DAILY TASK SCHEDULER

A PROJECT REPORT

Submitted by

SUREKA V (2303811710422162)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"DAILY TASK SCHEDULER"** is the bonafide work of **SUREKA V (2303811710422162)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANTP ROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 06-12-2024

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"DAILY TASK SCHEDULER"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING.**

Signature

SUREKA V

Place: Samayapuram

Date: 06-12-2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution **"K.Ramakrishnan College of Technology (Autonomous)"**, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The **Daily Task Scheduler** is a console-based application that helps users manage tasks by adding, viewing, and deleting them. Each task includes a description and due time, and users can mark tasks as completed. It provides an easy way to stay organized and track daily activities.

The Daily Task Scheduler is a console-based application designed to simplify task management for users. It allows users to organize their daily activities by adding tasks with descriptions and due times. Users can view a comprehensive list of tasks for the day, delete unwanted tasks, and mark tasks as completed once done.

This scheduler promotes productivity and time management by offering a straightforward and efficient way to keep track of daily responsibilities. With a user-friendly interface and essential functionality, it serves as a practical tool for managing and prioritizing tasks effectively.

Future enhancements could include features like task categorization, reminders, and recurring tasks, making it even more versatile for users with diverse needs. By implementing robust error handling and a simple yet powerful design, the Daily Task Scheduler aims to become an indispensable companion for daily planning and organization.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Daily Tasks Scheduler is a Java-based console application designed to help users manage their daily tasks efficiently. It allows users to add tasks with descriptions and due times, view the list of pending tasks, mark tasks as completed, and delete tasks when needed. The program simplifies task management by providing an intuitive and structured approach to organizing daily activities, enhancing productivity and time management. By offering a user-friendly interface and essential functionalities, this project demonstrates the practical use of object-oriented programming and file handling in Java.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the "Daily Task Scheduler" project is to create a robust, interactive, and easy-to-use Java application that assists users in managing their daily activities effectively. The scheduler is designed to enable users to add tasks with descriptive details and specific due times, view the list of tasks for the day, delete unnecessary or completed tasks, and mark tasks as done. This project focuses on simplifying task management by providing a clear and intuitive interface, ensuring users stay organized and productive. The application encourages time management, reduces the risk of missed deadlines, and promotes a structured approach to daily planning. By offering a practical and reliable solution, the project also enhances the user's ability to prioritize tasks and streamline workflow effectively.

1.1 Overview

The "Daily Task Scheduler" is a Java-based project that provides a practical solution for managing daily activities through a console interface. It is designed to help users stay organized by enabling them to add tasks with detailed descriptions and due times, view all tasks for the day, mark tasks as completed, and delete unnecessary tasks. The application ensures that users can efficiently plan their day, prioritize tasks, and track their progress in a structured manner. The project emphasizes simplicity, usability, and functionality, making it suitable for individuals with varying levels of technical expertise. By combining core Java programming concepts and logical task flow, the application showcases the practical implementation of object-oriented principles while addressing real-world productivity challenges.

1.2 Java Programming Concepts

The "Daily Task Scheduler" project leverages fundamental and advanced Java programming concepts, particularly emphasizing object-oriented programming principles. Below are the key concepts implemented in the project:

1. Basic Concepts

- **Data Types and Variables:** The project uses appropriate data types to store task descriptions, due times, and statuses.

- **Control Statements:** Conditional statements (if-else) and loops (for, while) are utilized for task management operations such as adding, deleting, and listing tasks.
- **Input and Output Handling:** Java's Scanner class is used for user input, and System.out.println is used for displaying outputs.
- **Arrays or Collections:** Tasks are stored and managed using collections such as ArrayList for dynamic and flexible task handling.

2. Object-Oriented Programming Principles

- Class and Objects: The project defines a Task class to represent individual tasks, with attributes like description, due Time, and status. Objects of this class are created for each task.
- Encapsulation: Private attributes in the Task class are accessed and modified using getter and setter methods, ensuring data integrity.
- Inheritance (Optional): If additional features like categorizing tasks (e.g., Work Task, Personal Task) are added, inheritance can be used to extend the Task class.
- Polymorphism (Optional): Overloading methods, such as having multiple ways to display tasks (e.g., by due time or completion status), demonstrates polymorphism.
- Abstraction: The application uses abstracted methods to define operations like adding, deleting, or displaying tasks, hiding implementation details.
- Constructor Usage: Constructors in the Task class are used to initialize task objects efficiently.

3. Advanced Java Concepts

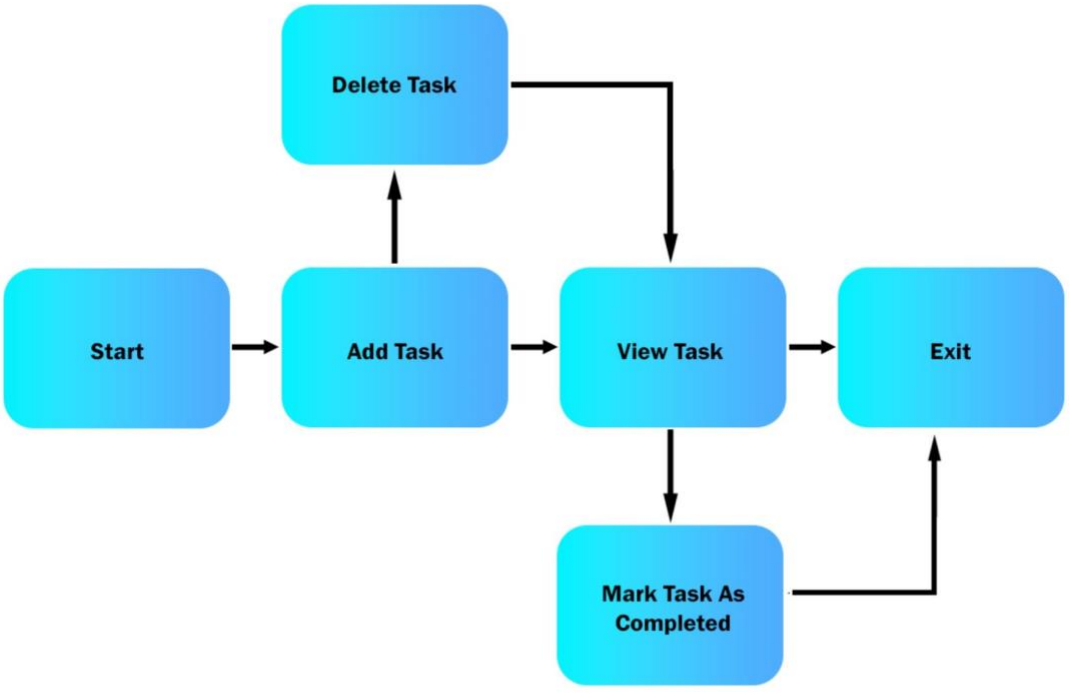
- Exception Handling: Proper handling of user input errors or invalid operations ensures a smooth user experience.
- File Handling (Optional): Tasks can be saved to and retrieved from a file for persistent storage.
- Java Collections Framework: ArrayList or HashMap is used for dynamic task storage and retrieval.
- Streams (Optional): Lambda expressions and streams can be employed for filtering and sorting tasks based on specific criteria.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The "Daily Task Scheduler" project proposes the creation of a console-based Java application designed to assist users in organizing and managing their daily tasks effectively. It will include essential features such as adding tasks with descriptions and due times, viewing the daily task list, marking tasks as completed, and deleting unnecessary tasks. By leveraging object-oriented programming principles and dynamic data structures, the application aims to provide a user-friendly and efficient platform for task management. Future enhancements may include persistent storage for saving tasks, sorting tasks by priority or due time, and generating reports to track task completion, ensuring a comprehensive solution for productivity challenges.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Task Management Module

The Task Management Module is responsible for storing and handling task attributes, such as description, due time, and completion status. It uses the Task class, which provides methods for retrieving task details, marking tasks as completed, and formatting them for display.

3.2 User Interface Module

The User Interface Module creates the graphical layout using AWT and Swing. It consists of a main JFrame with a task panel to display tasks in a list and a button panel for actions like adding, completing, and deleting tasks.

3.3 Task Input Module

The Task Input Module collects task details from the user using input dialogs. It uses JOptionPane to display input fields for task description and due time, ensuring a seamless task addition process.

3.4 Event Handling Module

The Event Handling Module manages user interactions, such as button clicks. It handles actions like adding new tasks, marking tasks as completed, deleting tasks, and exiting the application, ensuring the task list is updated accordingly.

3.5 Error Handling Module

The Error Handling Module validates user inputs and provides error or information messages for incomplete actions, such as selecting a task to mark as completed or filling in required fields, ensuring a smooth user experience.

CHAPTER 4

4.1 CONCLUSION

The **"Daily Task Scheduler"** project offers an effective and user-friendly solution for managing daily tasks, enabling users to organize, prioritize, and track their activities in a structured way. By leveraging key Java programming concepts such as object-oriented design, dynamic data structures, and efficient user interaction, the application allows users to easily add, update, delete, and view tasks. Features like task completion status and sorting tasks by due time help users stay focused and productive throughout the day. The project successfully meets its goal of providing an intuitive and reliable tool for task management, and with potential future enhancements such as task persistence and advanced sorting/filtering options, it can further enhance functionality, usability, and long-term task tracking.

4.2 FUTURE SCOPE

The future scope of the Daily Task Scheduler is vast and includes integrating advanced features like cloud synchronization to enable seamless access across devices and AI-driven insights for task prioritization based on user behavior, deadlines, and preferences. Adding voice command support will enhance accessibility, while cross-platform compatibility ensures usability across Android, iOS, and web platforms. Calendar and reminder integration can automate notifications and streamline task management. Gamification features, such as progress tracking and rewards for task completion, can boost user motivation. Enhanced security measures, including encryption and authentication, will protect sensitive data. Collaborative task management features can enable group planning for teams and families. Additionally, incorporating real-time analytics to track productivity trends, offline mode for uninterrupted usage, and localization for multilingual support can make the scheduler more versatile and inclusive. These advancements will not only improve functionality but also cater to diverse user needs, making the application indispensable in personal and professional environments.

APPENDIX A (SOURCE CODE)

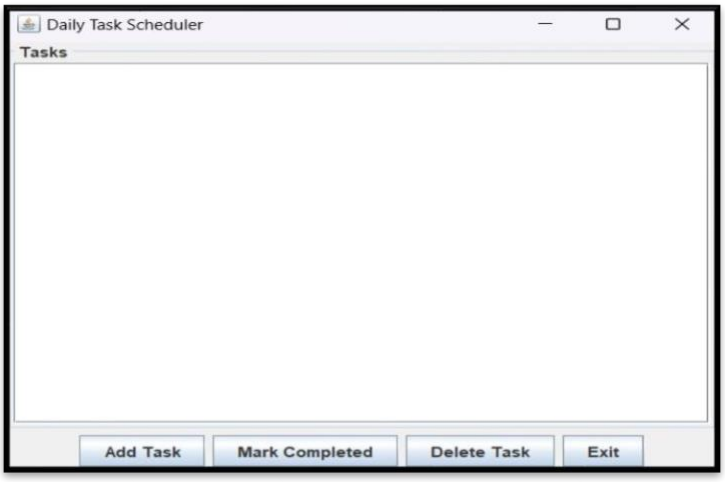
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

class Task {
    private String description;
    private String dueTime;
    private boolean isCompleted;

    public Task(String description, String dueTime) {
        this.description = description;
        this.dueTime = dueTime;
        this.isCompleted = false;
    }

    public String getDescription() {
        return description;
    }

    public String getDueTime() {
        return dueTime;
    }

    public boolean isCompleted() {
        return isCompleted;
    }

    public void markCompleted() {
        isCompleted = true;
    }

    @Override
    public String toString() {
        return (isCompleted ? "[Completed] " : "[Pending] ") +
                "Task: " + description + " | Due: " + dueTime;
    }
}

public class TaskScheduler extends JFrame {
    private ArrayList<Task> tasks = new ArrayList<>();
    private DefaultListModel<String> taskListModel = new DefaultListModel<>();
    private JList<String> taskList = new JList<>(taskListModel);

    public TaskScheduler() {
        setTitle("Daily Task Scheduler");
        setSize(500, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
```

```
        setLocationRelativeTo(null);

        // Layout
        setLayout(new BorderLayout());

        // Task List Panel
        JPanel taskPanel = new JPanel(new BorderLayout());
        taskPanel.setBorder(BorderFactory.createTitledBorder("Tasks"));
        taskList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        JScrollPane scrollPane = new JScrollPane(taskList);
        taskPanel.add(scrollPane, BorderLayout.CENTER);

        // Buttons Panel
        JPanel buttonPanel = new JPanel();
        JButton addTaskButton = new JButton("Add Task");
        JButton markCompletedButton = new JButton("Mark Completed");
        JButton deleteTaskButton = new JButton("Delete Task");
        JButton exitButton = new JButton("Exit");

        buttonPanel.add(addTaskButton);
        buttonPanel.add(markCompletedButton);
        buttonPanel.add(deleteTaskButton);
        buttonPanel.add(exitButton);

        add(taskPanel, BorderLayout.CENTER);
        add(buttonPanel, BorderLayout.SOUTH);

        // Button Actions
        addTaskButton.addActionListener(e -> addTask());
        markCompletedButton.addActionListener(e -> markTaskCompleted());
        deleteTaskButton.addActionListener(e -> deleteTask());
        exitButton.addActionListener(e -> System.exit(0));

        setVisible(true);
    }

    private void addTask() {
        JTextField descriptionField = new JTextField();
        JTextField dueTimeField = new JTextField();

        JPanel inputPanel = new JPanel(new GridLayout(2, 2));
        inputPanel.add(new JLabel("Task Description:"));
        inputPanel.add(descriptionField);
        inputPanel.add(new JLabel("Due Time (e.g., HH:MM):"));
        inputPanel.add(dueTimeField);

        int result = JOptionPane.showConfirmDialog(this, inputPanel, "Add Task",
JOptionPane.OK_CANCEL_OPTION);
        if (result == JOptionPane.OK_OPTION) {
            String description = descriptionField.getText().trim();
```

```
                String dueTime = dueTimeField.getText().trim();

                if (!description.isEmpty() && !dueTime.isEmpty()) {
                    Task newTask = new Task(description, dueTime);
                    tasks.add(newTask);
                    taskListModel.addElement(newTask.toString());
                    JOptionPane.showMessageDialog(this, "Task added successfully!");
                } else {
                    JOptionPane.showMessageDialog(this, "Please fill in all fields!", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        }

        private void markTaskCompleted() {
            int selectedIndex = taskList.getSelectedIndex();
            if (selectedIndex != -1) {
                Task selectedTask = tasks.get(selectedIndex);
                if (!selectedTask.isCompleted()) {
                    selectedTask.markCompleted();
                    taskListModel.set(selectedIndex, selectedTask.toString());
                    JOptionPane.showMessageDialog(this, "Task marked as completed!");
                } else {
                    JOptionPane.showMessageDialog(this, "Task is already completed!", "Info",
JOptionPane.INFORMATION_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(this, "Please select a task to mark as
completed!", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }

        private void deleteTask() {
            int selectedIndex = taskList.getSelectedIndex();
            if (selectedIndex != -1) {
                tasks.remove(selectedIndex);
                taskListModel.remove(selectedIndex);
                JOptionPane.showMessageDialog(this, "Task deleted successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Please select a task to delete!",
"Error", JOptionPane.ERROR_MESSAGE);
            }
        }

        public static void main(String[] args) {
            SwingUtilities.invokeLater(TaskScheduler::new);
        }
    }
```

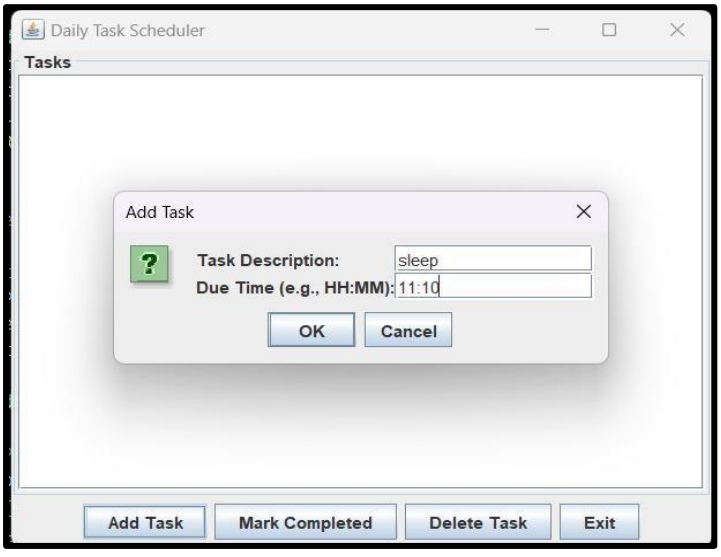
APPENDIX B (SCREENSHOT)

1. USER INTERFACE MODULE



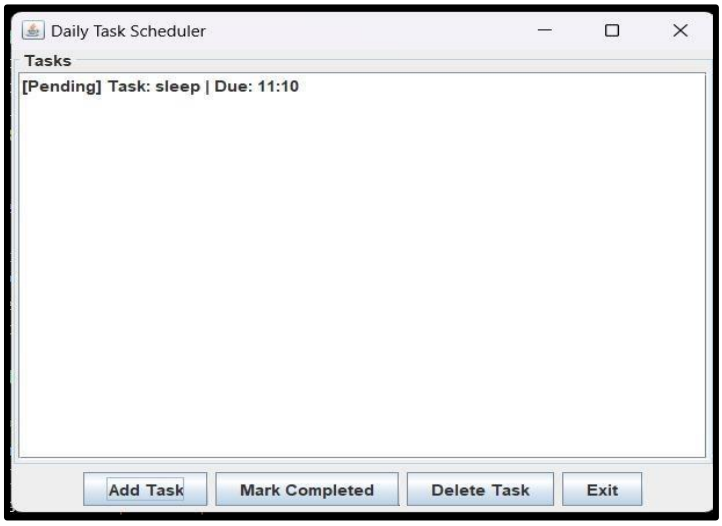
In this user interface module in it includes Add Task, Mark Completed, Delete Task, and Exit buttons at the bottom provide easy access to core functionalities and this is also the navigation buttons.

2. ADDING TASK



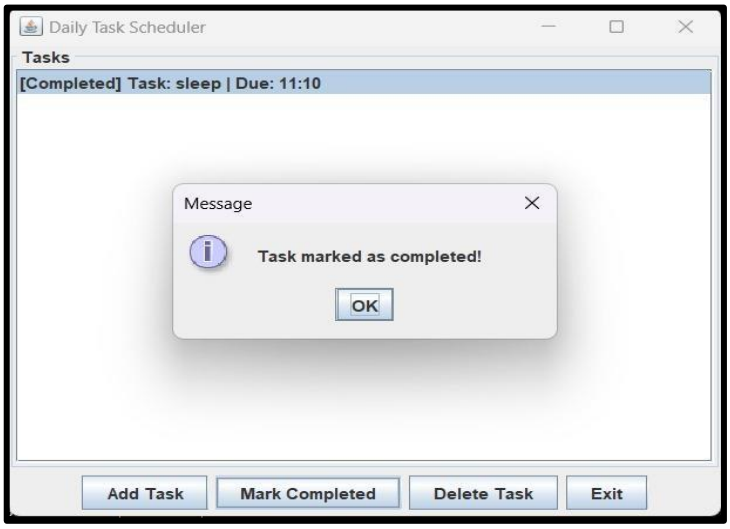
Allows the user to input task descriptions and specify due times through a pop-up dialog.

3. VIEWING TASK



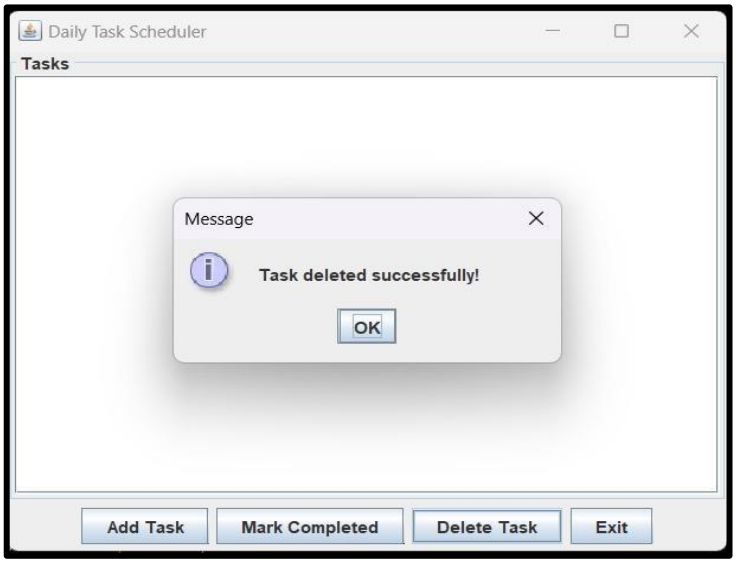
Displays the list of tasks, with updates reflecting actions like completions or deletion.

4. MARK COMPLETED



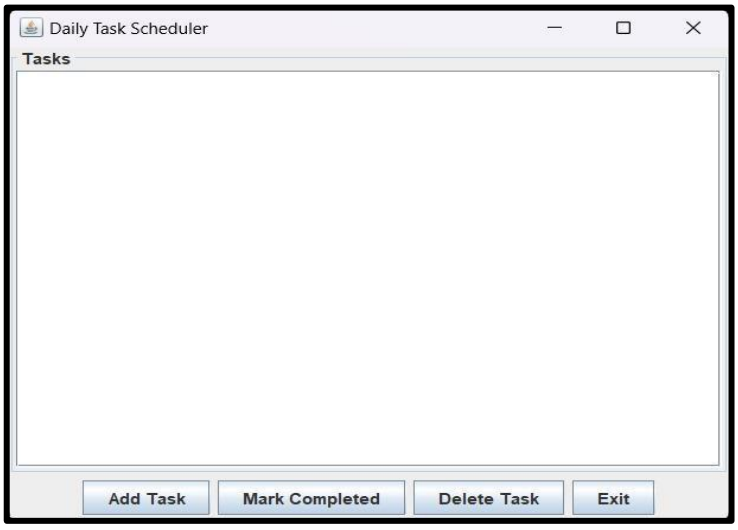
Enables marking tasks as completed with immediate visual feedback (e.g., [Completed] tag added to tasks).

5. DELETE TASK



Removes tasks and confirms the deletion via a dialog box.

6. EXIT MODULE



The Exit Module ensures safe termination of the application with a confirmation dialog to prevent accidental closure.

REFERENCES

1. Official Java Platform Documentation by Oracle. https://docs.oracle.com/javase/

2. Comprehensive tutorials on implementing data structures like lists and maps for task management. https://docs.oracle.com/javase/tutorial/collections/

3. Tutorials on implementing keyboard and mouse listeners for interactive console applications. https://www.tutorialspoint.com/java/java_event_handling.htm

4. Tutorials on file handling for saving and retrieving task data in Java. https://www.geeksforgeeks.org/file-handling-in-java/

5. Tutorials on managing threads and timers for scheduling tasks efficiently. https://www.baeldung.com/java-concurrency

6. Examples of task scheduler implementations available in open-source repositories like GitHub. https://github.com/