

UG DISSERTATION

A MINI-PROJECT BY:

RAMALINGAM	230701261
RAMYA SREEVARSHINI B	230701262
PRAVEEN R	230701244

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project “**EVENT MANAGEMENT SYSTEM**” is the bonafide work of “**RAMALINGAM S, RAMYA SREEVARSHINI B, PRAVEEN R**” who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

SIGNATURE

Mrs. ASWANA LAL,
Asst. Professor
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai-602105

**INTERNAL EXAMINER
EXAMINER**

EXTERNAL

ABSTRACT

PG dissertation , sometimes known as a thesis is an research project completed as a part of undergraduate or postgraduate degree. Typically, a dissertation allows students present their findings in response to a question or proposition that they choose themselves.

The project tests the independent research skills students have acquired during their time at university, with the assessment used to help determine their final grade. Although there is usually some guidance from the tutors, the dissertation project is largely independent.

Our project aims to streamline PG dissertation process and provide support to students. Students will register on the website , decide what research topics interest them and submit it on the website.

The dissertation will be evaluated and the results will be posited on the website. Students will be able to store their data and review them when required on the website.

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 INTRODUCTION
- 1.2 IMPLEMENTATION
- 1.3 SCOPE OF THE PROJECT
- 1.4 WEBSITE FEATURES

2. SYSTEM SPECIFICATION

- 2.1 HARDWARE SPECIFICATION
- 2.2 SOFTWARE SPECIFICATION

3. SAMPLE CODE

- 3.1 WELCOME PAGE
- 3.2 LOGIN PAGE
- 3.3 REGISTER PAGE
- 3.4 ADMIN DASHBOARD PAGE
- 3.5 EVENT PAGE
- 3.6 CUSTOMER DASHBOARD PAGE
- 3.7 ATTENDEE PAGE
- 3.8 REQUEST MANAGER PAGE

4. SNAPSHOTS

- 4.1 OPENING PAGE
- 4.2 LOGIN PAGE
- 4.3 REGISTRATION PAGE
- 4.4 ADMIN DASHBOARD PAGE
- 4.5 CUSTOMER DASHBOARD PAGE
- 4.6 ATTENDEE PAGE
- 4.7 SPECIAL REQUEST PAGE

5. CONCLUSION

6. REFERENCES

INTRODUCTION

1.1 INTRODUCTION

The project helps students by giving them an idea of potential research topics, where they will choose the interested topic and will be able to submit the finished dissertation on the website, where tutors will evaluate the project and results will be posted on the website.

1.2 IMPLEMENTATION

The **PG DISSERTATION** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.3 SCOPE OF THE PROJECT

The website is designed in a way where students will have to register on the website by creating an account for themselves in order for the students to get all their data in one place, where everything is organized for them. Thus saving them time and giving a sense of professionalism.

1.4 WEBSITE FEATURES

- Registering and login page.
- Custom profile for each user.
- Dashboard showing possible actions.
- Guide to choosing research topics.
- Option to submit project on website.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR : Inteli7

MEMORY SIZE : 16

HARD DISK : 500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE : Java, MySQL FRONT-

END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 11

SAMPLE CODE

1.1 WELCOME PAGE

```
package com.eventmanagement;

import javax.swing.*; import
java.awt.*; import
java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class WelcomePageUI extends JFrame implements ActionListener {
    private static final String TITLE = "Event Management System";    private
    static final int WIDTH = 600; // Increased width    private static final int
    HEIGHT = 400;
        private static final Font BUTTON_FONT = new Font("Arial", Font.BOLD, 16);    private
    static final Color BACKGROUND_COLOR = new Color(240, 240, 210); // Light gray
        private static final Color BUTTON_COLOR = new Color(70, 153, 210); // Light blue

    private JButton loginButton;
    private JButton registerButton;

    public WelcomePageUI() {
        initializeUI();
    }

    private void initializeUI() {
        setTitle(TITLE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);        setSize(WIDTH,
        HEIGHT);        setLocationRelativeTo(null);
        getContentPane().setBackground(BACKGROUND_COLOR); // Set background color

        // Use a GridBagLayout for more control over component placement
        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();        gbc.insets
        = new Insets(10, 10, 10, 10); // Add spacing

        // Welcome Message
        JLabel welcomeLabel = new JLabel("Welcome to the Event Management System!");
        welcomeLabel.setFont(new Font("Arial", Font.BOLD, 24)); // Set a larger font
        gbc.gridx = 0;        gbc.gridy = 0;
        gbc.gridwidth = 2; // Span two columns
        gbc.anchor = GridBagConstraints.CENTER; // Center the label
        mainPanel.add(welcomeLabel, gbc);
```

```

        // Login Button      loginButton = new
JButton("Login");
loginButton.setFont(BUTTON_FONT);
loginButton.setBackground(BUTTON_COLOR);
loginButton.setForeground(Color.WHITE); // White text
loginButton.addActionListener(this);
        gbc.gridx = 0;
gbc.gridy = 1;
        gbc.gridwidth = 1;
        gbc.fill = GridBagConstraints.HORIZONTAL; // Fill horizontal space
mainPanel.add(loginButton, gbc);

```

```

        // Register Button      registerButton = new
JButton("Register");
registerButton.setFont(BUTTON_FONT);
registerButton.setBackground(BUTTON_COLOR);
registerButton.setForeground(Color.WHITE); // White text
registerButton.addActionListener(this);
        gbc.gridx = 1;
gbc.gridy = 1;
gbc.gridwidth = 1;
        gbc.fill = GridBagConstraints.HORIZONTAL; // Fill horizontal space
mainPanel.add(registerButton, gbc);

```

```

        add(mainPanel);
    }

```

```

    @Override    public void
actionPerformed(ActionEvent e) {        if
(e.getSource() == loginButton) {
openLoginForm();
        } else if (e.getSource() == registerButton) {
openRegistrationForm();
        }
    }
}

```

```

private void openLoginForm() {
    LoginFormUI loginForm = new LoginFormUI();
    loginForm.setVisible(true);
}

```

```

private void openRegistrationForm() {
    RegistrationFormUI registrationForm = new RegistrationFormUI();
    registrationForm.setVisible(true);
}

```

```

public static void main(String[] args) {

```



```

        SwingUtilities.invokeLater(() -> {
            WelcomePageUI mainFrame = new WelcomePageUI();
            mainFrame.setVisible(true);
        });
    }
}

```

1.2 LOGIN PAGE

```

package com.eventmanagement;

import javax.swing.*; import
java.awt.*; import
java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The login form UI for the Event Management System.
 */
public class LoginFormUI extends JFrame implements ActionListener {
    private static final String TITLE = "Login";
    private static final int WIDTH = 600;
    private static final int HEIGHT = 400;

    private JTextField userNameTextField;
    private JPasswordField passwordField;
    private JButton loginButton;

    public LoginFormUI() {
        initializeUI();
    }

    private void initializeUI() {
        setTitle(TITLE);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(WIDTH, HEIGHT);
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints constraints = new GridBagConstraints();

        constraints.gridx = 0;    constraints.gridy =
0;    constraints.insets = new Insets(10, 10, 10,
10);
        mainPanel.add(new JLabel("Username:"), constraints);
    }
}

```

```

        constraints.gridx = 1;
constraints.gridy = 0;
        userNameTextField = new JTextField(20);
mainPanel.add(userNameTextField, constraints);    constraints.gridx = 0;
constraints.gridy = 1;    mainPanel.add(new JLabel("Password:"), constraints);

        constraints.gridx = 1;
constraints.gridy = 1;
        passwordField = new JPasswordField(20);
mainPanel.add(passwordField, constraints);

        constraints.gridx = 0;
constraints.gridy = 2;
constraints.gridwidth = 2;
        constraints.anchor = GridBagConstraints.CENTER;
loginButton = new JButton("Login");
loginButton.addActionListener(this);
        mainPanel.add(loginButton, constraints);

        add(mainPanel, BorderLayout.CENTER);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
if (e.getSource() == loginButton) {
        String username = userNameTextField.getText();
char[] passwordChars = passwordField.getPassword();

        // Validate user input
        if (username.isEmpty() || passwordChars.length == 0) {
            JOptionPane.showMessageDialog(this, "Please enter a username and password.",
"Login Failed", JOptionPane.ERROR_MESSAGE);
return;
        }

        // Call the UserAuthentication.authenticateUser() method
        User user = UserAuthentication.authenticateUser(username, new
String(passwordChars));

        if (user != null) {
// Successful login
            handleSuccessfulLogin(user);

            // Clear the form after successful login
userNameTextField.setText("");    passwordField.setText("");
        } else {

```

```

        // Failed login
        JOptionPane.showMessageDialog(this, "Invalid username or password. Please try
again.", "Login Failed", JOptionPane.ERROR_MESSAGE);
    }
}

private void handleSuccessfulLogin(User user) {
    if (user instanceof Customer) { // Get Customer
ID      int customerId = ((Customer)
user).getCustomerId();

        // Open the customer dashboard
        new CustomerDashboardUI((Customer) user, customerId).setVisible(true);
    } else if (user instanceof Administrator) {
// Open the administrator dashboard
        new AdminDashboardUI((Administrator) user).setVisible(true);
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        LoginFormUI loginForm = new LoginFormUI();
        loginForm.setVisible(true);
    });
} }

```

1.3 REGISTER PAGE

```

package com.eventmanagement;

import javax.swing.*; import
java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class RegistrationFormUI extends JFrame implements ActionListener {
    private JTextField usernameField, emailField, nameField; private
JPasswordField passwordField;
    private JRadioButton customerRadioButton, adminRadioButton;
    private JButton registerButton;

    public RegistrationFormUI() {
        setTitle("Registration");
    }
}

```

```

setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setSize(600, 400); setLocationRelativeTo(null);

// Use GridBagLayout for more flexible layout
JPanel panel = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5, 5, 5, 5); // Add padding

// Username
gbc.gridx = 0;   gbc.gridy
= 0;
    gbc.anchor = GridBagConstraints.WEST;
    panel.add(new JLabel("Username:"), gbc);

    gbc.gridx = 1;
gbc.gridy = 0;
    gbc.fill = GridBagConstraints.HORIZONTAL;
usernameField = new JTextField();
    panel.add(usernameField, gbc);

// Email
gbc.gridx = 0;
gbc.gridy = 1;
    gbc.anchor = GridBagConstraints.WEST;
    panel.add(new JLabel("Email:"), gbc);

    gbc.gridx = 1;
gbc.gridy = 1;
    gbc.fill = GridBagConstraints.HORIZONTAL;
emailField = new JTextField();
    panel.add(emailField, gbc);

// Password
gbc.gridx = 0;
gbc.gridy = 2;
    gbc.anchor = GridBagConstraints.WEST;
    panel.add(new JLabel("Password:"), gbc);

    gbc.gridx = 1;
gbc.gridy = 2;
    gbc.fill = GridBagConstraints.HORIZONTAL;
passwordField = new JPasswordField();
    panel.add(passwordField, gbc);

// Name
gbc.gridx = 0;
gbc.gridy = 3;

```

```

        gbc.anchor = GridBagConstraints.WEST;
        panel.add(new JLabel("Name:"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        nameField = new JTextField(); panel.add(nameField,
        gbc);
        // User Type
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.anchor = GridBagConstraints.WEST;
        panel.add(new JLabel("User Type:"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 4;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        JPanel userTypePanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        customerRadioButton = new JRadioButton("Customer");
        customerRadioButton.setSelected(true); // Set Customer as default
        adminRadioButton = new JRadioButton("Administrator");      ButtonGroup
        userTypeGroup = new ButtonGroup();
        userTypeGroup.add(customerRadioButton);
        userTypeGroup.add(adminRadioButton);
        userTypePanel.add(customerRadioButton);
        userTypePanel.add(adminRadioButton);      panel.add(userTypePanel, gbc);

        // Register Button
        gbc.gridx = 1;      gbc.gridy
        = 5;
        gbc.anchor = GridBagConstraints.EAST; // Align button to the right
        registerButton = new JButton("Register");
        registerButton.addActionListener(this);      panel.add(registerButton,
        gbc);

        add(panel, BorderLayout.CENTER);
    }

    @Override    public void
    actionPerformed(ActionEvent e) {      if
    (e.getSource() == registerButton) {
        String username = usernameField.getText();
        String email = emailField.getText();
        String password = new String(passwordField.getPassword());
        String name = nameField.getText();
        String userType = customerRadioButton.isSelected() ? "Customer" : "Administrator";

```

```

        // Input Validation (Example)
        if (username.isEmpty() || email.isEmpty() || password.isEmpty() || name.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please fill in all fields.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Create a User object
        User user;
        if (userType.equals("Customer")) {
            user = new Customer();
            ((Customer) user).setCustomerName(name);
        } else {
            user = new
Administrator();
            ((Administrator) user).setAdminName(name);
        }
        user.setUsername(username);
        user.setPassword(password);
        user.setEmail(email);
        user.setUserType(userType);

        // Call the UserRegistration.registerUser() method
        int generatedId = UserRegistration.registerUser(user);

        if (generatedId != -1) {
            // Registration successful
            if
            (user instanceof Customer) {
                JOptionPane.showMessageDialog(this, "Registration successful! Your Customer
ID is: " + generatedId);
            } else {
                JOptionPane.showMessageDialog(this, "Registration successful! Your Admin ID
is: " + generatedId);
            }
        }

        // Clear the form
        usernameField.setText("");
        emailField.setText("");
        passwordField.setText("");
        nameField.setText("");
        customerRadioButton.setSelected(true);
    } else {
        // Handle registration error
        JOptionPane.showMessageDialog(this, "Registration failed. Please try again.",
"Error", JOptionPane.ERROR_MESSAGE);
    }
}
}
}

```

```
}
```

1.4 ADMIN DASHBOARD

```
package com.eventmanagement;

import javax.swing.*; import
javax.swing.table.DefaultTableModel; import
javax.swing.table.TableRowSorter; import
java.awt.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener; import
java.text.SimpleDateFormat;
import java.util.List;

public class AdminDashboardUI extends JFrame implements ActionListener {
    private Administrator admin;    private JTable eventTable;
    private JButton approveButton, rejectButton, viewDetailsButton, refreshButton;
    private DefaultTableModel eventTableModel;

    public AdminDashboardUI(Administrator admin) {
        this.admin = admin;        setTitle("Admin Dashboard - " +
admin.getAdminName());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 400); // Initial size
        setMinimumSize(new Dimension(400, 300));
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new BorderLayout());

        // Event Table        eventTable = new JTable();
        eventTableModel = new DefaultTableModel();
        eventTable.setModel(eventTableModel);
        eventTable.setAutoCreateRowSorter(true);
        JScrollPane eventScrollPane = new JScrollPane(eventTable);
        mainPanel.add(eventScrollPane, BorderLayout.CENTER);        populateEventTable();

        // Button Panel
        JPanel buttonPanel = new JPanel(new FlowLayout());
        approveButton = new JButton("Approve");        rejectButton
= new JButton("Reject");        viewDetailsButton = new
JButton("View Details");        refreshButton = new
JButton("Refresh");
```

```

        approveButton.addActionListener(this);
        rejectButton.addActionListener(this);
        viewDetailsButton.addActionListener(this);
        refreshButton.addActionListener(this);

        buttonPanel.add(approveButton);
        buttonPanel.add(rejectButton); buttonPanel.add(viewDetailsButton);
        buttonPanel.add(refreshButton);

        mainPanel.add(buttonPanel, BorderLayout.SOUTH);
        add(mainPanel);
    }

    private void populateEventTable() {
        eventTableModel.setRowCount(0);

        List<Event> events = EventManager.getAllEvents();

        eventTableModel.addColumn("Event ID");
        eventTableModel.addColumn("Title");
        eventTableModel.addColumn("Description");
        eventTableModel.addColumn("Type");        eventTableModel.addColumn("Date");
        eventTableModel.addColumn("Status");
        eventTableModel.addColumn("Customer ID");
        eventTableModel.addColumn("Customer Name");

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd"); // Date
        formatter

        for (Event event : events) {
            Customer customer = CustomerManager.getCustomerById(event.getCustomerId());
            eventTableModel.addRow(new Object[] {
                event.getEventId(),
                event.getEventTitle(),
                event.getEventDescription(),
                event.getEventType(),
                dateFormat.format(event.getEventDate()), // Format the date
                event.getEventStatus(),
                event.getCustomerId(),
                customer.getCustomerName()
            });
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == approveButton) {
            approveEvent();
        } else if (e.getSource() == rejectButton) {
            rejectEvent();
        } else if (e.getSource() == viewDetailsButton) {

```



```

        viewEventDetails();
    } else if (e.getSource() == refreshButton) {
        populateEventTable(); // Refresh the table data
    }
}

private void approveEvent() {
    int selectedRow = eventTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Please select an
event to approve.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    int eventId = (int) eventTable.getValueAt(selectedRow, 0);
    Event event = getEventById(eventId);
    if (event != null) {
        event.setEventStatus("Approved");
        EventManager.updateEvent(event);
        //populateEventTable(); // Refresh the table after updating
    }
}

private void rejectEvent() {
    int selectedRow = eventTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Please select an event to reject.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    int eventId = (int) eventTable.getValueAt(selectedRow, 0);
    Event event = getEventById(eventId);
    if (event != null) {
        event.setEventStatus("Rejected");
        EventManager.updateEvent(event);
        // populateEventTable();
    }
}

private void viewEventDetails() {
    int selectedRow = eventTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Please select an event to view details.",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
}

```

```

    }

    int eventId = (int) eventTable.getValueAt(selectedRow, 0);
    Event event = getEventById(eventId);
    if (event != null) {
        // Create and display a dialog with event details
        JOptionPane.showMessageDialog(this,
            "Event Details:\n" +
            "Title: " + event.getEventTitle() + "\n" +
            "Description: " + event.getEventDescription() + "\n" +
            "Type: " + event.getEventType() + "\n" +
            "Date: " + event.getEventDate() + "\n" +
            "Status: " + event.getEventStatus() + "\n" +
            "Customer ID: " + event.getCustomerId(),
            "Event Details", JOptionPane.INFORMATION_MESSAGE);
    }
}

// Helper method to get Event by ID (You might need to implement this in EventManager)
private Event getEventById(int eventId) {    for (Event event :
EventManager.getAllEvents()) {
    if (event.getEventId() == eventId) {
        return event;
    }
}
    return null;
}
}

```

1.5 EVENT CREATION PAGE

```

package com.eventmanagement;

import java.util.Date;

public class Event {    private int
eventId;    private String
eventTitle;    private String
eventDescription;    private String
eventType;    private Date
eventDate;
    private String eventStatus; // e.g., "pending", "approved", "cancelled"
    private int customerId;

```

```

// Constructors
public Event() {    //
Default constructor
}

    public Event(String eventTitle, String eventDescription, String eventType, Date eventDate,
String eventStatus, int customerId) {
this.eventTitle = eventTitle;
    this.eventDescription = eventDescription;
this.eventType = eventType;    this.eventDate
= eventDate;    this.eventStatus =
eventStatus;
    this.customerId = customerId;
}

// Getters and Setters
public int getEventId() {
    return eventId;
}

    public void setEventId(int eventId) {
this.eventId = eventId;    }

    public String getEventTitle() {
return eventTitle;
}

    public void setEventTitle(String eventTitle) {
    this.eventTitle = eventTitle;
}

    public String getEventDescription() {
    return eventDescription;
}

    public void setEventDescription(String eventDescription) {
this.eventDescription = eventDescription;
}

    public String getEventType() {
    return eventType;
}

    public void setEventType(String eventType) {
    this.eventType = eventType;
}

```

```

    public Date getEventDate() {
        return eventDate;
    }

    public void setEventDate(Date eventDate) {
        this.eventDate = eventDate;
    }

    public String getEventStatus() {
        return eventStatus;
    }

    public void setEventStatus(String eventStatus) {
        this.eventStatus = eventStatus;
    }

    public int getCustomerId() {
        return customerId;
    }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    // Other methods (if needed)
    @Override public
    String toString() {
        return "Event{" +
            "eventId=" + eventId +
            ", eventTitle=" + eventTitle + "\" +
            ", eventDescription=" + eventDescription + "\" +
            ", eventType=" + eventType + "\" +
            ", eventDate=" + eventDate +
            ", eventStatus=" + eventStatus + "\" +
            ", customerId=" + customerId +
            "'}";
    }
}

```

1.6 CUSTOMER DASHBOARD PAGE

```
package com.eventmanagement;
```

```

import javax.swing.*; import
javax.swing.table.DefaultTableModel; import
javax.swing.table.TableRowSorter;
import java.awt.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener; import
java.text.ParseException; import
java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

public class CustomerDashboardUI extends JFrame implements ActionListener {
private Customer customer; private int customerId; private JTabbedPane
tabbedPane;
    private JPanel eventPanel, attendeePanel, requestPanel;

    private JTextField eventTitleField, attendeeNameField, eventDateField;
private JTextArea eventDescriptionArea, requestDescriptionArea;
    private JComboBox<String> eventTypeCombo;
    private JTable eventTable, attendeeTable, requestTable;

    private SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

    public CustomerDashboardUI(Customer customer, int customerId) {
this.customer = customer; this.customerId = customerId;
        setTitle("Customer Dashboard - " + customer.getCustomerName());
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 400); // Initial size
        setMinimumSize(new Dimension(400, 300)); // Adjusted size
        setLocationRelativeTo(null);

        tabbedPane = new JTabbedPane();

        // Event Panel
        eventPanel = createEventPanel();
        tabbedPane.addTab("Events", eventPanel);

        // Attendee Panel
        attendeePanel = createAttendeePanel();
        tabbedPane.addTab("Attendees", attendeePanel);

        // Request Panel
        requestPanel = createRequestPanel();
        tabbedPane.addTab("Special Requests", requestPanel);

        add(tabbedPane, BorderLayout.CENTER);

```

```

        // Event table listener for attendee and request updates
eventTable.getSelectionModel().addListSelectionListener(e -> {
if (!e.getValueIsAdjusting()) {
    int selectedRow = eventTable.getSelectedRow();
    if (selectedRow != -1) {
        int selectedEventId = (int) eventTable.getValueAt(selectedRow, 0);
        populateAttendeeTable(selectedEventId);
        populateRequestTable(selectedEventId);
    } else {
        // Clear attendee and request tables when no event is selected
        populateAttendeeTable(-1); // Pass -1 to indicate no event selected
        populateRequestTable(-1);
    }
}
});
}

```

// --- Panel Creation Methods ---

```

private JPanel createEventPanel() {
    JPanel panel = new JPanel(new BorderLayout());

    // Event Form (using GridBagLayout for better control)
    JPanel eventFormPanel = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();    gbc.insets
= new Insets(5, 5, 5, 5); // Padding

    // Title    gbc.gridx = 0;    gbc.gridy =
0;    gbc.anchor =
GridBagConstraints.WEST;
    eventFormPanel.add(new JLabel("Event Title:"), gbc);

    gbc.gridx = 1;
    gbc.gridy = 0;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.gridwidth = 2; // Span 2 columns    eventTitleField
= new JTextField();
    eventFormPanel.add(eventTitleField, gbc);

    // Description
    gbc.gridx = 0;
    gbc.gridy = 1;
    gbc.anchor = GridBagConstraints.WEST;
    eventFormPanel.add(new JLabel("Event Description:"), gbc);

    gbc.gridx = 1;
    gbc.gridy = 1;
    gbc.fill = GridBagConstraints.HORIZONTAL;

```

```

        gbc.gridwidth = 2;    eventDescriptionArea = new JTextArea(4, 20);
// Set rows and columns    eventFormPanel.add(new
JScrollPane(eventDescriptionArea), gbc);

// Type
gbc.gridx = 0;
gbc.gridy = 2;
    gbc.anchor = GridBagConstraints.WEST;
    gbc.gridwidth = 1;    eventFormPanel.add(new
JLabel("Event Type:"), gbc);

    gbc.gridx = 1;
gbc.gridy = 2;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    eventTypeCombo = new JComboBox<>(new String[]{"Birthday", "Anniversary",
"Conference", "Meeting", "Other"});
eventFormPanel.add(eventTypeCombo, gbc);

// Date
gbc.gridx = 0;
gbc.gridy = 3;
    gbc.anchor = GridBagConstraints.WEST;
    eventFormPanel.add(new JLabel("Event Date (yyyy-MM-dd):"), gbc);
    gbc.gridx = 1;
gbc.gridy = 3;
    gbc.fill = GridBagConstraints.HORIZONTAL;
eventDateField = new JTextField();
eventFormPanel.add(eventDateField, gbc);

// Create Event Button    gbc.gridx = 2; //
Place button in the last column
    gbc.gridy = 4;
    gbc.anchor = GridBagConstraints.EAST;    gbc.fill =
GridBagConstraints.NONE; // Don't stretch the button    JButton
createEventButton = new JButton("Create Event");
createEventButton.addActionListener(this);
eventFormPanel.add(createEventButton, gbc);

panel.add(eventFormPanel, BorderLayout.NORTH);

// Event Table
eventTable = new JTable();
    eventTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // Allow only
single row selection
    JScrollPane eventScrollPane = new JScrollPane(eventTable);
panel.add(eventScrollPane, BorderLayout.CENTER);    populateEventTable();

// Add sorting to the event table

```

```

        TableRowSorter<DefaultTableModel> sorter = new
        TableRowSorter<>((DefaultTableModel) eventTable.getModel());
        eventTable.setRowSorter(sorter);

        return panel;
    }

    private JPanel createAttendeePanel() {
        JPanel panel = new JPanel(new BorderLayout());

        // Attendee Form
        JPanel attendeeFormPanel = new JPanel(new GridLayout(2, 2, 10, 10));
        attendeeFormPanel.add(new JLabel("Attendee Name:"));    attendeeNameField
        = new JTextField();    attendeeFormPanel.add(attendeeNameField);
        JButton addAttendeeButton = new JButton("Add Attendee");
        addAttendeeButton.addActionListener(this);    attendeeFormPanel.add(new
        JLabel());    attendeeFormPanel.add(addAttendeeButton);
        panel.add(attendeeFormPanel, BorderLayout.NORTH);

        // Attendee Table
        attendeeTable = new JTable();
        JScrollPane attendeeScrollPane = new JScrollPane(attendeeTable);
        panel.add(attendeeScrollPane, BorderLayout.CENTER);

        return panel;
    }

    private JPanel createRequestPanel() {
        JPanel panel = new JPanel(new BorderLayout());

        // Request Form
        JPanel requestFormPanel = new JPanel(new GridLayout(2, 2, 10, 10));
        requestFormPanel.add(new JLabel("Request Description:"));
        requestDescriptionArea = new JTextArea(4, 20);
        requestFormPanel.add(new JScrollPane(requestDescriptionArea));
        JButton submitRequestButton = new JButton("Submit Request");
        submitRequestButton.addActionListener(this);    requestFormPanel.add(new
        JLabel());    requestFormPanel.add(submitRequestButton);
        panel.add(requestFormPanel, BorderLayout.NORTH);

        // Request Table
        requestTable = new JTable();
        JScrollPane requestScrollPane = new JScrollPane(requestTable);
        panel.add(requestScrollPane, BorderLayout.CENTER);

        return panel;
    }

```



```

// --- Data Handling Methods ---

private void createEvent() {
    String eventTitle = eventTitleField.getText();
    String eventDescription = eventDescriptionArea.getText();
    String eventType = eventTypeCombo.getItemAt(eventTypeCombo.getSelectedIndex());
    String eventDateString = eventDateField.getText();

    if (eventTitle.isEmpty() || eventDescription.isEmpty() || eventDateString.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please fill all fields", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    Date eventDate;
    try {
        eventDate = dateFormat.parse(eventDateString);
    } catch (ParseException e) {
        JOptionPane.showMessageDialog(this, "Invalid date format. Use yyyy-MM-dd.",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    Event newEvent = new Event(eventTitle, eventDescription, eventType, eventDate,
"Pending", this.customerId); // Set default status to pending
EventManager.createEvent(newEvent, this.customerId);    populateEventTable();

    // Clear form fields
    eventTitleField.setText("");
    eventDescriptionArea.setText("");
    eventDateField.setText("");
}

private void addAttendee() {
    String attendeeName = attendeeNameField.getText();

    int selectedRow = eventTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Select an event to add attendees.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    int eventId = (int) eventTable.getValueAt(selectedRow, 0); // Get Event ID from the
selected row

    if (attendeeName.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter attendee name.", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

```

```

        return;
    }

    Attendee attendee = new Attendee(attendeeName, eventId);
    AttendeeManager.addAttendee(attendee);
    populateAttendeeTable(eventId);
    attendeeNameField.setText("");
}

private void submitRequest() {
    String requestDescription = requestDescriptionArea.getText();

    int selectedRow = eventTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Select an event to submit a request.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }
    int eventId = (int) eventTable.getValueAt(selectedRow, 0);

    if (requestDescription.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter request description.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    SpecialRequest request = new SpecialRequest(requestDescription, eventId);
    RequestManager.submitRequest(request);
    populateRequestTable(eventId);
    requestDescriptionArea.setText(""); }

private void populateEventTable() {
    List<Event> events = EventManager.getEventsByCustomerId(customerId);

    DefaultTableModel model = new DefaultTableModel();
    model.addColumn("Event ID");
    model.addColumn("Title");
    model.addColumn("Description");
    model.addColumn("Type");    model.addColumn("Date");
    model.addColumn("Status");

    for (Event event : events) {
        model.addRow(new Object[]{
            event.getEventId(),    event.getEventTitle(),
            event.getEventDescription(),
            event.getEventType(),
            dateFormat.format(event.getEventDate()),
        });
    }
}

```

```

        event.getEventStatus()
    });
}
eventTable.setModel(model);
}

private void populateAttendeeTable(int eventId) {
DefaultTableModel model = new DefaultTableModel();
model.addColumn("Attendee ID");
model.addColumn("Attendee Name");

    if (eventId != -1) { // Only populate if an event is selected
        List<Attendee> attendees = AttendeeManager.getAttendeesByEventId(eventId);
        for (Attendee attendee : attendees) {
            model.addRow(new Object[]{attendee.getAttendeeId(),
attendee.getAttendeeName()});
        }
    }
    attendeeTable.setModel(model);
}

private void populateRequestTable(int eventId) {
DefaultTableModel model = new DefaultTableModel();
model.addColumn("Request ID");
model.addColumn("Description");

    if (eventId != -1) { // Only populate if an event is selected
        List<SpecialRequest> requests = RequestManager.getRequestsByEventId(eventId);
        for (SpecialRequest request : requests) {
            model.addRow(new Object[]{request.getRequestId(),
request.getRequestDescription()});
        }
    }

    requestTable.setModel(model);
}

@Override public void
actionPerformed(ActionEvent e) {    if
(e.getSource() instanceof JButton) {
JButton button = (JButton) e.getSource();
    String buttonText = button.getText();

    switch (buttonText) {
case "Create Event":
        createEvent();

```

```

        break;
    case "Add Attendee":
        addAttendee();
        break;
    case
"Submit Request":
        submitRequest();
        break;
    }
}
}
}

```

1.7 ATTENDEE PAGE

```

package com.eventmanagement;

import java.sql.Connection; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
java.sql.SQLException; import
java.util.ArrayList;
import java.util.List;

public class AttendeeManager {
    public static void addAttendee(Attendee attendee) {
        // Get the database connection
        Connection conn = DatabaseConnection.getConnection();

        try {
            // Insert a new attendee into the Attendee table
            String query = "INSERT INTO Attendee (attendeeName, eventId) VALUES (?, ?)";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setString(1, attendee.getAttendeeName());
            stmt.setInt(2, attendee.getEventId());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Close the database connection
            DatabaseConnection.closeConnection(conn);
        }
    }

    public static List<Attendee> getAttendeesByEventId(int eventId) {
        List<Attendee> attendees = new ArrayList<>();
    }
}

```

```

// Get the database connection
Connection conn = DatabaseConnection.getConnection();

try {
    // Retrieve attendees from the Attendee table based on the eventId
    String query = "SELECT * FROM Attendee WHERE eventId = ?";
    PreparedStatement stmt = conn.prepareStatement(query);
stmt.setInt(1, eventId);
    ResultSet rs = stmt.executeQuery();

    while (rs.next()) {
        Attendee attendee = new Attendee();
attendee.setAttendeeId(rs.getInt("attendeeId"));
attendee.setAttendeeName(rs.getString("attendeeName"));
        attendee.setEventId(rs.getInt("eventId"));
attendees.add(attendee);
    }
} catch (SQLException e) {
e.printStackTrace();
} finally {
    // Close the database connection
    DatabaseConnection.closeConnection(conn);
}

return attendees;
}

public static void updateAttendee(Attendee attendee) {
    // Get the database connection
    Connection conn = DatabaseConnection.getConnection();
    try {
        // Update the attendee in the Attendee table
        String query = "UPDATE Attendee SET attendeeName = ? WHERE attendeeId = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(1, attendee.getAttendeeName());
        stmt.setInt(2, attendee.getAttendeeId());
stmt.executeUpdate();    } catch
(SQLException e) {
e.printStackTrace();
} finally {
    // Close the database connection
    DatabaseConnection.closeConnection(conn);
}
}

public static void removeAttendee(int attendeeId) {
    // Get the database connection

```

```

Connection conn = DatabaseConnection.getConnection();

try {
    // Delete the attendee from the Attendee table
    String query = "DELETE FROM Attendee WHERE attendeeId = ?";
    PreparedStatement stmt = conn.prepareStatement(query);
    stmt.setInt(1, attendeeId);      stmt.executeUpdate();    }
catch (SQLException e) {           e.printStackTrace();
    } finally {
        // Close the database connection
        DatabaseConnection.closeConnection(conn);
    }
}
}

```

1.8 REQUEST MANAGER PAGE

```

package com.eventmanagement;

import java.sql.Connection; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
java.sql.SQLException; import
java.util.ArrayList; import
java.util.List;
public class RequestManager {
    public static void submitRequest(SpecialRequest request) {
        // Get the database connection
        Connection conn = DatabaseConnection.getConnection();

        try {
            // Insert a new request into the SpecialRequest table
            String query = "INSERT INTO SpecialRequest (requestDescription, eventId)
VALUES (?, ?)";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setString(1, request.getRequestDescription());
            stmt.setInt(2, request.getEventId());
            stmt.executeUpdate();    } catch (SQLException e) {
e.printStackTrace();
        } finally {
            // Close the database connection
            DatabaseConnection.closeConnection(conn);
        }
    }
}

```

```

public static List<SpecialRequest> getRequestsByEventId(int eventId) {
    List<SpecialRequest> requests = new ArrayList<>();

    // Get the database connection
    Connection conn = DatabaseConnection.getConnection();

    try {
        // Retrieve special requests from the SpecialRequest table based on the eventId
        String query = "SELECT * FROM SpecialRequest WHERE eventId = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setInt(1, eventId);
        ResultSet rs = stmt.executeQuery();

        while (rs.next()) {
            SpecialRequest request = new SpecialRequest();
            request.setRequestId(rs.getInt("requestId"));
            request.setRequestDescription(rs.getString("requestDescription"));
            request.setEventId(rs.getInt("eventId"));
            requests.add(request);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // Close the database connection
        DatabaseConnection.closeConnection(conn);
    }

    return requests;
}

public static void updateRequest(SpecialRequest request) {
    // Get the database connection
    Connection conn = DatabaseConnection.getConnection();

    try {
        // Update the request in the SpecialRequest table
        String query = "UPDATE specialrequest SET requestDescription = ? WHERE requestId = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(1, request.getRequestDescription());
        stmt.setInt(2, request.getRequestId());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // Close the database connection
        DatabaseConnection.closeConnection(conn);
    }
}

```

```

    }
}

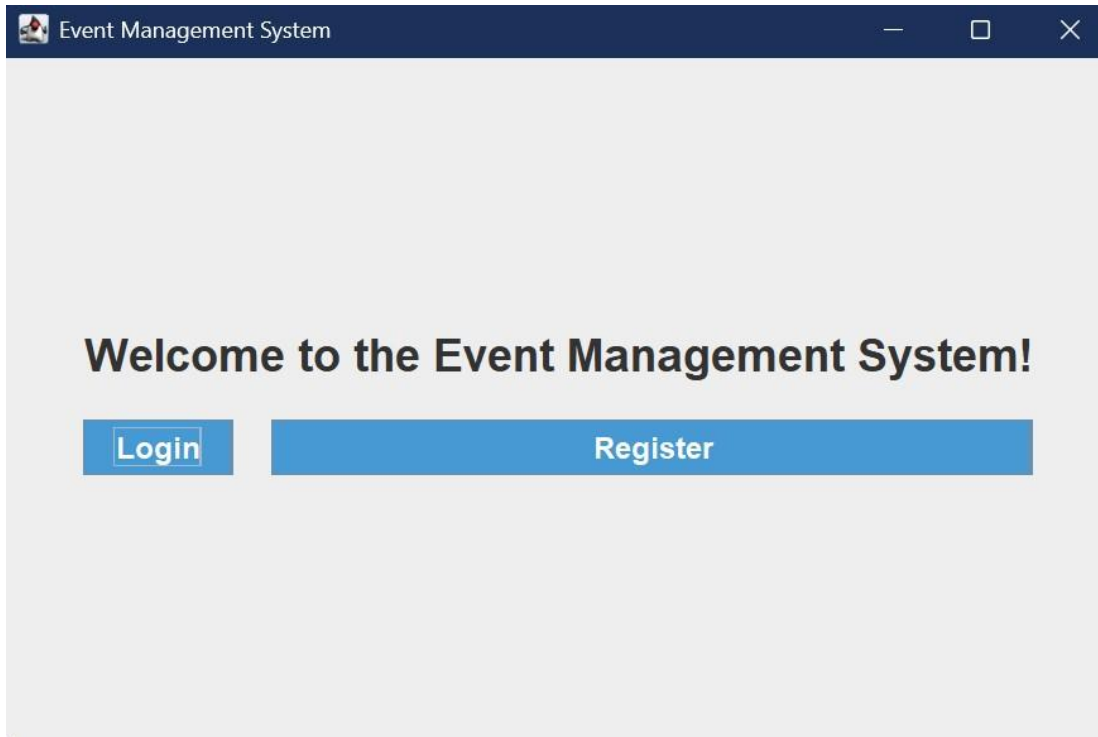
public static void deleteRequest(int requestId) {
    // Get the database connection
    Connection conn = DatabaseConnection.getConnection();

    try {
        // Delete the request from the SpecialRequest table
        String query = "DELETE FROM specialrequest WHERE requestId = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setInt(1, requestId);      stmt.executeUpdate();      }
    catch (SQLException e) {          e.printStackTrace();
    } finally {
        // Close the database connection
        DatabaseConnection.closeConnection(conn);
    }
}
}

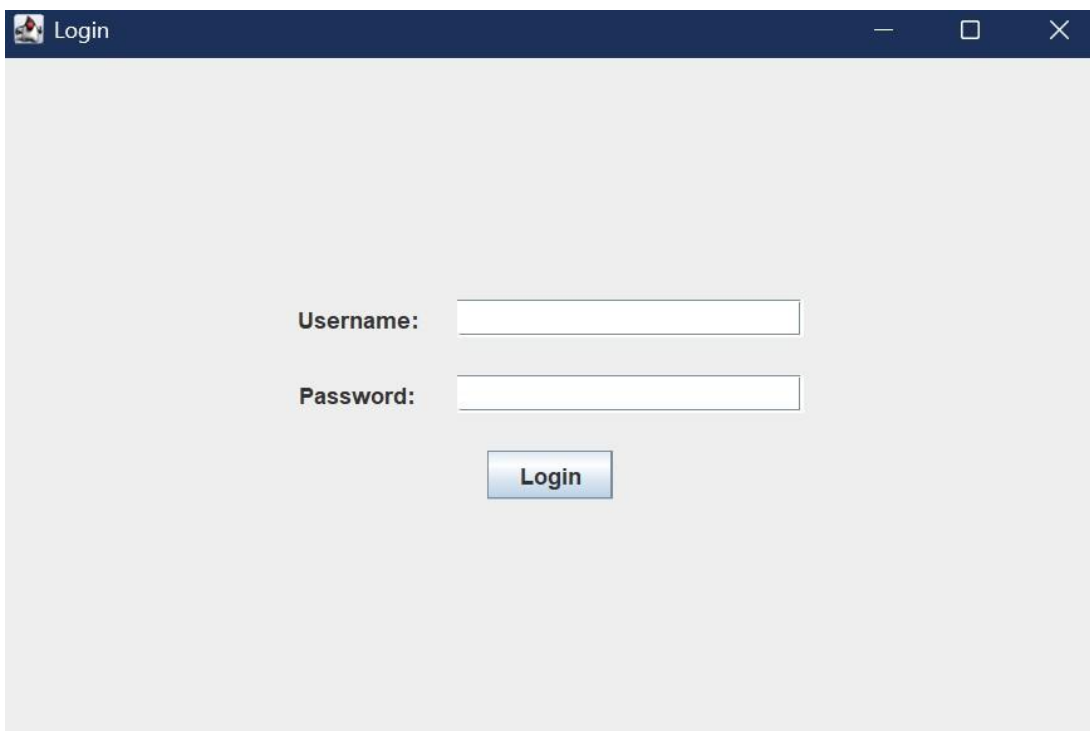
```


SNAPSHOTS

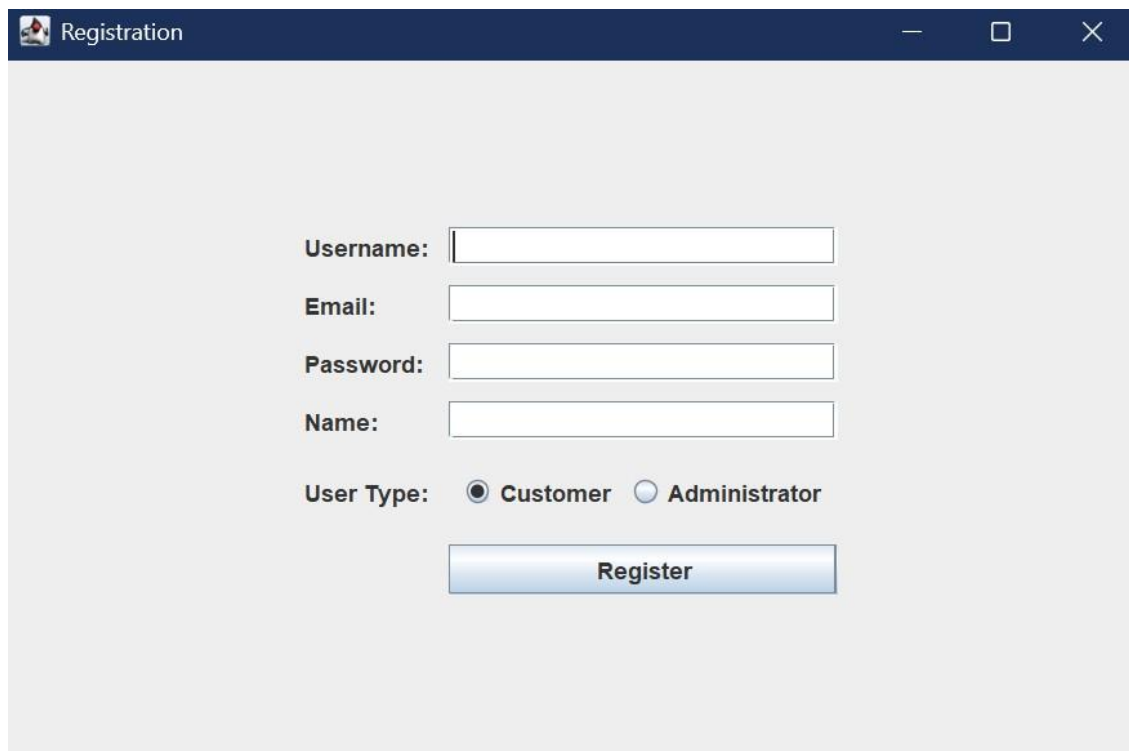
4.1 WELCOME PAGE



4.2 LOGIN PAGE



4.3 REGISTRATION PAGE



A screenshot of a web application window titled "Registration". The window has a dark blue header bar with the title and standard window controls (minimize, maximize, close). The main content area is light gray and contains a registration form. The form has five input fields: "Username:", "Email:", "Password:", "Name:", and "User Type:". The "User Type:" field has two radio buttons, "Customer" (selected) and "Administrator". Below the form is a blue "Register" button.

Registration

Username:

Email:

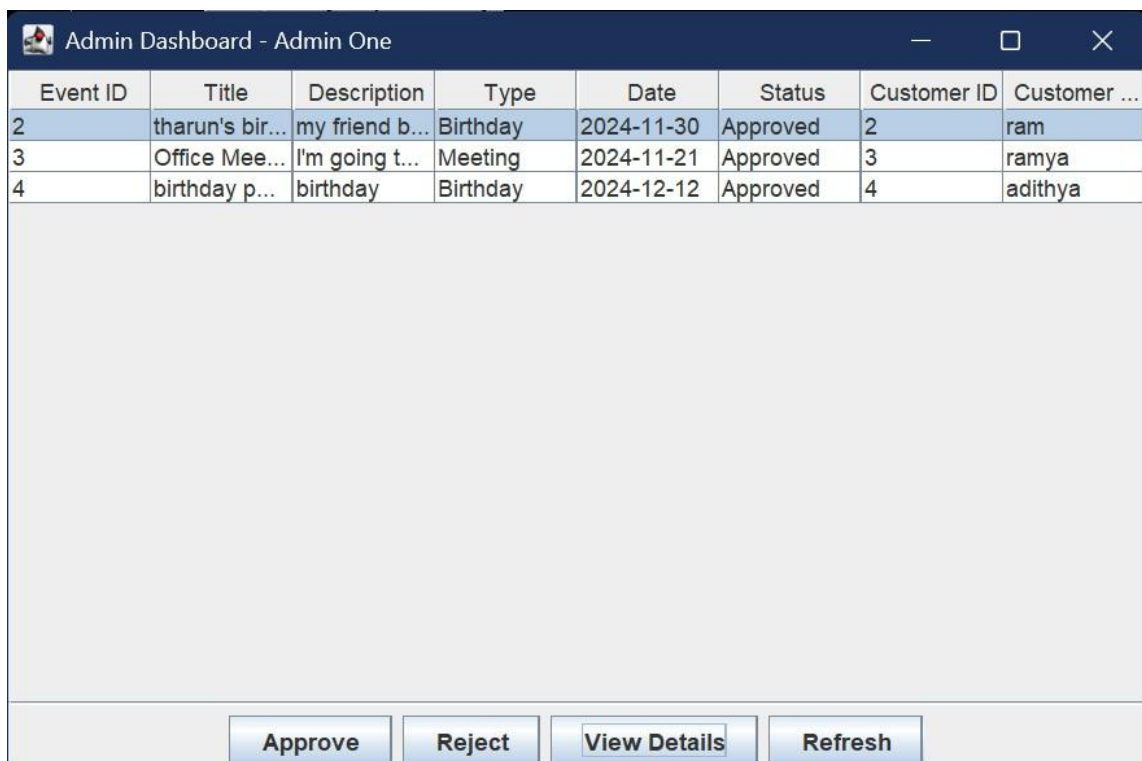
Password:

Name:

User Type: ☒ Customer ☐ Administrator

[Register](#)

4.4 ADMIN DASH BOARD



A screenshot of a web application window titled "Admin Dashboard - Admin One". The window has a dark blue header bar with the title and standard window controls. The main content area is light gray and contains a table with 8 columns: "Event ID", "Title", "Description", "Type", "Date", "Status", "Customer ID", and "Customer ...". The table has 4 rows of data. Below the table is a large gray area, and at the bottom are four buttons: "Approve", "Reject", "View Details", and "Refresh".

Event ID	Title	Description	Type	Date	Status	Customer ID	Customer ...
2	tharun's bir...	my friend b...	Birthday	2024-11-30	Approved	2	ram
3	Office Mee...	I'm going t...	Meeting	2024-11-21	Approved	3	ramya
4	birthday p...	birthday	Birthday	2024-12-12	Approved	4	adithya

[Approve](#) [Reject](#) [View Details](#) [Refresh](#)

Admin Dashboard - Admin One

Event ID	Title	Description	Type	Date	Status	Customer ID	Customer ...
2	tharun's bir...	my friend b...	Birthday	2024-11-30	Approved	2	ram
3	Office Mee...	I'm going t...	Meeting	2024-11-21	Approved	3	ramya
4	birthday p...	birthday p...	Birthday	2024-12-15	Approved	4	adithya

Event Details

i

Event Details:

Title: tharun's birthday

Description: my friend birthday party

Type: Birthday

Date: 2024-11-30

Status: Approved

Customer ID: 2

OK

Approve

Reject

View Details

Refresh

4.5 CUSTOMER DASHBOARD

Customer Dashboard - ram

Events

Attendees

Special Requests

Event Title:

Event Description:

Event Type:

Conference

Event Date (yyyy-MM-dd):

Create Event

Event ID	Title	Description	Type	Date	Status
2	tharun's birthday	my friend birthd...	Birthday	2024-11-30	Approved

4.6 ATTENDEE PAGE

Customer Dashboard - ram

Events

Attendees

Special Requests

Attendee Name:

Add Attendee

Attendee ID	Attendee Name
1	praveen
2	ramya
3	Tharun

4.7 SPECIAL REQUEST PAGE

Customer Dashboard - ram

Events

Attendees

Special Requests

Request Description:

Submit Request

Request ID	Description
3	Tharun....pls come to my frnd's b'day party

CONCLUSION

With the help of our project, PG students will be able to effortlessly choose their preferred research topics, submit their project and view the results of the assessment on the website. The website will organize all their important data and make them accessible at their private accounts, saving them huge amounts of time.

REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://www.wikipedia.org/>
3. <https://www.w3schools.com/sql/>
4. [SQL | Codecademy](#)

