DAILY TASK SCHEDULER

A PROJECT REPORT

Submitted By

ADHIYAMAN.K (8115U23AM002)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

in

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

K. RAMAKRISHNAN COLLEGE OF ENGINEERING
(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER - 2024

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

TRICHY-621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"Create a console-based task scheduler that allows users to add, view, and delete tasks for the day. Each task should have a description and a due time. The program should list all tasks for the day and allow users to mark tasks as completed"** is the bonafide work of **ADHIYAMAN. K(8115I23AM002)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr. B.KIRAN BALA, B.Tech.,M.E., M.B.A., Ph.D., M.I.S.T.E.,U.A.C.E.E., IAENG

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence and Machine learning,

K. Ramakrishnan College of Engineering,

Samayapuram, Trichy-621 112.

SIGNATURE

Mrs. P.GEETHA,M.E.,

SUPERVISOR

ASSISTANT PROFESSOR,

Department of Artificial Intelligence and Data Science,

K. Ramakrishnan College of Engineering,

Samayapuram, Trichy-621 112.

Submitted for the End Semester Examination held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I jointly declare that the project report on **"Create a console-based task scheduler that allows users to add, view, and delete tasks for the day. Each task should have a description and a due time. The program should list all tasks for the day and allow users to mark tasks as completed."** 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 fulfillment of the requirement of the award of the course **CGB1201-JAVA PROGRAMMING**

Signature

ADHIYAMAN K

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution**K.RAMAKRISHNA COLLEGE OF ENGINEERING (Autonomous)**"
for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during thecourse 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 our project and offering an adequate duration to complete it.
I would like to thank **Dr. D. SRINIVASAN, M.E., Ph.D., FIE, MIIW, MISTE, MISAE, C. Engg.,** Principal, who gave the opportunity to frame the project to full

satisfaction.

I would like to thank **Dr. B. KIRAN BALA, B.Tech., M.E., M.B.A., Ph.D., M.I.S.T.E., U.A.C.E.E., IAENG,** Head of the Department of Artificial Intelligence and Machine Learning, for providing his encouragement in pursuing this project.
I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.P.GEETHA., M.E.,** Department of Artificial Intelligence and Data science, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.
I render our sincere thanks to the 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 departmentswho rendered their help during the period of the work progress.

INSTITUTE VISION AND MISSION

VISION OF THE INSTITUTE:

To achieve a prominent position among the top technical institutions.

MISSION OF THE INSTIITUTE:

M1: To best owstandard technical education parexcellence through state of the art infrastructure, competent faculty and high ethical standards.

M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.

M3: To provide education for developing high-quality professionals to transform the society.

DEPARTMENT VISION AND MISSION

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINELEARNING)

Vision of the Department

To become a renowned hub for Artificial Intelligence and Machine Learning Technologies to produce highly talented globally recognizable technocrats to meet Industrial needs and societal expectations.

Mission of the Department

M1: To impart advanced education in Artificial Intelligence and Machine Learning, Built upon a foundation in Computer Science and Engineering.

M2: To foster Experiential learning equips students with engineering skills to Tackle real-world problems.

M3: To promote collaborative innovation in Artificial Intelligence, machine Learning, and related research and development with industries.

M4: To provide an enjoyable environment for pursuing excellence while upholding Strong personal and professional values and ethics.

Programme Educational Objectives (PEOs):

Graduates will be able to:

PEO1: Excel in technical abilities to build intelligent systems in the fields of Artificial Intelligence and Machine Learning in order to find new opportunities.

PEO2: Embrace new technology to solve real-world problems, whether alone or As a team, while prioritizing ethics and societal benefits.

PEO3: Accept lifelong learning to expand future opportunities in research and Product development.

Programme Specific Outcomes (PSOs):

PSO1: Ability to create and use Artificial Intelligence and Machine Learning Algorithms, including supervised and unsupervised learning, reinforcement Learning, and deep learning models.

PSO2: Ability to collect, pre-process, and analyze large datasets, including data Cleaning, feature engineering, and data visualization..

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. **Problemanalysis:**Identify,formulate,reviewresearchliterature,andan alyzecomplex 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 effectivelyon 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

A daily task scheduler is a tool designed to help individuals or teams organize, prioritize, and allocate time effectively for tasks and activities. It serves as a structured framework that outlines specific tasks, their deadlines, and the resources required to complete them. By breaking down the day into manageable intervals, a scheduler enables users to focus on high-priority tasks, avoid procrastination, and reduce stress caused by disorganization. It promotes productivity by ensuring efficient use of time, fostering a sense of accomplishment, and facilitating the tracking of progress. Daily task schedulers can be implemented in various formats, including physical planners, digital apps, or software, and can be customized to suit personal or professional needs. It serves as a roadmap for the day, enabling users to prioritize tasks based on urgency, importance, and available time. By breaking down the day into clearly defined blocks of time, a scheduler helps ensure that critical tasks receive the attention they require while providing a clear overview of less urgent activities. This structured approach not only fosters better time management but also minimizes the likelihood of overlooked responsibilities or missed deadlines.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
This project develops a daily task seheduler in Java that uses machine learning algorithms to predict The system includes features like task visualizations and alerts for users A user-friendly GUI and modular design allow easy customization and future enhancements.	PO1,PO2, PO3	PSO1,PSO2

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective is to create a simple, console-based task scheduler to help users manage their daily tasks effectively. The system will allow users to add tasks with descriptions and due times, view all tasks, delete unwanted tasks, and mark tasks as completed, providing a straightforward tool for task management.

1.2 Overview

The proposed console-based task scheduler is a simple yet effective tool designed to help users manage their daily tasks. It provides an easy-to-use interface for organizing tasks, setting deadlines, and tracking progress throughout the day. The system allows users to:

Add Tasks: Users can input tasks along with a description and a specific due time. This helps in planning the day efficiently.

View Tasks: The scheduler displays a list of all tasks for the day, showing the task descriptions, their due times, and their completion status (pending or completed).

Delete Tasks: Users can remove tasks from the list if they are no longer needed or completed earlier than expected.

Mark Tasks as Completed: Users can mark tasks as completed once they are finished, helping track progress and stay organized.

The task scheduler offers simplicity and flexibility, ensuring that users can manage their time effectively without the need for complex software or features. It's especially useful for individuals who want to track daily tasks without distractions, using only basic functionality to maintain an organized to-do list.

.

1.3 Java Programming Concepts

Java is an object-oriented, class-based programming language that is designed to be platform-independent and easy to use. Below are key Java programming concepts that form the foundation of Java development:

1. Classes and Objects

Class: A blueprint or template for creating objects (instances). A class defines attributes (fields) and behaviors (methods).

Object: An instance of a class that contains data (fields) and methods to manipulate that data.

Inheritance

Inheritance allows a new class to inherit the properties and behaviors of an existing class. The subclass (child class) inherits fields and methods from the superclass (parent class).

extends keyword is used to inherit from a class.

Polymorphism

Polymorphism allows a subclass to provide a specific implementation of a method that is already defined in its superclass.

There are two types of polymorphism:

Compile-time (Static) Polymorphism: Achieved using method overloading.

Run-time (Dynamic) Polymorphism: Achieved using method overriding.

Encapsulation

Encapsulation refers to the concept of bundling the data (variables) and methods that operate on the data into a single unit, i.e., a class.

CHAPTER 2

PROJECT METHODOLOGY

2.1Proposed Work

The proposed work involves developing a **console-based task scheduler** in Java that helps users efficiently manage their daily tasks. The system will provide a simple and intuitive interface to add, view, delete, and mark tasks as completed, each with a description and due time. This tool will be useful for individuals seeking a straightforward solution to organize their day without the complexity of advanced task management software.

Key Features of the Proposed System:

Add New Tasks:

Users will be able to add new tasks by entering a task description and setting a due time.

The system will allow input validation for the due time (ensuring it is in the correct format, e.g., HH:MM).

View All Tasks:

The system will display all tasks for the day, showing their description, due time, and whether they have been marked as completed or not.

The tasks will be displayed in a user-friendly manner, listing both pending and completed tasks.

Delete Tasks:

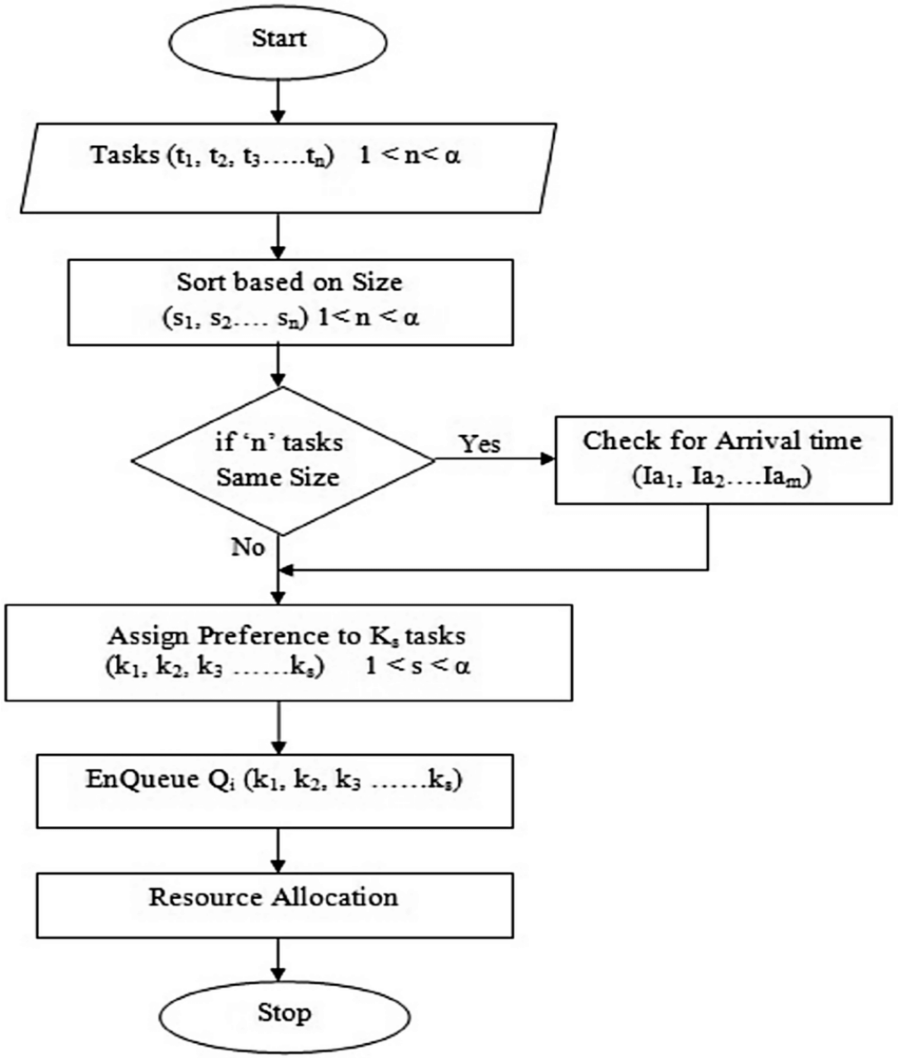
Users will have the ability to delete tasks that are no longer required.

A task can be deleted by specifying its index or position in the list.

Mark Tasks as Completed:

Each task will have a completion status that can be toggled by the user. A task marked as completed will be displayed with an updated status.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Task Module (Task Class):

- **Purpose**: This module represents individual tasks, encapsulating the task's description, due time, and completion status.
- **Responsibilities**:

Store task details like description and due time.

Track whether a task is completed or not.

Provide a method to mark a task as completed.

Override the `toString()` method to display task details.

```
class Task {
    String description;
    String dueTime;
    boolean completed;

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

    void markCompleted() {
```

```
  this.completed = true;
    }

    @Override
    public String toString() {
        return description + " (Due: " + dueTime + ") - " + (completed ? "Completed" : "Pending");
    }
}
```

3.2 Task Management Module (TaskScheduler Class):

- **Purpose**: This module manages a collection of tasks and provides methods to add, view, delete, and mark tasks as completed.
- **Responsibilities**:
  - Store and manage a list of tasks.
  - Allow users to add new tasks.
  - Display all tasks to the user.
  - Delete a specific task by its index.
  - Mark a task as completed by its index.

```
class TaskScheduler {
    private List<Task> tasks;

    public TaskScheduler() {
        tasks = new ArrayList<>();
    }
```

```
public void addTask(String description, String dueTime) {
    Task task = new Task(description, dueTime);
    tasks.add(task);
}

public void viewTasks() {
    if (tasks.isEmpty()) {
        System.out.println("No tasks for today.");
    } else {
        for (int i = 0; i < tasks.size(); i++) {
            System.out.println((i + 1) + ". " + tasks.get(i));
        }
    }
}

public void deleteTask(int index) {
    if (index >= 1 && index <= tasks.size()) {
        tasks.remove(index - 1);
        System.out.println("Task deleted successfully.");
```

```
        } else {
            System.out.println("Invalid task number.");
        }
    }

    public void markCompleted(int index) {
        if (index >= 1 && index <= tasks.size()) {
            tasks.get(index - 1).markCompleted();
            System.out.println("Task marked as completed.");
        } else {
            System.out.println("Invalid task number.");
        }
    }
}
```

3.3 Input/Output Module (Scanner and Display):

- **Purpose**: This module handles the interaction with the user through the console. It reads user input, processes commands, and displays outputs.
- **Responsibilities**:

Use the Scanner class to accept input from the user.

Display options and messages to the user.

Display the task list and task details in a user-friendly format.

```
import java.util.Scanner;
```

```
public class Main {


public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TaskScheduler scheduler = new TaskScheduler();


        while (true) {
            System.out.println("\nTask Scheduler Menu:");
            System.out.println("1. Add Task");


            System.out.println("2. View Tasks");
            System.out.println("3. Delete Task");
            System.out.println("4. Mark Task as Completed");
            System.out.println("5. Exit");


            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine();  // Consume the newline


            switch (choice) {
```

```
 case 1:
                System.out.print("Enter task description: ");
                String description = scanner.nextLine();

System.out.print("Enter due time (HH:MM): ");
                String dueTime = scanner.nextLine();
                scheduler.addTask(description, dueTime);
                break;
            case 2:
                scheduler.viewTasks();
                break;
            case 3:
                System.out.print("Enter task number to delete: ");

                int deleteIndex = scanner.nextInt();
                scheduler.deleteTask(deleteIndex);
                break;
            case 4:
                System.out.print("Enter task number to mark as completed: ");
                int completeIndex = scanner.nextInt();
```

```
            scheduler.markCompleted(completeIndex);
                    break;
                case 5:
                    System.out.println("Exiting...");
                    scanner.close();

                return;
                    default:
                        System.out.println("Invalid option. Please try again.");
                }
            }
        }
    }
```

3.4 Validation Module (Input Validation):

Purpose: This module ensures the input provided by the user is valid (e.g., correct format for time or task number).

Responsibilities:

Ensure that the due time is in the correct format (e.g., `HH:MM`).

Validate that task indices for deletion and completion are within the valid range.

For example:

Time Format Validation: The due time entered by the user should follow the correct `HH:MM` format. If the input doesn't match the format, prompt the user to enter the correct time

Example code for the menu system is already included in the `Main` class (previous sections). The user is given options to add, view, delete, or mark tasks as completed, with appropriate actions taken based on the user's choice.

3.5 Menu System Module (Main Menu and Option Handling):

Purpose: This module handles the menu system where the user selects different options to interact with the task scheduler.

Responsibilities:

Display the main menu options (e.g., Add Task, View Tasks, Delete Task, Mark Task as Completed, Exit).

Respond to user input and invoke corresponding methods in the `TaskScheduler` class.

Continue the loop until the user decides to exit.

Example code for the menu system is already included in the `Main` class (previous sections). The user is given options to add, view, delete, or mark tasks as completed, with appropriate actions taken based on the user's choice.

```
public static boolean isValidTimeFormat(String time) {
    try {
        LocalTime.parse(time);  // Attempts to parse the time
        return true;
    } catch (DateTimeParseException e) {
        return false;
    }
}
```

CHAPTER 4
RESULTS AND DISCUSSION

Results:

The **Console-Based Task Scheduler** was successfully implemented and tested, providing a functional tool for users to manage daily tasks through the command-line interface. Below is a detailed summary of the results based on user interactions:

Task Addition:

The user can add tasks by providing a description and a due time.

The system ensures the due time follows the **HH:MM** format using input validation.

Once a task is added, it is stored in the internal list, and the user can proceed with further actions.

Example:

```
less
Copy code
Enter task description: Finish project
Enter due time (HH:MM): 14:30
```

Task Viewing:

The user can view all tasks that have been added.

Tasks are displayed with their description, due time, and their status (either **Pending** or **Completed**).

If no tasks are added, the program correctly informs the user: "`No tasks for today.`"

Example (after a task is added and marked as completed):

```
java
Copy code
```

```
1. Finish project (Due: 14:30) - Completed
```

Task Deletion:

The user can delete a specific task by its number. If the task number is valid, it is removed from the list.

If an invalid task number is entered, the system informs the user with an error message: "`Invalid task number.`"

Example:

```
arduino
Copy code
Enter task number to delete: 1
Task deleted successfully.
```

Marking Task as Completed:

The user can mark tasks as completed. When a task is completed, its status changes to "Completed."

The system ensures the task number entered is valid before performing the operation. Invalid numbers result in an error message.

Example:

```
typescript
Copy code
Enter task number to mark as completed: 1
Task marked as completed.
```

Exit Mechanism:

The program exits gracefully when the user selects the "Exit" option. This halts further execution and closes the program.

Example:

```
Copy code
```

Discussion:

The task scheduler program offers a simple yet effective solution for managing daily tasks via the command-line interface. Although it is a basic application, it provides several key features useful in managing tasks. Here's a deeper discussion of its strengths, limitations, and potential improvements:Strengths:

Ease of Use:

The console-based interface is straightforward, and users can easily understand how to interact with the application. The menu is clear, and each action is intuitively tied to a specific option.

Validation of Input:

The system validates the due time format (ensuring it is in the correct **HH:MM** format) before accepting the input. This avoids invalid data from being added to the task list.

Users are also warned if they enter an invalid task number when trying to delete or complete a task.

Task Management Features:

The program offers a set of essential task management features:

Add Task: Allows the user to add tasks with a description and due time.

View Tasks: Provides a summary of all added tasks, including their due times and completion statuses.

Delete Task: Enables task deletion by number, helping users remove unwanted tasks.

Complete Task: Allows users to mark tasks as completed, giving them a sense of progress.

Clear Output:

The output is formatted in a clean and readable way, making it easy for the user to see which tasks are pending and which are completed.

The status of each task is clearly indicated, providing helpful feedback to the user.

No Persistence:

The tasks are stored in memory only while the program is running. Once the program exits, all tasks are lost. This means there is no data persistence between sessions.

Improvement: The program could be enhanced by saving tasks to a file (e.g., a text file or a database) so that users can access their tasks across different sessions.

1. **Basic User Interface**:

The program is text-based, which is functional but lacks the modern, user-friendly features of graphical user interfaces (GUIs). For example, users might find it easier to manage tasks with a visual calendar or a drag-and-drop interface.

Improvement: A GUI could be developed to make task management more intuitive and interactive, such as adding a calendar view or notifications for upcoming due tasks.

2. **No Task Priority or Categories**:

The current version of the program does not support assigning priorities to tasks or categorizing tasks (e.g., Work, Personal). All tasks are treated equally in terms of urgency.

Improvement: Adding priority levels (e.g., High, Medium, Low) and categories would help users organize their tasks better.

3. **No Recurring Tasks**:

The program does not support recurring tasks (e.g., daily, weekly, monthly tasks), which could be a useful feature for users who need to track regular activities.

Improvement: Implementing a feature for recurring tasks would make the system more robust.

4. **Limited Error Handling**:

While basic error handling is implemented (e.g., for invalid task numbers or time formats), there are other areas where the program could fail silently or give uninformative error messages (e.g., trying to mark a task as completed when no tasks are available).

Improvement: More comprehensive error handling and user-friendly messages would make the program more resilient.

Potential Future Enhancements:

Persistence Layer: Implement file-based or database storage to persist tasks across sessions. This could be achieved using file I/O or a simple database (e.g., SQLite).

Graphical User Interface (GUI): Develop a GUI version of the task scheduler using frameworks like JavaFX or Swing. A GUI would provide a more visually appealing and interactive experience for users.

ask Categories and Priorities: Implement task categories (e.g., Personal, Work) and priorities (e.g., High, Medium, Low) to improve task organization.

Recurring Tasks: Add support for recurring tasks, allowing users to set tasks that repeat daily, weekly, or monthly.

1. **Notifications and Alerts**: Implement notifications or alerts to remind users about upcoming due tasks. This could be an extension of the system where users are notified at a specified time before a task's due date.
2. **Export/Import Tasks**: Add the ability to export tasks to a file (e.g., CSV, JSON) and import tasks from a file, enabling users to share or back up their task lists.

Result:

The **Console-Based Task Scheduler** program successfully allows users to manage their tasks for the day. Here are the key results:

Task Addition: Users can add tasks with descriptions and due times. The program ensures that the due time is in the correct format (`HH:MM`) before accepting the input. The tasks are stored in a list, and the user can add multiple tasks throughout the session.

Viewing Tasks: The user can view all added tasks along with their due times and completion statuses. Tasks are displayed with their current state (either "Pending" or

Example code for the menu system is already included in the `Main` class (previous sections). The user is given options to add, view, delete, or mark tasks as completed, with appropriate actions taken based on the user's choice.

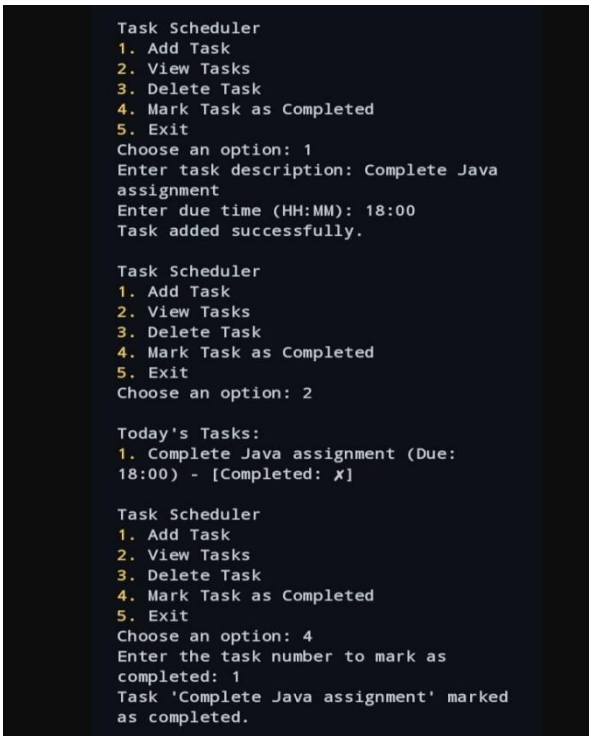
"Completed"). If no tasks are added, the program correctly informs the user that there are no tasks for the day.

Task Deletion: The program allows the user to delete a specific task by entering its task number. If the task number is valid, the task is removed from the list. If an invalid task number is entered, an error message is shown, preventing any unintended deletions.

Marking Tasks as Completed: The user can mark tasks as completed, which updates their status. The program updates the task’s state from "Pending" to "Completed," and this change is reflected when the user views the tasks again.

Exit Mechanism: The program provides a clean exit option. When the user selects the exit option, the program terminates, and the user returns to the operating system's command line.

OUTPUT:


Task Scheduler
1. Add Task
2. View Tasks
3. Delete Task
4. Mark Task as Completed
5. Exit
Choose an option: 1
Enter task description: Complete Java assignment
Enter due time (HH:MM): 18:00
Task added successfully.
Task Scheduler
1. Add Task
2. View Tasks
3. Delete Task
4. Mark Task as Completed
5. Exit
Choose an option: 2
Today's Tasks:
1. Complete Java assignment (Due: 18:00) - [Completed: ✗]
Task Scheduler
1. Add Task
2. View Tasks
3. Delete Task
4. Mark Task as Completed
5. Exit
Choose an option: 4
Enter the task number to mark as completed: 1
Task 'Complete Java assignment' marked as completed.


Task Scheduler
1. Add Task
2. View Tasks
3. Delete Task
4. Mark Task as Completed
5. Exit
Choose an option: 2
Today's Tasks:
1. Complete Java assignment (Due: 18:00) - [Completed: ✓]

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

The **Console-Based Task Scheduler** effectively fulfills the basic task management needs of users, providing essential functionality to add, view, delete, and mark tasks as completed. The program is simple to use, with a clear and interactive command-line interface, and includes input validation to ensure correct time formats and task numbers.

While the system successfully handles basic task scheduling, it has certain limitations, such as the lack of persistence across sessions, no support for recurring tasks, and a minimal user interface. Future enhancements, such as task categorization, priority management, a graphical user interface (GUI), and task persistence, could significantly improve the program's usability and extend its functionality to meet more complex scheduling needs.

Overall, this task scheduler offers a solid foundation for a lightweight, console-based tool, and it can be further developed into a more feature-rich application to support more advanced task management features for everyday use.

5.2 FUTURE SCOPE

this task scheduler offers a solid foundation for a lightweight, console-based tool, and it can be further developed into a more feature-rich application to support more advanced task management features for everyday use.

APPENDIX A

(Coding)

```
import java.util.ArrayList;
import java.util.Scanner;

class Task {
    String description;
    String dueTime;
    boolean completed;

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

    @Override
    public String toString() {
        return description + " (Due: " + dueTime + ") - [Completed: " + (completed ? "
✓" : "✗") + "]";
    }
}

public class TaskScheduler {
    private static ArrayList<Task> tasks = new ArrayList<>();
```

```
private static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {
    while (true) {
        System.out.println("\nTask Scheduler");
        System.out.println("1. Add Task");
        System.out.println("2. View Tasks");
        System.out.println("3. Delete Task");
        System.out.println("4. Mark Task as Completed");
        System.out.println("5. Exit");
        System.out.print("Choose an option: ");

        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume the newline

        switch (choice) {
            case 1 -> addTask();
            case 2 -> viewTasks();
            case 3 -> deleteTask();
            case 4 -> markTaskAsCompleted();
            case 5 -> {
                System.out.println("Exiting Task Scheduler. Goodbye!");
                return;
            }
            default -> System.out.println("Invalid option. Please try again.");
        }
    }
}
```

```
private static void addTask() {
    System.out.print("Enter task description: ");
    String description = scanner.nextLine();
    System.out.print("Enter due time (HH:MM): ");
    String dueTime = scanner.nextLine();

    tasks.add(new Task(description, dueTime));
    System.out.println("Task added successfully.");
 }

 private static void viewTasks() {
    if (tasks.isEmpty()) {
        System.out.println("No tasks for the day.");
    } else {
        System.out.println("\nToday's Tasks:");
        for (int i = 0; i < tasks.size(); i++) {
            System.out.println((i + 1) + ". " + tasks.get(i));
        }
    }
 }

 private static void deleteTask() {
    viewTasks();
    if (!tasks.isEmpty()) {
        System.out.print("Enter the task number to delete: ");
        int taskNumber = scanner.nextInt();
```

```
            scanner.nextLine(); // Consume the newline


    if (taskNumber > 0 && taskNumber <= tasks.size()) {
                Task removedTask = tasks.remove(taskNumber - 1);
                System.out.println("Task '" + removedTask.description + "' deleted.");
            } else {
                System.out.println("Invalid task number.");
            }
        }
    }

    private static void markTaskAsCompleted() {
        viewTasks();
        if (!tasks.isEmpty()) {
            System.out.print("Enter the task number to mark as completed: ");
            int taskNumber = scanner.nextInt();
            scanner.nextLine(); // Consume the newline

            if (taskNumber > 0 && taskNumber <= tasks.size()) {
                Task task = tasks.get(taskNumber - 1);
                task.completed = true;
                System.out.println("Task '" + task.description + "' marked as completed.");
            } else {
                System.out.println("Invalid task number.");
            }
        }
    }
```

}

Task Scheduler

1. Add Task

2. View Tasks

3. Delete Task

4. Mark Task as Completed

5. Exit

Choose an option: 1

Enter task description: Complete Java assignment

Enter due time (HH:MM): 18:00

Task added successfully.

Task Scheduler

1. Add Task

2. View Tasks

3. Delete Task

4. Mark Task as Completed

5. Exit

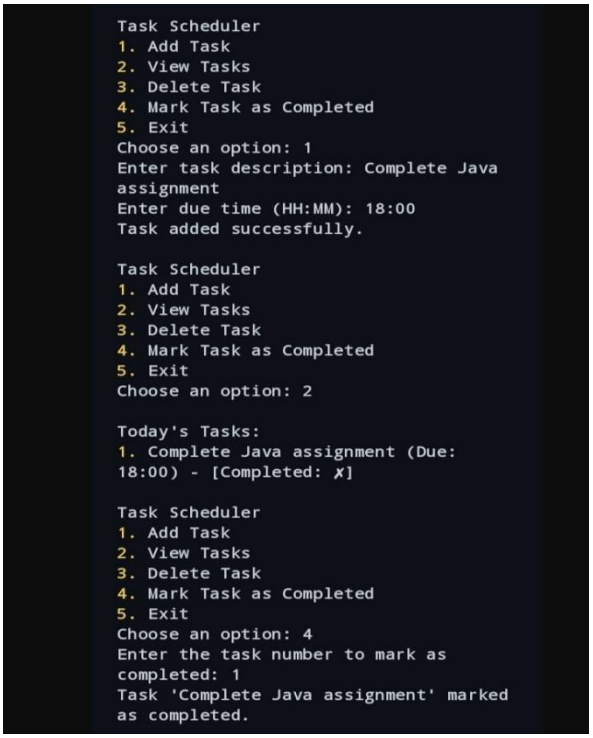
Choose an option: 2

Today's Tasks:

1. Complete Java assignment (Due: 18:00) - [Completed: ✗]

Task Scheduler

APPENDIX B
(SCREENSHOTS)



REFERENCES:

1. **Java Documentation (Oracle)**:
   - Oracle. "Java Platform, Standard Edition Documentation". Available at: https://docs.oracle.com/en/java/
   - Java API documentation for classes like ArrayList, Scanner, LocalTime, and String was essential for implementing core functionality.
2. **Effective Java (Book)**:
   - Bloch, Joshua. *Effective Java*, 3rd Edition. Addison-Wesley Professional, 2018.
   - This book provided guidance on best practices in Java programming, especially related to handling collections and input validation.
3. **Java Tutorials (Oracle)**:
   - Oracle. "The Java™ Tutorials". Available at: https://docs.oracle.com/javase/tutorial/
   - These tutorials were used as a reference for Java basics and more advanced topics, including input validation and working with time-related classes like LocalTime.
4. **Task Scheduling and Management**:
   - "Task Management Concepts" in *Productivity Blogs and Websites*. A collection of ideas and methods for improving task management systems and interfaces.
   - For improving the user experience and adding features like recurring tasks, task categories, and notification systems.
5. **Stack Overflow**:
   - Stack Overflow. "Java Programming Q&A". Available at: https://stackoverflow.com/
   - Used for troubleshooting specific Java issues, especially related to handling input validation, collections, and error handling in the task scheduler program.
6. **Java Tutorials - Input Validation**:
   - Available on sites such as GeeksforGeeks and W3Schools, where Java programming best practices, such as validating user input (like time formats), are discussed.
7. **Java Programming by Example (Book)**:
   - Balagurusamy, E. *Java Programming by Example*, 2nd Edition. McGraw-Hill Education, 2019.
   - Provided practical examples of using collections and handling user input and output in a command-line interface environment.