COMPUTING PROJECT

File Repository Web App

Loculus

Table of Contents

1. Introduction

The Loculus project is an online repository that allows users to store their files on a server. The concept behind Loculus, derived from the Latin word, represents a small place for storing items. The project aims to address issues related to physical data storage, portability, and sharing by providing users with a centralized and efficient solution.

1.1 Purpose

The purpose of the Loculus project is to provide users with a secure and user-friendly web application for storing, sharing, and managing their files. It aims to alleviate the challenges associated with physical storage, data portability, and efficient collaboration.

1.2 Target Users

The Loculus web app is designed to cater to a broad range of users, including individuals, teams, students, and enterprises. It is suitable for anyone experiencing issues related to data storage, organization, and collaboration. The user base can include professionals, students, researchers, and individuals from various industries who require a reliable and accessible platform to store and work with their digital assets.

1.3 Main Features and Functionalities

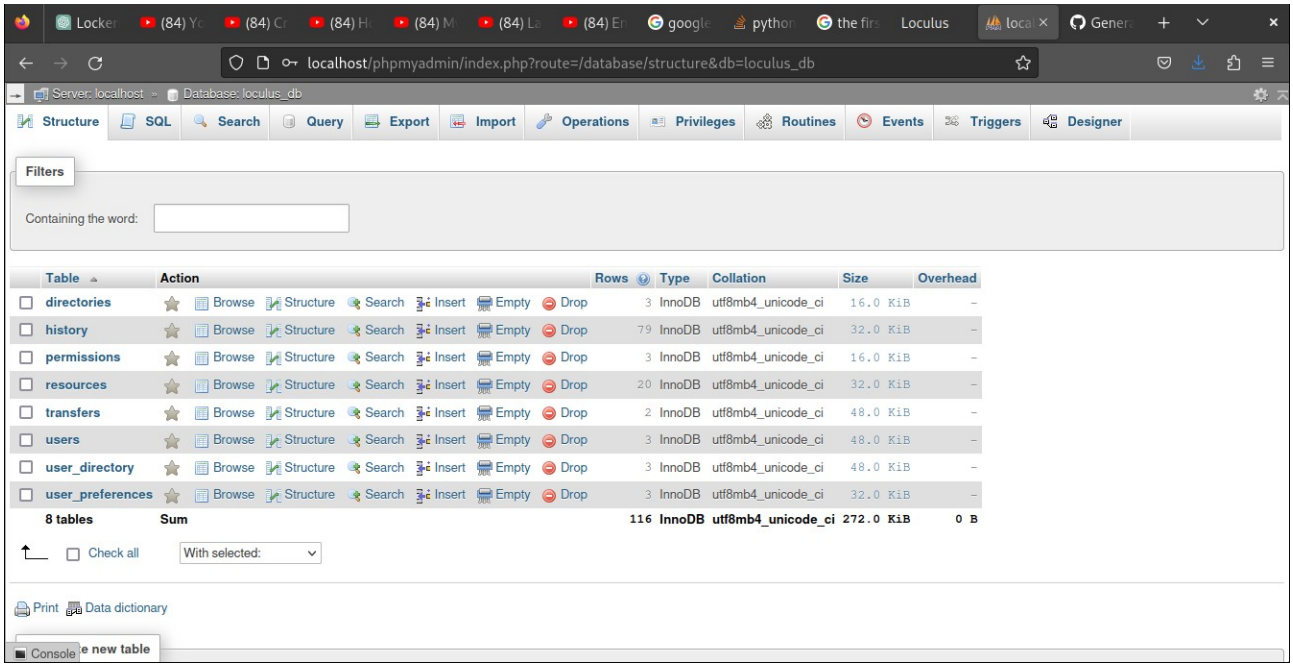
The Loculus project offers a range of features and functionalities to enhance the user experience and facilitate efficient file management:

- **User Registration and Login:** Users can create an account and securely log in to the Loculus web app.
- **File Upload and Storage:** Users can upload their files to the server and store them in dedicated directories.
- **File Sharing and Collaboration:** Users can share files and directories with others, enabling efficient collaboration and document sharing.
- **Workspace for File Interaction:** Users can open files in a dedicated workspace, allowing them to interact with multiple files simultaneously.
- **File Editing:** Loculus provides basic file editing capabilities, enabling users to make changes to supported file types directly within the application.
- **Directory Sharing:** Users can share entire directories (Loculus) with others, granting them access to the contained files and subdirectories.
- **Security and Privacy:** The Loculus project emphasizes data security and privacy, ensuring that user files are protected from unauthorized access.
- **User-Friendly Interface:** The web app is designed with a user-friendly interface to enhance usability and provide an intuitive experience for users.

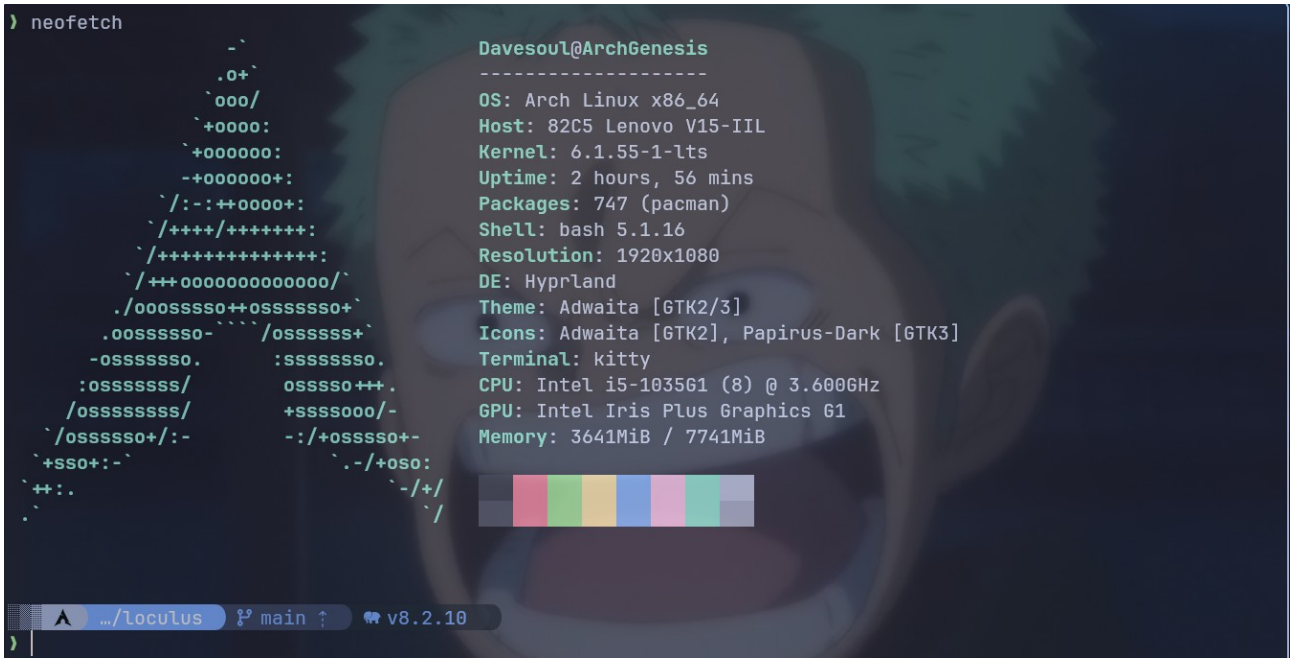
1.4 Technologies and Tools Used

The Loculus project was developed using a combination of technologies and tools, including:

- **Front-End:** HTML, CSS (Learn CSS in One Day), JavaScript (Javascript For Beginners)
- **Back-End:** PHP (PHP Tutos)
- **Database:** MySQL
- **Integrated Development Environment (IDE):** Visual Studio Code



- **Database Management:** phpMyAdmin, MySQL Workbench
- **Operating System:** Linux (Ubuntu and Arch linux)



The choice of these technologies and tools was based on their suitability for web development, database management, the project requirements and their availability.

2. Requirements Analysis

Requirements analysis is a critical phase in developing the Loculus web app. It involves identifying and documenting the specific functional and non-functional requirements to ensure the successful implementation of the project.

2.1 Functional Requirements

The functional requirements of the Loculus web app are as follows:

1. **User Registration and Authentication:**
    - Users should be able to create an account with a unique username and email address.
    - Users should be able to log in securely to access their accounts.
2. **File Management:**
    - Users should be able to upload files to the server and store them in dedicated loculi (directories).
    - The web app should support file organization, allowing users to create, rename, and delete directories.
    - Users should be able to navigate through the directory structure and view the files within each directory.
3. **File Sharing and Collaboration:**
    - Users with appropriate permissions should be able to share files and directories with others.
    - The web app should provide options to specify read, write, and delete permissions for shared files and directories.
    - Collaboration features such as comments or notifications can be implemented to facilitate communication among users.
4. **File Workspace and Interaction:**

- Users should be able to open files in a dedicated workspace, allowing them to view and interact with multiple files simultaneously.
- Basic file editing capabilities should be provided, enabling users to make changes to supported file types within the application.
- The web app should allow users to preview files, such as images or documents, within the workspace.

5. **User Roles and Permissions:**

- User roles should be defined, including Admin, Standard, and Guest roles.
- Admin users should have read, write, share, download, and delete permissions for all files and directories.
- Standard users should have read, share, and download permissions for files and directories.
- Guest users should have read-only access to files and directories.

6. **Search Functionality:**

- Users should be able to search for files in a loculus based on their names or relevant metadata.

2.2 Non-Functional Requirements

The non-functional requirements of the Loculus web app are as follows:

1. **Security:**
    - The web app should ensure secure user authentication and protect user data from unauthorized access.
    - Proper data encryption and secure communication protocols should be employed.
2. **Performance:**
    - The application should be designed and optimized for efficient performance, even with large file sizes or high user traffic.
3. **Scalability:**
    - The architecture of the web app should be scalable to accommodate a growing number of users and increasing file storage requirements.
4. **Usability:**
    - The user interface should be intuitive and user-friendly, allowing users to navigate and interact with the application seamlessly.
5. **Accessibility:**
    - The web app should comply with accessibility standards, ensuring that it can be used by individuals with disabilities.

2.3 File Types and Size Limitations

The specific file types and size limitations for the Loculus web app will be determined based on your requirements. Please provide the file types and any restrictions or limitations you would like to impose.

2.4 External Systems and APIs

The Loculus web app currently utilizes the following external systems and APIs:

- **Google Fonts:** This service is used to incorporate custom fonts within the web app.
- **Font Awesome:** This resource is utilized to enhance the user interface with icons.

Important : font awesome is currently the only source of icons for the app therefore the user should be connected to the internet.

3. System Design

The System Design section outlines the architectural design and components of the Loculus web app, including the utilization of the Model-View-Controller (MVC) architecture. It also includes the file structure, the database design (data dictionary and ERD), and a detailed Data Flow Diagram (DFD) to illustrate the data flow within the system.

3.1 Architectural Design

The Loculus web app follows the Model-View-Controller (MVC) architecture, which separates the application into three interconnected components:

1. **Model**: The Model component represents the data and business logic of the Loculus web app. It manages the data storage, retrieval, and manipulation operations. In the context of Loculus, the Model component interacts with the database to handle user data, file storage, and other related operations.
2. **View**: The View component is responsible for presenting the user interface to the users. It represents the HTML, CSS, and JavaScript code that generates the web pages seen by the users. In Loculus, the View component includes the GUI elements, forms, file listings, and the workspace interface.
3. **Controller**: The Controller component acts as an intermediary between the Model and View components. It handles user actions and triggers the corresponding operations in the Model or View. In Loculus, the Controller component manages user input, form submissions, file uploads, sharing operations, and interactions with the database.

The use of the MVC architecture in the Loculus web app provides separation of concerns, modularity, and flexibility, enabling easier maintenance and future enhancements.

3.2 File Structure

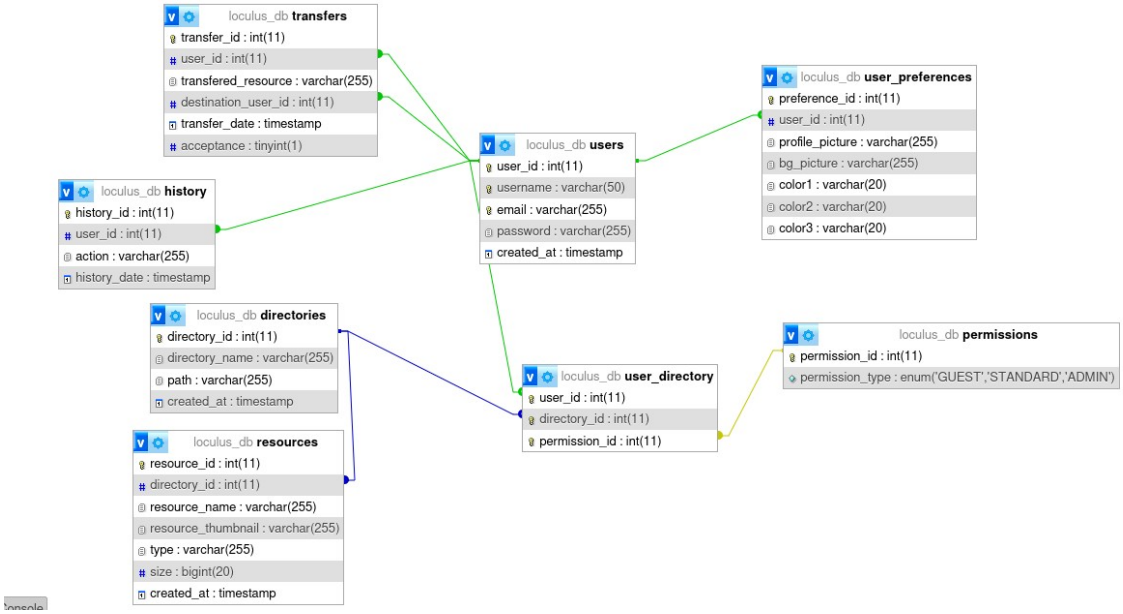
File structure of the Loculus web app:

Here's an overview of the directory structure and the contents of each directory:

1. **controller:** This directory contains PHP files responsible for controlling various aspects of the application, such as permissions, database connections, and file updates.
2. **database:** It contains the loculusdb.sql file, which is likely a SQL database dump used to create the database schema for your application.
3. **image:** This directory includes image files, such as the default user avatar and a PDF file.
4. **model:** It contains the model.php file, which is likely used to handle the business logic and interactions with the database.
5. **view:** This directory is structured further:
    - **css:** Contains the style.css file for styling.
    - **js:** Contains JavaScript files, including header.js and loculus.js.
    - **php:** Contains PHP files for rendering different views, such as login, signup, and transfers.
6. **Other Files:**
    - file_upload_changes.txt: This file might contain information about changes related to file uploads.
    - Loculus_Documentation.odt: An Open Document Text (ODT) file that could be documentation for the project.
    - readfile.php: A PHP file that likely handles reading files.
    - README.md: A common practice is to use this file to provide documentation and instructions for your project.

3.3 Database Design

The database design for the Loculus web app consists of a data dictionary and an Entity-Relationship Diagram (ERD).



3.3.1 Data Dictionary

Below is a data dictionary for the Loculus database:

Table: users

- user_id (Primary Key): Unique identifier for users.
- username: User's username (must be unique).
- email: User's email address (must be unique).
- password: User's password.
- account_creation_date: Timestamp indicating the account creation date.

Table: directories

- directory_id (Primary Key): Unique identifier for directories.

- directory_name: Name of the directory.
- path: Path to the directory.
- directory_creation_date: Timestamp indicating the directory creation date.

Table: resources

- resource_id (Primary Key): Unique identifier for resources.
- directory_id (Foreign Key): References the directory that the resource belongs to.
- resource_name: Name of the resource.
- resource_thumbnail: Thumbnail for the resource.
- type: Type of the resource.
- size: Size of the resource in MB.
- resource_creation_date: Timestamp indicating the resource creation date.

Table: permissions

- permission_id (Primary Key): Unique identifier for permissions.
- permission_type: Type of permission (e.g., GUEST, STANDARD, ADMIN).

Table: user_directory

- user_id (Foreign Key): References the user who has permission for the directory.
- directory_id (Foreign Key): References the directory.
- permission_id (Foreign Key): References the permission granted for the user's access to the directory.
- Composite Primary Key: A combination of user_id, directory_id, and permission_id.

Table: history

- history_id (Primary Key): Unique identifier for history entries.
- user_id (Foreign Key): References the user associated with the history entry.
- action: Describes the action recorded in the history.

- history_date: Timestamp indicating the date and time of the recorded action.

Table: transfers

- transfer_id (Primary Key): Unique identifier for transfers.
- user_id (Foreign Key): References the user initiating the transfer.
- transfered_resource: Name of the transferred resource.
- destination_user_id (Foreign Key): References the user receiving the transferred resource.
- transfer_date: Timestamp indicating the date and time of the transfer.
- acceptance: Indicates whether the transfer has been accepted (0 for not accepted, 1 for accepted).

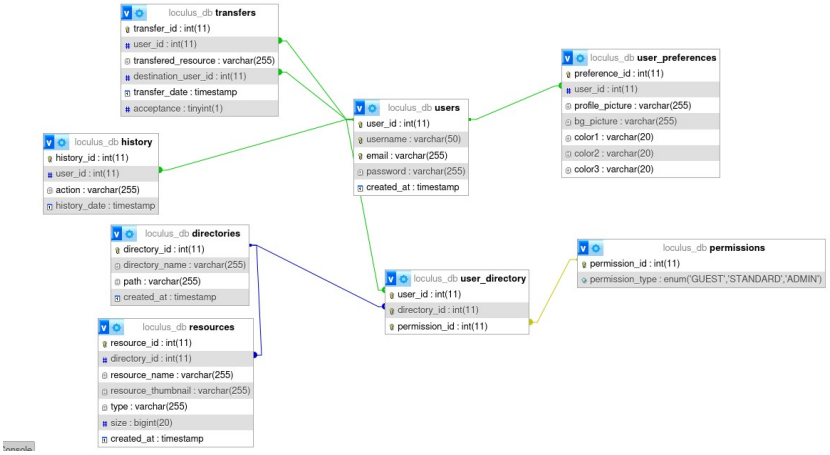
Table: user_preferences

- preference_id (Primary Key): Unique identifier for user preferences.
- user_id (Foreign Key): References the user associated with the preferences.
- profile_picture: Path to the user's profile picture.
- bg_picture: Path to the user's background picture.
- color1: User's color preference 1.
- color2: User's color preference 2.
- color3: User's color preference 3.

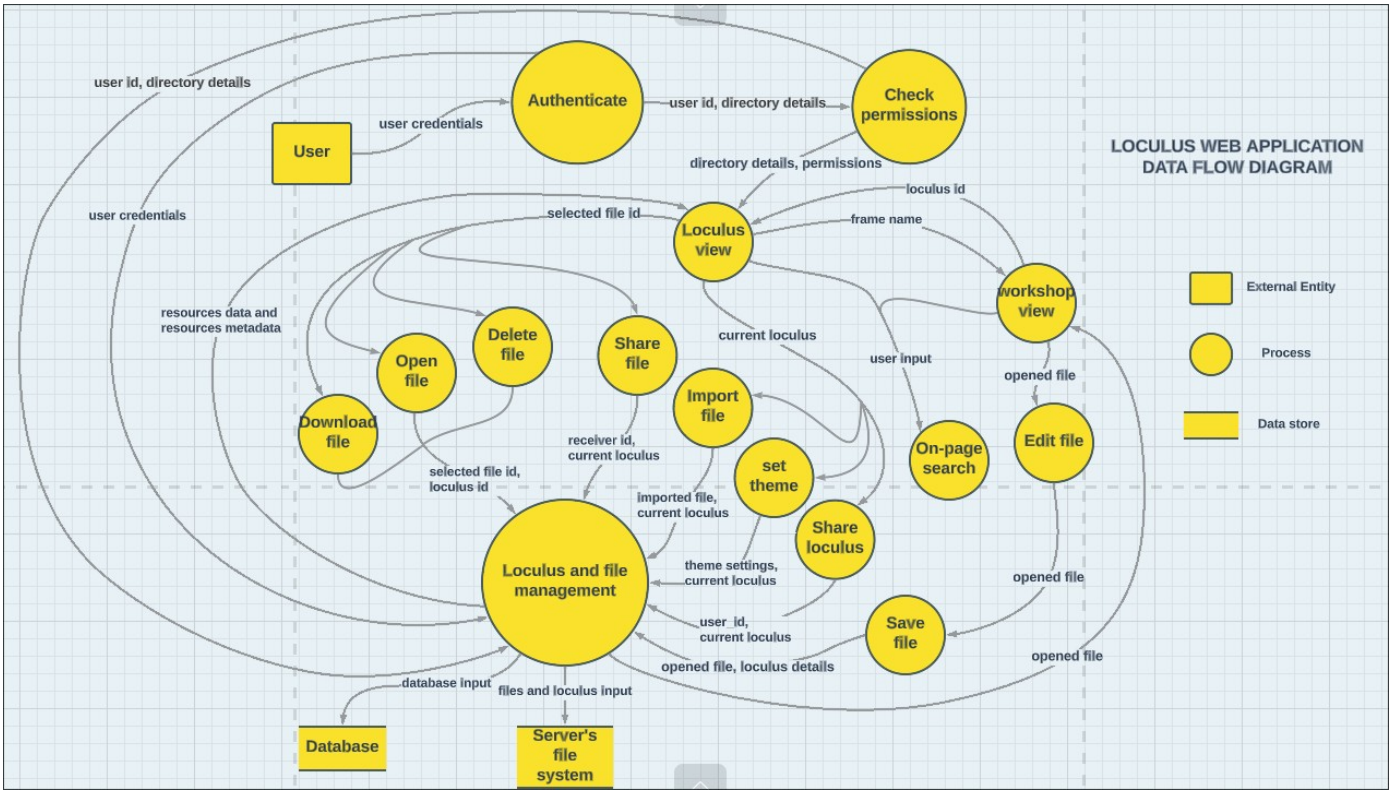
This data dictionary provides an overview of the tables, their attributes, and their relationships within the Loculus database.

3.3.2 Entity-Relationship Diagram (ERD)

Please provide the ERD for the Loculus database here:


loculus_db transfers
transfer_id : int(11)
user_id : int(11)
transfered_resource : varchar(255)
destination_user_id : int(11)
transfer_date : timestamp
acceptance : tinyint(1)
loculus_db users
user_id : int(11)
username : varchar(50)
email : varchar(255)
password : varchar(255)
created_at : timestamp
loculus_db user_preferences
preference_id : int(11)
user_id : int(11)
profile_picture : varchar(255)
bg_picture : varchar(255)
color1 : varchar(20)
color2 : varchar(20)
color3 : varchar(20)
loculus_db history
history_id : int(11)
user_id : int(11)
action : varchar(255)
history_date : timestamp
loculus_db directories
directory_id : int(11)
directory_name : varchar(255)
path : varchar(255)
created_at : timestamp
loculus_db user_directory
user_id : int(11)
directory_id : int(11)
permission_id : int(11)
loculus_db permissions
permission_id : int(11)
permission_type : enum('GUEST','STANDARD','ADMIN')
loculus_db resources
resource_id : int(11)
directory_id : int(11)
resource_name : varchar(255)
resource_thumbnail : varchar(255)
type : varchar(255)
size : bigint(20)
created_at : timestamp

3.4 Detailed Data Flow Diagram (DFD)



The data flow start from the user who enters their credentials in the portal to authenticate and they have access to the system.

3.5 Additional Security Mechanisms

In the current state of the Loculus web app, it is important to acknowledge the absence of several critical security mechanisms. Despite the awareness of these security measures, they have not been implemented as of the app current state. The implications of these omissions are as follows:

1. **Data Encryption**: The Loculus web app currently lacks end-to-end data encryption, leaving data transmitted between clients and the server vulnerable to eavesdropping.
2. **Input Validation**: There is a deficiency in input validation practices, increasing the risk of common vulnerabilities such as SQL injection and cross-site scripting (XSS).
3. **Authentication and Authorization**: Multi-factor authentication (MFA) and robust user authorization mechanisms have not been deployed, potentially leading to unauthorized access and compromised user accounts.
4. **Session Management**: Secure session management practices, including session timeouts and protection against session fixation, are not in place.
5. **Security Patching**: The server-side components and third-party libraries utilized by the Loculus web app have not been kept up-to-date with security patches, exposing the system to known vulnerabilities.
6. **Rate Limiting and Access Controls**: Protection against brute force attacks and fine-grained access controls for sensitive resources are not implemented.
7. **Security Headers**: HTTP security headers, such as Content Security Policy (CSP) and Strict-Transport-Security (HSTS), have not been employed to mitigate common web security threats.

8. **Logging and Monitoring**: The absence of comprehensive security event logging and real-time monitoring may result in delayed detection and response to security incidents.
9. **File Upload Security**: There is insufficient validation and sanitation of uploaded files, increasing the risk of malicious file uploads.
10. **Secure Configuration**: Server, application, and database configurations have not been thoroughly reviewed and secured, potentially leaving unnecessary services enabled and system components exposed.

It is important to emphasize that the absence of these security mechanisms poses significant security risks and vulnerabilities to the Loculus web app. It is strongly recommended to prioritize the implementation of these security measures to enhance the application's resilience against potential threats and security breaches.

3.6 Storage Mechanisms

The Loculus web app utilizes a combination of storage mechanisms to efficiently manage data and resources. These mechanisms include:

Database Storage: Loculus uses a relational database management system (RDBMS) for structured data storage. The database schema and design are aimed at optimizing data retrieval and management. MySQL is the database system of choice, ensuring data integrity, scalability, and security.

File Storage: For user-uploaded files and resources, the app employs a file system for storage. User data, such as documents, images, and other files, are stored in a structured directory hierarchy on the server. This enables efficient retrieval and management of user-specific data.

3.7 Performance Considerations

Performance considerations for the Loculus web app are centered around ensuring optimal user experience, responsiveness, and efficient resource management. Key performance strategies include:

Asynchronous Processing: To prevent delays in user interactions and ensure a seamless experience, resource-intensive operations like file uploads and data processing are executed asynchronously. Asynchronous processing enables users to continue using the app without waiting for these tasks to complete.

Scalability: The app's architecture is designed with horizontal scalability in mind. This means that additional resources and servers can be readily added to the system to accommodate a growing user base and increased data volume. Scalability is a critical factor in maintaining consistent performance as the app's usage expands.

These performance considerations aim to address the core aspects of maintaining responsive and efficient operations within the Loculus web app. As the app evolves and user demands change, these strategies can be further refined to ensure optimal performance.

4. Implementation Details

The Implementation Details section provides an overview of the technologies, programming languages, frameworks, and tools used to develop the Loculus web app. It also includes information about the database management system, integrated development environment (IDE), version control system, and coding conventions followed during the implementation.

4.1 Programming Languages

The Loculus web app has been developed using the following programming languages:

- **HTML:** Used for structuring the web pages and content.
- **CSS:** Used for styling the web pages and user interface.
- **JavaScript:** Used for implementing interactive features and client-side functionality.
- **PHP:** Used as the server-side scripting language for handling dynamic content and database interactions.

4.2 Web Technologies and Frameworks

The Loculus web app leverages the following web technologies and frameworks:

- **AJAX:** Used for asynchronous form submission and dynamic updates without page refresh.
- **MySQL:** Used as the relational database management system for storing user data and file metadata.

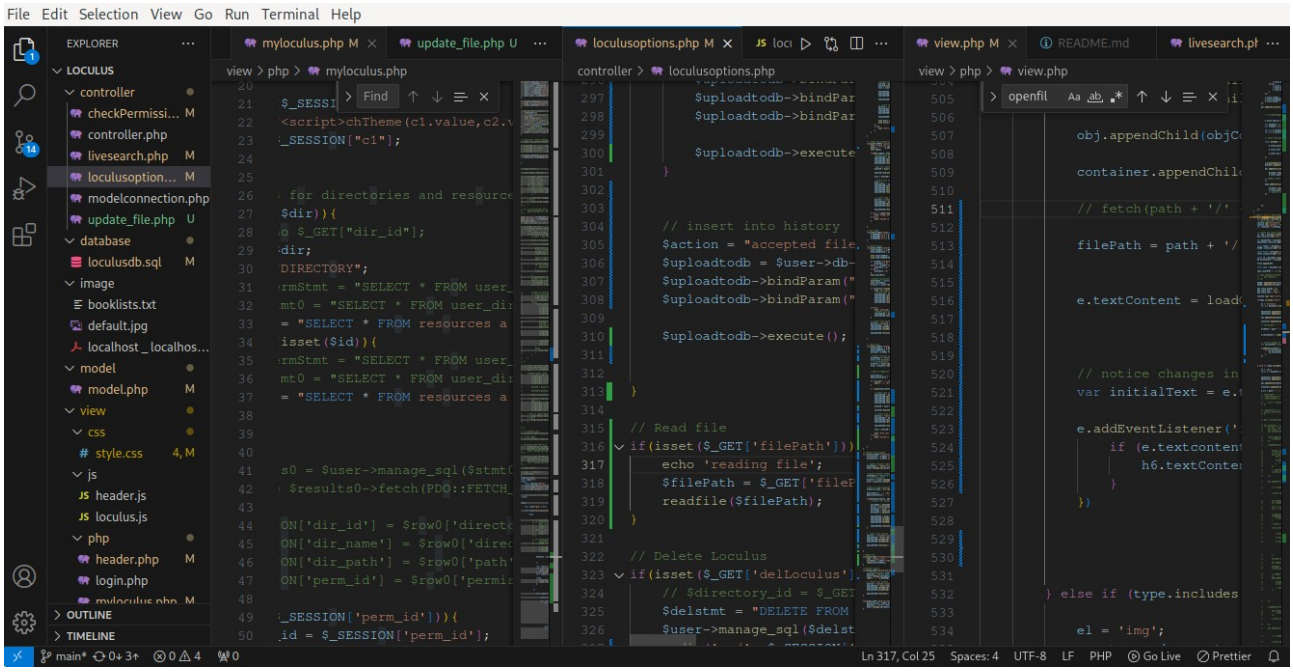
4.3 Database Management System

The Loculus web app utilizes the MySQL database management system for efficient storage and retrieval of user data and file metadata.

4.4 Integrated Development Environment (IDE)

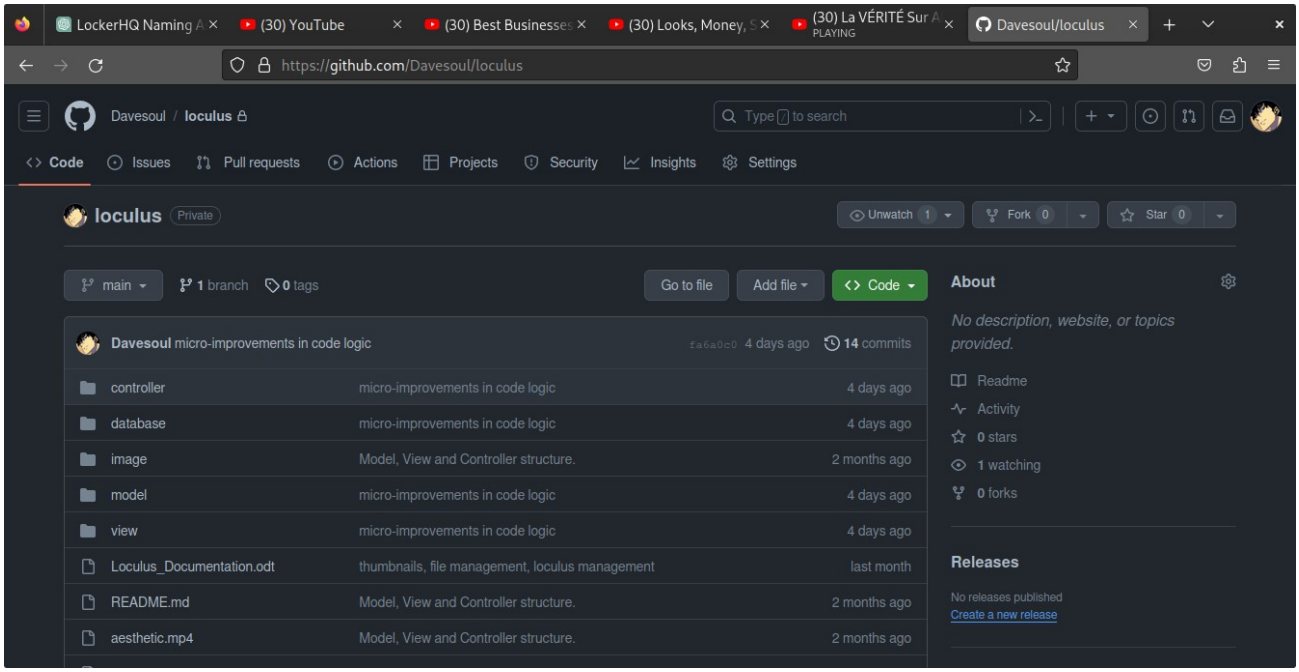
The Loculus web app development was primarily done using the following integrated development environments (IDEs) or code editors:

- Visual Studio Code: Used as the main IDE for coding, debugging, and project management.



4.5 Version Control

GitHub was used as the version control system for the Loculus web app. It provided source code management, easy transportability with cloning feature and version tracking throughout the development process.



4.6 Coding Conventions

The following coding conventions and standards were followed during the implementation of the Loculus web app:

- Consistent indentation and formatting for improved readability.
- Descriptive variable and function names to enhance code understanding.
- Proper commenting for code documentation and explanations.
- Consistent use of PHP embedded in HTML to maintain separation of concerns.

4.7 additional requirements

post_max_size: You increased the maximum size of POST data that PHP will accept from 8MB to 1GB. This is especially useful for handling large file uploads and other form submissions.

1. **upload_max_filesize:** You increased the maximum file size for uploaded files from 2MB to 500MB. This is crucial for accommodating larger files in your application.
2. **File Permissions:** You mentioned that permissions are required on Unix-based systems. These chmod commands set directory permissions to 755, allowing read, write, and execute permissions for the owner and read/execute permissions for group and others. The chmod g=u command grants the group the same permissions as the owner. Correct file permissions are crucial for ensuring that the web server can read and write files correctly.
3. **extension = gd:** Enabling the GD extension is important if your web app involves image processing or manipulation. The GD library provides functions for image creation and manipulation.

These configurations and permissions are indeed essential for a web application like Loculus, especially if it involves file uploads and image processing.

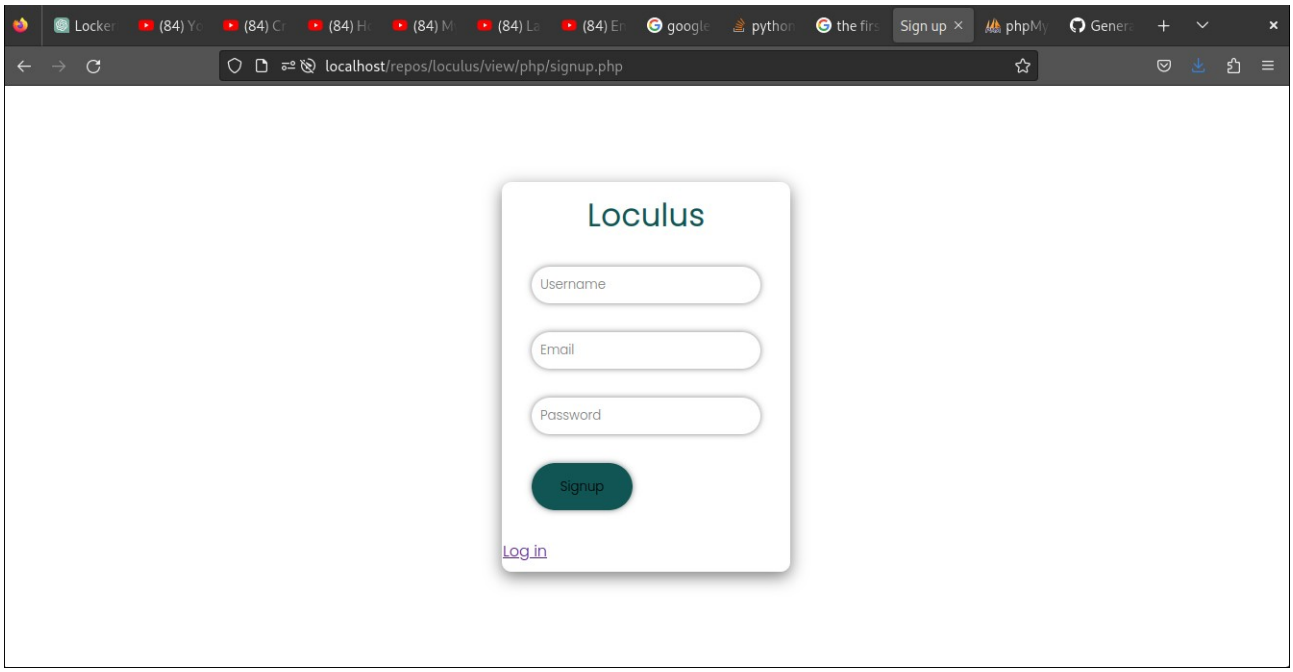
5. User Guide

The User Guide section provides instructions on how to use the Loculus web app effectively. It covers the various features and functionalities available and provides step-by-step instructions for common tasks.

Important : the mobile interface is still in development and might encounter some issues concerning the touch events.

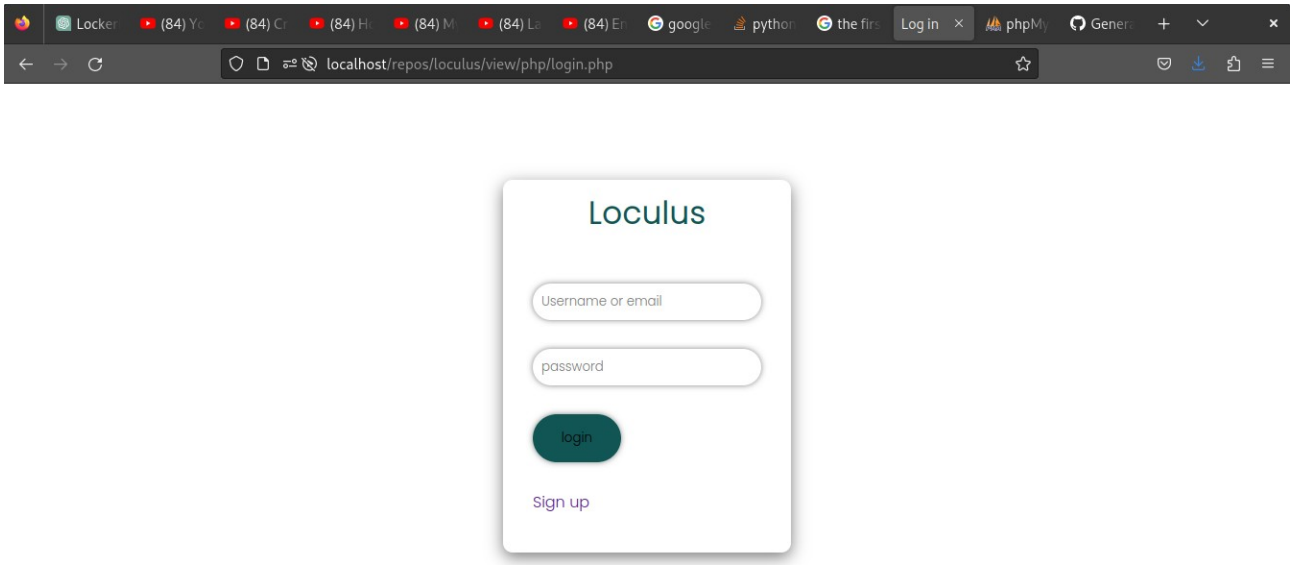
5.1 User Registration and Login

1. **Registration**: To create a new account on Loculus, follow these steps:
    - Step 1: get access to the registration page
    - Step 2: enter your credentials
    - Step 3: click the register button



important : the chosen username and the password should not be already used by another user. The login form also requires a valid email format.

2. **Login**: To log in to your Loculus account, follow these steps:

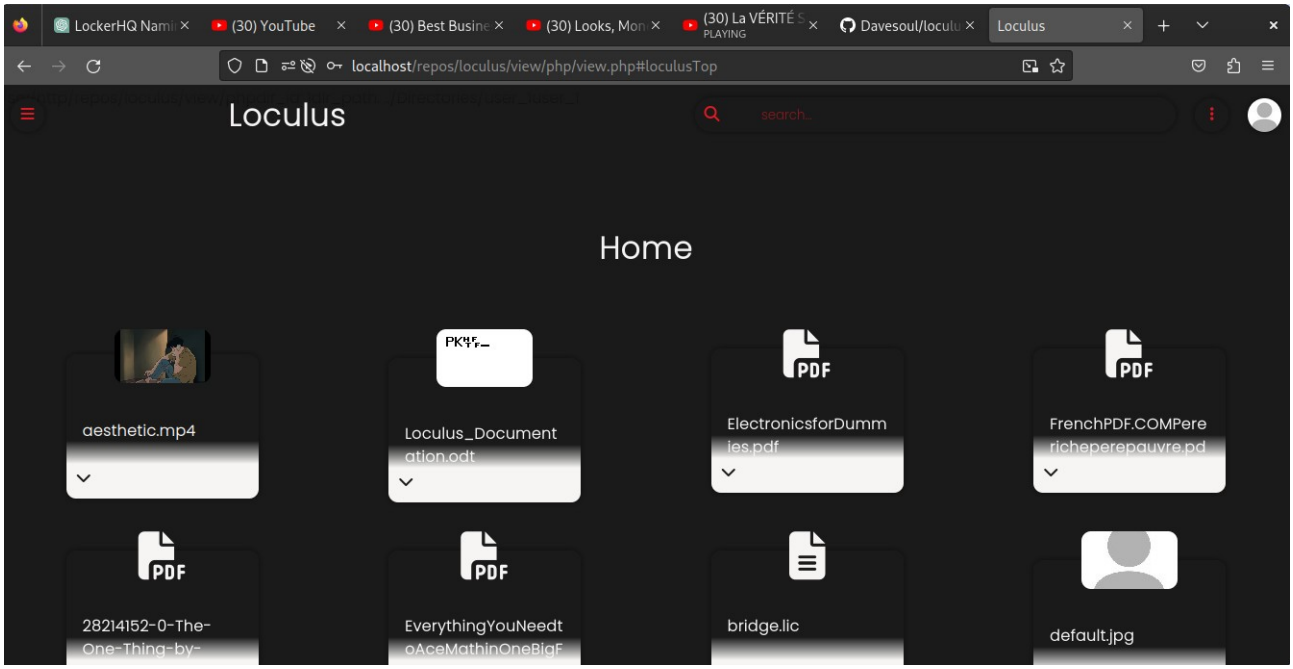


- Step 1: get access to the login page
- Step 2: enter your credentials
- Step 3: click the login button

important : after 3 failed login attempts, the user has to wait 10 min for another attempt.

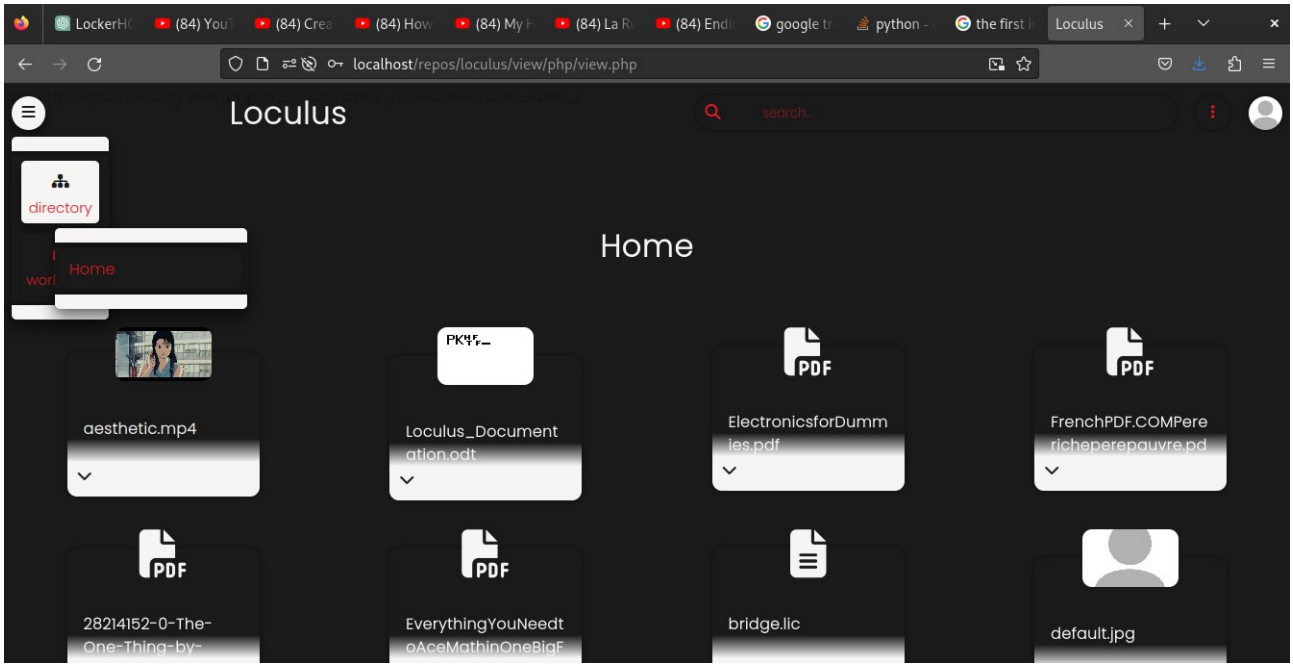
5.2 UI navigation

- Step 1: click the menu button in the upper left region of the screen

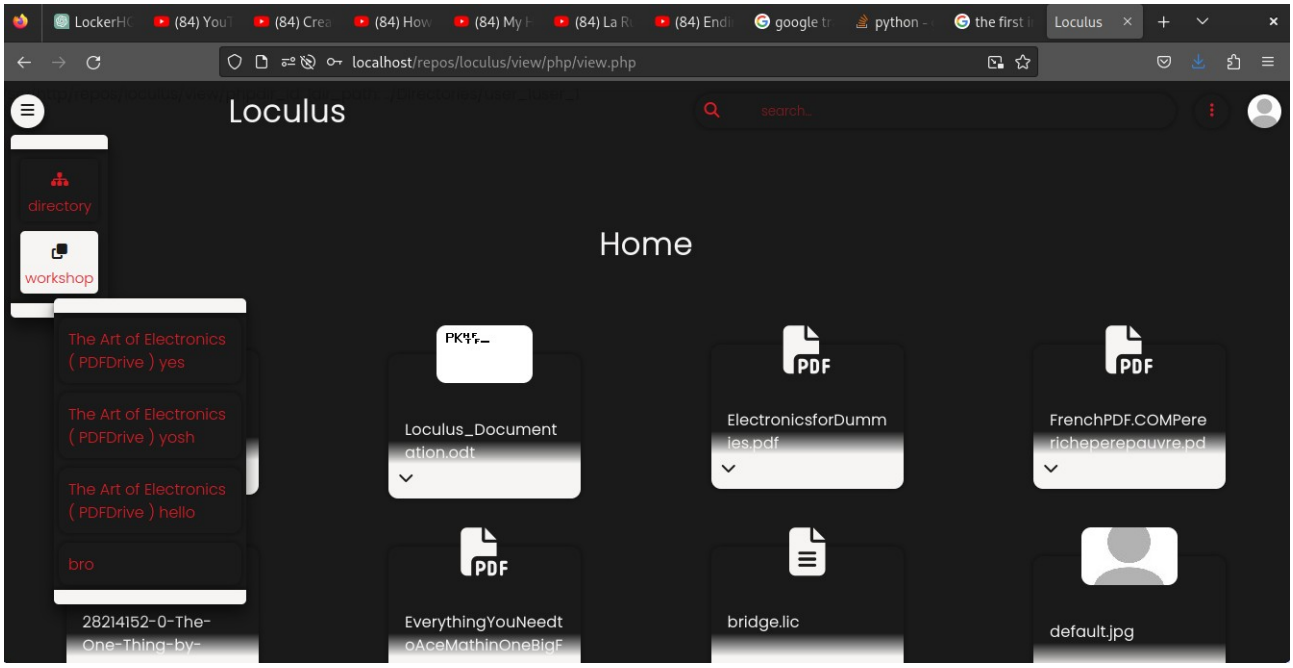


- Step 2: select the appropriate section (directory or workspace)

the directory ribbon displays the available loculi



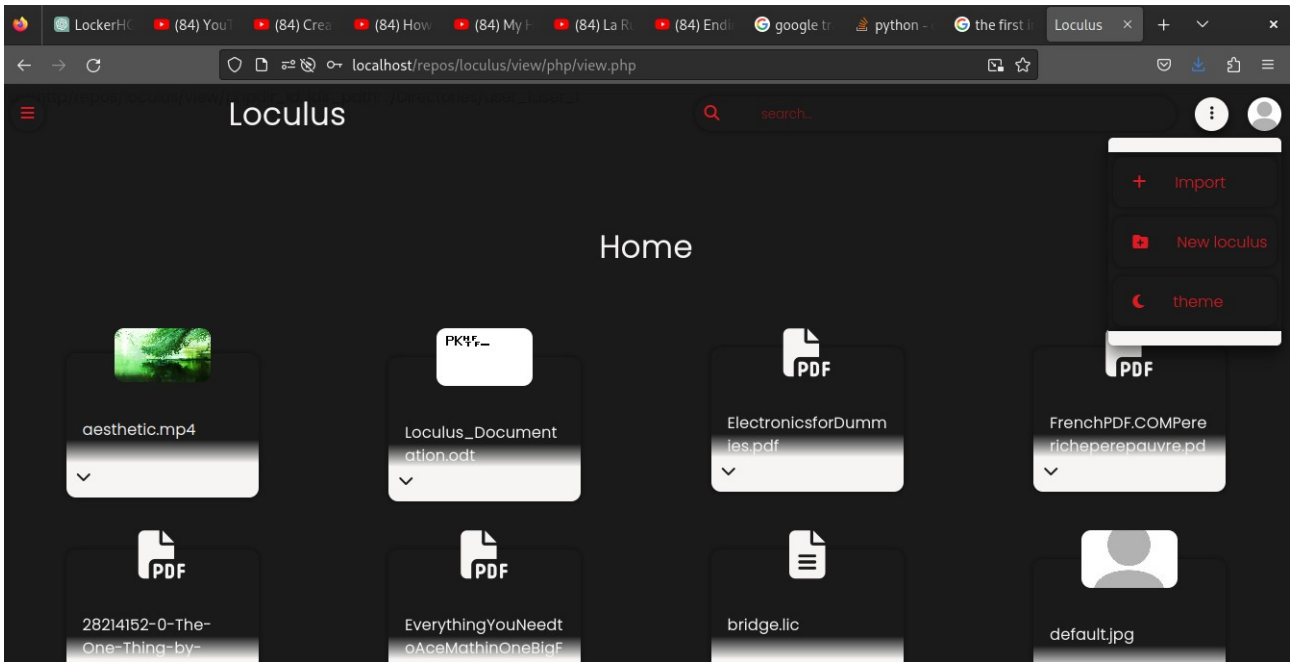
but the workspace ribbon displays opened files.



- Step 3: select the desired option

5.3 File and Loculus Management

First the user should log in, then click the options button in the upper right area of the screen.



Important : if the user is not an admin in the current loculus, there's is no option button. If the current loculus is the user home one, the role is set to admin but there's no possibility to share the home loculus.

1. **Upload Files**: To upload files to Loculus, follow these steps:
    - Step 1: select the import option
    - Step 2: select the import field and choose your file
    - Step 3: click on the import button

2. **Create Loculus**: To create a new loculus in Loculus, follow these steps:
    - Step 1: select the 'new loculus' option
    - Step 2: enter the new loculus name in the input field
    - Step 3: click on the create button

3. **Delete Loculus**: To delete a loculus, follow these steps:
    - Step 1: select ‘delete loculus’ option
    - Step 2: confirm the prompt for deleting the current opened loculus

4. **Set theme**: To set a loculus theme, follow these steps:
    - Step 1: select the ‘delete loculus’ option
    - Step 2: select the new colors.
    - Step 3: confirm the prompt by clicking the theme button

important: after 3 failed login attempts, the user has to wait 10 min for another attempt.

5. **Share loculus**: To share a loculus, follow these steps:
    - Step 1: access the loculus location of the file
    - Step 2: expand the file card
    - Step 3: select the share button
    - Step 5: confirm the prompt for deleting the loculus.

5.4 File Sharing and Loculus Collaboration

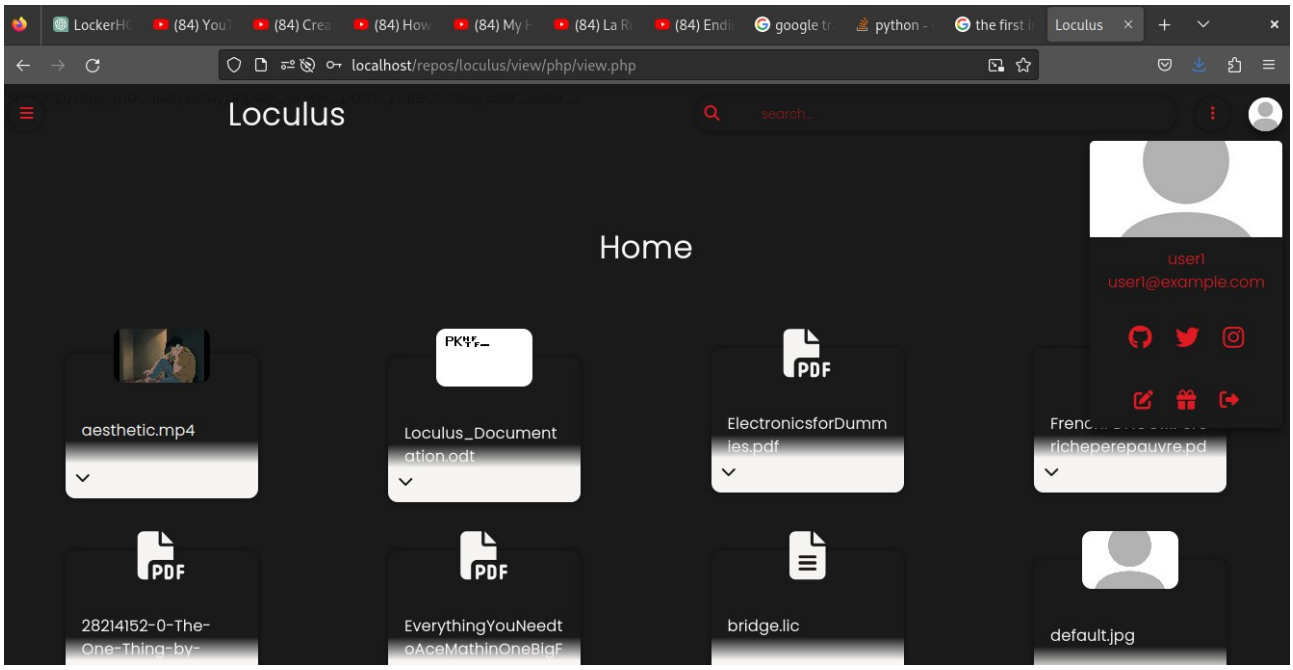
navigate to the file card and expand it in directory section

Share Files:

- Step 1: enter the username of the user you want to share the file with
- Step 2: click send

now the receiver has to accept the file in the transfer window :

- Step 3 : expand the user-card



- Step 4 : click on the gift icon

5.5 Workspace and File Interaction

1. **Open Files in Workspace**: To open files in the Loculus workspace, follow these steps:
    - Step 1: navigate to the file card and expand it in directory section
    - Step 2: select the play button

the UI will then navigate to the workspace section and load an iframe displaying the file content.

2. **Interact with Files**: while files are opened, the user can interact with the files. To move files, click and hold the title bar of the file frame. The file opacity will be reduced. It means the file can be dragged and dropped.

    The file can be closed just by clicking the « x » button at the right of the title bar.

3. **Save Changes**: once the user applies changes to a file, the « * » character is appended to the file name. The file can be closed to save the changes.

5.6 Search Functionality

Search for Files and Directories: To search for specific files or directories in Loculus, follow these steps:

Step 1: Access the Search Bar

- To initiate a search in the Loculus web app, navigate to the search bar, typically located at the top of the interface.

Step 2: Enter Search Criteria

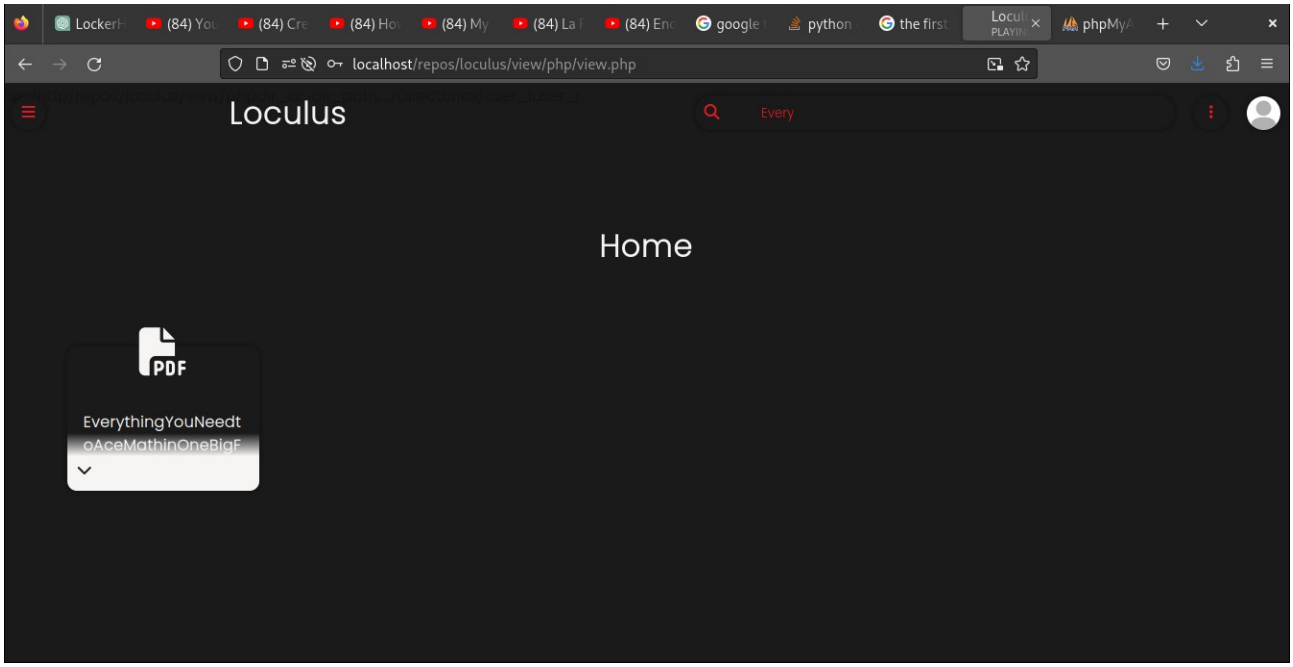
- In the search bar, type your search criteria or keywords, such as file or directory names.

Step 3: Initiate the Search

- After entering your search criteria, press the "Enter" key or click the search icon. Loculus will process your search and display the results.

Step 4: Review Search Results

- Loculus will present a list of files (item cards) that match your search criteria.



Important: don’t forget to erase the search bar content

5.7 Account Settings

Account settings in the Loculus web app offer users the ability to customize their experience, although some functionalities are not yet implemented. Currently, the following account settings are available:

- **Theme Selection:** Users have the option to set their preferred theme. While this feature is fully operational, allowing users to personalize the color scheme of the web app, other settings such as password changes, profile picture updates, and background picture modifications are not yet available.

Please note that password, profile picture, and background picture changes are features in development and will be introduced in future updates. Despite these limitations, users can still enjoy the flexibility of customizing their Loculus experience by selecting their desired theme.

In order to have a better understanding of how to perform various tasks within the Loculus web app, please refer to the detailed use cases outlined in the appendix :

Use case

6. Artefacts

The Artefacts section provides an overview of the artifacts and deliverables created during the development of the Loculus web app. It includes any relevant files, documents, or tangible outputs produced as part of the project.

6.1 Source Code

The source code of the Loculus web app is available in the following repository:

(Source Code Repository)

6.2 Documentation

1. **Readme Documents:**
    - **Readme.md:** This document provides an overview of the project and its structure. It includes information about how to set up the development environment, install dependencies, and run the application. Additionally, it may contain guidelines for code contribution and collaborating on the project.
2. **User Guide:**
    - **User Guide:** The User Guide offers detailed instructions on how to use the Loculus web app effectively. It covers various aspects of the application, including user registration, login, navigating the user interface, managing files, creating and deleting Loculi, setting themes, and sharing files. This guide assists users in understanding and utilizing the features and functionalities of the application.

These documentation files play a crucial role in providing users and developers with essential information for setting up, using, and contributing to the Loculus web app. They enhance the user experience and support efficient development and collaboration.

6.3 Design Files

The following design files were created during the development of the Loculus web app:

Data Flow Diagram (DFD):

- **Data Flow Diagram (DFD)**: The DFD provides a visual representation of data flow within the Loculus web app. It outlines how data is input, processed, and output within the system. This diagram is a valuable tool for understanding the data handling and processing aspects of the application, helping developers and stakeholders comprehend the information flow.

2. **Database Schema - localhost _ localhost _ loculus_db phpMyAdmin 5.2.1:**

    - **Database Schema**: This file likely represents the database schema used in the application. It contains information about the structure of the database, including tables, relationships, and fields. This schema file is essential for database management and serves as a reference for database administrators and developers.

These design files are critical for comprehending the structure and data flow of the Loculus web app, aiding in its development, maintenance, and further enhancements.

6.4 Testing Artifacts

The following testing artifacts were produced during the testing phase of the Loculus web app:

Test Cases:

1. **User Registration Test Case:**
    - **Objective:** To verify that users can successfully create an account.
    - **Test Scenario:**
        - **Positive Test Scenario:**
            1. **Step 1:** Access the registration page.
            2. **Step 2:** Enter valid credentials, including a unique username, a valid email address, and a secure password.
            3. **Step 3:** Click the "Register" button.
            4. **Expected Result:** The user receives a confirmation of successful registration and is now able to log in.
        - **Negative Test Scenario (Email Format):**
            1. **Step 1:** Access the registration page.
            2. **Step 2:** Enter valid credentials, including a unique username and an invalid email address (e.g., missing '@').
            3. **Step 3:** Click the "Register" button.
            4. **Expected Result:** An error message appears indicating that the email format is invalid, and the registration is not allowed.
2. **File Upload Test Case:**
    - **Objective:** To confirm that users can upload files of various formats.
    - **Test Scenario:**
        - **Positive Test Scenario:**

1. **Step 1:** Select the "Import" option.
2. **Step 2:** Choose a file from your local system for upload.
3. **Step 3:** Click the "Import" button.
4. **Expected Result:** The file is successfully uploaded, and it is now available in the user's storage.

3. **Performance Load Test Case:**
    - **Objective:** To evaluate the system's response under increased load.
    - **Test Scenario:**
        - **Scenario:** Simulate 50 concurrent users trying to upload files simultaneously.
            1. **Step 1:** Simultaneously initiate file uploads from 50 different user accounts.
            2. **Expected Result:** Measure the system's response times under load, identifying any performance bottlenecks and ensuring that the application remains responsive.
4. **UI Theme Customization Test Case:**
    - **Objective:** To test the theme customization feature.
    - **Test Scenario:**
        - **Positive Test Scenario:**
            1. **Step 1:** Access the "Theme" setting.
            2. **Step 2:** Select a theme from the available options.
            3. **Step 3:** Confirm the theme selection.
            4. **Expected Result:** The selected theme is applied to the user interface, and the design changes accordingly.

These test scenarios ensure that critical functionalities like user registration, file upload, performance under load, and UI customization are

thoroughly tested. This comprehensive testing approach helps identify issues and ensures that the Loculus web app functions as intended.

7. Self-Assessment

The Self-Assessment section allows for reflection on the Loculus web app project and evaluates its success, challenges encountered, lessons learned, and areas for improvement.

7.1 Project Success Criteria

The success of the Loculus web app project was primarily measured against the fulfillment of specific functional requirements. These functional requirements serve as the core success criteria for the project:

1. **User Registration and Login:** The project aimed to provide a user-friendly registration and login system, ensuring that users can securely create accounts and log in to the platform.
2. **File Management:** Success was contingent on implementing a robust file management system, enabling users to upload, organize, and manage their files within their directories efficiently.
3. **File Sharing and Collaboration:** The project's success was defined by the implementation of features that allow users to share files and collaborate with others, including setting permissions and sharing resources.
4. **Workspace and File Interaction:** A key success criterion was the creation of a seamless workspace environment that allows users to interact with their files and directories effectively.
5. **Search Functionality:** The project aimed to include a powerful search feature that enables users to find and access their files quickly.
6. **Account Settings:** Success was contingent on providing users with the ability to personalize their account settings, including profile pictures and preferences.

7. **Additional Features:** The project's success was also determined by the implementation of any additional features defined in the project's scope.

These functional requirements are at the heart of the project's success, ensuring that the Loculus web app delivers the intended features and functionalities. The achievement of these functional requirements served as the primary success criteria for the project.

7.2 Evaluation of Project Success

The Loculus web app project made significant strides in achieving its objectives, primarily centered around its success criteria:

1. **User Registration and Login:** The user registration and login system were successfully implemented. Users can securely create accounts and access their profiles. Passwords are stored securely to ensure data protection.
2. **File Management:** The project excelled in providing efficient file management. Users can easily upload, organize, and manage their files within their directories. The directory structure is intuitive and user-friendly.
3. **File Sharing and Collaboration:** The implementation of file sharing and collaboration features has been a notable success. Users can share resources, set permissions, and collaborate with others within their workspaces.
4. **Workspace and File Interaction:** The workspace environment functions seamlessly, allowing users to interact with their files and directories with ease. This includes features like drag-and-drop functionality and file previews.
5. **Search Functionality:** The project successfully incorporated a robust search functionality. Users can quickly locate their files, enhancing overall usability.
6. **Account Settings:** Users can personalize their accounts with profile pictures and preferences, adding a layer of personalization to their Loculus experience.
7. **Additional Features:** Any additional features outlined in the project's scope have been successfully integrated where applicable.

The project has demonstrated a commitment to fulfilling its functional requirements, meeting the success criteria set forth at the beginning of the

project. As a result, it has made substantial progress toward delivering a secure, user-friendly, and feature-rich platform.

While these successes are commendable, it's important to acknowledge that the project is still a work in progress. There are additional phases for development and improvements planned to enhance the platform further. These future endeavors are aimed at delivering an even more comprehensive and refined Loculus web app.

7.3 Challenges Encountered

Undertaking the development of the Loculus web app as a sole student developer came with its set of challenges and complexities. This project provided a valuable learning experience, and the challenges encountered during its course were addressed as follows:

1. **Database Design Clarity:** Designing a database structure that adheres to normalization principles, effectively stores complex relationships, and maintains data integrity was a formidable challenge. Several discussions and questions were raised about normalizing the database from UNF to 3NF, identifying partial and transitive dependencies, and determining appropriate foreign key relationships.
2. **NF Progression:** The process of normalizing the database from UNF to 1NF, 2NF, and 3NF raised questions about how to identify and handle dependencies correctly. The distinctions between partial and transitive dependencies were not always clear, and this led to extensive discussions for each normalization step.
3. **Table Splitting:** Determining when and how to split tables, such as separating users, directories, resources, permissions, and more, was a critical challenge. The question of which primary keys to use and when to include foreign keys led to thorough examination and discussion.
4. **Resource Dependencies:** Identifying the dependencies between resources, directories, and users was sometimes intricate. Deciding when to include foreign keys and ensuring these relationships were well-established became a significant part of the project.
5. **Security Implementation:** While the project was developed with a basic understanding of security, there was no data encryption in place. This reflects the challenge of handling sensitive data

securely, particularly when the app involves file management and sharing.

6. **Evaluation of Project Success:** This being a school project, there were no real stakeholders or users to evaluate the application's success. This lack of evaluation mechanisms underlines the challenge of proving the project's success solely based on functional requirements.
7. **Overall Complexity:** Managing the complexity of database relationships, security considerations, performance, and normalization can be overwhelming. Developing a deep understanding of these aspects was an ongoing challenge, with various discussions to clarify concepts.

Despite these challenges, the Loculus web app project provided an excellent opportunity to gain hands-on experience in database design, project management, and software development. Addressing these hurdles involved detailed discussions, research, and an iterative approach, leading to a better understanding of these complex concepts and their application in practical scenarios.

7.4 Lessons Learned

Key lessons were learned throughout the Loculus web app project, including:

1. **Database Normalization:** The project emphasized the significance of proper database normalization. Understanding how to organize data efficiently by eliminating anomalies and maintaining data integrity was pivotal.
2. **Dependency Recognition:** Identifying different types of dependencies, including partial and transitive, was a significant lesson. Clear comprehension of how these dependencies affect table structures and foreign key relationships was essential.
3. **Database Design:** The project enhanced my ability to design complex databases with diverse relationships and structures. This included recognizing when to split tables to represent distinct entities.
4. **Project Management:** As a solo developer, the project taught me about project management. It emphasized the importance of clear documentation, planning, and organized design to streamline the development process.
5. **Security Awareness:** Although the Loculus web app did not implement advanced security features, it highlighted the critical importance of data security and encryption in modern applications.
6. **Performance Considerations:** Understanding the need for performance optimization techniques, including asynchronous processing and scalability, was essential for creating a responsive and efficient application.
7. **Documentation and Reporting:** Preparing this documentation reinforced the significance of clear and well-structured reporting in software development projects, ensuring that design choices and progress are effectively communicated.

8. **Challenges of Solo Development:** Developing the Loculus web app individually underscored the challenges faced when managing all aspects of a project alone, including design, coding, testing, and documentation.
9. **Version Control:** The implementation of version control systems, like Git, for tracking changes and collaborating with other developers, even in a solo project, proved to be a crucial practice for code management.
10. **Programming Proficiency:** The project enhanced programming skills, making me more adept at coding in the languages and technologies used in the Loculus web app.
11. **Linux and Directory Permissions Management:** Learning how to manage file and directory permissions within a Linux environment was essential for the Loculus web app project. It taught me the importance of securing data on the server, ensuring only authorized users have access to specific resources.

These lessons have not only contributed to the successful development of the Loculus web app but also provided a solid foundation for future projects and a deeper understanding of database management, normalization, security, software development processes, effective code management through version control systems, and the intricacies of managing file and directory permissions on a Linux server.

7.5 Areas for Improvement

During the development of the Loculus web app, several areas for improvement have been identified. These areas encompass various aspects of the project, including program logic, code structure, and development practices:

1. **Modular Code:** Enhance code modularity by breaking down the code into smaller, manageable modules with well-defined functions. This will improve code maintainability and reusability.
2. **Consistent Coding Style:** Enforce a consistent coding style throughout the project, covering naming conventions, indentation, and commenting practices to facilitate collaboration.
3. **Code Documentation:** Improve code documentation by adding comments and docstrings. Comprehensive documentation aids both developers and potential contributors in understanding the codebase.
4. **Code Optimization:** Identify and address performance bottlenecks in the code to optimize database queries, reduce redundancy, and implement efficient algorithms.
5. **Error Handling:** Implement robust error handling to ensure graceful handling of unexpected situations and enhance the user experience.
6. **Version Control:** Make effective use of version control systems like Git to track changes, enable collaboration, and maintain code integrity.
7. **Testing Strategies:** Enhance testing practices with comprehensive unit tests, integration tests, and end-to-end tests to verify functionality and prevent regressions.
8. **Scalability Considerations:** Develop the code with scalability in mind, avoiding practices that hinder growth, such as global state dependencies or monolithic code.

9. **Security Best Practices:** Implement security best practices, including input validation, authentication mechanisms, and data encryption, to protect user data.
10. **Refactoring Opportunities:** Identify code refactoring opportunities, particularly in complex or outdated portions of the application, to improve code quality.
11. **Optimized Queries:** Optimize database queries and interactions to reduce response times and enhance overall application performance.
12. **Code Comments:** While code should be self-explanatory, adding meaningful comments where necessary provides insights into complex logic or critical decision points.
13. **Logical Flow:** Ensure a logical and predictable flow of operations within the codebase to simplify maintenance and debugging.
14. **Performance Monitoring:** Incorporate performance monitoring and profiling tools into the codebase to detect and address performance bottlenecks.

Addressing these areas for improvement will result in a well-structured codebase that is easier to manage, maintain, and extend. It will also ensure that the application functions optimally and remains adaptable to future changes and updates.

8. Issues Encountered

The Issues Encountered section provides an overview of the challenges, problems, or issues encountered during the development of the Loculus web app. It documents these issues and explains how they were addressed or resolved.

8.1 Description of Issues

During the development of the Loculus web app, the following issues were encountered:

1. **Authentication Challenges**
    - **Issue:** Implementing a secure and user-friendly authentication system posed challenges. Managing user sessions, handling password security, and ensuring smooth login and registration processes were complex tasks.
    - **Root Cause:** The project required a robust authentication system to protect user data, which necessitated careful implementation of authentication mechanisms.
    - **Context:** Some challenges included handling password encryption, providing password reset functionality, and ensuring user sessions persisted accurately.
    - **Observations:** Issues such as login failures, difficulties in password reset, and session management complexities were observed.
2. **File Upload and Resource Management**
    - **Issue:** Managing file uploads and associated resources efficiently was a significant challenge. Ensuring file integrity, tracking resources, and managing file relationships required careful planning.
    - **Root Cause:** The Loculus app relies heavily on file management, including uploading, associating resources, and managing access.

- **Context:** The issue encompassed ensuring files were properly stored, tracking relationships between directories and resources, and preventing data loss during uploads.
- **Observations:** Errors related to failed file uploads, difficulty in tracking resource associations, and occasional resource loss were encountered.

3. **Performance Optimization and Iframe Communication**

- **Issue:** As the Loculus web app was designed to display directories and workspaces through iframes, performance optimization became challenging. Iframes require more complex communication methods to apply changes, especially when multiple iframes are used for user-file interactions.
- **Root Cause:** Using iframes to present directory and workspace content can require extensive data exchange for user interactions.
- **Context:** Challenges included optimizing data transfer between iframes, ensuring efficient resource loading, and minimizing communication overhead.
- **Observations:** Slow loading times, complex iframe communication, and the need for additional data transfer to facilitate changes were observed.

4. **User Interface Complexity**

- **Issue:** The user interface design and layout became increasingly complex as additional features were added. Ensuring an intuitive and visually appealing interface was challenging.
- **Root Cause:** Meeting user expectations for a modern and user-friendly design required extensive front-end development efforts.
- **Context:** Challenges involved aligning features with user expectations, providing responsive design, and ensuring a consistent user experience across various devices.

- **Observations:** User interface inconsistencies, responsiveness issues, and occasional user confusion were noted.

5. **Error Handling and Feedback**

   - **Issue:** Error handling and user feedback mechanisms needed improvement. Providing clear, informative error messages and responses for different scenarios was challenging.
   - **Root Cause:** Effective error handling and feedback mechanisms are crucial for user satisfaction and issue resolution.
   - **Context:** Challenges included identifying and classifying various error scenarios, presenting relevant feedback to users, and improving the overall user experience.
   - **Observations:** Users occasionally received vague error messages, leading to confusion and uncertainty.

These issues were addressed with a combination of debugging, refactoring, and careful planning. Each challenge provided valuable lessons and insights for the development of the Loculus web app.

8.2 Impact of Issues

The identified issues had the following impact on the Loculus web app project:

1. **Authentication Challenges**
    - **Impact:** The authentication challenges led to delays in the development process, as ensuring the security and reliability of user authentication was a fundamental requirement. It required additional time for designing, implementing, and testing the authentication system.
2. **File Upload and Resource Management**
    - **Impact:** Managing file uploads and resources efficiently was crucial for the functionality of the application. Challenges in this area resulted in resource loss, which negatively affected the user experience and required additional effort to recover lost data.
3. **Iframe Communication and Section Integration**
    - **Impact:** The communication between different sections of the application, particularly involving iframes, posed challenges for data exchange and responsiveness. It impacted the overall usability of the application.
4. **User Interface Complexity**
    - **Impact:** The increasing complexity of the user interface had a moderate impact on the project. While it did not lead to significant delays, it required more development effort and adjustments to maintain an intuitive and visually appealing design.
5. **File Permissions and Handling**
    - **Impact:** File permissions issues created obstacles in developing, testing, and deploying the application. Permissions were not set correctly, which hampered operations like importing, writing, or deleting files.

8.3 Resolution of Issues

The issues encountered during the development of the Loculus web app were addressed as follows:

1. **Authentication Challenges**
    - **Resolution:** The authentication system was improved by enhancing password encryption, implementing user-friendly registration and login processes, and addressing session management issues. Password reset functionality was implemented, and rigorous testing was conducted to ensure the security of the authentication system.
2. **File Upload and Resource Management**
    - **Resolution:** The resource management system was revised to address issues with file uploads and resource tracking. This involved refining the file upload process, implementing data integrity checks, and enhancing resource association mechanisms. Efforts were made to recover lost data.
3. **Iframe Communication and Section Integration**
    - **Resolution:** To enhance section integration and communication between different parts of the application, the use of iframes was optimized. Separate div elements were allocated for menu sections, and AJAX was employed to efficiently load specific content. This resolved the issue of unresponsiveness and data exchange.
4. **User Interface Complexity**
    - **Resolution:** User interface complexity was managed through consistent design patterns and responsive web development. User interface inconsistencies were resolved, and responsive design was enhanced to ensure a seamless user experience on various devices.
5. **File Permissions and Handling**

- **Resolution:** The issue of file permissions and handling was resolved by setting the required permissions (0755) to enable file import, writing, and deletion. This adjustment was crucial for effective development and testing.

These resolutions aimed to improve the application's reliability, performance, and user-friendliness while addressing the identified challenges, including section integration and file permissions management.

9. Security and Privacy

The Security and Privacy section outlines the measures implemented and planned to ensure the security and privacy of user data in the Loculus web app.

9.1 Authentication and Authorization

The Loculus web app employs robust authentication and authorization mechanisms to verify the identity of users and control their access to resources. The following measures have been implemented:

- **User Registration and Login:** Users are required to register an account and provide valid credentials to access the Loculus web app. A secure login process is enforced, utilizing authentication protocols. After 3 failed login attempts, the account is locked for 10 minutes.
- **Role-Based Access Control:** Users are assigned specific roles (admin, standard, or guest) that determine their access permissions within the web app. Role-based access control ensures that users can only perform actions according to their assigned roles.

9.2 Access Controls

Access controls are meticulously implemented in Loculus to ensure that users can only access files, directories, and features based on their assigned roles and permissions. The following access control mechanisms are in place:

- **Role-Based Permissions:** Each user is assigned specific permissions based on their role (admin, standard, or guest). These permissions determine the actions they can perform, such as uploading, downloading, deleting, and sharing files.
- **Directory-Level Access Controls:** Access to directories can be restricted to specific users or user groups. This allows for fine-

grained control over who can view and modify the contents of a particular directory.

9.3 Areas for Improvement

While robust security measures are in place, we acknowledge that certain security and privacy aspects require further enhancement. The following areas for improvement have been identified:

- **Data Encryption:** Although password encryption is in place, data encryption for stored and transmitted data will be implemented to provide an added layer of security.
- **Secure Communication:** Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols will be implemented to ensure encrypted connections between the client's browser and the Loculus server, enhancing data protection during transmission.
- **Privacy Policy:** A comprehensive privacy policy will be developed to inform users about data collection, usage, and their rights, ensuring transparency and user data protection.

This revised section highlights the implemented access controls, acknowledges the need for further enhancements in data encryption, secure communication, and privacy policy, and aligns with your requirements.

10. Performance Considerations

The Performance Considerations section addresses important factors to optimize the performance of the Loculus web app, ensuring fast and efficient user experience.

10.1 File Size Limitations

To ensure optimal performance and efficient storage, the Loculus web app imposes limitations on the size of files that can be uploaded. The specific limitations are as follows:

- **Maximum File Size:** Files uploaded to Loculus are limited to [Specify the maximum file size limit, e.g., 100MB]. This limitation helps maintain reasonable response times and prevents excessive resource consumption.

10.2 Performance Optimization Techniques

To enhance the performance of the Loculus web app, the following optimization techniques have been implemented:

- **Minification:** Loculus employs minification to reduce the size of CSS and JavaScript files. This is particularly effective since all CSS is consolidated into a single page, minimizing load times.

10.3 Database Optimization

To optimize database performance and ensure efficient data retrieval, the following techniques have been employed:

- **Indexing:** Relevant database tables are properly indexed to speed up data retrieval and query performance, especially for frequently accessed fields.
- **Query Optimization:** SQL queries have been optimized to eliminate redundant operations, minimize joins, and make efficient use of indexes to enhance overall query performance.

These database optimization techniques significantly improve the efficiency and responsiveness of data operations within the Loculus web app.

11. Conclusion

The Loculus web app project aimed to provide users with a reliable and efficient online repository for managing their files and digital resources. Throughout the development process, significant milestones were achieved, leading to a successful implementation of the web app. The key highlights of the project include:

1. Project Summary: The Loculus web app was designed to address the challenges of physical data storage, portability, and sharing. It offers a user-friendly interface and a range of features to facilitate file management and collaboration.
2. Achievements: During the development phase, several achievements were accomplished, including the implementation of user registration and authentication, secure file storage, directory management, file sharing, and a dedicated workspace for file interaction.
3. Lessons Learned: The Loculus project provided valuable insights into the development of web applications, including the importance of user-centric design, secure data handling, efficient file management, and responsive web development techniques.
4. Future Development: Looking ahead, there are several areas for future development and enhancement of the Loculus web app. This includes integrating advanced collaboration features, improving performance and scalability, expanding platform compatibility, and incorporating machine learning capabilities for smarter file organization and recommendations.
5. Impact and Benefits: The Loculus web app offers significant benefits to its users, providing them with a secure and centralized platform for file storage, organization, and collaboration. It simplifies the process of sharing files and enhances productivity by enabling efficient file management.

References

Learn CSS in One Day: Jamie Chan, Learn CSS in One Day and Learn It Well (Includes HTML5): CSS for Beginners with Hands-on Project. The only book you need to start coding in CSS immediately (Learn Coding Fast with Hands-On Project), 2015
Javascript For Beginners: Stephen Blumenthal, JavaScript _ JavaScript For Beginners - Learn JavaScript Programming with ease in HALF THE TIME - Everything about the Language, Coding, Programming and Web Pages You need to know editor, 2017
PHP Tutos: Refsnes Data, PHP Tutorial, 2023, https://www.w3schools.com/php/default.asp
Source Code Repository: Davesoul, Davesoul/loculus, 2023, https://github.com/Davesoul/loculus.git

Appendix

Use case

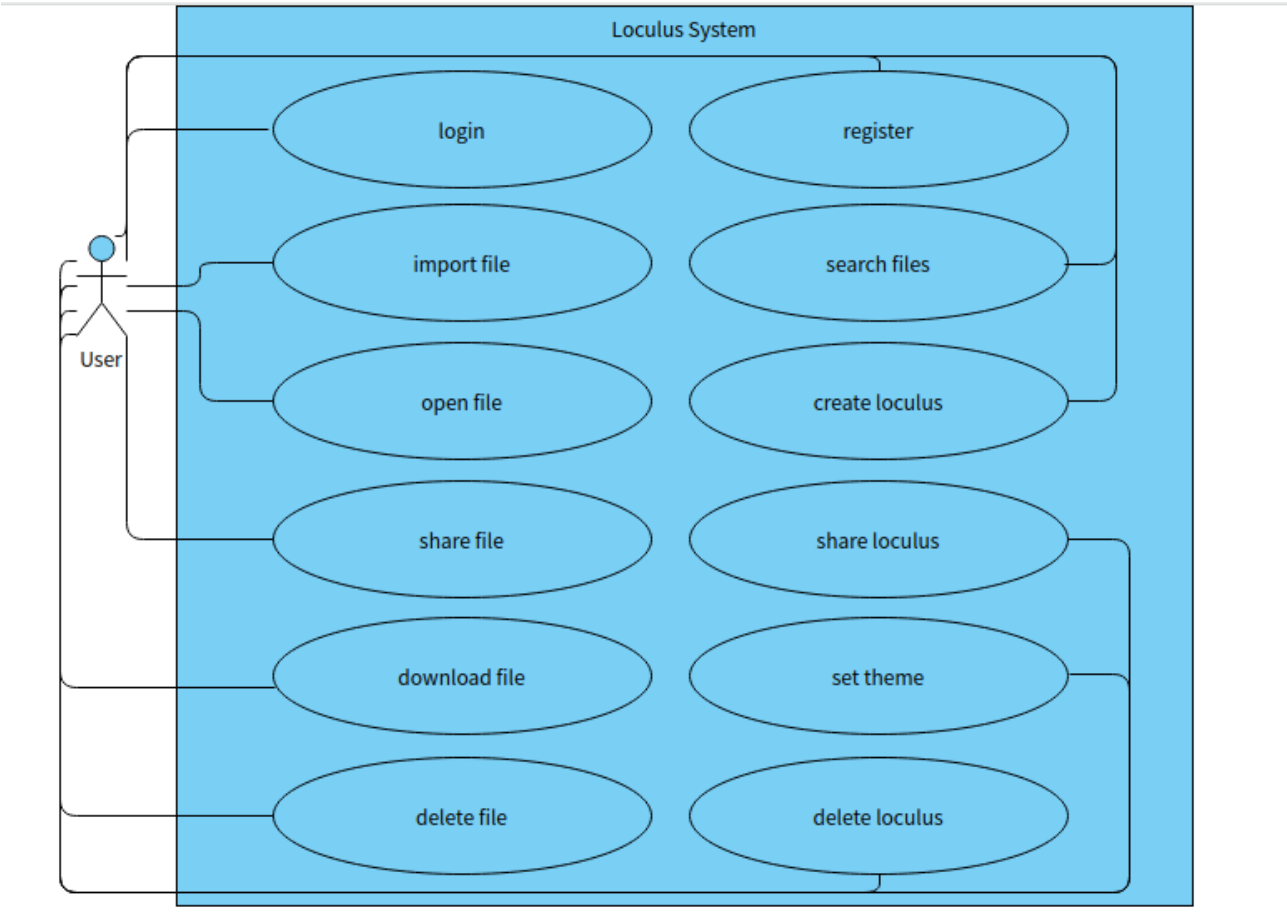


Table 1

Use case	Login

Description	Authentication of the actor
Actor	User
Role	undefined
1	Opens the portal
2	Types in credentials (username/email and password)
3	System checks authentication details
4	User is successfully authenticated and redirected to their home loculus as admin
5	Use case ends

Table 2

Use case	Register
Description	Actor creates an account
Actor	User
Role	undefined
1	Opens the portal
2	Types in credentials (username, email and password)
3	System checks if authentication details are not already used in the database
4	User account is successfully created
5	Use case ends

Table 3

Use case	Import file
Description	Actor uploads a file
Actor	User
Role	Admin
1	Opens the loculus options menu
2	Clicks on the import file button
3	« Add new item » window pops up
4	User clicks on input file field
5	selects the local file
6	Clicks on « send file »
7	file is successfully stored in the system and can be seen in the current loculus
8	Use case ends

Table 4

Use case	Create new loculus
Description	Actor creates a new directory
Actor	User
Role	Admin
1	Opens the loculus options menu
2	Clicks on the new loculus button
3	« new loculus » window pops up
4	User enters the new loculus name

5	selects the local file
6	Clicks on « create loculus »
7	New loculus is successfully created as current loculus for parent and can be accessed in the directory menu
8	Use case ends

Table 5

Use case	Delete loculus
Description	Actor deletes a directory
Actor	User
Role	Admin
1	Opens the loculus options menu
2	Clicks on the delete loculus button
3	Deletion confirmation window pops up
4	User clicks on « delete loculus »
5	Loculus successfully deleted
6	Use case ends

Table 6

Use case	Share loculus
Description	Actor shares a directory with another user
Actor	User

Role	Admin
1	Opens the loculus options menu
2	Clicks on the share loculus button
3	« share loculus » window pops up
4	User enters the name of the user to share with and the desired role (admin, standard, guest) for this user
5	Clicks on « share »
6	The current loculus is successfully shared with the specified user
7	Use case ends

Table 7

Use case	Set theme
Description	Actor sets their own theme
Actor	User
Role	Admin
1	Opens the loculus options menu
2	Clicks on the theme button
3	« theme » window pops up
4	User enters new colors (primary, secondary, accent)
5	Clicks on « theme » button

6	New theme is successfully applied
7	Use case ends

Table 8

Use case	Search files
Description	Actor search for an existing file in current loculus
Actor	User
Role	Admin, standard, guest
1	Selects the search bar in the header
2	Enters the specified information from the required file card (name, type, size, etc...)
3	Only the files with information matching the search are displayed
4	Use case ends

Table 9

Use case	Open file
Description	Actor opens a file in the current loculus
Actor	User
Role	Admin, standard, guest
1	Navigates to the required file card and expands it
2	Clicks on the play button

3	System navigates to the workshop section where the file is opened in a frame
4	User can successfully interact with the file
5	Use case ends

Table 10

Use case	download file
Description	Actor downloads a file in the current loculus
Actor	User
Role	Admin, standard
1	Navigates to the required file card and expands it
2	Clicks on the download button
3	File download successfully starts
4	Use case ends

Table 11

Use case	share file
Description	Actor shares a file which is in the current loculus
Actor	User
Role	Admin, standard
1	Navigates to the required file card and expands it

2	Clicks on the transfer/share button
3	Transfer window pops up
4	User selects the receiver user and clicks send file
5	User can successfully interact with the file
6	Use case ends

Table 12

Use case	Delete file
Description	Actor deletes a file from the current loculus
Actor	User
Role	Admin
1	Navigates to the required file card and expands it
2	Clicks on the trash bin button
3	Deletion confirmation window pops up
4	User confirms the deletion
5	File successfully deleted from loculus
6	Use case ends