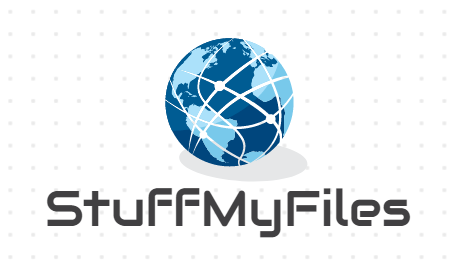
# **StuffmyFiles**

**Cloud Drive**

# **Software Design Document**



Name: Abdulsalam Bdeir, Kevin Tran, William Rood, and Jae Park

Lab Section: Team 1

WorkStation: Team 1

Date: 4/5/2021

**TABLE OF CONTENTS**

**1.** **INTRODUCTION**

1.1 Purpose

1.2 Scope

1.3 Overview

1.4 Reference Material

1.5 Definitions and Acronyms

**2.** **SYSTEM OVERVIEW**

**3.** **SYSTEM ARCHITECTURE**

3.1 Architectural Design

3.2 Decomposition Description

3.3 Design Rationale

**4.** **DATA DESIGN**

4.1 Data Description

4.2 Data Dictionary

**5.** **COMPONENT DESIGN**

**6.** **HUMAN INTERFACE DESIGN**

6.1 Overview of User Interface

6.2 Screen Images

6.3 Screen Objects and Actions

**7.** **REQUIREMENTS MATRIX**

### **1.** **INTRODUCTION**

**1.1** **Purpose**

This software design document describes the architecture and system structure of the StuffMyFiles cloud drive. The purpose of this SDD is to describe the design and how data and code is going to be structured. The intended audience for this design document is aimed towards developers and network admins.

**1.2** **Scope**

The necessary objectives of the project, in order:

1. Set up a host machine to contain the LEMP software stack
   1. Install the Unix operating system, Ubuntu Server, which will act as the foundation for all operations.
      1. Validate installation
   2. Register a domain name for the site. In our case, we chose *Stuffmyfiles.cf* because it was free from Freenom.com.
   3. Install/configure the Nginx web server to be able to deliver webpages to visitors of the site.
      1. Configure and enable a firewall on the server then validate the firewall is working.
      2. Configure the domain name to the server by setting the DNS records for the domain and updating the NameSpace records.
         1. validate that the domain points to the server by visiting the site and ensuring the page loads.
   4. Install/configure the MySQL server to be able to store and manage data for your site.
      1. Validate installation by checking if we are able to log in to the MySQL console
   5. Install PHP on Ubuntu.
      1. Validate installation
      2. Configure Nginx to use the PHP processor.
         1. Test PHP with Nginx to ensure Nginx can correctly hand off *.php* files to the PHP processor.
      3. Test that PHP is able to connect to MySQL and execute database queries by creating a small table and query for its contents from a PHP script.
2. Start coding the design template for web pages
   1. Create a CSS file on the web server to keep style uniform among all pages.
   2. Create the HTML/PHP files for each page of the site
      1. Reference the CSS files
      2. Connect the files to the Nginx web server and PHP processor
3. Initialize the SQL database
   1. Initialize the DDL to define the database schema, i.e. create tables, triggers, etc.
4. Start defining functions in HTML/PHP code
   1. Functions related to Signing up and signing in, user table instantiation, user folder table instantiation, allocating user storage space on hardware, etc.
   2. Functions related to loading user’s unique user page
   3. Functions related to file uploading, downloading and displaying files.
   4. Functions related to sorting files, renaming files, folder creation, moving files, and deleting files.
   5. Functions related to searching for users, befriending other users, populating the side navigation menu with “friends” of user.
   6. Functions related to setting File-User viewing permissions.
   7. Functions related to displaying storage bar percentage.
5. Push website live for initial testing/debugging purposes
   1. Test every function and process to check for errors/fail states
   2. Measure performance and load on host machine with varying amounts of users on at same time
   3. Measure upload/download speeds of site when various amounts of users call same function at same time
   4. Alter/update any functions
   5. Repeat 5
6. Push site live for anyone

Delivery Dates:

Complete step 1 by 3/16/2021

Complete step 2 by 4/12/2021

Complete step 3 by 4/12/2021

Complete step 4 by 4/15/2021

Complete step 5 by 4/25/2021

Complete step 6 by 5/1/2021

Final Project Due: May 3, 2021.

## 

## **1.3** **Overview**

The document will describe the design of the software including, but not limited to, architectural design, references for the software, dates for each part in the software development process, and etc. The document will first describe the architecture of the system. After, the design of the data and functions will be described. Then, the component design and human interface will be displayed.

## **1.4** **Reference Material**

List any documents, if any, which were used as sources of information for the test plan.

<https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04>

<https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-on-ubuntu-20-04>

<https://docs.digitalocean.com/products/networking/dns/>

<https://www.digitalocean.com/community/tutorials/how-to-point-to-digitalocean-nameservers-from-common-domain-registrars>

<https://www.serverlab.ca/tutorials/linux/web-servers-linux/how-to-configure-multiple-domains-with-nginx-on-ubuntu/>

<https://artisansweb.net/create-php-login-system-your-website/>

Login And Registration System Using PHP And MySQL <https://www.youtube.com/watch?v=pYVMQBwPJn8>

<https://www.w3schools.com/howto/howto_css_login_form.asp>

<https://phoenixnap.com/kb/ssh-to-connect-to-remote-server-linux-or-windows>

## **1.5** **Definitions and Acronyms**

PHP - A programming language that is geared towards web development.

MySQL - A database service that allows users to create databases and access them online through a cloud service

Server - A host device that sends and receives data from other computers in a network.

User - Someone who registers into the system through the sign up page.

Developer - Someone who seeks to see how the back-end of the system works/ or someone who wishes to create or edit the code of certain aspects of the system.

Function - A block of code or a method that aims to accomplish a certain task, whether that be to access something, return a value, or to redirect a user to another part of the system.

Host Machine - will run the web server, store and manage user-uploaded data, host the SQL database server needed to organize the files and set share permissions that the service allows for, and contain the PHP processor required to dynamically create user specific pages.

LEMP Stack - This is an acronym that describes a Linux operating system, with an Nginx web server, MySQL database and PHP processor working in conjunction with one another to serve dynamic web pages and web applications.

User Interface - The part of the software that the user will see and interact with.

### **2.** **SYSTEM OVERVIEW**

SuffMyFiles is an online service that will allow users to register for an account on the site and then allow registered users to upload an amount of files up to their assigned storage limit. Users will be able to share files between other registered users. Users who upload files or are shared files from other users will be able to download those files to their own computers without removing the file from the site. Users will be able to organize their files by placing them in folders, add descriptions to the files they upload, specify who their files can be shared with and delete no longer needed files. The website will prevent users from uploading files if the said file exceeds their storage limit. The website will allow the creation of accounts as long as the account’s username is not already registered in the database.

This project uses a LEMP software stack to serve dynamically created web pages. The Webserver is handled by Nginx. We chose Nginx because it was open source and it uses an asynchronous event-driven approach to handle multiple end user requests at the same time which should give our site better performance while supporting our limited hardware constraints. The backend data will be stored in a MySQL database and the dynamic processing will be done with PHP. All the software will be executed over the Ubuntu operating system. Specifically, we will be installing Ubuntu Server v20.04.

The host machine that will contain the LEMP software stack is a late 2013 Macbook Pro with an upgraded 500 GB SSD that houses the OS. A (1) 1000 GB external hard drive to hold user uploaded data. A 2.9GHz dual-core Intel Core i7 processor (Turbo Boost up to 3.6GHz) with 4MB L3 cache. 8GB of 1600MHz DDR3 memory, (2) USB 3 ports to connect to the external hard drives and 1 Gigabit Ethernet port that will connect directly to an available router using an Ethernet cable for direct internet connection.

The host machine will store and manage data offloaded to the site, run the nginx web server, host a SQLdatabase server to organize the files and share permissions that the service allows for, and contain the PHP processor required to dynamically create user specific pages.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

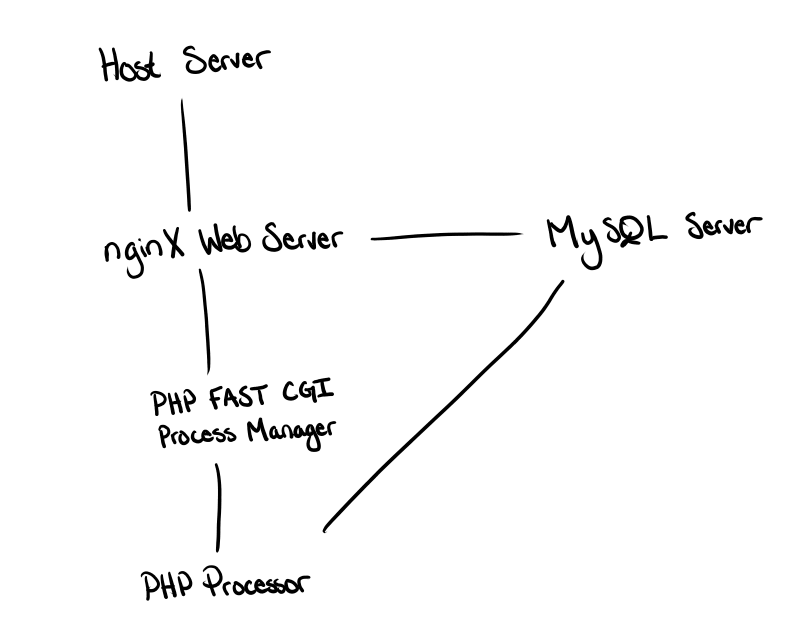
### 

### 

### 

### **3.** **SYSTEM ARCHITECTURE**

**3.1** **Architectural Design**



Host server: The host server is the part of the system that runs the Unix Linux Operating System required to store and manage the storage of data uploaded from users and will host the rest of the systems on the device that are necessary for the service to run.

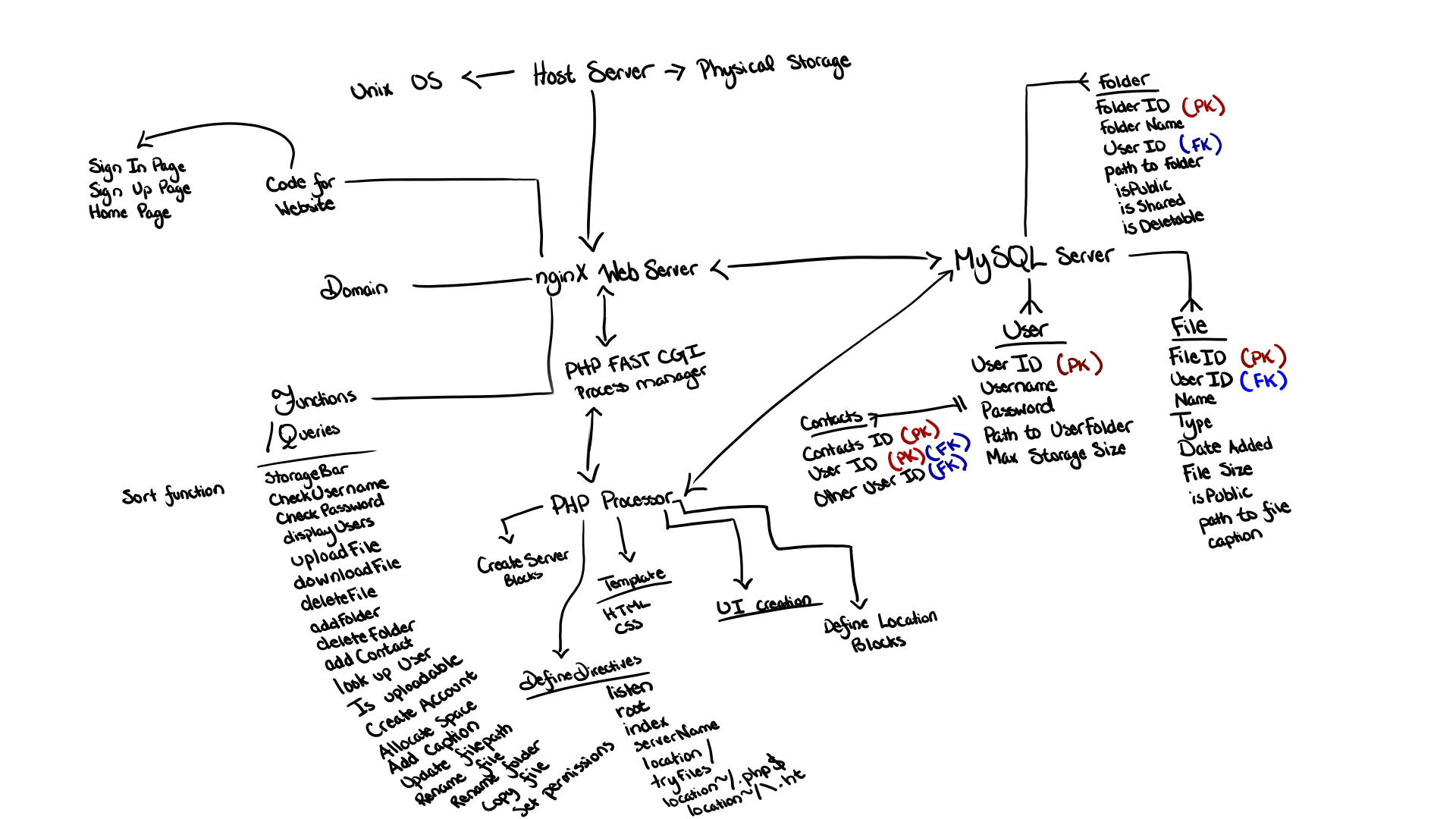
Nginx Web Server: This server will be handling all the code that is required to display and create web pages for users. This will be the server that handles all the functions required to make the server functionally sound as well as host the domain.

PHP FAST CGI Process Manager: This acts as a bridge between the Nginx web server and PHP processor so that we can use PHP to code the layouts on the page while still using an Nginx web server for the functions.

PHP Processor: This part of the system is used to dynamically create layouts for desired webpages.

MySQL Server: This part of the system is used to house the database, which stores all the metadata for registered users, their contacts, their folders and files that are uploaded to the system.

## **3.2** **Decomposition Description**



The MySQL server will house a database with 4 tables, a User table to keep track of all the registered users in the system, a File table to keep track of all the files uploaded to the system, a folder table to keep track of all the folders created by users in the system, and multiple instances of contact tables to keep track of each user’s friends in the system.

The Nginx web server will house the domain of the entire server, the code for the functionalities of the website, functions to interact with other aspects of the system, and all the functions needed to get the elements on the web pages to have a functional use.

The PHP processor acts as the intermediary between the user and the system, a dynamically created User Interface that allows users to use the functions provided by the system.

## 

## **3.3** **Design Rationale**

The reasoning behind choosing this kind of architecture was because this is a well known format, known as a LEMP Stack. LEMP is an acronym that stands for Linux OS NginX MySQL PHP stack, where the stack uses the 4 major components to make a cloud drive service that can be hosted on a linux based machine to give users access to the web services through the internet.

This was really the only option that we considered because we knew it was going to be necessary to have a server that could interpret a programming language geared towards web page programming and a database to store all the data for all the registered users and files uploaded to the server. A LEMP Stack provided us just that, a system that could interpret php and use MySQL to host a server that holds a database that we can change to our needs.

### **4.** **DATA DESIGN**

**4.1** **Data Description**

Username, File, Contact, and Folder information will all be stored in a database that will be hosted on a MySQL server. Whenever any of the information is needed, the Nginx server will send queries over to the MySQL server to either access data or edit data within the database.

Nginx will house the code for all the back-end functions, like logging users into the website, uploading files requested by users, viewing certain files, navigating through directories within user folders, requesting data from the database, etc. Nginx will also be in charge of the domain, so users can connect to the system via the internet.

The host computer that will be running the Unix Linux OS will be using an external hard drive that will be the storage of the system.

**4.2** **Data Dictionary**

**MySQL Server:**

User:

* User ID - int PK Not Nullable
* Username - String Not Nullable
* Password - String Not Nullable
* Path to User Folder - String Not Nullable
* Max Storage Size - int Nullable

File:

* File ID - int PK Not Nullable
* User ID - int Not Nullable
* Name - String Not Nullable
* Type - String Not Nullable
* Date Added - Date(10) (I think there is a date type? If so then change this to date) Not Nullable
* File Size - int Not Nullable
* isPublic - boolean Not Nullable
* Path to file - String Not Nullable
* Caption - String Nullable

Folder:

* Folder ID - int PK Not Nullable
* Folder Name - String Not Nullable
* User ID - int FK Not Nullable
* Path to Folder - String Not Nullable
* isPublic - Boolean Not Nullable
* isShared - Boolean Not Nullable
* isDeletable - Boolean Not Nullable
* Storage - float Not Nullable

Contact:

* Contact ID - int PK Not Nullable
* User ID - int PK FK Not Nullable
* User ID (other) - int FK Nullable

**Nginx Web server:**

Functions/Queries:

* addCaption(File file) / or (int FileID)
* addContact(int UserID, int otherUserID)
* addFolder(String directory, String folderName, int UserID, boolean deletable)
* allocateSpace()
* checkPassword(String password)
* checkUsername(String username)
* copyFile(File file) or (int FileID)
* createAccount(String username, String password)
* deleteContact(int UserID, int otherUserID)
* deleteFile(File file)
* deleteFolder(String pathway)
* downloadFile()
* isContact(int UserID)
* isUploadable(File file)
* lookupUser(String username)
* renameFile(File file) or (int FileID)
* setPermissionsFile(int FileID, int UserID, int number) - 0 is private, 1 is shared with all contacts, and 2 is public. All other values will be invalid.
* setPermissionsFolder(int FolderID, int UserID, int number) - 0 is private, 1 is shared with all contacts, and 2 is public. All other values will be invalid.
* signIn(String username, String password)
* signOut()
* StorageBar ()
* GetStorage()
* updateStorageBar(int newStorageValue)
* updateFilePath(int fileID, String filePathway)
* uploadFile(File file)

### **5.** **COMPONENT DESIGN**

* addCaption(File file) / or (int FileID)

Request access to the file in the database either by searching by file or File ID  
Change caption part of the table associated to that file

Update caption for the web page

* addContact(int UserID, int otherUserID)

Create new instance of Contact Table using user’s ID and new UserID

Contact ID is set equal to (contact ID of last contact created + 1)

User ID is set equal to UserID parameter

Other user ID is set equal to otherUserID parameter

Create another instance of Contact Table

Contact ID is set equal to (contact ID of last contact created + 1)

User ID is set to otherUserID parameter

Other user ID is set equal to UserID parameter

* addFolder(String directory, String folderName, int UserID, boolean deletable)

Request a query search for folders with same directory (directory will be directory/folderName), if there is a match, then stop and tell user folder cannot be created because folder with the name already exists

Create folder in the specified directory inside the storage of system

Create a new instance of Folder table in the directory

Set folderName equal to folderName parameter

Set Folder ID to (Folder ID of last folder added + 1)

Set User ID equal to the UserID parameter

Set path to folder equal to the directory given in the parameter

Set isPublic equal to false (default case)

Set isShared equal to false

Set isDeletable equal to deletable parameter

* allocateSpace()

Not really sure how this function will work

* checkPassword(String password)

Request a query search in the database for the User ID

If no match for requested UserID, return message indicating that there is not a valid username and return false

Check to see if passwords match, if they do return true, else return false

* checkUsername(String username)

Request a query search to look for user ID with specified username

If there is userID with a matching username, return true, else return false

* copyFile(File file) or (int FileID)

Request a query search in database looking for said File ID

Extract file directory from corresponding file table

Using file directory, find corresponding file in the storage

Create another instance of that file, and give the file a new name (add a (1) after file name)

Create another file table for the new file, with a new file ID and with the new name of the file

* createAccount(String username, String password)

Request a query search looking for a user ID that matches with inputted username

If there is a matching username with an existing user ID, stop program, else continue

Create a new instance of the User Table in the database

Set user ID to (User ID of last registered user + 1)

Set username to inputted username

Set password to inputted password

Call addFolder, with parameters fileDirectory = root/UserID, fileName =UserID, UserID = UserID (local variable), and isDeletable = false

Set Max Storage Size equal to maxStorage variable yet to be declared (maybe in bytes?)

* deleteContact(int UserID, int otherUserID)

Request a query search for contact table where UserID = UserID field and other User ID = otherUserID field

If query returns a table, then delete table and continue on with code. Else stop code and display error message

Delete instance of contact table where UserID = otherUserID field and Other User ID = UserID field

* deleteFile(int fileID)

Request a query for File table where FileID = fileID parameter

If a table is returned, then store path of file into a String variable

Use string variable to find file inside the storage

If the file is found, then delete the file inside the storage, and then delete the corresponding file table in the database

Else throw an error message saying that there is no file to be deleted

* deleteFolder(String pathway)

Request a query for Folder table where FolderID = folderID

If the table is returned, store the path of the file into a String Variable

Use the string Variable to find the folder

If the folder is found within the storage, then search for content inside the folder.

If a file is found in the folder, delete file table with that fileID and then use path to file to delete file off of the system

If a folder is found within the folder, call this function again to recurse on itself

Use path to folder argument to find and delete folder off of storage, and then delete folder table with the corresponding folderID

* downloadFile(int fileID)

Request a query for File table where fileID = fileID parameter

If a table is returned, then continue on with code. Else throw an error message

Store pathway of file into a string variable, and then use string variable to navigate to that address

Create a link where the user can download the file

* isContact(int UserID, int otherUserID)

Request a search query for contact table where user ID = UserID and other user ID = otherUserID

If a table is returned, return true

Else return false

* isUploadable(File file)

Find the remaining storage that is available

Store the remaining storageinto an int variable

Subtract the int variable with the file size

If the result is negative return False

Else return True.

* lookupUser(String username)

Request a query of the username

If the userID matches the UserID of username

Return UserID

Else throw an error message saying user does not exist

* renameFile(File file) or (int FileID)

Request a Query of the FileID

If the table is returned check permission if the user has the permission to modify the file

If it returns True

Update name of the File then Update path to file according to the name

Else send an error msg that user does not have permission

* setPermissionsFile(int FileID, int userID, int number)

Request a search query for File tables where FileID = FileID parameter

If a table is returned, then check if UserID field is equal to the UserID attribute

If userID field equals UserID attribute

Switch Number

Case 0: Remove sharing permissions from everyone but the user of the file

Case 1: Share with all the User’s contacts

* Run a search query for contact table where UserID = userID field
* Then loop through all the tables that are returned and share with all of those UserIDs

Case 2: Share will all users

* Run a search query for user table
* Loop through all the tables and share will all the tables

Else throw an error message

Else throw error message that file does not exist

* setPermissionsFolder(int FolderID, int number)

Request a search query for Folder tables where FolderID = FolderID parameter

If a table is returned, then check if UserID field is equal to the UserID attribute

If userID field equals UserID attribute

Use directory of folder to find contents inside folder

If file then:

Switch Number

Case 0: Remove sharing permissions from everyone but the user of the file

Case 1: Share with all the User’s contacts

* + Run a search query for contact table where UserID = userID field
  + Then loop through all the tables that are returned and share with all of those UserIDs

Case 2: Share will all users

* + Run a search query for user table
  + Loop through all the tables and share will all the tables

If folder then recall this function

Then run folder through the same switch as the files did

Else throw an error message

* signIn(String username, String password)

Use the method CheckUser

If the function returns false, show an error message that the Username does not match

Else use checkPassword

If the function returns false, show an error message that Password does not match with the username else proceed to sign in and redirect user to user home page

* signOut()

Check if there is any files that are being uploaded currently

Redirect user to sign in page

* StorageBar (unsure of parameter) (this is the getter right?)

Call getStorage() and store into a value

Use value to update graphics on the webpage

* GetStorage()

Return the value of storage in the user’s root folder

* updateStorageBar(int newStorageValue)

Call getStorage and store the value

Add newStorageValue to the value returned by getStorage

Then set the new value into the user’s root folder

* updateFilePath(int fileID, String filePathway)

Request a search query for file where FileID = fileID parameter

If a table is returned, then set path to file equal to the filePathway parameter

Move file to pathway specified by filePathway parameter

Else throw error message

* uploadFile(File file)

Use the IsUploadable function to check if the file is uploadable

If it returns True, upload the file

Else, deny uploading file into the cloud drive.

### **6.** **HUMAN INTERFACE DESIGN**

**6.1** **Overview of User Interface**

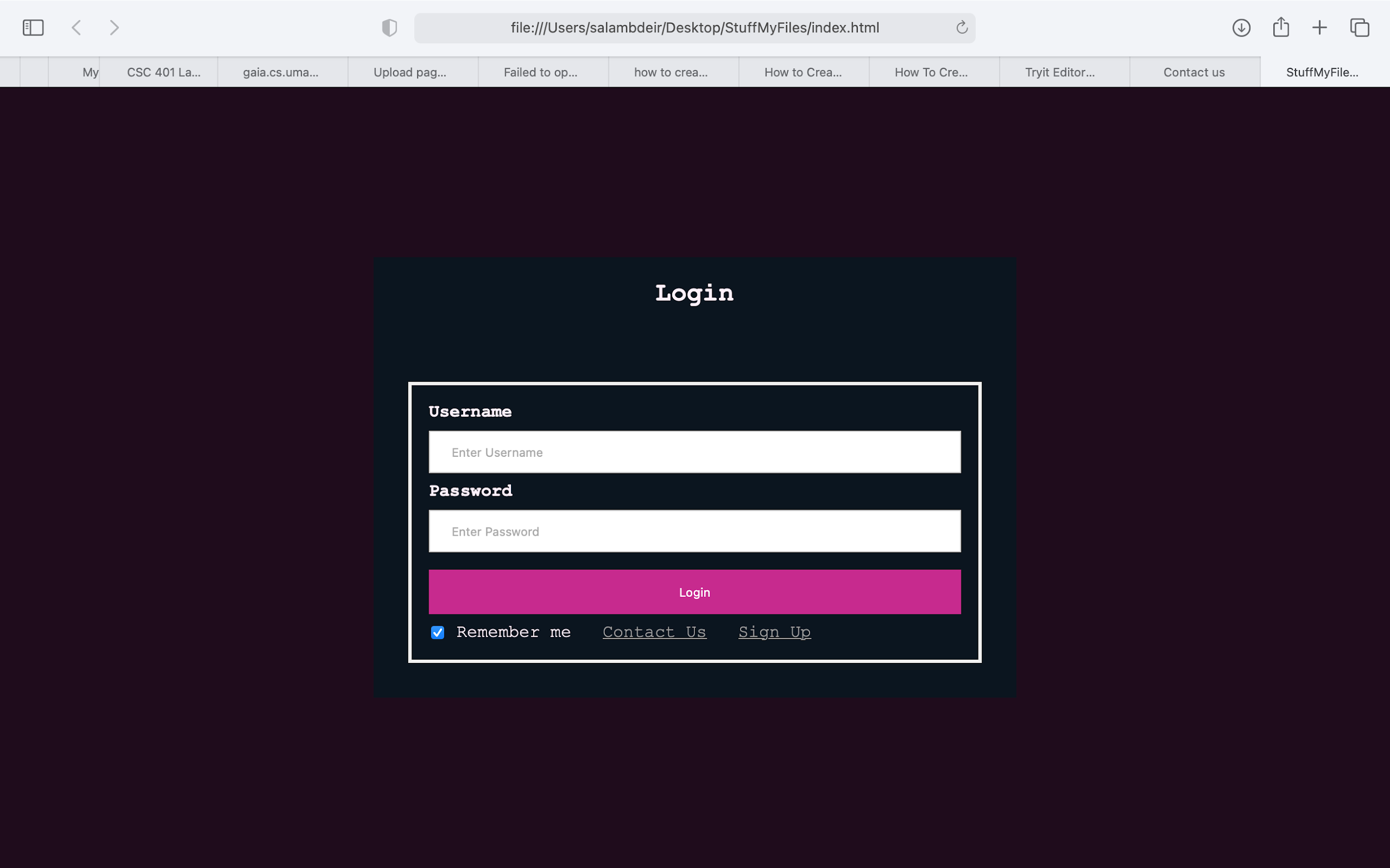
The user will upload the files they would like to save online by pressing on the upload button on the bottom right of their home screen. Each user will be able to see other users on a side-navigation menu on the left then click on each user to see which files are shared with them. When their drive gets messy, they are able to choose how they want to organize their files by clicking on the button that has specific organization methods. Each user will be able to download, delete, share, and rename their files by right-clicking on the file that they wish to edit.

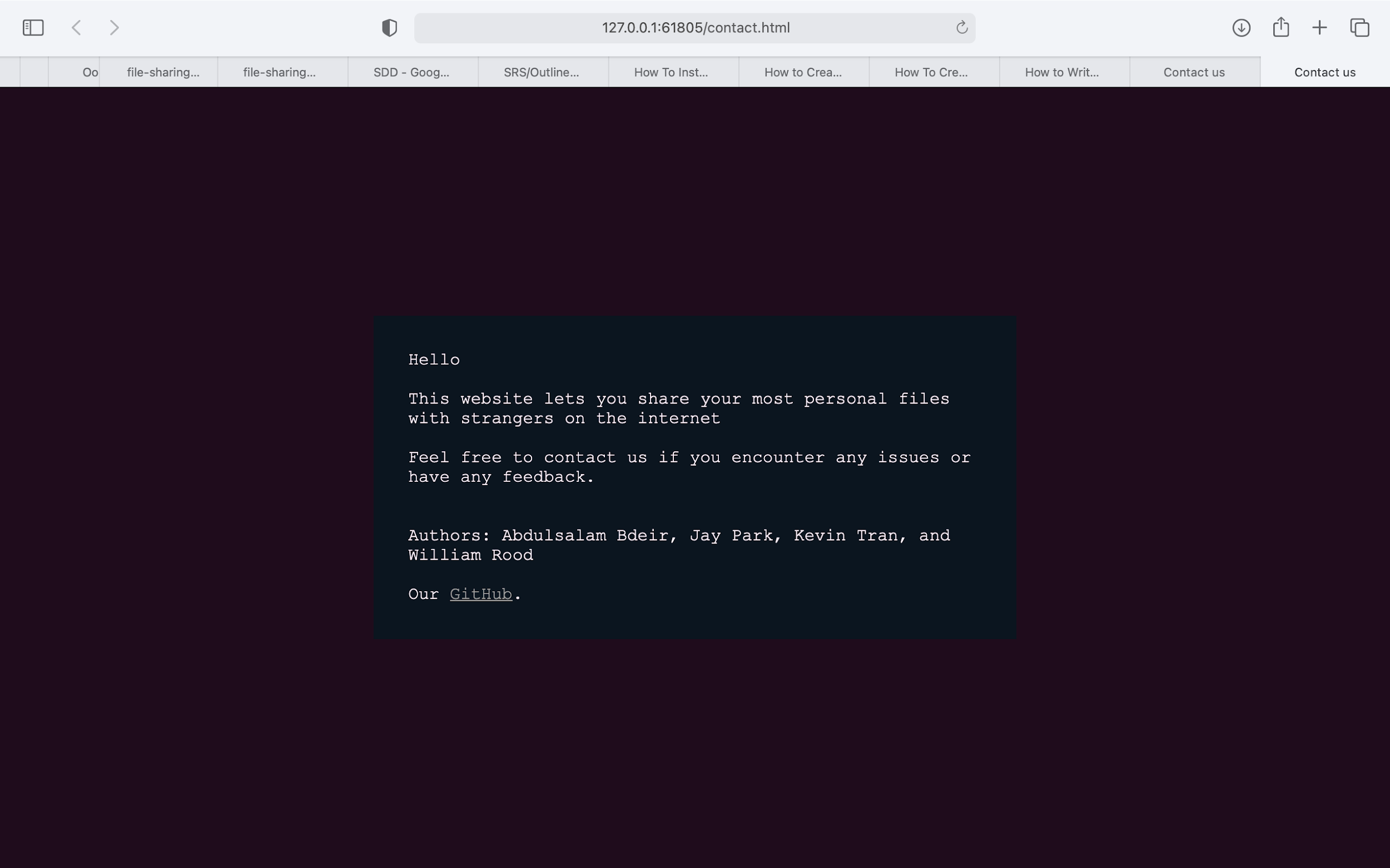
## **6.2** **Screen Images**

What user will see when first entering website (prototype):

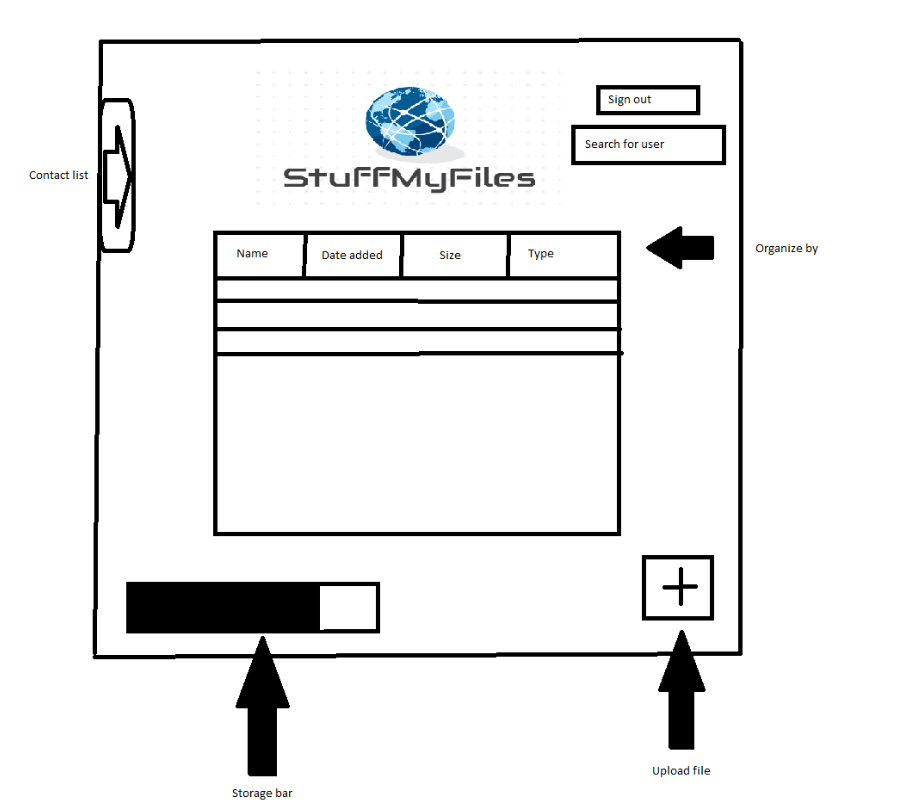
Elements included

* Enter username
* Enter password
* Create account
* Contact us
* Remember me



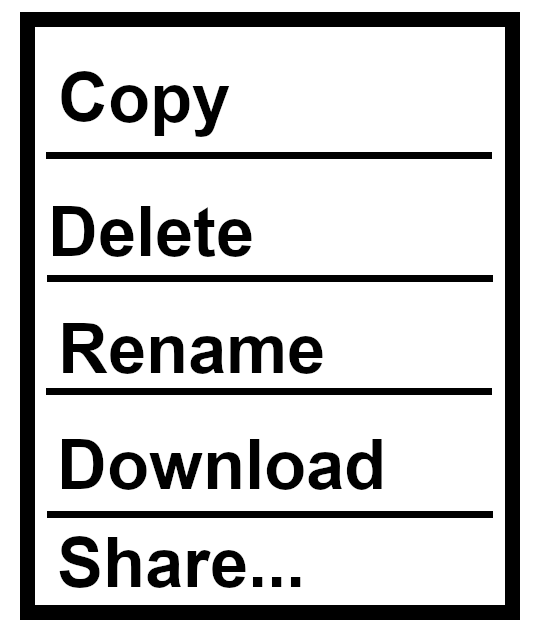


User account page once user signs in successfully (draft):

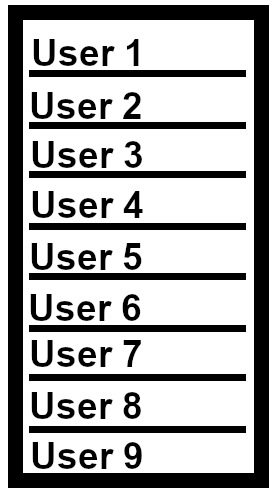


## **6.3** **Screen Objects and Actions**

Right click menu:



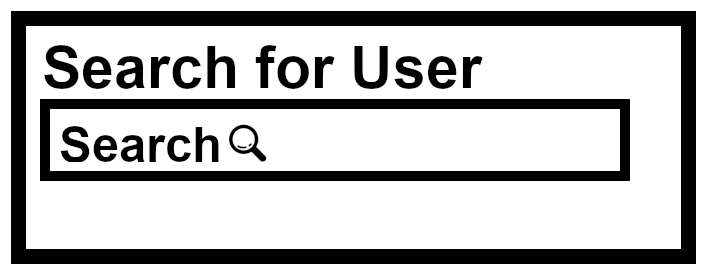
Side Navigation Menu:



Share pop-up:



Look up user pop-up:



### **7.** **REQUIREMENTS MATRIX**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements Description | Functional Requirement | Priority |
| 1 | Login page | 1.1 - checkPassword  1.2 - checkUsername  1.3 - signIn | High - User must login before doing anything else. |
| 2 | Create account | 2.1 - createAccount | High - User must create an account to access drive. |
| 3 | Home page | 3.1 - addFolder  3.2 - deleteFolder  3.3 - addCaption  3.4 - isUploadable  3.5 - lookupUser  3.6 - signOut  3.7 - GetStorage  3.8 - StorageBar  3.9 - updateStorageBar | High - These functions correlate to what the user will see when logging in. |
| 4 | File operations | 3.3 - addCaption  4.1 - copyFile  4.2 - deleteFile  4.3 - downloadFile  4.4 - renameFile  4.5 - allocateSpace  4.6 - uploadFile  4.7 - updateFilePath  4.8 - checkFileType | Medium - user has the choice to do any of these actions. |
| 5 | Sharing/Friend operations | 5.1 - setFilePermissions  5.2 - setFolderPermissions  5.3 - addContact  5.4 - deleteContact | Low - An extra operation available to the user. Not the main priority of software. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Constraints | Specification | Format |
| 1 | - Users must have already made an account, i.e. have a password and username within the MySQL database. | Started | Back end operations |
| 2 | - Limit of 16 characters for both password and username when creating an account. | Started | Back end operations |
| 3 | - Users must have enough storage space to upload files. | Not started | Front and back end operations |
| 4 | - User must have permission and ownership to edit the files  - User must upload a valid file type: images, text documents., salami, and sushi | Not Started | Front end operations |
| 5 | - Users are able to share the file with other users that have contact ID.  - Person that the user wants to share must exist in the database.  - User must have ownership of the original file in order to set permissions | Not started | Front end operations |