

EVENT SCHEDULING SYSTEM



A PROJECT REPORT

Submitted by

NIVEDHA R(2303811710422109)

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-621112

NOVEMBER- 2024

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM-621112

BONAFIDE CERTIFICATE

Certified that this project report on **“EVENT SCHEDULING SYSTEM”** is the bonafide work of **NIVEDHA R (2303811710422109)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

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.

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E., (Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mrs.K. Valli Priyadharshini, M.E., (Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 03.12.2024

CGB1201-JAVA PROGRAMMING
Mr.MANJUNATH A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

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 ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING**.



NIVEDHA R

Place: Samayapuram

Date: 03/12/2024

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 adequateduration 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 **Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patiencewhich 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 **Event Scheduling System** is a Java-based application developed to facilitate the creation, management, and registration of events. It provides a graphical user interface (GUI) built using the Abstract Window Toolkit (AWT), ensuring a user-friendly experience for both event organizers and attendees. The system consists of multiple components designed to streamline the event lifecycle, from creation to registration. The application incorporates two primary user roles: organizers and attendees. Organizers can create events by specifying details such as the title, date, time, location, description, and maximum capacity. These events are stored and managed using the EventManager class, which acts as a central repository for all event-related data. Organizers can view a summary of all created events, including information about available spots and registered attendees. Attendees can browse available events and register for those of interest. Registration functionality ensures that the event's capacity is respected, preventing overbooking. The system provides real-time updates to attendees about the status of their registration, including confirmation of successful registration or notification if an event is full. Both organizers and attendees are implemented as extensions of a generic User class, encapsulating shared attributes such as name and email.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The Event Scheduling System is a Java-based application that enables event organizers to create and manage events, while attendees can register for them. It includes a central Event Manager module to handle event listings and registrations. The system's modular structure ensures efficient event coordination and user interaction.	PO1 -3 PO2 -3 PO3 -3 PO5 -3 PO6 -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 NO.	TITLE	PAGE NO.
	ABSTRACT	viii
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	4
	3.1 User Module	4
	3.2 Event Module	4
	3.3 Event Manager Module	4
	3.4 GUI Module	5
	3.5 Organizer Module	5
	3.6 Attendee Module	5
4	CONCLUSION & FUTURE SCOPE	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	REFERENCES	22
	APPENDIX A (SOURCE CODE)	8
	APPENDIX B (SCREENSHOTS)	20

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this Event Scheduling System is to simplify the process of event creation, management, and registration for both organizers and attendees. It allows organizers to create events with details such as title, date, location, and capacity, while providing real-time updates on registration status. Attendees can easily browse and register for available events, with the system ensuring that event capacities are respected. The application aims to provide a user-friendly interface for both organizers and attendees, promoting an efficient and seamless event management experience.

1.2 Overview

The Event Scheduling System is a Java-based application designed to facilitate the creation and registration of events through a user-friendly graphical interface. Organizers can create events by providing essential details such as title, date, time, location, and capacity, with the system managing and storing event information. Attendees can browse available events and register based on their preferences, with real-time updates on registration status. The system ensures that event capacities are adhered to, preventing overbooking. It features two main user roles, organizers and attendees, and offers a seamless transition between event creation and attendee registration. The application simplifies event management, making it an efficient solution for both event organizers and participants.

1.3 Java Programming Concepts

- **Encapsulation:** Data is encapsulated within each class, meaning that the internal state of objects is protected from unauthorized access by using private fields and providing public getter and setter methods. For example, getTitle() and getCapacity() are public methods that provide access to the private fields of the Event class.
- **Inheritance:** The Organizer and Attendee classes inherit from the User class, demonstrating inheritance. Both Organizer and Attendee have common properties like name and email, which are inherited from the User class, reducing redundancy.
- **Exception handling:** The program uses basic exception handling for input validation. For instance, when an organizer enters the event date or time, the program ensures the format is correct and handles potential parsing errors with a try-catch block, notifying the user of invalid input.
- **Graphical User Interface (GUI):** The system uses AWT (Abstract Window Toolkit) to build a simple graphical user interface. Components like Frame, Label, TextField, Button, and TextArea are used to display and accept input from users.
- **Event handling:** The program uses event handling through listeners (ActionListener) to manage user actions like button clicks. For example, when the "Create Event" button is clicked, an event listener triggers the code that creates a new event and updates the display.
- **Date and Time handling:** The program makes use of Java's LocalDateTime and DateTimeFormatter from the java.time package to handle date and time, ensuring that event dates and times are parsed correctly and formatted in a consistent manner.

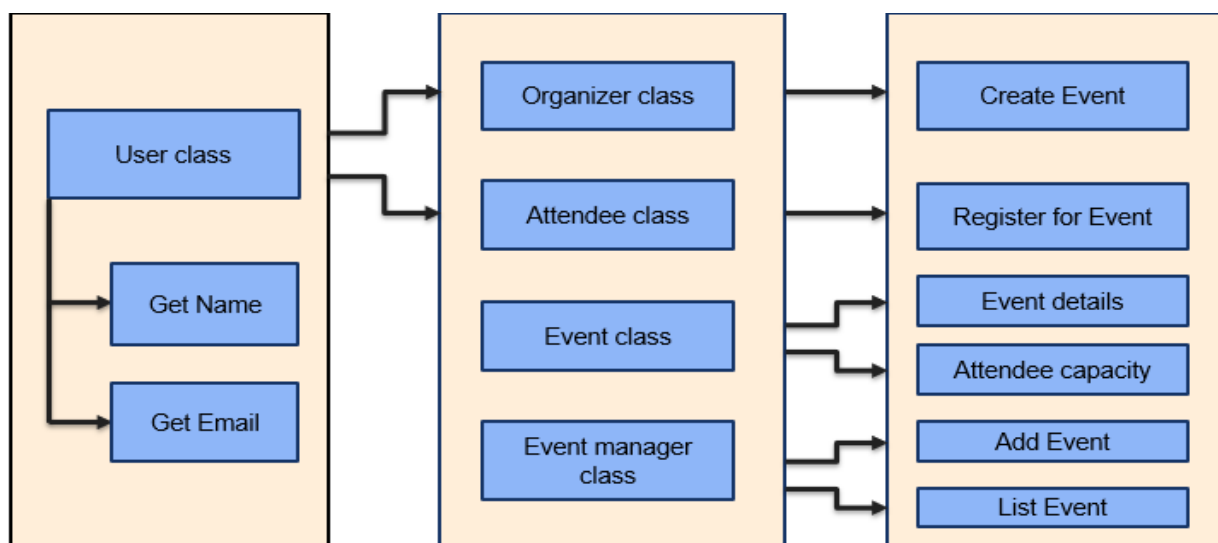
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed methodology for developing the Event Scheduling System follows a structured approach. The process begins with gathering the requirements, focusing on the needs of event organizers and attendees. Organizers must be able to create and manage events, while attendees should be able to browse and register for events. The system is designed using an object-oriented approach, where key entities such as users, events, and registrations are modeled as classes. The User class is extended by both Organizer and Attendee classes to handle specific behaviors, like event creation and registration. The Event class manages event details and registration. A graphical user interface (GUI) using Java's AWT framework is implemented to allow interaction. The system is then developed, tested for functional and usability requirements, and deployed for use, ensuring that it is intuitive, user-friendly, and efficient in managing events.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 User Module:

This module defines the User class, which serves as the base class for both organizers and attendees. It encapsulates common properties like name and email, providing basic attributes and methods for user management. The Organizer and Attendee classes inherit from this module, extending it with role-specific functionality.

3.2 Event Module:

This module represents events through the Event class, which includes properties such as title, date, time, location, description, and capacity. It allows the creation and management of events, and supports the registration of attendees while ensuring the event does not exceed its capacity.

3.3 Event Manager Module:

The EventManager class acts as the core of the system, managing multiple events. It provides methods to add events, list all events, and facilitate interactions between the Organizer and the Attendee classes.

3.4 GUI Module:

This module implements the graphical user interface (GUI) using Java AWT components such as Frame, Label, Textfield, TextArea, and Button. It includes two primary frames: the Organizer frame for event creation and management, and the User frame for attendee registration. The GUI handles user interactions, displays event information, and facilitates smooth transitions between different tasks.

3.5 Organizer Module:

This module is responsible for managing the creation of events. The Organizer class can create new events, specify details like date, time, location, and capacity, and add the events to the event manager for storage and management.

3.6 Attendee Module:

This module handles the functionality for attendees to register for events. The Attendee class allows users to select an event, register for it, and check the availability of spots. It communicates with the Event class to ensure that the attendee is successfully registered or notified if the event is full.

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

The Event Scheduling System successfully provides a comprehensive solution for organizing and managing events, catering to both event organizers and attendees. Through the use of object-oriented programming concepts such as inheritance, encapsulation, and polymorphism, the system is modular, easy to maintain, and scalable. Organizers can create and manage events with detailed attributes, while attendees can browse and register for events seamlessly. The system ensures that event capacities are respected, preventing over-registration. Additionally, the implementation of a user-friendly graphical interface makes the system accessible and easy to navigate for both user roles. Overall, the program offers an efficient and effective method for managing events, enhancing the experience for both organizers and attendees by automating and streamlining the event creation and registration processes. The system is a practical solution to address the complexities of event management in real-world applications.

4.2 FUTURE SCOPE

The future scope of the Event Management System includes several enhancements to improve functionality and user experience. First, adding user authentication and authorization would provide secure access for organizers and attendees, enabling personalized profiles and event management. Integrating online payment systems for paid events would streamline the registration process and facilitate transactions. Notifications and reminders for upcoming events via email or SMS would enhance user engagement and reduce no-shows. Additionally, advanced features such as event categorization, tagging, and better search functionalities would make it easier for attendees to find relevant events. Developing a mobile application would provide users with on-the-go access, while social media integration could help increase event visibility and reach.

Lastly, supporting multiple organizers and introducing an administrative module would allow for better management of large-scale events and provide enhanced control over the platform. These improvements would make the system more comprehensive and suitable for a wider range of event types.

APPENDIX A

```
import java.awt.*;
import java.awt.event.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
```

```
// User Class
```

```
class User {
    private String name;
    private String email;

    public User(String name, String email) {
        this.name = name;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }
}
```

// Event Class

```
class Event {  
    private String title;  
    private LocalDateTime dateTime;  
    private String location;  
    private String description;  
    private int capacity;  
    private List<Attendee> attendees;  
  
    public Event(String title, LocalDateTime dateTime, String location, String description, int  
capacity) {  
        this.title = title;  
        this.dateTime = dateTime;  
        this.location = location;  
        this.description = description;  
        this.capacity = capacity;  
        this.attendees = new ArrayList<>();  
    }  
  
    public boolean registerAttendee(Attendee attendee) {  
        if (attendees.size() < capacity) {  
            attendees.add(attendee);  
            return true;  
        }  
        return false;  
    }  
  
    public void displayEventInfo(TextArea textArea) {  
        textArea.append("Event Title: " + title + "\n");  
    }  
}
```

```

        textArea.append("Date and Time: " + dateTime + "\n");
        textArea.append("Location: " + location + "\n");
        textArea.append("Description: " + description + "\n");
        textArea.append("Capacity: " + capacity + "\n");
        textArea.append("Registered Attendees: " + attendees.size() + "/" + capacity + "\n");
        textArea.append("Spots Available: " + (capacity - attendees.size()) + "\n\n");
    }

    public String getTitle() {
        return title;
    }

    public int getCapacity() {
        return capacity;
    }

    public int getRegisteredCount() {
        return attendees.size();
    }
}

// Organizer Class
class Organizer extends User {
    public Organizer(String name, String email) {
        super(name, email);
    }

    public void createEvent(EventManager eventManager, String title, LocalDateTime dateTime,
String location, String description, int capacity) {

```

```
        Event event = new Event(title, dateTime, location, description, capacity);
        eventManager.addEvent(event);
    }
}
```

// Attendee Class

```
class Attendee extends User {
    public Attendee(String name, String email) {
        super(name, email);
    }

    public boolean registerForEvent(Event event) {
        return event.registerAttendee(this);
    }
}
```

// EventManager Class

```
class EventManager {
    private List<Event> events;

    public EventManager() {
        events = new ArrayList<>();
    }

    public void addEvent(Event event) {
        events.add(event);
    }
}
```

```

public List<Event> getEvents() {
    return events;
}

public void listEvents(TextArea textArea) {
    for (Event event : events) {
        event.displayEventInfo(textArea);
    }
}
}

// Organizer Frame - For Event Creation
class OrganizerFrame extends Frame {
    private EventManager eventManager;
    private TextArea eventDisplayArea;
    private TextField titleField, dateField, timeField, locationField, descriptionField, capacityField;
    private UserFrame userFrame;

    public OrganizerFrame(EventManager eventManager) {
        this.eventManager = eventManager;
        setTitle("Organizer - Create Event");
        setSize(400, 500);
        setLayout(new FlowLayout());

        // Create components
        Label titleLabel = new Label("Event Title:");
        titleField = new TextField(20);

```

```
Label dateLabel = new Label("Event Date (YYYY-MM-DD):");  
dateField = new TextField(10);
```

```
Label timeLabel = new Label("Event Time (HH:MM):");  
timeField = new TextField(5);
```

```
Label locationLabel = new Label("Location:");  
locationField = new TextField(20);
```

```
Label descriptionLabel = new Label("Description:");  
descriptionField = new TextField(20);
```

```
Label capacityLabel = new Label("Capacity:");  
capacityField = new TextField(5);
```

```
Button createButton = new Button("Create Event");  
Button proceedToUserButton = new Button("Proceed to User Registration");
```

```
eventDisplayArea = new TextArea(10, 50);
```

```
// Add components to frame
```

```
add(titleLabel);  
add(titleField);  
add(dateLabel);  
add(dateField);  
add(timeLabel);  
add(timeField);  
add(locationLabel);
```

```

add(locationField);
add(descriptionLabel);
add(descriptionField);
add(capacityLabel);
add(capacityField);
add(createButton);
add(eventDisplayArea);
add(proceedToUserButton);

// Event handler for createButton
createButton.addActionListener(e -> {
    String title = titleField.getText();
    String dateInput = dateField.getText();
    String timeInput = timeField.getText();
    String location = locationField.getText();
    String description = descriptionField.getText();
    int capacity = Integer.parseInt(capacityField.getText());

    // Convert date and time to LocalDateTime with validation
    try {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        LocalDateTime dateTime = LocalDateTime.parse(dateInput + " " + timeInput, formatter);

        // Create the event
        Organizer organizer = new Organizer("Organizer", "organizer@example.com");
        organizer.createEvent(eventManager, title, dateTime, location, description, capacity);

        // Display events

```



```

        eventDisplayArea.setText("");
        eventManager.listEvents(eventDisplayArea);
    } catch (Exception ex) {
        eventDisplayArea.append("Invalid date/time format. Please use YYYY-MM-DD and
HH:MM format.\n");
    }
});

```

```

// Event handler for proceedToUserButton
proceedToUserButton.addActionListener(e -> {
    if (userFrame == null) {
        userFrame = new UserFrame(eventManager);
    }
    userFrame.updateEventList();
    userFrame.setVisible(true);
});

```

```

// Frame setup
setVisible(true);
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});
}
}

```

```

// User Frame - For Event Registration
class UserFrame extends Frame {

```

```

private EventManager eventManager;
private TextArea eventDisplayArea;
private TextField nameField, emailField;
private Choice eventChoice;

public UserFrame(EventManager eventManager) {
    this.eventManager = eventManager;
    setTitle("User - Register for Events");
    setSize(400, 400);
    setLayout(new FlowLayout());

    // Create components
    Label nameLabel = new Label("Your Name:");
    nameField = new TextField(20);

    Label emailLabel = new Label("Your Email:");
    emailField = new TextField(20);

    Button registerButton = new Button("Register for Event");

    eventChoice = new Choice(); // Drop-down to choose an event
    eventDisplayArea = new TextArea(10, 50);

    // Add components to frame
    add(nameLabel);
    add(nameField);
    add(emailLabel);
    add(emailField);

```

```

add(new Label("Choose Event:"));
add(eventChoice);
add(registerButton);
add(eventDisplayArea);

// Event handler for registerButton
registerButton.addActionListener(e -> {
    String attendeeName = nameField.getText();
    String attendeeEmail = emailField.getText();
    Attendee attendee = new Attendee(attendeeName, attendeeEmail);

    // Get selected event
    String selectedEventTitle = eventChoice.getSelectedItemAt();
    Event selectedEvent = null;
    for (Event event : eventManager.getEvents()) {
        if (event.getTitle().equals(selectedEventTitle)) {
            selectedEvent = event;
            break;
        }
    }

    if (selectedEvent != null) {
        boolean registered = attendee.registerForEvent(selectedEvent);
        if (registered) {
            eventDisplayArea.append("Successfully registered for the event.\n");
        } else {
            eventDisplayArea.append("Registration failed. Event may be full.\n");
        }
    }
}

```

```

        // Refresh event display

        eventDisplayArea.append("Spots Available: " + (selectedEvent.getCapacity() -
selectedEvent.getRegisteredCount()) + "\n\n");

    } else {

        eventDisplayArea.append("No event selected or event not found.\n");

    }

});

// Frame setup
setVisible(false);
addWindowListener(new WindowAdapter() {

    public void windowClosing(WindowEvent we) {

        System.exit(0);

    }

});

}

public void updateEventList() {

    eventChoice.removeAll();

    for (Event event : eventManager.getEvents()) {

        eventChoice.add(event.getTitle());

    }

}

}

// Main class to run the application
public class EventManagerApp {

    public static void main(String[] args)

```

```
EventManager eventManager = new EventManager();
```

```
    // Create Organizer Frame for creating events
```

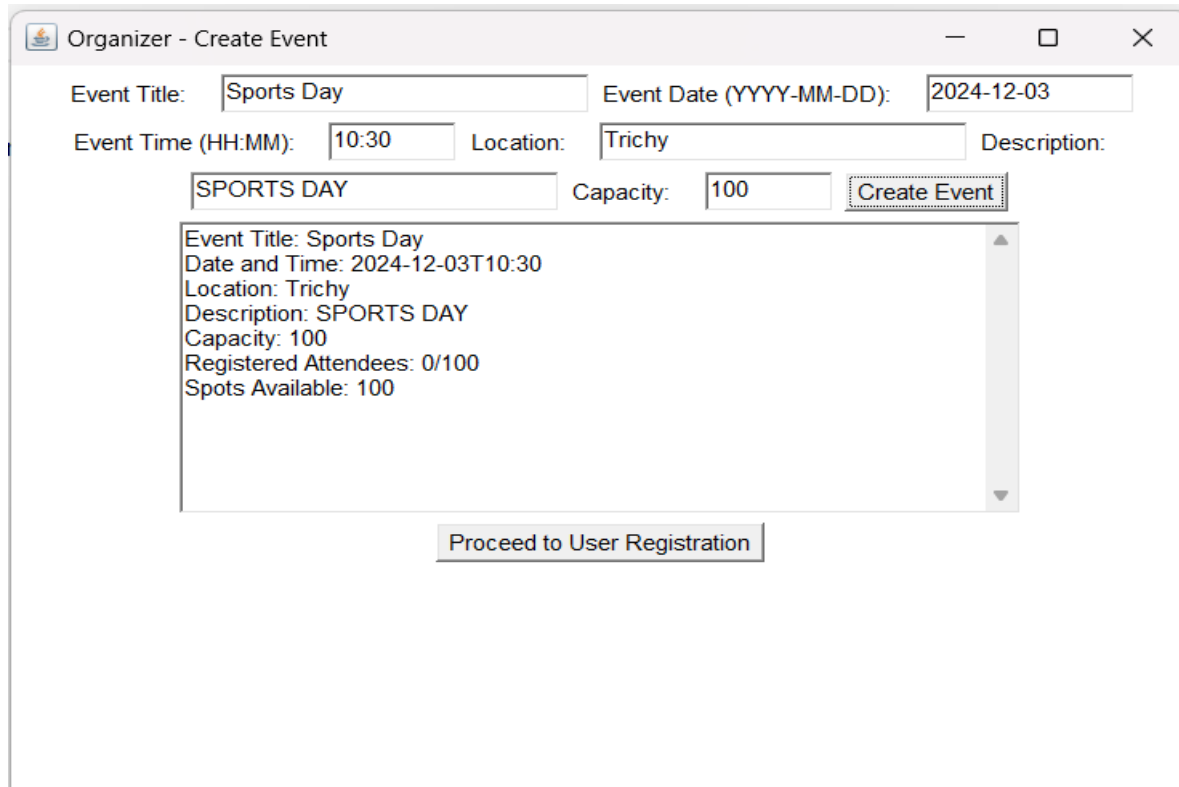
```
    new OrganizerFrame(eventManager);
```

```
}
```

```
}
```

APPENDIX B

Event Creation:



The screenshot shows a window titled "Organizer - Create Event" with standard window controls. The form contains the following fields and values:

- Event Title: Sports Day
- Event Date (YYYY-MM-DD): 2024-12-03
- Event Time (HH:MM): 10:30
- Location: Trichy
- Description: SPORTS DAY
- Capacity: 100

A "Create Event" button is located to the right of the Capacity field. Below the input fields is a scrollable list box containing the following summary information:

- Event Title: Sports Day
- Date and Time: 2024-12-03T10:30
- Location: Trichy
- Description: SPORTS DAY
- Capacity: 100
- Registered Attendees: 0/100
- Spots Available: 100

At the bottom of the window is a "Proceed to User Registration" button.

User Registration for Event:

User - Register for Events

Your Name: NIVEDHA

Your Email: nivedha@gmail.com

Choose Event:

Sports Day

Register for Event

Successfully registered for the event.
Spots Available: 99

REFERENCES

WEBSITES

- <https://www.javatpoint.com/java-awt>
- <https://www.geeksforgeeks.org/java-awt-tutorial//>
- <https://docs.oracle.com/javase/8/docs/technotes/guides/awt/->

YOUTUBE LINK

- <https://youtu.be/YSpqHOwYrk4?si=h41zciRqqMjVZiwX>
- <https://youtu.be/M03qEx6BYkM?si=taaFKBUGM2yyXdUQ>

BOOK

- Java: "The Complete Reference" by Herbert Schildt
- "Java Swing" by Robert Eckstein, Marc Loy, Dave Wood