

## Project Features list:

- User account creation **COMPLETE**
  - New users must be able to create an account that will be used to manipulate files later on.
- File upload **IN PROGRESS**
  - Being able to put compressed files into our cloud system.
- File download
  - Being able to download files from the cloud system that your user account has authorization to read.
- File permission modifications **TODO**
  - Allow users to provide other users authorization to their files, or to remove this access. This should enable file sharing with a file owner.
- File browsing **TODO**
  - Allow users to view files with which they have permission to view/download.
- File encryption **TODO**
  - Encrypt files that are in the cloud in order to protect them from other users who are not owners of them.
- File decryption
  - Be able to decrypt files when the owner or another user with permission to use it wants to use them.
- CLI access **COMPLETE**
  - Allow users to use their terminal interface in order to perform file manipulations (sharing, uploading, downloading, etc.). No user account manipulations may be performed on this interface.
- GUI Access **IN PROGRESS**
  - Allow users to have an in browser method to perform file manipulations which are the same as above. Also this will be the source of user account manipulations.
- Account deletion **COMPLETE**
  - Allow users to remove their accounts.
- File deletion **IN PROGRESS**
  - Allow users to be able to remove a file from the cloud interface.

## Project Plan:

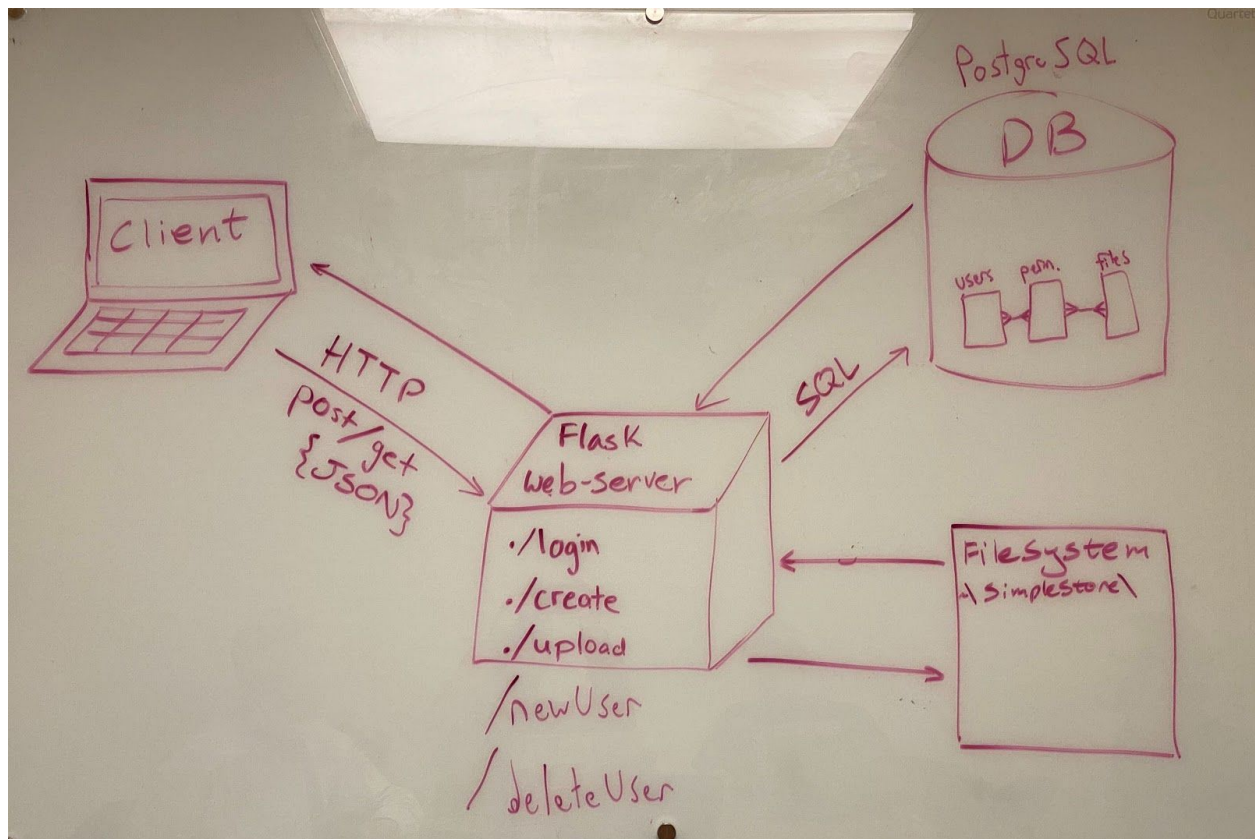
Can be found [here](#). This is a kanban chart showing the current status of all of our features and who they are assigned to.

The order of development we are going to try and achieve is:

1. CLI (10/18/19 - 10/29/19)
2. User Account Creation and Account Deletion (10/29/19 - 11/4/19)
3. File Compression and File Upload (11/4/19 - 11/11/19)
4. File Permission Modification and File Download (11/11/19 - 11/18/19)
5. File Encryption (11/18/19 - 12/2/19)

The dates for completion of each of these tasks will be on Mondays. The items listed in step 1 will be worked on now until 10/29/19 where they should be functional enough to move on. Step 2 will start once step 1 finishes so at the latest this will be 10/29/19 and run until the following week (11/4/19). Step 3 will follow the same pattern. At the latest it will start on the fourth and it will then be expected to be completed on the 11<sup>th</sup>. Step 4 will be the same, ending on the 18<sup>th</sup>. On the final step we will do a two week sprint. This means that work on encryption will start on the 18<sup>th</sup> and be completed no later than 12/2/19 which is the deadline.

### Architecture Design:



### Frontend Design:

CLI - The design for the CLI is that there will be a term for every action that a user may want that will be in the form `-term`. For terms that need an argument the argument will simply

follow the term in the form '-term arg'. Multiple arguments can be accepted at one time by simply having repeated args in the form '-term arg1 arg2'. The CLI will only recognize the first instance of any term that is input so any repeated term will be ignored. All terms are defined from the features list with an additional help command that can accommodate up to 1 argument (a term).

GUI -

The image displays four hand-drawn GUI wireframes for a 'SimpleStore' application, arranged in a 2x2 grid. Each wireframe is enclosed in a rectangular border and contains various input fields, buttons, and tables.

- Login:** Features a title 'SimpleStore.', an 'email' input field, a 'Password' input field with three asterisks, a 'log in' button, and a 'create acct.' link.
- Create Account:** Features a title 'SimpleStore', an 'email' input field, a 'Password' input field with three asterisks, a 'Verify' input field with three asterisks, a 'Create acct' button, and a 'log in' link.
- Home Page:** Features a title 'SimpleStore' and a table with three columns: 'Name', '#', and a checkbox. The first row contains 'file1', '2', and an unchecked checkbox. Below the table are 'upload' and 'Delete' buttons.
- Change Permissions:** Features a title 'SimpleStore' and a table with three columns: 'file1', 'R', 'W', and a checkbox. The first row contains '1. User1', a checked checkbox, an unchecked checkbox, and an 'X'. The second row contains '2. User2', a checked checkbox, a checked checkbox, and an 'X'. Below the table is an input field and an 'Add' button.

**Web Service Design:**

We have defined an API between our client-side interfaces and our back-end web server. Here it is:



Auth = Sha 3-256 (Username + Password)

login

10/22/19 SD Notes

Request Code = {Download, Upload, GetList, ~~Set~~ Set Auth, New User, Delete File, Delete User}

Download: requires a file ID & Auth. will return file to source URL if Auth is authorized for file ID.

Upload: requires Auth. Will not return, stores file on remote server with user supplied auth.

Set Auth: requires Auth, file ID, and change options per User ID. will change authorization settings on remote server for file if auth is authorized for file, and does not remove all authorized user IDs.

GetList: ~~requires~~ requires Auth, returns list of file ID & file Name & file type & auth. ~~for~~ that supplied auth has access to. (all users & usernames)

New User: requires Auth. Creates new UID, Auth supplied.

Delete File: requires Auth, File ID. Checks Auth for FID and if UID & Auth then deletes file from Remote server & removes DB record.

Delete User: requires Auth & User ID. if Auth Matches UID then remove UID from all files Auth. if only UID & Auth then Delete file. remove UID from DB.

Login: requires Auth. returns UID for Auth. ~~GetList & delete~~ or failure Message (Nice).

FID = Integer File ID

UID = Integer User ID

Opt = {UID: <INT>, OptCode: {Add, Remove} <sup>Read/Write</sup>}

FileName = String File Name

UserName = String User Name

Database Design:

We will create three PostgreSQL tables called Users, Permissions, and Files. In the Users table we will store a uid as the primary key for each user, the user's username, and the user's password. In the permissions table we will store a primary key for each file permission called pid, the file id (fid), and the user id that has permission to access that file (uid). The files table will store a file name, file id (fid) and the path to the file.

