Software Requirements and Design Document

for

Smart University Management System (SUMS)

Prepared by:   
Muhammad Asjad(22i-1227)  
Hassaan Afzal(22i-0918)  
Hamid Ali(22i-1047)

BlackCode

27th November , 2024

Table of Contents

Table of Contents ii

1. Introduction 3

1.1 Purpose 3

1.2 Product Scope 3

1.3 Title 3

1.4 Objectives 3

1.5 Problem Statement 4

2. Overall Description 5

2.1 Product Perspective 5

2.2 Product Functions 5

2.3 List of Use Cases 6

2.4 Extended Use Cases 7

2.5 Use Case Diagram 12

3. Other Nonfunctional Requirements 13

3.1 Performance Requirements 13

3.2 Safety Requirements 13

3.3 Security Requirements 14

3.4 Software Quality Attributes 15

3.5 Business Rules 15

3.6 Operating Environment 16

3.7 User Interfaces 16

4. Domain Model 26

5. System Sequence Diagram 26

6. Sequence Diagram 29

7. Class Diagram 44

8. Package Diagram 45

9. Deployment Diagram 45

# Introduction

## Purpose

The **Smart University Management System (SUMS)** aims to streamline and automate the administrative, academic, and operational processes of a university. It provides an integrated platform for students, faculty, and administrators to manage tasks like course enrollment, attendance, marks management, scheduling, and leave requests efficiently. By reducing manual efforts, SUMS ensures accuracy, transparency, and real-time data accessibility across all stakeholders.

## Product Scope

The **Smart University Management System** is a comprehensive solution designed to:

* Manage user roles for **students**, **faculty**, and **administrators**.
* Automate course-related processes such as **enrollment**, **attendance tracking**, **marks recording**, and **exam scheduling**.
* Provide faculty with tools to manage classes, record attendance, and approve leave requests.
* Offer students real-time access to their profiles, academic progress, attendance records, and course materials.
* Enable administrators to oversee the overall functioning of the system, manage users, and address operational needs.
* Enhance operational efficiency by eliminating manual errors and redundant tasks.

SUMS is a web-based application that can be deployed on-campus or hosted on a cloud platform, ensuring scalability and secure access for authorized users.

## Title

**Smart University Management System (SUMS)**

## Objectives

The key objectives of SUMS are:

1. **Streamline University Operations**: To digitize and automate core administrative processes, minimizing manual interventions and errors.
2. **Enhance User Experience**: Provide an intuitive platform for students, faculty, and administrators to perform tasks seamlessly.
3. **Improve Academic Management**: Enable effective management of course enrollments, schedules, attendance, and exams.
4. **Facilitate Data Access**: Allow real-time access to academic and operational data for informed decision-making.
5. **Optimize Resource Utilization**: Ensure optimal utilization of classrooms, exam halls, and other resources through automated scheduling.
6. **Boost Transparency and Accountability**: Offer clear insights into academic progress, attendance, and operational processes.

## Problem Statement

Managing a university’s operations manually or using disparate systems can lead to significant inefficiencies and challenges:

1. **Fragmented Data**: Universities often rely on multiple, disconnected systems to handle student records, faculty data, attendance, and exam schedules, resulting in data silos and inconsistencies.
2. **Manual Errors**: Recording attendance, managing marks, and scheduling exams manually are prone to human error, affecting the reliability of records.
3. **Time-Consuming Processes**: Processes like course registration, attendance tracking, and leave management consume excessive time when done manually, delaying decisions and operations.
4. **Lack of Transparency**: Students and faculty often lack real-time access to critical information, such as attendance records, exam schedules, or course statuses.
5. **Operational Overheads**: Administrative staff spend a disproportionate amount of time on repetitive tasks instead of focusing on strategic initiatives.

The **Smart University Management System (SUMS)** addresses these challenges by offering a centralized platform that integrates all core functions, reduces manual dependency, and enhances operational efficiency for all stakeholders.

# Overall Description

## Product Perspective

The Smart University Management System (SUMS) is a comprehensive software solution designed to streamline administrative, academic, and operational processes within a university. This system centralizes and automates key functions such as managing attendance, marks, class schedules, exam schedules, course enrollments, faculty leave requests, and user authentication. SUMS acts as a unified platform that bridges the gap between administrators, faculty, and students, promoting efficient communication and seamless operations.

The system is divided into modular packages, including UI Controllers, Data Access Objects (DAO), Services, and Entity Models. Each module is designed with a clear separation of concerns to ensure maintainability and scalability. The product is implemented using a robust and user-friendly architecture with intuitive interfaces for different user roles: admin, faculty, and students.

SUMS is developed as a standalone application, but it can integrate with existing university management tools via database connections. It offers an efficient database structure, ensuring data consistency and security for all users.

## Product Functions

The Smart University Management System provides the following key functionalities:

1. **User Authentication and Management**
   * Login functionality for admins, faculty, and students.
   * Role-based access control to ensure secure and restricted functionalities.
2. **Student Registration (Admin)**
   * Allows administrators to register new students and manage their profiles.
3. **Class Schedule Management**
   * Admins can create, update, and view class schedules for courses.
   * Students can view their respective class schedules.
4. **Exam Schedule Management**
   * Admins can create, update, and view exam schedules for all courses.
   * Students can view the exam schedule for their registered courses.
5. **Attendance Management**
   * Faculty can mark, update, and view attendance for enrolled students.
   * Students can view their attendance records.
6. **Marks Management**
   * Faculty can input, update, and manage marks for assignments, quizzes, or exams.
   * Students can view their marks for respective courses.
7. **Course Enrollment**
   * Students can enroll in available courses through an interactive interface.
   * Faculty can view the list of enrolled students.
8. **Faculty Leave Management**
   * Faculty can apply for leave, specifying reasons, start dates, and the number of days.
   * Admins can view pending leave requests and approve or disapprove them.
9. **Profile Management**
   * Students can view and update their profiles, ensuring accurate data in the system.

Each of these functions is implemented with robust error handling, input validation, and role-specific interfaces to provide a seamless user experience. The system is designed to reduce administrative overhead, improve transparency, and promote efficient academic management.

## List of Use Cases

**Use Cases Implemented**

1. **User Authentication**
   * Login for Admin, Faculty, and Students
2. **Student Profile Management**
   * View/Update Profile (Student)
3. **Student Registration**
   * Register new students (Admin)
4. **Course Enrollment**
   * Enroll in courses (Student)
5. **Class Schedule Management**
   * Manage Class Schedules (Admin)
   * View Class Schedules (Student)
6. **Exam Schedule Management**
   * Manage Exam Schedules (Admin)
   * View Exam Schedules (Student)
7. **Leave Management**
   * Request Leave (Faculty)
   * Manage Leave Requests (Admin)
8. **Attendance Management**
   * Mark Attendance (Faculty)
   * View Attendance (Student)
9. **Marks Management**
   * Add/Update Marks (Faculty)
   * View Marks (Student)

## Extended Use Cases

**Use Case 1: User Authentication**

**Name:** Login for Admin, Faculty, and Students  
**Primary Actor:** Admin, Faculty, Student  
**Scope:** Authentication System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Admin/Faculty/Student:** Want secure access to the system.
* **System:** Ensures only authorized users access functionalities.

**Precondition:**

* The user is registered in the system with valid credentials.

**Success Guarantee:**

* The user successfully logs in and is redirected to their dashboard.

**Main Success Scenario:**

1. User enters their username and password.
2. System validates the credentials.
3. If valid, the user is logged in and redirected to their respective dashboard (Admin, Faculty, or Student).

**Use Case 2: Student Profile Management**

**Name:** View/Update Profile  
**Primary Actor:** Student  
**Scope:** Profile Management System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Student:** Wants to view or update personal information accurately.

**Precondition:**

* The student is logged into the system.

**Success Guarantee:**

* The student's updated profile is saved successfully.

**Main Success Scenario:**

1. Student navigates to the profile section.
2. Student views or updates password.
3. System saves the changes.

**Use Case 3: Student Registration**

**Name:** Register New Students  
**Primary Actor:** Admin  
**Scope:** Student Management System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Admin:** Wants to register new students into the system.

**Precondition:**

* The admin is logged into the system.

**Success Guarantee:**

* The new student is registered successfully.

**Main Success Scenario:**

1. Admin navigates to the student registration page.
2. Admin enters the student’s details (e.g., name, email, etc.).
3. System saves the student's information and creates a new student record.

**Use Case 4: Course Enrollment**

**Name:** Enroll in Courses  
**Primary Actor:** Student  
**Scope:** Course Management System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Student:** Wants to enroll in courses for the current semester.
* **System:** Ensures that students are enrolled in active courses.

**Precondition:**

* The student is logged into the system.

**Success Guarantee:**

* The student is successfully enrolled in the selected course.

**Main Success Scenario:**

1. Student navigates to the course enrollment section.
2. Student selects an available course.
3. System enrolls the student in the course.

**Use Case 5: Class Schedule Management**

**Name:** Manage/View Class Schedules  
**Primary Actor:** Admin (for management), Student (for viewing)  
**Scope:** Class Scheduling System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Admin:** Wants to create or update the class schedules.
* **Student:** Wants to view the class schedule for enrolled courses.

**Precondition:**

* Admin or student is logged into the system.

**Success Guarantee:**

* Class schedules are created or updated (Admin).
* Students can view their class schedules.

**Main Success Scenario (Admin):**

1. Admin navigates to the class schedule management page.
2. Admin adds or updates the schedule (day, time, room, etc.).
3. System saves the changes.

**Main Success Scenario (Student):**

1. Student navigates to the class schedule page.
2. System displays the schedules for the courses the student is enrolled in.

**Use Case 6: Exam Schedule Management**

**Name:** Manage/View Exam Schedules  
**Primary Actor:** Admin (for management), Student (for viewing)  
**Scope:** Exam Scheduling System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Admin:** Wants to create or update exam schedules.
* **Student:** Wants to view exam schedules.

**Precondition:**

* Admin or student is logged into the system.

**Success Guarantee:**

* Exam schedules are created or updated (Admin).
* Students can view exam schedules.

**Main Success Scenario (Admin):**

1. Admin navigates to the exam schedule management page.
2. Admin adds or updates the exam schedule (course, date, time, room, etc.).
3. System saves the changes.

**Main Success Scenario (Student):**

1. Student navigates to the exam schedule page.
2. System displays the exam schedules for all courses.

**Use Case 7: Leave Management**

**Name:** Request/Manage Faculty Leaves  
**Primary Actor:** Faculty (for requesting), Admin (for management)  
**Scope:** Leave Management System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Faculty:** Wants to request leave for a specific period.
* **Admin:** Wants to approve or disapprove leave requests.

**Precondition:**

* Faculty or admin is logged into the system.

**Success Guarantee:**

* Leave requests are created successfully (Faculty).
* Admin successfully approves or disapproves leave requests.

**Main Success Scenario (Faculty):**

1. Faculty navigates to the leave request page.
2. Faculty enters leave details (reason, start date, duration).
3. System saves the leave request with a "Pending" status.

**Main Success Scenario (Admin):**

1. Admin navigates to the leave management page.
2. Admin views all pending leave requests.
3. Admin approves or disapproves a selected leave request.
4. System updates the leave status.

**Use Case 8: Attendance Management**

**Name:** Mark/View Attendance  
**Primary Actor:** Faculty (for marking), Student (for viewing)  
**Scope:** Attendance Management System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Faculty:** Wants to mark attendance for a selected course.
* **Student:** Wants to view their attendance records.

**Precondition:**

* Faculty or student is logged into the system.

**Success Guarantee:**

* Attendance is marked successfully (Faculty).
* Students can view their attendance records.

**Main Success Scenario (Faculty):**

1. Faculty navigates to the attendance marking page.
2. Faculty selects a course and marks students as present, absent, or late.
3. System saves the attendance record.

**Main Success Scenario (Student):**

1. Student navigates to the attendance page.
2. System displays the attendance records for the courses the student is enrolled in.

**Use Case 9: Marks Management**

**Name:** Add/Update/View Marks  
**Primary Actor:** Faculty (for adding/updating), Student (for viewing)  
**Scope:** Marks Management System  
**Level:** User Goal  
**Stakeholders and Interests:**

* **Faculty:** Wants to add or update marks for assessments.
* **Student:** Wants to view marks for assessments.

**Precondition:**

* Faculty or student is logged into the system.

**Success Guarantee:**

* Marks are added or updated successfully (Faculty).
* Students can view their marks.

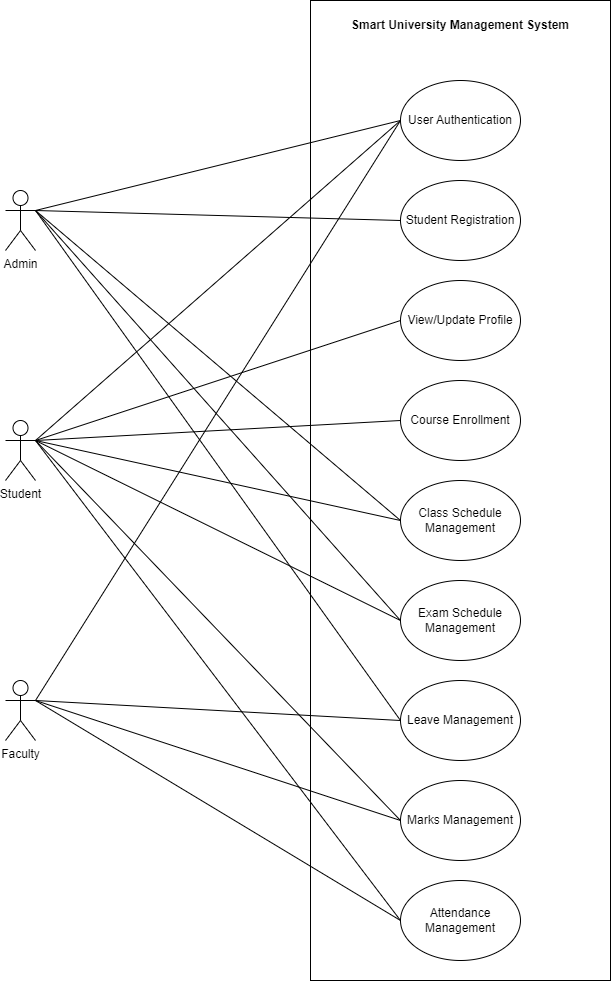
**Main Success Scenario (Faculty):**

1. Faculty navigates to the marks management page.
2. Faculty enters or updates marks for a student (course, assessment type, marks).
3. System saves the marks.

**Main Success Scenario (Student):**

1. Student navigates to the marks page.
2. System displays marks for the courses the student is enrolled in.

## Use Case Diagram



# Other Nonfunctional Requirements

## Performance Requirements

1. **System Responsiveness**:
   * The application should initialize the dashboard and load all user-specific data (courses, schedules, attendance) within **2 seconds** of login.
   * All interactions, including data fetch and updates (e.g., marks submission, attendance marking), should complete within **1 second**.
2. **Database Performance**:
   * SQL queries should execute in **500 milliseconds** for operations like fetching enrolled courses, updating marks, or retrieving attendance records.
   * The system must handle a **database size of up to 50,000 records** across tables (students, courses, marks, attendance) without significant latency.
3. **Concurrency**:
   * The system should support up to **10 concurrent sessions** (e.g., multiple users accessing or modifying data simultaneously via a shared database).
4. **Data Updates**:
   * Real-time operations like attendance marking, marks updates, or course enrollment should reflect in the database immediately (<500ms latency).

## Safety Requirements

1. **Data Integrity**:
   * All data entered by users (e.g., marks, attendance, enrollment status) must be validated before being saved to ensure no invalid or incomplete data is stored.
   * Use **transaction-based database operations** to ensure atomicity during critical updates.
2. **Crash Recovery**:
   * If the application or system crashes, users should be able to recover the last session with minimal data loss.
   * Scheduled database backups must be enabled, with admins configuring backup frequency (e.g., daily or weekly).
3. **Error Handling**:
   * Critical errors (e.g., database disconnections, invalid inputs) must be handled gracefully, with user-friendly error messages guiding corrective actions.
   * The system should log errors for administrator review to assist in troubleshooting.
4. **Data Redundancy**:
   * The system should ensure redundancy through regular backups, reducing the risk of data loss due to system failures.

## Security Requirements

1. **User Authentication**:
   * The application must enforce **role-based authentication** to restrict access:
     + **Admin**: Full access to all features.
     + **Faculty**: Restricted to managing classes, attendance, and leave requests.
     + **Students**: Access limited to personal data, marks, schedules, and attendance.
   * Passwords must be stored using a secure hashing algorithm such as **BCrypt**.
2. **Access Control**:
   * **SessionManager** must ensure only logged-in users with valid credentials can access their respective dashboards.
   * Session data (e.g., user type, email, name) must be securely maintained during the session.
3. **Data Privacy**:
   * Sensitive data, including passwords, CNICs, and contact details, must not be accessible to unauthorized users or appear in logs.
4. **SQL Injection Prevention**:
   * Use **prepared statements** in all database queries to safeguard against injection attacks (as seen in DAOs).
5. **Network Security**:
   * If used in a networked environment, communications between the JavaFX application and the MySQL database should use **SSL/TLS encryption**.

## Software Quality Attributes

1. **Usability**:
   * The user interface, built with **JavaFX Scene Builder**, must provide an intuitive layout for all user roles.
   * Error messages and navigation must be simple and clear, ensuring ease of use for students, faculty, and admins.
2. **Reliability**:
   * The system should maintain an uptime of **99.9%** during academic sessions, ensuring consistent availability.
   * All user actions (e.g., attendance marking, leave requests) must be saved securely without any transaction loss.
3. **Maintainability**:
   * Code modularity (with distinct DAO, service, and controller layers) must facilitate easy updates or addition of new features.
   * All methods, classes, and services should be documented for future maintenance.
4. **Scalability**:
   * The MySQL database should handle expanding datasets (e.g., increasing student, course, and attendance records) without significant performance degradation.
5. **Portability**:
   * The application must be deployable as a standalone JAR or EXE file, ensuring compatibility across **Windows**, **macOS**, and **Linux** systems with Java Runtime Environment (JRE).

## Business Rules

1. **Student Course Enrollment**:
   * A student can enroll in a maximum of **5 active courses** per semester.
   * Students must fulfill prerequisites before enrolling in advanced courses.
2. **Attendance Management**:
   * Faculty must mark attendance for each class session within **24 hours**.
   * Students with less than **75% attendance** are automatically disqualified from exams for the corresponding course.
3. **Marks Management**:
   * Faculty must submit marks within **7 days** after an exam or assignment.
   * Modifications to marks require explicit admin approval.
4. **Leave Requests**:
   * Faculty can apply for up to **30 days of leave per academic year**.
   * All leave requests must be approved or rejected by the admin within **3 business days**.
5. **Exam Scheduling**:
   * Exams must not overlap for students enrolled in multiple courses.
   * Rooms must be checked for conflicts before scheduling exams.

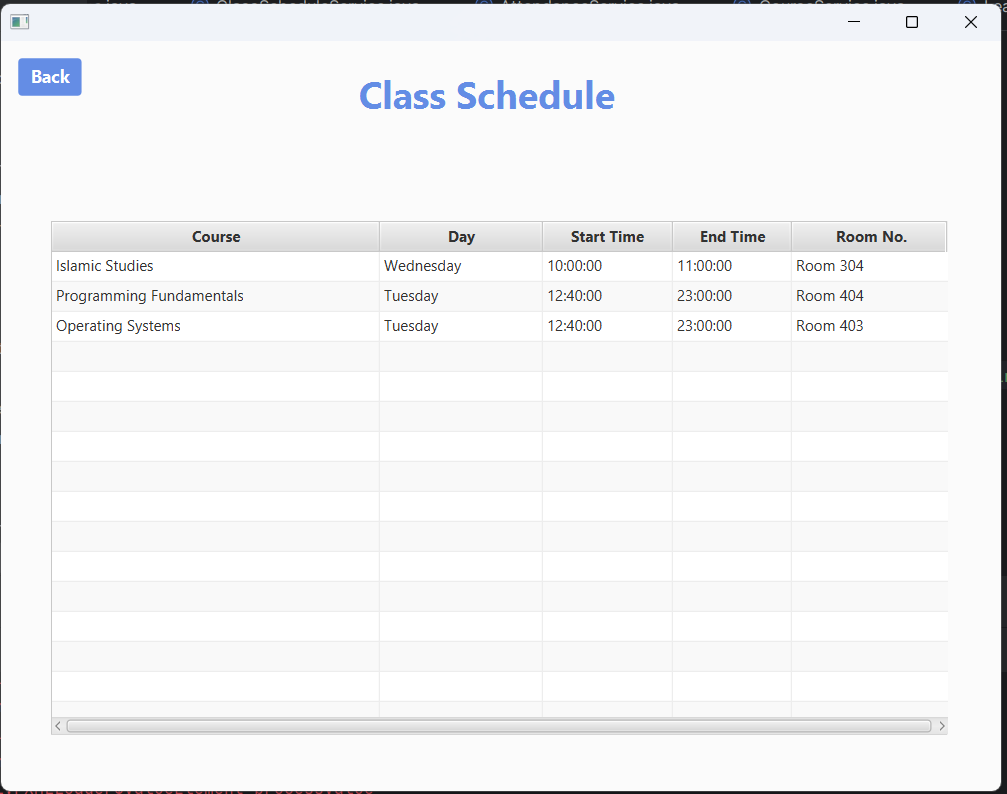
## Operating Environment

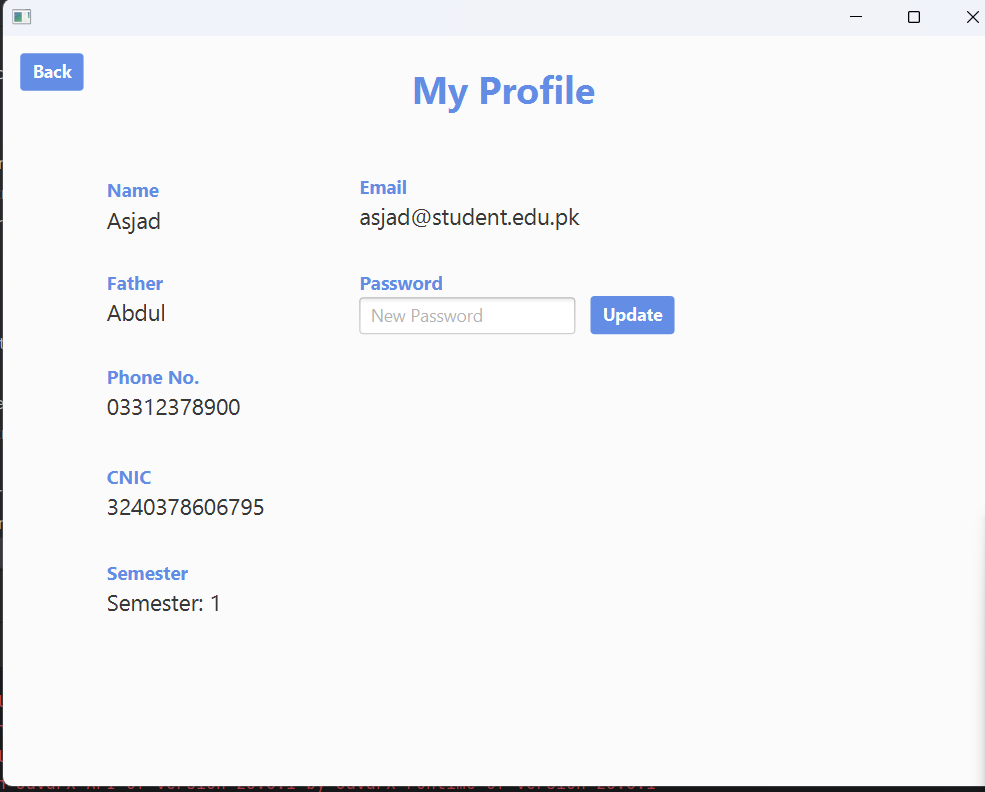
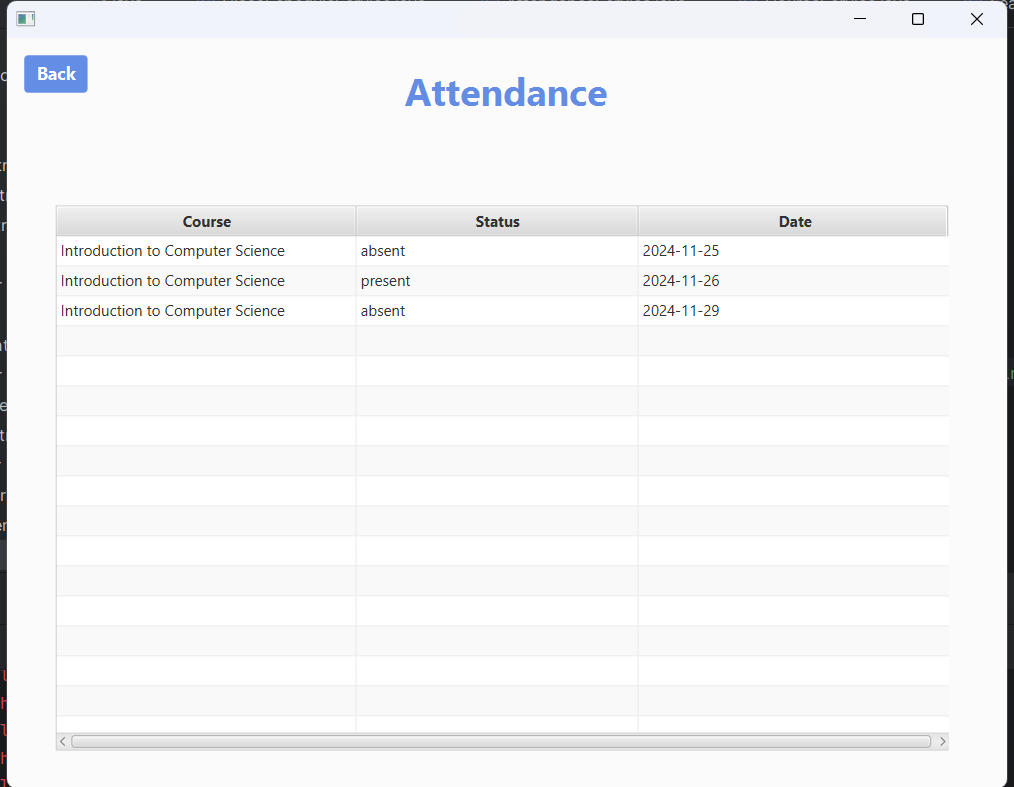
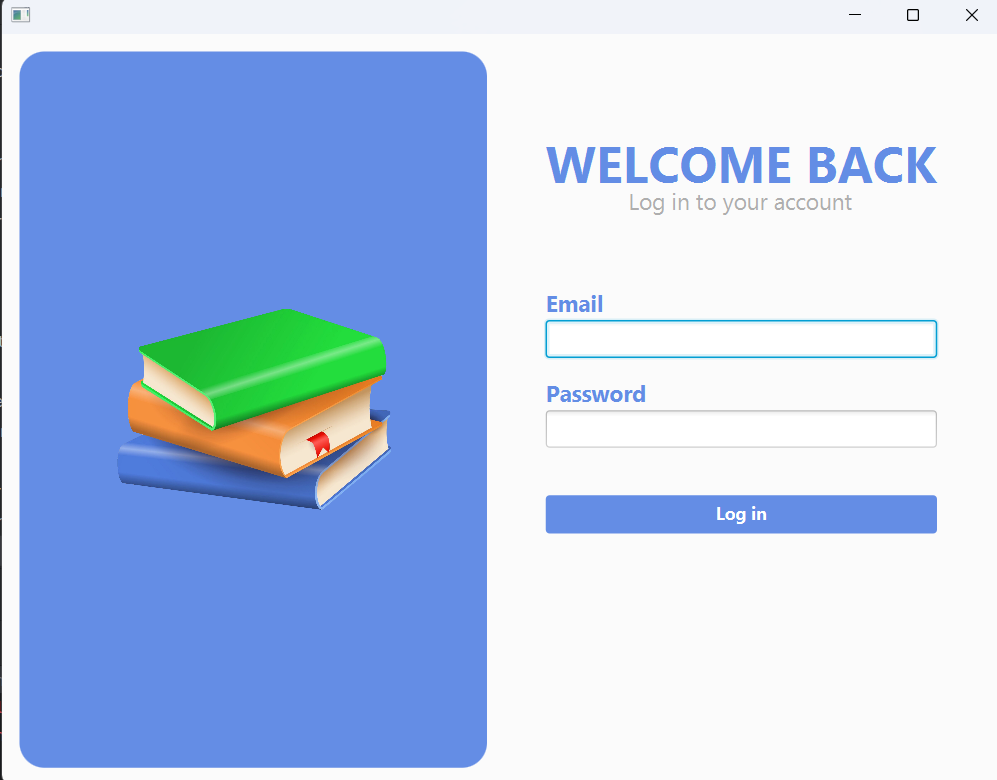
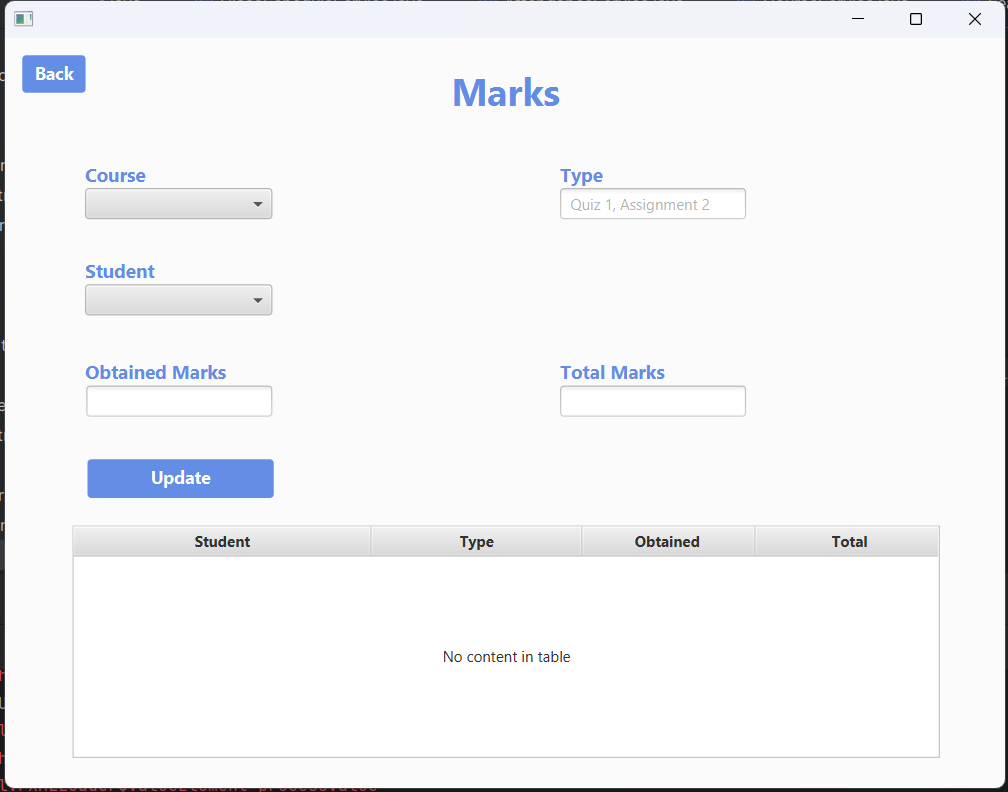
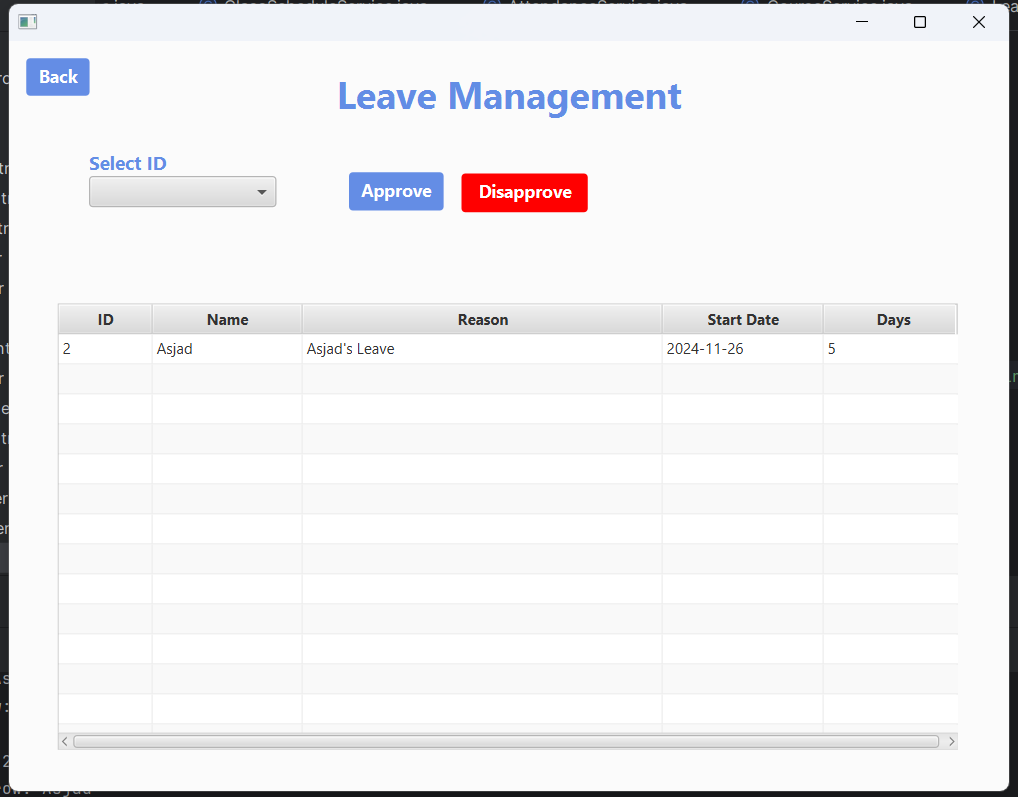
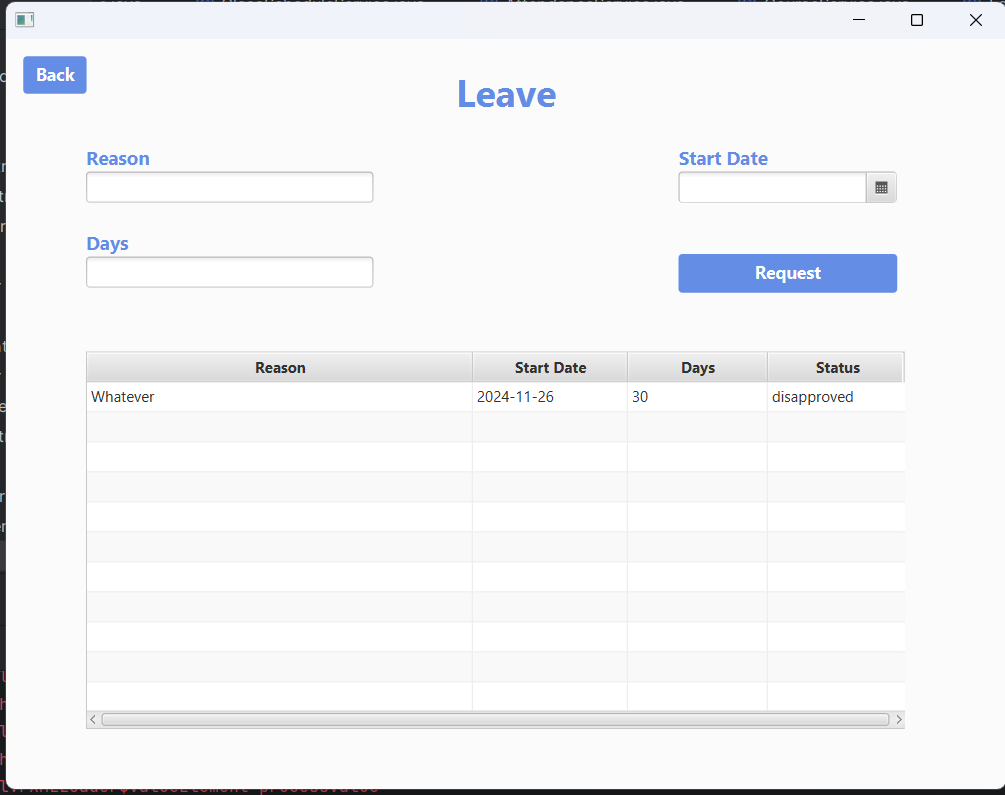
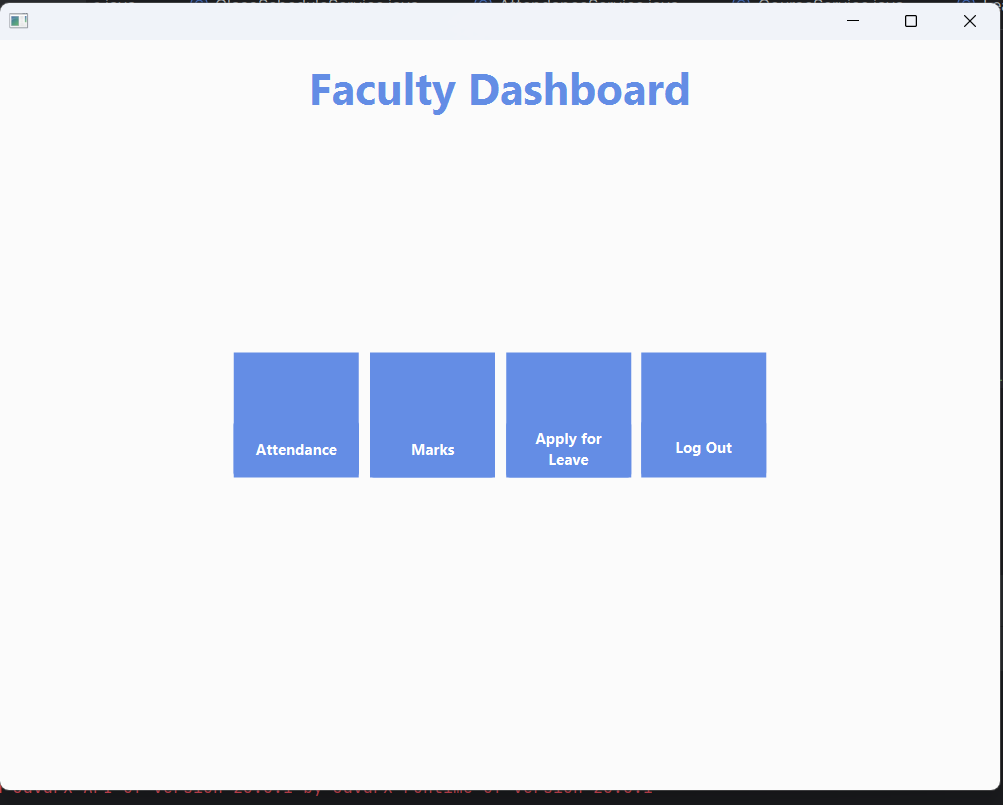
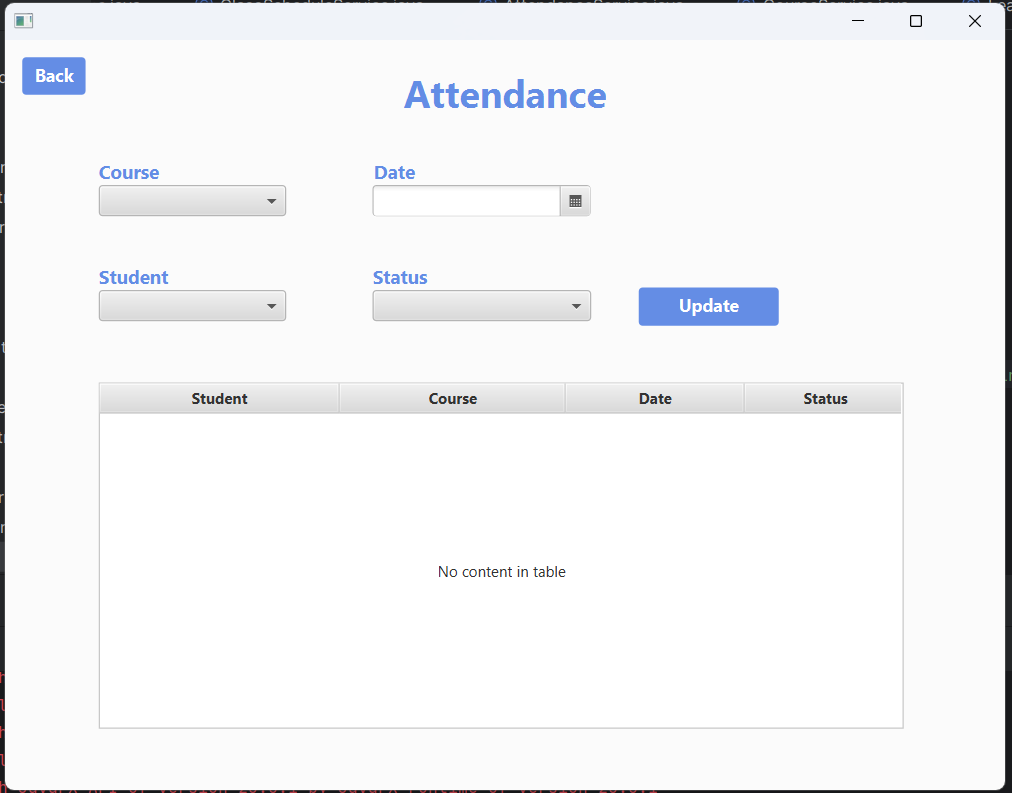
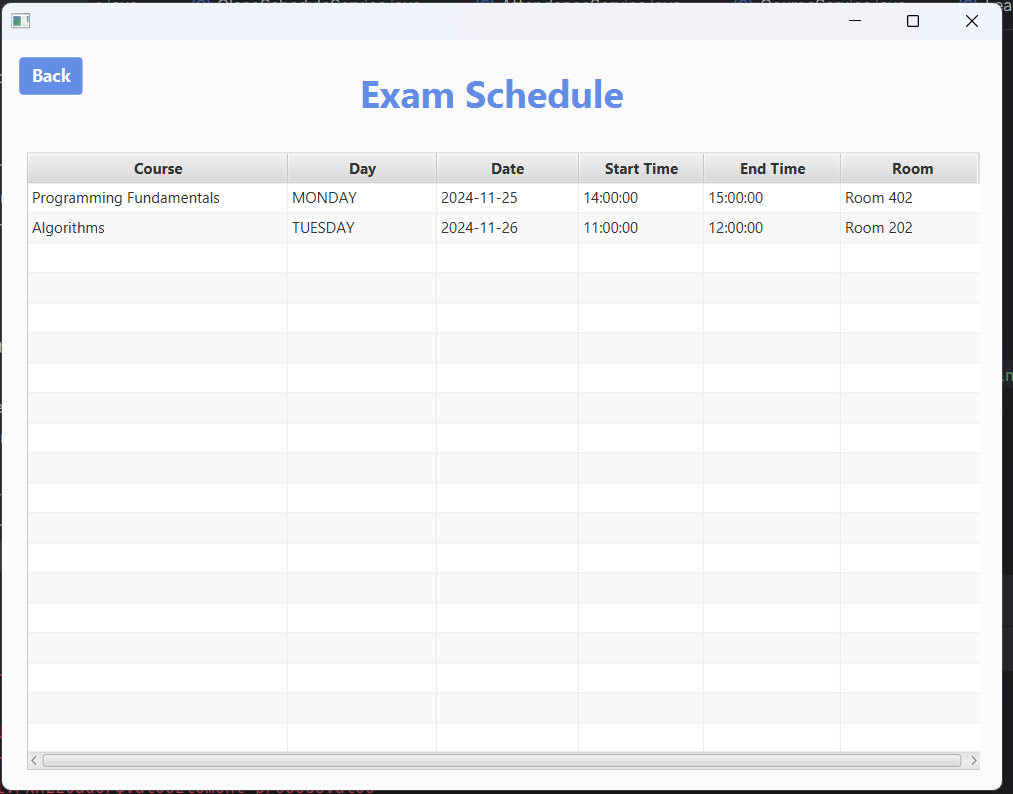
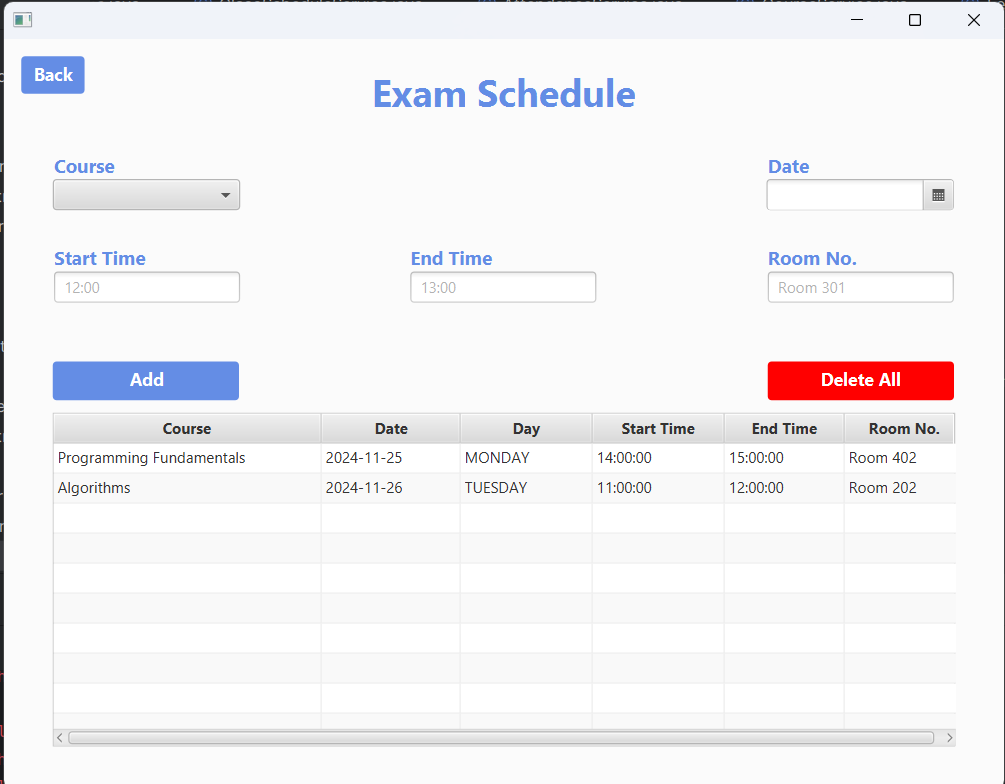
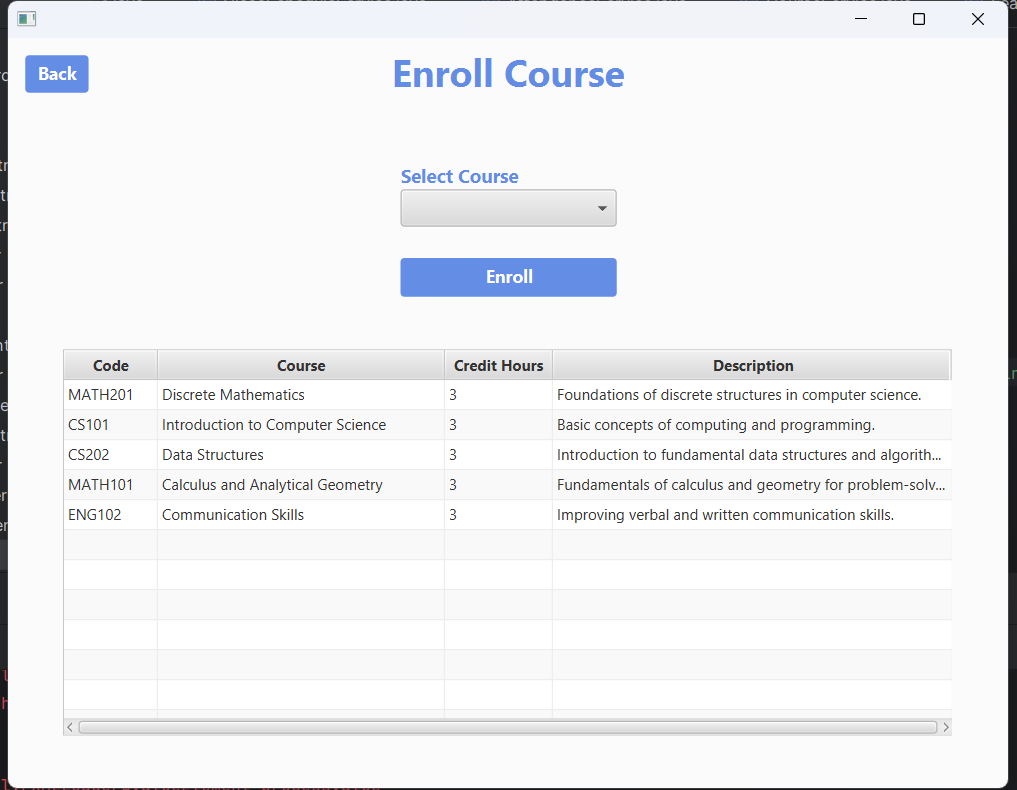
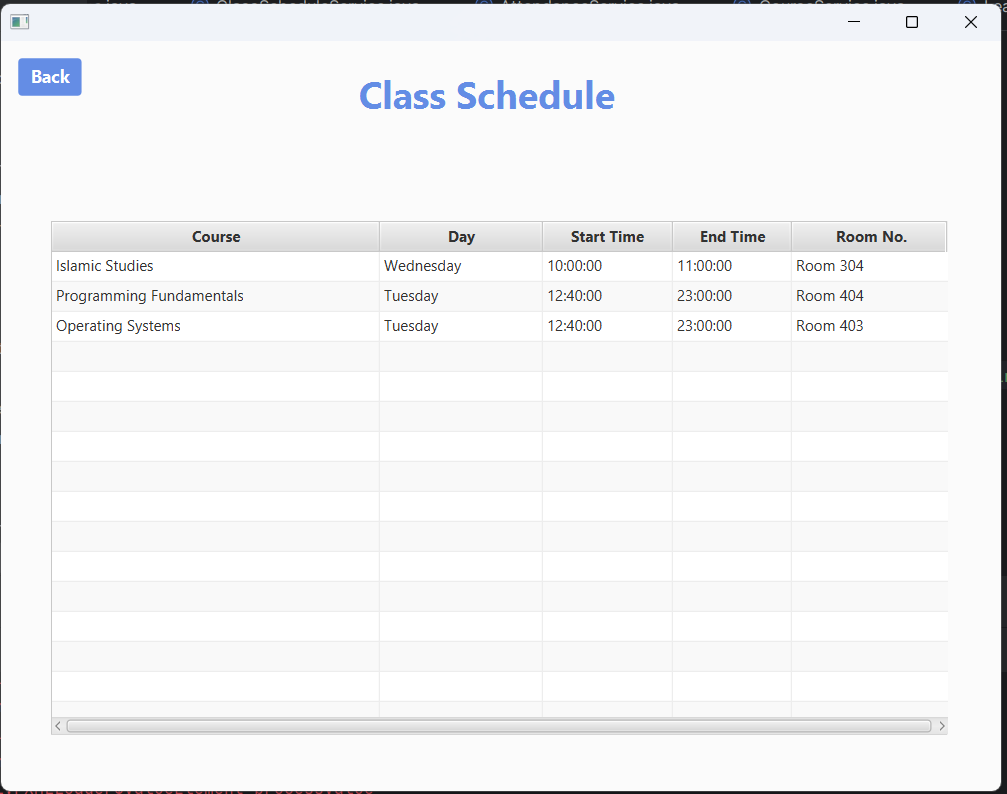
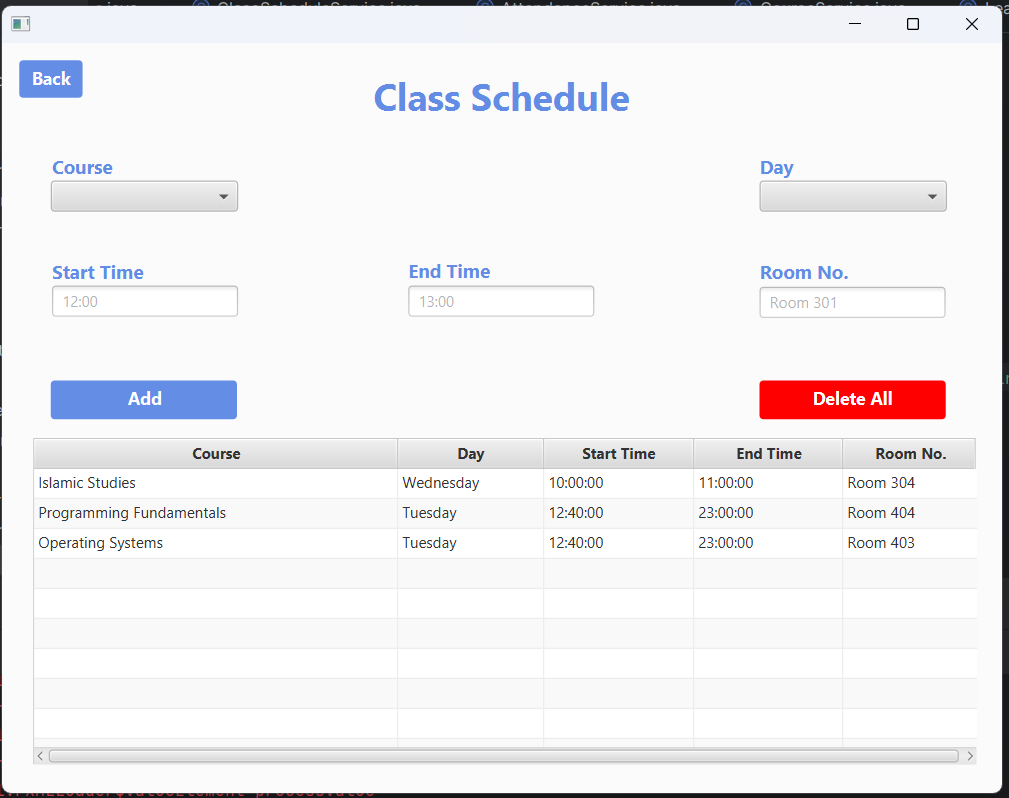
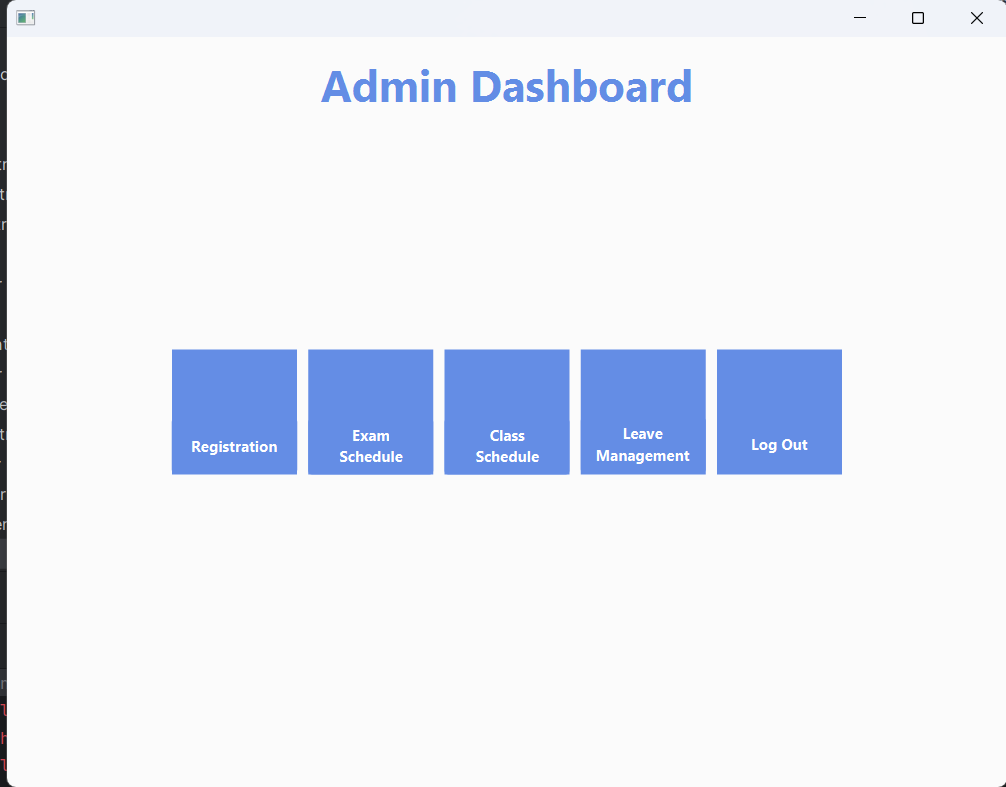
1. **Hardware Requirements**:
   * Processor: Minimum **2 GHz dual-core CPU**.
   * Memory: **4 GB RAM** or higher.
   * Storage: At least **500 MB** for application and MySQL database files.
2. **Software Requirements**:
   * **Operating System**: Windows 10/11, macOS, or Linux with JRE installed.
   * **Java Version**: JDK 11 or higher for development and execution.
   * **Database**: MySQL 8.0 or higher for managing backend operations.
3. **Development Tools**:
   * **IDE**: IntelliJ IDEA, Eclipse, or NetBeans.
   * **UI Development**: JavaFX Scene Builder.
4. **Deployment**:
   * The application must be packaged as a **standalone desktop executable** (JAR or EXE) with all dependencies bundled.
   * The MySQL database should support both **local deployments** and **LAN-based multi-user setups**.

## User Interfaces

The **User Interfaces** of the Smart University Management System (SUMS) are designed to be intuitive, user-friendly, and role-specific. The interface is developed to cater to three primary user roles: **Admin**, **Faculty**, and **Student**. Each user interface is tailored to provide the necessary functionality based on the user’s role while maintaining simplicity and ease of navigation.

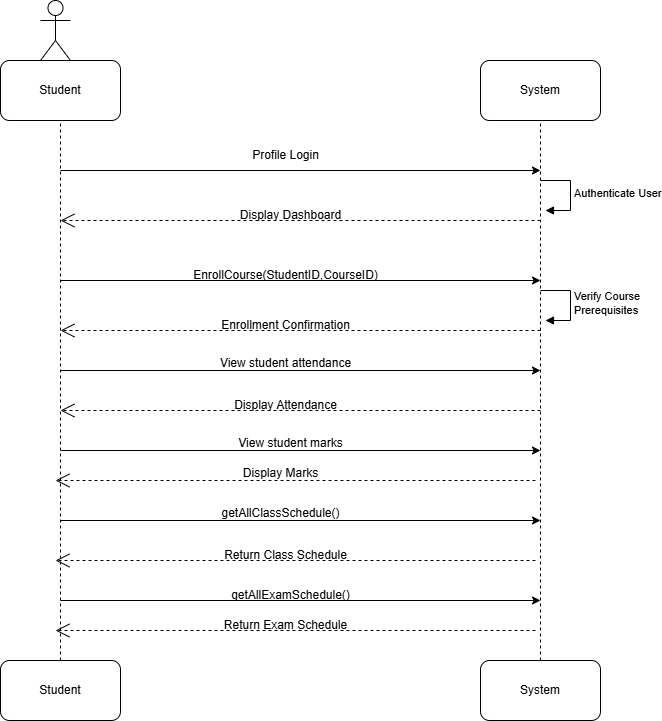
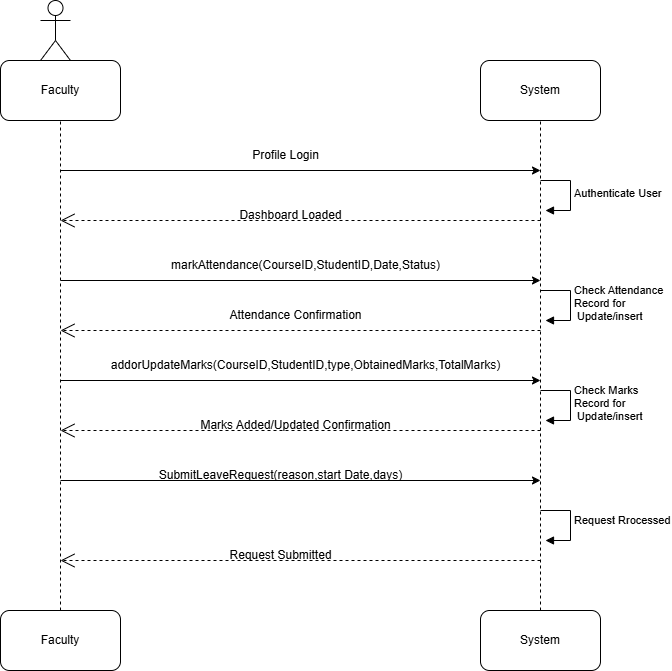
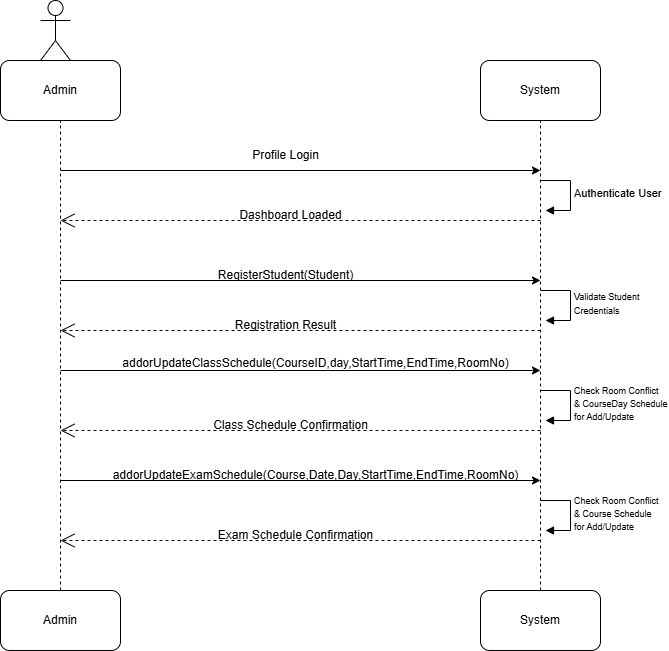
Key characteristics of the user interfaces include:

1. **Consistency**
   * The interface design follows a consistent layout and theme across all modules.
   * Buttons, menus, and tables are styled uniformly to provide a cohesive experience.
2. **Accessibility**
   * The system is designed to be accessible to users with minimal training.
   * Clear labels and tooltips are provided to guide users through complex actions.
3. **Role-Based Interfaces**
   * **Admin Interface**: Provides access to manage schedules, leave approvals, registrations, and overall system administration.
   * **Faculty Interface**: Allows faculty members to manage attendance, marks, and leave requests.
   * **Student Interface**: Displays attendance records, marks, class schedules, and exam schedules.
4. **Dynamic Dropdowns and Tables**
   * Dropdown menus dynamically populate based on context (e.g., pending leave requests, course lists).
   * Tables are used for data visualization, with sortable columns for ease of use.
5. **Responsive Feedback**
   * The system provides immediate feedback through alert dialogs for actions such as updates, errors, and confirmations.
6. **Error Prevention and Handling**
   * Forms validate user inputs (e.g., dates, numeric fields) before submission.
   * Error messages are displayed in a clear and non-technical manner to inform users of issues.
7. **Navigation and Accessibility**
   * Simple navigation menus and back buttons allow users to move easily between screens.
   * Each interface has a dashboard view for a quick overview of the user's role-specific functionalities.
8. **Compatibility**
   * The interface is optimized for use on desktop system The overall goal of the user interfaces is to ensure that all stakeholders—students, faculty, and administrators—can interact with the system effectively, reducing the learning curve and enhancing productivity.  
     



# Domain Model

# System Sequence Diagram



**.**

**6.Sequence Diagram**

# 

.

# 

# Class Diagram

# 

# Package Diagram

# 

# Deployment Diagram

# 