EVENT SCHEDULING SYSTEM

A PROJECT REPORT

Submitted by

THARUNIYA T (2303811710422171)

In partial fulfillment of requirements for the award of the course

CGB1201-JAVA PROGRAMMING

In

COMPUTERSCIENCE ANDENGINEERING

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 **THARUNIYA T (2303811710422171)** 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 [illegible]	CGB1201-JAVA PROGRAMMING Mr.MALARMANNAN A, M.E., ASSISTANTP ROFESSOR [illegible]
SIGNATURE	**SIGNATURE**
Mrs.A.Delphin Carolina Rani,M.E.,Ph.D.,	Mr. A. Malarmannan ,M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K. Ramakrishnan College of Technology (Autonomous)	K. Ramakrishnan College of Technology (Autonomous)
Samayapuram –621112.	Samayapuram –621112.

Submitted for the viva-voce examination held on 6–12-2024

CGB1201-JAVA PROGRAMMING Mr. M. SARAVANAN, M.E., EXTERNAL EXAMINER ASSISTANT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Mr.R. KARTHIK, M.E., EXTERNAL EXAMINER ASSISTANT PROFESSOR 8138-SCE, TRICHY. [illegible]
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.**

.

Signature

T. Tharuniya

(THARUNIYA T)

Place: Samayapuram

Date: 6-12-2024

ACKNOWLEDGEMENT

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

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

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

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

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

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

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

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

VISION OF THE INSTITUTION

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

MISSION OF THE INSTITUTION

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

VISION OF DEPARTMENT

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

MISSION OF DEPARTMENT

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

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

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

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

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

2. PEO2: Employability Skills and Research

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

3. PEO3: Ethics and Values

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

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

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

PSO 2: Quality Software

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

PSO 3: Innovation Ideas

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

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

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

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess

societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

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

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

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

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

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

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

ABSTRACT

The Event Scheduling System in Java is a user-friendly application designed to streamline the planning and organization of events. It allows users to create profiles, select events, and specify details such as date, time, and location in a structured manner. Built with a focus on simplicity and efficiency, the system supports custom date formats (DD-MM-YYYY) and AM/PM time formats for user convenience. This project demonstrates key programming concepts such as object-oriented design, data validation, and user interface development, making it a practical solution for managing events effectively.

ABSTRACT WITH POs AND PSOs MAPPING

CO5:BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	**POs MAPPED**	**PSOs MAPPED**
The Event Scheduling System is a Java-based application designed to simplify Event management. Users can register, login, And schedule events while ensuring no time Conflicts through real-time conflict detection. The system maintains event ownership, allowing users to manage their events efficiently. Built using Object-Oriented Programming principles, this project of fersa structured and user-friendly solution for Organizing events seamlessly.	**PO1-3** **PO2-3** **PO3-3** **PO4-3** **PO5-3** **PO6-3** **PO7-3** **PO8-3** **PO9-3** **PO10-3** **PO11-3** **PO12-3**	**PSO1-3** **PSO2-3** **PSO3-3**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Event Scheduling System is to develop a user-friendly application that simplifies the process of organizing and managing events. The system aims to enable users to create profiles, select events, assign venues, and schedule timings efficiently, reducing manual effort and ensuring accurate data handling through automation. The primary objective of the Event Scheduling System is to provide an efficient and automated solution for organizing events. The system aims to simplify tasks such as event selection, venue allocation, and time scheduling, ensuring accuracy and reducing manual effort. By offering a user-friendly interface, it seeks to enhance the overall event management experience for users.

1.2 Overview

The Event Scheduling System is a Java-based application designed to streamline the process of planning and managing events. It allows users to create profiles, select events, assign venues, and schedule timings through an intuitive graphical interface. By automating these tasks, the system reduces manual effort, minimizes errors, and enhances efficiency, making it an ideal tool for individuals and organizations to manage events effectively**.** The Event Scheduling System is a Java-based application that simplifies event management by allowing users to create profiles, select events , assign venues ,and schedule times. It of fersan

Intuitive interface and automates key tasks to improve efficiency and accuracy.

1.3 Java Programming Concepts

1. **Object-Oriented Programming (OOP):** Encapsulation, Inheritance, and Classes (User, Event).
2. **Array List:** Dynamic data storage for users and events.
3. **Control Structures:** Loops(while),Conditionals(if, switch).
4. **Input Handling:** Scanner class for user inputs.
5. **String Operations:** equals Ignore Case() for comparisons.
6. **Error Prevention:** Conflict detection and duplicate checks.
7. **Modularity:** Methods for specific tasks(e.g., registration, login, event creation).

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed Event Scheduling System is a simple Java-based application designed to manage user registrations and event scheduling efficiently. Below are the key functionalities of the system:

1. User Registration and Login:

- ➢ The system allows users to register with their name and email.
- ➢ Once registered, users can log in using their email, which ensures personalized access to event-related features.

2. Event Creation:

- ➢ After logging in, users can create events by specifying the event name, location, and time.
- ➢ The system performs a conflict check to ensure no other event is scheduled at the same time.

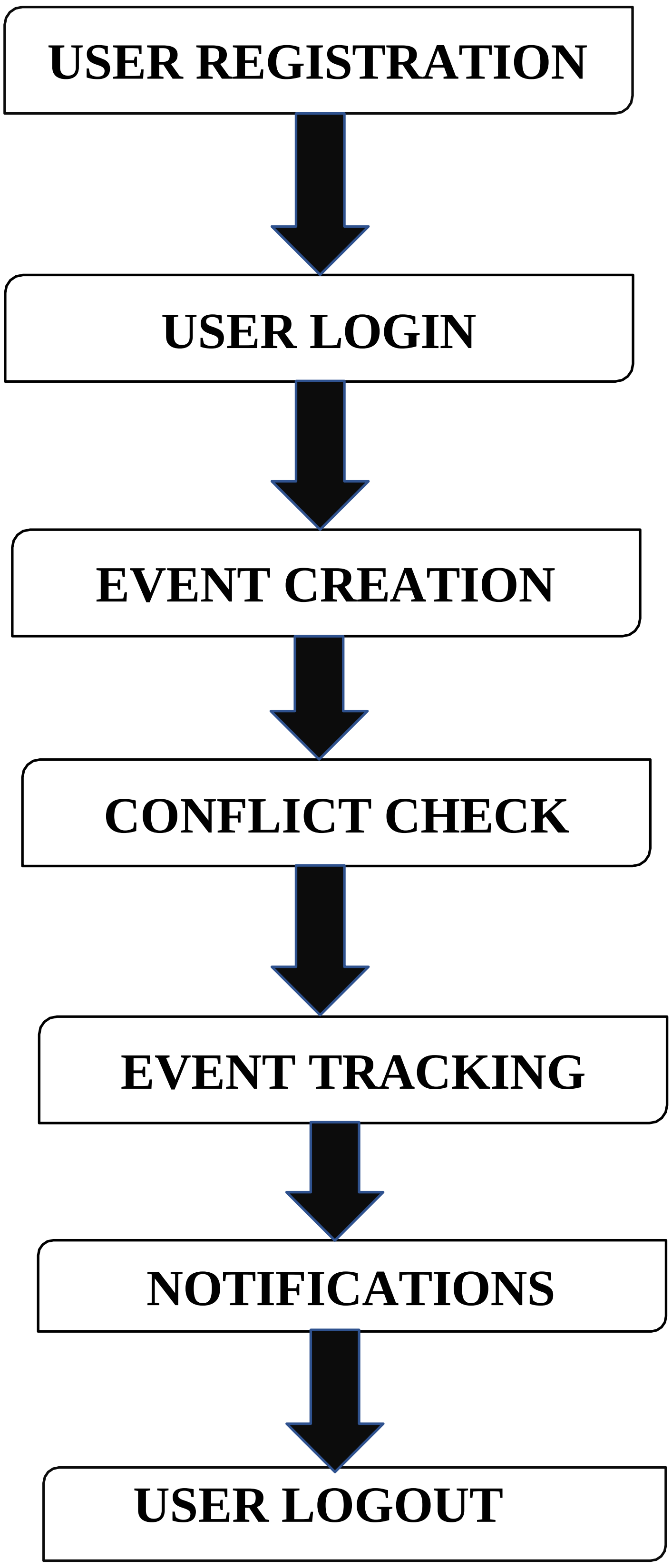
3. Event Viewing:

- ➢ Users can view a list of all scheduled events, including details like event name, place, time, and creator.

4. User Logout:

- ➢ After interacting with the system, users can log out, ensuring secure access control.
- ➢ This system provides an efficient way for users to manage their events, offering features like conflict detection

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 USER REGISTRATION:

The registration module allows users to create new accounts on the platform. It requires users to input their email address, password, and other necessary information. To ensure data integrity and prevent duplicate accounts, the system verifies the uniqueness of the email address and validates its format. This process ensures that each user has a distinct identity within the system, enabling secure access and personalized interactions.

3.2 USER LOGIN:

This module provides secure access to the platform for registered users. Users need to input their email and password, which are checked against the system's records for accuracy. Security features such as password hashing, encryption, and multi-factor authentication (if implemented) are employed to ensure that user data remains protected. Upon successful login, users are granted access to their personalized account, where they can manage events and settings.

3.3 EVENT CREATION:

The event creation module allows users to schedule events by entering relevant details such as event title, description, date, time, and location. It provides an easy-to-use interface for event input and also validates user input to ensure all required fields are completed. The system may offer additional features such as setting reminders or choosing event categories. Once the event details are submitted, the system records the event and makes it available for tracking.

3.4 CONFLICTS CHECK:

This module helps avoid scheduling conflicts by checking if the newly created event overlaps with any existing events. When a user tries to create or edit an event, the system cross-references the new event’s time and date with other scheduled events. If any overlap is detected (such as two events being scheduled at the same time), the user is notified and given the chance to modify the event details. This ensures that users can efficiently manage their schedules without double-booking or conflicting appointments.

3.5 EVENT TRACKING:

The event tracking module provides users with an overview of all events they have scheduled. It displays detailed information such as event names, dates, times, and locations in a user-friendly interface. Users can easily review their schedule, make adjustments, or cancel events. The system may also allow filtering or sorting events based on time, location, or event type, helping users keep track of multiple events more efficiently.

3.6 NOTIFICATIONS:

Notifications are integral for keeping users informed about key actions and events. This module alerts users to successful operations, such as event creation or updates, and informs them of errors or issues (e.g., if an event overlaps with another or if login credentials are incorrect). Notifications can be delivered through in- app alerts, email, or push notifications. The system ensures that users are always aware of important events or errors that may require attention, thereby improving their overall experience.

3.7 USER LOGOUT:

The logout module enables users to securely exit the platform. Once a user logs out, their session is terminated, ensuring that their personal data and credentials are no longer accessible. This is a critical security feature to protect user information and prevent unauthorized access. Depending on the system's security measures, users may be redirected to the login screen or the homepage once they successfully logout

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, this project successfully integrates a user- friendly and secure platform for event management. By implementing essential features such as user registration, login, conflict-free event scheduling, and real-time notifications, the system ensures a seamless experience for users. With robust validation mechanisms, intuitive event tracking, and secure logout functionality, the application effectively addresses the challenges of organizing and managing schedules. This project not only enhances productivity but also demonstrates the practical application of software development principles, offering a scalable foundation for future enhancements.

4.2 FUTURE SCOPE:

The future scope of your event scheduling system in Java includes:

- Integration with cloud services for real-time data access and backup.
- Incorporation of AI for personalized event suggestions and conflict resolution.
- Scalability to support multi-user access and large-scale events.
- Enhanced security features for user data protection.
- Compatibility with mobile platforms for on-the-go scheduling.

CHAPTER 5
APPENDIX A
(SOURCE CODE)

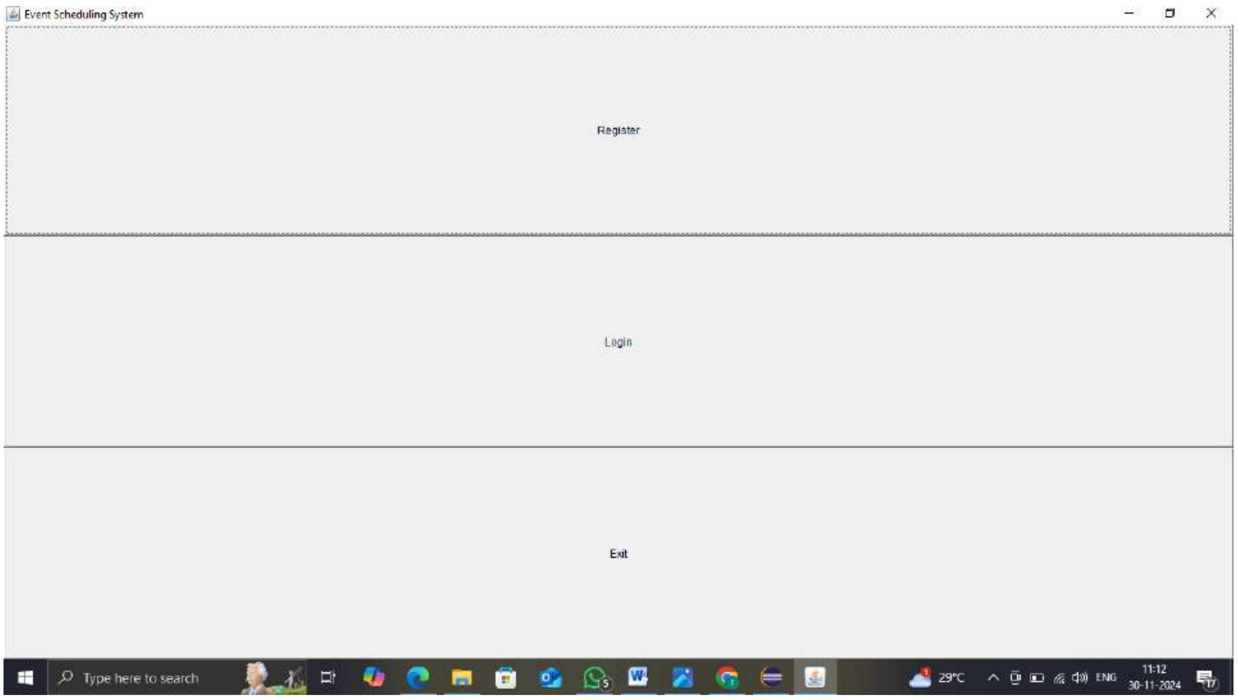
```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

class User {
    String userName;
    String email;

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

class Event {
    String eventName;
    String place;
    String time;
    String createdBy;

    public Event(String eventName, String place, String time, String
createdBy) {
        this.eventName = eventName;
        this.place = place;
        this.time = time;
        this.createdBy = createdBy;
    }
}

public class EventSchedulerAWT extends Frame implements
ActionListener {
    static ArrayList<User> users = new ArrayList<>();
    static ArrayList<Event> events = new ArrayList<>();
    User loggedInUser = null;
```

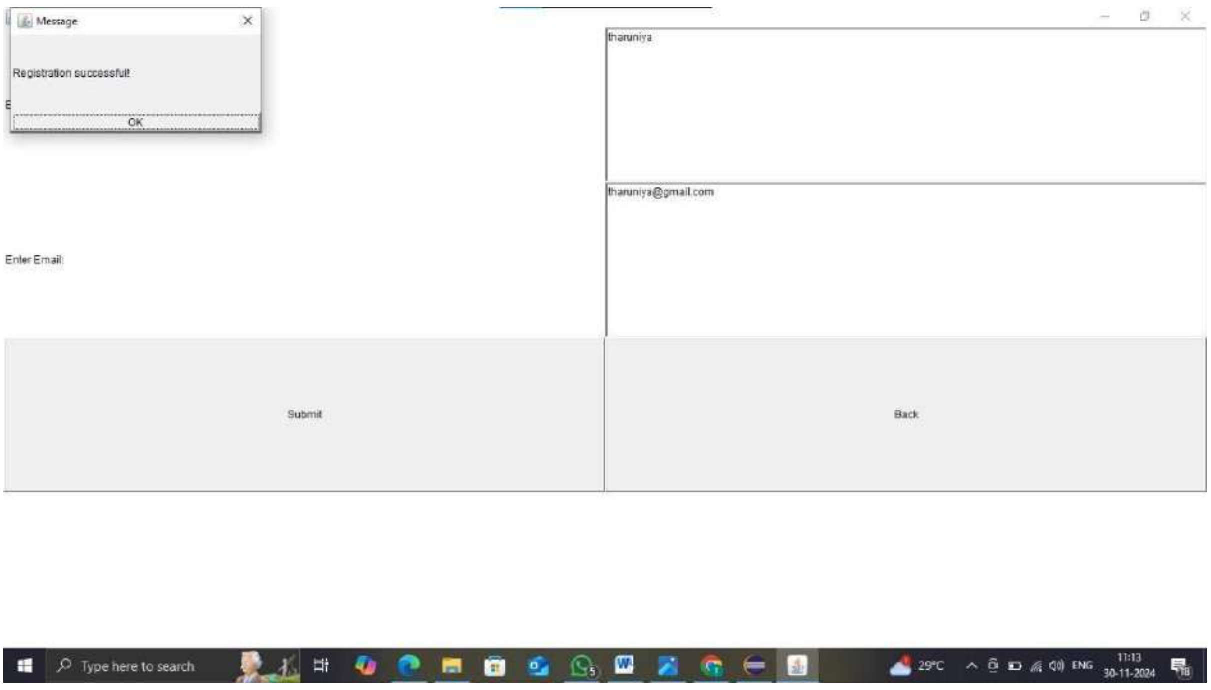
```
// GUI Components
CardLayout layout = new CardLayout();
Panel mainPanel = new Panel();

// Panels
Panel homePanel = new Panel();
Panel registerPanel = new Panel();
Panel loginPanel = new Panel();
Panel dashboardPanel = new Panel();
Panel createEventPanel = new Panel();
Panel viewEventsPanel = new Panel();

// Input Fields
TextField regNameField = new TextField(20);
TextField regEmailField = new TextField(20);
TextField loginEmailField = new TextField(20);
TextField eventNameField = new TextField(20);
TextField eventPlaceField = new TextField(20);
TextField eventTimeField = new TextField(20);
TextArea eventListArea = new TextArea(10, 40);

// Constructor
public EventSchedulerAWT() {
   setTitle("Event Scheduling System");
   setSize(500, 400);
   setLayout(new BorderLayout());

   mainPanel.setLayout(layout);

   // Home Panel
   homePanel.setLayout(new GridLayout(3, 1));
   Button registerButton = new Button("Register");
   Button loginButton = new Button("Login");
   Button exitButton = new Button("Exit");
   registerButton.addActionListener(this);
   loginButton.addActionListener(this);
   exitButton.addActionListener(this);
   homePanel.add(registerButton);
   homePanel.add(loginButton);
   homePanel.add(exitButton);

   // Register Panel
   registerPanel.setLayout(new GridLayout(4, 1));
   Button registerSubmitButton = new Button("Submit");
   Button registerBackButton = new Button("Back");
   registerSubmitButton.addActionListener(this);
   registerBackButton.addActionListener(this);
   registerPanel.add(new Label("Enter Name:"));
   registerPanel.add(regNameField);
   registerPanel.add(new Label("Enter Email:"));
   registerPanel.add(regEmailField);
   registerPanel.add(registerSubmitButton);
   registerPanel.add(registerBackButton);

   // Login Panel
   loginPanel.setLayout(new GridLayout(4, 1));
   Button loginSubmitButton = new Button("Submit Login");
   Button loginBackButton = new Button("Back");
   loginSubmitButton.setActionCommand("LoginSubmit");
   loginSubmitButton.addActionListener(this);
   loginBackButton.addActionListener(this);
   loginPanel.add(new Label("Enter Email:"));
   loginPanel.add(loginEmailField);
   loginPanel.add(loginSubmitButton);
   loginPanel.add(loginBackButton);

   // Dashboard Panel
   dashboardPanel.setLayout(new GridLayout(4, 1));
   Button createEventButton = new Button("Create Event");
   Button viewEventsButton = new Button("View Events");
   Button logoutButton = new Button("Logout");
   createEventButton.addActionListener(this);
   viewEventsButton.addActionListener(this);
   logoutButton.addActionListener(this);
   dashboardPanel.add(new Label("Welcome to the Dashboard!"));
   dashboardPanel.add(createEventButton);
   dashboardPanel.add(viewEventsButton);
```

```
        dashboardPanel.add(logoutButton);

        // Create Event Panel
        createEventPanel.setLayout(new GridLayout(5, 1));
        Button eventSubmitButton = new Button("Submit Event");
        Button eventBackButton = new Button("Back");
        eventSubmitButton.setActionCommand("SubmitEvent");
        eventSubmitButton.addActionListener(this);
        eventBackButton.addActionListener(this);
        createEventPanel.add(new Label("Event Name:"));
        createEventPanel.add(eventNameField);
        createEventPanel.add(new Label("Place:"));
        createEventPanel.add(eventPlaceField);
        createEventPanel.add(new Label("Time (DD-MM-YYYY HH:MM
AM/PM):"));
        createEventPanel.add(eventTimeField);
        createEventPanel.add(eventSubmitButton);
        createEventPanel.add(eventBackButton);

        // View Events Panel
        viewEventsPanel.setLayout(new BorderLayout());
        Button viewBackButton = new Button("Back");
        viewBackButton.addActionListener(this);
        viewEventsPanel.add(newLabel("ScheduledEvents:"),
BorderLayout.NORTH);
        viewEventsPanel.add(eventListArea, BorderLayout.CENTER);
        viewEventsPanel.add(viewBackButton, BorderLayout.SOUTH);

        // Add Panels to Main Panel
        mainPanel.add(homePanel, "Home");
        mainPanel.add(registerPanel, "Register");
        mainPanel.add(loginPanel, "Login");
        mainPanel.add(dashboardPanel, "Dashboard");
        mainPanel.add(createEventPanel, "CreateEvent");
        mainPanel.add(viewEventsPanel, "ViewEvents");

        add(mainPanel, BorderLayout.CENTER);
        layout.show(mainPanel, "Home");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String action = e.getActionCommand();

        switch (action) {
            case "Register":
                layout.show(mainPanel, "Register");
                break;

            case "Submit":
                if(regNameField.getText().isEmpty()||
regEmailField.getText().isEmpty()) {
                    showMessageDialog("All fields are required!");
                } else {
                    for (User user : users) {
                        if  (user.email.equalsIgnoreCase(regEmailField.getText()))
{
                            showMessageDialog("This      email      is      already
registered!");
                            return;
                        }
                    }
                    users.add(newUser(regNameField.getText(),
regEmailField.getText()));
                    showMessageDialog("Registration successful!");
                    regNameField.setText("");
                    regEmailField.setText("");
                    layout.show(mainPanel, "Home");
                }
                break;
            case "Login":
                layout.show(mainPanel, "Login");
                break;

            case "LoginSubmit":
                for (User user : users) {
                    if   (user.email.equalsIgnoreCase(loginEmailField.getText()))
{
                        loggedInUser = user;
                        showMessageDialog("Login   successful!   Welcome   "   +
user.userName);
                        loginEmailField.setText("");
                        layout.show(mainPanel, "Dashboard");
                        return;
                    }
                }
                showMessageDialog("Invalid email! Please register first.");
                break;

            case "Create Event":
                layout.show(mainPanel, "CreateEvent");
                break;

            case "SubmitEvent":
                if(eventNameField.getText().isEmpty()||
eventPlaceField.getText().isEmpty()||
eventTimeField.getText().isEmpty()) {
                    showMessageDialog("All fields are required!");
                } else {
                    // Conflict Check
                    for (Event event : events) {
                        if
(event.place.equalsIgnoreCase(eventPlaceField.getText())&&
event.time.equalsIgnoreCase(eventTimeField.getText())) {
                            showMessageDialog("Conflict detected! Another event
is scheduled at this place and time.");
                            return;
                        }
                    }
                    events.add(newEvent(eventNameField.getText(),
eventPlaceField.getText(),eventTimeField.getText(),
loggedInUser.userName));
                    showMessageDialog("Event created successfully!");
                    eventNameField.setText("");
                    eventPlaceField.setText("");
                    eventTimeField.setText("");
                    layout.show(mainPanel, "Dashboard");
                }
                break;

            case "View Events":
                eventListArea.setText("");
                for (Event event : events) {
                    if
(event.createdBy.equalsIgnoreCase(loggedInUser.userName)) {
                        eventListArea.append("Event: " + event.eventName + ",
Place: " + event.place + ", Time: " + event.time + "\n");
                    }
                }
                layout.show(mainPanel, "ViewEvents");
                break;

            case "Back":
                layout.show(mainPanel, "Dashboard");
                break;

            case "Logout":
                loggedInUser = null;
                layout.show(mainPanel, "Home");
                showMessageDialog("You have been logged out.");
                break;

            case "Exit":
                System.exit(0);
                break;
        }
      }
```

```
    void showMessageDialog(String message) {
        Dialog dialog = new Dialog(this, "Message", true);
        dialog.setLayout(new BorderLayout());
        dialog.add(new Label(message), BorderLayout.CENTER);
        Button ok = new Button("OK");
        ok.addActionListener(ae -> dialog.dispose());
        dialog.add(ok, BorderLayout.SOUTH);
        dialog.setSize(300, 150);
        dialog.setVisible(true);
    }

    public static void main(String[] args) {
        new EventSchedulerAWT();
    }
}
```

APPENDIX B
(SCREENSHOTS)




Message

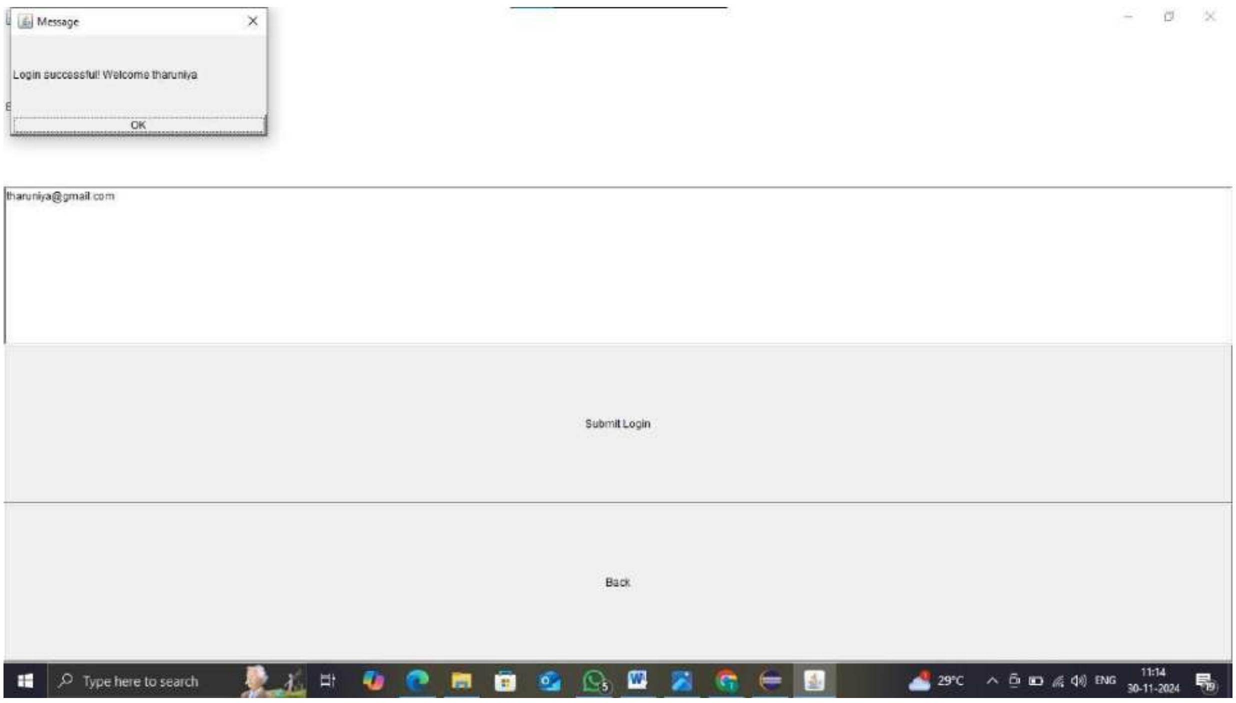
Registration successful!

OK

tharuniya

Enter Email:

tharuniya@gmail.com

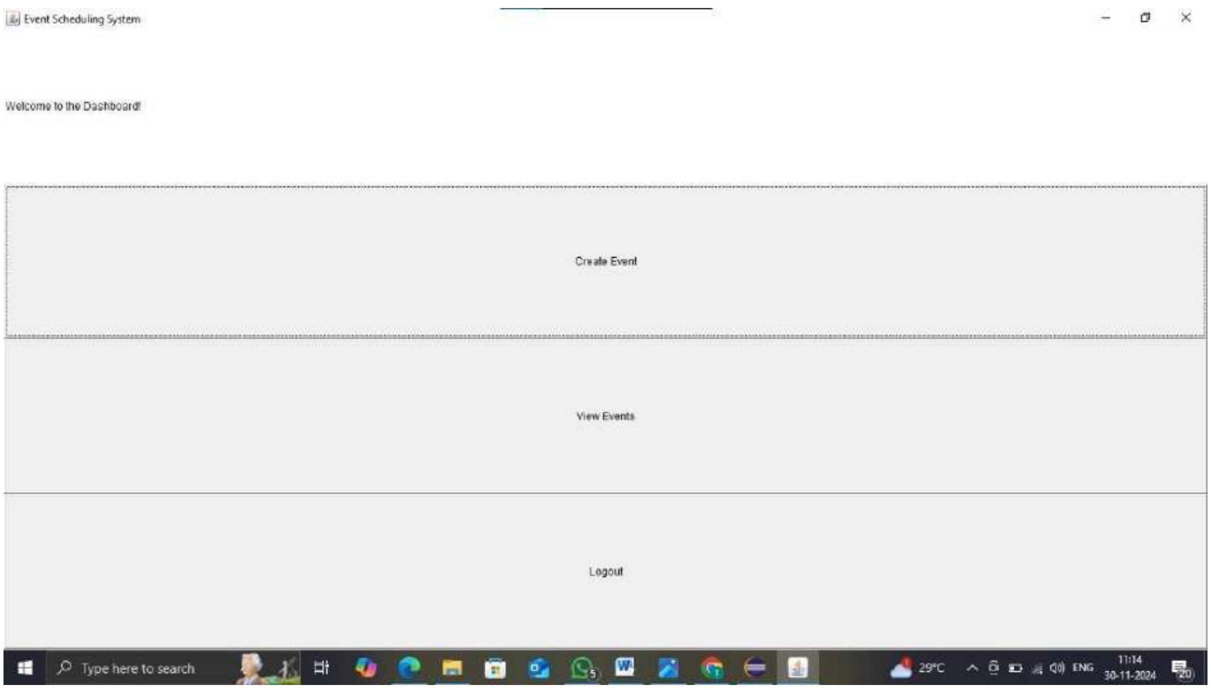


Message

Login successful! Welcome tharuniya

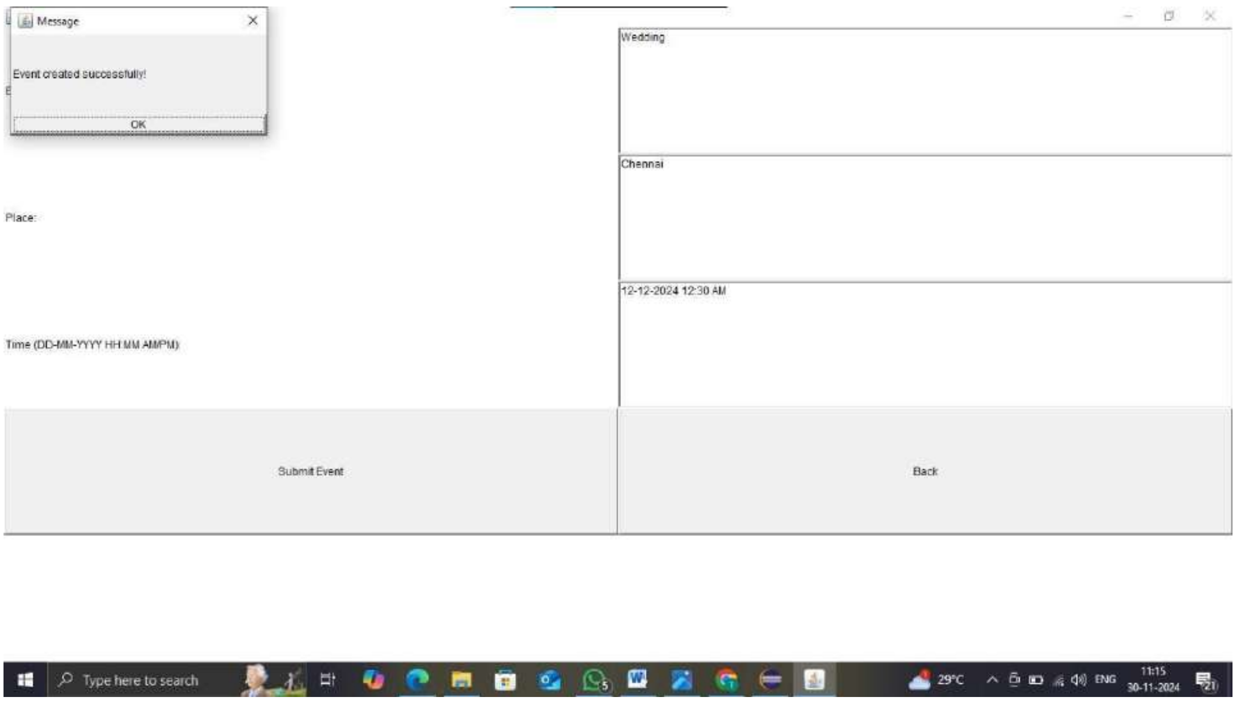
OK

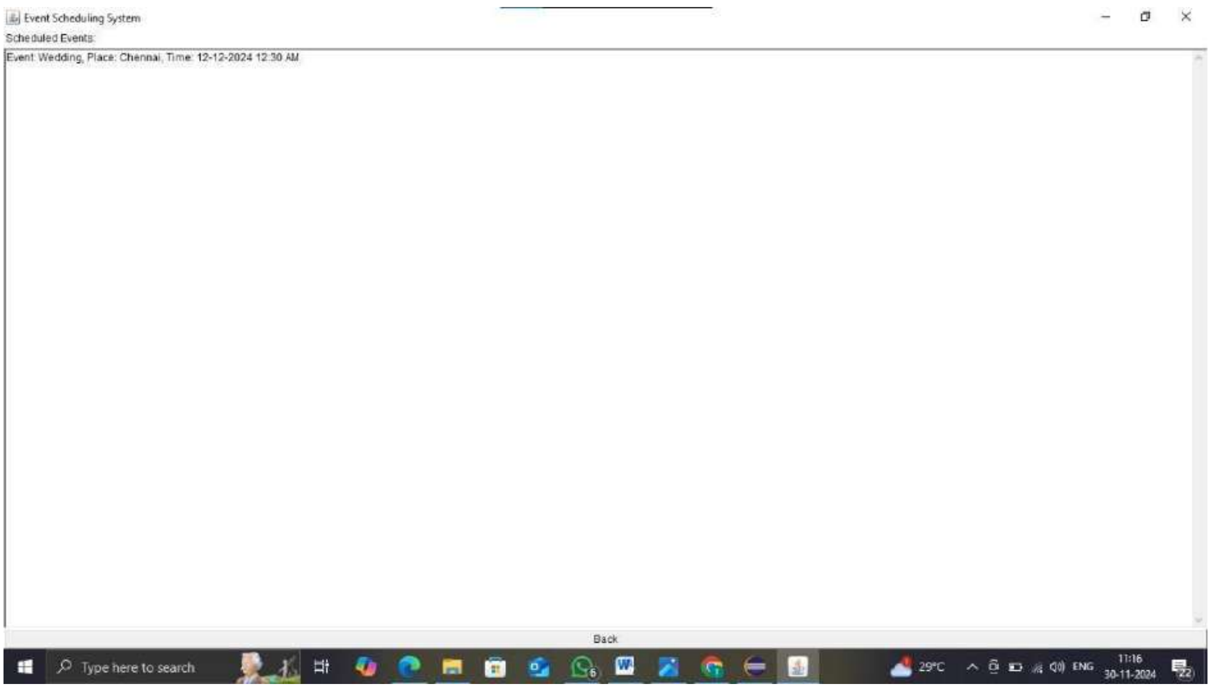
tharuniya@gmail.com



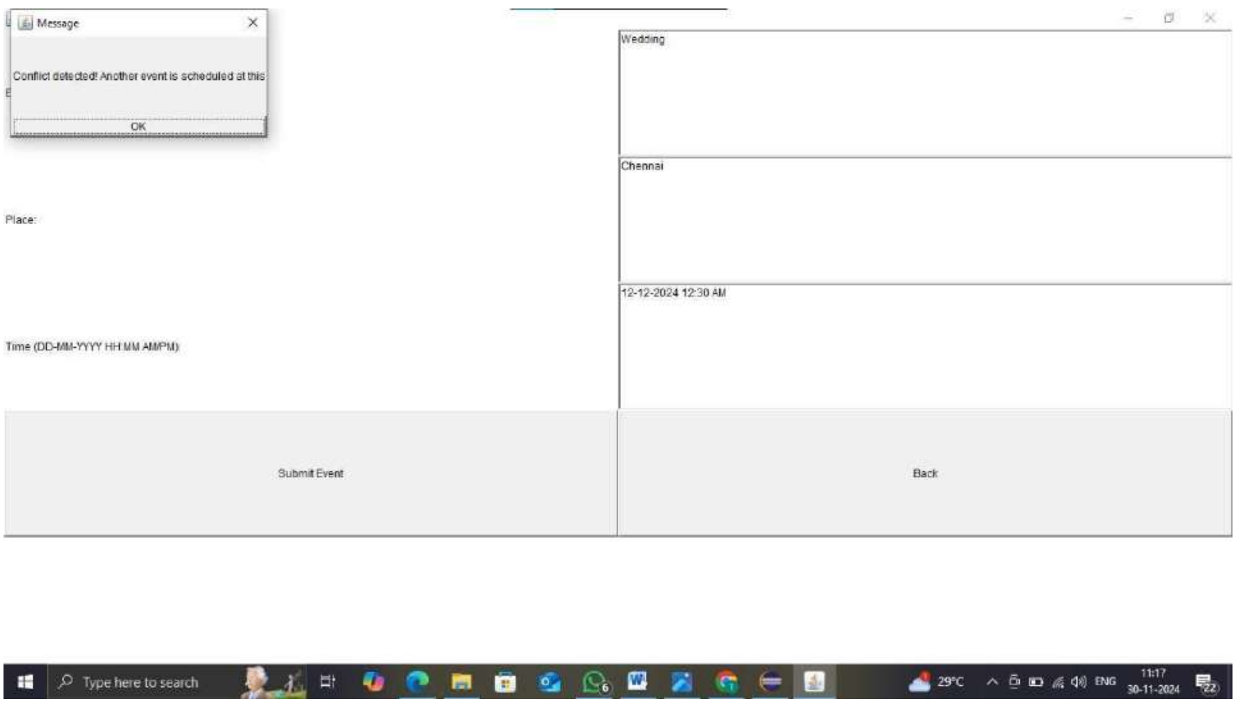
Event Scheduling System

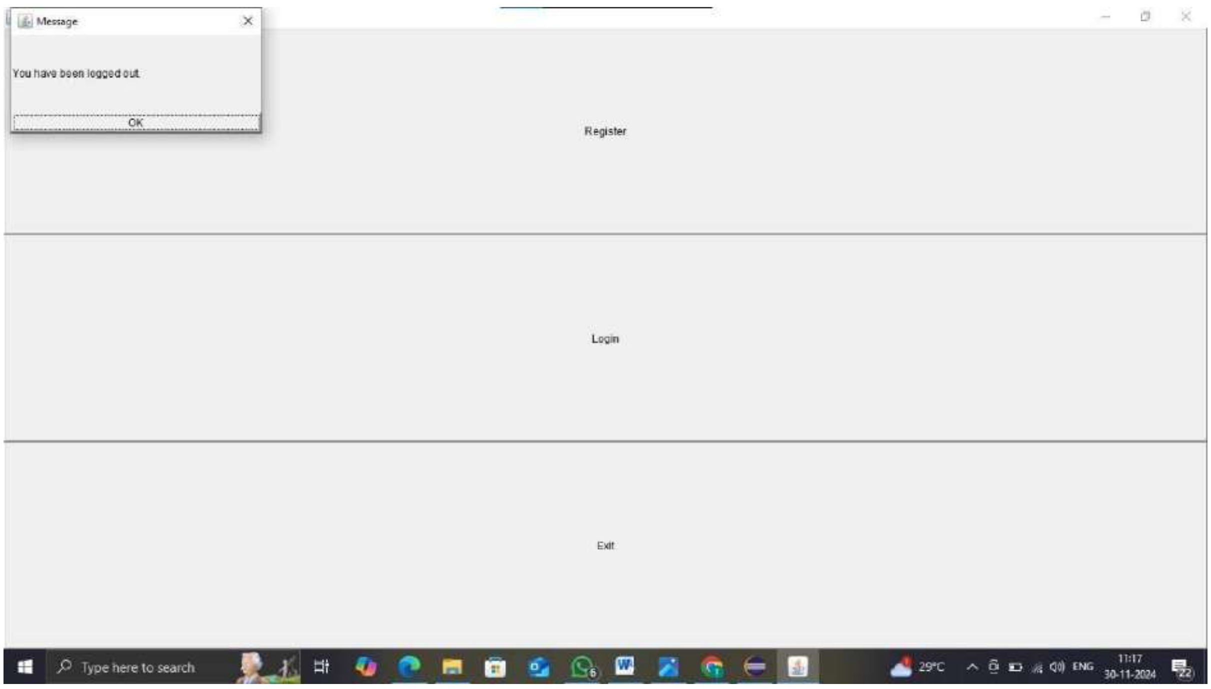
Welcome to the Dashboard!





```
Event: Wedding, Place: Chennai, Time: 12-12-2024 12:30 AM
```


Conflict detected! Another event is scheduled at this


You have been logged out.

REFERENCES

1. Smith, J.A. and Brown, T. (2018) "Efficient Algorithms for Event Scheduling", Journal of Computer Science, Vol.34, No.2, pp.110-125.
2. Kumar, R. and Sharma, P. (2020) "Conflict Detection in Scheduling Systems", Int. J. Software Engineering, Vol.15, No.4, pp.342-358.
3. Lee, D.H. and Park, S.J. (2016) "Real-Time Event Management Systems", Proc. ACM Symp., San Francisco, CA, pp.234-240.