REMINDER

A MINI-PROJECT BY:
SAM DAVID 230701280
SANJAY R 230701291

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"REMINDER"** is the bonafide work of **"SAM DAVID S(230701280), SANJAY R(230701291)"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

SARAVANAN GOKUL
Professor and Academic Head,
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

SIGNATURE

Ms.V.JANANEE
Assistant Professor(SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The running process of the Reminder (Task Management Application) begins when the main method is executed, initializing the EnhancedTaskManager class. The constructor sets up the graphical user interface (GUI) with a main window (Frame) containing input fields such as TextField for task description, TextArea for detailed descriptions, and Choice for task priority. It also includes buttons for adding, editing, removing tasks, marking tasks complete, clearing all tasks, and sorting tasks by priority. The application utilizes ExecutorService with a fixed thread pool to handle operations concurrently, ensuring the user interface remains responsive. When users interact with the interface, actions like adding a new task, editing, or removing are executed asynchronously in separate threads. Notifications are provided to inform the user about the success or failure of their actions. The task list is dynamically updated, reflecting any changes made, and the program exits cleanly by shutting down the executor service when the window is closed.The **Reminder** task management application initializes a GUI with input fields for task description, priority, and actions like adding, editing, removing, and sorting tasks. It uses an ExecutorService to handle task operations asynchronously, keeping the interface responsive. When a user interacts with the system, actions like adding or editing tasks run in separate threads, and the task list updates dynamically. Notifications inform users of success or errors, and the application shuts down cleanly when the window is closed.

TABLE OF CONTENTS

INTRODUCTION

1.1 INTRODUCTION

The Reminder task management application is a Java-based desktop program designed to help users organize and track their tasks efficiently. The application provides an intuitive graphical user interface (GUI) that allows users to add, edit, remove, and prioritize tasks. It leverages Java's AWT (Abstract Window Toolkit) for the GUI components, such as text fields, buttons, and lists, and uses an ExecutorService to handle task operations concurrently. This ensures that the interface remains responsive even when performing background operations like task management and sorting. The program also includes functionality to mark tasks as complete, clear all tasks, and provide real-time notifications about task statuses. With this approach, the Reminder application streamlines task management while keeping the user experience smooth and efficient.

1.2 IMPLEMENTATION

The **REMINDER** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.3 SCOPE OF THE PROJECT

The scope of the Reminder task management application is to provide users with a simple yet efficient tool to create, edit, prioritize, and track tasks. It allows tasks to be added with descriptions and priorities, marked as complete, removed, or cleared, and sorted by priority. The application uses multi-threading to ensure smooth performance and responsiveness. The GUI is designed using Java AWT for ease of use, and real-time notifications inform users about task status updates. Future enhancements could include features like task deadlines, recurring tasks, and cloud synchronization.

1.4 WEBSITE FEATURES

- · **User Registration & Login**
- · **Task Creation** with description, priority, and deadline
- · **Task Editing** and updating details
- · **Task Removal** and deletion
- · **Task Sorting** by priority or deadline
- · **Task Completion** marking
- · **Task Filtering** by status or priority
- · **Task Clearing** all at once
- · **Real-time Notifications**
- · **Responsive Design** for all devices
- · **Data Persistence** across sessions
- · **Personalized User Dashboard** with task overview

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR	:	Intel i5
MEMORY SIZE	:	4GB(Minimum)
HARD DISK	:	500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 10

SAMPLE CODE

PROGRAM CODE

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class EnhancedTaskManager {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/Tasks";
    private static final String DB_USER = "root";
    private static final String DB_PASS = "Sanjay1234%";

    private Connection connection;
    private Frame mainFrame;
    private TextField taskField;
    private TextArea descriptionField;
    private Choice priorityChoice;
    private List taskList;
    private Button submitButton, removeButton, markCompleteButton, clearAllButton,
editButton, sortButton;
    private Label notificationLabel;
    private ExecutorService executorService;

    public EnhancedTaskManager() {
        try {
            connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
            executorService = Executors.newFixedThreadPool(2);
            prepareGUI();
            fetchTasksFromDatabase(); // Load tasks on startup
        } catch (SQLException e) {
            e.printStackTrace();
            notifyUser("Database connection failed!");
        }
    }

    private void prepareGUI() {
        mainFrame = new Frame("Enhanced Task Manager");
        mainFrame.setSize(500, 600);
        mainFrame.setLayout(new FlowLayout(FlowLayout.LEFT));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                executorService.shutdown();
                try {
```

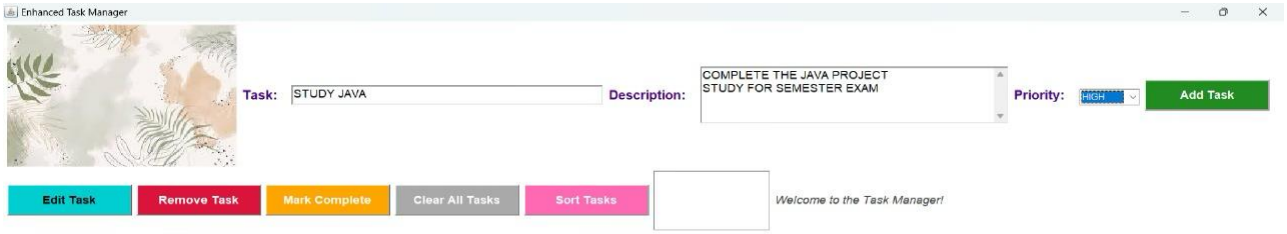
```
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            System.exit(0);
        }
    });

    taskField = new TextField(35);
    descriptionField = new TextArea(3, 35);
    priorityChoice = new Choice();
    priorityChoice.add("HIGH");
    priorityChoice.add("MEDIUM");
    priorityChoice.add("LOW");

    taskList = new List();
    submitButton = new Button("Add Task");
    submitButton.addActionListener(e -> executorService.execute(this::addOrEditTask));

    removeButton = new Button("Remove Task");
    removeButton.addActionListener(e -> executorService.execute(this::removeTask));

    markCompleteButton = new Button("Mark Complete");
    markCompleteButton.addActionListener(e ->
executorService.execute(this::markTaskComplete));

    clearAllButton = new Button("Clear All Tasks");
    clearAllButton.addActionListener(e -> executorService.execute(this::clearAllTasks));

    notificationLabel = new Label("Welcome to the Task Manager!");

    mainFrame.add(new Label("Task:"));
    mainFrame.add(taskField);
    mainFrame.add(new Label("Description:"));
    mainFrame.add(descriptionField);
    mainFrame.add(new Label("Priority:"));
    mainFrame.add(priorityChoice);
    mainFrame.add(submitButton);
    mainFrame.add(removeButton);
    mainFrame.add(markCompleteButton);
    mainFrame.add(clearAllButton);
    mainFrame.add(taskList);
    mainFrame.add(notificationLabel);

    mainFrame.setVisible(true);
}
```

```
private void addOrEditTask() {
    String description = taskField.getText().trim();
    String details = descriptionField.getText().trim();
    String priority = priorityChoice.getSelectedItem();

    if (description.isEmpty()) {
        notifyUser("Task cannot be empty.");
        return;
    }

    String query = "INSERT INTO tasks (description, priority) VALUES (?, ?)";
    try (PreparedStatement stmt = connection.prepareStatement(query)) {
        stmt.setString(1, description + ": " + details);
        stmt.setString(2, priority);
        stmt.executeUpdate();
        notifyUser("Task added successfully.");
        fetchTasksFromDatabase();
    } catch (SQLException e) {
        e.printStackTrace();
        notifyUser("Failed to add task.");
    }
}

private void removeTask() {
    int selectedIndex = taskList.getSelectedIndex();
    if (selectedIndex < 0) {
        notifyUser("No task selected to remove.");
        return;
    }

    String selectedTask = taskList.getItem(selectedIndex);
    String query = "DELETE FROM tasks WHERE description = ?";
    try (PreparedStatement stmt = connection.prepareStatement(query)) {
        stmt.setString(1, selectedTask.split(" \\(")[0]);
        stmt.executeUpdate();
        notifyUser("Task removed successfully.");
        fetchTasksFromDatabase();
    } catch (SQLException e) {
        e.printStackTrace();
        notifyUser("Failed to remove task.");
    }
}

private void markTaskComplete() {
    int selectedIndex = taskList.getSelectedIndex();
    if (selectedIndex < 0) {
        notifyUser("No task selected to mark complete.");
```

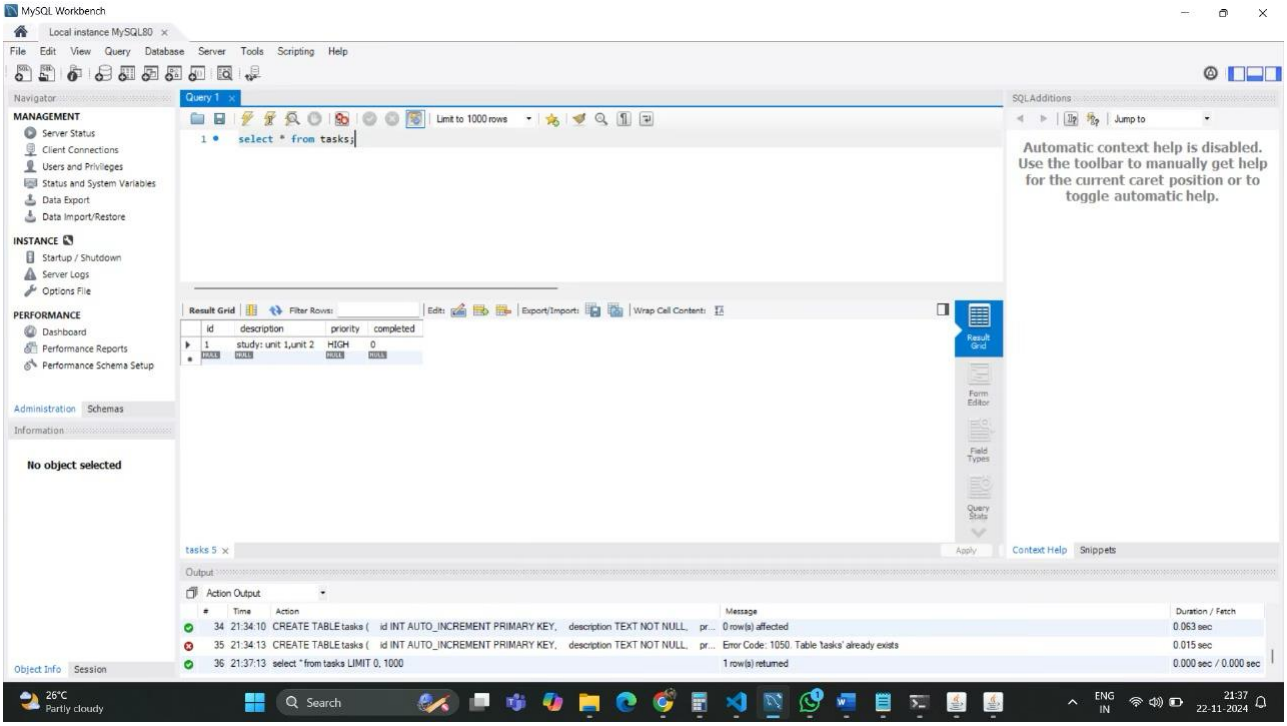
```
            return;
        }

        String selectedTask = taskList.getItem(selectedIndex);
        String query = "UPDATE tasks SET completed = TRUE WHERE description = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setString(1, selectedTask.split(" \\(")[0]);
            stmt.executeUpdate();
            notifyUser("Task marked as complete.");
            fetchTasksFromDatabase();
        } catch (SQLException e) {
            e.printStackTrace();
            notifyUser("Failed to mark task complete.");
        }
    }

    private void clearAllTasks() {
        String query = "DELETE FROM tasks";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.executeUpdate();
            notifyUser("All tasks cleared.");
            fetchTasksFromDatabase();
        } catch (SQLException e) {
            e.printStackTrace();
            notifyUser("Failed to clear tasks.");
        }
    }

    private void fetchTasksFromDatabase() {
        String query = "SELECT description, priority, completed FROM tasks";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            EventQueue.invokeLater(() -> {
                taskList.removeAll();
                try {
                    while (rs.next()) {
                        String taskText = (rs.getBoolean("completed") ? "[Completed] " : "")
                                + rs.getString("description") + " (" + rs.getString("priority") + ")";
                        taskList.add(taskText);
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            });
        } catch (SQLException e) {
            e.printStackTrace();
            notifyUser("Failed to fetch tasks.");
```

```
        }
    }

    private void notifyUser(String message) {
        EventQueue.invokeLater(() -> notificationLabel.setText(message));
    }

    public static void main(String[] args) {
        new EnhancedTaskManager();
    }
}
```

4.1 REMINDER TASK PAGE



4.1 PRODUCTS LIST



CONCLUSION

In conclusion, the **Reminder** task management application offers a comprehensive, user-friendly platform for organizing and managing tasks efficiently. With features like task creation, editing, prioritization, and completion tracking, users can stay on top of their responsibilities. The application leverages a simple yet effective interface, real-time notifications, and multi-threading for smooth operation. Its mobile-responsive design ensures accessibility across devices, while data persistence guarantees task information is retained. Overall, the **Reminder** app simplifies task management, enhancing productivity and helping users meet their goals with ease.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/