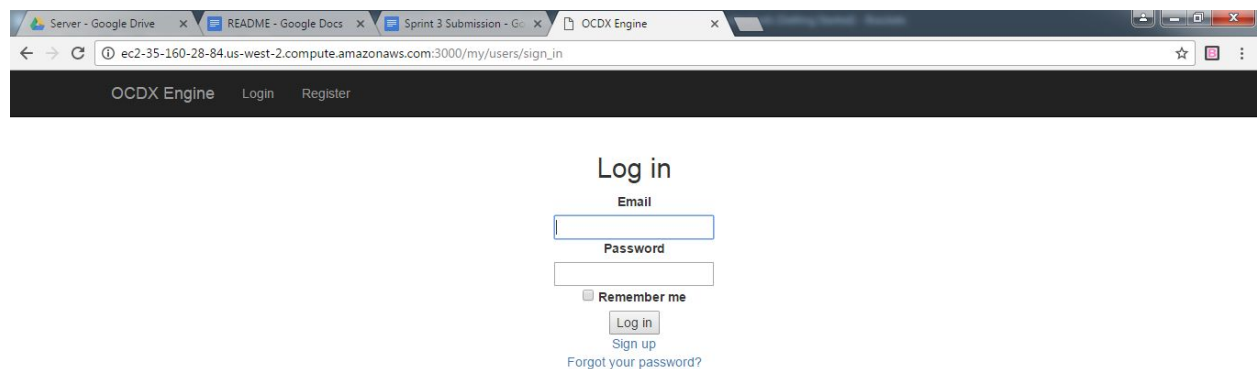
```
<%= render "devise/shared/links" %>
```

UI Pics

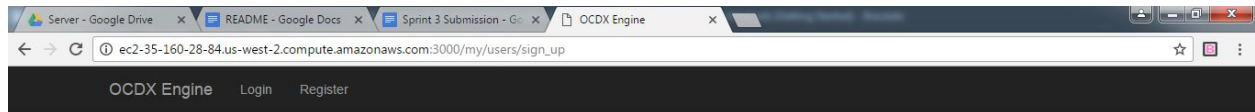
Home Page



Login Page



Register



Sign up

First name

Last name

Username

Birthday

Email

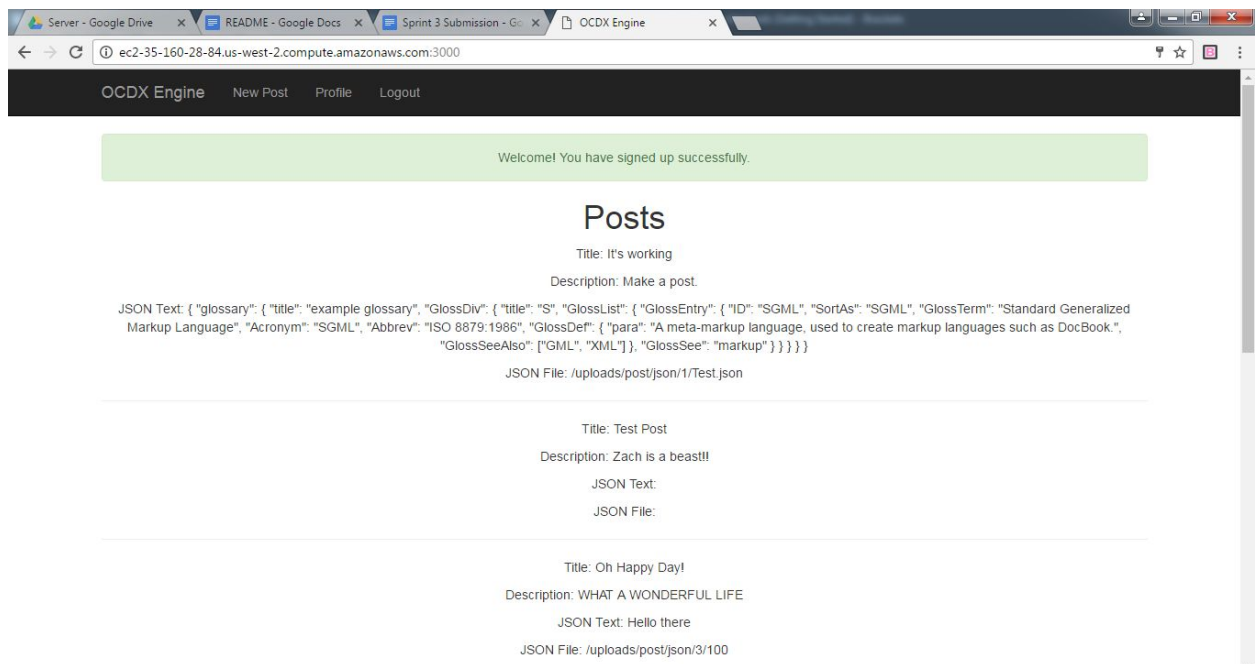
Password (6 characters minimum)

Password confirmation

Sign up

Log in

After Login



Posts

Title: It's working

Description: Make a post.

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

JSON File: /uploads/post/json/1/Test.json

Title: Test Post

Description: Zach is a beast!!!

JSON Text:

JSON File:

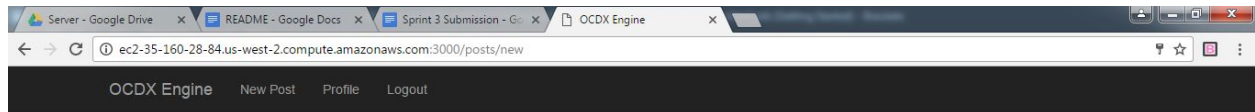
Title: Oh Happy Day!

Description: WHAT A WONDERFUL LIFE

JSON Text: Hello there

JSON File: /uploads/post/json/3/100

New Post



New Post

Title

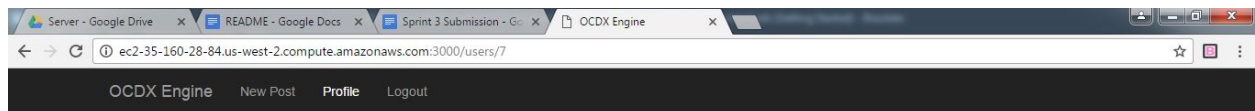
Description

Choose File No file chosen

Jsontxt

Create Post

Profile



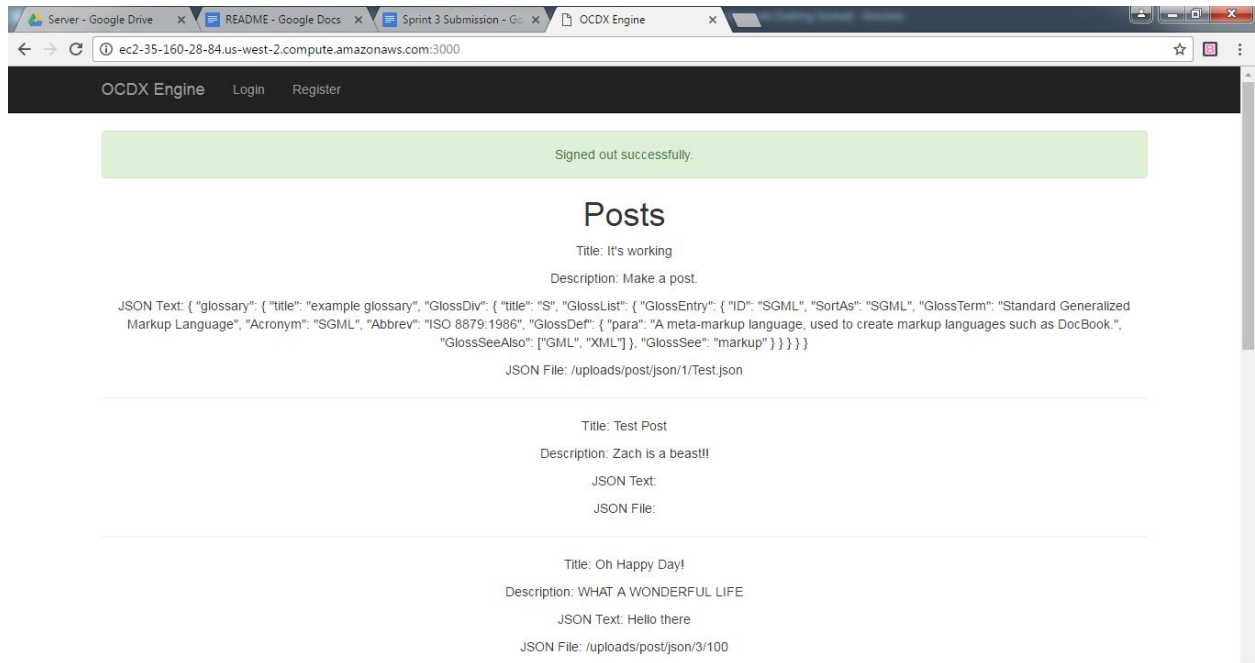
asdfsdf asfsadf

qwerty@asdf.com

asdfsadf

2016-11-01

After Logout



We are not doing any **VISUALIZATIONS.**

Completed Tasks

Kurt:

- Sprint Documentation
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/e5aac9b37d6d0fc5ee34a2d7215b70698365ec2f>
- Change log
- Testing architecture

Yi:

- Updated database scripts
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/fda24f29fc6b0f7426b179331aaca1971315178b>
 - <https://github.com/ztbc68/OCDX-Engine-Group5/commit/203ad53968412efd8767dcc87672ddeccb54d99c>
- Documented temporal logic
- Updated requirement analysis

Liang:

Draft structure of user documentation

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/401bebbf5e9fb28851e0638577914bd25800233b>

Luke:

UI work - See UI section of this document (pgs. 46-51)

Editing the Gemfile:

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/4f267651811d6f1c20d9785692bbf9386b25c34c>

Kyle:

Created search bar:

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/4f267651811d6f1c20d9785692bbf9386b25c34c>

Zach:

Uploaded Ruby on Rails code to github:

<https://github.com/ztbc68/OCDX-Engine-Group5/commit/4f267651811d6f1c20d9785692bbf9386b25c34c>

UI work - See UI section of this document (pgs. 46-51)

SWE Team 5 Sprint 4 Kickoff Meeting

Kurt Bognar, Zach Bryant, Luke Jehle, Yi Hou, Hanyu Liang, Kyle Whitney attended at Strickland Hall 11.28.16 at 7pm.

Tasks

- Review Milestones
- Delegation
- Drill-down tasks needed for each milestone.
- Member show on the wiki Completed by linking the wiki to the GitHub Repo Hash that completed the task.
- During Sprint evaluations, evaluation will be performed for both Team and Individual aspects.

Task Assignment

Kurt **Liang** **Yi** **Zach** **Kyle** **Luke**

Sprint 4

| General | User Interface | Other | Testing | Deployment Instructions |
|--|------------------------|------------------------|--|--|
| 1. Sprint Documentation 2. Deploy to Group account 3. You should have a script that allows automated deployment from your GitHub Repo to a deployment environment. | 6. Clean up / catch up | 4. Clean up / catch up | Elaborate on test cases from Sprint 1 and 3. | Complete user documentation Your "Readme.MD" file on Github should clearly explain how your application will be deployed. |

Testing Architecture

Regression Testing

Regression testing is software testing that is to confirm that a recent change to the program or code has not adversely affected the existing functions. This test is done to make sure that new code changes should not have any side effects on the existing functionalities. It ensures that the old code still works when the new code changes are added.

The regression testing can be carried out by using techniques such as retest all, regression test selection and prioritization of test cases. In this case retest all will be selected, because the software is not very complicated and there are only a few user cases. The regression testing will be performed at the end of every sprint. Any adverse effect caused by the code changes will be well documented and fixed.

Unit testing

For unit testing since we are using Ruby on Rails we cannot separate out the individual functions. However, we need to check that inputting and displaying usernames are implemented correctly, uploading, displaying and downloading a file works and inputting and displaying user info are all able to be used and interact correctly with the database. To do this, we will need to input fake usernames to make sure these are shown correctly when users login. With files, we will need to upload fake files and make sure these are uploaded to the correct places, shown correctly on their appropriate web pages, and download correctly for users. For user info, we will need to input fake user info and make sure this data is displayed correctly for a user's profile.

User Acceptance Test Scenarios

Login

The user will enter their userID and password

Success - password and userID will match the userID and password in the database.

Result – User will get a message saying successfully logged in

Failure – password or/and userID don't match userID or password in database.

Result – User will get an error message saying that password and/or userID don't match

File Upload

The user will click the upload file tab on the nav bar, a form will display. User will fill out the form and search for a file to upload.

Success – User filled out all the text boxes and file successfully uploaded

Result – message is displayed saying “file successfully uploaded” bring the user back to the main page.

Failure – The User didn't fill out one or more of the text boxes or the file wasn't successfully uploaded.

Result – A error message will display saying that the file wasn't successfully uploaded and it will highlight fields that weren't successful.

Search

The user clicks the search bar and types in keywords or userID and clicks the search button.

Success – The keywords are used to search the database and results are displayed by relevance.

Result - The main page is populated with the title and a short description of the relevant files. With a clickable link to see the full file and download link.

Failure – The keywords or userID doesn't match any files or userID in the database.

Result – The main page displays an error saying "nothing matching your search word"

Download File

The user clicks the download link next to a file title displayed on the main page.

Success – Checks user restrictions and if file is shared with all users.

Result – The file starts to download and shows the files download progress on the browser.

Failure – The user doesn't have permissions to download or the uploader put restrictions on the file.

Result – An error message will display saying that the file can't be downloaded.

Adding a Comment

The user clicks on the comment tab under a displayed result and they can add a comment up to 500 characters.

Success – First it checks if the user is logged in then the userPermissions is checked to see if the user is not banned from commenting and the text box is checked to see if the comment is not blank and has less than 500 characters

Result – the comment is added to the bottom of the comments

Failure – The user isn't signed in or their userPermissions is banned from commenting then it checks to see if the text box is blank or has more than 500 characters.

Result – A message is displayed explaining the error that has occurred. Error 1

"Your not signed in. Please Login" Error 2 "Your profile is banned from commenting"

Error 3 " The text box is blank" Error 4 " Your comment is too long. Please use less than 500 characters"

Integration Testing

All of these test will be done during developing and before the final deployment. The main concern of these tests are to make sure that the UI and database are integrated seamlessly and that the user can display and upload any research files, documents and comment on others research. Creating an open source research community making researcher be able to reach out to other researchers with ease.

Login/Signup

To test Signup/Login we will create a few different accounts and test if we are able to login with different computers and on different networks to make sure that any user can login or create a

account from anywhere they have internet access. We will also make sure that the logins are secure.

Upload/download files

To test if files can be uploaded we will try to upload many different types of files and make sure that when the user adds a title and description of the file that it is stored with the file. Also we will test different file sizes. To test downloading files we will attempt to download files from different browsers and different computers. We will also make sure that the permissions for a user to download a file works for different scenarios. Not allowed and successful.

Search Feature

To test the search features we will add files to the database and search for them by keywords and by userID. We will also try it on different browsers and computers to ensure that it is accessible by any computer and browser. We will test keywords failures like non matching keywords or userIDs. Also we will make sure that all the fields are displayed correctly. Last we will make sure that the download button displays next to the displayed search.

Comments Feature

To test the comments feature we will create a comment on a post and test if all the failure scenarios can be achieved. First if a user is signed in before posting a comment. Second is the user banned from commenting. Third is the text box blank. Last is the comment over 500 characters. We will test all these scenarios and make sure that the correct error message is displayed.

NavBar

To Test the NavBar we will click on all the tabs making sure that they all navigate to the correct pages and forms. We will also test other UI elements like the profile page by making sure that all the features display and function correctly.

Verification Vs. Validation Tests

Verification tests are for verifying user or sensitive information. For example, verify the email address of a user. Validation tests are for verifying that things in the back end of the system are working. For example, make sure a database query is successful.

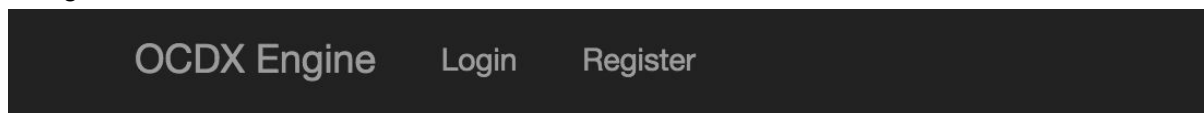
Test Scripts: We cannot include scripts at this time as we are just testing as we build.

User Manual:

Description: a metadata specification (instantiated as an OCDX manifest) and infrastructure for storing, describing, analyzing, and sharing datasets for making sense of online human interactions. This work will enable scientists and citizens to answer fundamental questions about our society as it exists on the Internet.

Before login in:

Navigation bar1:



Choose the page you want to go in here through clicking appropriate button.

"OCDX Engine" -> OCDX page.

"Login"-> login-in page.

"Register"-> sign-up page.

OCDX page:

| Posts | |
|-----------------------------------|----------------------|
| Title: testingK | |
| Description: asdf | |
| JSON File: | File |
| Post Link: | Post |
| | |
| Title: kurt_test3 | |
| Description: this is another test | |
| JSON File: | File |
| Post Link: | Post |
| | |
| Title: kurt_test2 | |
| Description: this is a test | |
| JSON File: | File |
| Post Link: | Post |

we will post information that we want to tell to persons in this page.

```
{
  "widget": {
    "debug": "on",
    "window": {
      "title": "Sample Konfabulator Widget",
      "name": "main_window",
      "width": 500,
      "height": 500
    },
    "image": {
      "src": "Images/Sun.png",
      "name": "sun1",
      "hOffset": 250,
      "vOffset": 250,
      "alignment": "center"
    }
  }
}
```

Title: testingK

Description: asdf

Data:

JSON: [File](#)

Created At: November 27 2016, 10:42pm

when click “file” button,it will go to file page that shows the content of file.

when click “ Post” button, it will show the title,description ,data and time. It also have a “file” button. this button is same as “file” button above.

Login_in page:



Log in

Email

Password

☐ Remember me

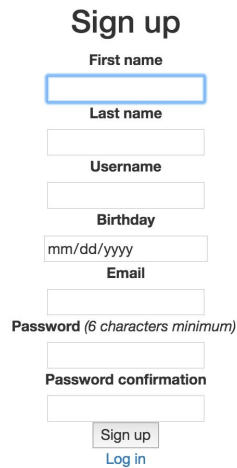
Log in

[Sign up](#)

[Forgot your password?](#)

- 1.Enter your personal email and password, check it correct. And click login in button.
(if you are lazy and do not want to enter the e-mail and password again,click “Remember me”)
- 2.If you forget password, click “forgot your password?” . the password will automatically sent to your e-mail.
3. If you do not register before, click ”sign up” . You will go to register page and can create your account in the register page.

Register page:



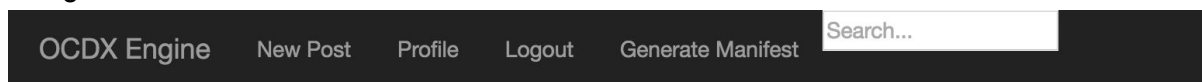
The image shows a 'Sign up' form with the following fields and labels: 'First name' (text input), 'Last name' (text input), 'Username' (text input), 'Birthday' (text input with a placeholder 'mm/dd/yyyy'), 'Email' (text input), 'Password (6 characters minimum)' (text input), and 'Password confirmation' (text input). At the bottom, there are two buttons: 'Sign up' and 'Log in'.

Create your account in this page.

1. enter your First name and Last name.
2. enter your username (This is the your name that let others look in this system.)
ps: hope the username is not same as others, so create a unique username.
3. enter your birthday.
4. enter your e-mail: this is your account when you login in.
5. enter your password: at least 6 characters.
6. enter your password again: must be same as password you first enter.
7. check the information you enter. If think is correct, click "sign up" button.
8. when finish your register and want to enter this system as soon as possible, click "login in" button and go to login-in page.

After login in:

Navigation bar2:



The image shows a navigation bar with a dark background. It contains the following links: 'OCDX Engine', 'New Post', 'Profile', 'Logout', and 'Generate Manifest'. On the right side, there is a search box with the placeholder text 'Search...'.

1. Choose the page you want to go in here through clicking appropriate button.
"OCDX Engine" -> post page.
"New Post" -> NewPost page.
"Profile" -> profile page.
"Loginout" -> Logout page.
"Generate Manifest" ->
2. if you want to search something, please enter the keywords in the search box and click search button

OCDX page:

Posts

Title: I different
Description: Ayyyyy
Author: Zach Bryant
JSON File: [File](#)
Post Link: [Post](#)
1st Additional File: [File](#)

Title: This is a test
Description: Test description
Author: Zach Bryant
JSON File: [File](#)
Post Link: [Post](#)
1st Additional File: [File](#)
2nd Additional File: [File](#)

we will post file in this page.

Title: I different
Author: Zach Bryant
Description: Ayyyyy
Tags: yo hi different
Manifest: [File](#)

1st Additional File: [File](#)
Created At: November 29 2016, 4:39am
Updated At: November 29 2016, 4:39am

[Edit](#)

Editing Post

Title
Description
Tags
Manifest File:
 No file chosen
Additional Files:
 No file chosen
 No file chosen
 No file chosen
 No file chosen
 No file chosen

it is different with post page before login in. It has “edit” button. when you click “edit” button, it will go to editing post page. In this page, you can change information about this file and add more file and then click “update” button. All change will be saved.

NewPost page:

New Post

Title

Description

Tags

Manifest File:
 No file chosen

Additional Files:
 No file chosen

No file chosen

No file chosen

No file chosen

No file chosen

This page is for your new post.

1. enter the title.
2. enter description of your new post.
3. enter the tags
4. can choose many files you want to upload if you want to.
5. check the information that you enter and click "Create Post" button, your new post will show in the OCXD page.

Profile page:

Name: abc abc

Email: aaa@bbb

Username:

Birthday: 1994-08-13

Posts:

[Edit Profile](#)

This page will show your all personal information. (like example above).
If want to edit profile, please click "Edit Profile". I will go to edit_profile page.

Edit Profile page:

Edit User

| | |
|--|---|
| First name | <input type="text" value="abc"/> |
| Last name | <input type="text" value="abc"/> |
| Username | <input type="text"/> |
| Birthday | <input type="text" value="08/13/1994"/> |
| Email | <input type="text" value="aaa@bbb"/> |
| Password (leave blank if you don't want to change it) | <input type="password"/> |
| 6 characters minimum | |
| Password confirmation | <input type="password"/> |
| Current password (we need your current password to confirm your changes) | <input type="password"/> |
| | <input type="button" value="Update"/> |

This page you can change everything you want to change and then click “update” button. your information will be changed. If you do not want to change password, please enter your current password in the last box.

Cancel my account

| |
|--|
| Unhappy? |
| <input type="button" value="Cancel my account"/> |
| Back |

If you do not like our system, you can cancel your account in this page. Just click “cancel my account” button, we will delete your information and account.

Login out page:

| | | |
|-------------|-------|----------|
| OCDX Engine | Login | Register |
|-------------|-------|----------|

Signed out successfully.

Posts

When click “Login out “ button, it will go to OCDX page before login in. And it will tell you signed out successfully. If you want to do something, login in again. If not, you can close the system.

Added Code:

Author: Zach Bryant

File: file_uploader.rb

^There are also 4 more files like this (file_two_uploader.rb, file_three_uploader.rb, file_four_uploader.rb, file_five_uploader.rb) and they all have the same code.

```
class FileUploader < CarrierWave::Uploader::Base

  # Include RMagick or MiniMagick support:
  # include CarrierWave::RMagick
  # include CarrierWave::MiniMagick

  # Choose what kind of storage to use for this uploader:
  storage :file
  # storage :fog

  # Override the directory where uploaded files will be stored.
  # This is a sensible default for uploaders that are meant to be mounted:
  def store_dir
    "uploads/#{model.class.to_s.underscore}/#{mounted_as}/#{model.id}"
  end

  # Provide a default URL as a default if there hasn't been a file uploaded:
  # def default_url
  #   # For Rails 3.1+ asset pipeline compatibility:
  #   # ActionController::Base.helpers.asset_path("fallback/" + [version_name, "default.png"].compact.join('_'))
  #
  #   "/images/fallback/" + [version_name, "default.png"].compact.join('_')
  # end

  # Process files as they are uploaded:
  # process scale: [200, 300]
  #
  # def scale(width, height)
```



```

# # do something
# end

# Create different versions of your uploaded files:
# version :thumb do
#   process resize_to_fit: [50, 50]
# end

# Add a white list of extensions which are allowed to be uploaded.
# For images you might use something like this:
# def extension_whitelist
#   %w(jpg jpeg gif png)
# end

# Override the filename of the uploaded files:
# Avoid using model.id or version_name here, see uploader/store.rb for details.
# def filename
#   "something.jpg" if original_filename
# end

end

```

Authors: Zach Bryant and Luke Jehle

File: posts_controller.rb

```

class PostsController < ApplicationController
  def edit
    @post = Post.find(params[:id])
  end
  def update
    @post = Post.find(params[:id])

    #respond_to do |format|
      #if @post.update_attributes(post_params)
        #flash[:notice] = 'Post was successfully updated.'
        #redirect_to post_path(@post)
      #else
        #format.html { render :action => "edit" }
      end
    end
  end
end

```

```

    #end
    @post.update_attributes(post_params)
    redirect_to post_path(@post)
  #end
end

def new
  #@post = posts_url.build(post_params)
  @post = Post.new
  @post.user_id = current_user.id
  #@post.user_id = current_user.id
  #redirect_to posts_path(@post)
end

def index
  @posts = Post.all
  if params[:search]
    @posts = Post.search(params[:search]).order("created_at DESC")
  else
    @posts = Post.all.order('created_at DESC')
  end
end

def show
  #@posts = Post.all
  @post = Post.find_by_id(params[:id])
end

def create
  #@post.user_id = current_user.id
  #@post = posts_url.create(post_params)
  #@post.user_id = current_user.id
  #@post = current_user.posts.build(permit_post)
  @post = Post.new(post_params)

  if @post.save
    flash[:success] = "Success!"
    redirect_to post_path(@post)
  else
    flash[:error] = @post.errors.full_messages
    redirect_to new_post_path
  end
end

```

```

    end
  end

  def destroy
    @post = Post.find(params[:id])
    @post.destroy
    redirect_to new_post_path
  end

private

  def post_params
    params.require(:post).permit(:json, :description, :title, :data, :user_id, :file, :filetwo, :filethree, :filefour, :filefive, :tags)
  end
end

```

Author: Zach Bryant

File: edit.html.erb

```

<h1>Editing Post</h1>
<hr>
<div align="center"><%= form_for (@post) do |f| %>
  <div class="field">
    <%= f.label :title %>
    <%= f.text_field :title %>
  </div>
  <br>
  <div class="field">
    <%= f.label :description %>
    <%= f.text_area :description %>
  </div>
  <br>
  <div class="field">
    <%= f.label :tags %>
    <%= f.text_field :tags, :required => true %>
  </div>
  <br>
  <div class="field">
    Manifest File:<%= f.file_field :json, :required => true %>
    <%= f.hidden_field :json_cache %>
  </div>
</div>

```

```

<br>
<div class="field">
  Additional Files:<%= f.file_field :file, :required => false %>
  <%= f.hidden_field :file_cache %>
</div>
<br>
<div class="field">
  <%= f.file_field :filetwo, :required => false %>
  <%= f.hidden_field :filetwo_cache %>
</div>
<br>
  <div class="field">
    <%= f.file_field :filethree, :required => false %>
    <%= f.hidden_field :filethree_cache %>
  </div>
<br>
<div class="field">
  <%= f.file_field :filefour, :required => false %>
  <%= f.hidden_field :filefour_cache %>
</div>
<br>
<div class="field">
  <%= f.file_field :filefive, :required => false %>
  <%= f.hidden_field :filefive_cache %>
</div>
<br>
<% if false %>
<div class="field">
  <%= f.label :jsonTxt %>
  <%= f.text_area :jsonTxt %>
</div>
<br>
<% end %>
<% if false %>
  <div class="field">
    <%= f.label :data %>
    <%= f.text_area :data %>
  </div>
<br>

```

```

    <% end %>

    <div class="actions">
      <%= f.submit "Update" %>
    </div>
  <% end %></div>

```

Author: Zach Bryant Contributor: Luke Jehle

File: index.html.erb

```

<h1>Posts</h1>
<hr>

<% @posts.each do |x| %>
  <style></style>
  <p>Title: <%= x.title %>
</p>
  <p>Description: <%= x.description %>
</p>
  <p>Author: <%= x.user.first_name + ' ' + x.user.last_name %>
  <p>JSON File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%= x.json
%>"target="_blank">File</a>
  </p>
  <p>Post Link: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000/posts/<%= x.id
%>"target="_blank">Post</a>
  </p>
  <% if x.file.present? %>
    <p>1st Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
x.file %>"target="_blank">File</a></p>
    <% end %>
  <% if x.filetwo.present? %>
    <p>2nd Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
x.filetwo %>"target="_blank">File</a></p>
    <% end %>
  <% if x.filethree.present? %>
    <p>3rd Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
x.filethree %>"target="_blank">File</a></p>
    <% end %>
  <% if x.filefour.present? %>

```

```

    <p>4th Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
x.filefour %>"target="_blank">File</a></p>
    <% end %>
    <% if x.filefive.present? %>
    <p>5th Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
x.filefive %>"target="_blank">File</a></p>
    <% end %>
    <hr>
<% end %>

```

Author: Zach Bryant

File: new.html.erb

```

<h1>New Post</h1>

```

```

<hr>

```

```

<div align="center"><%= form_for (@post) do |f| %>
  <div class="field">
    <%= f.label :title %>
    <%= f.text_field :title, :required => true %>
  </div>
  <br>
  <div class="field">
    <%= f.label :description %>
    <%= f.text_area :description, :required => true %>
  </div>
  <br>
  <div class="field">
    <%= f.label :tags %>
    <%= f.text_field :tags, :required => true %>
  </div>
  <br>
  <div class="field">
    Manifest File:<%= f.file_field :json, :required => true %>
    <%= f.hidden_field :json_cache %>
  </div>
  <br>
  <div class="field">
    Additional Files:<%= f.file_field :file, :required => false %>
  </div>
</div>

```

```

        <%= f.hidden_field :file_cache %>
    </div>
    <br>
    <div class="field">
        <%= f.file_field :filetwo, :required => false %>
        <%= f.hidden_field :filetwo_cache %>
    </div>
    <br>
        <div class="field">
            <%= f.file_field :filethree, :required => false %>
            <%= f.hidden_field :filethree_cache %>
        </div>
    <br>
    <div class="field">
        <%= f.file_field :filefour, :required => false %>
        <%= f.hidden_field :filefour_cache %>
    </div>
    <br>
    <div class="field">
        <%= f.file_field :filefive, :required => false %>
        <%= f.hidden_field :filefive_cache %>
    </div>
    <br>
        <% if false %>
        <div class="field">
            <%= f.label :jsonTxt %>
            <%= f.text_area :jsonTxt %>
        </div>
    <br>
        <div class="field">
            <%= f.label 'JSON' %>
            <%= f.text_area :data, :required => true %>
            <%= f.hidden_field :user_id , :value => current_user.id %>
        </div>
    <br>
        <% end %>
        <%= f.hidden_field :user_id , :value => current_user.id %>

    <div class="actions">

```

```
        <%= f.submit %>
      </div>
<% end %></div>
```

Author: Zach Bryant and Luke Jhele

File: show.html.erb

```
<h1>Title: <%= @post.title %> </h1>
<h2>Author: <%= @post.user.first_name + ' ' + @post.user.last_name %> </h2>
<h2>Description: <%= @post.description %> </h2>
<h2>Tags: <%= @post.tags %> </h2>
<h3>Manifest: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%= @post.json
%>"target="_blank">File</a> </h3>
<% if @post.file.present? %>
  <h3>1st Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
@post.file %>"target="_blank">File</a> </h3>
<% end %>
<% if @post.filetwo.present? %>
  <h3>2nd Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
@post.filetwo %>"target="_blank">File</a>
<% end %>
<% if @post.filethree.present? %>
  <h3>3rd Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
@post.filethree %>"target="_blank">File</h3>
<% end %>
<% if @post.filefour.present? %>
  <h3>4th Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
@post.filefour %>"target="_blank">File</h3>
<% end %>
<% if @post.filefive.present? %>
  <h3>5th Additional File: <a href="http://ec2-35-160-28-84.us-west-2.compute.amazonaws.com:3000<%=
@post.filefive %>"target="_blank">File</h3>
<% end %>
<h4>Created At: <%= @post.created_at.strftime("%B %d %Y, %l:%M:%P") %> </h4>
<h4>Updated At: <%= @post.updated_at.strftime("%B %d %Y, %l:%M:%P") %> </h4>
<% if current_user.id == @post.user_id %>
  <%= link_to 'Delete', @post, method: :delete, data: { confirm: 'Are you sure you want to delete this post?' } %>
<% end %>
<%= link_to 'Edit', [:edit, @post], data: { confirm: 'Are you sure you want to edit this post?' } %>
```


Completed Tasks

Kurt:

- Sprint Documentation
- Change log
- Testing architecture

Yi:

- Had a baby! Congrats!!
 - A girl!

Liang:

- Complete user document. <https://github.com/ztbc68/OCDX-Engine-Group5/commit/58f42e0cf929d545348a284b6f57ff8fdd918cc1>

Zach:

- File uploading
 - Allowed for multiple file uploads
 - Allowed files to be downloaded
- File profiles
 - Updated to allow user to edit their profiles
- Controllers

- Updated posts controller
- Show posts
 - Updated format
- New post
 - Added tags and ability to upload multiple files
- See pages 72-80 for work done by me

Luke:

- Search bar logic
- Show posts
- Controllers and Models

Kyle:

- Created link to generate manifest Json file
- Created 404 error page and error controller
- Worked on formatting UI
- Debugged UI logic

Change Log

OCT 31st:

Added ERD, DDL, information architecture and class diagram

UI rough designs added

Added regression, integration testing, and verification vs validation

Added link to website

NOV 11th:

Seed data insertion and DDL script added

Changed link to login page

Stub calls added

NOV 18:

Updated requirements document (top of doc)

Added UI pics/scripts

Added/updating testing architecture

Added User Documentation 1st draft

Added temporal logic

NOV 28:

Updated user document

Updated Testing architecture

Added cleaned up Ruby code