Event Management System

**CSE241 Object-Oriented Computer Programming – 2nd Semester 2024/2025**

***Group 5 section 10***

Team Members:

* Adham Amr – 24P0261 (All Organizer Class (Backend) , All Organizer Dashboard (Front End) , Wrote Report ,demo video)
* Omar Ayman – 24P0267 (Login /Register Classes , User Class , Database Class , Attendee Page (Front End) , validation)
* Amr Ahmed Hashem – 24P0417 ( Attendee Class (Backend) , Attendee Dashboard (Front End) )
* Ahmed Ashraf – 24P0405 ( Room Class , Admin Page )
* Mohamed Osama – 24P0235 ( Admin Class , Admin Page )
* Abdulrahman Baher – 24P0263 ( Organizer Dashboard , Events Class )

*Note that some responsibilities are repeated across members as they worked on them together.*

***Github Repository*** : <https://github.com/OmarAyman33/Event-management-system>

Contents

[1. Introduction 2](#_Toc197709738)

[2. Problem Description 2](#_Toc197709739)

[3. Detailed Analysis 3](#_Toc197709740)

[3.1 Functional Requirements 3](#_Toc197709741)

[3.2 Non-Functional Requirements 3](#_Toc197709742)

[4. Design 3](#_Toc197709743)

[4.1 Overall System Structure 3](#_Toc197709744)

[4.2 Object Oriented Backend ( phase 1 ) 4](#_Toc197709745)

[4.2.1 Class Diagram ( UML ) 4](#_Toc197709746)

[4.2.2 Description of core classes 5](#_Toc197709747)

[4.2.3 Application of OOP Principles 5](#_Toc197709748)

[4.3 Graphical User Interface Design 5](#_Toc197709749)

[5. Testing 6](#_Toc197709750)

[6. Sample Output 6](#_Toc197709751)

[6.1 GUI Screenshots 6](#_Toc197709752)

[6.2 Video Demo 9](#_Toc197709753)

[7. Conclusion 9](#_Toc197709754)

# Introduction

This report’s purpose is to summarize the actions that our team did to complete the project. All functionalities were designed and implemented with scalability, modularity, and user-friendliness in mind. The source code, class designs, and sample data are publicly available on the project’s GitHub repository: <https://github.com/OmarAyman33/Event-Management-System.git>

# Problem Description

The scope of the project was to develop a functional java-based Event Management System with:

* OOP backend for users (Admin, Organizer, Attendee), events, rooms, categories.
* JavaFX GUI for registration, login, and role-specific dashboards.
* Admin management of rooms and categories.
* Organizer creation and management of events, including room booking.
* Attendee event browsing, ticket purchasing via a wallet system.

# Detailed Analysis

## 3.1 Functional Requirements

* **Admin:** Register/Login; Add/Remove rooms; Add categories; View all rooms, events, attendees; Remove events/attendees.
* **Organizer:** Register/Login; CRUD operations for their events (booking rooms, room cost deducted from wallet); View available rooms by date; Manage wallet.
* **Attendee:** Register/Login; View/Filter events; Purchase tickets (debits wallet); View registered events; Manage wallet and interests; Request refunds.
* **Event/Room/Category:** Events have date, room, category, price, organizer, attendees. Rooms have name, capacity, price, booked events (day-based availability). Categories classify events.
* **GUI:** User-friendly screens for registration, login, dashboards. Interactive forms and lists. Feedback messages.

## Non-Functional Requirements

* Usability: Intuitive GUI.
* Modularity: Distinct backend/GUI modules.
* Readability: Well-commented, consistent code.
* Error Handling: Basic input and operational error management.
* Performance: Responsive for typical desktop use with in-memory data.

# 4. Design

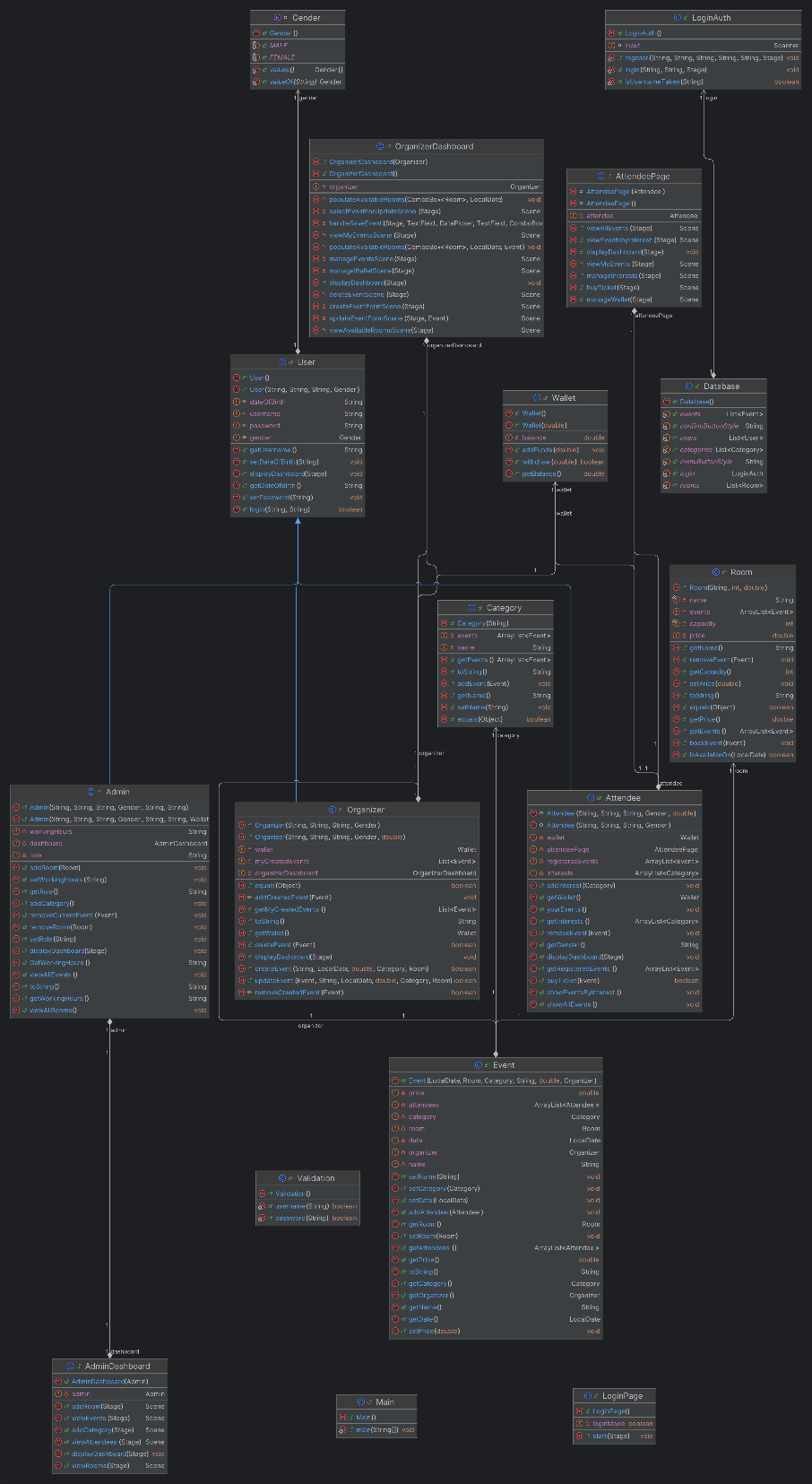
## 4.1 Overall System Structure

A two-tier Java desktop application:

1. **Backend Logic Layer:** Core classes, business logic, data management (Database class).
2. **Presentation Layer (GUI):** JavaFX for user interaction, communicating with the backend. LoginPage is the entry point, leading to role-specific dashboards.

## 4.2 Object Oriented Backend ( phase 1 )

### 4.2.1 Class Diagram ( UML )



### 4.2.2 Description of core classes

* **User (Abstract):** Base for users; common attributes (username, password), login, abstract displayDashboard.
* **Admin:** Manages rooms, categories, views system data. Has role, workingHours.
* **Organizer:** Manages own events (CRUD); wallet, myCreatedEvents list.
* **Attendee:** Registers for events; wallet, registeredEvents, interests.
* **Event:** Details like date (LocalDate), room, category, name, price, attendees, organizer.
* **Room:** name, capacity, price, events list. Day-based booking (isAvailableOn(LocalDate)).
* **Category:** name, list of events.
* **Wallet:** balance; methods addFunds, withdraw.
* **Database:** static ArrayLists for users, events, rooms, categories; dummy data initialization.
* **LoginAuth:** Handles login/registration logic.
* **Validation:** Static methods for username/password format validation.

### 4.2.3 Application of OOP Principles

* **Encapsulation:** Private/protected data members, public accessors/mutators.
* **Inheritance:** User as base for Admin, Organizer, Attendee.
* **Abstraction:** User class defines abstract displayDashboard.
* **Polymorphism:** user.displayDashboard(stage) invokes the correct user-specific GUI.
* **Association/Aggregation:** Links between classes like Event-Room, Organizer-Wallet.

## 4.3 Graphical User Interface Design

Implemented using JavaFX:

* **Key Screens:** LoginPage (toggles login/register). Role-specific dashboards (AdminDashboard, OrganizerDashboard, AttendeePage) with buttons for actions leading to sub-screens/forms using VBox, Button, Label, TextField, ListView, ComboBox, DatePicker.
* **Navigation:** Scene replacement on the primary Stage for different views. "Back" and "Log Out" buttons provide navigation control

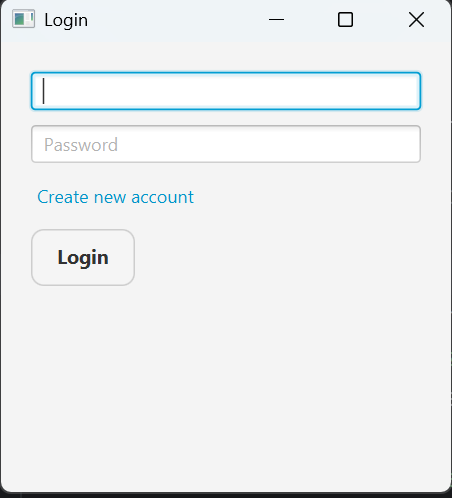
# 5. Testing

The primary testing approach involved manual walkthroughs of all GUI functionalities, covering different user roles and actions. This implicitly tested backend logic and integration between the GUI and backend. Key areas of focus included user authentication, data entry validation, CRUD operations for events and rooms, and financial transactions via the wallet system. Basic error conditions and edge cases (e.g., insufficient funds, unavailable rooms) were also manually tested.

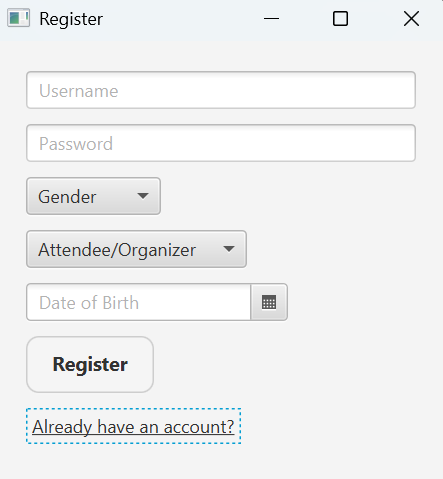
# 6. Sample Output

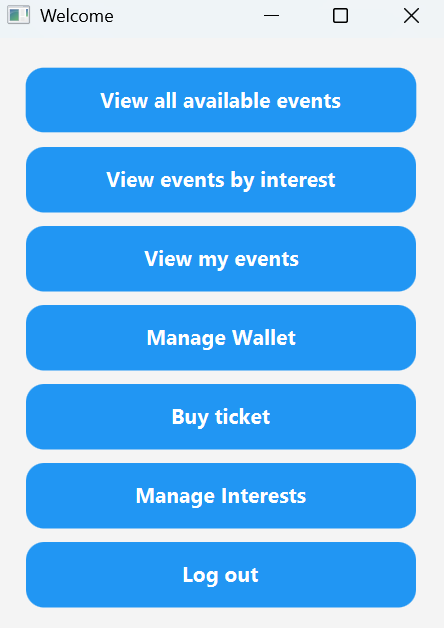
## 6.1 GUI Screenshots

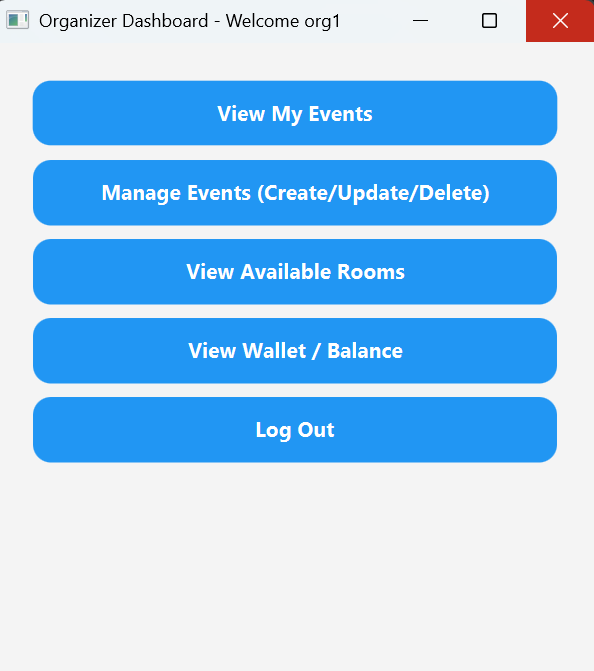
Login page:



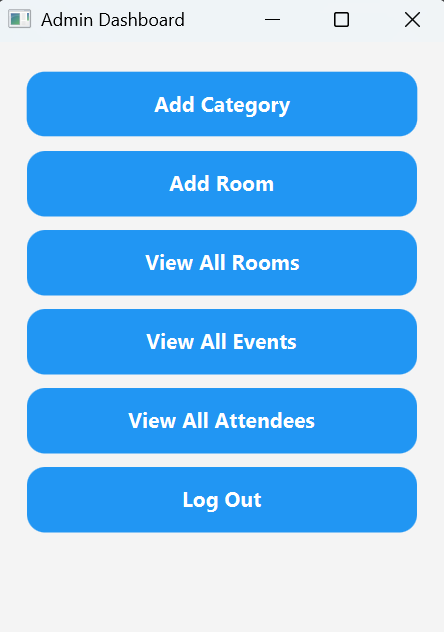
Register Page:



Attendee Dashboard:  


Organizer Dashboard:   


Admin Dashboard :



## 6.2 Video Demo

One of our team members recorded a two minute demo of the program showing core functionalities: <https://youtu.be/iDvzZphapYo>

# 7. Conclusion

The Event Management System project successfully implements a Java-based desktop application using OOP and JavaFX. It provides core functionalities for Admins, Organizers, and Attendees to manage events, rooms, and user interactions. The system demonstrates effective use of OOP principles for a modular backend and JavaFX for an interactive GUI. While some advanced features like persistent storage and networking were not realized, the project achieves its primary goal of creating a foundational event management platform and provided significant learning in software design and development.