**Todo application**

Documentation

ACP (Advanced Computer Programming)

Submitted by

**Ali Hamza**

4289-FBAS/BSSE/F21(A)

Submitted to

**Sir Zohair Ahmed**



Faculty of Computing and Information Technology

INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD

Semester spring 2024

**TABLE OF CONTENTS**

Contents

[Chapter 1: Introduction 3](#_Toc168161328)

[Project Overview: 3](#_Toc168161329)

[Chapter 2: GUI Design 4](#_Toc168161330)

[Overview: 4](#_Toc168161331)

[Navigation Flow: 8](#_Toc168161332)

[Chapter 3: Database Design 10](#_Toc168161333)

[Table Descriptions: 10](#_Toc168161334)

[Sample Data: 11](#_Toc168161335)

[Chapter 4: Implementation 12](#_Toc168161336)

[Database Connection: 12](#_Toc168161337)

[CRUD operations 13](#_Toc168161338)

[Chapter 5: Summary 17](#_Toc168161339)

[Source Code 18](#_Toc168161340)

# Chapter 1: Introduction

## Project Overview:

The application developed is a comprehensive task management system designed to empower users in organizing and overseeing their tasks effectively. It serves as a centralized platform where users can seamlessly create, update, delete, and view tasks, thereby enhancing productivity and task management efficiency. The application caters to individuals, teams, or organizations seeking a user-friendly solution for managing their tasks.

* 1. **Key Features:**
* Task Creation: Users can effortlessly add new tasks with detailed information such as task name and status.
* Task Deletion: Unwanted or completed tasks can be easily removed from the system.
* Task Viewing: Users have the ability to browse through a comprehensive list of all tasks stored in the system.
* Task Updating: Existing tasks can be edited and updated to modify task details or status as needed.
  1. **Technologies Used:**

The development of the application involved the utilization of various technologies and tools, including:

* Java: The core programming language employed for application development.
* Swing: Java Swing library was utilized extensively to create the graphical user interface (GUI) components, ensuring a rich and interactive user experience.
* AWT (Abstract Window Toolkit): AWT components were integrated with Swing to enhance GUI development and provide additional functionality.
* JDBC (Java Database Connectivity): JDBC played a pivotal role in facilitating seamless interaction with the MySQL database, enabling efficient storage and retrieval of task data.
* MySQL: MySQL was selected as the relational database management system (RDBMS) for storing and managing task-related information, ensuring robust data management capabilities.

# Chapter 2: GUI Design

## Overview:

The graphical user interface (GUI) of the task management application was meticulously designed to prioritize user experience, ease of navigation, and aesthetic appeal. The design choices were made with careful consideration of modern design principles, ensuring a seamless and intuitive user interaction. Below are the key aspects of the GUI design:

* 1. **Layout and Organization:**
* The GUI features a clean and organized layout, with well-defined sections for different functionalities such as task insertion, deletion, and viewing.
* The main window layout follows a BorderLayout, providing distinct regions for the task table (center) and control buttons (south).
* Each functionality, including task insertion, deletion, and viewing, is represented by a dedicated button, enhancing accessibility and user-friendliness.
  1. **Design Patterns**

The design adheres to the Model-View-Controller (MVC) architectural pattern, ensuring a clear separation of concerns between the data model (task data), the presentation layer (GUI components), and the controller logic (event handling).

The use of the MVC pattern facilitates modular development, making it easier to maintain and extend the application in the future.

* 1. **User Experience Considerations**
* Emphasis on Simplicity: The GUI design prioritizes simplicity and clarity, avoiding clutter and unnecessary complexity. Users can quickly grasp the functionality and navigate through the application with minimal effort.
* Consistency: A consistent visual design and interaction patterns are maintained across all GUI components, promoting familiarity and reducing cognitive load.
* Feedback and Confirmation: Interactive elements such as buttons and dialogs provide informative feedback to users, ensuring they are aware of the actions they perform and any consequential changes.
* Error Handling: The GUI incorporates robust error handling mechanisms to gracefully handle unexpected user inputs or system errors, guiding users towards resolution without disruption.
  1. **Visual Aesthetics:**

The GUI employs a visually appealing color scheme and font selection, enhancing readability and aesthetics without compromising usability.

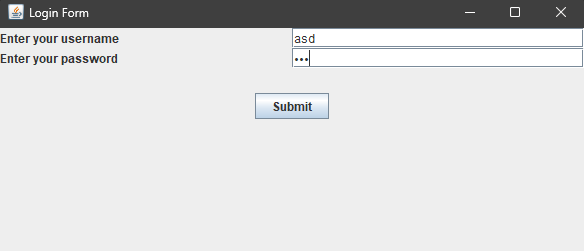
Proper alignment, spacing, and sizing of GUI elements contribute to a harmonious and polished appearance, instilling confidence in the application's quality and professionalism.

Overall, the GUI design aims to deliver an intuitive, efficient, and visually pleasing user experience, empowering users to manage their tasks seamlessly and effectively.

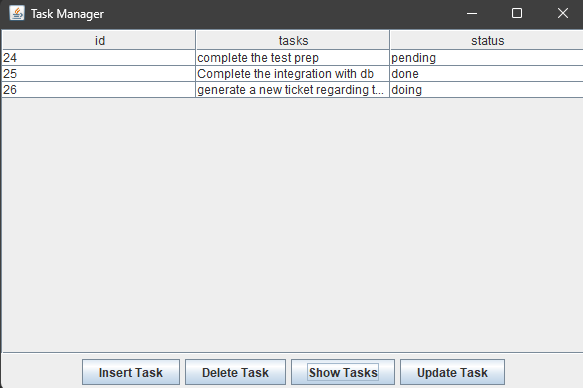
* 1. CRUD operation visuals

Following includes the Login and crud operation UI of the todo app.

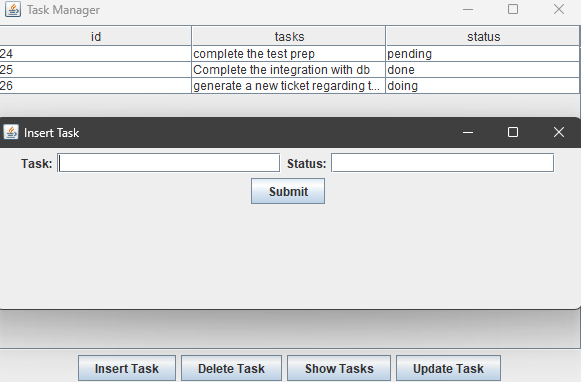
* **Login screen**



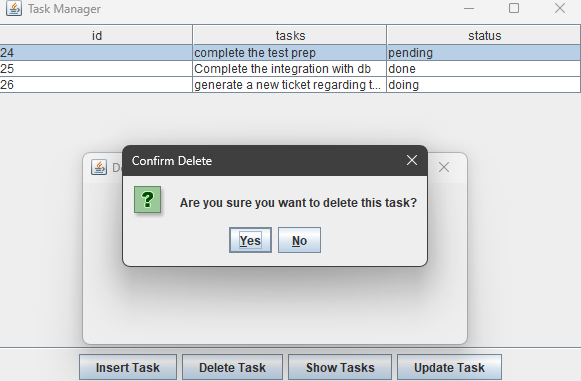
* **Task Viewing Screen**



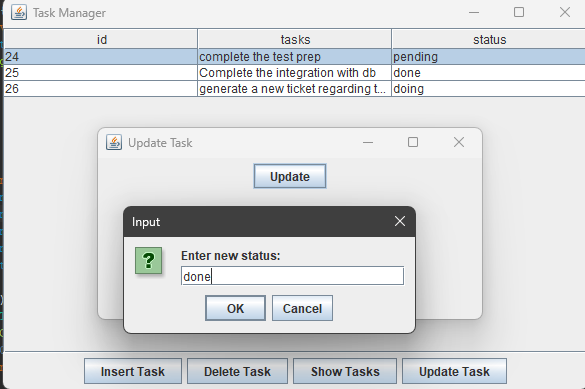
* **Task Insertion Screen**



* **Task Deletion Screen**



* **Task Update Screen**



## Navigation Flow:

The navigation flow within the task management application is designed to be intuitive and seamless, allowing users to effortlessly move between different screens and functionalities. Below is an overview of the navigation flow:

**Main Interface:**

Upon launching the application, users are presented with the main interface, which serves as the central hub for accessing various functionalities.

The main interface prominently displays a table listing all existing tasks, providing users with an overview of their current tasks at a glance.

**Task Management Functionality:**

* **Insert Task:** Users can add new tasks by clicking the "Insert Task" button, which opens a dialog or window for entering task details such as task name and status. After submitting the task, it is immediately added to the task list.
* **Delete Task:** To remove a task, users can either select a task from the table and click the "Delete Task" button or directly click on a task row to initiate deletion. A confirmation dialog prompts users to confirm the deletion before proceeding.
* **Show Tasks:** Clicking the "Show Tasks" button updates the task table with the latest task data retrieved from the database, ensuring users have access to up-to-date information.
* **Update Tasks:** Clicking the “Update Task button” shows you 2 dialogues in a row, one for updating the task and the other for updating the status.

**Interactivity:**

Mouse Interaction: Users can interact with the task table using mouse clicks. Single-clicking on a task row initiates actions such as deletion or updating, depending on the selected functionality.

Button Clicks: Dedicated buttons for each functionality provide users with an alternative method for performing tasks, offering flexibility based on user preference.

**Feedback and Confirmation:**

Throughout the navigation flow, the application provides feedback and confirmation to users to ensure clarity and prevent accidental actions. Dialog boxes and messages notify users of successful operations, errors, or pending confirmations.

**Seamless Transition:**

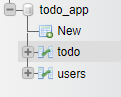
The navigation flow is designed to be seamless, allowing users to transition between different screens and functionalities without interruption or loss of context. The consistent layout and interaction patterns further enhance the user experience.

By prioritizing intuitive navigation and clear feedback mechanisms, the task management application aims to streamline user interactions and enhance productivity in managing tasks effectively.

# Chapter 3: Database Design

## Table Descriptions:

1. **DB overview:**



1. **`users` Table:**



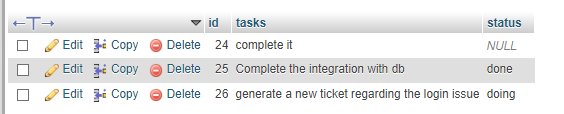
**Purpose:** This table stores user information for authentication and authorization purposes.

**Columns:**

* `username`: VARCHAR(50), NOT NULL, Unique - Stores the username for user login.
* `Password`: VARCHAR (255), NOT NULL - Stores the hashed password for user login.

1. **`todo` Table:**

**Purpose:** This table stores the tasks created by users, including task details and status.



**Columns:**

* **id:** INT, Primary Key, Auto-increment - Unique identifier for each task.
* **tasks:** VARCHAR(255), NOT NULL - Stores the description of the task.
* **status:** VARCHAR(50), NOT NULL - Stores the current status of the task (e.g., "Pending", "Completed").

## Sample Data:

1. **users Table:**

**Explanation**

* The **username** column stores the unique username for each user.
* The **password** column stores the hashed password for security.

1. **todo Table:**

**Explanation**

* The `id` column uniquely identifies each task.
* The `tasks` column contains the description of the task.
* The `status` column indicates the current status of the task, helping users track their progress.

By organizing data into these tables, the application ensures efficient storage and retrieval of user and task information, facilitating seamless task management and user authentication processes.

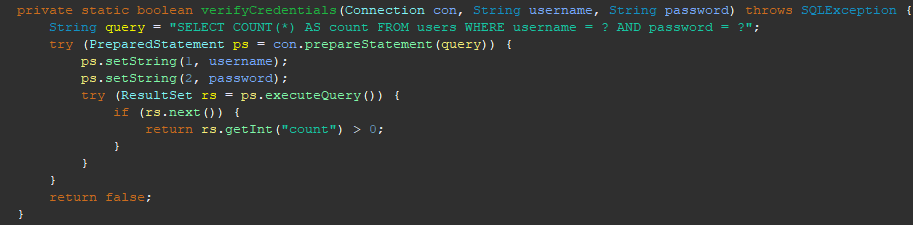
# Chapter 4: Implementation

## Database Connection:



**Explanation:** This action listener method is used to establish a connection to the MySQL database. The getConnection method returns a Connection object, which is used in other parts of the application for executing SQL queries.

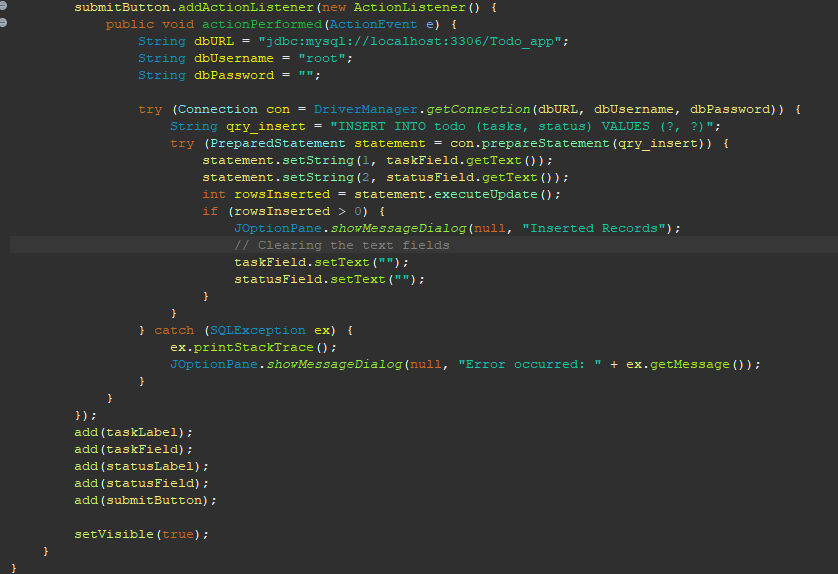
**User Authentication:**

****

**Explanation:** This method checks if the provided username and password match any record in the users table. It uses a PreparedStatement to prevent SQL injection. The method returns true if the credentials are valid, and false otherwise.

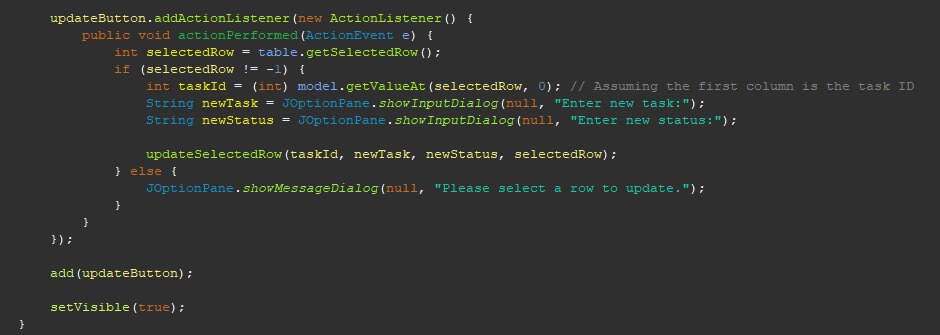
## CRUD operations

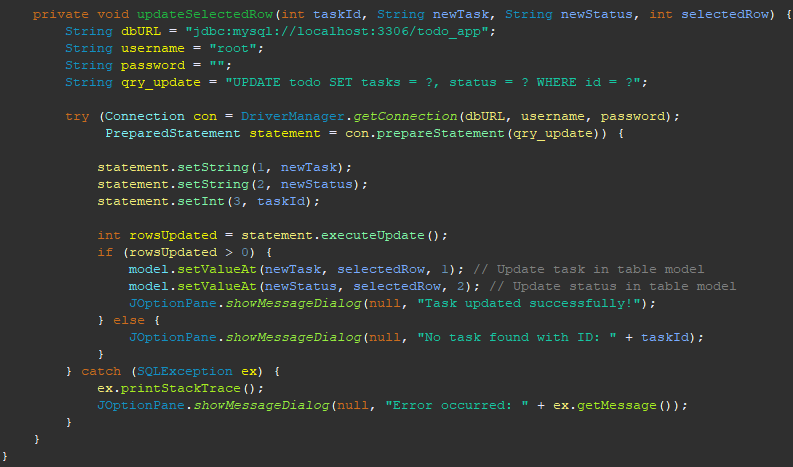
**Insert Task:**

****

**Explanation:** This method inserts a new task into the todo table. It takes the task description and status as parameters and executes an INSERT SQL query using a PreparedStatement.

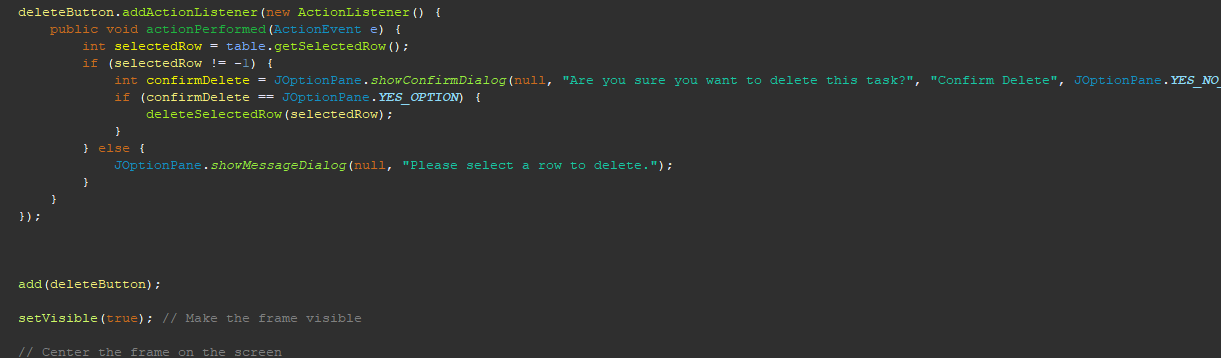
**Update Task:**

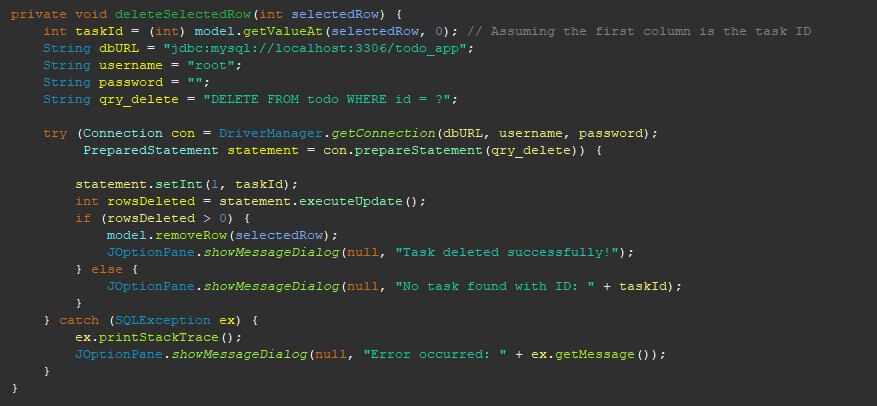
****

****

**Explanation:** This method updates an existing task in the todo table. It takes the task ID, updated task description, and updated status as parameters and executes an UPDATE SQL query.

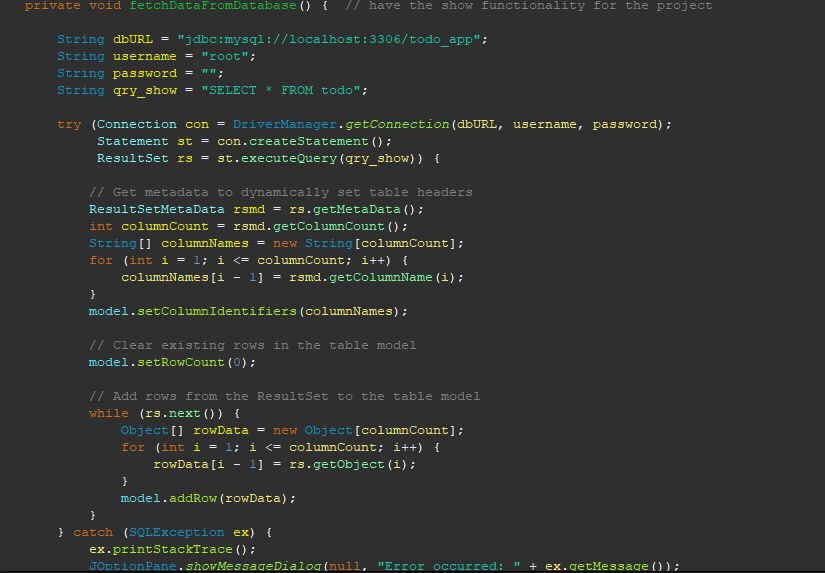
**Delete Task:**

****

****

**Explanation:** This method deletes a task from the todo table based on the task ID. It uses a PreparedStatement to execute the DELETE SQL query, ensuring that only the specified task is removed from the database.

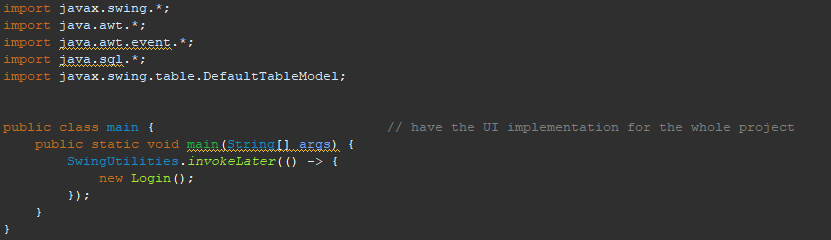
**Show Task:**

****

**Explanation:** This method fetches the todo table from the database using the SELECT query

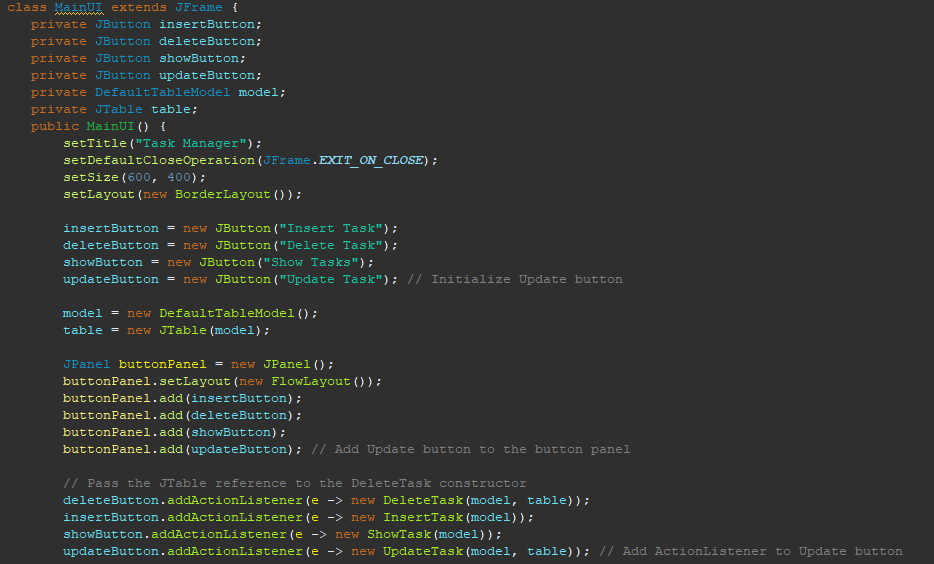
Then stores it in the result set to show it in the table.

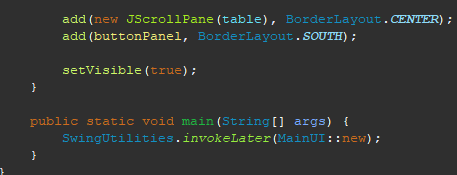
**Main (runner) class:**

****

**Explanation:** The main class just initiates the Login class, and then in the login class after authentication the user is sent to MainUI.

**MainUI class:**

****

****

**Explanation:** The MainUI class is just implementing the ui for the whole app, so that whatever event happens it calls the suitable classes for those button.

# Chapter 5: Summary

The project is a task management application designed to help users organize their tasks effectively. It allows users to perform CRUD (Create, Read, Update, Delete) operations on their tasks, including inserting new tasks, viewing existing tasks, updating task details, and deleting tasks. The application provides a user-friendly graphical interface built using Java Swing and AWT.

**Lessons Learned:**

During the development of the project, I gained valuable experience in GUI design using Java Swing and AWT. I learned how to create responsive and intuitive user interfaces, handle user interactions, and manage data using JDBC for database connectivity. Additionally, I enhanced my skills in database design, including defining table structures, establishing relationships between tables, and implementing CRUD operations.

**Future Improvements:**

For future improvements, several enhancements could be considered to further enhance the functionality and user experience of the application. Some potential improvements include:

1. Implementing user authentication and authorization to ensure secure access to the application.

2. Adding support for task categorization and prioritization to help users better organize their tasks.

3. Implementing reminders and notifications for upcoming tasks to help users stay on track.

4. Enhancing the GUI with modern design patterns and visual elements to improve usability and aesthetics.

5. Implementing data synchronization across multiple devices to enable users to access their tasks from anywhere.

6. Optimizing database queries and improving performance for handling large datasets.

7. Adding support for exporting tasks to different file formats for backup and sharing purposes.

**Conclusion:**

In conclusion, the task management application project provided a valuable learning experience and an opportunity to apply various programming concepts and techniques. By developing this application, I gained practical experience in GUI development, database management, and software engineering principles. I look forward to applying these skills in future projects and continuing to explore new technologies and development practices.

# Source Code

Following is the link to my git repository that has the source code for my project

https://github.com/hamzadoescoding/TodoApp