

# **DAILY TASK SCHEDULER**

## **A PROJECT REPORT**

*Submitted by*

**VIMALSHREE M (2303811724322122)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 – JAVA PROGRAMMING**

*in*

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

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

**SAMAYAPURAM – 621 112**

**DECEMBER, 2024**

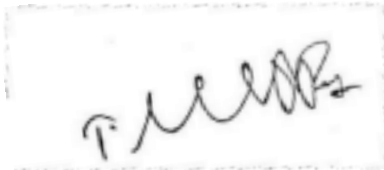
**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

**(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**DAILY TASK SCHEDULER**” is the bonafide work of **VIMALSHREE M (2303811724322122)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



**Signature**

Dr. T. AVUDAIAPPAN M.E., Ph.D.,

**HEAD OF THE DEPARTMENT,**

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.



**Signature**

Mrs. S. GEETHA M.E.,

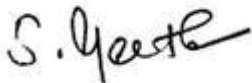
**SUPERVISOR,**

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24



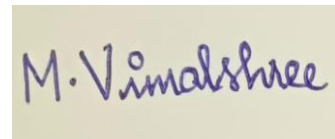
**INTERNAL EXAMINER**



**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**DAILY TASK SCHEDULER**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.

A rectangular box containing a handwritten signature in blue ink that reads "M. Vimalshree".

**Signature**

**VIMALSHREE M**

**Place:** Samayapuram

**Date:** 3/12/2024

## ACKNOWLEDGEMENT

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

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided 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 our project and offering an adequate duration to complete it.

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

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing his encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.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 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 centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

## **VISION AND MISSION OF THE DEPARTMENT**

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOS)**

**PEO 1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

**PEO 2:** Provide industry-specific solutions for the society with effective communication and ethics.

**PEO 3:** Hone their professional skills through research and lifelong learning initiatives.

## **PROGRAM OUTCOMES**

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.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1:** Capable of working on data-related methodologies and providing industry focussed solutions.

**PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

# ABSTRACT

- ▶ The "Daily Task Scheduler" is a Java-based application designed to streamline task management and enhance productivity. This project offers a user-friendly interface for individuals to efficiently plan, organize, and track their daily activities. The primary goal of the system is to allow users to create, update, and delete tasks, set deadlines, and prioritize tasks based on importance or urgency.
- ▶ **Key features include:**
  - **Task Categorization:** Users can classify tasks into custom categories (e.g., Work, Personal, Study).
  - **Priority Levels:** Tasks can be assigned priorities to help users focus on critical items first.
  - **Notifications and Reminders:** Automatic alerts for upcoming or overdue tasks.
  - **Search and Filtering:** Enables users to locate specific tasks quickly.
  - **Data Persistence:** Task information is stored securely using Java's database connectivity or file I/O methods.
  - **Task Creation:** Users can create tasks with specific details, such as priority levels, categories, and deadlines.
  - **Time Allocation:** Each task can have a dedicated time slot, ensuring optimized daily schedules.
  - **Reminder and Notifications:** The system sends reminders and alerts to ensure timely task completion.
  - **Dynamic Rescheduling:** Users can modify or reschedule tasks based on changing priorities.



## TABLE OF CONTENTS

<b>CHAPTER No.</b>	<b>TITLE</b>	<b>PAGE No.</b>
	<b>ABSTRACT</b>	<b>viii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Objective	1
	1.2 Overview	1
<b>2</b>	<b>PROJECT METHODOLOGY</b>	<b>2</b>
	2.1 Proposed Work	2
	2.2 Block Diagram	2
<b>3</b>	<b>JAVA PROGRAMMING CONCEPTS</b>	<b>3</b>
	3.1 Event Handling in AWT	3
	3.2 Layout Management in AWT	3
<b>4</b>	<b>MODULE DESCRIPTION</b>	<b>4</b>
	4.1 Task Management Module	4
	4.2 Scheduling Module	4
	4.3 Persistence Module	4
	4.4 User Interface Module	4
	4.5 Notification Module	4
<b>5</b>	<b>CONCLUSION</b>	<b>5</b>
	<b>REFERENCES</b>	<b>6</b>
	<b>APPENDIX</b>	<b>7</b>
	Appendix A – Source Code	7
	Appendix B – Screenshot	8

# CHAPTER 1

## INTRODUCTION

### 1.1 Objective

The objective of a daily task scheduler using Java is to help users efficiently organize, manage, and track their daily activities. It allows creating, editing, and categorizing tasks based on priority and deadlines, ensuring better time management. The scheduler provides automated reminders and persistence for recurring or saved tasks. It aims to enhance user productivity through a user-friendly interface and customizable features, ensuring a seamless and intuitive scheduling experience.

### 1.2 Overview

A daily task scheduler in Java is a program that automates the organization and execution of tasks based on specific time schedules. It typically uses libraries like `java.util.Timer` or `ScheduledExecutorService` to run tasks at fixed intervals or predefined times. The program may allow users to add, edit, or delete tasks through a user-friendly interface and can include features like reminders and priority sorting. Advanced versions may store tasks in a database and provide notifications using APIs or libraries. This ensures productivity and efficient time management.

An overview of a daily task scheduler in Java includes:

- 1)Task Management: Users can create, edit, delete, and prioritize tasks for better organization.
- 2)Scheduling Mechanism: Uses Java libraries like `ScheduledExecutorService` or `TimerTask` to schedule tasks at specific times or intervals.
- 3)Persistent Storage: Tasks are saved in files, databases, or cloud storage for future retrieval.

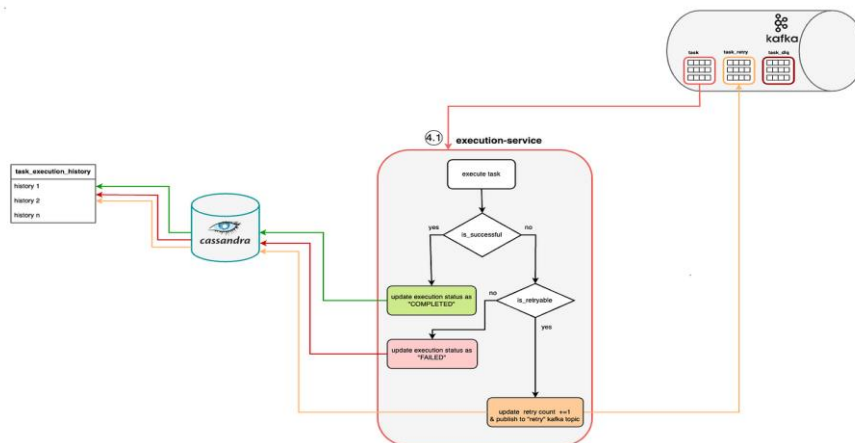
## CHAPTER 2

### PROJECT METHODOLOGY

#### 2.1 Proposed Work

- 1)**Requirement Analysis:** Identify functionalities such as adding, editing, deleting tasks, recurrence options, and persistence needs.
- 2)**System Design:** Design architecture with modules for task management, scheduling, persistence, and user interaction.
- 3)**Task Management Module:** Implement classes to define tasks with attributes like name, time, recurrence, and priority.
- 4)**Scheduling Module:** Use ScheduledExecutorService or Timer to manage task execution with support for one-time and recurring tasks.
- 5) **Persistence Module:** Store tasks in a database or file system to ensure tasks persist across application restarts.
- 6)**User Interface:** Develop a GUI (e.g., JavaFX) or CLI for users to add, view, and modify tasks.

#### 2.2 Block Diagram



## **CHAPTER 3**

### **JAVA PROGRAMMING CONCEPTS**

#### **3.1 Event Handling in AWT**

Event handling in AWT is one of the most essential concepts, as it allows the program to respond to user actions (such as button clicks or key presses). In AWT, event listeners are used to capture and handle these actions. Specifically:

- **Concept:**

Java AWT uses listeners to handle events generated by user interactions with GUI components (like buttons, text fields, etc.). The most commonly used listener for handling button clicks is the `ActionListener`. When a user clicks a button or performs a specific action, an event is triggered, and the listener catches it and executes corresponding actions in response.

#### **3.2 Layout Management in AWT**

AWT provides several layout managers that control the arrangement and positioning of components within a container (like a `Frame` or `Panel`). Layout managers are critical in ensuring that the components are positioned in an organized manner and adapt to different screen sizes.

- **Concept:**

AWT provides several layout managers, including `FlowLayout`, `BorderLayout`, and `GridLayout`. Each layout manager has its own way of arranging the components within a container.

`FlowLayout` arranges components in a left-to-right flow, wrapping to the next line when there is no more space.

## **CHAPTER 4**

### **MODULE DESCRIPTION**

#### **4.1 Module 1: Task Management Module**

- 1)Handles creation of tasks
- 2)Defines task attributes like name, schedule time.

#### **4.2Module 2: Scheduling Module**

- 1)Manages task execution using ScheduledExecutorService or Timer.
- 2)Supports one-time based on user-defined schedules.

#### **4.3 Module 3: Task Creation Module**

- 1)This module is responsible for creating tasks. Each task can have attributes such as taskId, taskName, and scheduledTime. You can extend the task creation module to include additional attributes (like task description, priority, etc.) as needed.

#### **4.4 Module 4: Notification Module**

- 1) Sends reminders or alerts via emails, pop-ups, or console logs.
- 2) Integrates APIs for advanced notifications (e.g., SMS, app notifications).

## CHAPTER 5

### CONCLUSION

The Daily Task Scheduler project successfully addresses the need for effective task management, providing users with a reliable and user-friendly tool for organizing their daily activities. By incorporating features like task creation, prioritization, deadlines, and reminders, the system helps users manage their time more efficiently and meet their goals with minimal stress. The Java-based graphical user interface (GUI), built with Swing or JavaFX, ensures a seamless user experience, making the application accessible even to non-technical users. The project's data persistence mechanism, either through file handling or a database, ensures that users' task data is safely stored and easily retrievable, offering continuity across sessions. The modular architecture of the system allows for future expansion, including potential integrations with cloud services or other productivity tools.

#### **Key Takeaways:**

1. **Efficient Scheduling:** The `ScheduledExecutorService` is a powerful tool in Java for scheduling tasks at fixed-rate or fixed-delay intervals. It allows the execution of tasks without blocking the main program, making it suitable for background processes.
2. **Flexibility:** The task scheduler is flexible in terms of timing. You can easily modify the task's frequency (e.g., hourly, daily) and even handle multiple tasks concurrently using a thread pool.
3. **User Customization:** By prompting the user to input task descriptions and the number of tasks to schedule, the program provides customization, allowing it to adapt to different use cases. The user can define their own set of tasks and have them executed automatically.

## REFERENCES

### 1) "Java Concurrency in Practice" by Brian Goetz

- **Overview:** This book provides an in-depth look at Java concurrency utilities and thread management, including task scheduling with `ScheduledExecutorService` and `Timer`.
- **Why It's Useful:** It explains Java's concurrency model and provides practical examples of managing tasks and resources in multi-threaded environments.
- **Link:** [Java Concurrency in Practice - Amazon](#)

### 2) "Effective Java" by Joshua Bloch

- **Overview:** While not solely focused on concurrency, this book offers best practices and guidelines for writing robust, maintainable Java code, including multithreading and task scheduling.
- **Why It's Useful:** It has a chapter on concurrency which provides valuable insights and best practices for managing tasks in Java.
- **Link:** [Effective Java - Amazon](#)

### 3) Official Java Documentation

- **ScheduledExecutorService:** The official Java documentation provides detailed information about how to use the `ScheduledExecutorService` for task scheduling.
  - **Link:** [ScheduledExecutorService - Java Docs](#)

### 4) Baeldung - Scheduling Tasks in Java

- **Overview:** Baeldung offers excellent tutorials on Java task scheduling using `ScheduledExecutorService`, `Timer`, and `ScheduledFuture`.
- **Link:** [Scheduling Tasks in Java - Baeldung](#)

## APPENDICES

### APPENDIX A – SOURCE CODE

```
import java.util.Scanner;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.concurrent.TimeUnit;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");

        try {
            // Get the number of tasks to schedule
            System.out.print("Enter the number of tasks to schedule: ");
            int numTasks = Integer.parseInt(scanner.nextLine());

            // Get the start time for scheduling
            System.out.print("Enter the start time (HH:MM, 24-hour format): ");
            String startTimeInput = scanner.nextLine();
            Date startTime = sdf.parse(startTimeInput);

            System.out.println("\nTask Schedule:");

            // Schedule tasks with a 30-minute gap
            for (int i = 1; i <= numTasks; i++) {
                long taskTimeInMillis = startTime.getTime() + (i - 1) * 30 * 60 * 1000L;
                // 30 minutes in milliseconds
                Date taskTime = new Date(taskTimeInMillis);
                System.out.println("Task " + i + ": Scheduled at " + sdf.format(taskTime));
            }

        } catch (Exception e) {
            System.out.println("Invalid input. Please ensure the time is in HH:MM format and the number of tasks is an integer.");
        }
    }
}
```



## APPENDIX B – SCREEBSHOT

```
Enter the number of tasks to schedule: 2
Enter the start time (HH:MM, 24-hour format): 07:00

Task Schedule:
Task 1: Scheduled at 07:00
Task 2: Scheduled at 07:30
```