Software Requirements Specification

Shareflow Project

Created by Antoine Ratat

Version 1.0 - Issued January 25,2024

This requirements specification is used to record the user requirements.

1. VERSIONS

Ver.	Author(s)	Date	Description
1.0	Antoine RATAT	25/01/2024	Creating Document

2. INTRODUCTION

2.1 Purpose

The purpose of ShareFlow is to provide a platform for users to store, share, and collaborate on their files and documents. ShareFlow allows users to easily upload and organize their files, create folders, and share them with others. With its user-friendly interface and robust features,

ShareFlow facilitates seamless collaboration among individuals or teams, enabling real-time editing, commenting. Users can access their files from any device with an internet connection, ensuring easy accessibility and continuity of work. ShareFlow aims to simplify file management and enhance productivity, serving as a reliable tool for individuals, businesses, and organizations alike.

2.2 Project Scope

The project scope for my shareflow application includes designing and developing a platform for users to easily share and collaborate on documents, files, and ideas. The application will have features such as real-time collaboration, activity tracking. The scope also includes ensuring the application is accessible on various devices, user-friendly, and secure. Regular updates and enhancements will be necessary to adapt to changing user needs and technological advancements.

- **Document sharing**: provides a platform to share files and collaborate with colleagues or clients, ensuring easy access and efficient communication.

- **Commenting and feedback**: enables users to comment on specific sections of a document, facilitating feedback and discussion, promoting effective collaboration.

- **File storage**: serves as a centralized storage system where users can save and access their files, maintaining an organized and secure repository.

- **Security and access controls**: provides robust security measures to protect sensitive data and confidential information, allowing users to control access permissions for their shared documents.

- **Mobile compatibility**: ensures the application's functionality and usability across various devices, allowing users to collaborate on the go.

3. DESCRIPTION

3.1 Product Perspective

The product is a full-stack web application implemented on both the front and back end, it is using API calls to operate the database.

3.2 Features

The following are the main features that are included in the system:

3.2.1 File Management

➔ **Upload File**
Allows users to add files to the system.

➔ **Rename File**
Enables users to change the name of a file.

➔ **Trash File**
Temporarily moves a file to the trash.

➔ **Remove File**
Permanently deletes a file from the system.

➔ **Restore File**
Recovers a file from the trash to its original location.

➔ **Favorite File**
Flags a file as a favorite for quick access.

➔ **Comment on File**
Permits users to add comments to provide context or information about a file.

➔ **Delete Comment**
Removes a comment associated with a file.

➔ **Add Tags to File**
Attaches tags to a file for better organization and searchability.

➔ **Search Files by Tag**
Allows users to find files based on assigned tags.

➔ **Search Favorite Files**
Displays a list of files marked as favorites.

➔ **View Recently Deleted Files**
Shows a list of files that have been recently deleted.

➔ **Download File**
Allows users to download a copy of a file.

➔ **Preview File**
Provides a preview of the content of a file without fully opening it.

3.2.2 Folder Management

➔ **Create Folders**
Enables users to create new folders and organize files within them.

➔ **Navigate Folders Tree**
Allows users to explore the hierarchical structure of folders.

➔ **Keep Track of Activity Feed**
Records and displays a chronological list of user activities within the system.

3.2.3 Performance

➔ **Async Data Loading**
Ensures asynchronous loading of data for improved user experience.

➔ **Loading State**
Each page has a loading state to handle API delays.

➔ **Error State**
Each page includes an error state if no data is returned from the API.

➔ **Disable Actions While Loading**
Disables user actions during data loading to prevent unintended interactions.

3.2.3 Security

➔ **HTTPS**
Uses HTTPS to transmit sensitive data securely.

➔ **JSON Web Token (JWT)**
Utilizes JWT to authenticate users and transmit information in the payload.

➔ **Cross-Origin Resource Sharing (CORS)**
Utilizes remote API with CORS to allow communication across different origins.

➔ **Data Validation (Front-end)**
Implements data validation in the front-end using Formik and Yup.

➔ **Data Validation (Back-end)**
Enforces data validation on the back-end for additional security.

➔ **Verify User Email**
Verifies user email to activate the account, enhancing security.

3.2.4 Language

➔ **English**
The website is entirely in English.

3.2.5 Design

➔ **Graphic Charter**
Adheres to a strict graphic charter for consistency and uniformity on the website, including colors, font, and graphics.

➔ **Responsive Design**
The website design automatically adjusts for different screen sizes and viewports.

➔ **MUI**
Utilizes MUI for element positioning and enhanced design.

3.3 User Overview

Users are assumed to possess basic internet operation knowledge and have internet access.

Administrators are expected to be well-versed in the following technologies for Front-End development: HTML, CSS, JavaScript, React library, AJAX (Asynchronous JavaScript and XML), React Router, Redux, TypeScript, React Token Authentication, React Dropzone, React Hook Form and MUI.

From the Back-End perspective, administrators should be familiar with the following technologies: Node, Express, CORS configuration, JWT Tokens, Authentication, and Password hashing mechanisms.

This set of skills will empower administrators to effectively manage and navigate both the Front-End and Back-End aspects of the system.

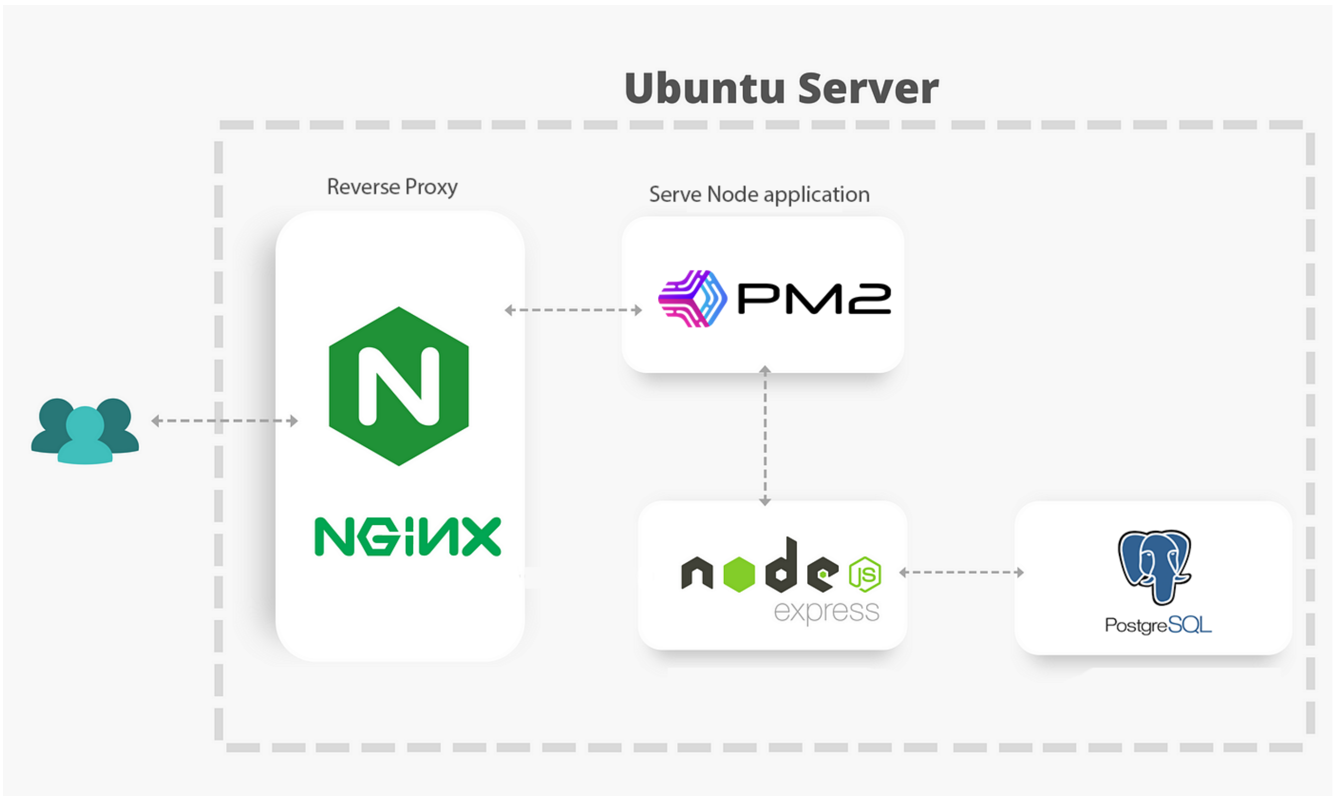
3.4 Operating Environment

This is a web-based system and hence will require the operating environment for a client and server GUI.

Dependencies: This software highly depends on the type and version of the browser being installed in the system i.e. browser version should be used which has HTML5 support.

3.5 Constraints: Implementation / Design

3.5.1 Application Communication Schema



3.5.2 Client-side rendering (frontend)

This system is provisioned to be built in JavaScript using React library which is highly flexible.

The browser will be in charge of rendering this application in its final form, HTML. Some of the logic involved in creating the web page, especially the one in charge of dealing with presentation logic is handled on the client-side.

List of frontend dependencies and version used:

@emotion/react: v11.11.1
@emotion/styled: v11.11.0
@material-ui/core: v4.12.4
@mui/material: v5.14.19
@reduxjs/toolkit: v2.0.1
jwt-decode: v4.0.0
lodash: v4.17.21
query-string: v8.1.0
react: v18.2.0
react-dom: v18.2.0
react-dropzone: v14.2.3
react-hook-form: v7.49.2
react-redux: v9.0.4
react-router-dom: v6.20.1
redux: v5.0.0
uuid: v9.0.1

3.5.3 Server-side rendering (backend)

The system is designed to connect and communicate with a custom-made API, which manages CRUD requests and operations on the database.

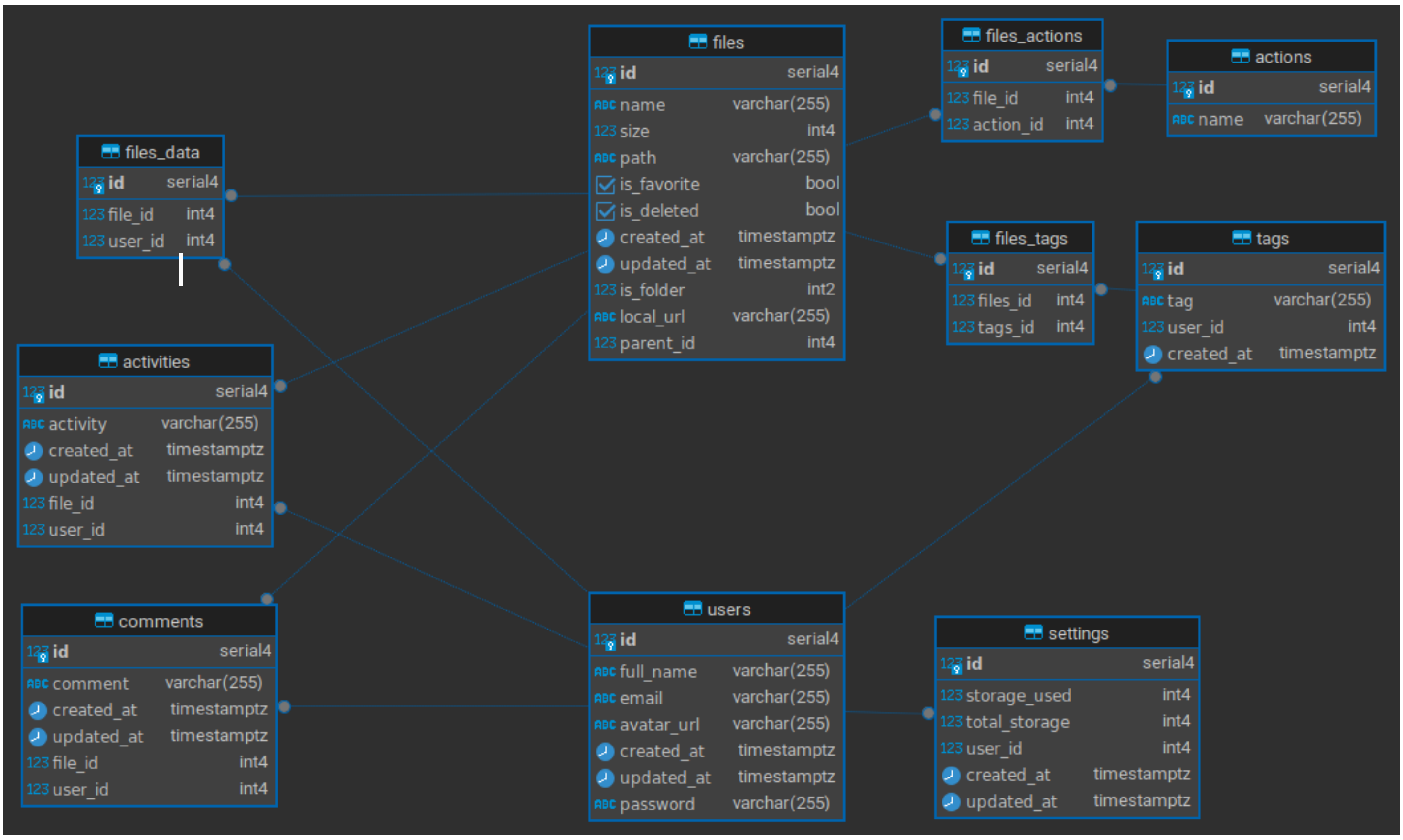
The API is constructed using Node.js with the Express web framework and connects to a PostgreSQL database. The application server is served using PM2, a process manager for Node.js applications, and is deployed behind an Nginx reverse proxy for enhanced security and performance.

Authentication is implemented using the Flask-JWT-Extended library, which introduces support for JSON Web Tokens (JWT) in Flask to secure views.

List of used dependencies and their versions:

archiver: v6.0.1
bcrypt: v5.1.1
cors: v2.8.5
dotenv: v16.3.1
express: v5.0.0-beta.1
helmet: v7.1.0
jsonwebtoken: v9.0.2
moment: v2.30.1
multer: v1.4.5-lts.1
pg: v8.11.3

You can find an ER Entity Relationship Diagram to understand its structure:



files	
id	serial4
name	varchar(255)
size	int4
path	varchar(255)
is_favorite	bool
is_deleted	bool
created_at	timestamptz
updated_at	timestamptz
is_folder	int2
local_url	varchar(255)
parent_id	int4

files_actions	
id	serial4
file_id	int4
action_id	int4

actions	
id	serial4
name	varchar(255)

files_data	
id	serial4
file_id	int4
user_id	int4

files_tags	
id	serial4
files_id	int4
tags_id	int4

tags	
id	serial4
tag	varchar(255)
user_id	int4
created_at	timestamptz

activities	
id	serial4
activity	varchar(255)
created_at	timestamptz
updated_at	timestamptz
file_id	int4
user_id	int4

comments	
id	serial4
comment	varchar(255)
created_at	timestamptz
updated_at	timestamptz
file_id	int4
user_id	int4

users	
id	serial4
full_name	varchar(255)
email	varchar(255)
avatar_url	varchar(255)
created_at	timestamptz
updated_at	timestamptz
password	varchar(255)

settings	
id	serial4
storage_used	int4
total_storage	int4
user_id	int4
created_at	timestamptz
updated_at	timestamptz

3.6 System Feature Frontend

3.6.1 Not Authenticated Features

➔ **Create an account**
Allow users to register an account

➔ **Login**
Allow user to connect their account

3.6.2 Authenticated Features

➔ **Upload File:**
Allows users to add files to the system.

➔ **Rename File:**
Enables users to change the name of a file.

➔ **Trash File:**
Temporarily moves a file to the trash.

➔ **Remove File:**
Permanently deletes a file from the system.

➔ **Restore File:**
Recovers a file from the trash to its original location.

➔ **Favorite File:**
Flags a file as a favorite for quick access.

➔ **Comment on File:**
Permits users to add comments to provide context or information about a file.

➔ **Delete Comment:**
Removes a comment associated with a file.

➔ **Add Tags to File:**
Attaches tags to a file for better organization and searchability.

➔ **Search Files by Tag:**
Allows users to find files based on assigned tags.

➔ **Filter Favorite Files:**
Displays a list of files marked as favorites.

➔ **View Recently Deleted Files:**
Shows a list of files that have been recently deleted.

➔ **Download File:**
Allows users to download a copy of a file.

➔ **Preview File:**
Provides a preview of the content of a file without fully opening it.

➔ **Create Folder and Subdirectories:**

Enables users to create new folders and organize files within them.

➔ **Navigate Folders Tree:**
Allows users to explore the hierarchical structure of folders.

➔ **Keep Track of Activity Feed:**
Records and displays a chronological list of user activities within the system.

3. 8 System Feature Database

actions
Stores information about different actions.

activities
Records various activities, associating them with a file and a user.

comments
Manages comments linked to specific files and users.

files
Holds details about files, including their name, size, path, and other attributes.

files_actions
Represents the relationship between files and actions.

files_data
Manages data associated with files and users.

files_tags
Establishes connections between files and tags for organizational purposes.

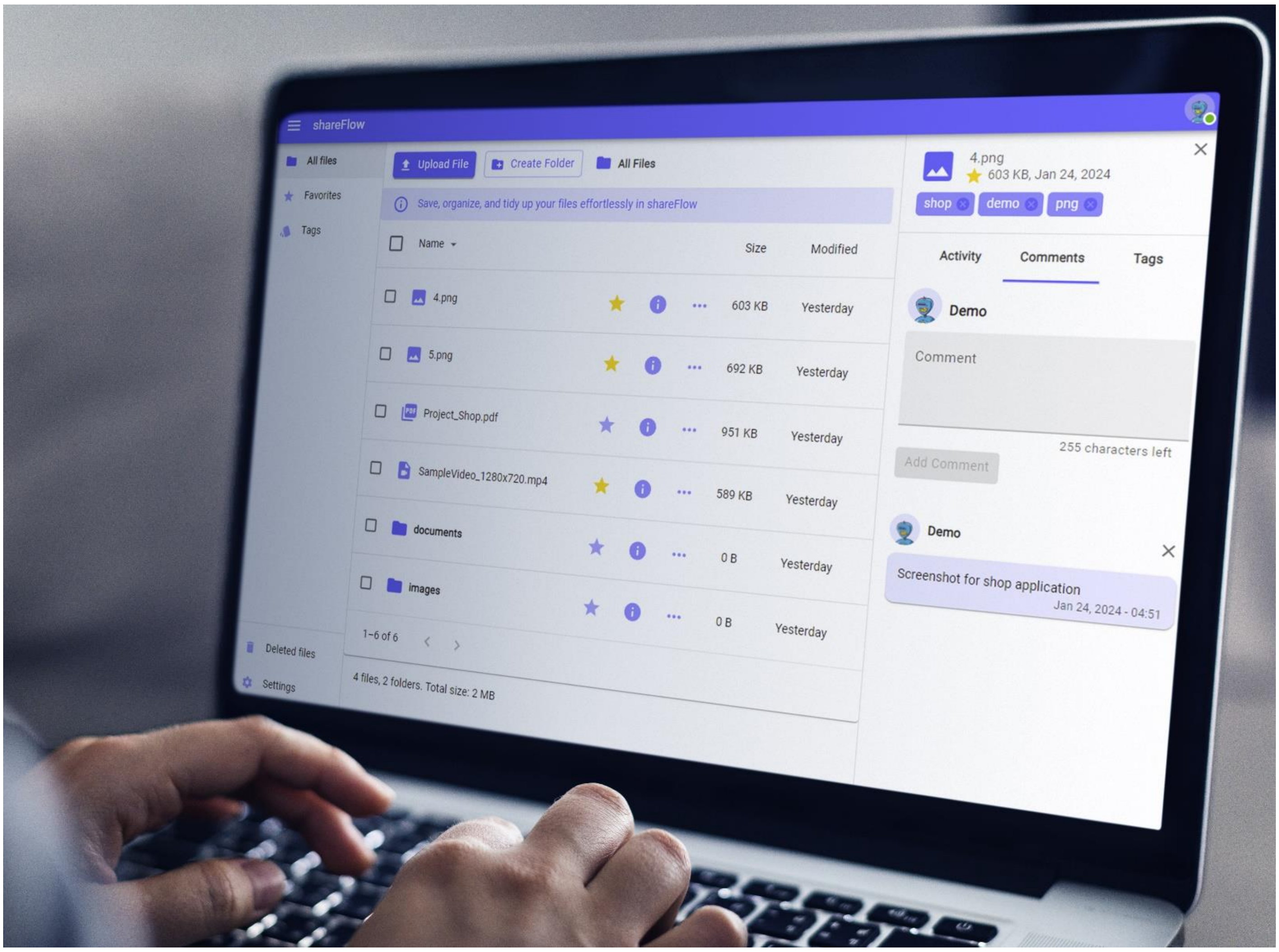
settings
Stores user-specific settings, such as storage usage and total storage limits.

tags
Contains information about tags created by users.

users
Stores user details, including full name, email, avatar URL, and password.

4. REQUIREMENTS OF EXTERNAL INTERFACE

4.1 User Interfaces



Save, organize, and tidy up your files effortlessly in shareFlow

Name	Size	Modified
4.png	603 KB	Yesterday
5.png	692 KB	Yesterday
Project_Shop.pdf	951 KB	Yesterday
SampleVideo_1280x720.mp4	589 KB	Yesterday
documents	0 B	Yesterday
images	0 B	Yesterday

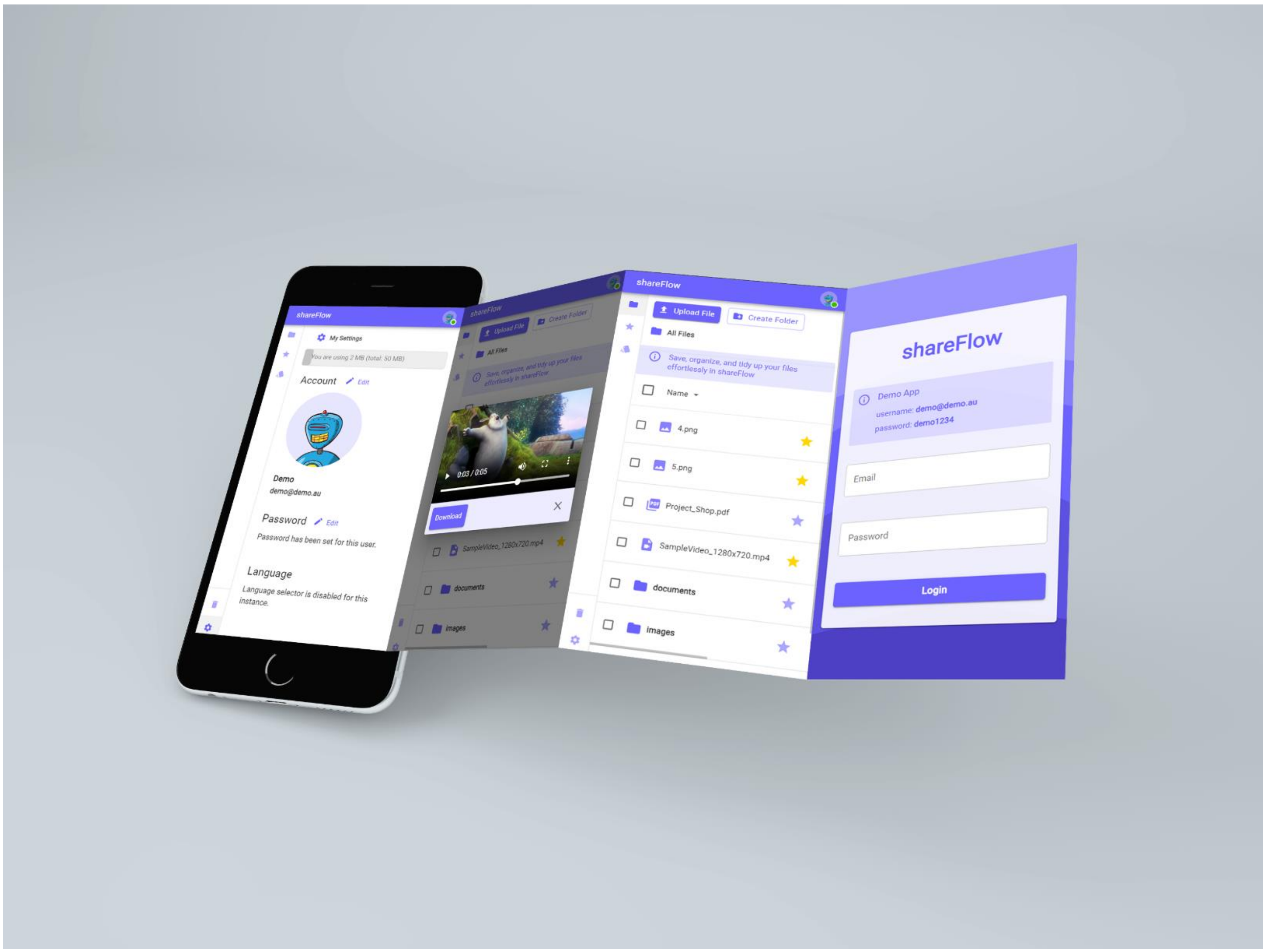
1–6 of 6

4 files, 2 folders. Total size: 2 MB

4.png
603 KB, Jan 24, 2024
shop demo png

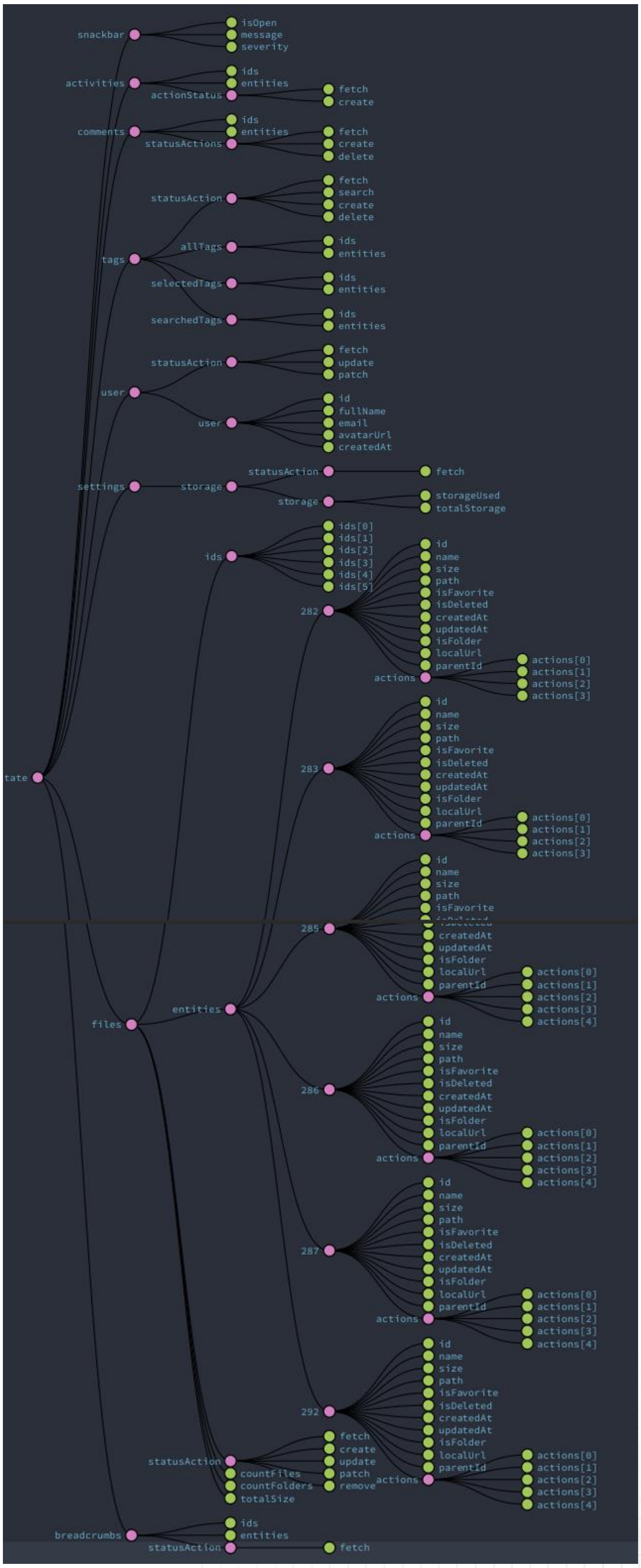
Demo
Comment
255 characters left

Demo
Screenshot for shop application
Jan 24, 2024 - 04:51



4.2 Software Interfaces

The application utilizes Redux for state management on the front end. The central store holds the main slices of the application state



4.3 Communication Interfaces

Communication is assured to external interfaces. The system is connected to several APIs using REST, the payload is defined in the request itself and is formatted in JSON. This API is a CRUD API created with Express to communicate safely with a PostgreSQL database.

5. NON-FUNCTIONAL REQUIREMENTS

5.1 Performance

The system must be interactive and the delays involved must be fewer. So, in every action-response of the system, there are no immediate delays. In the case of opening new windows, popping error messages, and saving the settings or sessions there is a delay depending on the API response time and availability.

In the case of calling external APIs, the delay is based on editing on the distance of the secondary system and the configuration between them so there is a high probability that there will be or not a successful connection in less than 5 seconds for sake of good communication.

The following also needs to be implemented:

- Async Data Loading: To Load prior fast elements
- Each page has a loading state to handle API delay
- Each page has an Error state if no data is returned from the API
- Disable actions while loading

5.2 Availability

In the event of an internet service disruption during the retrieval of information from external APIs, the system allows for a subsequent attempt to fetch the information.

The system heavily relies on external APIs, and if communication between the client and the API is interrupted, the system supports re-requesting the information.

The API accessed is: https:antoineratat.xyz/api_shareflow

5.3 Security

- Check input data with Form Validation using Formik and Yup
- Back end verification before any database operation
- Authentication mechanism with JWT
- HTTPS transfer protocol to secure communication over the network
- Hashing sensitive information with Bcrypt before storing it in the database
- Storing an application's sensitive information in an environment variable on the server
- JSON Web Token to Authenticate user and transmit information in the payload
- Use remote API with Cross-Origin Resource Sharing (CORS) to allow communication origin

5.4 Usability

As the system is easy to handle and navigates in the most expected way with no delays. In that case, the system program reacts accordingly and transverses quickly between its states.