EVENT REGISTRATION SYSTEM

A MINI PROJECT REPORT

Submitted by

ABIRAMI K	**230701008**
ABHINAYA LAKSHMI S	**230701004**

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
DECEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report **" EVENT REGISTRATION SYSTEM "** is the **bonafide work of**

ABIRAMI K (230701008) AND ABHINAYA LAKSHMI S(230701004)

who carried out the project work under my supervision.

Submitted for the Practical Examination held on ____________________

SIGNATURE **SIGNATURE**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Event Registration System is a robust database application designed to streamline the management of events and participant registrations. It incorporates modules for user registration, secure authentication, event management, and data handling. Users can easily register by providing personal details and selecting events, while administrators manage event information through features that allow for adding, editing, or removing event records. The system also supports updates to user data and the removal of records when necessary, ensuring efficient database maintenance.

Built on a relational database foundation, the system prioritizes data integrity, security, and scalability. SQL-based operations ensure optimized performance and adherence to normalization standards, minimizing redundancy. This versatile system is well-suited for a wide range of use cases, including academic, corporate, and professional event management, offering a seamless and secure solution for automating the event registration process.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The Event Registration System is a database-centric application designed to facilitate the registration process for events. Built using Java for the front-end interface and MySQL for the backend, this system enables users to register for events, update their information, and manage event data in a structured, database-driven environment. The project focuses on effective data management through DBMS principles, making it suitable for event organizers requiring efficient and accurate data handling

1.2 OBJECTIVE

1. To develop a system that centralizes event registration data in a structured database.
2. To streamline event registration, modification, and deletion for users.
3. To implement a secure, scalable, and user-friendly interface for both users and administrators.
4. To ensure data integrity and security through robust DBMS practices

1.3 MODULES

1. **User Registration Module**: Allows users to register by entering personal information and selecting an event.
2. **Authentication Module**: Verifies user credentials for secure access.
3. **Event Management Module**: Enables admins to add, edit, and delete event details.
4. **Data Modification Module**: Allows users to update their data in the system.
5. **Data Deletion Module**: Provides users or admins the ability to delete user records as needed.

CHAPTER 2
SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The Event Registration System is built using Java for the front-end GUI, JDBC for database connectivity, and MySQL as the relational database management system. MySQL's structured data storage allows for efficient querying and data manipulation, which are essential for managing user registrations and event information.

2.2 LANGUAGES

2.2.1 SQL

Structured Query Language (SQL) is used for all database operations within the system, including CRUD (Create, Read, Update, Delete) functions. SQL queries are embedded in Java using JDBC, allowing for seamless interaction with the MySQL database.

2.2.2 Java

Java is used for the application's front-end and logic, enabling the creation of a user-friendly interface with Swing components. Java's JDBC API facilitates secure and efficient database connections to MySQL.

CHAPTER 3
REQUIREMENTS AND ANALYSIS

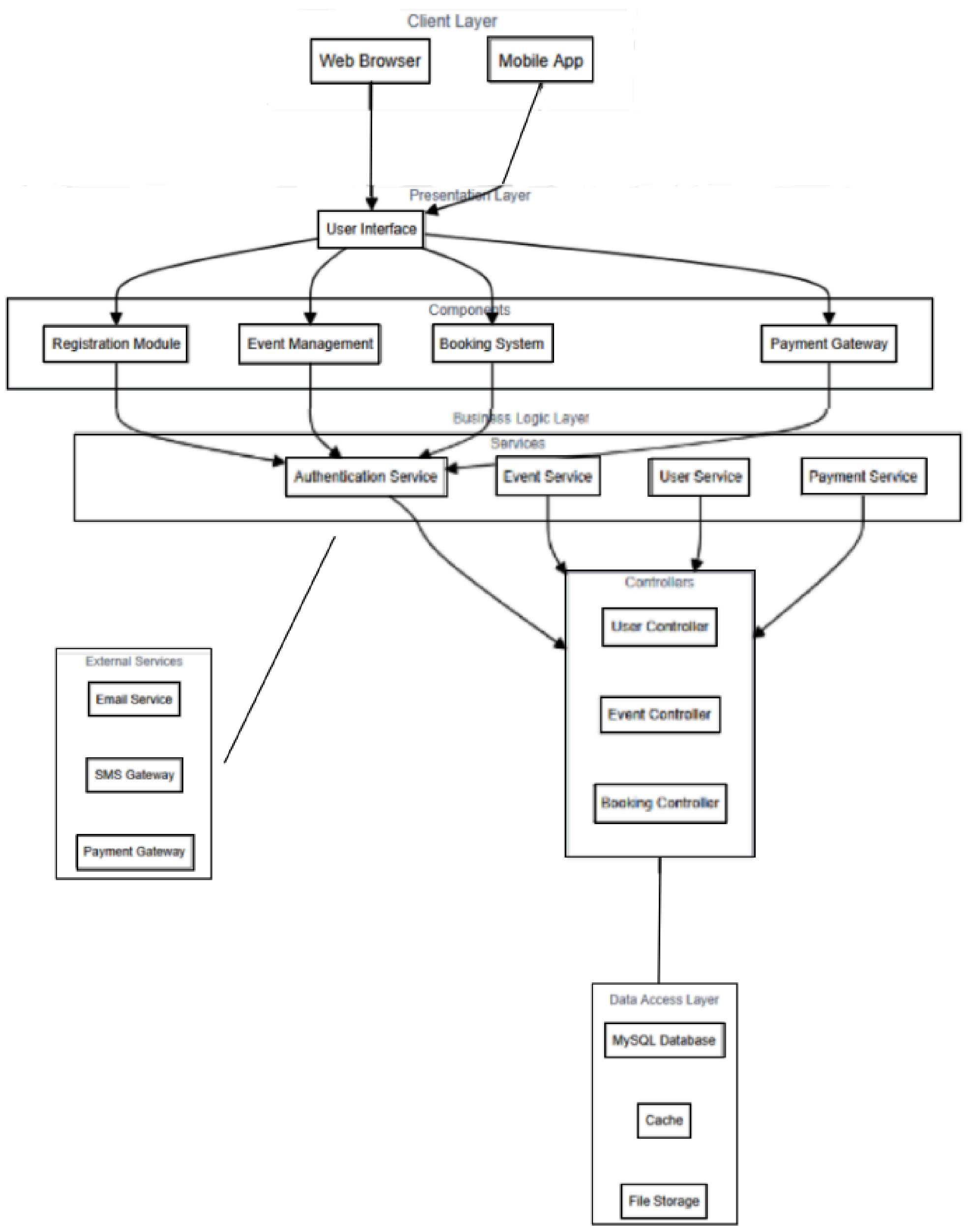
3.1 REQUIREMENT SPECIFICATIONS:

The system should support user registration, data modification, event management, and data deletion functionalities. Additionally, the application must ensure data integrity, security, and efficient retrieval of event-related information.

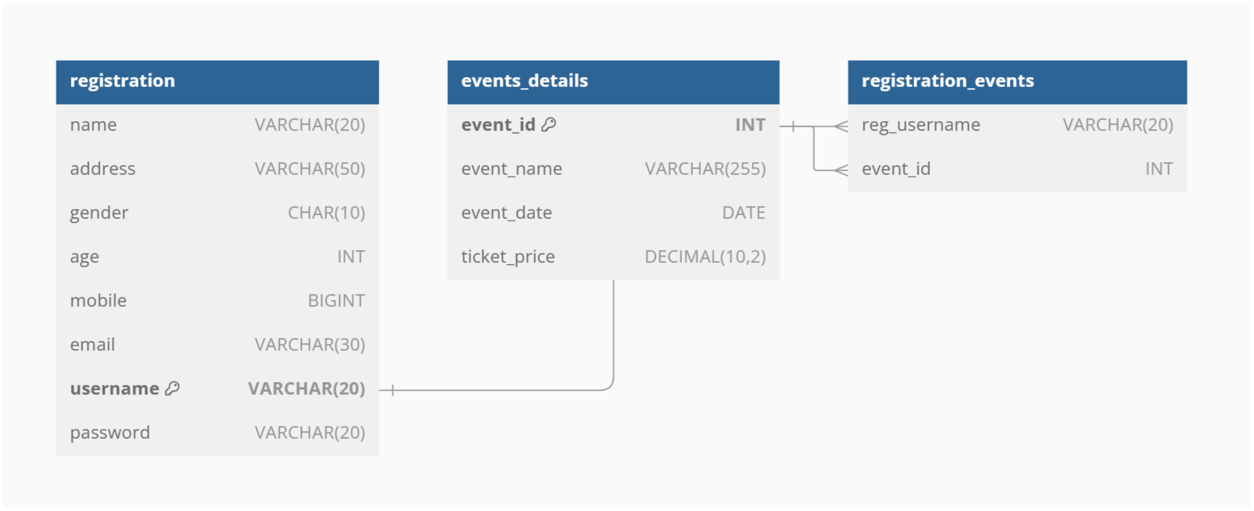
3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

- **Hardware:**
    - Processor: Intel i5 or equivalent
    - RAM: 4GB minimum
    - Storage: 500MB free disk space
- **Software:**
    - Java Development Kit (JDK)
    - MySQL Database Server
    - Java Database Connectivity (JDBC)
    - Integrated Development Environment (IDE) like Eclipse or IntelliJ

3.3 ARCHITECTURE DIAGRAM



3.4 ER DIAGRAM



3.5 NORMALIZATION

After normalization:

- Event Details Table: Already in 3NF, no changes needed.
- Registration Table: Split into:
  - User_Info for user personal details.
  - User_Auth for authentication details.

a. User_Info Table

Stores user personal details:

```
CREATE TABLE User_Info (
   user_id INT NOT NULL AUTO_INCREMENT,
   name VARCHAR(20),
   address VARCHAR(50),
   gender CHAR(10),
   age INT,
   mobile INT,
   email VARCHAR(30),
   PRIMARY KEY (user_id)
);
```

b. User_Auth Table

Stores user authentication details:

```
CREATE TABLE User_Auth (

   user_id INT NOT NULL,

   username VARCHAR(20) NOT NULL,

   password VARCHAR(20) NOT NULL,

   PRIMARY KEY (user_id),

   FOREIGN KEY (user_id) REFERENCES User_Info(user_id)

);
```

CHAPTER 4
PROGRAM CODE

EVENT DETAILS

```
CREATE TABLE events_details (
   event_id INT NOT NULL AUTO_INCREMENT,
   event_name VARCHAR(255) NOT NULL,
   event_date DATE NOT NULL,
   ticket_price DOUBLE DEFAULT NULL,
   PRIMARY KEY (event_id)
);

Select * from event_details;
```

REGISTRATION

```
create database users;
use users;
create table registration(name varchar(20),address varchar(50),gender char(10),
age int, mobile int, email varchar(30),username varchar(20), password
varchar(20));
desc registration;
select * from registration;
```

REGISTRATION FORM

Load Events into Dropdown

```
public static void loadEventsIntoDropdown() {
    try {
        Connection conn = createConnection();
        String sql = "SELECT event_name FROM event_details";
        PreparedStatement statement = conn.prepareStatement(sql);
        ResultSet resultSet = statement.executeQuery();
        eventDropdown.removeAllItems();
        while (resultSet.next()) {
            String eventName = resultSet.getString("event_name");
            eventDropdown.addItem(eventName);
        }
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Register User

```
private void registerUser(JRadioButton rbFemale, JRadioButton rbMale) {
    try (Connection con = createConnection()) {
        String query = "INSERT INTO registration VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setString(1, txtName.getText());
        ps.setString(2, txtAddress.getText());
        ps.setString(3, rbFemale.isSelected() ? "Female" : "Male");
        ps.setInt(4, Integer.parseInt(txtAge.getText()));
        ps.setString(5, txtmb.getText());
        ps.setString(6, txtEmail.getText());
        ps.setString(7, txtUsername.getText());
        ps.setString(8, new String(txtPassword.getPassword()));
        ps.executeUpdate();
        JOptionPane.showMessageDialog(this, "Registered successfully!");
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    }
    loadEventsIntoDropdown();
}

public static Connection createConnection() throws SQLException, ClassNotFoundException {
    Class.forName("com.mysql.cj.jdbc.Driver");
    return DriverManager.getConnection("jdbc:mysql://127.0.0.1:3308/users", "root", "Abi");
}
```

ADD EVENT FORM

Adding an Event to the Database

```
private void addEventToDatabase() {
    String eventName = txtEventName.getText();
    String eventDate = txtEventDate.getText();
    String ticketPrice = txtTicketPrice.getText();

    if (eventName.isEmpty() || eventDate.isEmpty() || ticketPrice.isEmpty()) {
        JOptionPane.showMessageDialog(this, "All fields are required!");
        return;
    }

    try (Connection con = createConnection()) {
        String query = "INSERT INTO event_details (event_name, event_date, ticket_price)
VALUES (?, ?, ?)";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setString(1, eventName);
        ps.setString(2, eventDate);
        ps.setString(3, ticketPrice);
        ps.executeUpdate();

        // Show success message
        JOptionPane.showMessageDialog(this, "Event Added Successfully!");

        // Clear the text fields after adding the event
        txtEventName.setText("");
        txtEventDate.setText("");
        txtTicketPrice.setText("");
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error adding event. Please try again.");
    }
}

private static Connection createConnection() throws SQLException, ClassNotFoundException {
   Class.forName("com.mysql.cj.jdbc.Driver");
   return DriverManager.getConnection("jdbc:mysql://127.0.0.1:3308/users", "root", "Abi");
}
```

Delete Event

```
private void deleteEvent(String eventName) {
   try (Connection conn = createConnection()) {
      String sql = "DELETE FROM event_details WHERE event_name = ?";
      PreparedStatement stmt = conn.prepareStatement(sql);
      stmt.setString(1, eventName);
      int rowsAffected = stmt.executeUpdate();
      if (rowsAffected > 0) {
         JOptionPane.showMessageDialog(this, "Event " + eventName + " deleted successfully.");
      } else {
         JOptionPane.showMessageDialog(this, "Event not found.");
      }
   } catch (SQLException | ClassNotFoundException ex) {
      ex.printStackTrace();
      JOptionPane.showMessageDialog(this, "Error deleting event.");
   }
}
```

Modify Event

```
private void modifyEvent() {
   String eventName = JOptionPane.showInputDialog("Enter the event name to modify:");

   if (eventName != null && !eventName.isEmpty()) {
      JTextField txtEventDate = new JTextField(20);
      JTextField txtTicketPrice = new JTextField(20);

      Object[] message = {
         "Event Date:", txtEventDate,
         "Ticket Price:", txtTicketPrice
      };

      int option = JOptionPane.showConfirmDialog(this, message, "Modify Event - " +
eventName, JOptionPane.OK_CANCEL_OPTION);
```

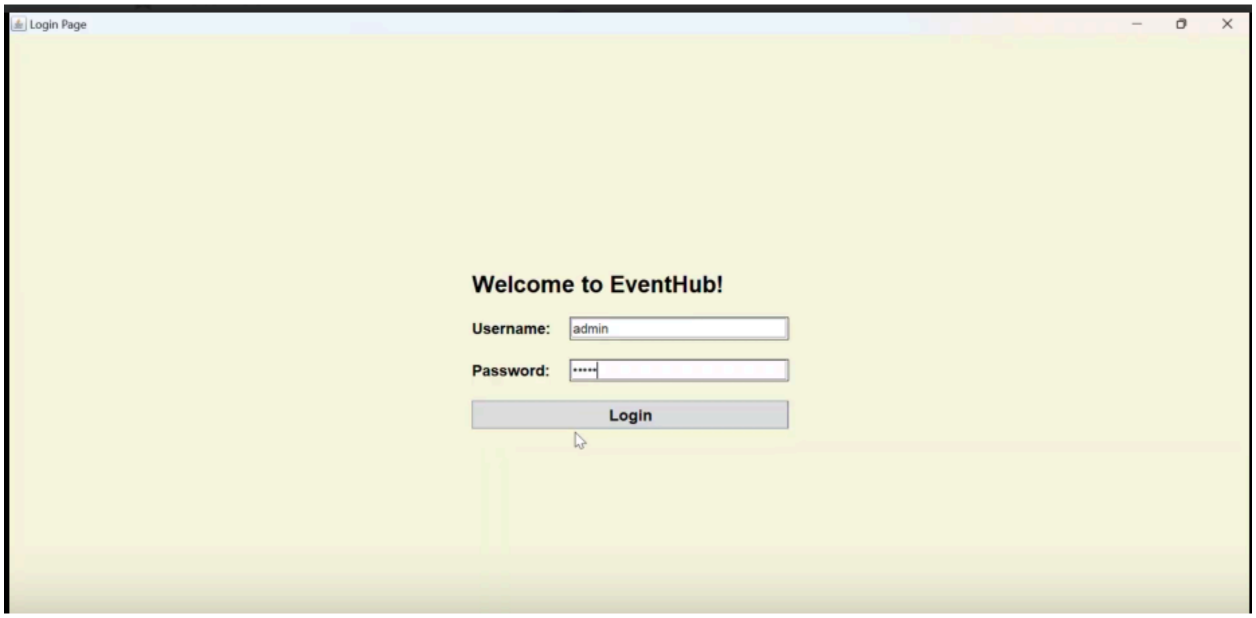
```
        if (option == JOptionPane.OK_OPTION) {
            String newEventDate = txtEventDate.getText();
            String newTicketPrice = txtTicketPrice.getText();

            if (newEventDate.isEmpty() || newTicketPrice.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill in both fields.");
                return;
            }

            updateEvent(eventName, newEventDate, newTicketPrice);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Event name cannot be empty.");
    }
}
```

Update Event

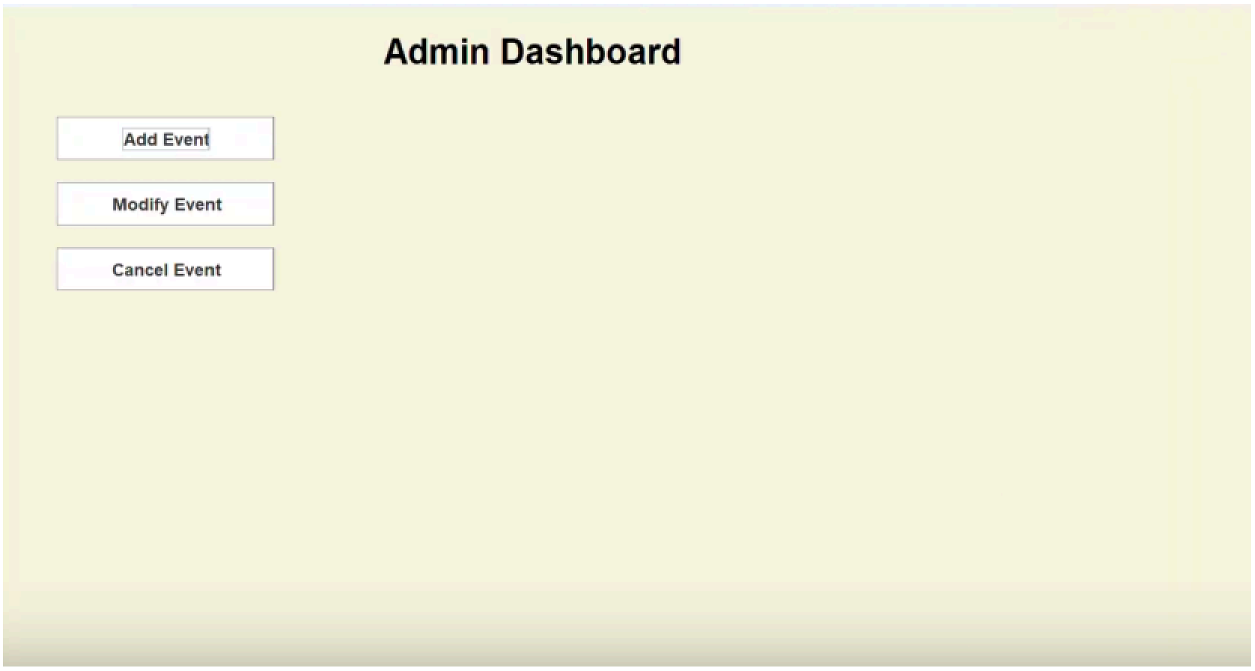
```
private void updateEvent(String eventName, String newEventDate, String newTicketPrice) {
    try (Connection conn = createConnection()) {
        String sql = "UPDATE event_details SET event_date = ?, ticket_price = ? WHERE
event_name = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, newEventDate);
        stmt.setString(2, newTicketPrice);
        stmt.setString(3, eventName);

        int rowsAffected = stmt.executeUpdate();

        if (rowsAffected > 0) {
            JOptionPane.showMessageDialog(this, "Event updated successfully!");
        } else {
            JOptionPane.showMessageDialog(this, "Event not found.");
        }
    } catch (SQLException | ClassNotFoundException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error updating event.");
    }
}
```

RESULTS SNAPSHOTS

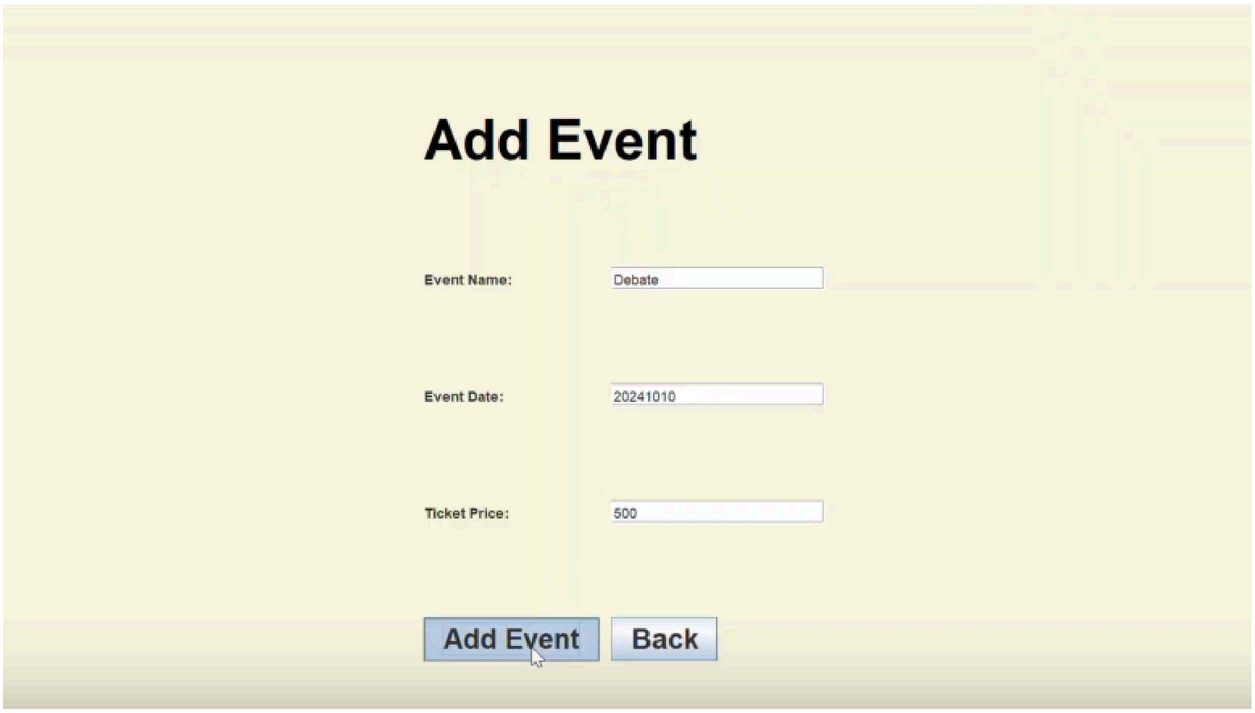
LOGIN PAGE



ADMIN DASHBOARD

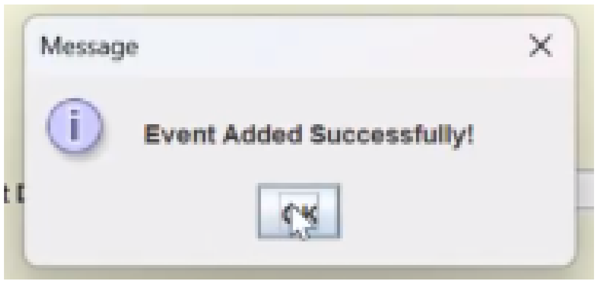


ADD EVENT



Add Event

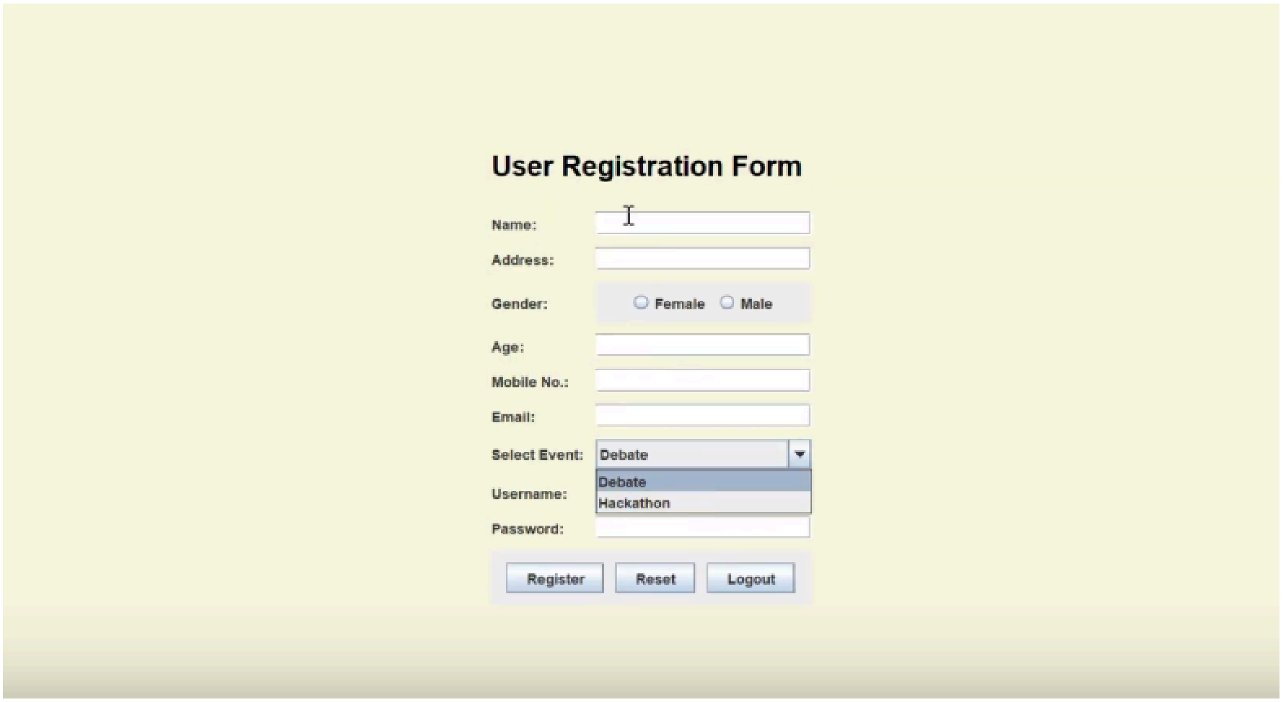
Event Name:	Debate
Event Date:	20241010
Ticket Price:	500



Message

Event Added Successfully!

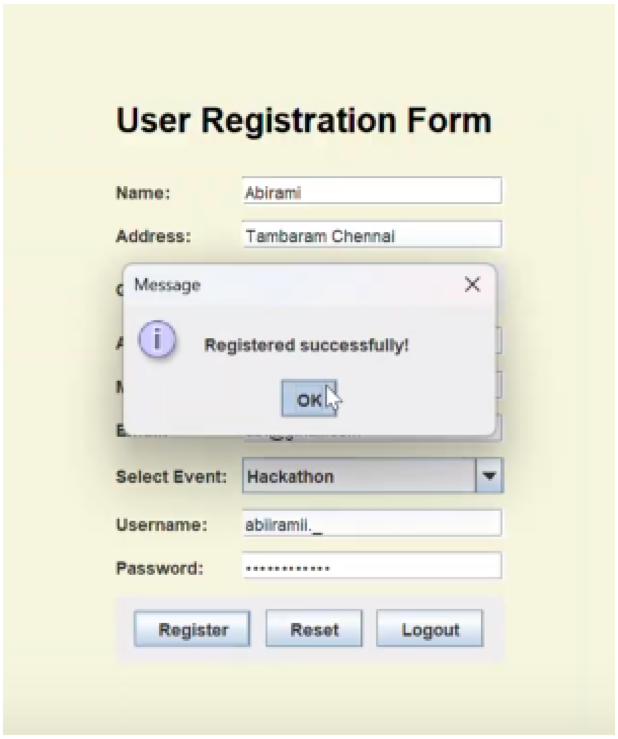
EVENT REGISTRATION - USER



User Registration Form

Name:	
Address:	
Gender:	Female Male
Age:	
Mobile No.:	
Email:	
Select Event:	Debate
Username:	
Password:	

Debate
Hackathon

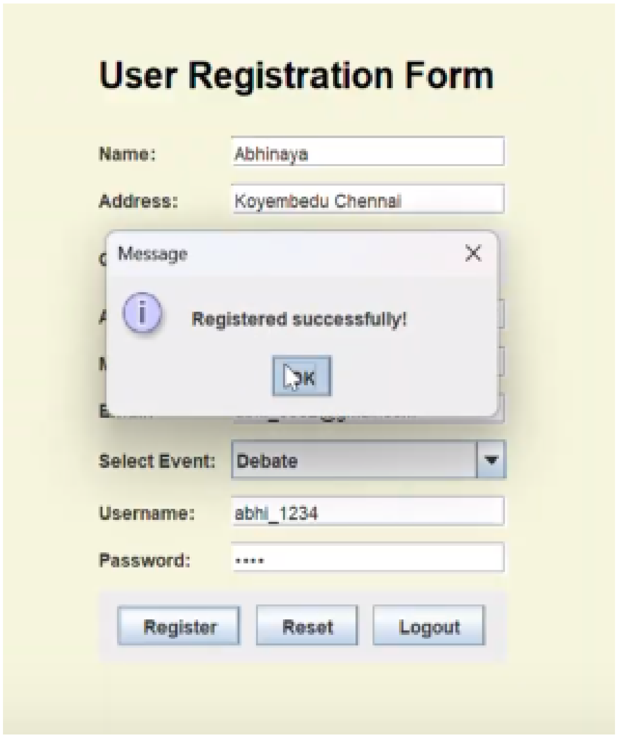


User Registration Form

Name:	Abirami
Address:	Tambaram Chennai
Select Event:	Hackathon
Username:	abiiramii._
Password:	•••••••••••••

Message

Registered successfully!

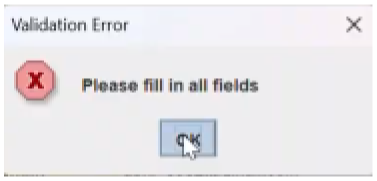


User Registration Form

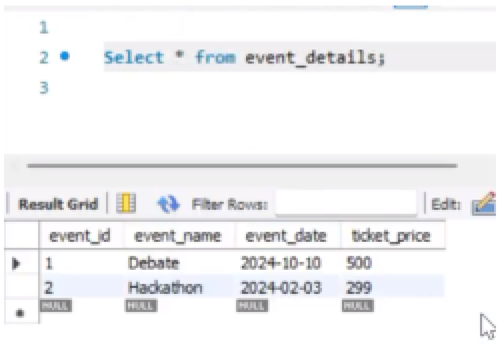
Name:	Abhinaya
Address:	Koyembedu Chennai
Select Event:	Debate
Username:	abhi_1234
Password:	••••

Message

Registered successfully!

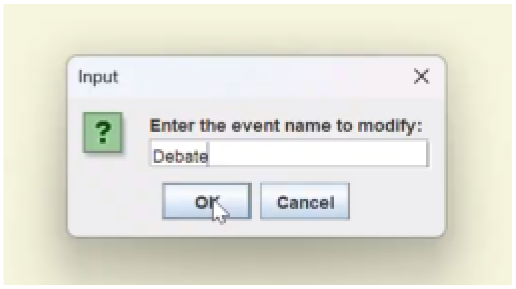

Validation Error

Please fill in all fields


```
Select * from event_details;
```

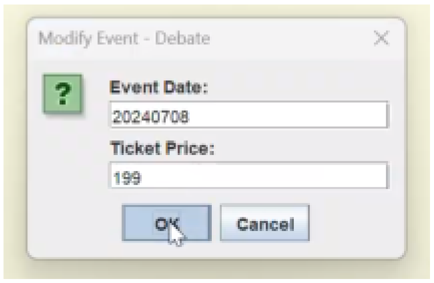
event_id	event_name	event_date	ticket_price
1	Debate	2024-10-10	500
2	Hackathon	2024-02-03	299
NULL	NULL	NULL	NULL

MODIFY EVENT


Input

Enter the event name to modify:

Debate

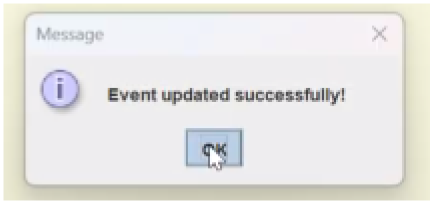

Modify Event - Debate

Event Date:

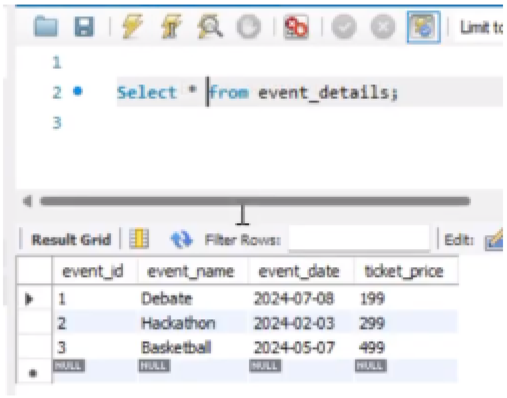
20240708

Ticket Price:

199

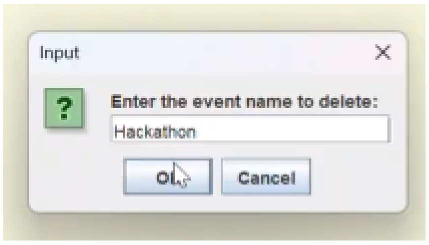

Message

Event updated successfully!


```
Select * from event_details;
```

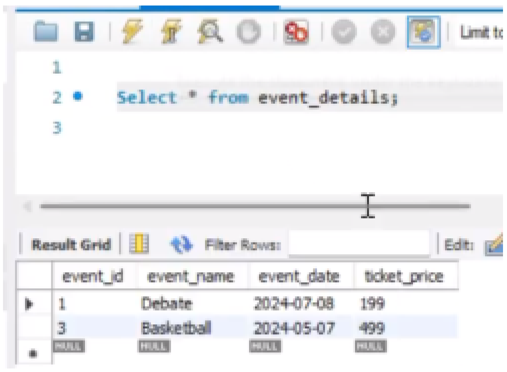
event_id	event_name	event_date	ticket_price
1	Debate	2024-07-08	199
2	Hackathon	2024-02-03	299
3	Basketball	2024-05-07	499
NULL	NULL	NULL	NULL

CANCEL EVENT


Input

Enter the event name to delete:

Hackathon


```
Select * from event_details;
```

event_id	event_name	event_date	ticket_price
1	Debate	2024-07-08	199
3	Basketball	2024-05-07	499
NULL	NULL	NULL	NULL

6. CONCLUSION

The Event Registration System demonstrates how DBMS principles and tools can streamline event management. By using MySQL, the system effectively handles complex data requirements while maintaining data integrity and security. The project showcases the synergy between Java and SQL, providing a scalable, efficient solution for event registration. This system has potential for expansion, including adding reporting capabilities, advanced data analytics, or a web-based interface.

7. REFERENCES

1. https://docs.oracle.com/javase/
2. https://dev.mysql.com/doc/
3. https://www.javacodegeeks.com/