Floor Management System with Plan Administration

GitHub Repo Link

https://github.com/jahnavichalla/Floor-Plan-Management

Use case Slides Link

Minimal

A presentation created with Slides.

https://slides.com/chakradharvanarasi-1/minimal/fullscreen

MoveInSync

Floor Management System with Plan Administration

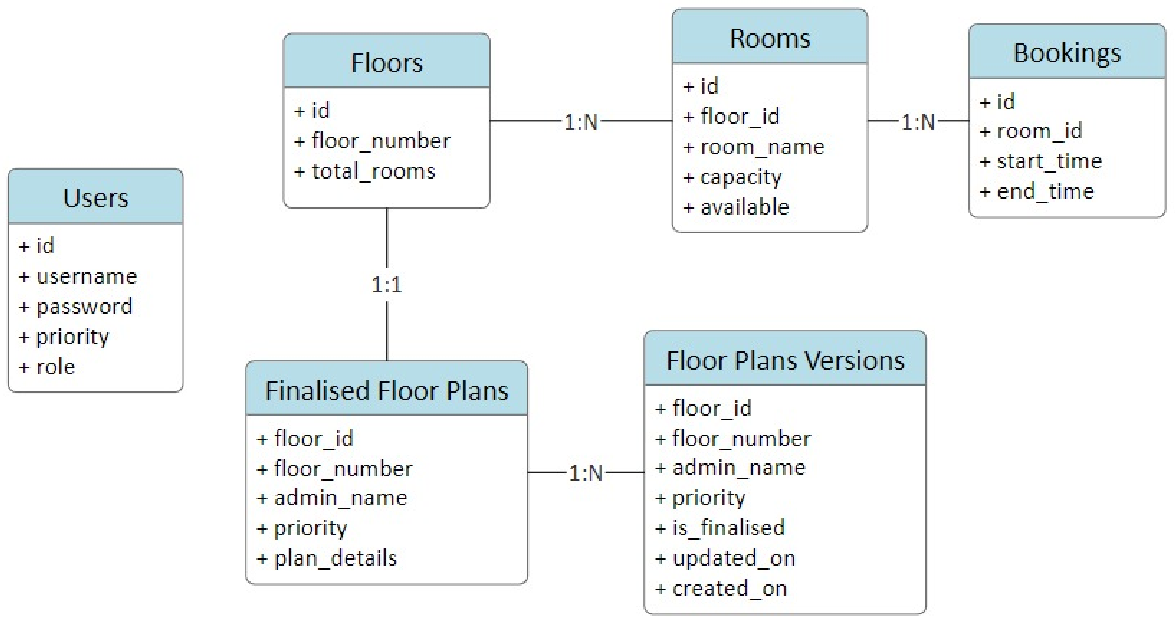
Demo Video Link

https://iiitbac-my.sharepoint.com/:f:/g/personal/vanarasi_chakradhar_iiitb_ac_in/Eij2kMlps_xLvto6oKeAhaEBRoiQ5HK5COwc04YT7Vc6WQ?e=dYCQIq

Introduction

Objective: This system is designed to streamline and optimize room and floor management in large facilities such as offices, universities, and event venues. It enables administrators to efficiently manage floor plans while allowing users to book rooms based on real-time availability and tailored recommendations. Additionally, the system handles delayed database connections by temporarily storing plan data locally during connection attempts, automatically syncing it once the connection is restored. This ensures seamless operation and data integrity, even in network disruptions.

UML Class Diagram



Technology Stack:

- **Java**: Core programming language used for developing the application.
- **JDBC (Java Database Connectivity)**: Facilitates the connection between the application and the MySQL database.

- **MySQL**: Relational database used for storing data related to users, rooms, and floor plans.
- **File I/O**: Manages temporary storage of plan data locally during database connection delays.
- **Terminal-Based Interface**
  - The application is designed with a command-line interface (CLI) for user interaction.
  - Users and admins interact with the system via prompts and text input, making it a lightweight and straightforward solution for managing room bookings and floor plans.
- **Libraries**:
  - *MySQL Connector*: A JDBC driver connects Java applications to the MySQL database.
  - *Java IO*: Handles file input/output operations for temporary data storage.

Installation and Setup

Prerequisites:

- **Java**: Ensure Java Development Kit (JDK) version 8 or above is installed on your system.
- **MySQL**: MySQL database server should be installed and running.
- **MySQL Connector for Java**: Make sure the JDBC driver (MySQL Connector) is available and added to your project classpath.

Database Setup:

- **Create Database and Tables**: Use the provided SQL schema to create the necessary tables(`users`, `floors`, `rooms`, `bookings`, `floor_plans`, `floor_plan_versions`).
- Insert initial data for testing.

Code Setup:

- Clone repository or download project files
- Open in IDE (e.g., IntelliJ IDEA, Eclipse)
- Update the database connection details in the `floorManagement` class (i.e., `DB_URL`, `USER`, `PASS`).

Running the Project:

- Execute `floorManagement` class and follow console prompts.

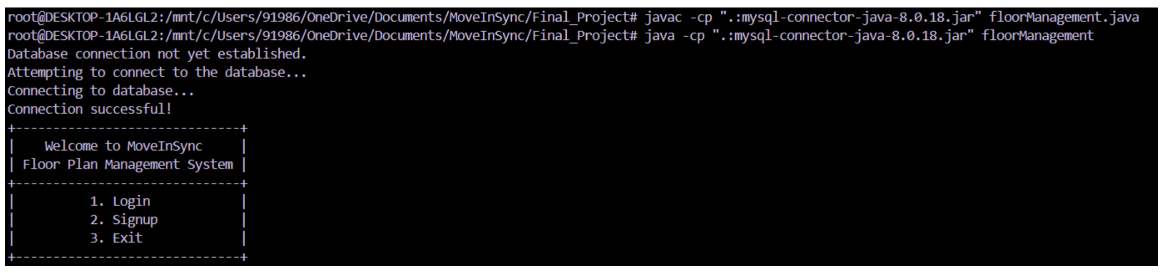
Common Issues:

- **Database Connection Issues**:
  - Ensure that the MySQL service is running.
  - Verify that the database connection credentials in the code align with your MySQL database settings.
- **Java Version Conflicts**:
  - Confirm that the installed version of Java on your system matches the version required for the project setup.

Features Overview:

1. Database Connection:

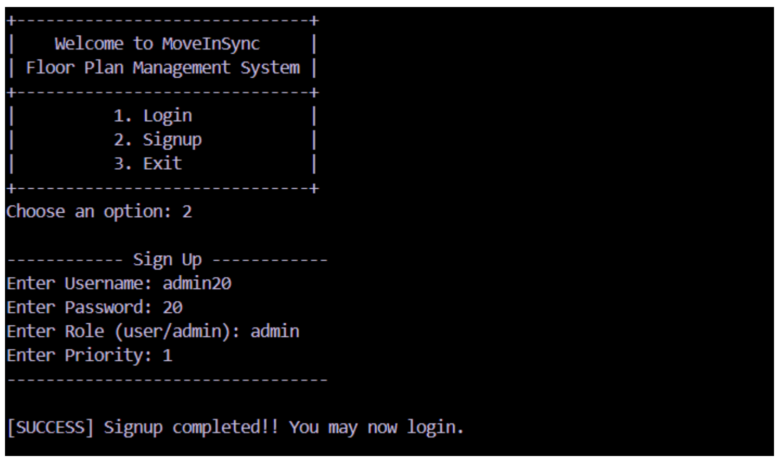
- When the server connection is down in offline mode, the system temporarily stores user inputs locally, including admin floor and room data.
- Once the database connection is successfully established, the system automatically syncs the locally stored data with the database, ensuring no information is lost during connection attempts.



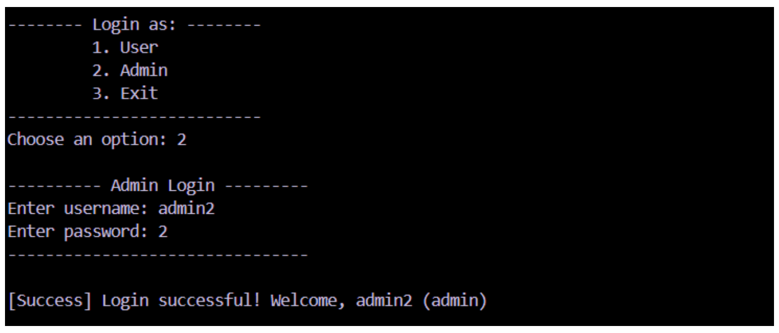
```
root@DESKTOP-1A6LGL2:/mnt/c/Users/91986/OneDrive/Documents/MoveInSync/Final_Project# javac -cp ".:mysql-connector-java-8.0.18.jar" floorManagement.java
root@DESKTOP-1A6LGL2:/mnt/c/Users/91986/OneDrive/Documents/MoveInSync/Final_Project# java -cp ".:mysql-connector-java-8.0.18.jar" floorManagement
Database connection not yet established.
Attempting to connect to the database...
Connecting to database...
Connection successful!
+-----------------------------+
|    Welcome to MoveInSync     |
| Floor Plan Management System |
+-----------------------------+
|         1. Login            |
|         2. Signup           |
|         3. Exit             |
+-----------------------------+
```

2. Username and Password Login System

- **Purpose**: The system offers a secure login mechanism for both admin and regular users.
- Each user has a unique username and is authenticated with a password.
- The system employs **Role-Based Access Control (RBAC)**:
  - **Admins**: Can access plan management features to manage floor plans and room data.
  - **Regular Users**: Can view available rooms, get personalized recommendations, and book rooms based on availability.



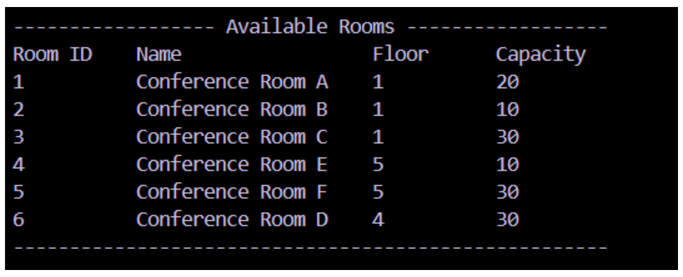
```
+-----------------------------+
|    Welcome to MoveInSync     |
| Floor Plan Management System |
+-----------------------------+
|         1. Login            |
|         2. Signup           |
|         3. Exit             |
+-----------------------------+
Choose an option: 2

----------- Sign Up -----------
Enter Username: admin20
Enter Password: 20
Enter Role (user/admin): admin
Enter Priority: 1
-------------------------------

[SUCCESS] Signup completed!! You may now login.
```



```
------- Login as: -------
        1. User
        2. Admin
        3. Exit
-------------------------
Choose an option: 2

--------- Admin Login --------
Enter username: admin2
Enter password: 2
------------------------------

[Success] Login successful! Welcome, admin2 (admin)
```

3. User Features:

1. **View Available Rooms**:
   - Regular users can see a list of rooms currently available for booking, based on real-time data.
   - The system displays only rooms open for booking at that specific moment, ensuring up-to-date availability.

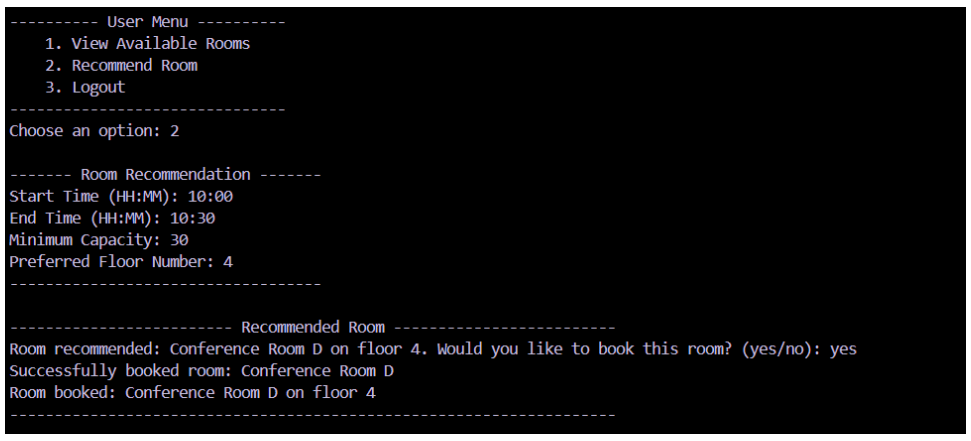


```
----------------- Available Rooms -----------------
Room ID    Name                Floor      Capacity
1          Conference Room A    1          20
2          Conference Room B    1          10
3          Conference Room C    1          30
4          Conference Room E    5          10
5          Conference Room F    5          30
6          Conference Room D    4          30
---------------------------------------------------
```

2. **Room Recommendation System**:
   - **User Input**: The system collects user preferences to provide personalized room suggestions:
     - **Time Slot**: The desired start and end times for the room.
     - **Minimum Capacity**: The minimum number of seats required.
     - **Preferred Floor**: The user's preferred floor for booking.
   - **Step-by-Step Process**:

1. **Filter by Time Slot**: The system first identifies rooms available within the specified time slot.
2. **Filter by Capacity**: It then checks for rooms that meet or exceed the required seating capacity.
3. **Preferred Floor Check**: If no rooms match the capacity requirement, the system suggests available rooms on the user's preferred floor.
4. **Room Prioritization**: For rooms meeting the capacity requirement, the system prioritizes those on the preferred floor.
5. **Nearby Floors**: If no rooms are available on the preferred floor, it recommends rooms on adjacent floors (above or below).
6. **Room Suggestions**: The user receives a list of recommended rooms based on their preferences.

- **Room Booking**: After reviewing recommendations, the user can select and book a room. The system then finalizes the booking and updates the room's availability status in real time.

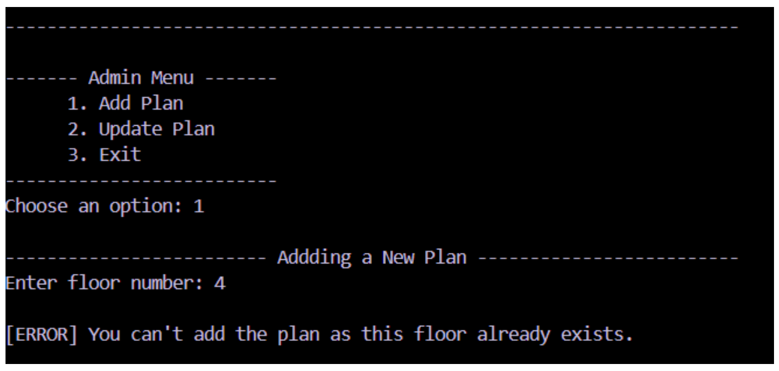


```
---------- User Menu ----------
    1. View Available Rooms
    2. Recommend Room
    3. Logout
-------------------------------
Choose an option: 2

------- Room Recommendation -------
Start Time (HH:MM): 10:00
End Time (HH:MM): 10:30
Minimum Capacity: 30
Preferred Floor Number: 4
-----------------------------------

------------------------- Recommended Room -------------------------
Room recommended: Conference Room D on floor 4. Would you like to book this room? (yes/no): yes
Successfully booked room: Conference Room D
Room booked: Conference Room D on floor 4
--------------------------------------------------------------------
```

4. Admin Features:

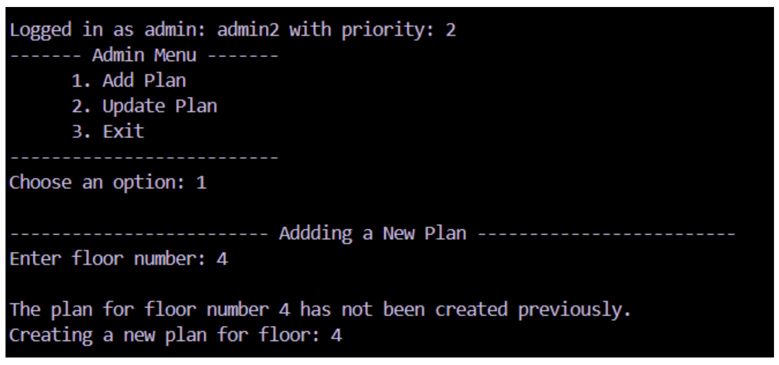
1. **Add Plan**:

- **Purpose**: Allows admins to add new floor plans and room details to the system.
- **Functionality**:
  - When an admin attempts to add a new floor plan, the system first checks if the specified floor number already exists in the database.
  - If the floor number is found, the admin is notified that the plan cannot be added.



```
--------------------------------------------------------------------

------- Admin Menu -------
      1. Add Plan
      2. Update Plan
      3. Exit
--------------------------
Choose an option: 1

------------------------ Addding a New Plan ------------------------
Enter floor number: 4

[ERROR] You can't add the plan as this floor already exists.
```

  - If the floor number doesn't exist, the admin is prompted to input floor details, such as the total number of rooms and their capacities. The system then successfully adds the new plan.

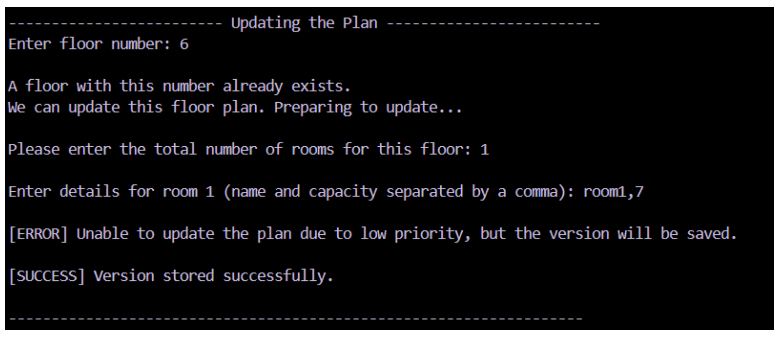


```
Logged in as admin: admin2 with priority: 2
------- Admin Menu -------
      1. Add Plan
      2. Update Plan
      3. Exit
--------------------------
Choose an option: 1

------------------------ Addding a New Plan ------------------------
Enter floor number: 4

The plan for floor number 4 has not been created previously.
Creating a new plan for floor: 4
```

2. **Update Plan**:

- **Purpose**: Enables admins to update existing floor plans within the system.
- **Functionality**:

- The system checks if the specified floor number exists in the database.
- If the floor number doesn't exist, the system creates and adds a new plan.
- If the floor number is found, the system compares the priority of the current plan's finalizer with the admin attempting the update.
- **Priority Check**:
    - If the current admin's priority is higher than the previous plan's finalizer, the system updates the existing plan.
    - If the current admin's priority is lower, the system saves the updated version without replacing the existing plan. Instead, it archives the lower-priority update for future reference.

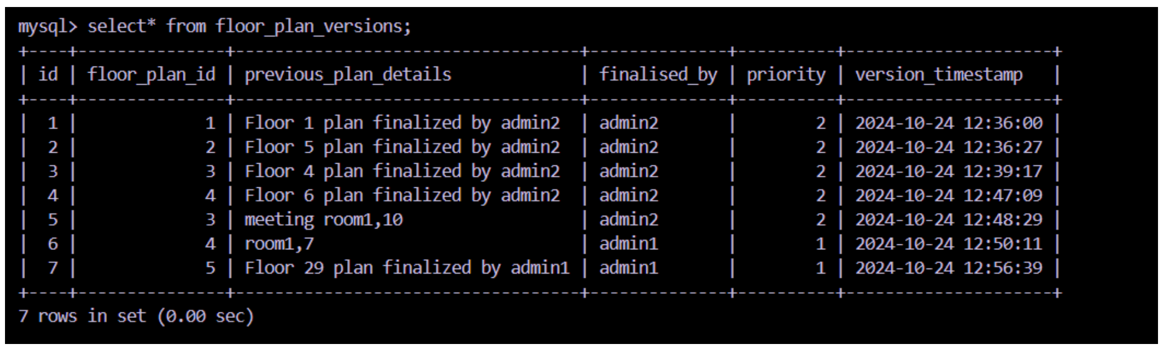


```
------------------------ Updating the Plan ------------------------
Enter floor number: 6

A floor with this number already exists.
We can update this floor plan. Preparing to update...

Please enter the total number of rooms for this floor: 1

Enter details for room 1 (name and capacity separated by a comma): room1,7

[ERROR] Unable to update the plan due to low priority, but the version will be saved.

[SUCCESS] Version stored successfully.

-------------------------------------------------------------------
```

- This process ensures that the highest-priority plan remains active while preserving lower-priority updates for accountability and potential future use.

5. Plan Versioning

1. **Plan Finalization**:
    - Each floor plan includes a "finalized by" field, recording the name and priority of the admin who finalized it.
    - When updating a plan, the system evaluates the priority of the admin making the change to determine if it should replace the existing plan.
2. **Plan Version History**:
    - If an admin with a lower priority than the current finalizer submits an update, the system preserves the existing plan.
    - The new version is saved in a version history table, enabling easy reference and retrieval. This approach ensures all updates are tracked and archived, maintaining a comprehensive history of changes for each floor plan.



```
mysql> select* from floor_plan_versions;
+----+---------------+------------------------------------+--------------+----------+---------------------+
| id | floor_plan_id | previous_plan_details              | finalised_by | priority | version_timestamp   |
+----+---------------+------------------------------------+--------------+----------+---------------------+
|  1 |             1 | Floor 1 plan finalized by admin2   | admin2       |        2 | 2024-10-24 12:36:00 |
|  2 |             2 | Floor 5 plan finalized by admin2   | admin2       |        2 | 2024-10-24 12:36:27 |
|  3 |             3 | Floor 4 plan finalized by admin2   | admin2       |        2 | 2024-10-24 12:39:17 |
|  4 |             4 | Floor 6 plan finalized by admin2   | admin2       |        2 | 2024-10-24 12:47:09 |
|  5 |             3 | meeting room1,10                   | admin2       |        2 | 2024-10-24 12:48:29 |
|  6 |             4 | room1,7                            | admin1       |        1 | 2024-10-24 12:50:11 |
|  7 |             5 | Floor 29 plan finalized by admin1  | admin1       |        1 | 2024-10-24 12:56:39 |
+----+---------------+------------------------------------+--------------+----------+---------------------+
7 rows in set (0.00 sec)
```

Performance Considerations

Efficient Room Recommendation Algorithm:

- The room recommendation system uses optimized data structures for quick and efficient user suggestions.
- **TreeMap for Nearby Floor Selection**:
    - A TreeMap manages rooms by floor numbers, keeping keys (floor numbers) sorted. This allows fast lookups for nearby floors when the preferred floor has no available rooms.

Optimized Queries for Room Recommendations:

- The system uses optimized SQL queries to quickly fetch room data based on availability, capacity, and floor preferences, reducing database load.
- **Step-by-Step Filtering**:

1. First, query rooms available within the specified time slot.
2. Then, filter for rooms meeting the required capacity.
3. If no rooms are found on the preferred floor, use the TreeMap to suggest rooms on nearby floors.

This streamlined approach, combining optimized data structures and queries, boosts the room recommendation system's performance and enhances user experience.

Error Handling

The system effectively manages common errors, ensuring a seamless user experience even in unexpected situations. Here are key areas where error handling has been implemented:

Database Connection Errors:

- **Scenario**: The system relies on a live MySQL database connection for data storage and retrieval. If unavailable due to network issues or database downtime, users may be unable to complete tasks.
- **Solution**:
    - A delayed connection retry mechanism attempts to reconnect every 10 milliseconds until successful.
    - During this delay, users (especially admins) can continue inputting data, which is temporarily stored locally. Once the connection is restored, the system syncs this data with the database.
- **Example**: If the database connection fails, admins can still add plan details, which are saved temporarily. Upon reconnection, the system syncs this data, ensuring no work is lost.

Invalid User Inputs:

- **Scenario**: Users or admins may enter incorrect or out-of-bounds values (e.g., negative room capacities).
- **Solution**:
    - The system implements validation checks at critical points. For instance, when adding a new plan, it verifies floor numbers and room capacities.
    - For regular users, the system validates time slots and room capacities before processing recommendations or bookings.
- **Example**: If an admin enters a negative room capacity, the system immediately rejects it and prompts for a valid input.

Data Synchronization Errors:

- **Scenario**: The system may fail to sync locally stored data due to connection issues, potentially leading to incomplete database uploads.
- **Solution**:
    - The system monitors whether all locally stored data has been successfully synced. If syncing fails, it either retries or notifies the admin of incomplete data upload.
- **Example**: If plan data fails to sync after connection restoration, the system retries until successful. For persistent failures, it notifies the admin to investigate the underlying issue.

These error-handling mechanisms are crucial for maintaining system reliability and user trust, enabling continued operations even in challenging circumstances.

Future Enhancements

1. **Password Hashing**:
    - **Implementation**: Future versions will incorporate password hashing using a secure algorithm like bcrypt.
    - **Benefits**: This enhancement will protect user credentials and reduce data breach risks by ensuring passwords aren't stored in plain text.
2. **Graphical User Interface (GUI)**:
    - **Development**: A GUI will replace the current console-based interface.

- **Benefits**: This will greatly improve user experience, offering a more intuitive and visually appealing way to interact with the system, and easing navigation and task execution.

These enhancements will bolster security and usability, resulting in a more robust and user-friendly application.