**PERSONAL ID AND DOCUMENT KEEPER SYSTEM**

**SYSTEM DOCUMENTATION**

**SUBMITTED BY,**

**JOHNSON KAMAU KABIRU,**

**DCS-01-8245/2023**

**A SYSTEM DOCUMENTATION SUBMITTED IN PARTIAL FULFILMENT FOR THE AWARD OF DIPLOMA IN INFORMATION TECHNOLOGY BY ZETECH UNIVERSITY**

**NOV, 2024**

**DECLARATION**

I, Johnson Kamau , hereby declare that this system documentation titled "Personal ID and Document Keeper System" is original work done by me and no part of it has been presented for any academic award in any institution. I confirm that all sources of information have been acknowledged.

**ACKNOWLEDGEMENT**

I hereby take this opportunity to express my profound gratitude to my supervisor, lecturers, and all individuals who guided and supported me in the development of this project. I also give special thanks to my family and friends for their encouragement and patience during this time. Finally, I recognize the rich resources and tools that Zetech University offered that have facilitated the realization of this project.

**TABLE OF CONTENTS**

**System Overview**

1.1 Statement of Problem

1.2 Study Justification

1.3 System Objectives

1.4 Functional Requirements

**Front End (User Interfaces)**

2.1 Introduction to User Interfaces

2.2 Individual User Interfaces

2.2.1 Sign Up Form

2.2.2 Log in Page

2.2.3 Account / Profile Pages

2.2.4 Document Interactions

2.2.5 Document Preview

2.2.6 Documents Download

2.2.7 End Session

**Back End**

3.1 Introduction

3.2 Database Implementation

3.2.1 Setting up SQL Database

3.2.2 Creating and Testing the Tables

3.3 Logic

3.3.1 Login Functionality

3.3.2 User Management

3.3.3 Document Management

**System Quality Tests**

4.1 Introduction

4.2 User Entry and Validation Test (Feature Test)

4.3 Document Upload Validation Test (Feature Test)

4.4 Search and Retrieval Test (Feature Test)

4.5 Database Integration Test: Save and Retrieve Records

4.6 Delete Functionality Test (Feature and DB Test)

4.7 Summary of Tests

**Conclusion and Recommendations**

**Appendices**

* Appendix I: Gantt Chart
* Appendix II: Resource Requirement and Economic Feasibility Assessment
* Appendix A: Source Code
* Appendix B: User Manual
* Appendix C: Test Results

**LIST OF FIGURES**

**Figure 2.2.1.1**: Sign Up Form

**Figure 2.2.1.2**: XML Interface generator code for individual sign up form

**Figure 2.2.1.3**: XML Interface generator code for individual sign up.css form

**Figure 2.2.2.1**: Login Page

**Figure 2.2.2.2**: XML Interface generator code for individual login page

**Figure 2.2.2.3**: XML Interface generator code for individual login.css page

**Figure 2.2.3.1**: Profile Page

**Figure 2.2.3.2**: XML Interface generator code for profile accounts

**Figure 2.2.3.3**: XML Interface generator code for profile accounts

**Figure 2.2.4.1**: Document Interactions Page

**Figure 2.2.4.2**: XML Interface generator code for ID and document uploads

**Figure 2.2.4.3**: XML Interface generator code for upload.css

**Figure 2.2.5.1**: Documents Section

**Figure 2.2.6.1**: Download Section

**Figure 2.2.6.2**: XML Interface generator code for ID and document downloads

**Figure 2.2.6.3**: XML Interface generator code for download.css

**Figure 2.2.7.1**: System Logout

**Figure 2.2.7.2**: XML Interface generator code for logout button

**LIST OF TABLES**

**Table 1.4**: Functional Requirements Table

**ABBREVIATIONS AND ACRONYMS**

**ID**: Identification

**XML**: eXtensible Markup Language

**CSS**: Cascading Style Sheets

**CRM**: Customer Relationship Management

**DCS**: Diploma in Computer Science

**ICT**: Information and Communication Technology

**DEFINITION OF KEY TERMS**

**Personal ID**: A document that serves as a means of personal identification, such as a passport or national ID.

**Document Keeper**: A system designed to securely store and manage personal documents.

**Authentication**: The process of verifying the identity of a user or system.

**Cloud Storage**: A model of data storage where digital data is stored in logical pools, the physical storage spans multiple servers.

**Encryption**: The process of encoding data to prevent unauthorized access.

**ABSTRACT**

The Personal ID and Document Keeper System is designed to offer a secure and efficient solution for managing personal documents. This system addresses the problem of misplacing or losing important documents by providing a virtual online vault for storage. The system features encrypted authentication, ensuring that all documents stored are secure. This project documentation provides a detailed overview of the system, including its objectives, functional requirements, user interfaces, backend logic, and quality tests. It also discusses the system's effectiveness in ensuring the security and traceability of personal documents.

**CHAPTER ONE: SYSTEM OVERVIEW**

* 1. **Statement of Problem**

My sister was planning to travel abroad in the next 3 months to pursue her higher level of education. She seems to have no clue of where all her personal and issued documents are. Due to this she finds it so hard to make the planned trip to pursue her interest in leaving the country. This pressures her and gives her headache in finding herself in crisis where she can’t lay hands on the vital documents and ID’s, either misplaced or lost or just far away when she needs them in this critical time.Now she can’t run her errands as desired as the trip requires her to have a passport, visa and even the national ID.

* 1. **Study Justification**

The solution aims at gatekeeping important documents safely in some sort of virtual online vault that contains a cloud storage. The virtual online vault contains features that include encrypted authentication such as passwords’ that make all stored documents to be secure at all costs. The system provides efficiency and effectiveness in tracing of saved documents in the system. The system stores all user data in a database which only the specific user can obtain from making the system to be free from any vulnerable attacks eyeing on sensitive information.

* 1. **System Objectives**

**1.3.1 General Objective**

1. To develop a personal ID and Document keeper system.
   * 1. **Specific Objectives**
2. To develop encrypted authentication passwords to secure user data in the system.
3. To develop a proper functional database that interacts with the interfaces.
4. To implement the system’s efficiency to the end-user when using the system.
5. To initiate a fully responsive system that enhances the effectiveness of the system.
   1. **Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **User** | **User Activities** | **Features** |
| Individual | -Sign Up | -Sign Up form |
|  | -Log in | -Log in page |
|  | -Manage account | -Account /Profile pages |
|  | -Upload document type | -Document Interactions |
|  | -Document Retrieval and Search | -Document Preview |
|  | -Document Sharing and Exporting | -Documents Download |
|  | -Log Out | -End session |

**Table 1.4 Functional Requirements Table**

**CHAPTER TWO: FRONT END**

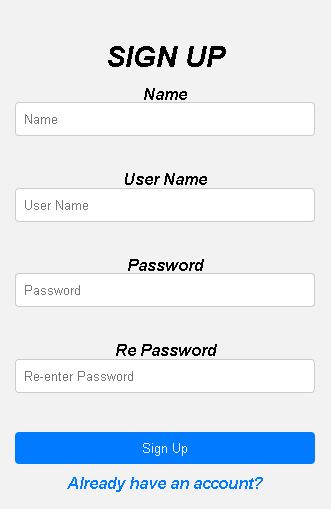
**(USER INTERFACES)**

**2.1 Introduction to User Interfaces**

This chapter contains screenshots of all the different user pages and which purpose they are built to serve the user.

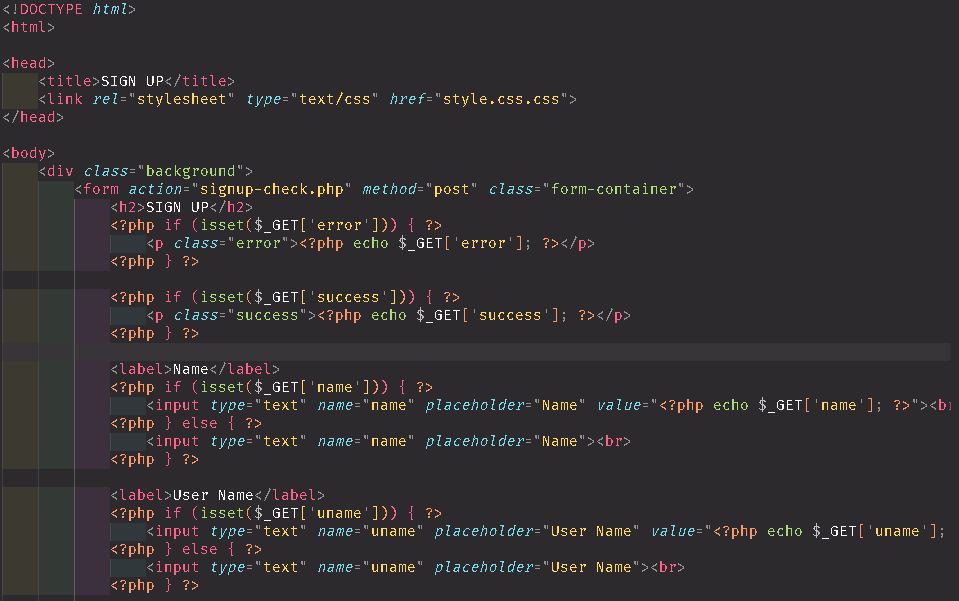
**2.2 Individual User Interfaces**

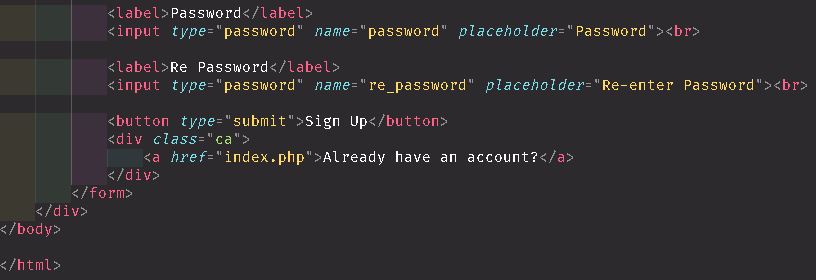
**2.2.1 Sign Up Form**



**Fig 2.2.1.1 Sign Up Form**

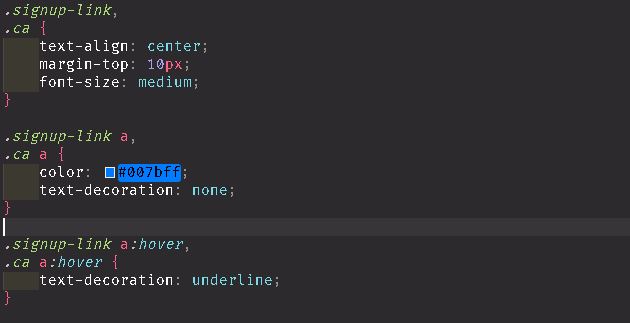
The above figure is a sign-up form where the customer will enter their details to create a new account.





**Fig 2.2.1.2. XML Interface generator code for individual sign up form**

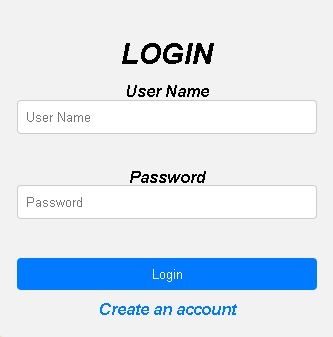
The above is the activity\_individual\_sign\_up\_form.xml code containing the properties of the sign up form.



**Fig 2.2.1.2. XML Interface generator code for individual sign up.css form**

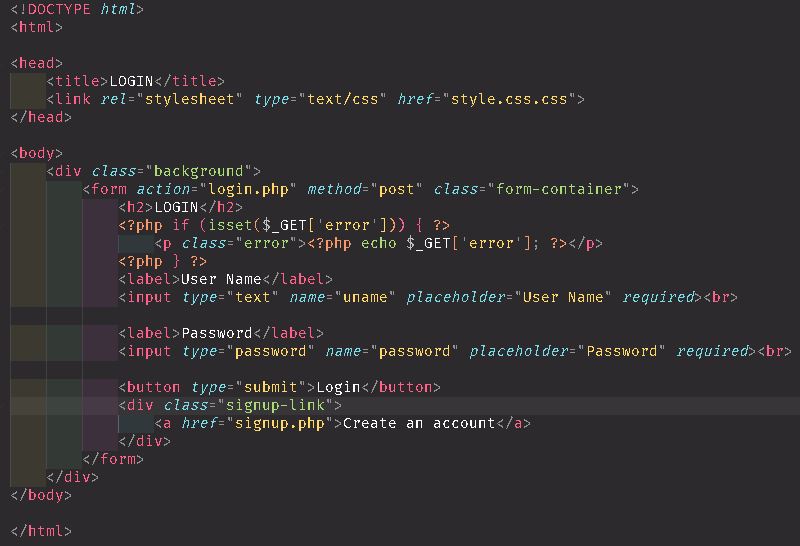
The above is the activity\_individual\_sign\_up\_form.xml code containing the properties of the sign up.css form.

**2.2.2 Log in Page**



**Fig 2.2.2.1 Login Page**

The above figure is a login page where the individual will enter their details to sign in to an account.



**Fig 2.2.2.2. XML Interface generator code for individual login page**

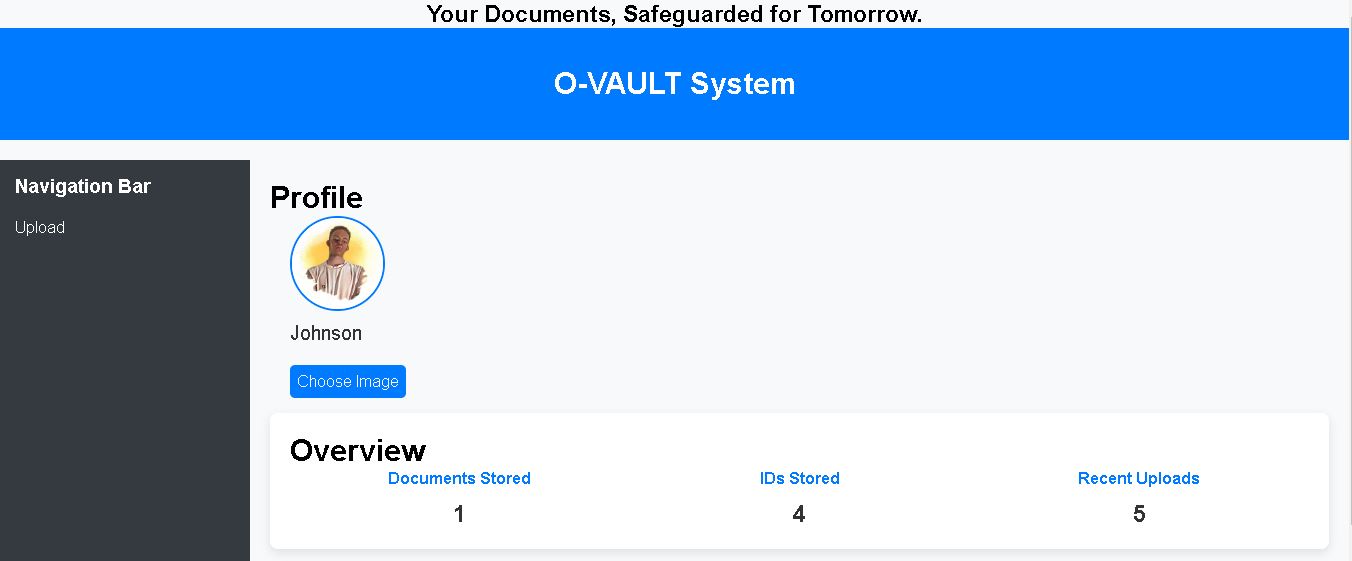
The above is the activity\_individual\_sign\_up\_form.xml code containing the properties of the login page.





**Fig 2.2.2.3. XML Interface generator code for individual login.css page**

The above is the activity\_individual\_sign\_up\_form.xml code containing the properties of the login.css page.

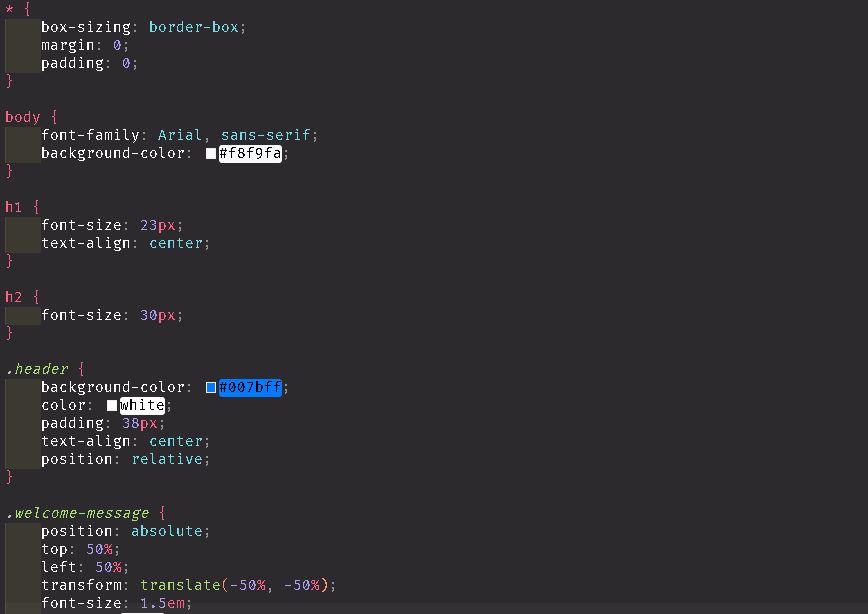
 **2.2.3 Account /Profile pages**

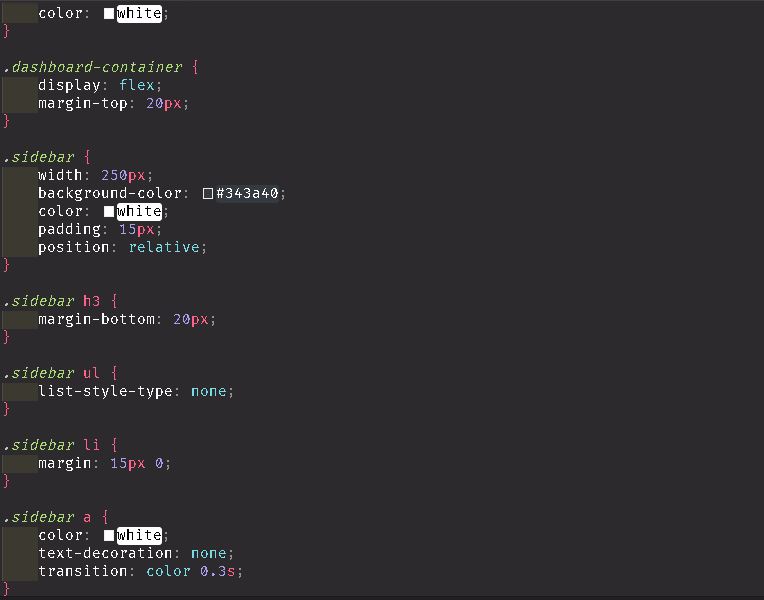
**Fig 2.2.3.1 Profile Page**

The above figure is profile accounts page where the dashboard displays all details of the individual as registered in the system.

**Fig 2.2.3.2. XML Interface generator code for profile accounts**

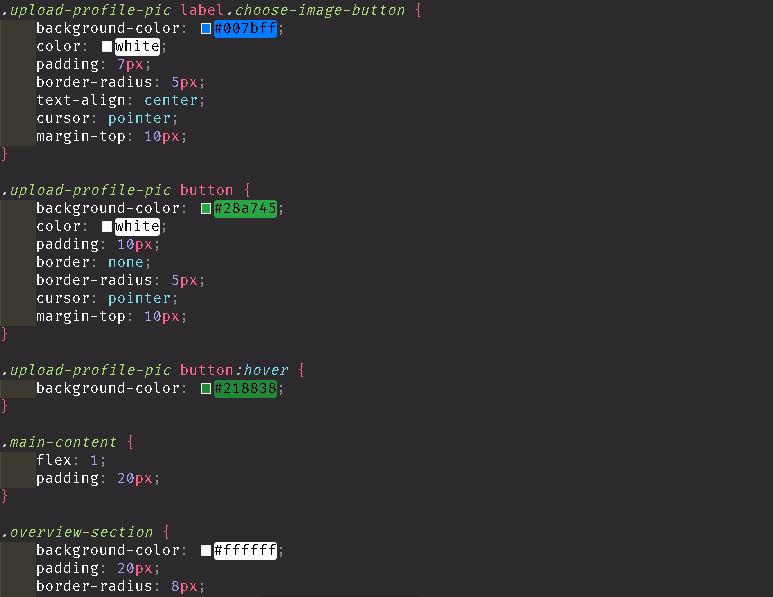
The above is the activity\_individual\_sign\_up\_form.xml code containing the properties of profile accounts of the individual.

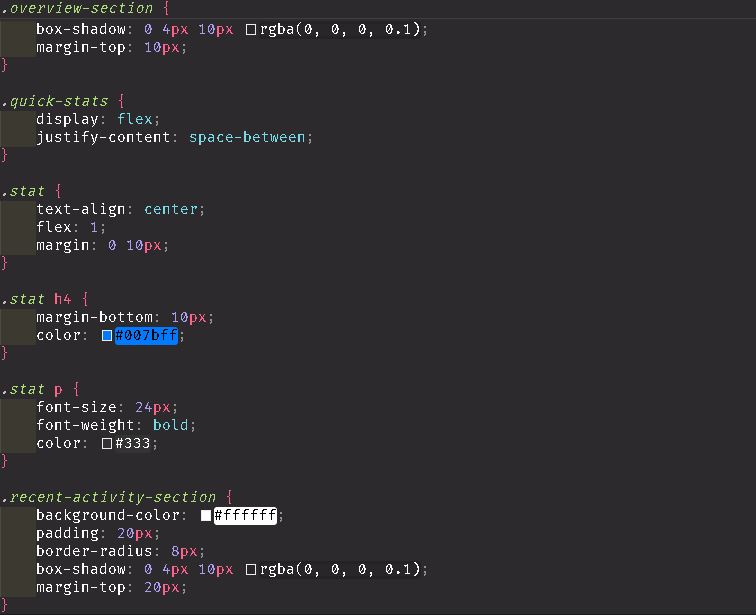










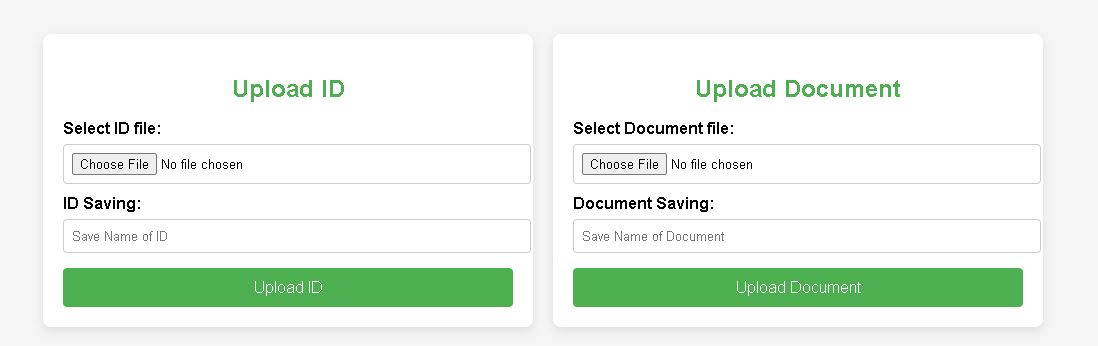




**Fig 2.2.2.3. XML Interface generator code for profile accounts**

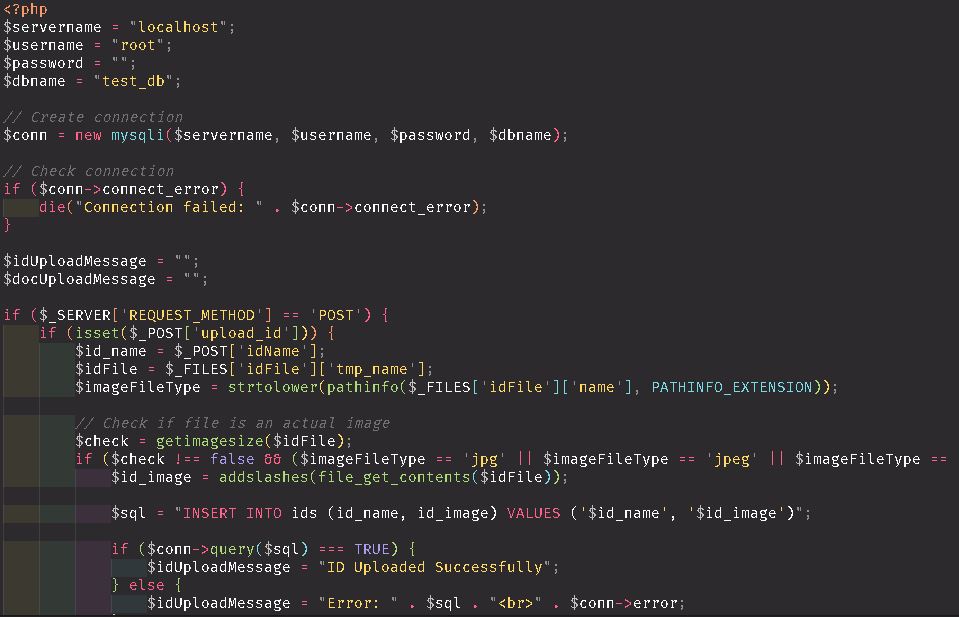
The above is the activity individual profile accounts form.xml code containing the properties of the dashboard page.

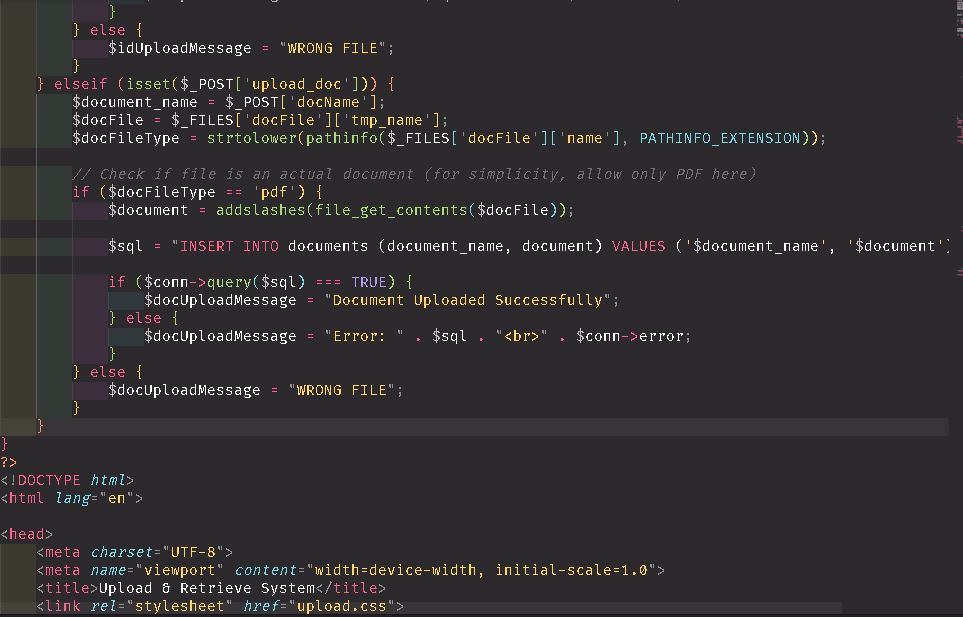
**2.2.4 Document Interactions**

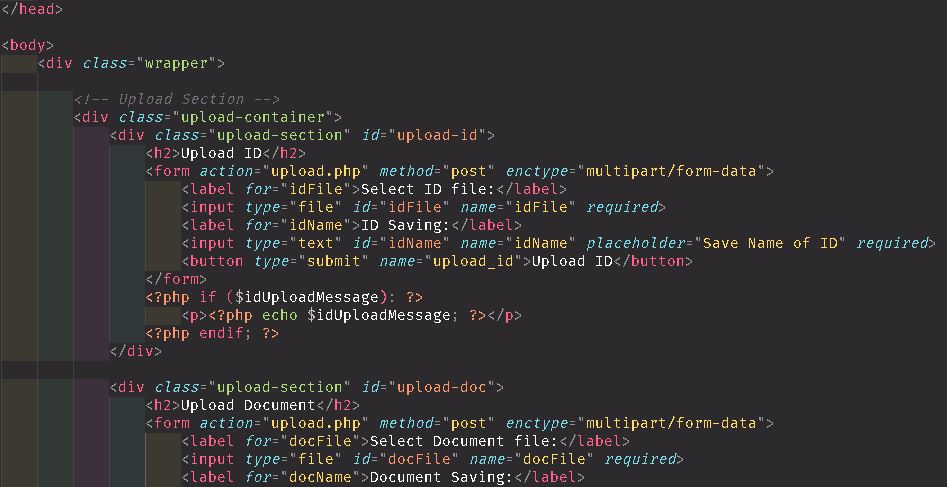


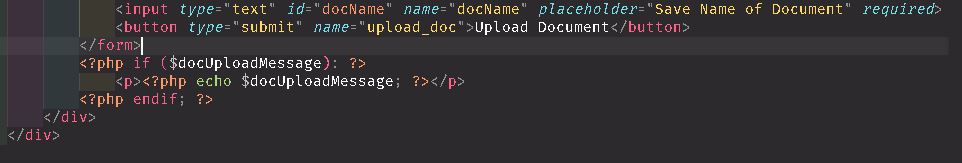
**Fig 2.2.4.1 Document Interactions Page**

The above figure is an ID and document interactions page where the individual will upload either ID and document to the database.







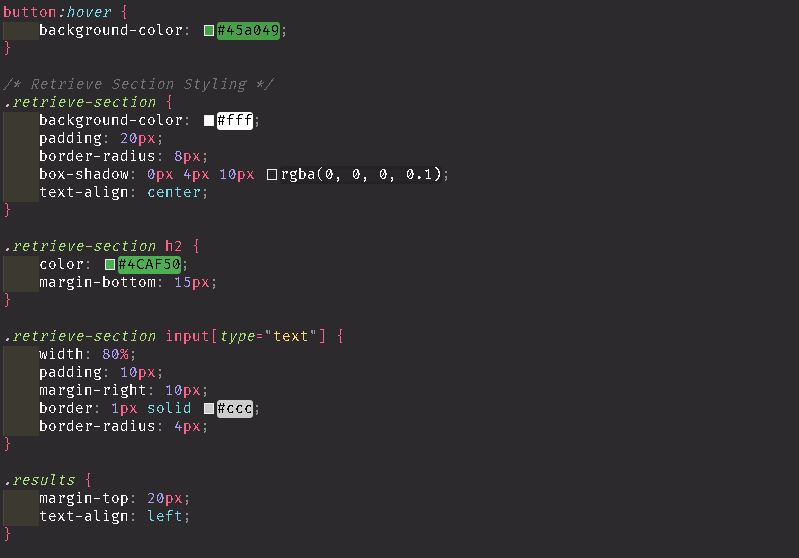


**Fig 2.2.4.2. XML Interface generator code for ID and document uploads**

The above is the activity individual document interactions form.xml code containing the ID and document properties.







**Fig 2.2.4.3. XML Interface generator code for upload.css**

The above is the activity individual document interactions form.xml code containing the ID and document for upload.css properties.

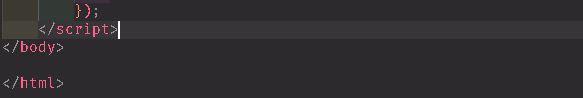
**2.2.5 Document Preview**



**Fig 2.2.5.1 Documents Section**

The above figure is an ID and document preview page where the individual will retrieve any ID and document stored in the database.

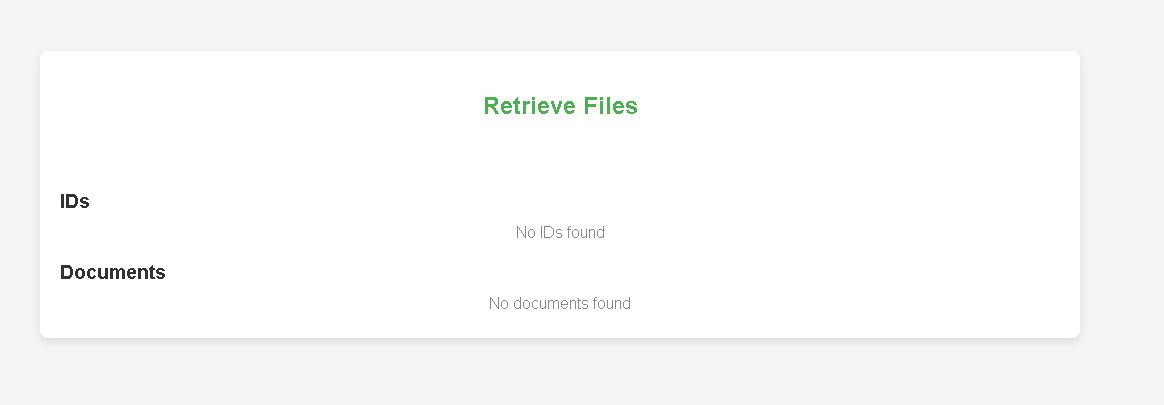




**Fig 2.2.5.2. XML Interface generator code for ID and document retrieve**

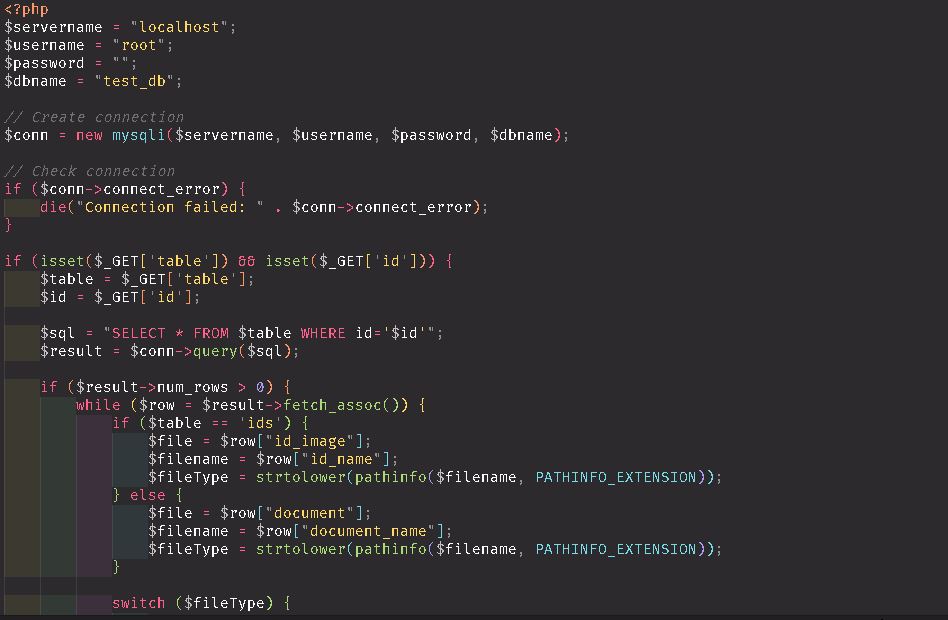
The above is the activity individual document fetching form.xml code containing the ID and document retrieve properties.

**2.2.6 Documents Download**



**Fig 2.2.6.1 Download Section**

The above figure is an ID and document download page where the individual will download any ID and document fetched in the database.



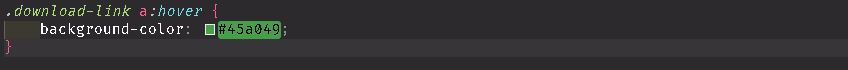


**Fig 2.2.6.2. XML Interface generator code for ID and document downloads**

The above is the activity for individual document downloading form.xml code containing the ID and document download properties.







**Fig 2.2.6.3. XML Interface generator code for download.css**

The above is the activity individual document downloads form.xml code containing the download.css properties.

**2.2.7 End session**



**Fig 2.2.7.1 System Logout**

The above figure is an Logout page where the individual will click the logout button to be directed out of the system closing all operations.



**Fig 2.2.7.2. XML Interface generator code for logout button**

The above is the activity for logout activity which enable the individual to end the sessions done to the system.

**CHAPTER THREE: BACK END**

**(DATABASE AND LOGIC)**

**3.1 Introduction**

This chapter focuses on the implementation of database structure and back-end code ensuring interactivity of the system.

**3.2 Database Implementation.**

The database implementation was conducted using SQL, integrated within Visual Studio Code as the development environment. All other components of the system's development were also performed in VS Code, which provided a robust platform for efficient development. SQL was selected as the database language due to its structured and systematic approach to data organization, facilitating easy storage, management, and retrieval of information from the created database. Data was organized into tables structured with rows and columns, ensuring clarity and order. Each table corresponds to a specific entity, such as users, document uploads, or ID uploads. Relationships between these tables were established in a responsive manner to maintain consistency across the interfaces. Visual Studio Code's powerful extensions and tools for SQL development enhanced the process of designing, testing, and defining the database while allowing seamless integration with other system components. This integration facilitated a streamlined workflow and effective database management.

### 3.2.1 Setting up SQL Database

1. *Install SQL and Setup Visual Studio Code*

SQL tools were also installed/configured to have a good experience developing these steps in VS CODE and setting up the database. Here's how it was done:

*Installing SQLite or MySQL*

*SQLite Installation:*

SQLite is lightweight, simple to use and great for small projects that I downloaded from their official site.

After installation, it has created a database file file in local directory.

*MySQL Installation:*

For smaller projects, we considered SQLite an option; and for larger projects that needed a server, MySQL was another candidate.

Use MySQL, setup the server to host on port 3306 (default), and provide a root password (example: password123) for security.

*Installing Visual Studio Code*

Visual Studio Code was downloaded from its official website and installed

We added extensions like SQLTools from the Extensions Marketplace for SQL in VS Code. They further simplified the process of writing and executing SQL queries.

*Creating a Workspace in VS Code*

Since all project files need to be organized, I created a new folder and opened it in VS Code as the workspace.

We modified configuration settings in VS Code to enable seamless integration with the SQL tools.

*Step 1: Install SQL Extensions in Visual Studio Code*

Few extensions installed and configured for VS Code to be able work with SQL databases dynamically;

*Adding Extensions for SQL*

We have already installed SQLTools extension from the Extensions Marketplace in order to manage our database using VS Code.

Support for SQLTools MySQL/MariaDB driver was added to connect to MySQL databases (my), with similar setups available as above for PostgreSQL, Oracle, and Microsoft SQL.

*Configuring the SQL Extension*

This added a new entity in the SQLTools extension settings:

Driver: Choose SQLite or MySQL, based on the database

MySQL: localhost for host SQLite does not need a host.

Port: Specific port 3306 which is the MySQL default was used.

Database File: Used for SQLite, the provided path to the database file.

Credentials: For MySQL, the username and password were provided (root/password123).

*2.Table Configuration In Databases*

After connecting to the database, I created tables which would organize the data in VS Code:

*Writing SQL Scripts*

Table definitions were then written in SQL script in VS Code. This code created a table for user data, for example:

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Constraint** |
| UserID | INT | Primary Key |
| UserName | VARCHAR(255) | Not Null |
| Password | VARCHAR(255) | Not Null |
| Confirm Password | VARCHAR(255) | Not Null |

Testing and Running Queries

The SQLTools extension was utilized to run the SQL scripts and test whether the setup of tables was proper.

*Populating Data*

SQL INSERT statements for adding test data into the tables were done to ensure that the database would work as expected.

**3.2.2 Creating and Testing the Tables**

All data in the system are stored in a relational SQL database. Tables are the repositories that hold the data. The steps outlined below were undertaken to create and validate the database tables.

*Step 1: Normalization*

Normalizing the database means it is arranged in a manner to reduce redundancy, avoid anomalies, and ensure consistency. The following normal forms were used for this purpose:

*First Normal Form (1NF)*

The raw data was organized into rows and columns with atomic values. Any repeating groups were removed.

Raw Data:

Username, Password, Email, IDType, IssuedDate, FileName, FileType, UploadDate.

*Tables Identified:*

Users: Contains details about system users.

Personal IDs: Stores government-issued or personal identification information.

Documents: Records information about uploaded documents.

Document Categories: Categorizes documents into types (e.g., Academic, Personal, Employment).

*Second Normal Form (2NF)*

Non-key attributes were fully dependent on the primary key. Composite keys were avoided.

The Document Categories table was introduced to remove category information from the Documents table.

Personal IDs were stored separately in the Personal IDs table to maintain modularity.

*Third Normal Form (3NF)*

Transitive dependencies were removed such that non-key attributes depend only on the primary key.

Final Tables and Relationships

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Constraint** |
| UserID | INT | Primary Key |
| UserName | VARCHAR(255) | Not Null |
| Password | VARCHAR(255) | Not Null |
| Confirm Password | VARCHAR(255) | Not Null |

*1.Users Table*

This holds the login credentials and personal details of users, such as students.

*2.Personal IDs Table*

Stores personal identification information for users.

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Constraint** |
| ID | INT(11) | Primary Key |
| User ID | INT(11) | Not Null |

*Table 3.2.2-1 Users Table*

3*.Documents Table*

Stores information about documents uploaded by users.

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Constraint** |
| ID | INT(11) | Primary Key |
| User ID | INT(11) | Not Null |

*Table 3.2.2-2 Documents Table*

4.*Activity Logs*

Displays Activity Logs what user has performed.

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Constraint** |
| ID | INT(11) | Primary Key |
| User ID | INT(11) | Users(id) |

*Table 3.2.2-3 Activity Logs Table*

*Step 3: Relationships*

One-to-many relationship. One user can have multiple personal IDs.

One-to-many relationship. One user can upload many documents.

Many-to-one relationship. There can be one category that each document belongs to, but many documents might be in each category.

**Vault System**

**ID’s**

**Documents**

**Database System**

**Users**

**UserID**

**UserName**

**Role**

**Password**

**Admin**

**UserID**

**UserName**

**Role**

**Password**

**3.2.2-1 Entity Relationship Diagram**

ERD describes the entities involved (Users, Documents, PersonalIDs, DocumentCategories), their properties, and their relationships.

*Step 4: Table Creation in PyCharm*

Creating the tables in PyCharm requires that the following steps are followed;

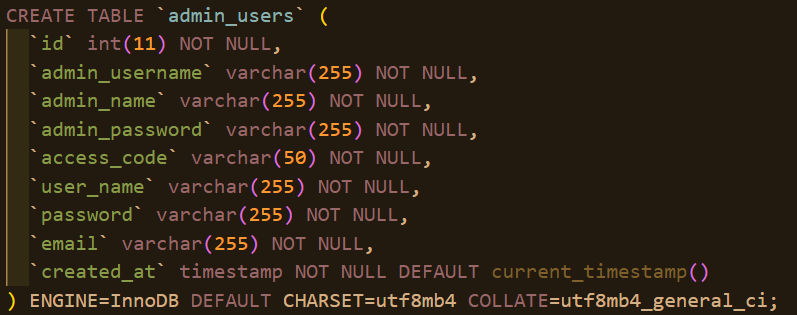
Open the Database Tool Window:

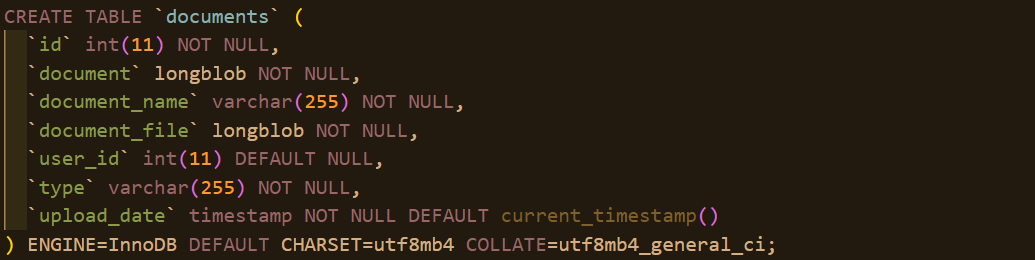
View > Tool Windows > Database

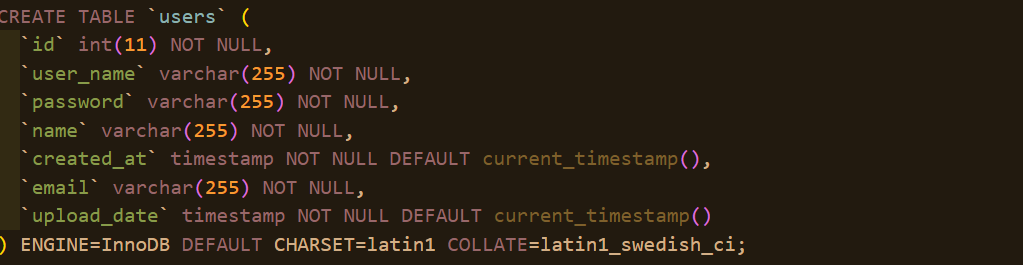
proceeding to click on an icon + then select Data Source > MySQL then fill in your connection details and test the connection.

*SQL Console:*

Right-click your database node, for example, PersonalIDDB then New >> Console. Run SQL Commands: Paste the SQL commands below into the console:







*Run Commands:*

Click the green Run button. You should see Query executed successfully for each command.

**3.3 Logic**

This section will include samples of the backend code that would make the system work, such as allowing user logins, uploading documents, and filing documents into categories.

**3.3.1 Login Functionality**

In the Personal ID and Document Keeper System, the login functionality allows users to securely access their accounts. Depending on their credentials, users can either access the Personal Dashboard (for students) or the Admin Dashboard (for coordinators/administrators).

*Key Features*

Input Validation: Checks for empty fields or invalid formats (e.g., incorrect email).

Authentication: Validates user credentials against stored data in the database.

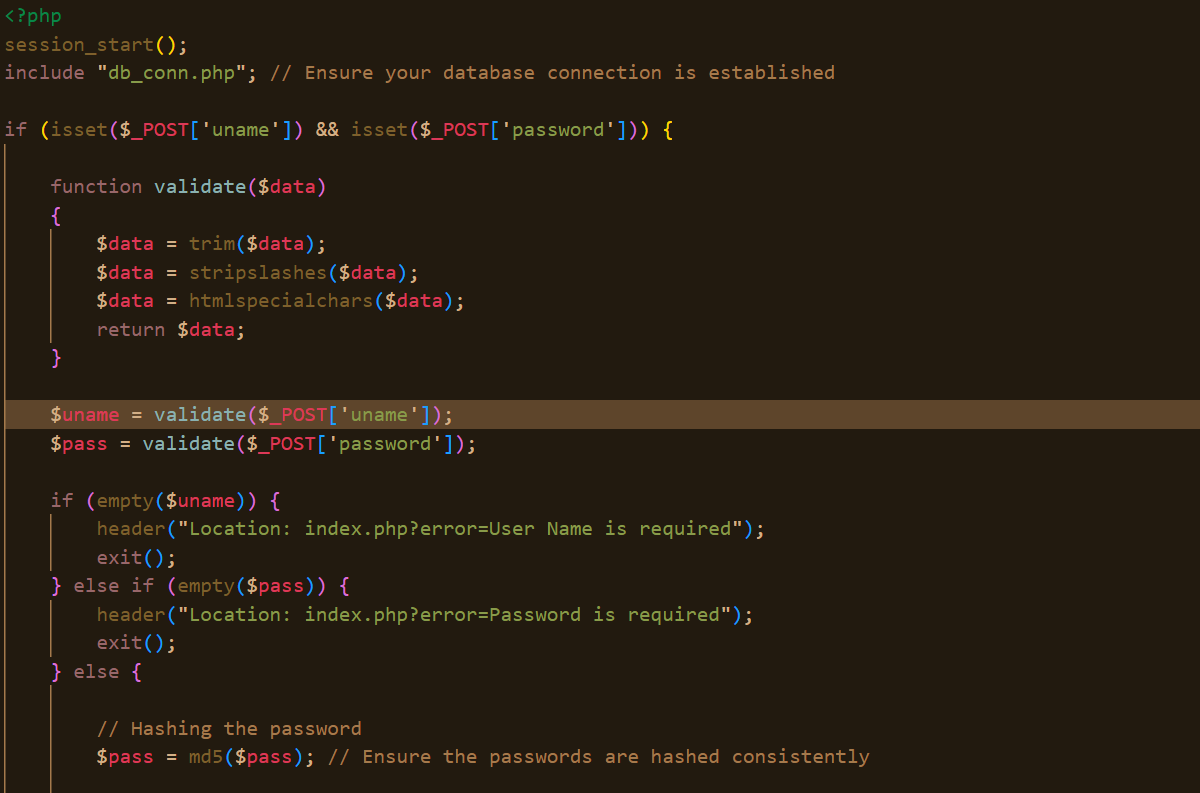
Navigation: It routes students to their personal dashboard while the coordinator is routed to the admin's dashboard.

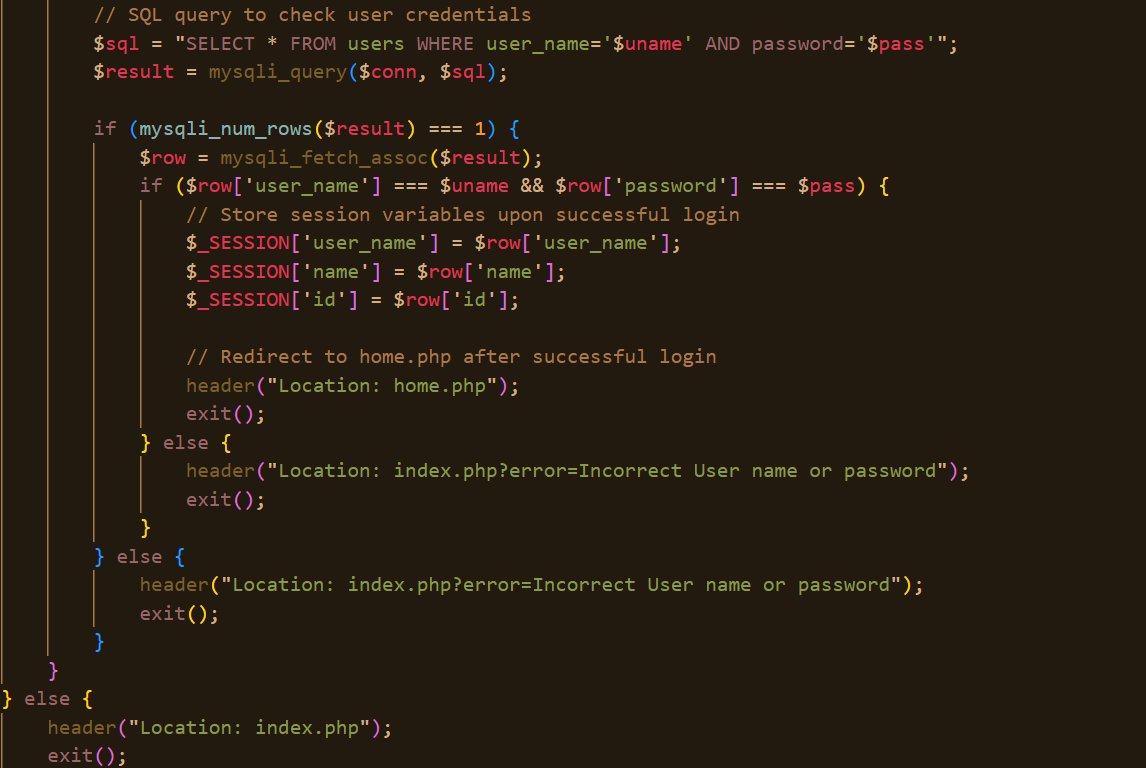
Error Handling: Displays proper error messages in case of wrong credentials or blank fields.

Reset: The cancel button resets all input fields.

*Logic Implementation*

Following is a code snippet taken from the LoginPage.php file, which does database connectivity, field validation, and processing of login data.





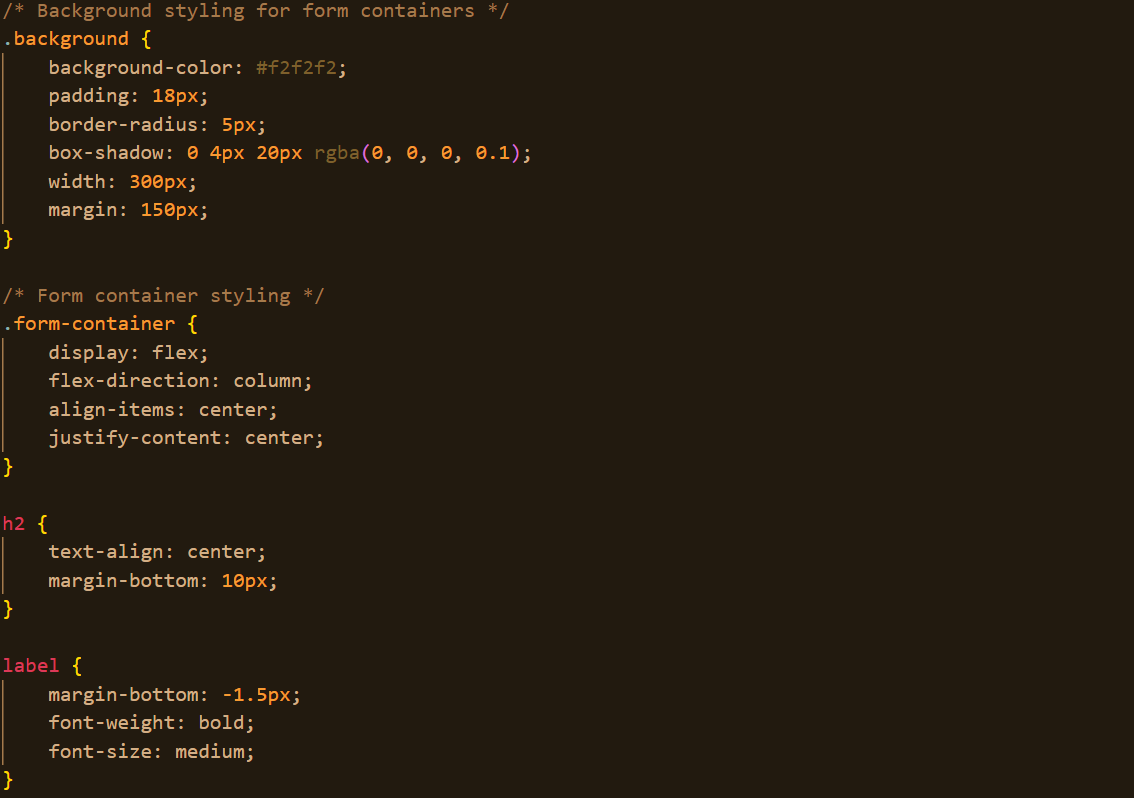
*Error Messages*

Empty Fields:

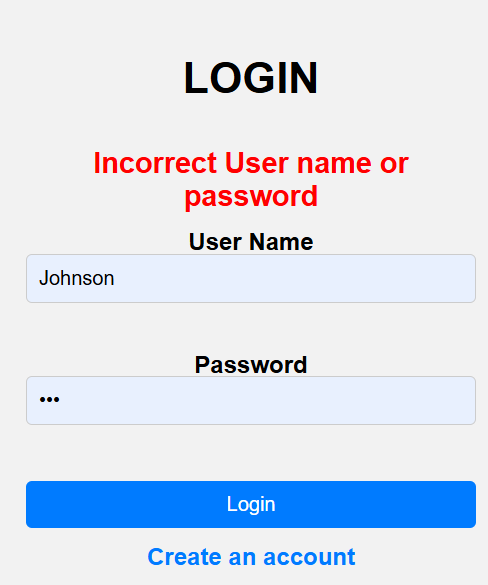
Message: "All fields are required!"

Triggered when the user clicks Log In without filling all fields.

Css code…



Error UI Example:



*[Login Error]*

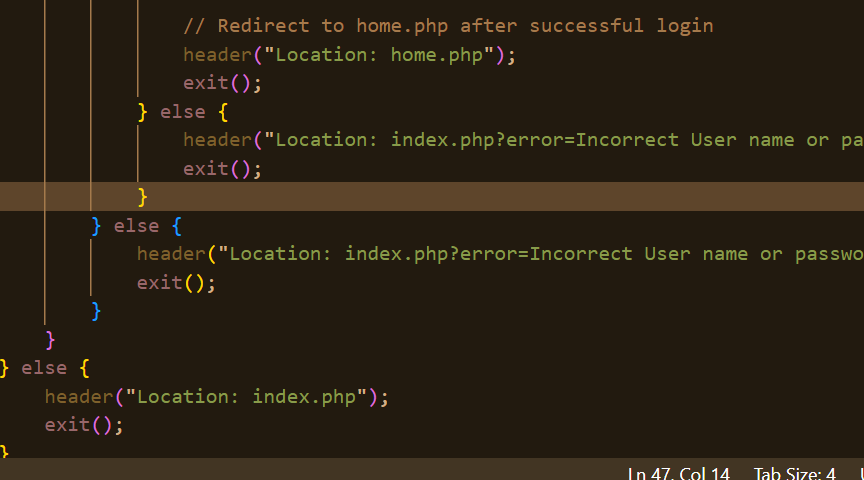
All fields are required!

*Invalid Credentials:*

Message: "Incorrect username or password!"

Trigger: When credentials provided do not match any record in the database.

Sample Error UI:



*[Login Error]*

Incorrect username or password!

*Database Connection Error:*

Message: "Error connecting to database."

Trigger: When there is an error in connecting to the database.

**3.3.2 User Management**

Admins (e.g., coordinators) can manage users by adding, editing, or deleting entries. This feature ensures that only authorized individuals can access the system.

*Key Features*

Adding Users: Allows the admin to add a new user (student or admin).

Editing User Details: Enables updating user information.

Deleting Users: Provides a delete option with confirmation prompts.

*Logic Implementation*

Below is the logic for managing users, extracted from AdminDashboard.php





*Error Handling*

Duplicate Email:

Message: "This email is already in use!"

Empty Fields:

Message: "All fields must be filled to save changes."

Delete Confirmation:

Prompt: "Are you sure you want to delete this user?"

**3.3.3 Document Management**

Users can manage and view their documents, which can be uploaded and organized into categorized sections.

*Key Features*

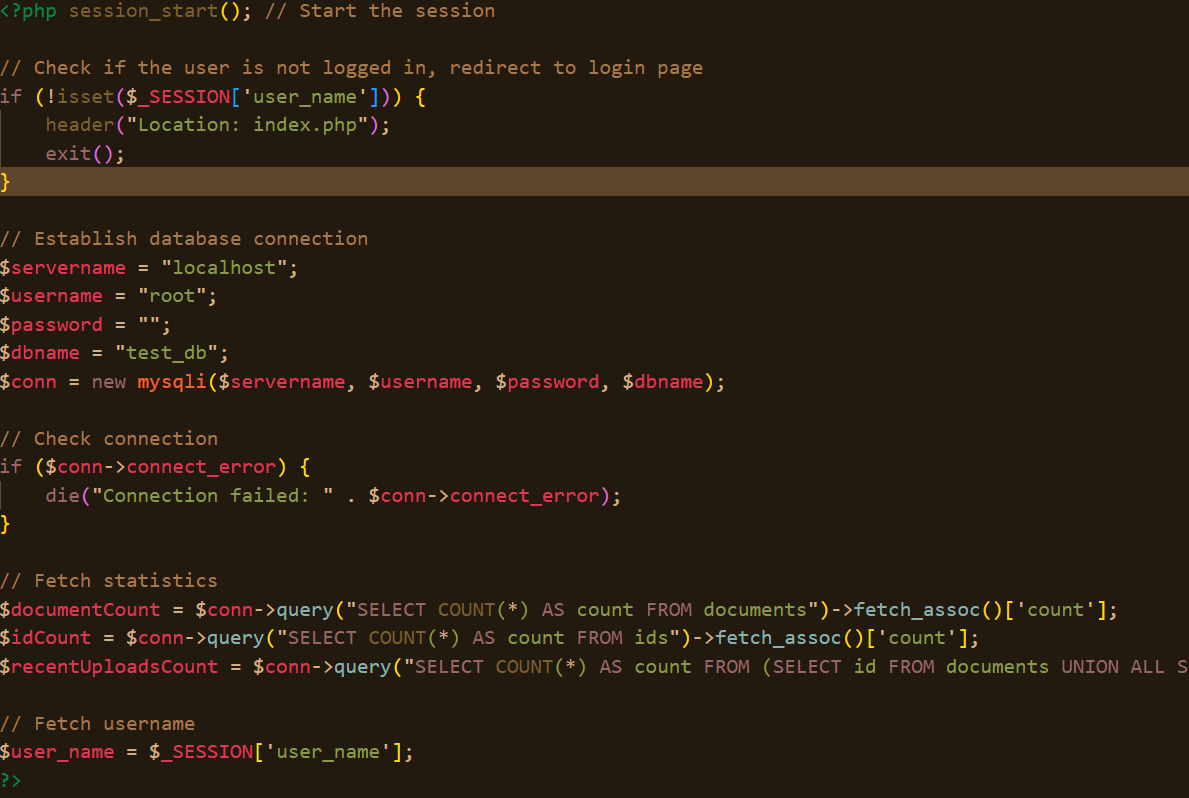
Uploading Documents: Users are able to upload documents that will fall under the database along with metadata such as the date of upload and type of file.

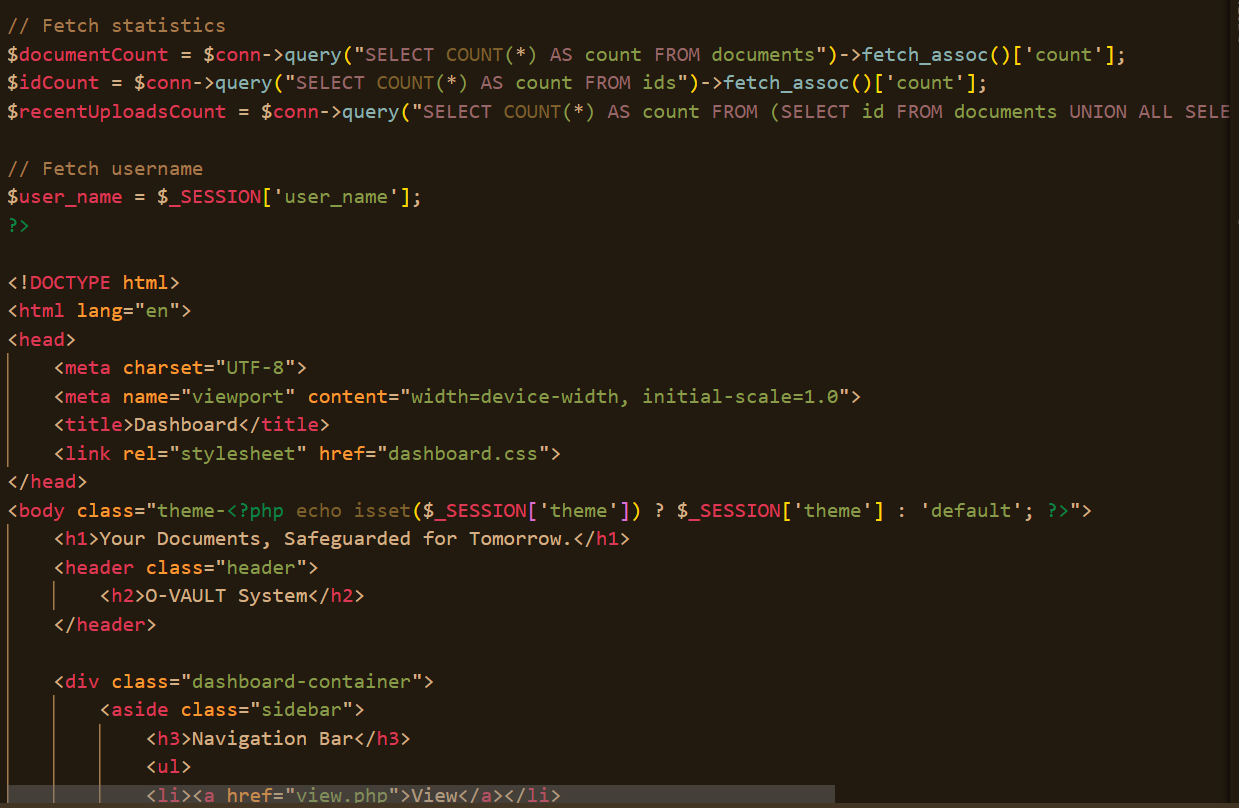
Document View: Lists uploaded documents with display categorized.

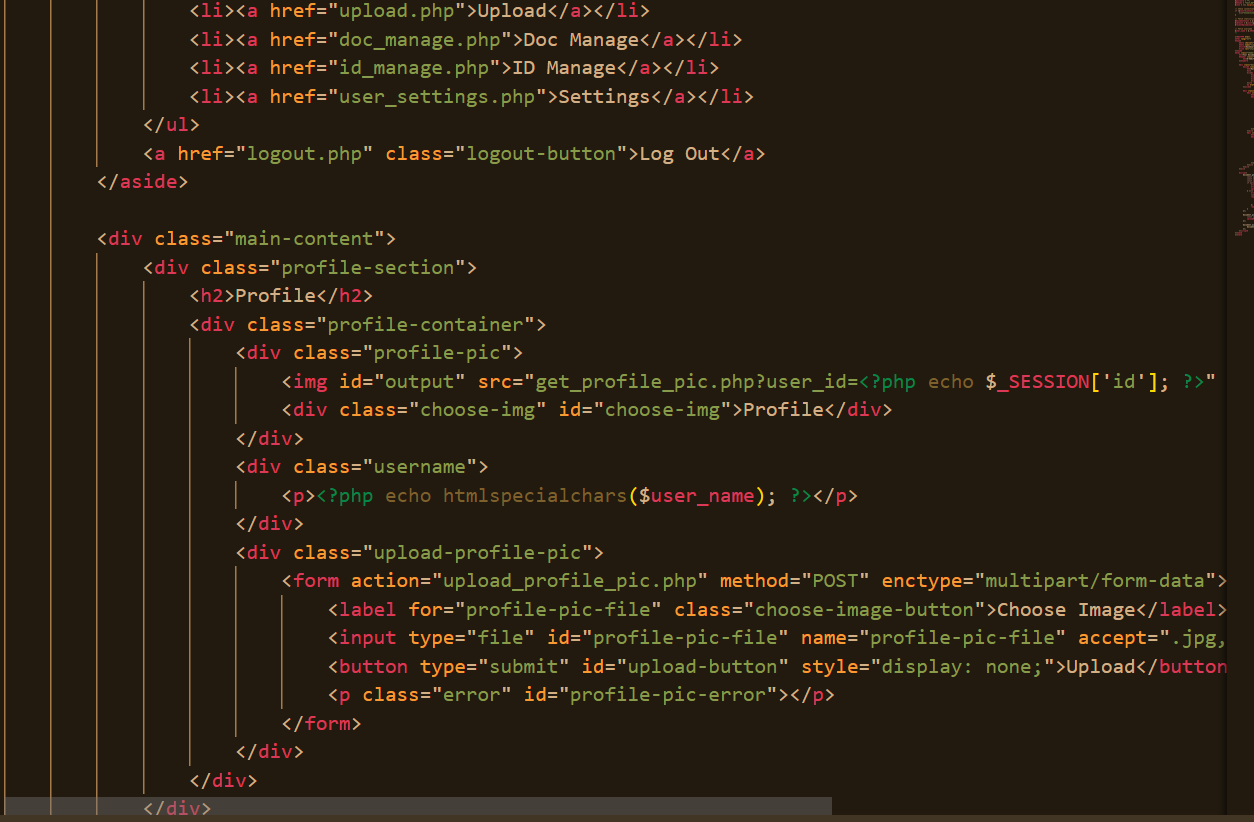
Document Organization: Users can categorize their documents.

Implement Logic

Here is the logic to manipulate documents extracted from UserDashboard.php:











*Handling Errors*

*Upload File Errors:*

Message : "Unsupported file type."

Upload Fields Empty:

Message: "Please select a file to upload."

**CHAPTER FOUR: SYSTEM QUALITY TESTS**

**4.1 Introduction**

The following feature and database integration tests will ensure that the system is working as required, handling users' entries, uploading documents, validating, and storing them securely.

**4.2 User Entry and Validation Test (Feature Test)**

Objective: To validate whether users' personal information has been entered correctly and if any errors are handled accordingly.

*Steps:*

Open the Admin Dashboard.

Click on Add Record to add a new row.

Fill in the details:

Name: Johnson Kamau.

Admin Serial Code: 0464.

Phone: 0768280952.

*Test by:*

Empty one of the fields.

Inserting an invalid email, for example, johnsonkamau10@gmail.com

Using an ID Number that is a duplicate.

*Expected behavior :*

Valid Input:

Row goes into read-only state.

Data saved in database.

Invalid Input:

Highlighted error messages like

"Email must be in valid format!".

Doesn't allow moving further without changing.

*Incorrect behavior :*

Allows saving of empty or invalid fields.

Doesn't throw appropriate error messages.

**4.3 Document Upload Validation Test (Feature Test)**

Objective: Test document upload for certain criteria, such as file format and size limits.

*Steps:*

From the application, open the Document Manager for a user record.

Click the Upload Document button.

Select a document to upload:

Valid document: A PDF document ID\_Card.pdf (size 500 KB).

Invalid document: A text file or a PDF over 5 MB.

*Expected Results:*

*Upload Valid Document:*

Should show "Upload successful!".

Save the document to the database or file storage system.

Upload Invalid Document:

Should display one of the following error messages:"Unsupported file format. Only PDF/JPEG allowed.".

"File exceeds maximum size of 5 MB.".

*Incorrect Output:*

Accepts unsupported formats or oversized files.

Uploads invalid formats/oversized files without an error message.

**4.4 Search and Retrieval Test (Feature Test)**

Objective: To confirm that records and associated documents are searchable and retrievable.

*Steps:*

On the dashboard, go to upload page,

Choose whether to save ID’s or Documents:

Upload the desired FILE and click upload:

You can later retrieve the desired Id or Document by searching how you saved it,

Attempt to access and view the user's record and associated documents.

*Expected Output:*

Valid Query:

Displays user's details and a list of uploaded documents.

Allows document download or preview.

Invalid Query:

Displays: "No matching records found!".

*Incorrect Output:*

Incorrect or incomplete search results.

Documents fail to load or display errors during retrieval.

**4.5 Database Integration Test: Save and Retrieve Records**

Objective: Verify that personal records and uploaded documents are correctly stored and retrieved from the database.

*Steps:*

Add a user record:

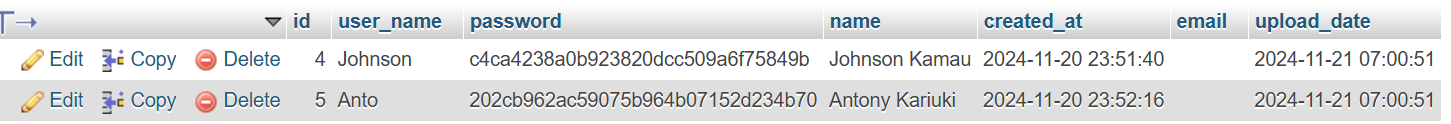
Name: Jane Smith.

ID Number: 987654321.

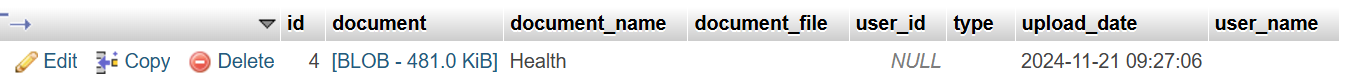
Upload a document: Certificate.pdf.

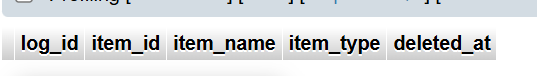
Restart the application and check if the record reappears.

*Expected Output:*

User Details Saved:

Saved Document:



The record and document are displayed properly after refresh.

*Incorrect Output:*

Data is missing or incomplete in the database.

The record doesn't load in the dashboard.

**4.6 Delete Functionality Test (Feature and DB Test)**

Objective: Verify that when a user is deleted, their record gets deleted along with their documents from the system.

*Steps:*

Delete any user record, say John Doe.

Check for deletion in database:

Expected Result: No rows returned

Verify associated documents are also removed from storage.

*Expected Output:*

Record and documents are removed from the interface and database.

Files are deleted from the storage directory.

*Bad Output:*

Record still appears in interface.

Data still exists in database or storage.

**4.7 Summary of Tests**

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Output** | **Incorrect Output** |
| User Input Validation | Valid entries saved; invalid entries prevented | Invalid/blank entries save without errors. |
| Document Upload | Validation Valid files uploaded; errors on invalid files | Invalid files uploaded; no error messages. |
| Search and Retrieval | Accurate results, document access | Missing/incorrect results/documents |
| Database | Save/Retrieve\_records/documents saved/retrieved accurately | Documents incomplete/missing upon restart. |
| Delete Functionality | Record and documents completely removed | Data still exists in the interface/database/storage. |

**CHAPTER FOUR: CONCLUSION AND RECOMMENDATIONS**

**5.1 Conclusion**

The Personal ID and Document Keeper System was developed, and the most important goal it aimed to satisfy is to manage and store personal identification information and documents in a secure way. It integrates main features that include user authentication, secure storage of documents, and a role-based access control mechanism so that only authorized users can interface with it. It offers an easy-to-understand and user-friendly interface that facilitates document uploads, retrievals, and organization.Although the system has faced various challenges in development, especially in integrating SQL with the Python-based application and the implementation of secure data storage, it has demonstrated that it can work effectively. This system will be useful for an individual or organization seeking a safe and secure way of managing sensitive personal data due to the backend structure, which provides support for document organization, retrieval of data, and data security.It was clear through this project that automation and digital storage solutions can very much ease the management of personal records by offering the most user-friendly and secure platform to satisfy most of the fundamental needs of its users. This system could potentially act as a scalable solution ranging from simple use cases for personal document management to organizations that deal with a wide variety of sensitive information.

**5.2 Recommendations**

While the core functionality of the Personal ID and Document Keeper System has been achieved, there are numerous areas where it could be enhanced to improve its usability, scalability, and overall performance. This would include enhancement in the user interface of the system. Inasmuch as the current design is somewhat functional, embracing interactivity like smooth transitions, animations, and tooltips will go a long way in enhancing the user experience. Incorporating multimedia elements, such as tutorial videos or manuals explaining each step of the process, is also recommended to help first-time users learn how to use the system more effectively-a key goal in onboarding and accessibility.This relates to multi-device and multi-platform support. Currently, the system is designed as a stand-alone desktop application. A much-wanted adaptation to multiple devices, such as smartphones, tablets, or web browsers, would greatly enhance its accessibility and convenience. This is where one may consider hosting the database on a cloud server and developing client-side applications or web interfaces that can let users safely access their personal documents from anywhere and with any device, which enhances flexibility and convenience.About resource optimization, the system can be adjusted to perform more effectively. For example, letting the system run in the background when not in use would conserve resources. The system would be thus designed to "wake up" only when necessary, for example: whenever the user wants to upload a document or when the scheduled reminder is triggered as a phone alarm might ring at a specific time. In this way, this system would not have excessive resource consumption and would be as efficient as possible.Since the level of the data managed is sensitive, security enhancements must be a priority. The system should be strengthened with higher-ordered encryption algorithms at the data storage and transmission levels for better protection of personal information. Additionally, the integration of sophisticated authentication methods, such as biometric authentication or 2FA, can help further secure access to the system and prevent unauthorized usage.The system could also be improved in terms of scalability and customizability. Making some features of the system tunable by users or administrators-for example, document types, fields, or security settings-would easily adapt it to the different needs of other organizations or personal needs. When the amount of documents being stored in the system is increasing, scaling will be crucial. This will ensure that the system optimizes its database for efficiency and can perform suitably while it's able to scale.One of the key features to implement would be event logging and notifications. Adding logging for user actions, document uploads, or changes in personal information will increase accountability and transparency. Another feature would include using notification systems through email reminders or even SMS alerts for users on document expiring date or updates needed, so usability would be an added dimension of proactive help towards the users.Further testing of the system is recommended to ensure that the system performs as anticipated in real-world environments. Further tests in different environments will afford an opportunity for certain performance bottlenecks or usability issues not apparent during development to be identified. Testing will enable collection of feedback from a larger user base, with refinement of the system being responsive to the needs of a wider audience. This will also be tested on various devices and operating systems to ensure that it works properly with all of them.Finally, integration with other systems could make the Personal ID and Document Keeper even more powerful. For example, integrations with cloud storage services such as Google Drive or Dropbox, or with government databases or HR systems, might make it easier to retrieve and manage documents. In addition, integration with calendar applications, such as Google Calendar, will enable users to synchronize document expiration reminders and update schedules, making the system proactive and seamless.By mooting these recommendations, the Personal ID and Document Keeper System would emerge as more robust, scalable, and user-friendly in handling personal identification information. In addition to making the system easily accessibly to a larger circle of people, this shall also guarantee security, efficiency, and adaptability of their needs in various environments.

**APPENDICES**

**Appendix I: Gantt Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **September** | | | | **October** | | | | **November** | | | **Dec** |
| **Activity** | **Week 1**  **(2nd-6th Sep)** | **Week 2**  **(9th-13th Sep)** | **Week 3**  **(16th-20th Sep)** | **Week 4**  **(23rd-27th Sep)** | **Week 5**  **(30th – 4th Oct)** | **Week 6**  **(7th – 11th Oct )** | **Week 7**  **(14th-18th Oct)** | **Week 8**  **(21st-25th Oct)** | **Week 9**  **(28th-1st Nov)** | **Week 10**  **(4th-8th Nov)** | **Week 11**  **(8th – 15th Nov)** | **Week 12**  **(2nd Dec)** |
| 1. Lecturer’s Introduction to Fundamental Concepts of a Software Project |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Development of a Workplan |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Developing U.I. |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Developing DB |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Logic implementation with tests |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Panel presentation |  |  |  |  |  |  |  |  |  |  |  |  |

**Appendix II: Resource Requirement and Economic Feasibility Assessment**

|  |  |  |
| --- | --- | --- |
| Resources | Purpose | Cost(Ksh) |
| Hardware Resource |  |  |
| 1.Laptop/PC | -Develop project(front end, back end-testing) -Presentation | -Already Available |
| Software Resource |  |  |
| 1.Visual Studio Code  2.xampp | -Front end implementation  -Logic code writing and testing  -Running the project file for HTM,CSS,Javascript,PHP,SQl languages. | -Open source |
| Labour |  |  |
| Development |  | -Free |
|  | TOTAL | Ksh 0 |
|  |  |  |

**REFERENCES**

Aspose. (n.d.). Merge images online tool. Retrieved from <https://products.aspose.app/imaging/image-merge>

CloudConvert. (n.d.). Convert files online for free. Retrieved from

<https://cloudconvert.com>

Dev MySQL. (n.d.). MySQL installer downloads. Retrieved from <https://dev.mysql.com/downloads/installer>

Dev MySQL. (n.d.). MySQL tutorial excerpt. Retrieved from

<https://dev.mysql.com/doc/mysql-tutorial-excerpt/8.0/en>

Python Software Foundation. (n.d.). Python documentation. Retrieved from <https://docs.python.org/3>

Python Software Foundation. (n.d.). Winsound module. Retrieved from <https://docs.python.org/3/library/winsound.html>

YouTube. (2023). School bell sound effect [Video]. Retrieved from <https://www.youtube.com/watch?v=4p5A0l75bUU>