BSCH-CSP/Dub/PT

Cloud Services & Platforms

Application Development Assignment

“Building Dropbox with Google App Engine”

Part 2 of 2: Documentation

Alex Meade Wilson – Student No. 2950871

Monday 24th May 2021

Table of Contents

1. Assignment Brief

1.1 Introduction

In this assignment you will be tasked with building a replica of Dropbox, a file storage service. It is expected to be capable of dealing with directory structures and multiple files. It is expected that proper segregation of user data is implemented. In the first part of the assignment you will be tasked with enabling a full directory structure for each user. This includes the ability to create and delete directories, assuming no subdirectories or files are present. Navigation between directories should also be implemented.

Note that Google App Engine does not permit recursive object relationships i.e. you cannot store a directory as part of another directory. You will need another mechanism to represent directories. The suggestion here would be to use the user id combined with the full absolute path of a directory. You will also be required to implement file upload and download functionality along with some other more advanced functionality.

1.2 Submission and Penalties

You are required to submit two separate components:

- A copy of your complete Google App Engine project. The accepted archive formats are .zip, .rar, .7z. The use of any other archive format will incur a 10% penalty before grading.
- A PDF containing documentation of your code. If you do not provide documentation your code will not be marked. Copying and pasting code will not count as documentation.

There are also penalties you should be aware of:

- Code that fails to compile will incur a 30% penalty before grading. At this stage you should be completely capable of producing fully compiling code. I should be able to compile and run without having to fix syntax errors.
- The use of libraries outside the SDK will incur a 20% penalty before grading. You have all you need in the standard SDK. I shouldn't have to figure out how to install and use an external library to get your application to work. If you are not sure if a library is outside the SDK, please ask beforehand.
- The standard late penalties will also apply.

1.3 Coding & Documentation

Very Important: Take note of the grade brackets listed below. These are meant to be completed in order. If you skip a bracket or do not complete a bracket, the subsequent brackets will not be considered for marking. You should be well capable of producing strong and generally robust software by now. For example, if you fail the fourth bracket then brackets five, six, and seven will not be marked. Documentation will still be marked independent of code.

Marks will also be removed for the presence of bugs anywhere in the code and this will incur a deduction between 1% and 15% depending on the severity. If you have enough of these bugs it is entirely possible that you may not score many marks overall despite answering all brackets. I want robust bug free code that also validates all user input to make sure it is sensible in nature.

Also **note that the percentage listed** before the bracket is the **maximum mark** you can obtain if you complete that many brackets without error.

1.4 Coding Brackets (70%)

No	%	Coding Brackets
1	10	• Generate an application shell with a working login/logout system • Generate models to represent users, directories, and files
2	20	• When a user logs in for the first time generate a user object and a root (absolute path of /) directory for them and store both. • Default the user's path to / • Enable the user to add directories using a form. You will fail this bracket if two directories of the same name can be added to the current directory.
3	30	• Enable the user to delete directories. • Enable the user to navigate down into a subdirectory • If the user is in a sub directory, then display the special entry "../" that will move up a directory when clicked
4	40	• Add the ability to upload a file to the current directory. Use the blob store for storage • Add the ability to delete each file.
5	50	• Add the ability to download a file. • Add information to each file in the current directory including its: filename, size, type, and creation date
6	60	• Add in the ability to detect duplicate files in the same directory • Add in the ability to detect all duplicate files in a user's directory structure
7	70	• Add in the ability to move files from one directory to another • Add in the ability to share a file with another user.

1.5 Documentation Brackets (30%)

No	%	Coding Brackets

1	15	• Document why you designed the UI the way you did.
		• This should detail your choices in widget layout, position & how they make user interaction easier
		• Examples of what I am looking for are as follows:
		  ○ The login button was placed at the top right as this is where it is placed in many web applications.
		  ○ The colour scheme that was chosen to avoid the main form of colour blindness and produce high contrast for the visually impaired.

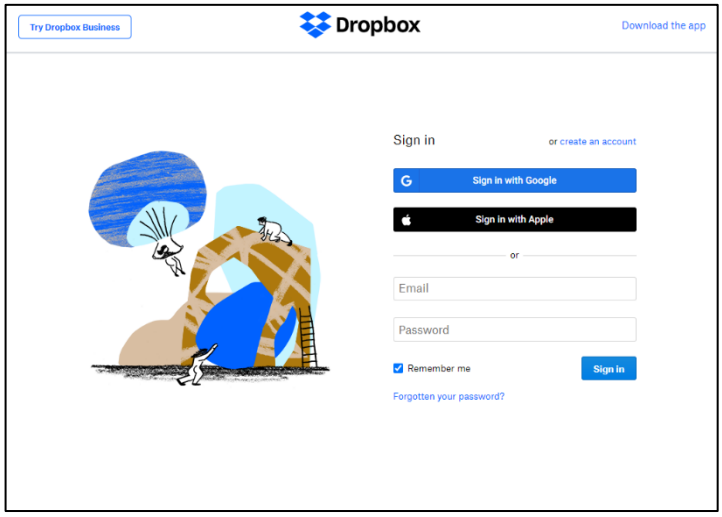
2	30	• Give a high-level description of every method in your Python code. • You should also document the data structures you have used and why they are used. **Note:** there should be no copying and pasting of code here.

(Kazmi, 2021)

2. GUI Design

2.1 Ideation & Mock-Ups

When setting out to build 'Dropbox' with Google App Engine, and initially considering the design and layout of my User Interface, I took most of my initial inspiration directly from Dropbox themselves, in terms of both Login and main Home or Directory and File views. Afterall, if I am trying to replicate Dropbox, then who knows best on how to deliver an Interactive User flow, other than 'Dropbox' themselves. That said, I needed to consider minimal UI views, as most of the work would be on the back-end routes and integration. So, I opted for a dual view, dynamic Login & Home view.



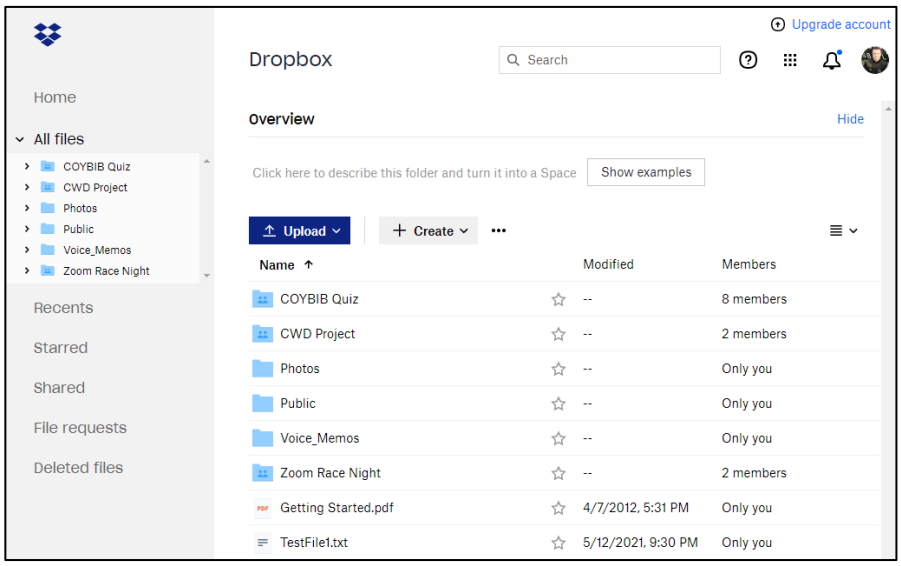
Dropbox – Login

For my Login, like Dropbox, I wanted a simple, Single Sign-On option to the right, of an inline-block placed image or logo to the left. It should be a responsive design, compatible across multiple devices, and if and where possible, accommodate varied login profiles and permissions types.

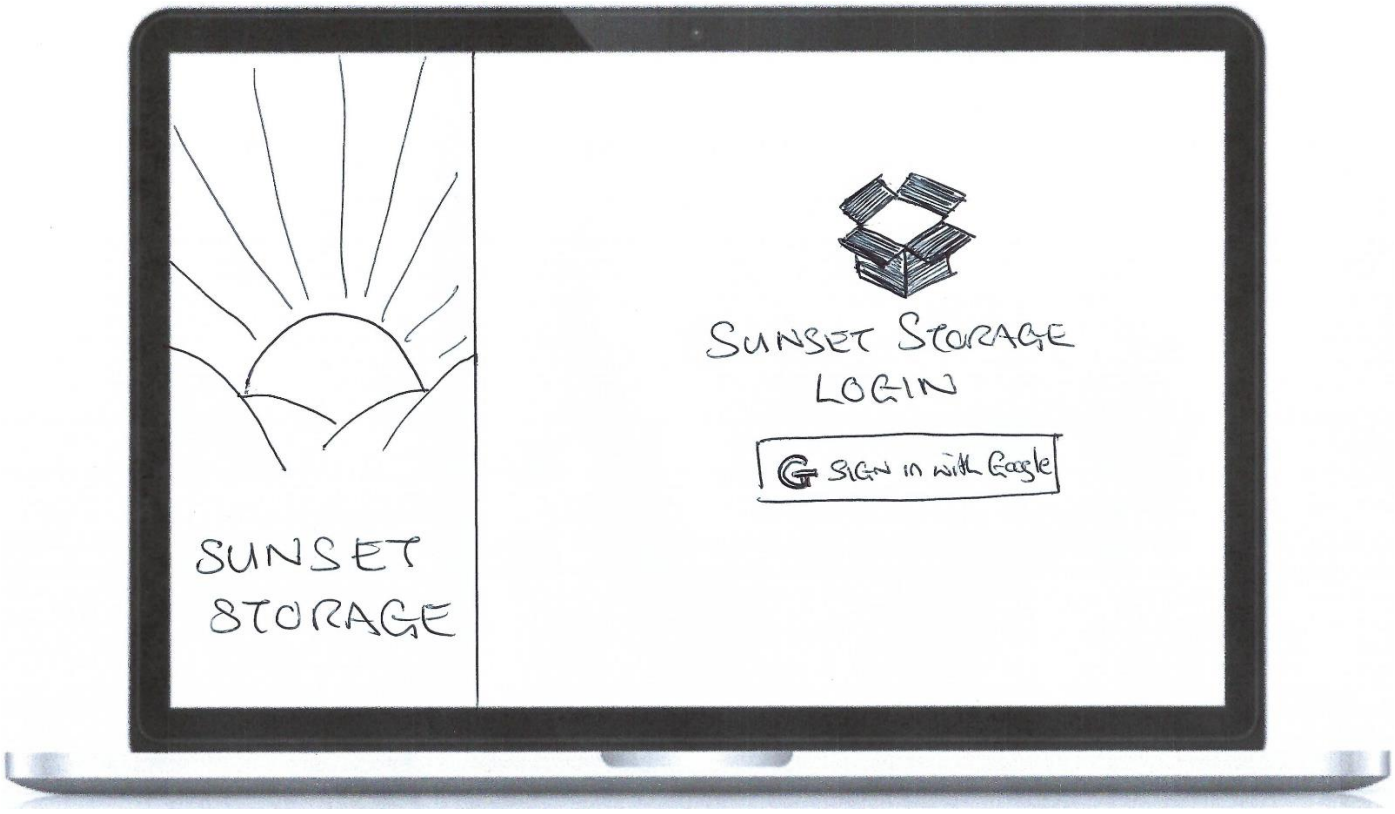
(Dropbox.com, 2021)

Dropbox – Home (Logged In)

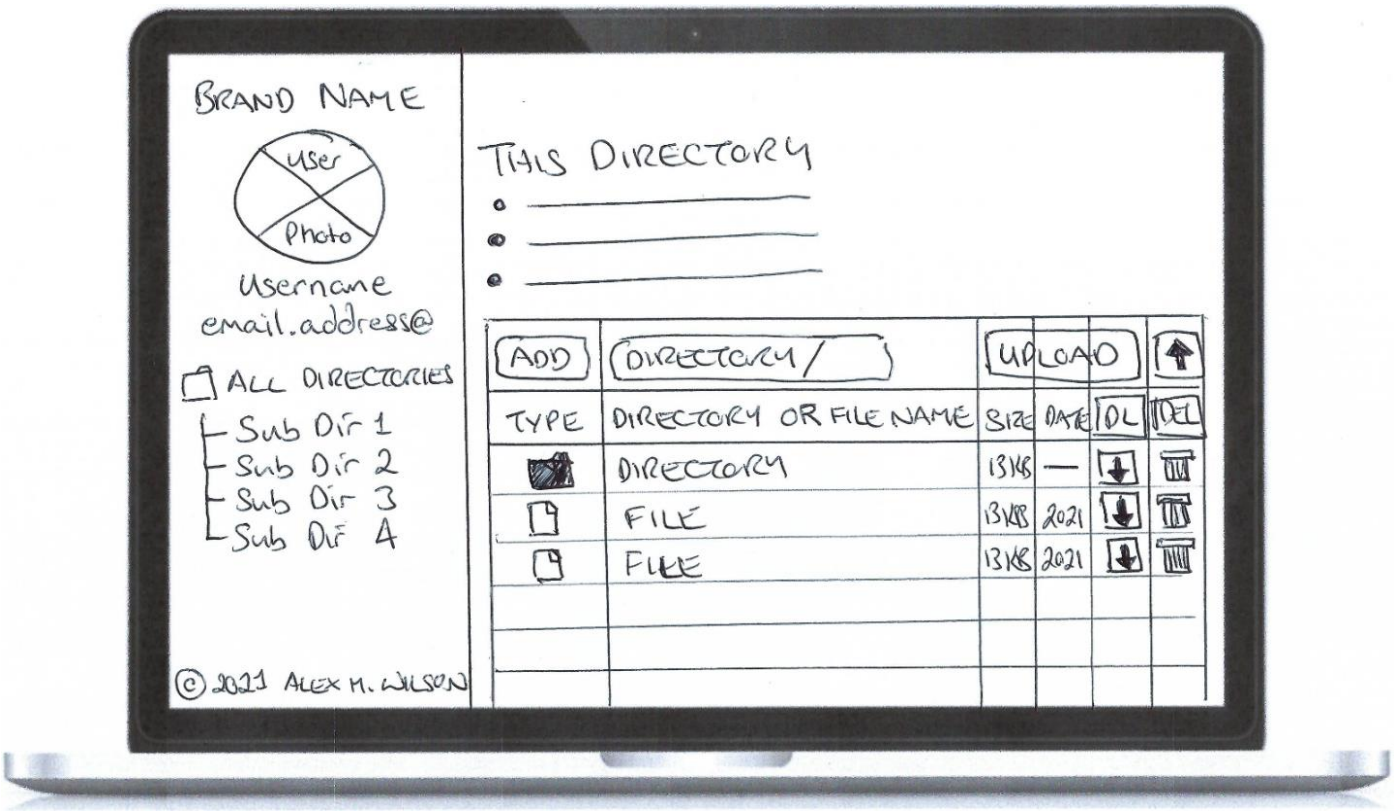
For the main logged in view, I wanted to design a left floated side bar menu, that offered a more prominent logo or User Profile Photo at the top, together with links to the various Directories & Files available to that User. This was to include a dynamic, collapsible/expandable list of Directories & Files. In the main container, it would display a table of all Directories / Sub-Directories & Files available, with an Action Bar at the top for new Directories or Files and individual Action Buttons per row, for each Directory or File that could be Downloaded, Deleted or Shared. See the following initial Mock-Ups.



Mock-Up - Login



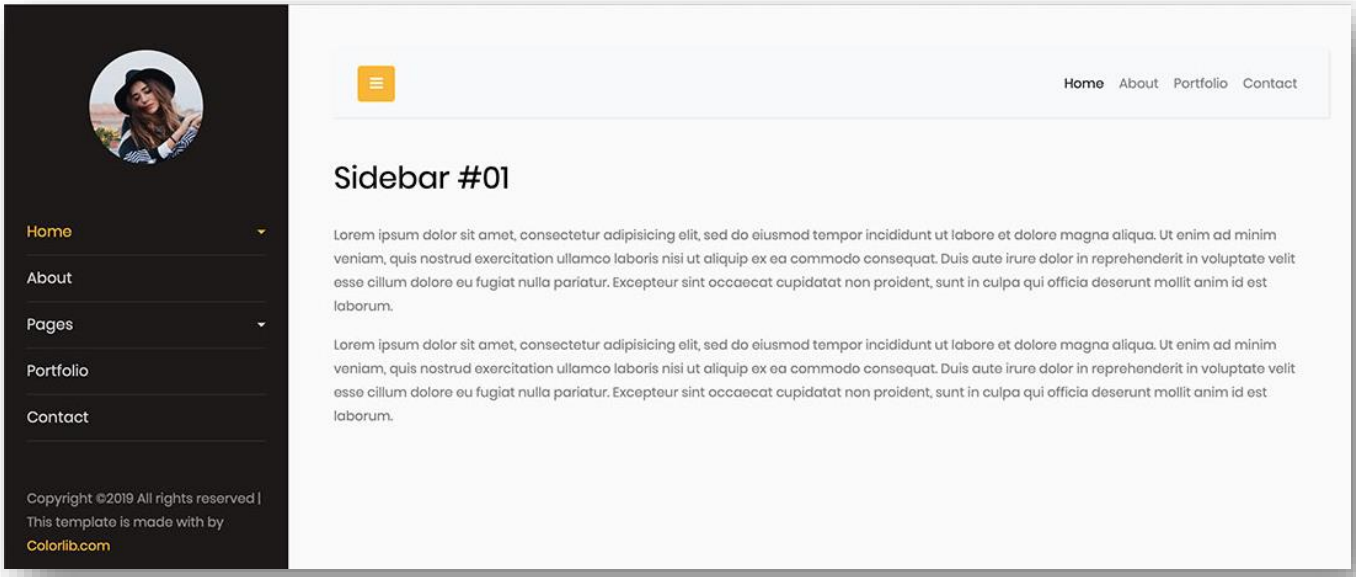
Mock-Up – Home (Logged In)



Final Design – Brand & Logo Design

When considering an initial Brand Name, Design, Logo, and overall Colour Scheme, whilst not getting drawn too deep into it, my aim was to maintain the association to 'Dropbox' whilst deriving a new brand. I came across this generic storage logo, illustrated here, that was on offer by an online Logo Maker. (LogoDesign.net, 2021)

From there, I began to search for variations of the 'Dropbox' logo and came across details of the company's rebrand. (Beck, 2017). This gave me further inspiration to adopt a similar logo. I also began searching for Bootstrap HTML Templates and came across the ***Bootstrap Sidebar by Colorlib V1*** below, which gave me the inspiration for my colour palette, based on a Primary Colour of Saffron. This Saffron colour gave birth to the name "Sunset Storage" and finalised this 'Dropbox' rebrand style fictional brand & logo.

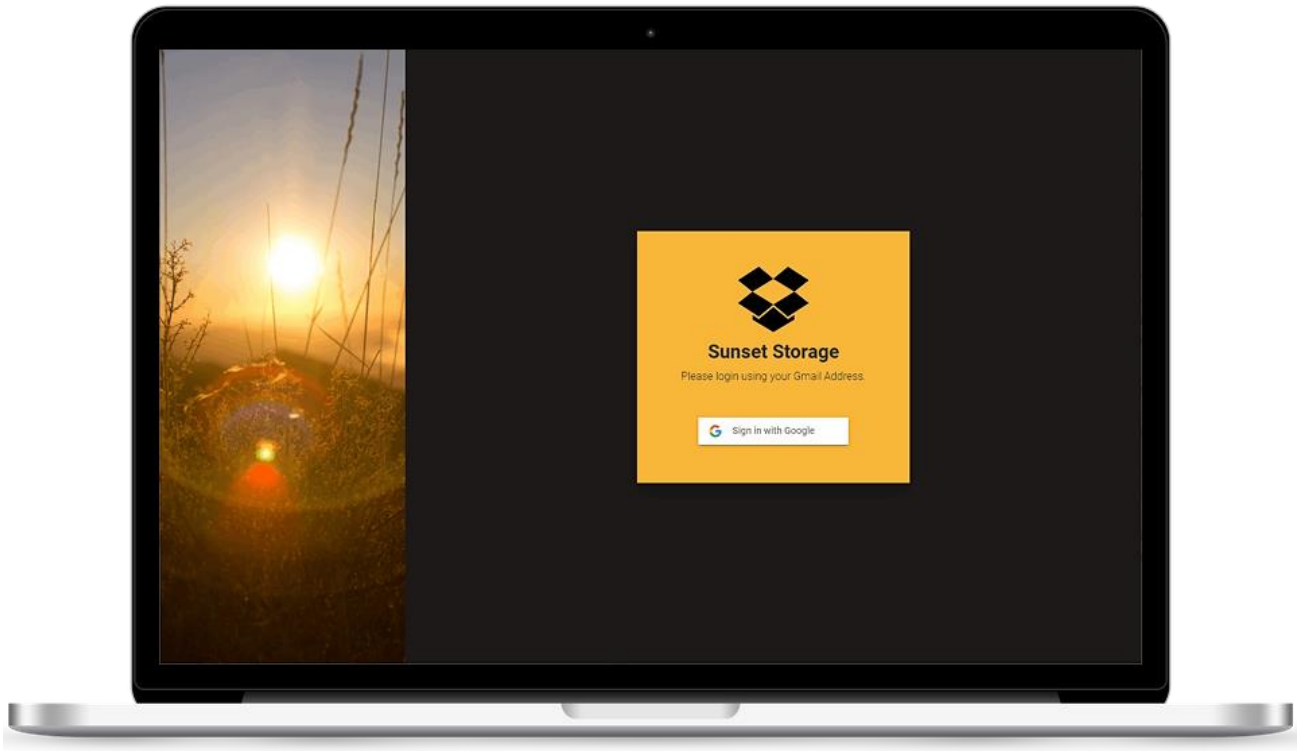


(colorlib, 2020)

Final Design – Colour Palette / Swatch

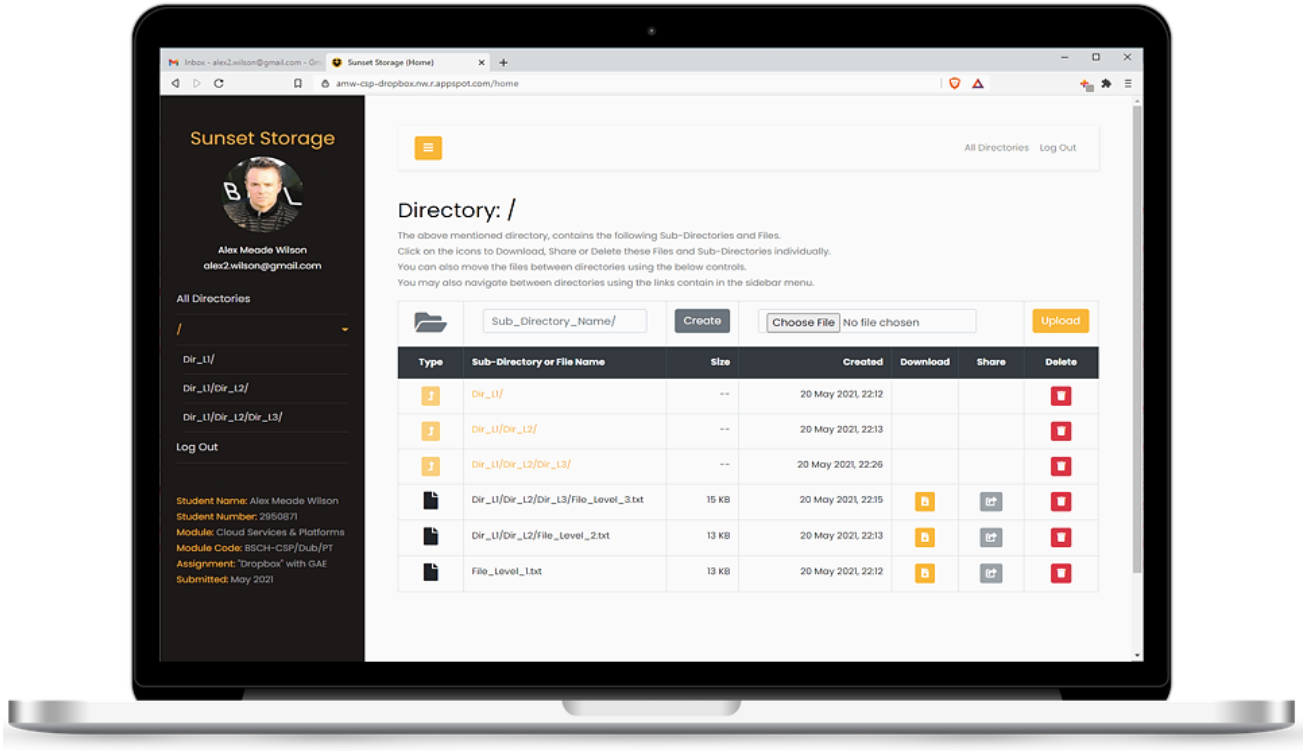
Sample					
Colour	Saffron	Cod Gray	White	Gray	Punch
Hex Code	#F8B739	#1D1919	#FFFFFF	#808080	#DC3545
RGB Code	248, 183, 57	29, 25, 25	255, 255, 255	128, 128, 128	220, 53, 69

Final Design – "Sunset Storage" – Login Screen



(colorlib, 2021)

Final Design – "Sunset Storage" – Home (Logged In)

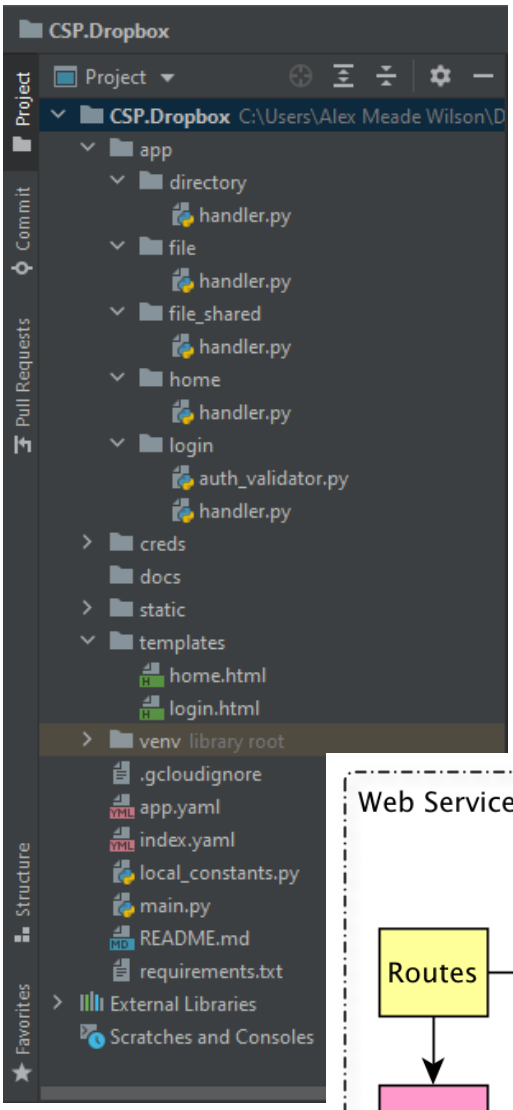


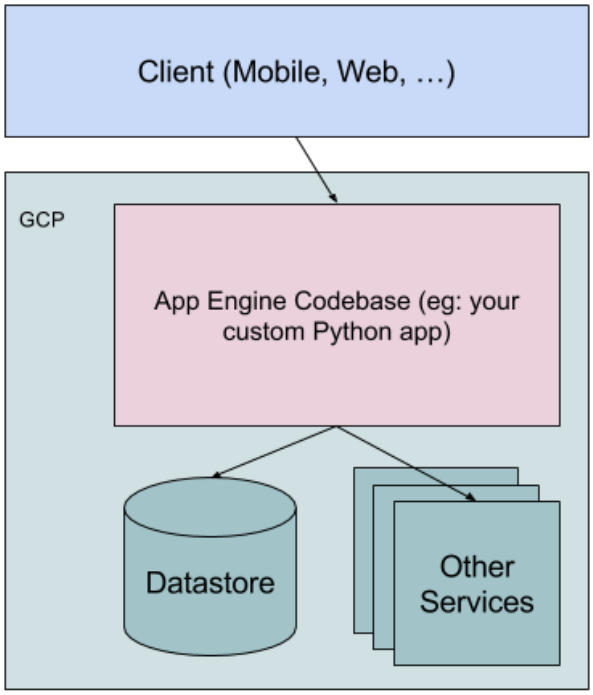
(colorlib, 2021)

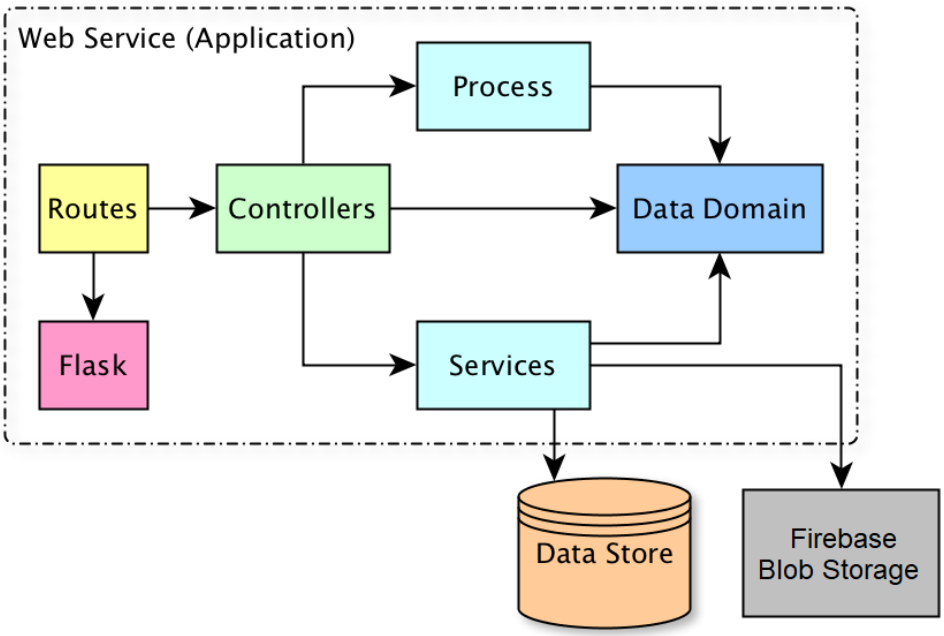
3. Coding Brackets

3.0 Architectural Overview & Data Structures

From the diagrams of my Application Architecture left and Data Structure below, I have attempted to apply a basic Flask Application Structure to my Google App Engine 'Dropbox' Application, whilst correctly defining Routes and Views, together with a Data Model Structure to represent Users, Directories and Files. At the time of coding I was unaware of 'Dynamic Routes & Variable Rules', so several routes overly extended with some repeat code, but I have since learned and understood where to improve these. (Birchard, 2020)





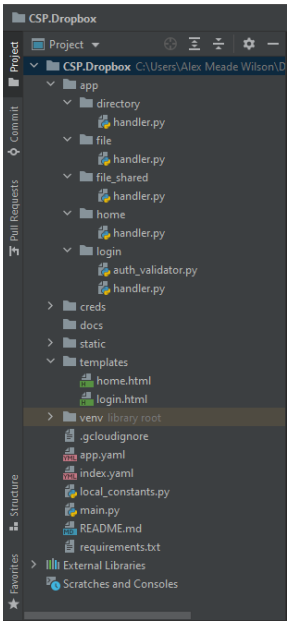


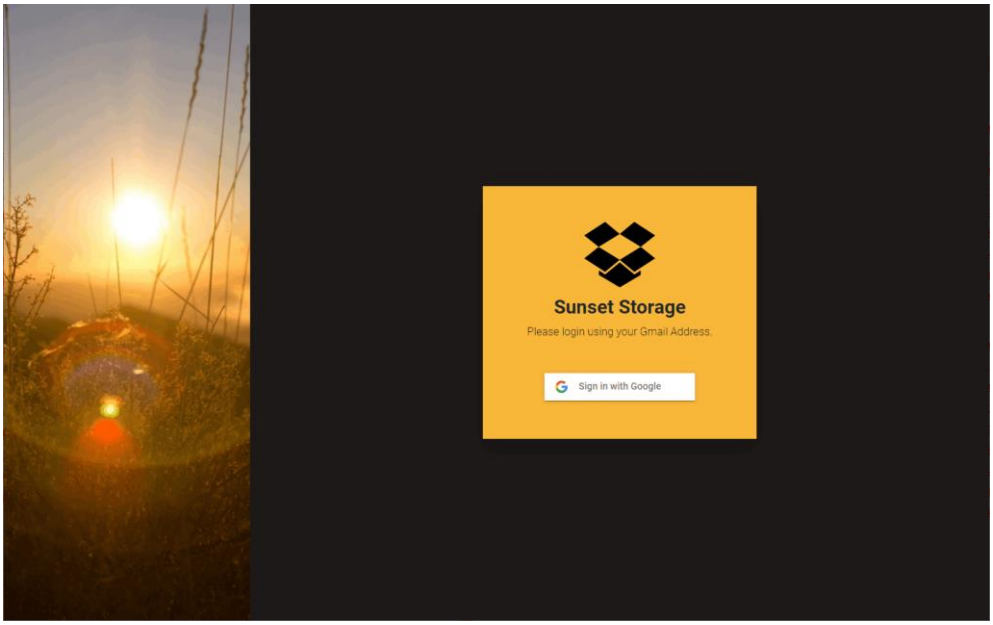
3.1 Coding Bracket 1 (10%)

Requirements

- ✓ Generate an application shell with a working login/logout system
- ✓ Generate models to represent users, directories, and files

Screenshot(s)





Requirements – Delivered & Described

1. Upon hitting routes like '/' or 'login' the user can successfully login using their Gmail Address, powered by Firebase Authentication.
2. Upon successfully logging in, models are generated for User Info, Directories and Files.

Described – Routes & Handler Methods

These are the Routes & Handler Methods with descriptions, used to deliver the above requirements.

App Directory and Source File: app/login/auth_validator.py

App Directory & File	Action Function(s)	High Level Description
app/login/auth_validator.py	retrieveUserInfo(claims)	Used to retrieve all UserInfo entities from the Datastore.
app/login/auth_validator.py	createUserInfo(claims)	Used to create all UserInfo entities in the Datastore.
app/login/auth_validator.py	validateAuth()	Used to validate the User during each function of the Application.

App Directory and Source File: app/login/auth_validator.py

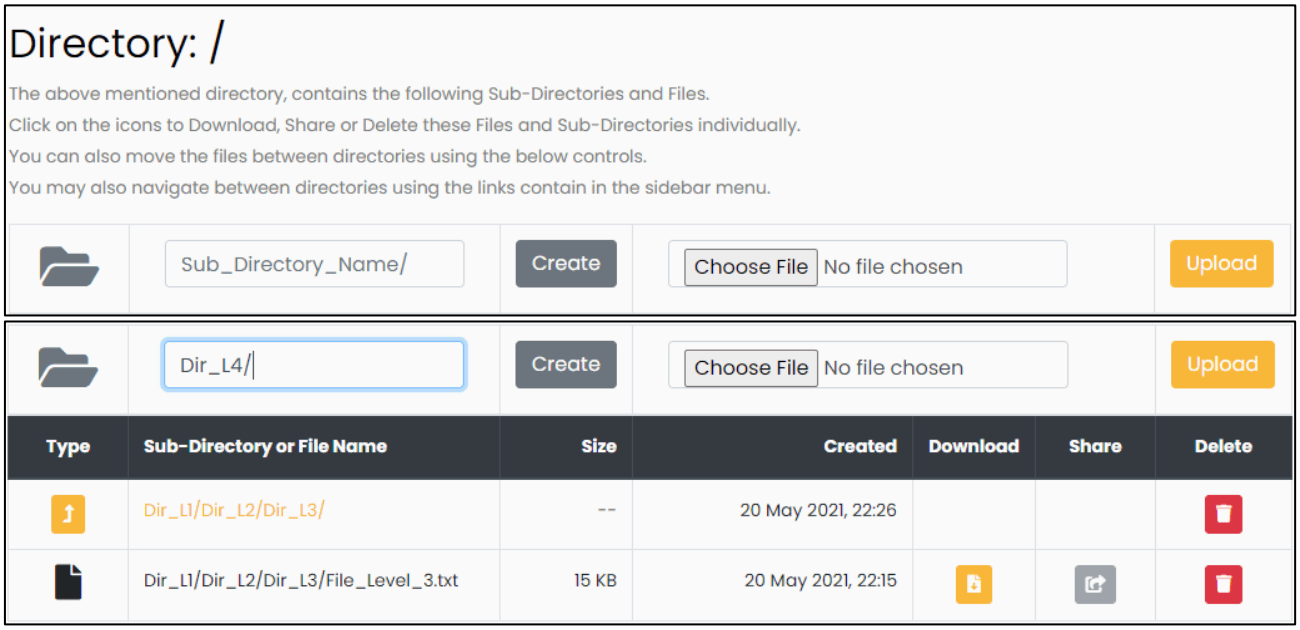
App Route	Route Handler Function	High Level Description
/login	login()	Checks if the User is validated or not, and redirects to Login, if not.
/logout	logout()	Unset the Token and log the user out, returning to Login.

3.2 Coding Bracket 2 (20%)

Requirements

- ✓ When a user logs in for the first time generate a user object and a root (absolute path of /) directory for them and store both.
- ✓ Default the user's path to /
- ✓ Enable the user to add directories using a form. You will fail this bracket if two directories of the same name can be added to the current directory.

Screenshot(s)



Requirements – Delivered & Described

3. Each new first time User that successfully logs on has a root path of '/' generated for them.
4. The Users first view on a logged in view of 'Home/' is a default path of the root directory '/'.
5. To add a Directory, input the name of the new Directory, ending in '/', in the action bar form, above the table, as shown above.

Described – Routes & Handler Methods

These are the Routes & Handler Methods with descriptions, used to deliver the above requirements.

App Directory and Source File: app/directory/handler.py

App Route	Route Handler Function	High Level Description
/create_root	createRootHandler()	Creates a Root Directory as a Blob in the Bucket.
/add_directory/	addDirectoryNoPrefixHandler()	Creates a New Directory at Root as a Blob in the Bucket.
/add_directory/<path:prefix>	addDirectoryHandler(prefix)	Creates a New Sub-Directory as a Blob in the Bucket.

3.3 Coding Bracket 3 (30%)

Requirements

- ✓ Enable the user to delete directories.
- ✓ Enable the user to navigate down into a subdirectory.
- ✓ If the user is in a sub directory, then display the special entry "../" that will move up a directory when clicked.

Screenshot(s)

Requirements – Delivered & Described

1. To delete a directory, click on the red trashcan icon, from within the 'Delete' column of the table.
2. To navigated down into a Sub-Directory, click on the dynamically linked name of the directory.
    a. Alternatively, you can navigate directly to a directory from the sidebar menu also.
3. If a User is in a Sub-Directory, navigate back/up a level by clicking the sunset filled 'Up' Arrow icon, from within the column on the farthest left.
    a. This icon button to back-up a level, is disabled when in the Root Directory.

Described – Routes & Handler Methods

These are the Routes & Handler Methods with descriptions, used to deliver the above requirements.

App Directory and Source File: app/directory/handler.py

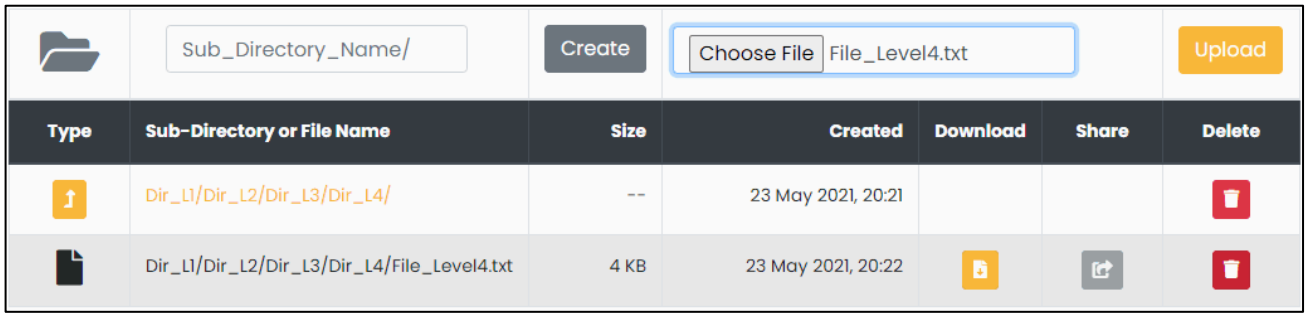
App Route	Route Handler Function	High Level Description
/delete_directory/<path:path>	deleteDirectoryHandler(path)	Deletes a Directory Blob from the Bucket using the path.
/this_directory/<path:path>	thisDirectoryHandler(path)	Never Used.

3.4 Coding Bracket 4 (40%)

Requirements

- ✓ Add the ability to upload a file to the current directory. Use the blob store for storage.
- ✓ Add the ability to delete each file.

Screenshot(s)



Requirements – Delivered & Described

1. To Upload a File is a two-step process.
    a. Firstly, 'Choose File' in the highlighted section of the Action Bar above.
    b. Then click 'Upload', and the file will automatically refresh with a new file_list.
2. To delete a file, click on the red trashcan icon, from within the 'Delete' column of the table.

Described – Routes & Handler Methods

These are the Routes & Handler Methods with descriptions, used to deliver the above requirements.

App Directory and Source File: app/file/handler.py

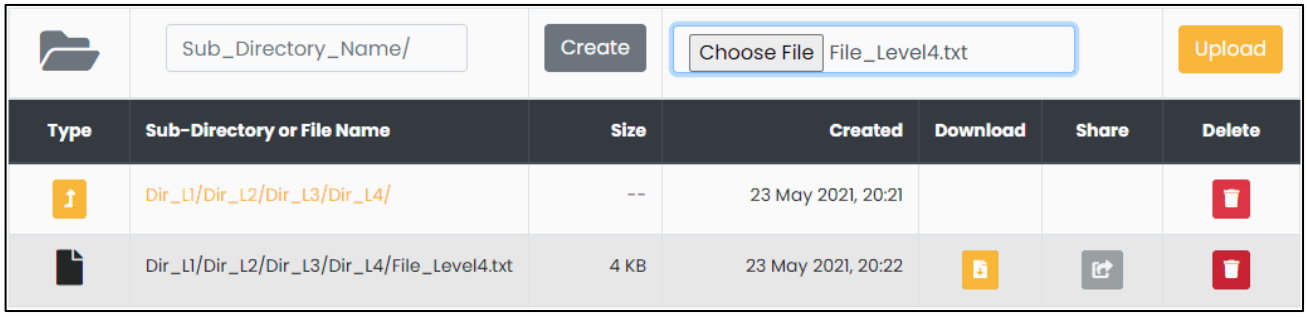
App Route	Route Handler Function	High Level Description
upload_file/	uploadFileNoPrefixHandler()	Uploads a File to Blob Storage using a blank prefix and filename.
upload_file/<path:prefix>	uploadFileHandler(prefix)	Uploads a File to Blob Storage using a filled prefix and filename.
delete_file/<path:path>	deleteFileHandler(path)	Deletes the file from Blob Storage using the full path.

3.5 Coding Bracket 5 (50%)

Requirements

- ✓ Add the ability to download a file.
- ✓ Add information to each file in the current directory including its: filename, size, type, and creation date.

Screenshot(s)



Requirements – Delivered & Described

1. To download a file, click on the Saffron download icon in the 'Download' column of the table.
2. The table displays the following information about each Directory or File.
    a. Type: Different icons for Directories and Files
    b. Sub-Directory or File Name: Dynamic linked Directory names, or unlinked File Names
    c. Size: The size in KB of each File only.
    d. Created: The creation date of the Directory or File.

Described – Routes & Handler Methods

These are the Routes & Handler Methods with descriptions, used to deliver the above requirements.

App Directory and Source File: app/file/handler.py

App Route	Route Handler Function	High Level Description
upload_file/	uploadFileNoPrefixHandler()	Uploads a File to Blob Storage using a blank prefix and filename.
upload_file/<path:prefix>	uploadFileHandler(prefix)	Uploads a File to Blob Storage using a filled prefix and filename.
download_file/<path:path>	downloadFileHandler(path)	Downloads the File from Blob Storage using the full path.
share_file/<path:path>	shareFileHandler(path)	Shares the file with other Users of the App (Incomplete).

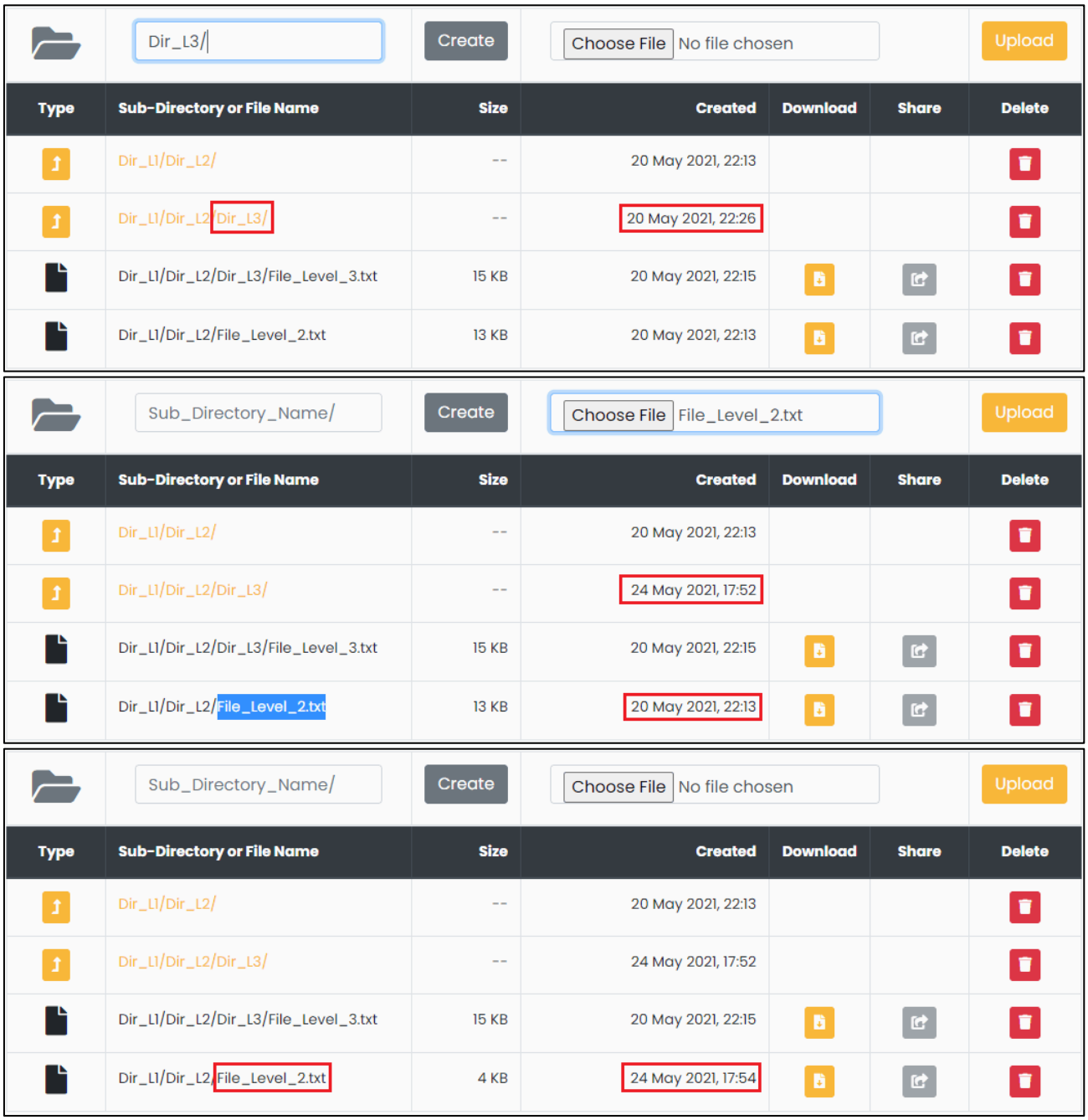
3.6 Coding Bracket 6 (60%)

Requirements

X Add in the ability to detect duplicate files in the same directory.

X Add in the ability to detect all duplicate files in a user's directory structure.

Screenshot(s)



Requirements – Delivered & Described

X I was unable to deliver any kind of duplicate Directory or File detection or validation in time.

X As you can see from the screenshots above, both duplicate Directories & Files are overwritten.

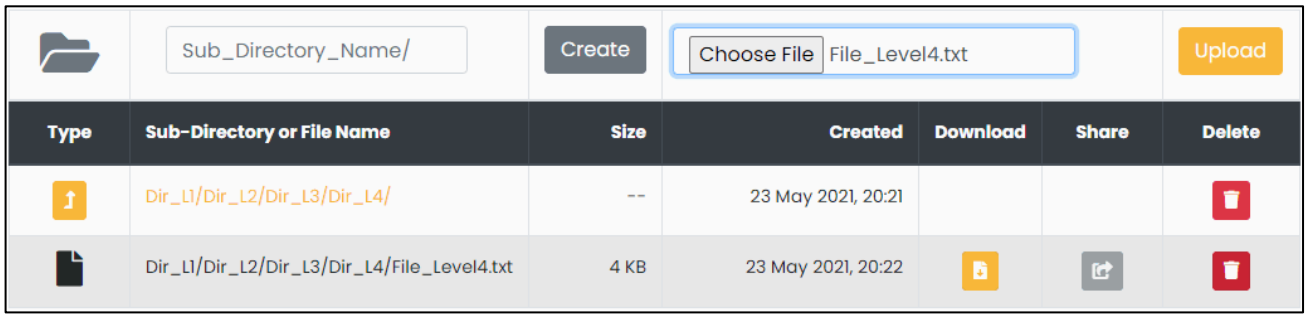
3.7 Coding Bracket 7 (70%)

Requirements

X Add in the ability to move files from one directory to another.

X Add in the ability to share a file with another user.

Screenshot(s)



Requirements – Delivered & Described

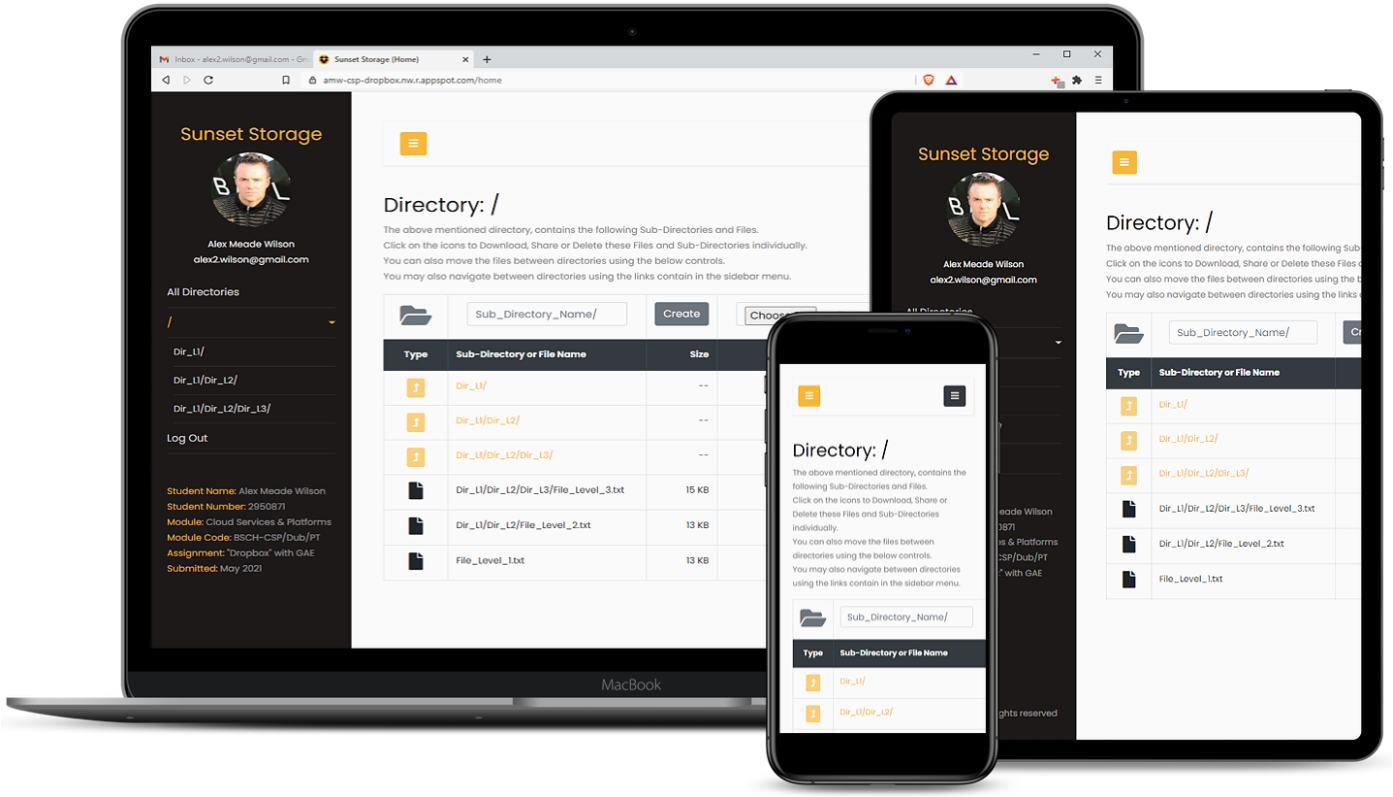
X I was unable to deliver any ‘Move File’ between Directories functionality in the time provided.

X I did provision for the functionality to be able to ‘Share’ files between Users, in both the UI and the back-end Routes & Data Model. However, I was unable to complete this in the time provided.

4. Final Application – "Sunset Storage"

In the end, despite not completing only a couple of the latter Coding Brackets, or Functional Features of the App, like Move Files between Directories, and Share Files between Users, I was still able to deliver a wholly responsive, Google App Engine driven Web Application with Flask Web Service and API Routing. One the performs significant User Controlled, Cloud based Directory Management and File Storage services, which I am very pleased with. I hope you agree.

4.1 Responsive Design



4.2 Web Application Site and Code Repository

The following repositories below, should remain active for the duration of this ungraded assignment and perhaps for a short time thereafter. The Web Application and Cloud based Web Service will expire at the end of the 'Free Trial' services by Google Cloud Storage and Google Cloud Platform. If there are any issues accessing the Site, Source Code or with the Service itself, please let me know.

"Sunset Storage" Web Application: https://amw-csp-dropbox.nw.r.appspot.com/

"Sunset Storage" GitHub Repo: https://github.com/AlexMeadeWilson/CSP.Dropbox

Bibliography

Beck, C., 2017. *Dropbox's rebrand will probably be successful whether you like the design or not.* [Online]
Available at: https://medium.com/better-product/dropboxs-rebrand-will-probably-be-successful-whether-you-like-the-design-or-not-ff2ad5e7975b
[Accessed 1 May 2021].

Birchard, T., 2020. *The Art of Routing in Flask.* [Online]
Available at: https://hackersandslackers.com/flask-routes
[Accessed 24 May 2021].

colorlib, 2020. *Bootstrap Sidebar V01.* [Online]
Available at: https://colorlib.com/wp/template/bootstrap-sidebar-01/
[Accessed 1 May 2021].

colorlib, 2021. *Login Form V04.* [Online]
Available at: https://colorlib.com/wp/template/login-form-04/
[Accessed 1 May 2021].

Dropbox.com, 2021. *Dropbox.com.* [Online]
Available at: https://www.dropbox.com/
[Accessed 1 May 2021].

Kazmi, D. A., 2021. Application Development Assignment - Building Dropbox with Google App Engine. *Cloud Services & Platforms,* 15 April.p. 3.

LogoDesign.net, 2021. *Design Free Storage Logos Here.* [Online]
Available at: https://www.logodesign.net/logos/storage
[Accessed 1 May 2021].

End of Document