Data Foundations System

Data Drive

Team - Red Flags

Team Members:

1. Karmanjyot Singh
2. Divya Varshini Yakkanti
3. Soveet Kumar Nayak

This document entails the entire documentation of the work done, during the project ‘Data Drive’, it’s key features and the implementation of the individual features, that would be useful for future extensions, development of the application.

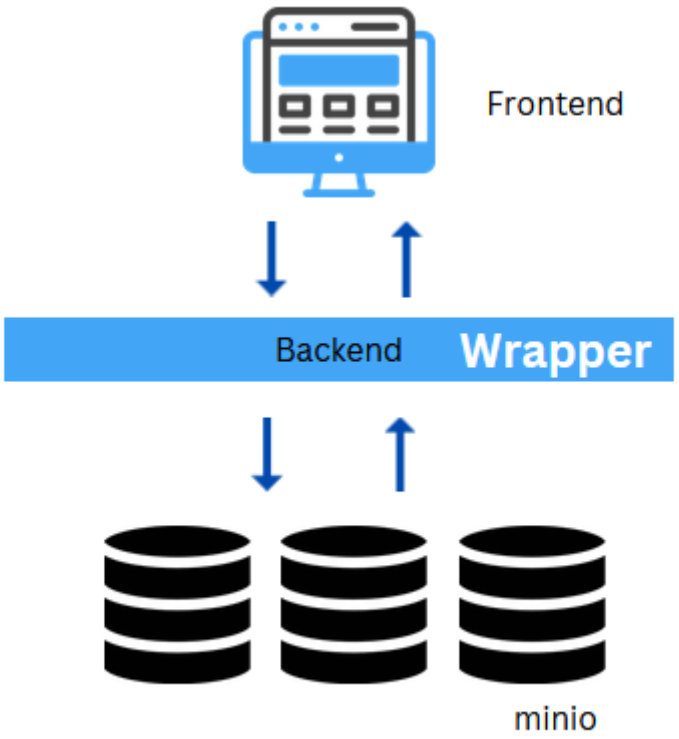
Architecture

The architecture, could broadly be separated into 3 sections:

1. The Interface (frontend)
   The entire frontend, is built on react using modern coding practices, modular code, and good coding practices. The implemented features, have been built using individual components, which could be utilised elsewhere in the application. We use `MUI` design library, for styling the components and the application.

2. The Wrapper (backend)
   We build a wrapper around the minio-python-sdk, using flask, to support the CRUD, etc. operations on the frontend side. We use Flask, to create a basic REST API, service to the Interface. The interactions to the minio-db, are integrated using the python-sdk.

3. The Data Storage (minio-db).
   It emulates the S3 bucket storage, architecture for our application. Once, a user logs into the application, he is allocated, a bucket which entails the entire user’s data.



Features

The architecture, could broadly be separated into 3 sections:

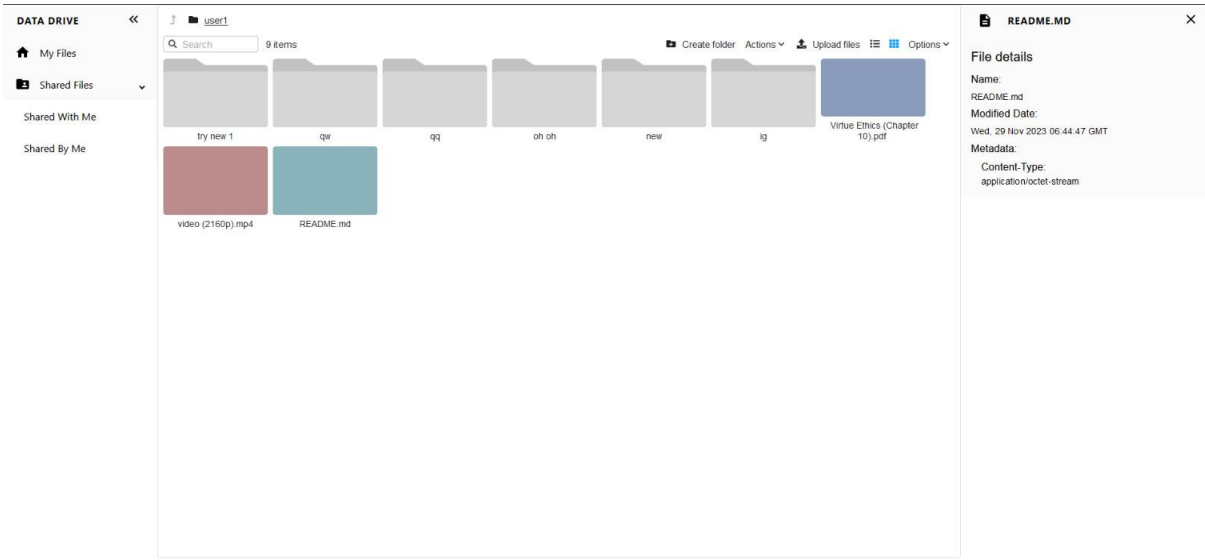
1. User Creation and Bucket Management

The application has a user creation and login page. When a user is created, a corresponding folder is created in the minio and authorization token is generated from the minio server, which we use to manage access of a user to the buckets, and manages access of users to the buckets. The user authorisation, entails the user to access only the bucket and the content stored in the allocated bucket. An SQL server is used to store the information regarding user credentials, bucket name in which the files/folders related to the particular user are stored, the storage space used by the user and the storage limit for the user.

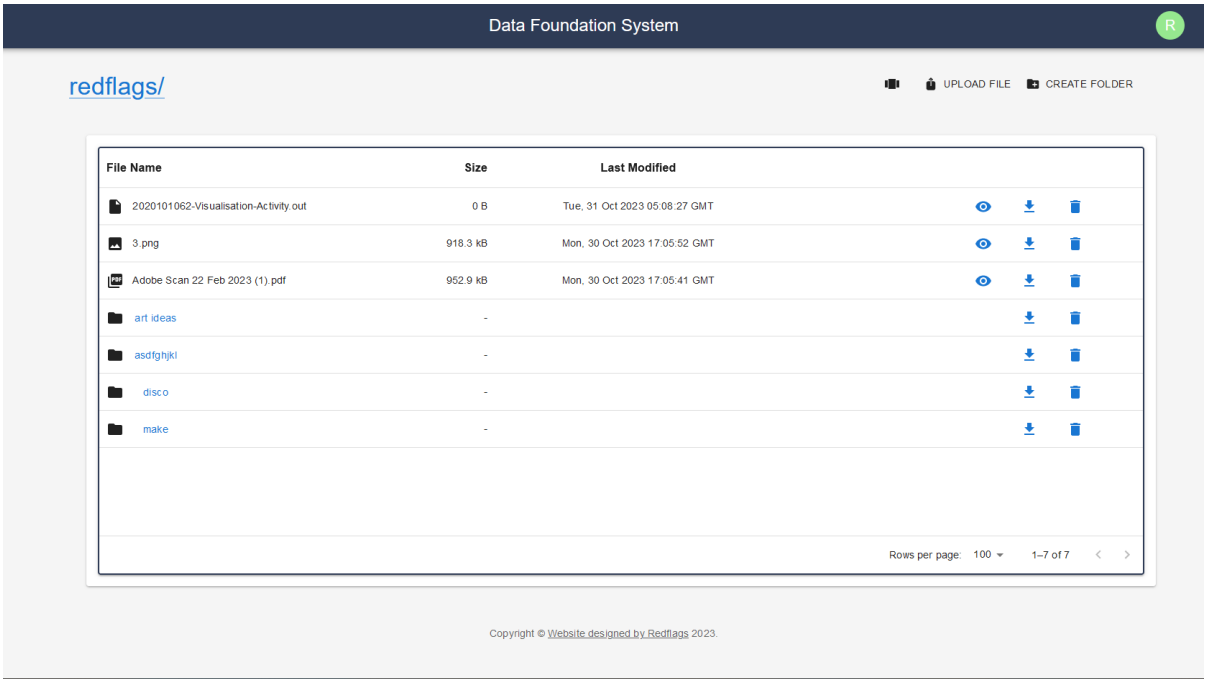
2. Dashboard Layout

The dashboard could be split into multiple components, stacked together:

a. The Top bar:
   Renders, the profile icon
b. Navigation Pane
   The section to the left, renders the different sections and components of the data-drive, like shared etc. for ease of access of the files and folders, and their management by the user.
c. Content Pane
   This section renders the files and contents from the user bucket.
   For the ease of feasibility, and usage, we have implemented 2 different views, for listing the folders and files within the current directory:

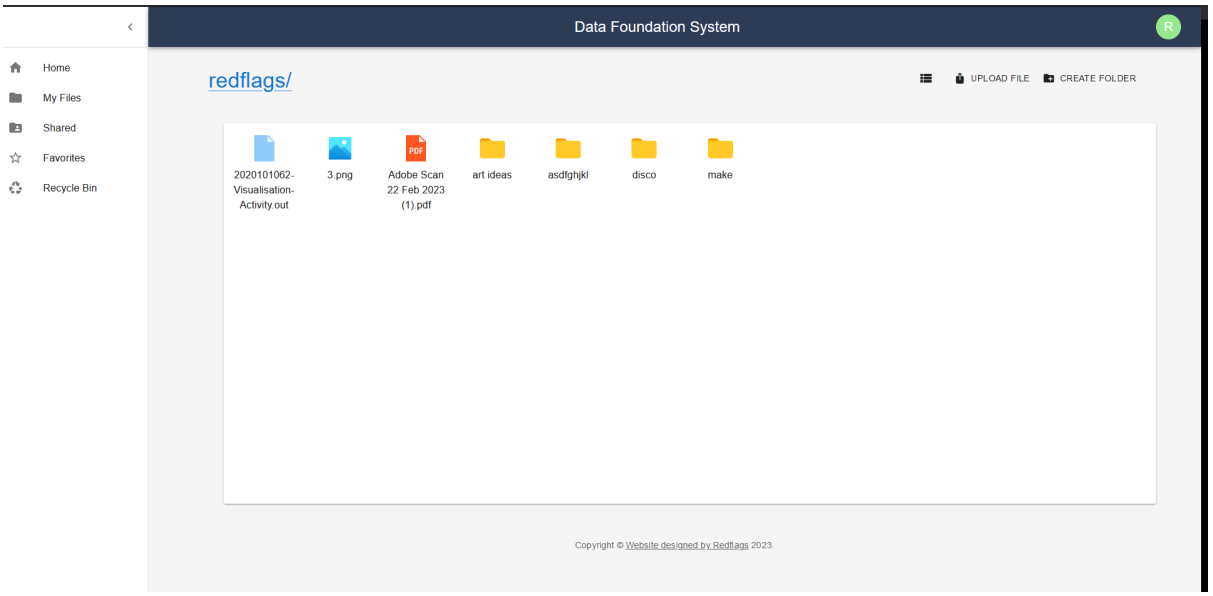


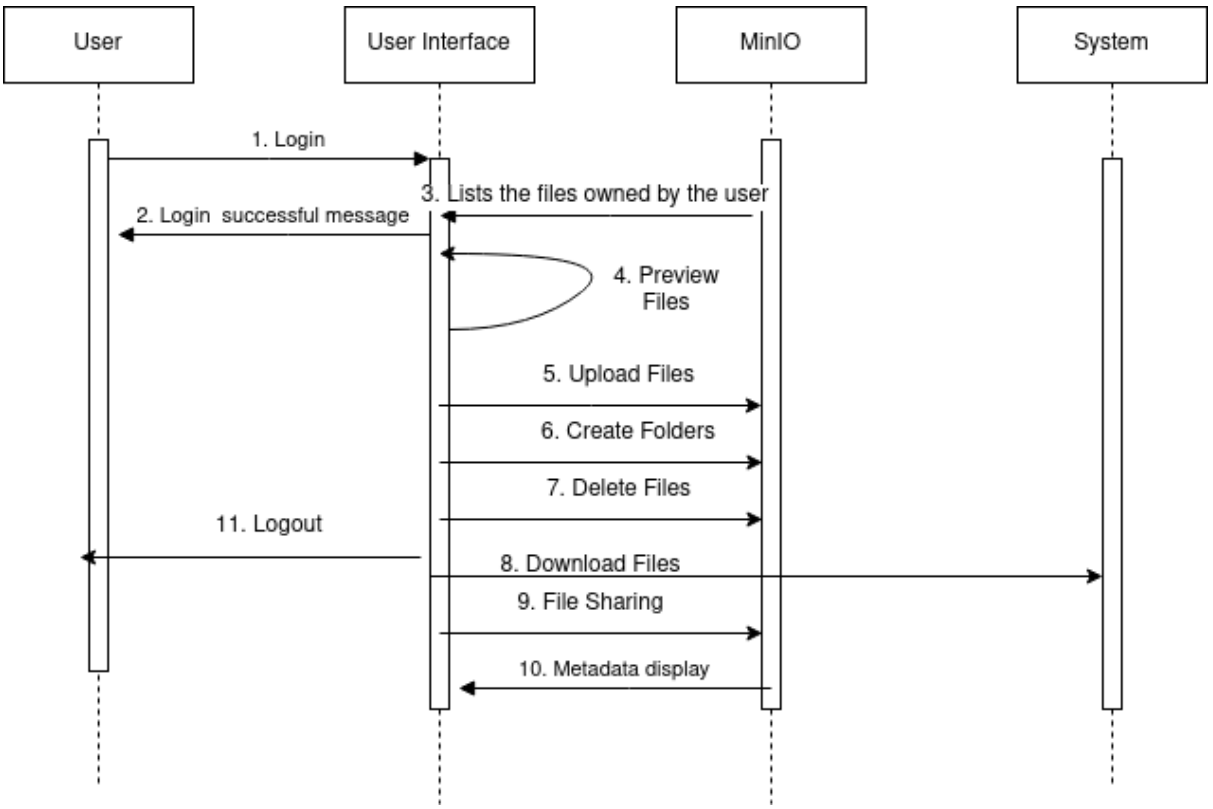
1. List View
   In this view, all the items are listed as a list, with their corresponding icons, file size, last Modified date etc. and a set of action buttons for preview, download and deletion of the selected file.



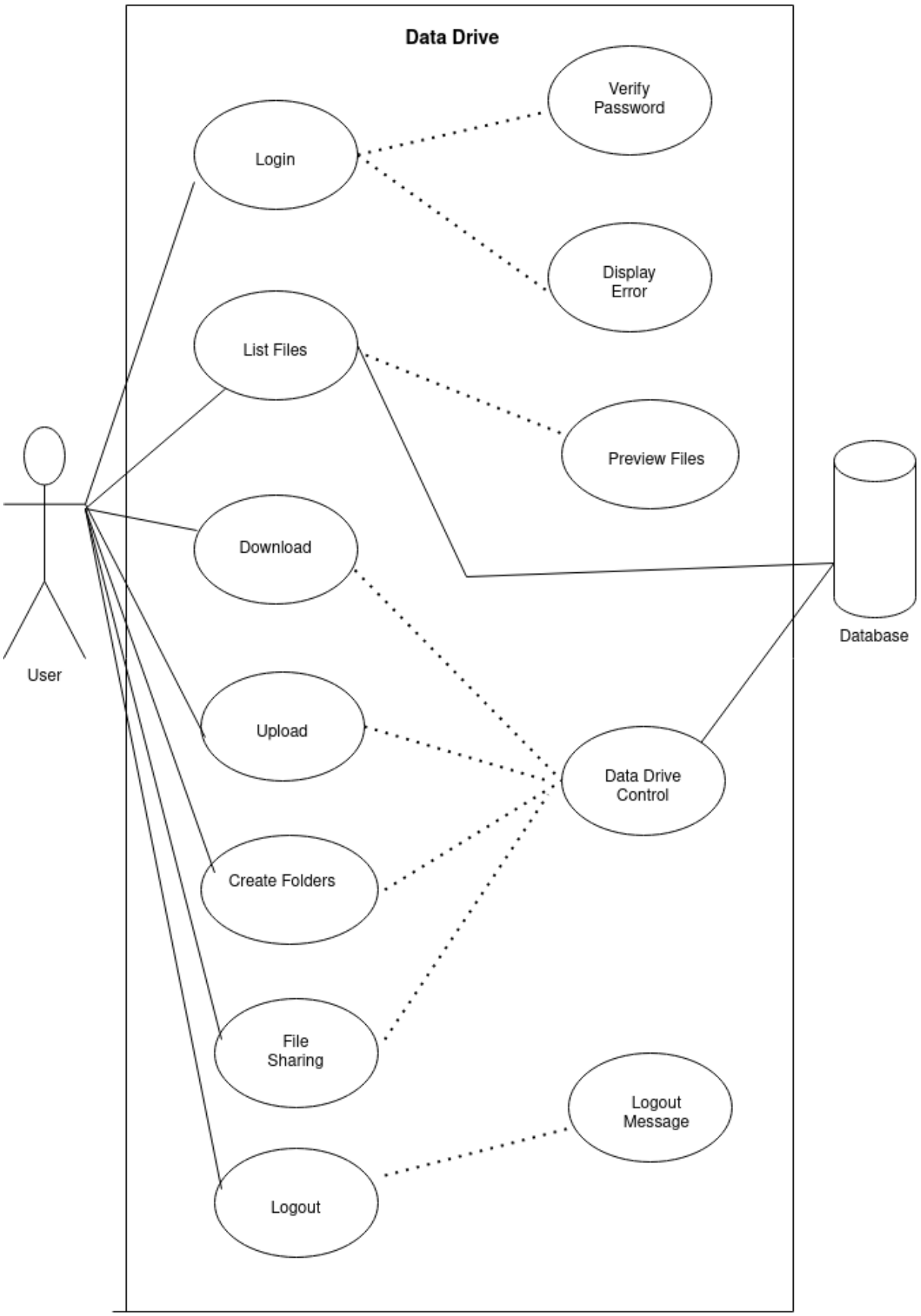
2. Grid View

   In this view, the items are listed, in a responsive grid style fashion, with corresponding file-icons for each of the file, and folder within the directory.





Sequence Diagram



Use Case Diagram

The user, can toggle between the different views using a toggle button present on the dashboard.

Since, the number of files, and folders could be a large number for a given user, we implemented pagination using ‘Datagrid’ component from MUI, that displays, a certain number of files/folders on each page, allowing user to easily navigate and access the data stored in the bucket storage.

3. Folder System and Navigation

We Implemented the folder, the main challenge over here, is that minio-sdk doesn’t provide the size, last modified meta data for folder systems, hence we plan to store and retrieve these details in our custom-meta data, and update/modify them, whenever the directory is updated.

Whenever the folder is clicked, the content within that directory is rendered into the content-pane, and the current-directory gets updated. The user can navigate amongst the folder structure using, the hyperlink.

The user can create/upload a folder from the react-dashboard onto the user, bucket.

4. File Interactions

The rendered file, component could provide below listed features:

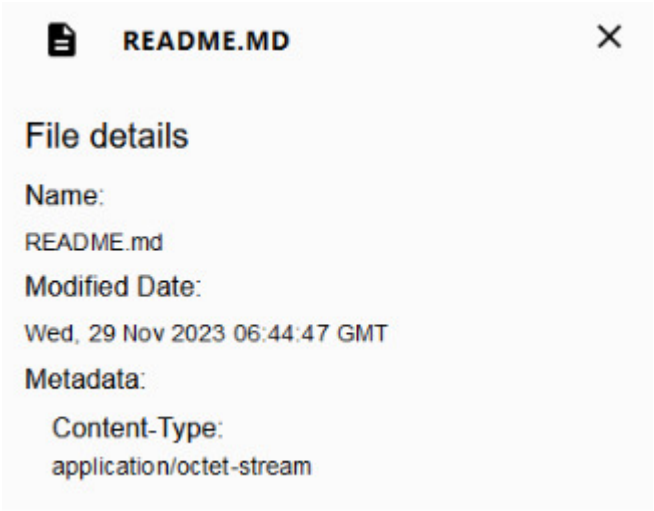
1. Preview
   We have built a custom component, for rendering different types of data files - pdfs, images, audio and video in a good and presentable format using the react-pdf, etc. for rendering the pdf files.
   Using this feature, the user could preview the content without downloading the content.
2. Download
   The user could, download a copy of the folder/file from the hosted minio server into the local system
3. Upload
   The user can upload a file/folder into the current directory.
4. Delete
   This operation deletes the folder/file from the minio server.
5. Share

We allowed sharing of files. Files owned by one user can be shared to other users with read/write permissions. Public sharing of sharing can also be done.

5. Display of Metadata

When a file is selected, the details of the file are displayed on the right side of the page. The File details include:

1. Name
   The name of the file is displayed along with the appropriate extension
2. Modified Date:
   The day, date, time at which the file was last modified is also stored.
3. Metadata:
   The Content-Type of the file, which is the type of file is also stored in the metadata.



Future Extensions

1. Integrate with DFS minIO client, to further test the code.
2. Implement Favourites tact.b.
3. Work on the UI to make it more seamless and minimal.