International Islamic University, Islamabad

Project Report

Event Organizer App

May 27, 2024

M Asad Bashir | 4296-BSSE-F21-A

Advance Computer Programming

Sir Zohair Ahmed

This is a report about semester project made for ACP, it was made using java swing, now we will breifly discuss the project application.

Contents

Event Organizer

1. Introduction

This Java Swing application functions as a simple event organizer tool. It allows users to create, view, update, and delete event information (CRUD operations). Additionally, the app incorporates user authentication to control access and maintain data integrity.

1.1 Project Overview

1.1.1 Purpose

This is a Java Swing application designed to be an event organizer tool. It allows users to create, view, edit, and delete events. Additionally, it implements user authentication to ensure only registered users can manage events.

1.1.2 Main Features

User Authentication: Users can register for an account with a username and password. Login is required to access event management functionalities.

Event Management:

- Create new events with details like title, date, time, location, and description.
- View a list of all created events.
- Edit existing events to modify details.
- Delete unwanted events.

Basic User Interface: The application utilizes Swing components for a user-friendly interface with clear functionalities.

1.1.3 Target Users

This application is targeted towards individuals or groups who need to organize events. It can be used for personal event management, club activities, or small business event planning.

1.2 Technologies Used

- **Java:** The primary programming language used to develop the application logic.
- **Java Swing:** A library within Java for creating graphical user interfaces (GUI) with components like buttons, text fields, and tables.
- **AWT (Abstract Window Toolkit):** A lower-level toolkit within Java that Swing utilizes for basic GUI functionalities.
- **JDBC (Java Database Connectivity):** An API that allows Java applications to connect and interact with relational databases.
- **MySQL:** A popular open-source relational database management system used to store and manage event data (user accounts and event details).

2. GUI Design

2.1 Overview

2.1.1 Graphical User Interface Design

The application utilizes a simple and intuitive design with Swing components to ensure user-friendliness. Here's a breakdown of the design choices:

2.1.2 Layout

- **Main Window:** Four buttons are displayed in the center for CRUD functionality, and an exit button is available in the end to quit the app.
- **Event List:** A JTable is used to display a list of all created events with columns for title, date, and time. Users can select an event for further actions. Refresh, Update and delete buttons are available at end of the table.
- **Event Form:** Event forms like update, delete or create forms have similar interface, size and structure. The only difference is that main screen has buttons in the center while event forms have textFields and labels in those locations.

2.1.3 Design Patterns

Model-View-Controller (MVC): This pattern can be loosely implemented to separate the application logic (model), data presentation (view), and user interaction handling (controller). This promotes code maintainability and reusability.

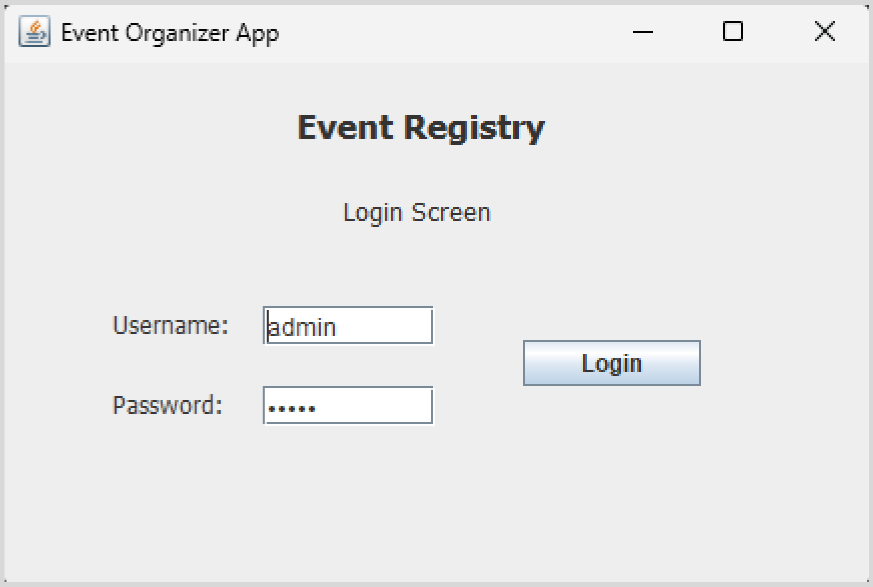
2.1.4 User Experience Considerations

- **Clear Labeling:** All components are labeled clearly to indicate their purpose.
- Intuitive Button Placement: Buttons for common actions (create, edit, delete) are placed strategically for easy access.
- **Input Validation:** User input is validated to ensure data integrity (e.g., checking for empty fields or invalid dates).
- **Error Handling:** Informative messages are displayed in case of errors during login, event creation, or other actions.

2.2 Screenshots

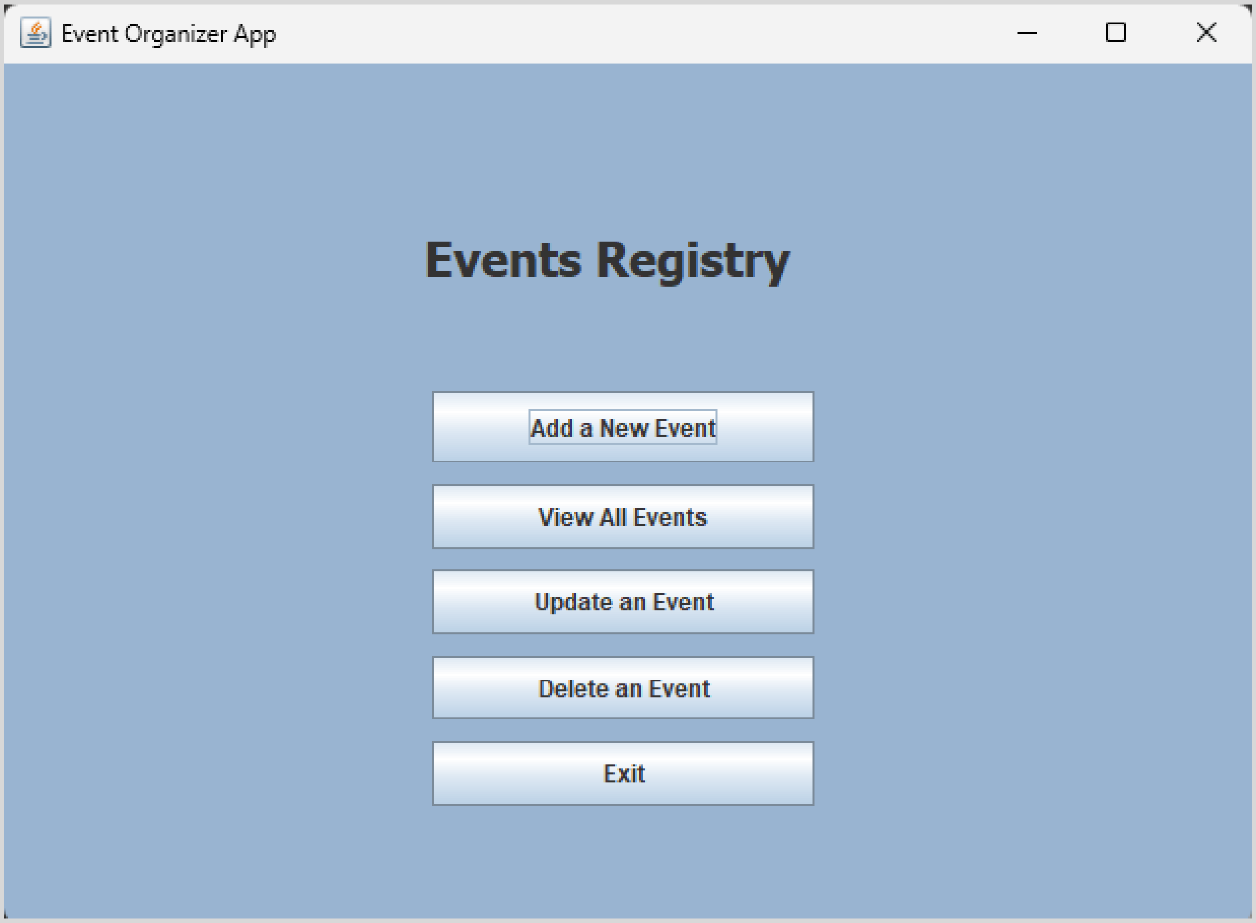
2.2.1 Login Screen (initial view)

- **Username field:** Users enter their username for login.
- **Password field:** Users enter their password (masked for security).
- **Login button:** Attempts to authenticate the user based on entered credentials.



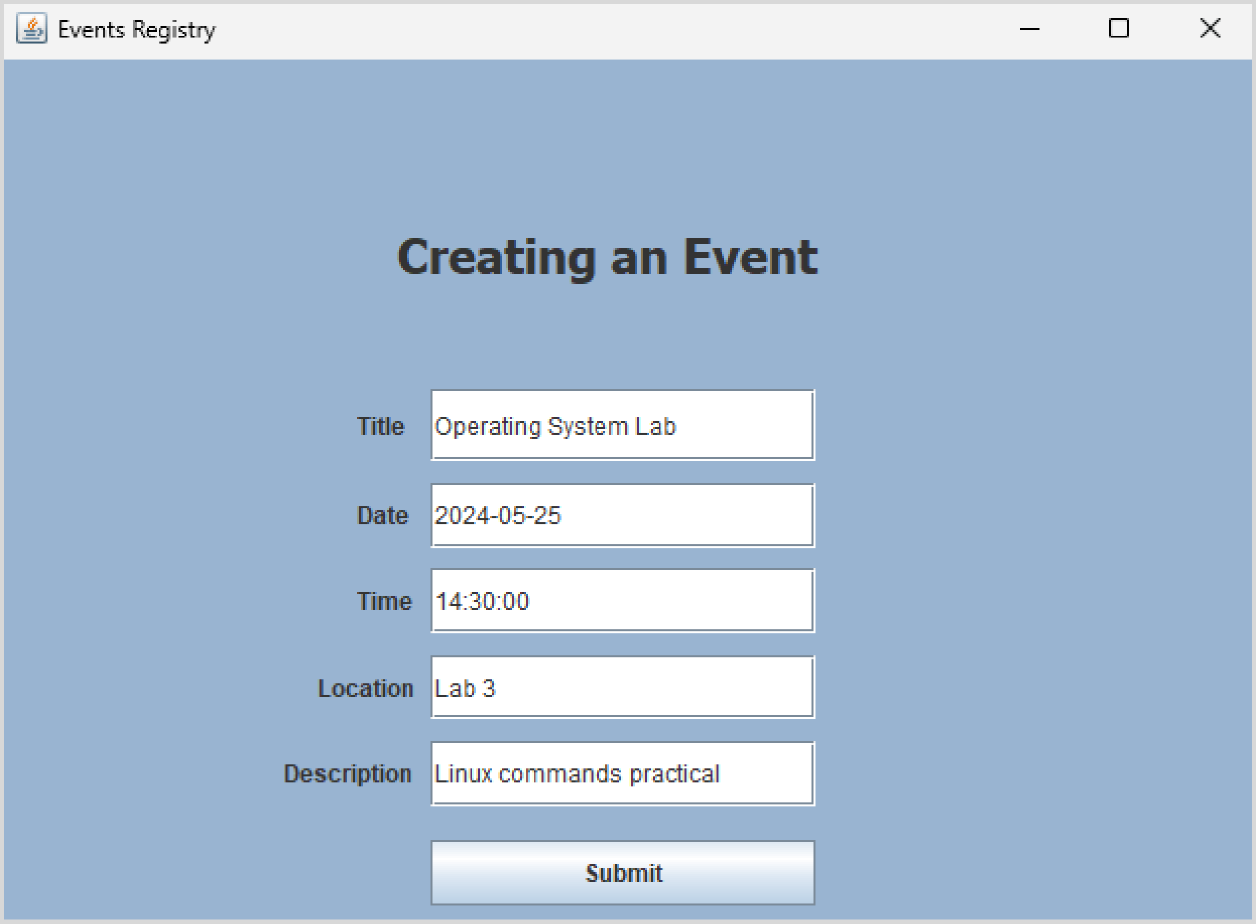
2.2.2 Main Menu (after successful login)

- **Menu bar:** Might contain options for user profile management, logout, or application settings (depending on implementation).
- **Event List (JTable):** Displays a list of all created events with title, date, and time. Selecting an event allows further actions.
- **Create Event button:** Opens a new window for creating a new event.
- **Edit Event button:** Enables editing the details of the selected event (navigates to an edit form).
- **Delete Event button:** Prompts for confirmation before deleting the selected event.



2.2.3 Create Event Form

- **Event Details:** Labeled fields for title, date (date picker), time (time picker), location, and description.
- **Save button:** Saves the entered event details (creating a new event or updating an existing one).
- **Cancel button:** Closes the form without saving changes.



Creating an Event

Title	Operating System Lab
Date	2024-05-25
Time	14:30:00
Location	Lab 3
Description	Linux commands practical

2.3 Navigation Flow

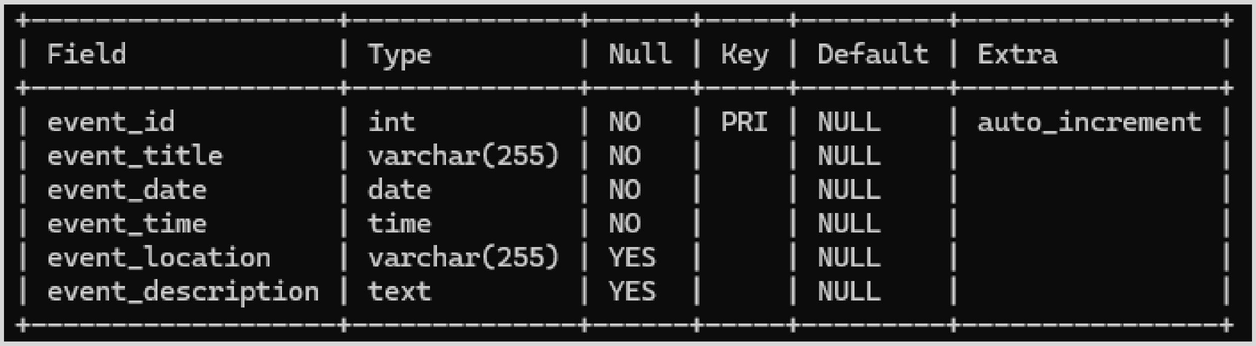
The application follows a simple navigation flow:

- Initially, the login screen is displayed.
- Users enter their credentials and click "Login".
    - Upon successful login, the main menu is displayed with the list of events.
    - If login fails, an error message is shown, and users can retry.
- In the main menu, users can:
    - Click "Create Event" to open the event creation form.
    - Select an event in the list and click "Edit Event" to open the edit form pre-populated with existing details.
    - Click "Delete Event" to initiate the deletion process with confirmation.
- The event form allows users to enter details and save the event (creating a new one or updating an existing one).

This design focuses on a clear and user-friendly experience for managing events. It can be customized further based on specific needs and preferences.

3. Database Design

3.1 Schema Diagram



```
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| event_id          | int          | NO   | PRI | NULL    | auto_increment |
| event_title       | varchar(255) | NO   |     | NULL    |                |
| event_date        | date         | NO   |     | NULL    |                |
| event_time        | time         | NO   |     | NULL    |                |
| event_location    | varchar(255) | YES  |     | NULL    |                |
| event_description | text         | YES  |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
```

3.2 Table Descriptions

This table describes the structure of a database table likely used for storing event information. Let's break down each column:

3.2.1 event_id | int | NO | PRI | NULL | auto_increment |

- This is the primary key of the table. It's an integer (int) that uniquely identifies each event.
- The NO in the Null column indicates that this field cannot be empty for any record in the table.
- PRI stands for Primary Key, which enforces uniqueness of this column's values.
- auto_increment means the database automatically assigns a unique increasing integer for each new event record inserted.

3.2.2 event_title | varchar(255) | NO | | NULL | |

- This column stores the title of the event as a string (varchar).
- The maximum length allowed for the title is 255 characters.
- NO in the Null column indicates the title cannot be empty.

3.2.3 event_date | date | NO | | NULL | |

- This column stores the date of the event as a date data type.
- It cannot be empty (indicated by NO in the Null column).

3.2.4 event_time | time | NO | | NULL | |

- This column stores the time of the event as a time data type.
- Similar to date and title, it cannot be empty.

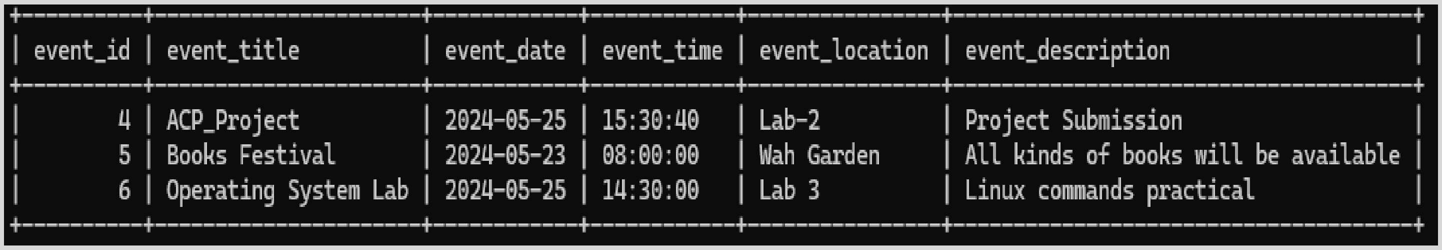
3.2.5 event_location | varchar(255) | YES | | NULL | |

- This column stores the location of the event as a string (varchar) with a maximum length of 255 characters.
- Unlike other columns so far, location can be empty (indicated by YES in the Null column).

3.2.6 event_description | text | YES | | NULL | |

- This column stores a detailed description of the event as text.
- Text data type allows for longer descriptions compared to varchar.
- Similar to location, the description can be empty.

3.3 Sample Data

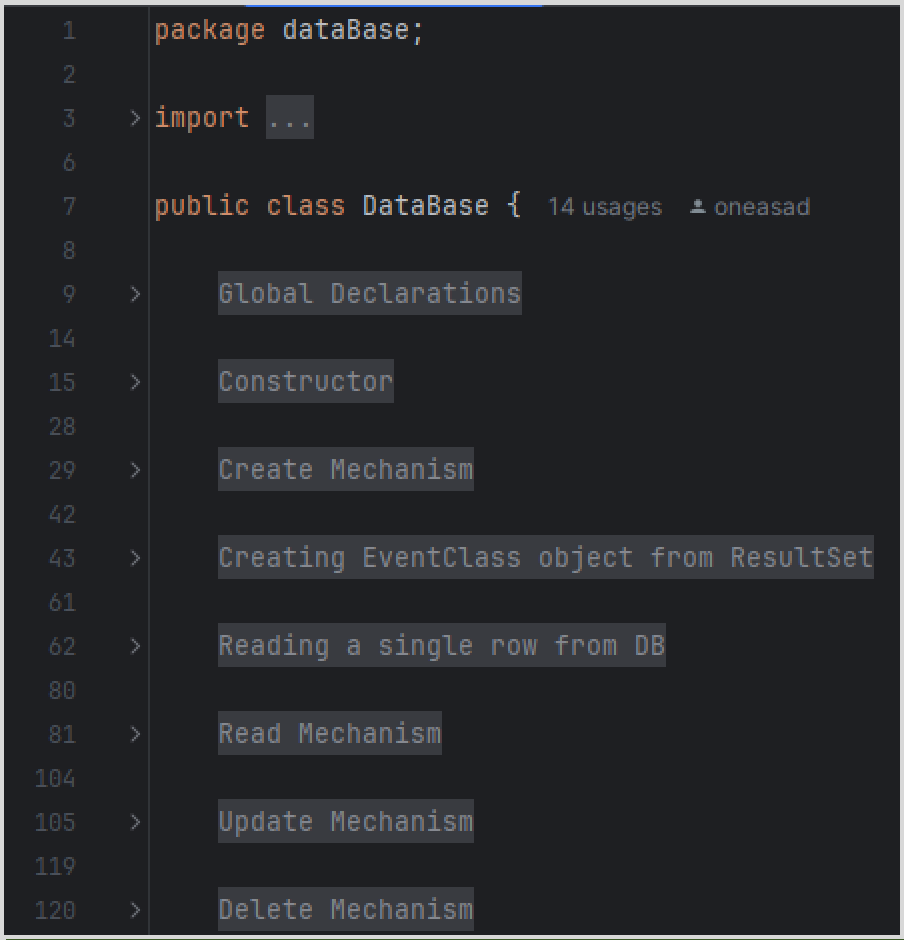


```
+----------+---------------------+------------+------------+----------------+-----------------------------------+
| event_id | event_title         | event_date | event_time | event_location | event_description                 |
+----------+---------------------+------------+------------+----------------+-----------------------------------+
|        4 | ACP_Project         | 2024-05-25 | 15:30:40   | Lab-2          | Project Submission                |
|        5 | Books Festival      | 2024-05-23 | 08:00:00   | Wah Garden     | All kinds of books will be available |
|        6 | Operating System Lab | 2024-05-25 | 14:30:00   | Lab 3          | Linux commands practical          |
+----------+---------------------+------------+------------+----------------+-----------------------------------+
```

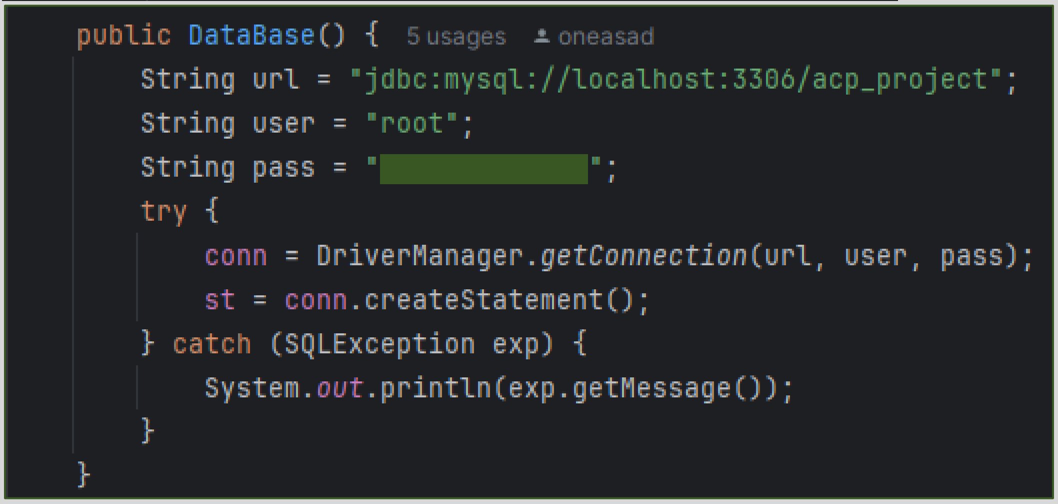
4. Implementation

4.1 Code snippets

4.1.1 Database Connection

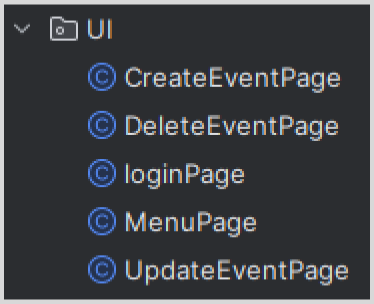


```
package dataBase;

import ...

public class DataBase {  14 usages  oneasad

    Global Declarations

    Constructor

    Create Mechanism

    Creating EventClass object from ResultSet

    Reading a single row from DB

    Read Mechanism

    Update Mechanism

    Delete Mechanism
```

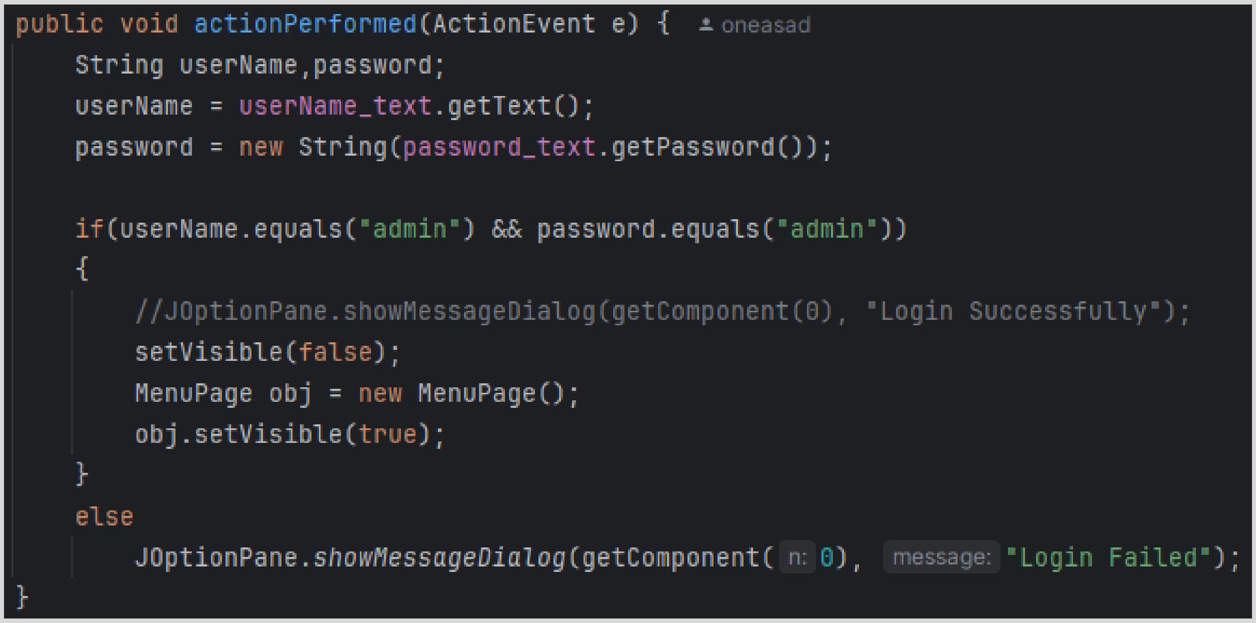


```
public DataBase() {  5 usages  oneasad
    String url = "jdbc:mysql://localhost:3306/acp_project";
    String user = "root";
    String pass = "[redacted]";
    try {
        conn = DriverManager.getConnection(url, user, pass);
        st = conn.createStatement();
    } catch (SQLException exp) {
        System.out.println(exp.getMessage());
    }
}
```

4.1.2 CRUD Operations



4.1.3 User Authentication



```
public void actionPerformed(ActionEvent e) {  oneasad
    String userName,password;
    userName = userName_text.getText();
    password = new String(password_text.getPassword());

    if(userName.equals("admin") && password.equals("admin"))
    {
        //JOptionPane.showMessageDialog(getComponent(0), "Login Successfully");
        setVisible(false);
        MenuPage obj = new MenuPage();
        obj.setVisible(true);
    }
    else
        JOptionPane.showMessageDialog(getComponent( n: 0),  message: "Login Failed");
}
```

4.2 Explanation

The logic behind my code is that I have divided my code into packages and classes, each class has its own specific job to do, there are separate java classes for each CRUD operation interface, and the all interfaces are placed inside a UI package, database logic and CRUD mechanism is implemented in a separate java class that is placed inside a database package. There is main class as well that initiates the app and displays login page, if login is successful that an instance of main menu page is created, that displays various options. When user selects an option from Main Menu, then a new class of that option is initiated and destroyed after user has done its work. There is an option to view all events. That view is made up in a separate read class. It contains UI of that displays all entries in table and three buttons at the bottom namely Update, Refresh and Delete. The refresh button refreshes the view to integrate any changes caused. The other two buttons are in-active by default but when user selects one of the rows, they become active and usable.

5. Conclusion

This Java Swing CRUD application provides a basic framework for managing events. By implementing user authentication and leveraging MVC principles, created a robust and secure tool for organizing events. The project is uploaded to Github repository, you can check it out here.

5.1 Project Summary

This project involved developing a Java Swing application for event management. It allows users to log in, and manage events. Key functionalities include:

- **User Authentication:** Users can create accounts and log in for secure access.
- **Event Management:**
  - Create new events with details like title, date, time, location, and description.
  - View a list of all created events.
  - Edit existing events to modify details.
  - Delete unwanted events.
- **Basic User Interface:** A Swing-based interface provides a user-friendly experience for managing events.

5.2 Learnings

Here are some potential takeaways from this project:

- **Java Swing:** I gained experience working with Swing components to design and developed a graphical user interface.
- **JDBC and Database Interaction:** I learned how to connect to a database (MySQL) using JDBC to store and retrieve event data.
- **User Authentication:** I implemented a basic user authentication system for secure access control.
- **Project Planning and Development:** I practiced planning, development, and potentially deployment aspects of a software application.

5.3 Future Improvements

Several improvements can be considered to enhance the application:

- **Guest Management:** We can implement functionalities to invite guests (registered users or others) and track their RSVPs.
- **Event Categories:** We can allow users to categorize events for better organization.
- **Calendar Integration:** We can integrate with a calendar application to display events visually.
- **Notifications and Reminders:** Implement features for sending notifications or reminders about upcoming events.
- **User Interface Enhancements:** We can improve the UI design for better aesthetics and user experience (icons, themes, etc.)
- **Security Enhancements:** We can consider more robust password hashing techniques and potential access control mechanisms.
- **Scalability:** Design the application to handle a larger number of users and events efficiently.