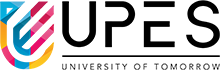
**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

DEHRADUN, UK-IN



**SCHOOL OF COMPUTER SCIENCE**

OBJECT ORIENTED PROGRAMMING SYSTEM LAB (CSEG2120\_1)

**OOPS PROJECT REPORT**

### ****Submitted To:****

**Dr. Suvojit Dhara**  
Assistant Professor  
School of Computer Science  
University of Petroleum and Energy Studies (UPES)

### ****Submitted By:****

**Parth Garg**  
B.Tech CSE (Cyber Security)   
Section: B-08 (4th Semester)  
SAP ID: 500122496

**Manu Randheer Yadav**  
B.Tech CSE (Cyber Security)  
Section: B-08 (4th Semester)  
SAP ID: 500125477

**Task Management system**

## ****1. Objective****

The primary objective of this project is to design and implement a **Task Management System** that empowers users to efficiently organize and manage their day-to-day tasks through a user-friendly graphical interface. The system is built to support a range of task-related operations including the ability to **add**, **edit**, **delete**, **view**, and **mark tasks as completed**. Each task includes relevant details such as a title, description, due date, and current status, enabling users to keep track of deadlines and progress.

## ****2 Technologies Used****

| Technology | Purpose |
| --- | --- |
| **Java Swing** | GUI development |
| **JDBC** | Database connectivity |
| **SQLite** | Lightweight relational database |
| **Maven** | Project management and dependency handling |
| **Git & GitHub** | Version control and collaborative development |

## ****3. Features****

* **User Authentication** (Login/Signup)
* **Add Task** – Add new task with title, description, and due date
* **Edit Task** – Update existing task details
* **Delete Task** – Remove a task permanently
* **View Tasks** – List all tasks of the logged-in user
* **Mark as Completed** – Update the status of a task
* **Multi-user Support** – Each user sees only their tasks
* **Persistent Storage** – Tasks are stored in a database using JDB

## ****3. System Architecture****

The Task Management System follows a **modular 3-tier architecture**, with responsibilities separated across five core packages for clarity, scalability, and maintainability. The folder structure is as follows:

### 1. ****Presentation Layer****

* **Folder**: ui
* **Description**: Contains all Java Swing GUI components and user interface logic. This includes login screens, task dashboards, forms for adding/editing tasks, and other visual elements with which users interact.

### 2. ****Business Logic Layer****

* **Folder**: model and util
* **Description**:
  + model: Defines the core data structures (e.g., User, Task) that represent entities in the application.
  + util: Contains utility classes and helper methods that support core logic (e.g., date formatters, input validators, and status converters).

### 3. ****Data Access Layer****

* **Folder**: dao
* **Description**: Handles all database interactions using JDBC. This includes CRUD operations for users and tasks, such as inserting, updating, retrieving, and deleting records from the SQLite database.

### 4. ****Resources****

* **Folder**: resources
* **Description**: Stores external assets required by the application, such as the , icons, and any other static files.

## ****5. Tools & Environment****

* **IDE**: Visual Studio Code / IntelliJ IDEA
* **Build Tool**: Apache Maven
* **JDK Version**: Oracle JDK 21/24
* **OS**: Windows 11
* **Version Control**: Git
* **Repository Hosting**: GitHub

## ****6. Database Design****

The application uses **SQLite** as its backend database to store and manage data persistently. It consists of two main tables: users and tasks, with proper relationships and normalization.

### ****1. users****

This table stores the authentication credentials and unique identification for each user.

| Field | Type | Description |
| --- | --- | --- |
| user\_id | INTEGER | Primary key, auto-incremented unique ID for each user |
| username | TEXT | Unique username chosen by the user |
| password\_hash | TEXT | Secure hash of the user's password |

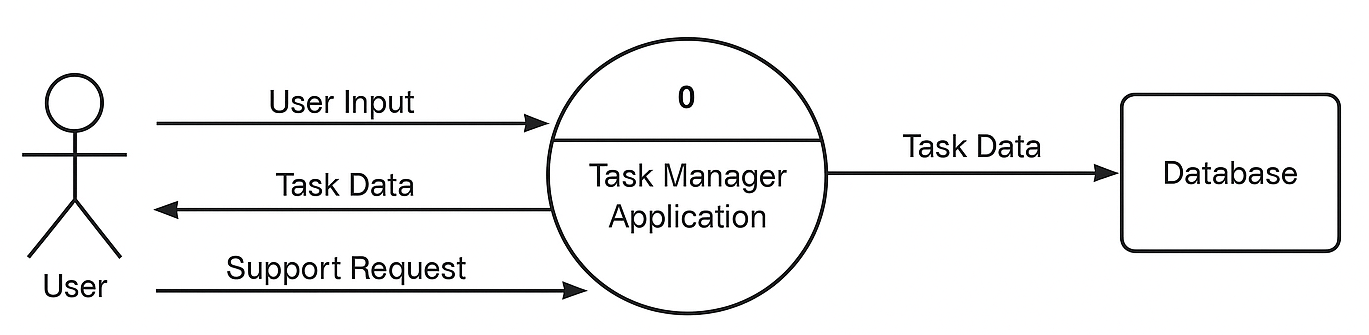
### ****2. tasks****

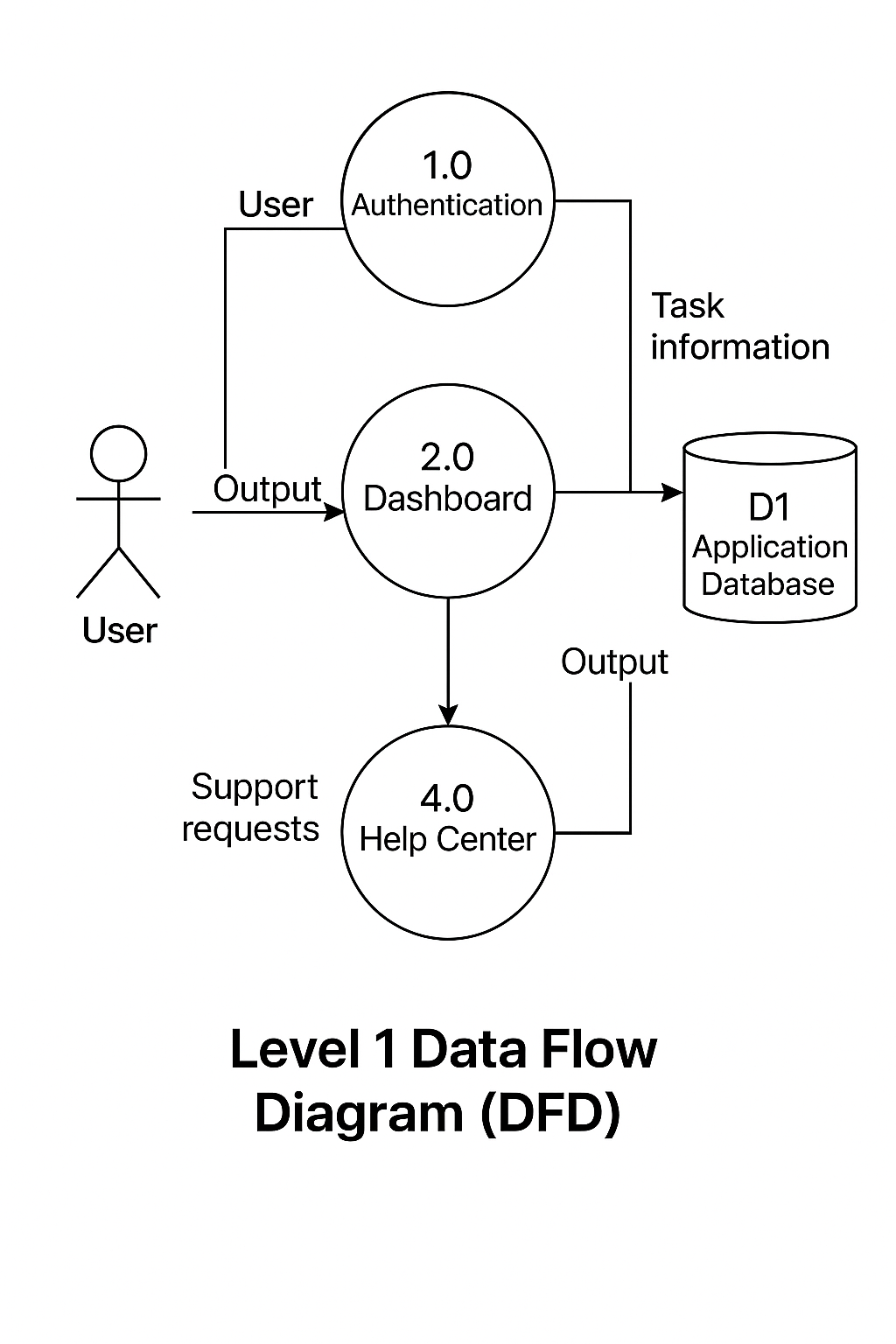
This table stores all the tasks created by users. Each task is linked to a specific user via user\_id.

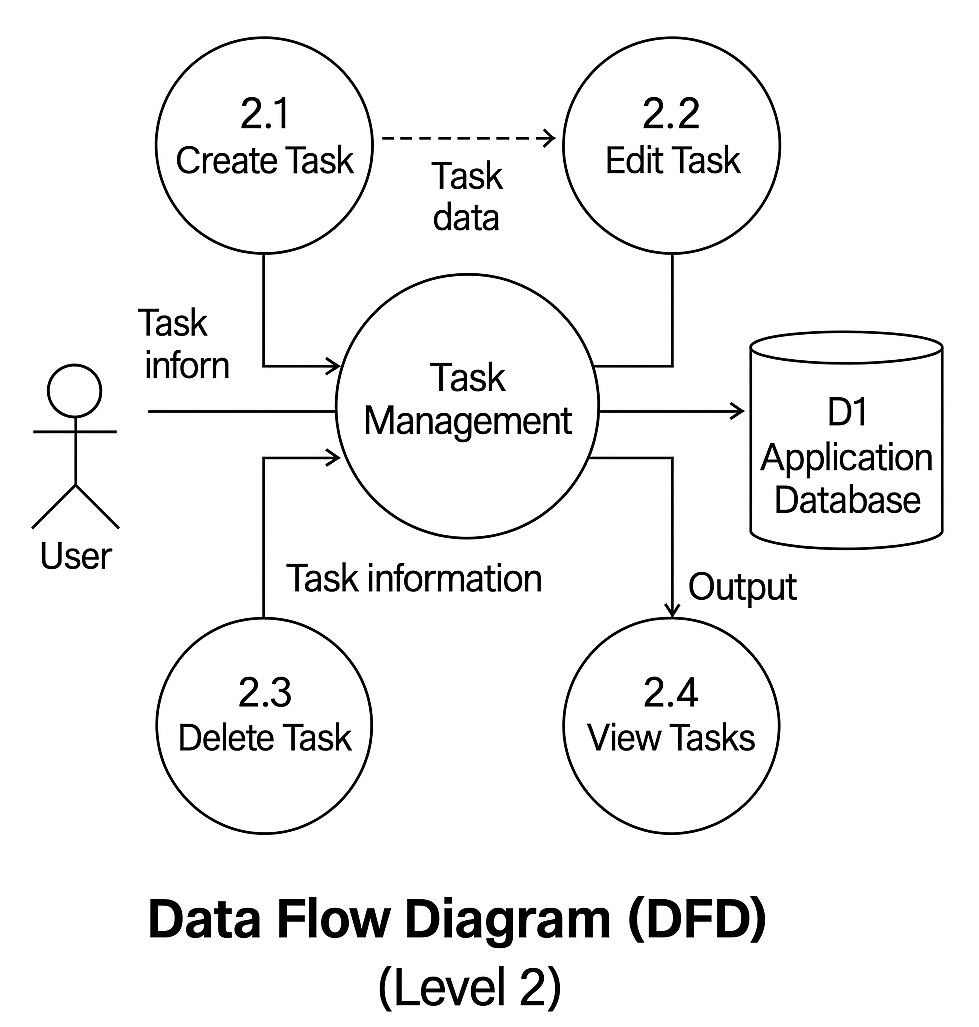
| Field | Type | Description |
| --- | --- | --- |
| task\_id | INTEGER | Primary key, auto-incremented task ID |
| user\_id | INTEGER | Foreign key referencing users(user\_id) |
| title | TEXT | Title or name of the task |
| description | TEXT | Detailed description of the task |
| due\_date | TEXT | Due date for task completion (stored as string) |
| status | TEXT | Current status (e.g., Pending, Completed) |
| priority | TEXT | Task priority (e.g., Low, Medium, High) |
| created\_at | TEXT | Timestamp when the task was created |
| updated\_at | TEXT | Timestamp of the most recent update |

This schema ensures efficient data organization, easy scalability, and proper user-task association. The created\_at and updated\_at fields help in tracking the lifecycle of each task, while priority adds a layer of task importance classification.

**7. ER diagrams**

 **DFD LEVEL 0**

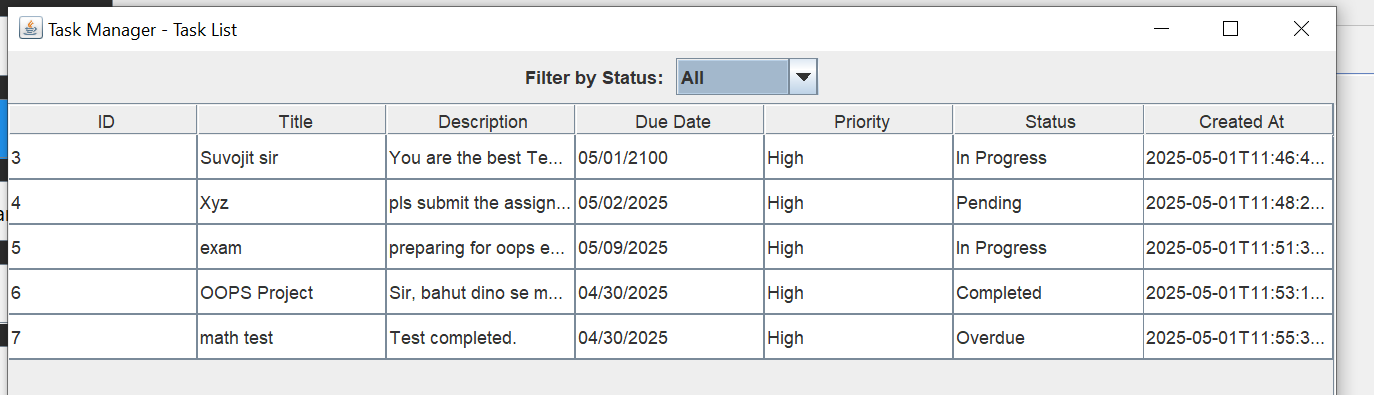
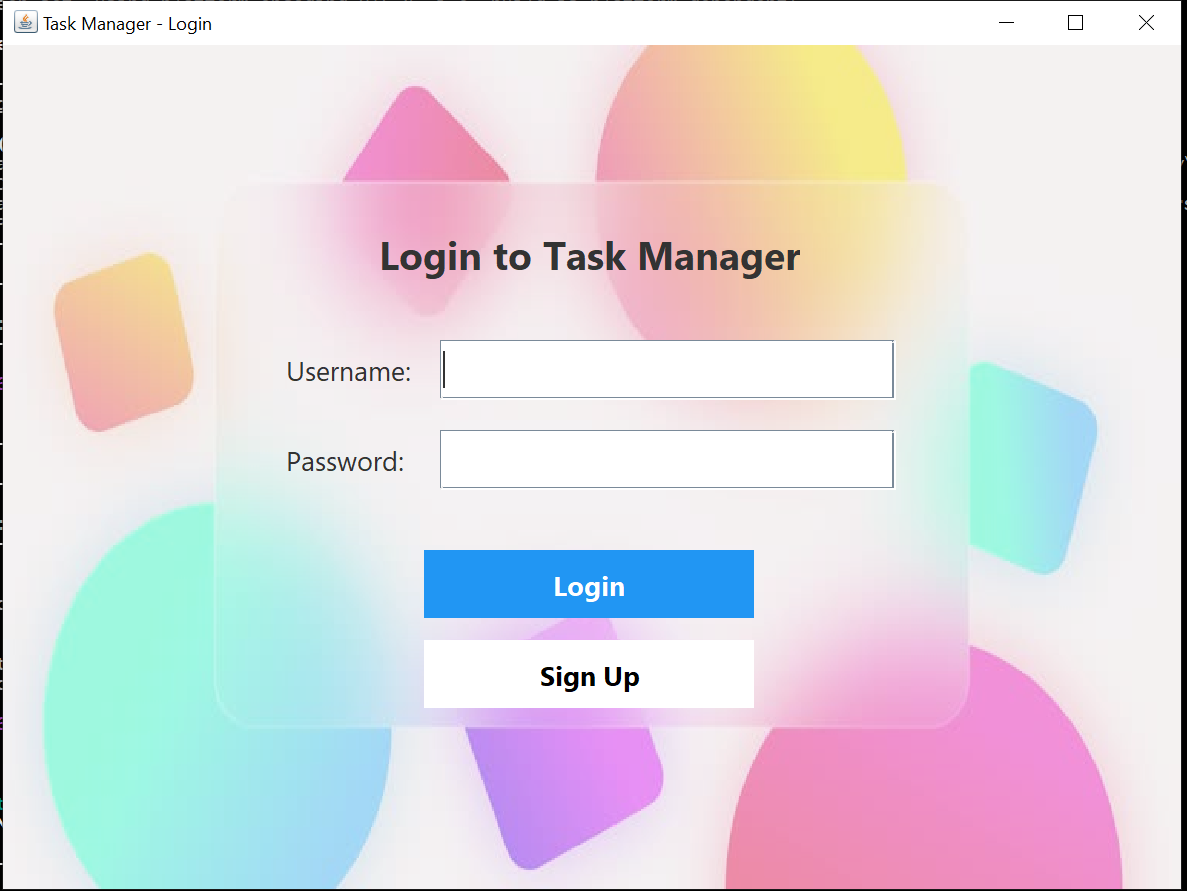
****

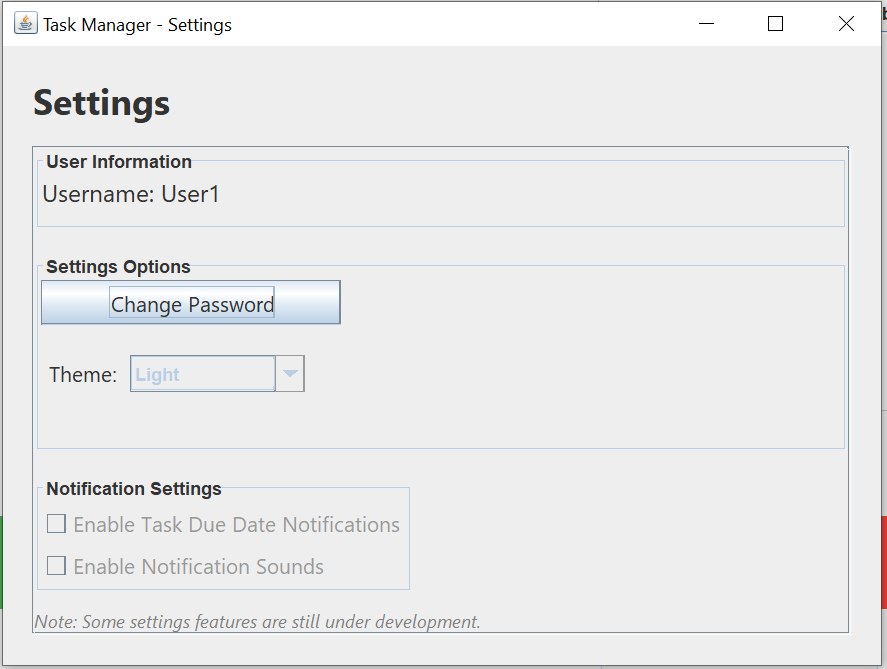


## ****8. Implementation Overview****

* The Task Management System is developed using a modular and scalable approach. The implementation integrates multiple technologies and libraries to create a robust and user-friendly application. Below is an overview of the key components involved in building the system:
* **Swing UI**:  
  The graphical user interface is built using **Java Swing**, which includes components like JFrame, JPanel, JTextField, JButton, JTable, and others. The interface is designed to be intuitive, enabling users to easily perform operations such as adding, updating, and managing their tasks. All UI classes are organized within the ui package.
* **JDBC Connectivity**:  
  Database interaction is facilitated using **Java Database Connectivity (JDBC)**. It allows the application to perform SQL operations like inserting, updating, deleting, and fetching records from the **SQLite** database. The JDBC logic is encapsulated within the dao package, ensuring separation from business logic and UI layers.
* **Maven**:  
  **Apache Maven** is used for project management and build automation. It helps in maintaining a clean project structure, managing external dependencies like sqlite-jdbc, and simplifying the build process. The pom.xml file is configured to include all required libraries and plugins.
* **Git & GitHub**:  
  Version control is managed using **Git**, with all commits and branches maintained on **GitHub**. This not only enables efficient tracking of code changes but also facilitates collaboration, code backup, and issue tracking during development.

## ****9. Sample Screenshots****

****



## ****10. Challenges Faced****

During the development of the Task Management System, several technical and architectural challenges were encountered. Tackling these issues provided valuable hands-on experience and deepened understanding of Java desktop application development. The key challenges included:

* **Integrating SQLite with JDBC in a Maven Project**  
  Setting up JDBC connectivity with **SQLite** inside a Maven-managed Java project required careful configuration of dependencies in the pom.xml file. Ensuring the correct SQLite JDBC driver was compatible with the JDK version and resolving Maven build issues was initially time-consuming.
* **Managing State Across Multiple Swing Components**  
  Coordinating the flow of data and user actions between multiple Swing windows and panels (such as Login, Dashboard, Add/Edit Task dialogs) posed challenges. It required a well-structured approach to maintain the state of the logged-in user, selected task, and current UI context.
* **Ensuring Data Integrity and UI Synchronization**  
  Keeping the **UI and database in sync** was critical. After any task operation (add/edit/delete), the table view needed to reflect real-time changes. This was managed through listener callbacks and refresh mechanisms, which had to be carefully designed to avoid bugs and data inconsistencies.
* **Designing a Responsive and User-Friendly GUI**  
  Although Swing is powerful, making the UI visually appealing and responsive across different resolutions was a challenge. Layout managers like GridBagLayout, BorderLayout, and manual component tuning were used to enhance usability.
* **Creating the Task Calendar**  
  Implementing a **calendar-style view** to visualize tasks by date added a layer of complexity. Mapping tasks to specific dates and handling overlaps, color-coding priorities, and integrating with the existing task list required custom logic and dynamic rendering using Swing components.
* **Password Hashing for Secure Authentication**  
  To protect user credentials, passwords were not stored in plain text. Instead, a **secure hashing algorithm** (like SHA-256) was used before saving passwords to the database. Implementing this required working with Java's cryptographic libraries and ensuring compatibility during login verification.

Overcoming these challenges significantly strengthened the robustness, security, and usability of the final application.

## ****11. Conclusion****

The **Task Management System** project offered a comprehensive and practical experience in building a desktop-based productivity application using core Java technologies. It successfully demonstrated the integration of **Java Swing** for GUI development, **JDBC** for database connectivity with **SQLite**, and the use of **Maven** for dependency management and project structure. Additionally, **Git and GitHub** were leveraged for version control and collaborative development.

This project not only reinforced foundational programming concepts but also highlighted the importance of clean architecture, modular design, and secure coding practices. By solving real-world challenges such as UI responsiveness, data synchronization, and user authentication, the project helped bridge the gap between academic knowledge and real-world application development.

## ****12. Future Enhancements****

To further improve functionality, user experience, and scalability, the following enhancements are proposed for future iterations of the system:

* **Implement Reminders and Notifications**  
  Introduce deadline reminders and real-time task alerts to help users stay on top of their schedules.
* **Add File Attachment Support to Tasks**  
  Enable users to upload and associate documents, images, or notes with individual tasks for better task context and documentation.
* **Enhance UI Using JavaFX or External Libraries**  
  Upgrade the visual design by transitioning from Java Swing to JavaFX or using third-party libraries like FlatLaf to create a more modern and responsive interface.
* **Add Data Encryption for Passwords and Sensitive Information**  
  Go beyond hashing by incorporating encryption mechanisms for storing and transmitting sensitive user data securely.
* **Build a REST API for Future Web/Mobile Integration**  
  Design and implement a backend REST API to allow the application to scale into a cross-platform solution, supporting mobile apps or web clients.