RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

DAILY TASK MANAGER

A MINI PROJECT REPORT

Submitted by

KARTHICK S	**231801079**
KABEL B	**231801076**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 2025

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME……………………………………………………………………………..

ACADEMIC YEAR……………SEMESTER…………BRANCH ………

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled "**Daily Task Manager**" in the subject **CS23333 – OBJECT ORIENTED PROGRAMMING USING JAVA** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on ---------------------

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

Initially I thank the Almighty for being with us through every walk of my life and showering his blessings through the endeavor to put forth this report. My sincere thanks to our Chairman Mr. S. MEGANATHAN, M.E., F.I.E., and our Chairperson Dr. (Mrs.)THANGAM MEGANATHAN, M.E., Ph.D., for providing me with the requisite infrastructure and sincere endeavoring educating me in their premier institution.

My sincere thanks to Dr.S.N. MURUGESAN, M.E., Ph.D., our beloved Principal for his kind support and facilities provided to complete our work in time.

I express my sincere thanks to Dr. J M GNANASEKAR M.E.,Ph.D., Head of the Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. I convey my sincere and deepest gratitude to our internal guide, Ms. Y. RENUKA DEVI, ., Assistant Professor, Department of Artificial Intelligence and Data Science, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally I express my gratitude to my parents and classmates for their moral support and valuable suggestions during the course of the project.

TABLE OF CONTENTS

ABSTRACT

This project features a To-Do List application developed using Java, providing an efficient and interactive way to manage tasks. The system utilizes a MySQL database to store and retrieve task details such as task name, due date, priority, description, and status. Users can add, delete, and update tasks, with real-time changes reflected in the interface. The system also includes a filtering feature to sort tasks based on their status (Pending or Completed), making it easy for users to track progress.

The application provides a user-friendly graphical interface built with Java Swing, allowing seamless task management. With the integration of SQL queries for data handling and a responsive design, this To-Do List application ensures an intuitive experience for users to organize and prioritize their daily tasks.

Introduction

1.1 PROJECT OVERVIEW

Managing daily tasks effectively is a common challenge for individuals and organizations. Traditional methods like paper planners and sticky notes are increasingly replaced by digital solutions. However, many existing tools are either overly complex or lack essential functionality.
The To-Do List Management System provides a balanced solution with a Java Swing-based GUI and MySQL integration via JDBC. It allows users to manage tasks by creating, updating, deleting, and filtering them while securely storing data for easy retrieval. Designed for simplicity and offline functionality, it suits students, professionals, and small teams, offering a robust platform without requiring internet access or subscriptions.

1.2 OBJECTIVES AND GOALS

1. **Task Management:** Create a system that simplifies task creation, updating, and deletion through an intuitive interface.
2. **Data Security:** Ensure tasks are securely stored and retrievable using MySQL.
3. **Efficiency:** Design a lightweight application that works seamlessly offline.
4. **Scalability:** Make the system adaptable for future enhancements, such as cloud integration or mobile compatibility.

1.3 NEED FOR THE PROPOSED SYSTEM

Managing tasks effectively in today's fast-paced world requires a system that balances simplicity and functionality. Existing solutions often fail to address user needs for ease of use and offline accessibility.
The proposed To-Do List Management System fulfills this gap by offering a feature-rich platform designed for task organization. By integrating with a robust database system like MySQL and using Java Swing for the interface, it ensures secure and efficient task management. Additionally, its offline functionality and open-source nature make it a reliable and cost-effective solution for users.

1.4 APPLICATIONS OF PROPOSED SYSTEM

1) **Personal Task Management:** Helps individuals organize and prioritize daily tasks.
2) **Professional Use:** Assists small teams in tracking project milestones and deadlines.
3) **Educational Applications:** Provides students with a tool for managing assignments and exam schedules.
4) **Scalable Deployment:** Can be integrated into larger project management systems with minimal modifications.
5) **Offline Accessibility:** Serves users without requiring continuous internet access.

1.5 SCOPE OF THE PROJECT

The scope of the To-Do List Management System encompasses creating a functional and user-friendly task management tool tailored for a broad audience. The project aims to provide:

1. **Core Features:**
    - Task creation, updating, deletion, and filtering.
    - Data security through robust MySQL integration.
    - Status tracking with easy categorization (e.g., Pending, Completed).
2. **Offline Functionality:**
    - Operates independently of an internet connection, ensuring accessibility for all users.
3. **User Adaptability:**
    - Designed for students, professionals, and small teams to suit diverse needs.
    - Modular architecture that supports future enhancements, such as cloud sync and mobile app integration.
4. **Scalability:**
    - Capable of managing extensive task databases without compromising performance.
5. **Ease of Use:**
    - Intuitive Java Swing-based GUI for straightforward navigation and operation.

LITERATURE REVIEW

To-do list applications have evolved over time to offer a range of functionalities, from basic task management to more advanced features. Initially, to-do lists focused on simple task tracking, allowing users to add tasks with due dates, priorities, and descriptions. Over time, more sophisticated features were added, including task categorization, recurring tasks, and notifications, providing better organization and time management. Today, collaborative task management has become a significant trend, allowing users to share tasks and collaborate in real time, making these applications more versatile and suitable for both individual and team use.

2.1 TYPES OF TASK MANAGEMENT SYSTEMS:

1. Manual Task Management:
   Traditional methods like diaries, sticky notes, or whiteboards allow users to track tasks. While simple and effective for small-scale use, these methods lack scalability, data security, and accessibility.
2. **Basic Digital Tools:**
   Applications like spreadsheets or simple text editors offer minimal functionality for organizing and prioritizing tasks. However, they lack automation, user interaction features, and real-time collaboration capabilities.
3. **Advanced Task Management Systems:**
   These include software solutions like Trello, Asana, and Microsoft To-Do that use graphical interfaces, cloud synchronization, and API integrations. While highly functional, such systems can be complex or require paid subscriptions, limiting accessibility for some users.

2.2 EXISTING SYSTEMS:

1. **Standalone Systems:**
   Applications like Microsoft To-Do and Google Keep focus on task organization, providing basic to-do list functionalities such as reminders and task prioritization. However, they often rely on cloud services, making offline usage difficult.
2. **Database-Integrated Solutions:**
   Systems integrated with relational databases like MySQL or SQLite offer greater

flexibility and security for storing and retrieving tasks. They ensure data is consistently managed and easily accessible.

3. **GUI-Based Management Tools:**
   Tools built using frameworks like Java Swing or Python Tkinter allow for custom offline solutions, combining intuitive interfaces with robust database support.

2.3 WHAT WE DID:

The proposed system combines the strengths of standalone tools and database-driven solutions by offering:

1. **Java Swing Interface:**
   - A user-friendly GUI that simplifies task creation, updating, filtering, and deletion.
2. **JDBC Integration with MySQL:**
   - Secure, efficient, and scalable storage for tasks, enabling easy retrieval and management.
3. **Offline Functionality:**
   - Designed to function without internet connectivity, unlike many commercial task management applications.
4. **Real-Time Updates:**
   - Users can update statuses, filter tasks based on criteria like "Pending" or "Completed," and immediately view changes in the interface.
5. **Custom Features:**
   - Priority setting (1-5 scale), task descriptions, and deadline tracking ensure comprehensive task management.

2.4 KEY FINDINGS AND GAPS:

1. **Strengths of Existing Systems:**
- GUI-based tools with database integration address user needs for simplicity and functionality.
- Advanced systems incorporate cloud synchronization for multi-platform accessibility.
2. **Identified Gaps:**
- Many tools require internet access, limiting usability in offline environments.
- Paid subscription models exclude users seeking open-source or free alternatives.
3. **Addressed Gaps:**

- The proposed system ensures offline functionality and open-source availability.
- Simplified GUI design provides core functionalities without overwhelming users.

2.5 APPLICATIONS IN THE INDUSTRY

1. **For Individuals:**
   Simplifies personal task management, helping users organize their daily schedules, deadlines, and priorities.
2. **For Small Teams or Startups:**
   Offers a collaborative tool for managing team tasks without requiring a complex setup.
3. **Educational Use:**
   Ideal for students and teachers to track assignments, project deadlines, and academic goals.
4. **Workforce Productivity:**
   Enhances employee efficiency by providing a structured platform for organizing tasks and priorities.
5. **Scalability for Future Integration:**
   The system's modular architecture can be expanded for multi-user support, mobile applications, or cloud synchronization.

PROBLEM FORMULATION

3.1 MAIN OBJECTIVE

The primary objective of this project is to develop a task management system that enables users to efficiently create, update, and manage tasks. This system will use a user-friendly Java Swing-based interface combined with a robust backend powered by MySQL through JDBC. The system should allow users to manage their tasks offline, ensuring a smooth user experience without the need for internet connectivity. Additionally, it should allow filtering tasks based on their status (e.g., Pending, Completed) and provide the option to assign priority levels to tasks.

3.2 SPECIFIC OBJECTIVE

1. **User-Friendly Interface:**
   To create a simple, intuitive interface using Java Swing that enables users to create, update, and delete tasks efficiently.
2. **Task Management Features:**
   To implement essential features such as task categorization (Pending, Completed), setting task priority (1-5 scale), and assigning deadlines.
3. **Database Integration:**
   To integrate MySQL using JDBC for secure, efficient task storage and retrieval, ensuring all task data is persistently saved and easily retrievable.
4. **Offline Functionality:**
   To ensure the system is fully operational offline, providing users with full control over their data without requiring internet access.
5. **Responsive User Experience:**
   To provide a responsive system that functions smoothly on different devices and platforms, offering a seamless user experience.

3.3 METHODOLOGY

1. **Database Setup:**
   The first step involves setting up a MySQL database to store task data. The database will contain tables for storing tasks, including fields such as task name, description, priority, and due date.

2. **GUI Development:**
   A Java Swing-based graphical user interface (GUI) is developed to allow users to interact with the task management system. The interface will include components like buttons, text fields, checkboxes, and tables for viewing tasks.
3. **Backend Development:**
   The backend is built using JDBC for seamless integration between the Java application and the MySQL database. This allows for task creation, updating, deletion, and filtering, while ensuring data consistency and security.
4. **Offline Functionality:**
   The system will not rely on internet connectivity. All data will be stored locally in the MySQL database, ensuring full functionality without requiring online access.
5. **Testing and Evaluation:**
   Once the system is developed, it will undergo rigorous testing to ensure its functionality. This includes testing for proper task storage, user interaction, and system stability.

3.4 PLATFORM

1. **Java Swing:**
   A lightweight Java GUI framework used to develop the user interface. Java Swing offers the flexibility to design interactive components while keeping the interface simple and responsive.
2. **JDBC and MySQL:**
   The MySQL database is used for persistent storage of tasks, while JDBC facilitates the connection between the Java application and the MySQL database for task data management.
3. **Offline Operation:**
   Since the system is designed to function offline, no internet connectivity is required for users to interact with the system. Data is stored locally and can be accessed anytime.
4. **Platform Independence:**
   The system can be run on any machine with Java and MySQL installed, making it cross-platform and accessible on both Windows and Linux systems. The system’s modular design allows for future expansion, including mobile application integration if desired.

SYSTEM ANALYSIS AND DESIGN

4.1 FACT FINDING

The fact-finding process for the Task Management System involved determining the user requirements and expectations for task creation, task tracking, and task prioritization. Users needed an intuitive system where they could efficiently create, update, and delete tasks with minimal friction. Additionally, they required features to categorize tasks based on completion status (e.g., Pending, Completed) and set priorities.
The system's design was based on feedback from potential users such as students, professionals, and small teams. User expectations included a user-friendly graphical interface, offline functionality, and secure data storage. MySQL was selected as the database to ensure the system's scalability and reliability, while JDBC provided seamless connectivity between the Java application and the database.

4.2 FEASIBILITY ANALYSIS

1. **Technical Feasibility:**
   The project leverages Java Swing for the graphical interface, which is a mature and widely-used GUI toolkit for Java applications. JDBC and MySQL are integrated for backend functionality, providing robust and efficient task management. The architecture is simple, ensuring that the system can be developed with available resources and expertise.
2. **Economic Feasibility:**
   The system uses open-source tools such as Java, Swing, MySQL, and JDBC, making it cost-effective. No licensing costs are involved, and the system can be deployed on any standard machine, reducing operational expenses. MySQL's open-source nature and JDBC's compatibility with various databases further contribute to its economic feasibility.
3. **Operational Feasibility:**
   The system is designed to operate offline, which ensures that users do not require an internet connection to manage tasks. It is also lightweight and can run on standard desktop computers without demanding significant hardware resources. Java Swing and MySQL are both cross-platform, making the system accessible to a wide range of users across different operating systems.

SYSTEM DESIGN AND ARCHITECTURE

5.1 PROJECT WORKFLOW

1. **User Input:**
   The user enters task details such as task name, description, priority, and due date into the interface.
2. **Task Creation and Update:**
   The system stores the task information in the MySQL database using JDBC. Users can edit task details, change task priority, or mark a task as completed.
3. **Task Filtering:**
   The system allows users to filter tasks based on their status (Pending, Completed) or priority.
4. **Task Management:**
   Users can delete tasks or update existing ones via the user interface. The backend ensures that all modifications are reflected in the database.
5. **Data Storage and Retrieval:**
   All task data is stored in a MySQL database, with secure retrieval and storage processes facilitated by JDBC. Users can access their tasks in real-time, and data is persistently stored offline.

5.2 TOOLS AND TECHNOLOGIES USED

1. **Java:** The core programming language used to build the task management system. Java provides the necessary tools and features for developing a scalable and reliable desktop application.
2. **Java Swing:**Java Swing is used for creating the graphical user interface (GUI) of the application. It allows the design of buttons, text fields, tables, and other components that users interact with.
3. **JDBC (Java Database Connectivity):** JDBC is used to connect the Java application to the MySQL database. It ensures smooth communication between the front end and the backend, allowing task data to be stored and retrieved efficiently.
4. **MySQL Database:** MySQL is chosen for storing task information such as task name, description, priority, and status. It is a reliable, scalable, and open-source relational database management syste**m.**

5.3 TASK MANAGEMENT ALGORITHMS

5.3.1 TASK CREATION AND UPDATE

The system supports adding new tasks and updating existing ones using the following approach:

1. **Create Task:**
   Users input details such as the task name, description, priority, and deadline. The data is then validated and stored in the MySQL database through JDBC.
2. **Update Task:**
   If a user wishes to update a task, they can select the task from the list, modify the required fields (e.g., priority or due date), and save the changes. These changes are reflected in the MySQL database.
3. **Delete Task:**
   The user can select a task and delete it from the system. The corresponding task record is removed from the MySQL database.

5.3.2 TASK PRIORITIZATION AND FILTERING

To ensure effective task management, the system allows filtering tasks based on priority and status:

1. **Priority Filtering:**
   The user can filter tasks by their priority (e.g., low, medium, high), which helps them focus on the most important tasks.
2. **Status Filtering:**
   Tasks can be filtered based on their completion status, such as Pending or Completed, ensuring that users can easily manage ongoing and finished tasks.

5.3.3 IMPLEMENTATION

1. **Task Identification:**
   When a user creates or updates a task, the task is assigned a unique ID that allows it to be easily identified within the database.
2. **Task Retrieval:**
   The system retrieves tasks from the database based on user input, using queries that filter tasks by priority, status, or deadline.
3. **Task Update and Deletion:**
   Users can select tasks to update or delete. The system ensures that these changes are reflected in the MySQL database through JDBC operations.

5.4 SYSTEM ARCHITECTURE OVERVIEW

1. User Interface (UI)

- **Swing GUI:**
  The user interacts with the system through a desktop interface, where they can add, update, and delete tasks. The GUI displays tasks in a table, with columns for task name, description, priority, and due date.

2. Backend (Application Logic)

- **JDBC for Database Integration:**
  The application logic is handled by Java, which connects to the MySQL database via JDBC. The logic is responsible for managing task data, handling CRUD (Create, Read, Update, Delete) operations, and ensuring data consistency.
- **Task Filtering Logic:**
  Filtering tasks based on priority and status is handled by Java, which queries the database using SQL queries.

3. Database (MySQL)

- **Task Data Storage:**
  The task data, including task names, descriptions, priorities, and deadlines, are stored in MySQL tables. The database is used for efficient data storage and retrieval.
- **Security and Data Integrity:**
  Data integrity is ensured by enforcing primary keys, constraints, and foreign keys within the database. Tasks are secured in the local database, and changes are committed through secure JDBC transactions.

4. Platform Independence

- **Cross-Platform Functionality:**
  The system is designed to work on any operating system that supports Java and MySQL, making it accessible across various platforms (e.g., Windows, Linux, macOS). The system operates offline, ensuring full functionality without internet access.

To-Do List Manager
Task Name
Prepared Date
Descopion
TASKS
ask Name
Due Date
Desorpty
Sacutun
Prepared Statement
Uplatieg Statusion
Prepareo Statesmen
MYSQL
Prepared Statement
Prepared Statement
STATUS
TASKS TATUS
TASKS
TASKS
Adding
Statement
Tasks
Task Name
Due Date
Due Date
Desc ation
Tasks
Status
TASKS
TASKS
STATUS
STATUS
Defart Table Modus
DEFART TABLE MODEL

FUNCTIONAL DESCRIPTION

1. Input Processing

- **User Input:** Accepts task details like name, description, priority, and due date.
- **Sanitization:** Ensures proper format by trimming spaces and validating inputs.

2. Feature Extraction

- **Task Metadata:** Features include task name, description, priority, and deadline.
- **Preprocessing:** Cleans data for accuracy, handling missing or invalid values.

3. Task Management

- **Add, Edit, Delete, Complete Tasks:** Users can create, update, delete, and mark tasks as completed.

4. Data Storage

- **Task Database:** Stores tasks in a list or MySQL database. Optional local file storage for persistence.

5. Sorting & Filtering

- **Sorting:** By priority, deadline, or completion status.
- **Filtering:** By priority, date, or status.

6. User Interface (UI)

- **Task Form & Display:** Form for task input and a list for displaying tasks.
- **Actions:** Add, edit, delete, and complete tasks via UI buttons.

7. Error Handling & Feedback

- **Error Handling:** Displays messages for invalid input (e.g., empty name, invalid date).
- **Feedback:** Provides confirmation messages for actions (e.g., "Task added successfully").

8. Scalability & Adaptability

- **Scalability:** Handles increasing tasks with no performance issues.
- **Adaptability:** Can add new features (e.g., recurring tasks, notifications).

9. Integration & Application

- **Web/Desktop:** Can be deployed as a web app (Spring Boot) or desktop app (JavaFX).
- **Recommendations:** Optional task suggestions based on priority or deadlines.

10. Enhancements

- **Future Features:** User authentication, notifications, and calendar integration.

TESTING

Input Processing (Task Name, Due Date, Priority, Description)

Test Case 1.1: Add a task with valid inputs

- **Input:**
  - Task Name: "Complete Java Assignment"
  - Due Date: "2024-11-30"
  - Priority: "3"
  - Description: "Complete the Java assignment by end of the month."
- **Expected Output:**
  - Task is successfully added to the database and displayed in the task table with correct details.

Test Case 1.2: Add a task with invalid date format

- **Input:**
  - Task Name: "Attend meeting"
  - Due Date: "30/11/2024" (invalid format)
  - Priority: "2"
  - Description: "Attend project meeting."
- **Expected Output:**
  - Error message: "Invalid input. Ensure date is YYYY-MM-DD and priority is 1-5."

Test Case 1.3: Add a task with invalid priority

- **Input:**
  - Task Name: "Prepare presentation"
  - Due Date: "2024-12-01"
  - Priority: "6" (invalid priority)
  - Description: "Prepare slides for the meeting."
- **Expected Output:**
  - Error message: "Invalid input. Ensure date is YYYY-MM-DD and priority is 1-5."

2. Task Addition to Database

Test Case 2.1: Successfully add task to database

- **Input:**
  - Task Name: "Submit Report"
  - Due Date: "2024-12-05"
  - Priority: "4"
  - Description: "Submit the annual financial report."
- **Expected Output:**

    - Task should be added to the database and reflected in the task table.

Test Case 2.2: Database connection failure during task addition

- **Input:**
    - Task Name: "Update Software"
    - Due Date: "2024-12-10"
    - Priority: "5"
    - Description: "Update the version to latest."
- **Expected Output:**
    - Error message: "Error adding task: Database connection failed."

3. Task Deletion

Test Case 3.1: Successfully delete selected task

- **Input:**
    - Select the task "Complete Java Assignment" from the table
    - Press the "Delete Task" button
- **Expected Output:**
    - The task is removed from the task table and the database.

Test Case 3.2: Attempt to delete task with no selection

- **Input:**
    - No task is selected
    - Press the "Delete Task" button
- **Expected Output:**
    - Error message: "Select a task to delete."

4. Update Task Status

Test Case 4.1: Successfully update task status

- **Input:**
    - Select the task "Complete Java Assignment"
    - Change status to "Completed" in the status combo box
    - Press the "Update Status" button
- **Expected Output:**
    - The task status should be updated in the task table and database to "Completed."

Test Case 4.2: Attempt to update status with no selection

- **Input:**
    - No task is selected
    - Press the "Update Status" button

- **Expected Output:**
    - Error message: "Select a task to update."

5. Filtering Tasks by Status

Test Case 5.1: Filter tasks by "All" status

- **Input:**
    - Select "All" in the filter combo box
    - Press the "Apply Filter" button
- **Expected Output:**
    - All tasks are displayed in the table regardless of their status.

Test Case 5.2: Filter tasks by "Completed" status

- **Input:**
    - Select "Completed" in the filter combo box
    - Press the "Apply Filter" button
- **Expected Output:**
    - Only tasks with status "Completed" are displayed in the table.

Test Case 5.3: Filter tasks by "Pending" status

- **Input:**
    - Select "Pending" in the filter combo box
    - Press the "Apply Filter" button
- **Expected Output:**
    - Only tasks with status "Pending" are displayed in the table.

6. Database Error Handling

Test Case 6.1: Database query failure when loading tasks

- **Input:** Any scenario where the database connection fails.
- **Expected Output:**
    - Error message: "Error loading tasks: Database connection issue."

7. Task Table Display

Test Case 7.1: Verify that all columns display correctly

- **Input:**
    - Ensure tasks are added and displayed in the table
- **Expected Output:**

    - Task table should show columns: "ID", "Task Name", "Due Date", "Priority", "Description", "Status".

8. Clear Input Fields

Test Case 8.1: After adding a task, clear input fields

- **Input:** Add a new task.
- **Expected Output:**
    - Input fields for Task Name, Due Date, Priority, Description, and Status should be cleared after task is added.

9. Validations

- **Test Case 9.1: Validate the due date format**

- **Input:** Enter a date "2024-12-32" (invalid date).
- **Expected Output:**
    - Error message: "Invalid input. Ensure date is YYYY-MM-DD and priority is 1-5."

Test Case 9.2: Validate the priority field

- **Input:** Enter priority as "abc" (non-numeric).
- **Expected Output:**
    - Error message: "Invalid input. Ensure date is YYYY-MM-DD and priority is 1-5."

10. UI Responsiveness

Test Case 10.1: Ensure UI is responsive on resizing

- **Input:** Resize the window and observe the input fields, table, and buttons.
- **Expected Output:**
    - The layout should adjust according to window size, maintaining usability.

TO-DO LIST APPLICATION IMPLEMENTATION OVERVIEW

The To-Do List Manager is built using Java with a MySQL database to store task data. The application allows users to manage and organize tasks with a graphical user interface (GUI). Below is a breakdown of the system's implementation:

1. Key Features:

2. **Task Management**: Allows users to add, update, delete, and filter tasks.
3. **Task Details**: For each task, users can input task name, due date, priority, description, and status.
4. **Filtering Tasks**: Users can filter tasks based on their status (e.g., Pending, Completed, or All).
5. **Database Integration**: The application uses MySQL to persist tasks and their attributes. It connects to the database using JDBC (Java Database Connectivity).
6. **GUI Interface**: The interface allows users to interact with the task list in a friendly environment. It includes input fields, buttons for managing tasks, and a table to display tasks.

7. Core Components:

- **JDBC Database Connection**: Manages database operations such as adding, deleting, updating, and retrieving tasks.
- **JTable**: Displays tasks in a table format, allowing users to view and manage tasks easily.
- **Action Listeners**: These are used to handle button clicks for adding, deleting, updating, and filtering tasks.
- **Task Table**: Displays the task details (ID, name, due date, priority, description, and status).

8. Example of Task Management Operations:

1. **Add Task**: Users input details for a task (e.g., name, due date, priority), and the system stores them in the database.
2. **Delete Task**: Users select a task from the table, and the system deletes it from the database.
3. **Update Task Status**: Users can update the status of a selected task (e.g., mark as completed or pending).
4. **Filter Tasks**: The system allows users to filter tasks based on their status (Pending, Completed, or All).

9. Task Operations Example:

1. Adding a Task:

```
java
Copy code
String query = "INSERT INTO tasks (task_name, due_date, priority,
description, status) VALUES (?, ?, ?, ?, ?)";
```

When a user adds a task, the information is inserted into the MySQL database.

2. Deleting a Task:

```
java
Copy code
String query = "DELETE FROM tasks WHERE id = ?";
```

When a user deletes a task, it is removed from the database using the task's ID.

3. Updating Task Status:

```
java
Copy code
String query = "UPDATE tasks SET status = ? WHERE id = ?";
```

When a user updates the status of a task (e.g., from Pending to Completed), the database is updated accordingly.

4. Filtering Tasks:

```
java
Copy code
String query = "SELECT * FROM tasks WHERE status = ? ORDER BY
due_date";
```

When the user selects a filter (Pending, Completed, or All), the tasks are filtered accordingly, and the results are displayed in the table.

10. GUI Interface:

- **Input Panel**: Allows users to input task details such as name, due date, priority, description, and status.
- **Buttons**: Used for adding, deleting, and updating tasks.
- **Task Table**: Displays a list of tasks and their details.
- **Filter Panel**: Provides an option to filter tasks by their status.

FUTURE ENHANCEMENTS

1. **User Feedback Integration**:

- **Feature**: Allow users to rate tasks (e.g., priority level, satisfaction after completion).
- **Benefit**: Enhances the task management experience by learning from user interactions, offering suggestions for future tasks based on prior user ratings.

2. **Task Prediction Model**:

- **Feature**: Implement machine learning to predict upcoming tasks based on user behavior and history.
- **Benefit**: Offers smart task recommendations, anticipating future to-dos based on patterns (e.g., recurring tasks or tasks related to previous ones).

3. **Customizable Task Categories**:

- **Feature**: Let users categorize tasks with customizable labels (e.g., Work, Personal, Urgent).
- **Benefit**: Improves task organization and prioritization by allowing users to filter tasks by category.

4. **•Collaborative Task Management**:

- **Feature**: Allow users to share tasks or lists with others for collaborative work.
- **Benefit**: Ideal for team or family projects, making it easier to coordinate tasks and monitor progress together.

5. **Advanced Task Filtering and Sorting**:

- **Feature**: Introduce advanced filtering options based on attributes like due date, priority, category, or user-defined tags.
- **Benefit**: Enables more flexible task management, making it easier for users to organize tasks according to their unique preferences.

6. **Cloud Synchronization**:

- **Feature**: Integrate cloud storage (e.g., Google Drive, Dropbox) to save tasks and sync across multiple devices.
- **Benefit**: Users can access their to-do lists from any device, ensuring seamless and uninterrupted task management.

7. **Real-Time Notifications**:

- **Feature**: Add real-time notifications for upcoming tasks, overdue tasks, or reminders.

- **Benefit**: Keeps users on track by providing timely reminders and updates, helping avoid missed deadlines.

8. **Task Analytics and Reporting**:

- **Feature**: Provide insights into task completion rates, overdue tasks, and time spent on tasks.
- **Benefit**: Helps users monitor their productivity and improve time management by providing actionable analytics.

9. **Voice Command Integration**:

- **Feature**: Enable voice commands for adding, updating, or deleting tasks (e.g., using Google Assistant or Alexa).
- **Benefit**: Increases convenience, allowing users to interact with the app hands-free while on the go.

10. **Multilingual Support**:

- **Feature**: Support multiple languages for global accessibility.
- **Benefit**: Expands the reach of the application, making it usable for users from different language backgrounds.

CONCLUSION

The To-Do List Manager application effectively allows users to organize and manage their tasks with a simple yet powerful interface. By integrating real-time updates, advanced filtering, and a clean, intuitive GUI, it offers a seamless experience for task management.

The database-driven backend ensures that tasks are stored persistently and can be retrieved at any time. The system is scalable and can be enhanced in future iterations with features such as user feedback integration, cloud synchronization, and collaborative task management. With these future enhancements, the system could cater to a broader user base, become more personalized, and provide deeper insights into user productivity.