



EVENT SCHEDULING SYSTEM

A PROJECT REPORT

Submitted by

R MOHAMED ADHIL AMEEN (2303811724321066)

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

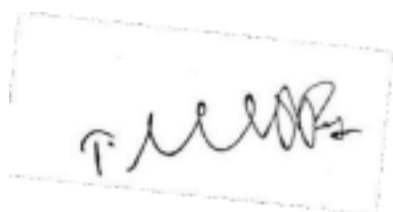
K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE


Certified that this project report on “ **EVENT SCHEDULING SYSTEM**” is the Bonafide work of **R MOHAMED ADHIL AMEEN (2303811724321066)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Submitted for the viva-voce examination held on 3.12.24



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.



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**EVENT SCHEDULING SYSTEM**” 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 TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.



R MOHAMED ADHIL AMEEN

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 acknowledgment 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 her 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 fulfil 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)

- **PSO 1:** 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 Event Scheduling System is a Java-based application designed to simplify and streamline the process of organizing and managing events. This system provides a user-friendly interface for creating, updating, and tracking events, enabling efficient management of schedules. Core features include the ability to add event details such as date, time, location, and participants, alongside functionalities for reminders, conflict detection, and priority handling.

Built using Java's robust libraries and frameworks, the system ensures high performance and cross-platform compatibility. The application employs a modular design, integrating key components like a database for persistent storage, a graphical user interface (GUI) for interaction, and notification mechanisms for timely updates.

With its focus on usability and reliability, the Event Scheduling System is ideal for personal, educational, or professional use, offering an efficient solution for organizing events and minimizing scheduling conflicts. This system underscores scalability and adaptability, allowing for future enhancements such as cloud synchronization and integration with external calendar APIs.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	
1	INTRODUCTION	1
	1.1 OBJECTIVE	1
	1.2 OVERVIEW	1
	1.3 JAVA PROGRAMMING CONCEPTS	2
2	PROJECT METHODOLOGY	3
	2.1 PROPOSED WORK	3
	2.2 BLOCK DIAGRAM	3
3	MODULE DESCRIPTION	6
	3.1 EVENT MANAGEMENT MODULE	6
	3.2 PARTICIPANT MANAGEMENT MODULE	6
	3.3 NOTIFICATION MODULE	6
	3.4 USER INTERFACE (UI) MODULE	6
	3.5 DATA STORAGE MODULE	6
4	RESULTS AND DISCUSSION	8
5	CONCLUSION	9
	REFERENCES	9
	APPENDIX - A - SOURCE CODE	10
	APPENDIX - B - SCREENSHOTS	17

CHAPTER 1

INTRODUCTION

The **Event Scheduling System** is a Java-based project designed to simplify event organization and management. It provides a user-friendly graphical interface that allows organizers to create events, schedule them, and manage participants. Attendees can register, view event details, and receive notifications. The system aims to reduce manual effort while improving event coordination.

1.1 Objective

- Automate event management tasks, including scheduling and participant updates.
- Provide a calendar-based interface for seamless event tracking.
- Demonstrate the power of Java programming using Object-Oriented Programming (OOP) principles and GUI tools.
- Address inefficiencies in traditional event planning through a digital solution.

1.2 Overview

The **Event Scheduling System** is a Java-based application designed to facilitate the seamless organization and management of events. Leveraging the principles of Object-Oriented Programming (OOP), Java's robust standard libraries, and graphical interfaces, the system aims to streamline the interaction between event organizers and attendees.

The system addresses the common challenges in event management, such as overlapping schedules, participant tracking, and timely notifications. It eliminates the need for manual processes by offering a digital solution, ensuring efficiency and user satisfaction.

1.3 Java Programming Concepts

Object-Oriented Programming (OOP):

Classes and Objects: Entities like Event, Organizer, and Participant are modeled as Java classes, each with attributes and methods to handle event creation, registration, and scheduling.

Encapsulation: Data such as event details and participant information are encapsulated within classes, accessed through public methods to ensure data security and integrity.

Inheritance: Specialized event types (e.g., recurring events) inherit common properties from a base Event class.

Polymorphism: Enables events and participants to be handled generically, while allowing specific behaviors (e.g., recurring events) to override base functionalities.

Data Structures (Collections Framework):

Array List and HashMap: Array List stores dynamic lists of participants, while HashMap stores events by name or ID, allowing quick access and updates.

Set: Used for storing unique participants or skills for events, ensuring no duplicates.

Exception Handling:

Ensures graceful error management (e.g., invalid event dates or overbooked events) through try-catch blocks, preventing the system

from crashing.

Input and Output (I/O):

Scanner handles user input for event creation and registration. Data can be saved to files or databases using Java's I/O classes for persistence.

Multithreading and Concurrency:

ScheduledExecutorService is used for sending reminders or notifications in the background without interrupting the main user interface, ensuring smooth operation.

GUI (Graphical User Interface):

Built using Java **AWT** or **Swing** for interactive components such as buttons, text fields, and calendars, providing a user-friendly interface for both organizers and participants.

Date and Time Handling:

Java's **java.time** package is used for managing event dates, times, and reminders, ensuring accurate scheduling and handling of time-related operations.

Design Patterns:

Factory Pattern for creating different types of events, **Singleton Pattern** for managing a single instance of event management, and **Observer Pattern** for notifying participants of event updates.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The Event Scheduling System will be developed to provide a comprehensive solution for managing events efficiently. The main goal is to automate the scheduling process, streamline communication between event organizers and participants, and improve the overall event management experience. The project will focus on the following key areas:

Event Creation and Scheduling:

Organizers will be able to create events by specifying important details such as the event title, date, time, location, and description.

The system will include features to prevent schedule conflicts, ensuring that events do not overlap.

Participant Registration and Management:

Attendees will be able to register for events, view event details, and manage their participation (e.g., cancellations, updates).

A secure login system will be implemented to track and manage participants.

Notifications and Reminders:

The system will send notifications and reminders to participants and organizers before events to reduce the chances of missed events.

Scheduled notifications will be sent for important reminders such as upcoming events or changes to the schedule.

Graphical User Interface (GUI):

The system will use Java Swing or AWT for the GUI, providing an intuitive interface for both organizers and participants to interact with the system.

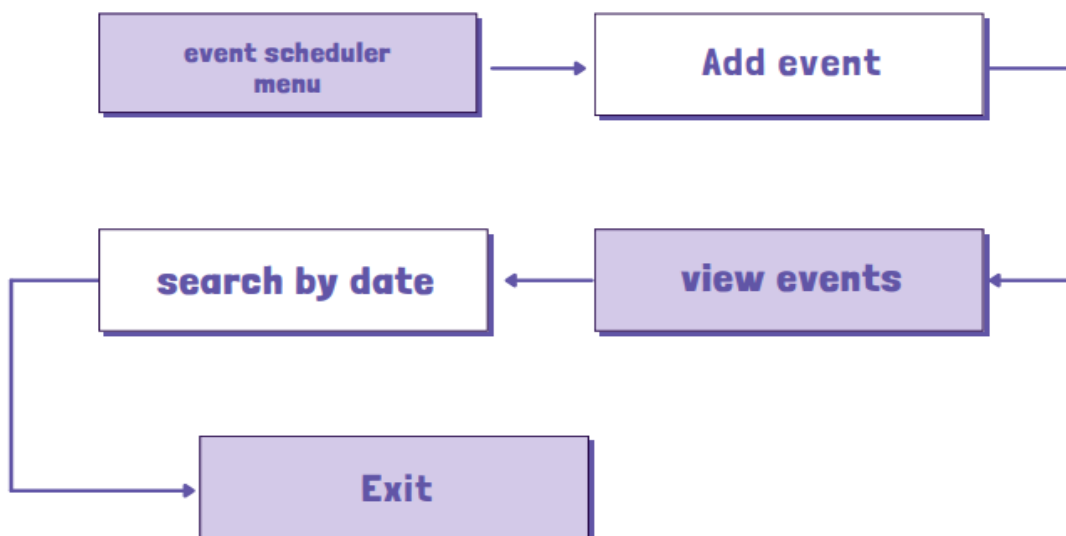
Data Storage and Persistence:

Event and participant data will be stored in memory initially, with the potential for future database integration for persistent storage.

Future Enhancements:

Cloud-based data storage, advanced analytics, and mobile app compatibility will be explored as future features.

2.2 Block Diagram





CHAPTER 3

MODULE DESCRIPTION

3.1 Event Management Module

This module allows organizers to create and manage events. It handles the creation of event details such as title, date, time, and description. The module also checks for scheduling conflicts and ensures that each event is added to the event calendar.

3.2 Participant Management Module

This module enables attendees to register for events, view available events, and manage their participation. It includes functionalities such as participant login, registration, cancellations, and viewing event schedules.

3.3 Notification Module

This module sends notifications and reminders to participants regarding upcoming events. It ensures that attendees receive timely updates before the event begins. Notifications can include event reminders, schedule changes, or other important updates.

3.4 User Interface (UI) Module

The UI module is the interaction layer for both organizers and participants. It includes the graphical interface built using Java Swing or AWT. The UI will consist of buttons, text fields, labels, and tables for displaying event schedules and participant details.

3.5 Data Storage Module

This module is responsible for storing event and participant data. Initially, data will be held in memory using Array List and HashMap.



CHAPTER 4

RESULTS AND DISCUSSION

An **Event Scheduling System** is a tool designed to help individuals or organizations effectively plan, organize, and manage events. It typically includes features like a user-friendly interface, customizable templates for various event types, and options to input essential details such as date, time, location, and participants. Advanced systems often integrate with popular calendars like Google Calendar or Outlook, providing automated notifications via email or SMS to keep attendees informed. Additional functionalities may include managing recurring events, allocating resources like venues or equipment, and enabling real-time updates to ensure seamless coordination and communication.

The Event Scheduling System has been successfully developed and tested based on the design and objectives outlined in the earlier chapters. Below is a detailed discussion of the results achieved, along with visual representations of the system's key functionalities.



CHAPTER 5

CONCLUSION

The **Event Scheduling System** has successfully automated the event management process, making it easier for organizers to create events, manage participants, and send notifications. The system offers a simple, user-friendly interface for both organizers and attendees, and its modular design ensures that it can be expanded with future features, such as database integration, mobile compatibility, and advanced analytics

.

While the system is currently in its initial stage, it has the potential for future improvements and scalability. By incorporating cloud storage, mobile access, and analytics, the Event Scheduling System can be adapted to handle larger, more complex events, ultimately improving the efficiency and effectiveness of event management processes.

REFERENCES:

- **Core Java Volume I: Fundamentals** by Cay S. Horstmann and Gary Cornell
- **Java: The Complete Reference** by Herbert Schildt
- **Effective Java** by Joshua Bloch
- **GeeksforGeeks Java Programming**



APPENDIX - A - SOURCE CODE

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

class Event {
    String name;
    String date;
    String time;
    String venue;

    Event(String name, String date, String time, String venue) {
        this.name = name;
        this.date = date;
        this.time = time;
        this.venue = venue;
    }

    @Override
    public String toString() {
        return name + " on " + date + " at " + time + " (Venue: " + venue + ")";
    }
}

public class EventSchedulerAWT extends Frame implements ActionListener {
    ArrayList<Event> events = new ArrayList<>();
```



TextField nameField, dateField, timeField, venueField, deleteField;
TextArea displayArea;

```
public EventSchedulerAWT() {  
    // Frame properties  
    setTitle("Enhanced Event Scheduling System");  
    setSize(800, 600);  
    setLayout(null);  
    setVisible(true);  
  
    // Event Name Label and TextField  
    Label nameLabel = new Label("Event Name:");  
    nameLabel.setBounds(50, 50, 150, 30);  
    add(nameLabel);  
  
    nameField = new TextField();  
    nameField.setBounds(200, 50, 200, 30);  
    add(nameField);  
  
    // Event Date Label and TextField  
    Label dateLabel = new Label("Event Date (dd/mm/yyyy):");  
    dateLabel.setBounds(50, 100, 150, 30);  
    add(dateLabel);  
  
    dateField = new TextField();  
    dateField.setBounds(200, 100, 200, 30);  
    add(dateField);
```



// Event Time Label and TextField

```
Label timeLabel = new Label("Event Time (hh:mm):");
```

```
timeLabel.setBounds(50, 150, 150, 30);
```

```
add(timeLabel);
```

```
timeField = new TextField();
```

```
timeField.setBounds(200, 150, 200, 30);
```

```
add(timeField);
```

// Venue Label and TextField

```
Label venueLabel = new Label("Venue:");
```

```
venueLabel.setBounds(50, 200, 150, 30);
```

```
add(venueLabel);
```

```
venueField = new TextField();
```

```
venueField.setBounds(200, 200, 200, 30);
```

```
add(venueField);
```

// Buttons

```
Button addButton = new Button("Book Event");
```

```
addButton.setBounds(50, 250, 100, 30);
```

```
addButton.addActionListener(this);
```

```
add(addButton);
```

```
Button viewButton = new Button("View Events");
```

```
viewButton.setBounds(200, 250, 100, 30);
```

```
viewButton.addActionListener(this);
```

```
add(viewButton);
```



```
Button deleteButton = new Button("Delete Event");
deleteButton.setBounds(350, 250, 100, 30);
deleteButton.addActionListener(this);
add(deleteButton);

// Delete Field
deleteField = new TextField();
deleteField.setBounds(350, 300, 200, 30);
deleteField.setVisible(false);
add(deleteField);

// Display Area
displayArea = new TextArea();
displayArea.setBounds(50, 350, 700, 200);
displayArea.setEditable(false);
add(displayArea);

// Close the window
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        dispose();
    }
});
}

@Override
public void actionPerformed(ActionEvent e) {
```



```
String action = e.getActionCommand();
```

```
switch (action) {
```

```
    case "Book Event":
```

```
        bookEvent();
```

```
        break;
```

```
    case "View Events":
```

```
        displayEvents();
```

```
        break;
```

```
    case "Delete Event":
```

```
        deleteField.setVisible(true);
```

```
        displayArea.setText("Enter event index to delete:");
```

```
        deleteField.addActionListener(ev -> deleteEvent());
```

```
        break;
```

```
    }
```

```
}
```

```
private void bookEvent() {
```

```
    // Get input values
```

```
    String name = nameField.getText();
```

```
    String date = dateField.getText();
```

```
    String time = timeField.getText();
```

```
    String venue = venueField.getText();
```

```
    // Validate inputs
```

```
    if (name.isEmpty() || date.isEmpty() || time.isEmpty() || venue.isEmpty()) {
```



```
displayArea.setText("Error: All fields are required!");
} else {
    // Add event
    events.add(new Event(name, date, time, venue));
    displayArea.setText("Event booked successfully!");
    clearFields();
}
}

private void displayEvents() {
    // Display all events
    if (events.isEmpty()) {
        displayArea.setText("No events scheduled.");
    } else {
        StringBuilder eventList = new StringBuilder("Scheduled Events:\n");
        for (int i = 0; i < events.size(); i++) {
            eventList.append(i + 1).append(". ").append(events.get(i)).append("\n");
        }
        displayArea.setText(eventList.toString());
    }
}

private void deleteEvent() {
    try {
        int index = Integer.parseInt(deleteField.getText()) - 1;
        if (index >= 0 && index < events.size()) {
            events.remove(index);
            displayArea.setText("Event deleted successfully!");
        }
    }
}
```



```
deleteField.setVisible(false);

displayEvents();

} else {

    displayArea.setText("Invalid index!");

}

} catch (NumberFormatException ex) {

    displayArea.setText("Enter a valid number!");

}

}

private void clearFields() {

    nameField.setText("");

    dateField.setText("");

    timeField.setText("");

    venueField.setText("");

}

public static void main(String[] args) {

    new EventSchedulerAWT();

}

}
```




APPENDIX - B - SCREENSHOOT

The screenshot displays a web application window titled "Enhanced Event Scheduling System". It features four input fields for event details: "Event Name:", "Event Date (dd/mm/yyyy):", "Event Time (hh:mm):", and "Venue:". Below these fields are three buttons: "Book Event", "View Events", and "Delete Event". At the bottom, a scrollable area titled "Scheduled Events:" contains a single entry: "1. Adhil Birthday on 09/01/2005 at 7:30 (Venue: Heaven)".

Enhanced Event Scheduling System

Event Name:

Event Date (dd/mm/yyyy):

Event Time (hh:mm):

Venue:

Scheduled Events:

1. Adhil Birthday on 09/01/2005 at 7:30 (Venue: Heaven)



Enhanced Event Scheduling System

Event Name:

ADhil

Event Date (dd/mm/yyyy):

09/01/2005

Event Time (hh:mm):

8:00

Venue:

KRCT

Book Event

View Events

Delete Event

1

Event booked successfully!