NORTH SOUTH UNIVERSITY

Department of Electrical and Computer Engineering

Project Report

On

Class Schedule Management System

COURSE NAME: Software Engineering

COURSE CODE: CSE327

SECTION: 10

SUBMITTED BY:

Name	ID
Susmita Goswani	2221409042
Nafis Karim	2221342642
Zinat Imteaz	2222004642

SUBMITTED TO: Dr. Mohammad Rezwanul Huq

Drive link:https://docs.google.com/document/d/1Ih-bSDmviKCUUTCB7qKOA_xRgRugXGtnNGDN9tZNW3k/edit?tab=t.0#heading=h.lhbmxoehpkdk

Abstract

The Class Schedule Management System (CSMS) is a web-based administrative application designed to support universities in creating and managing academic class schedules in a structured, efficient, and conflict-free manner. The primary objective of the system is to automate the scheduling process by integrating course information, sections, instructors, rooms, and predefined timeslots while strictly enforcing institutional scheduling constraints.The system enables authorized administrators to manage courses, instructors, rooms, and timeslots through a secure login mechanism and an intuitive user interface. CSMS incorporates automated conflict detection mechanisms to prevent common scheduling issues such as instructor double-booking, room double-booking, and mismatches between course types and room types (THEORY or LAB). Predefined, non-overlapping timeslot patterns ensure consistency, repeatability, and reliability across the scheduling process.Developed using a modular architecture, the system utilizes HTML, Tailwind CSS, and JavaScript on the frontend, PHP on the backend, and MySQL for persistent data storage. Emphasis is placed on usability, performance, security, and maintainability, ensuring fast response times, secure data handling, and ease of future enhancement. By providing real-time validation, comprehensive search and review features, and clear conflict feedback, the CSMS significantly reduces manual effort and human error in academic scheduling. Overall, the system offers a reliable and scalable solution for effective class schedule management within a university environment.

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Class Schedule Management System (CSMS), designed to assist university administrators in creating conflict-free academic class schedules each semester. The system intends to handle courses, sections, instructors, rooms, and timeslot constraints to ensure an efficient scheduling process.

1.2 Scope

The Course Scheduling Management System (CSMS) helps administrators easily manage courses, sections, instructors, rooms, and timeslots, all while making sure everything follows the set rules. Some of its main features are logging in securely, creating and editing schedules, detecting conflicts (like when rooms or instructors are double-booked), and searching through the schedules. Before saving any schedule, the system will check for issues like room assignments, instructor availability, and timeslot conflicts. The system is designed to be fast, reliable, and secure, with easy-to-understand error messages. It also ensures the system stays organized and can grow as needed, and it keeps track of all actions for safety and future reference.

1.3 Definitions and Acronyms

Definitions

- Timeslot: A named code representing a repeating day pattern with a fixed start and end time (e.g., ST1 = Sunday & Tuesday, 08:00–09:30).
- Timeslot Pattern: The day pair or pattern used by the timeslot (e.g., ST = Sun/Tue, MW = Mon/Wed, RA = Thu/Fri).
- Schedule Entry: A single scheduled class instance defined by (Course, Section, Instructor, Timeslot, Room).
- Theory Room: Classroom suited for lectures; indicated by room type THEORY.
- Lab Room: Room with laboratory facilities; indicated by room type LAB.
- Conflict: Any violation of scheduling constraints (e.g., instructor double-booking, room double-booking, lab mismatch).

Acronyms

- CSMS- Class Schedule Management System
- SRS- Software Requirements Specification
- UI- User Interface
- DBMS- Database Management System
- API- Application Programming Interface
- FR- Functional Requirement
- NFR- Non-Functional Requirement

1.4 Overview

This document outlines:

- High-level product environment.
- Functional and non-functional requirements.
- System constraints and assumptions.
- User characteristics.

- External interface expectations.
- Essential data structures.
- Scheduling and timeslot rules.

1.5 Technological Infrastructure

It is the set of technologies stacked together to build any application. In this section, we will discuss the technologies used to develop the website.

1.5.1 Frontend:

(i) HTML5 (Semantic markup)

(ii) Tailwind CSS 3 (Responsive utility-first CSS)

(iii) Vanilla JavaScript (Form interactions)

(iv) Responsive Design (Mobile, Tablet, Desktop)

1.5.2 Backend:

(i) PHP 7.4+ (Server-side logic)

(ii) MySQLi (Database connectivity)

(iii) Session Management (Authentication)

1.5.3 Database layer:

(i) MySQL 5.7+ (Data persistence)

(ii) 7 Interconnected Tables

(iii) Referential Integrity & Constraints

1.5.4 Server & Build

(i) Apache (Web server via XAMPP)

(ii) Node.js (Build tool)

(iii) Tailwind CLI (CSS compilation)

2. Summary of SRS:

Overall Description

- **Product Perspective:** Web application with backend DBMS storing courses, instructors, rooms, and schedule entries. Scheduling rules engine ensures constraint enforcement.
- **Functions:**
  - Catalog, Resource, Timeslot, Schedule, Validation, Search, and User Access Management.
- **User Characteristics:** Administrator with scheduling knowledge, capable of navigating UI efficiently.
- **Constraints:** Predefined timeslots, room types, instructor and room mutual exclusivity, access control, concurrency management, localhost deployment.
- **Assumptions & Dependencies:** Accurate instructor availability and room inventory; database and authentication services; optional third-party UI libraries.

2.1 System Requirements

Functional Requirements (FR)

- Authentication and secure login.
- Course, section, instructor, and room management.
- Timeslot management with predefined patterns.
- Schedule creation, editing, and viewing.
- Constraint validation (room type, instructor/time conflicts, room availability).
- Schedule search and review functionality.

Non-Functional Requirements (NFR)

- **Performance:** Fast loading and schedule validation (<3s and <1s).
- **Reliability:** 95% availability, daily backups, clear error handling.
- **Security:** Encrypted passwords, role-based access, data protection.

- **Usability:** Intuitive UI, easy navigation, beginner-friendly.
- **Maintainability:** Modular architecture, coding standards, easy updates.
- **Compatibility:** Works on major browsers and desktops.
- **Data & Audit:** Consistent data, validated input, logs all admin actions.

Data Descriptions

- Key entities: Department, Administrator, Timeslot, Course, Instructor, Room, Schedule Entry.
- PKs and FKs defined for relationships.

Timeslot Rules & Constraints

- Fixed timeslot patterns with 10-minute breaks.
- Constraints: Instructor availability, room availability, room type matching, unique section entries, complete schedule entries, no duplicates.
- Conflict detection shows errors and allows draft saving for later resolution.

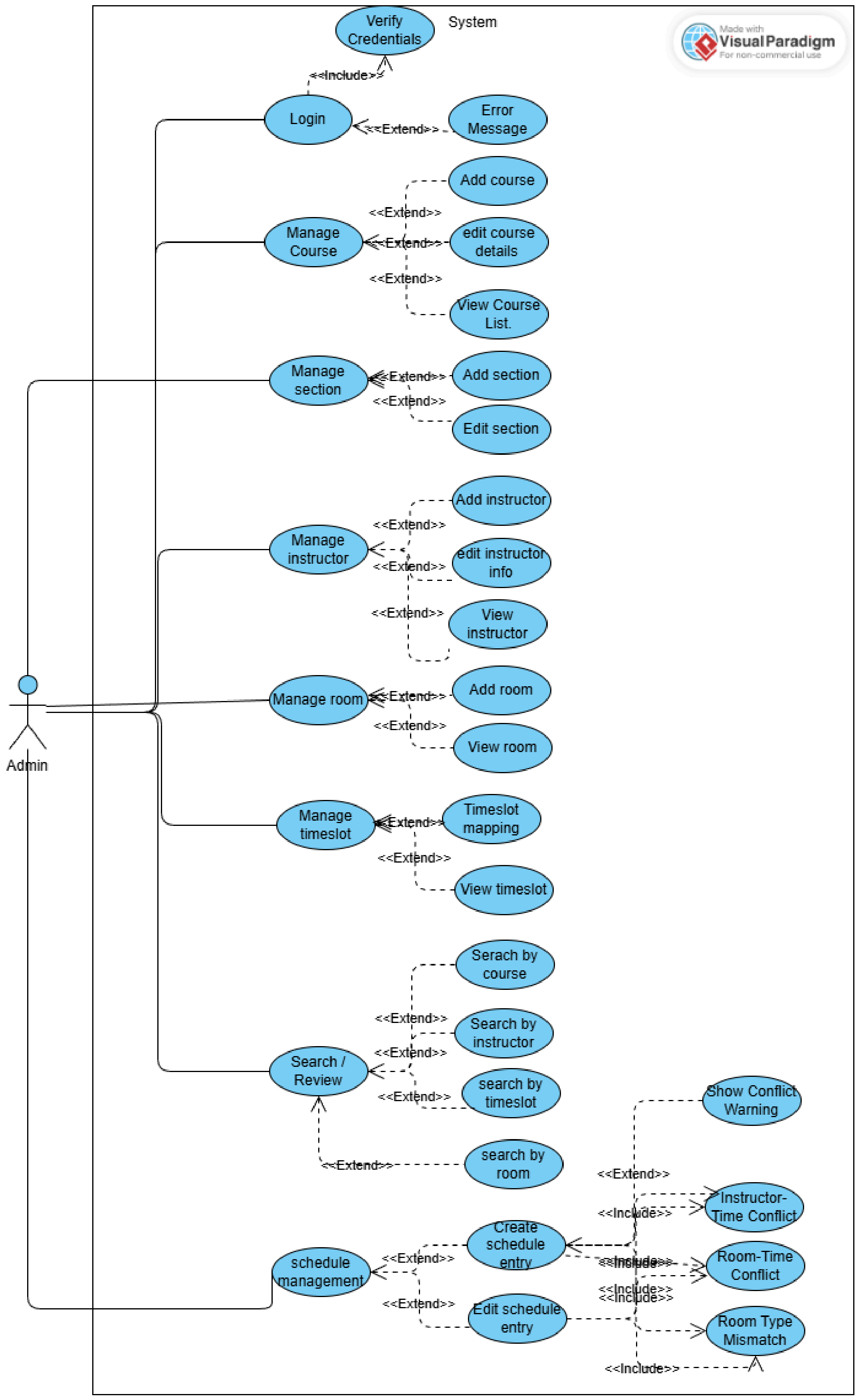
2.2 External Interface

- **User Interface Expectations:** Login screen, Dashboard, Manage Courses/Instructors/Rooms, Schedule Management with conflict detection.
- **System Interaction Points:**
  - Student/public: view schedules, courses, instructors, rooms, system info.
  - Administrator: login, dashboard, manage resources, create/edit schedules, resolve conflicts, logout.
- Data flows: Courses, instructors, and rooms populate schedule creation; conflicts drafts managed separately; confirmed schedules appear in lists and student views.

3. System modeling summary

3.1: Use Case Diagram

This section has been covered by Susmita Goswami (2221409042)


System
Verify Credentials
<<Include>>
Login
<<Extend>>
Error Message
Add course
<<Extend>>
Manage Course
<<Extend>>
edit course details
<<Extend>>
View Course List.
Manage section
<<Extend>>
Add section
<<Extend>>
Edit section
Add instructor
<<Extend>>
Manage instructor
edit instructor info
<<Extend>>
<<Extend>>
View instructor
Manage room
<<Extend>>
Add room
<<Extend>>
View room
Admin
Manage timeslot
<<Extend>>
Timeslot mapping
<<Extend>>
View timeslot
Serach by course
<<Extend>>
Search by instructor
<<Extend>>
Search / Review
<<Extend>>
search by timeslot
<<Extend>>
search by room
Show Conflict Warning
<<Extend>>
<<Include>>
Instructor-Time Conflict
Create schedule entry
<<Extend>>
schedule management
<<Include>>
Room-Time Conflict
<<Include>>
<<Include>>
<<Extend>>
Edit schedule entry
Room Type Mismatch
<<Include>>

3.2: Use Case Descriptions

UC-1: Login to System (FR-1, FR-1.2, FR-1.1)

- Goal: Securely authenticate the Administrator and grant system access.
- Actor Action: Enters Username and Password.
- System Response:
  - Performs Verify Credentials (<<include>>).
  - If successful, grants access and displays the Dashboard (FR-1.1).
  - If unsuccessful, performs Show Login Error Message (<<extend>>) and displays the error notification (FR-1.2).

UC-2: Add Course (FR-2)

- Goal: Successfully define and save a new Course entity.
- Actor Action: Navigates to Manage Course and selects Add Course. Inputs required Course attributes (ID, Name, Type, Capacity). Selects 'Save'.
- System Response:
  - Displays the Course creation form.
  - Validates input data (NFR-19).
  - If valid, saves the new Course entity (FR-2) to the database.
  - Displays success confirmation.
  - If invalid, displays a detailed error message (NFR-6) and awaits correction.

UC-3: Create Schedule Entry (FR-14)

- Goal: Create and save a new, validated, conflict-free schedule entry.
- Actor Action: Selects Create Schedule Entry. Selects/Inputs Course, Section, Instructor, Timeslot, and Room. Submit the entry.
- System Response:
  - Performs Schedule Constraint Validation (<<include>>). (FR-21).
  - If valid, save the new Schedule Entry (FR-14).
  - If invalid (conflict detected), performs Show Conflict Warning (<<extend>>)(NFR-6), detailing the conflict, and requires adjustment.

UC-4: Schedule Constraint Validation (FR-17, FR-19, FR-20, FR-21)

- Goal: Verify a Schedule Entry against all defined conflict rules.
- Actor Action: N/A (Internal System Process).
- System Response:

    - Checks for Instructor-Time Conflict (FR-19).
    - Checks for Room-Time Conflict (FR-20).
    - Checks for Room Type Mismatch (FR-17, FR-18).
    - If all checks pass, return status: 'Valid'.
    - If any check fails, returns status: 'Conflict' and triggers Show Conflict Warning (<<extend>>) (NFR-6).

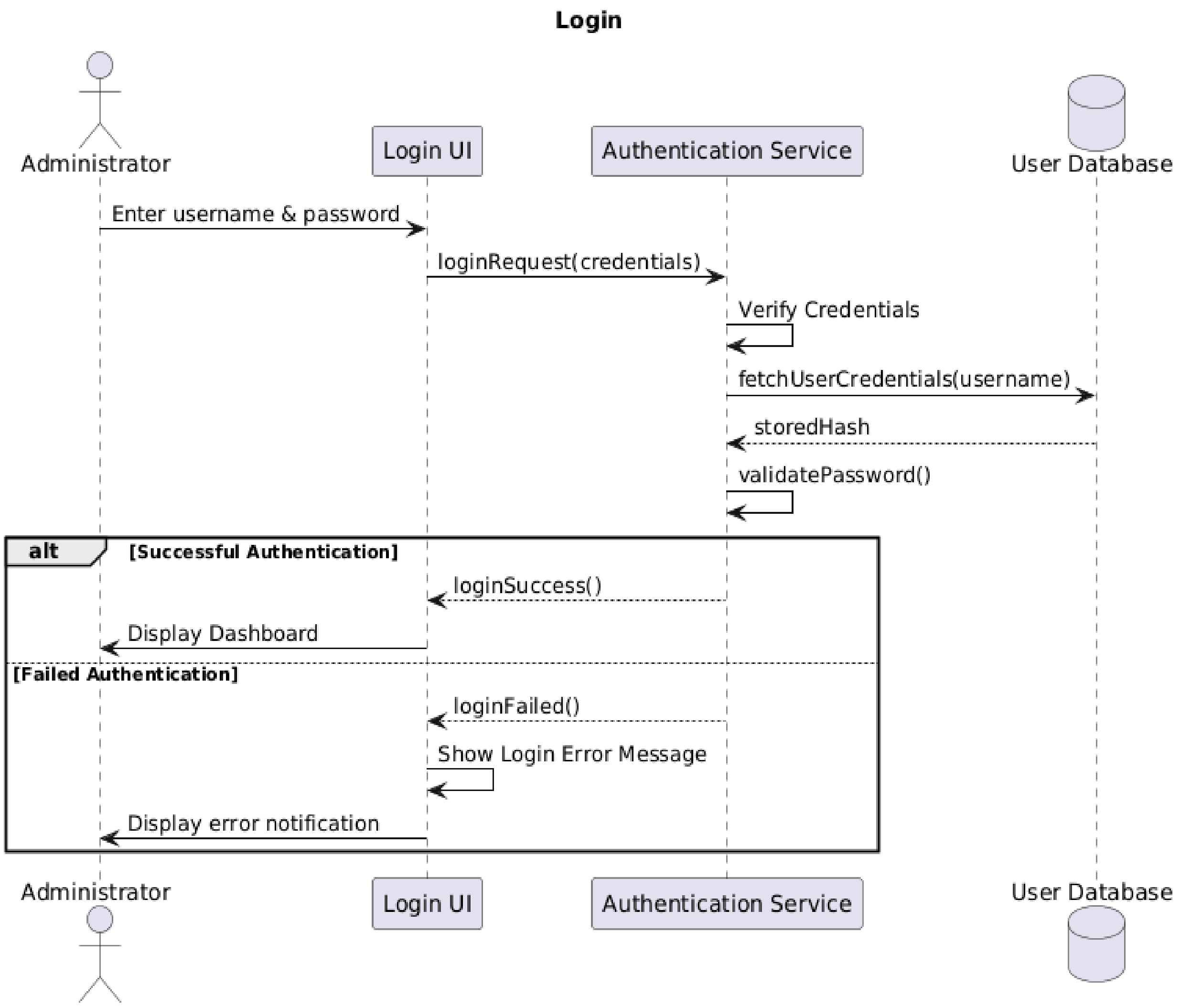
UC-5: Search Schedule (FR-23)

- Goal: Quickly locate and view relevant class schedules based on filtering criteria.
- Actor Action: Selects Search / Review. Selects a search method (e.g., Search by Instructor) and enters the query.
- System Response:
    - Executes the search query against schedule data.
    - Displays a filtered list of all matching Schedule Entries.
    - Loads the schedule list within the required response time (NFR-1.2).
    - If no results are found, display a "No entries found" message.

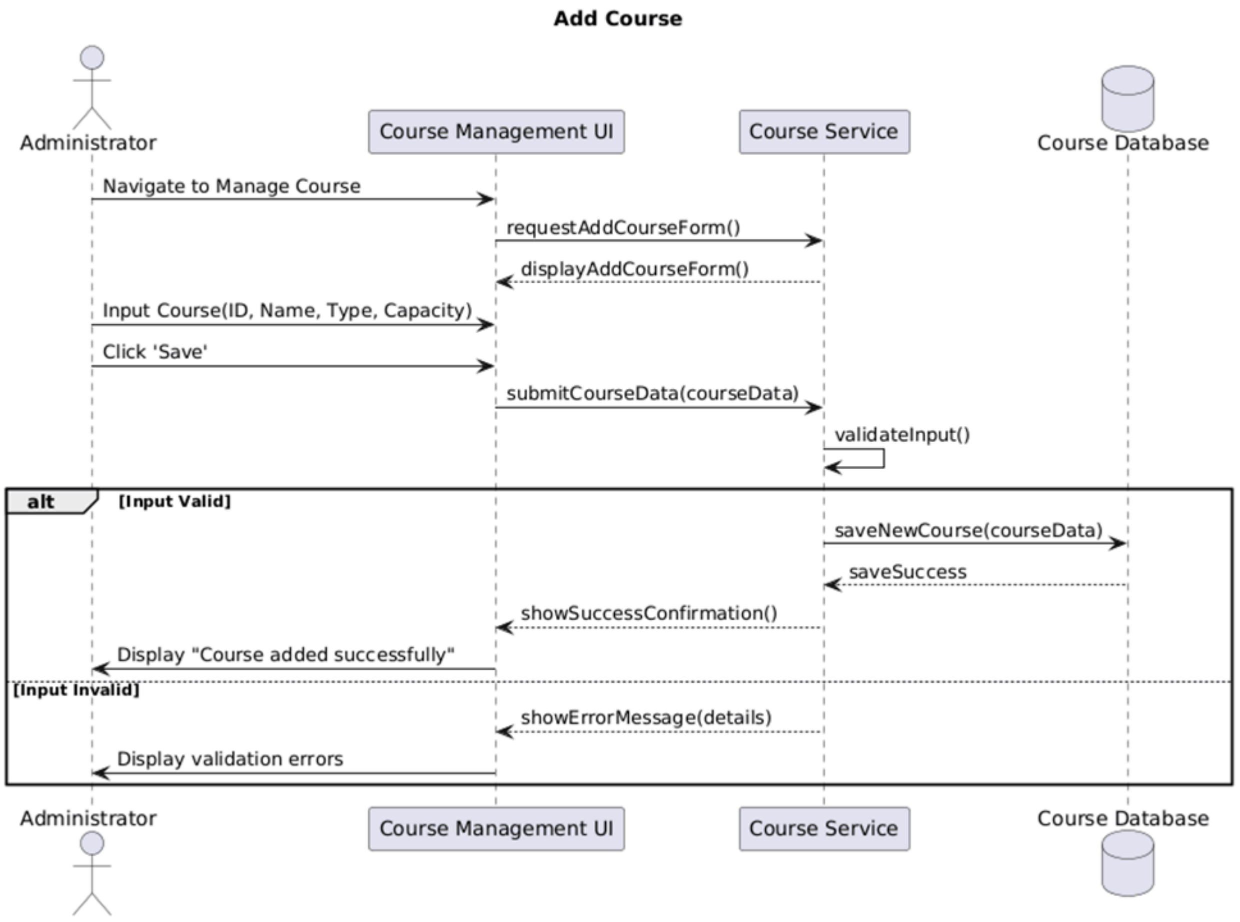
3.3: Sequence Diagrams

This section has been covered by Zinat Imteaz (2222004642)

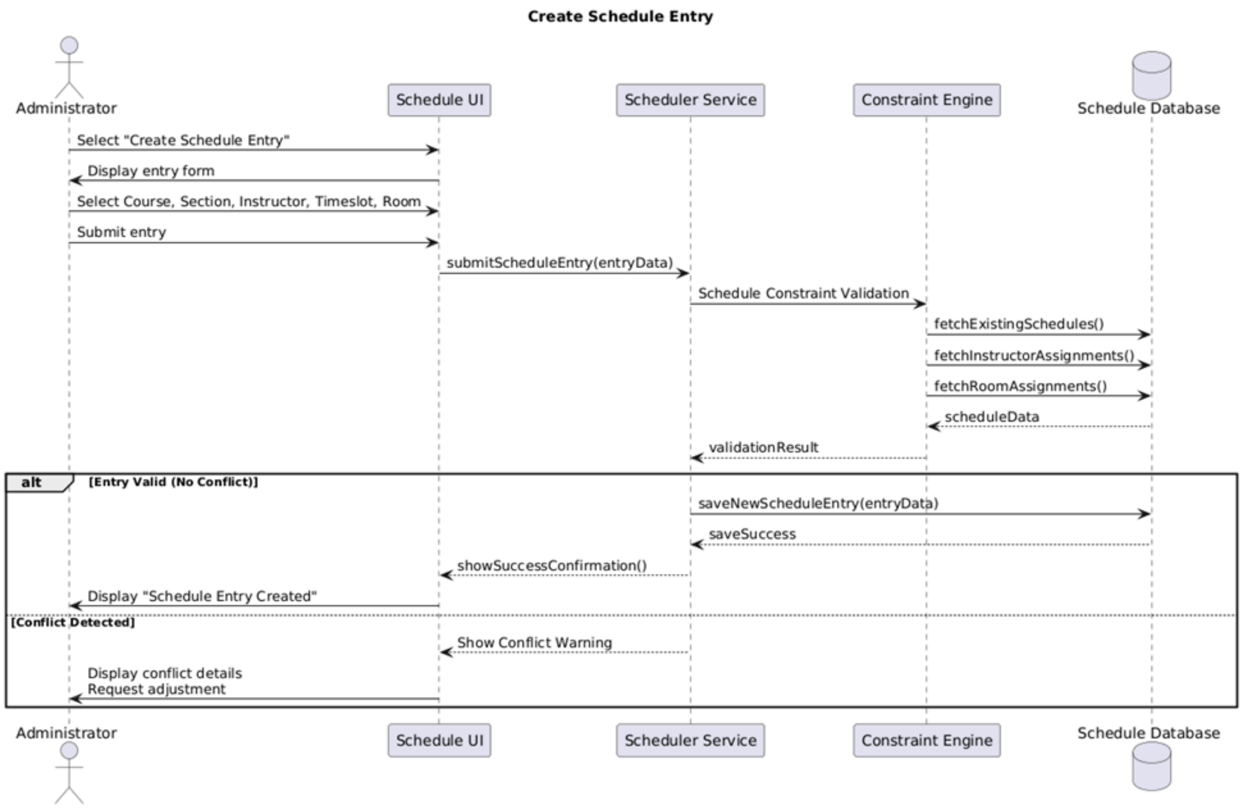
3.3.1:


Login
Administrator
Login UI
Authentication Service
User Database
Enter username & password
loginRequest(credentials)
Verify Credentials
fetchUserCredentials(username)
storedHash
validatePassword()
alt
[Successful Authentication]
loginSuccess()
Display Dashboard
[Failed Authentication]
loginFailed()
Show Login Error Message
Display error notification

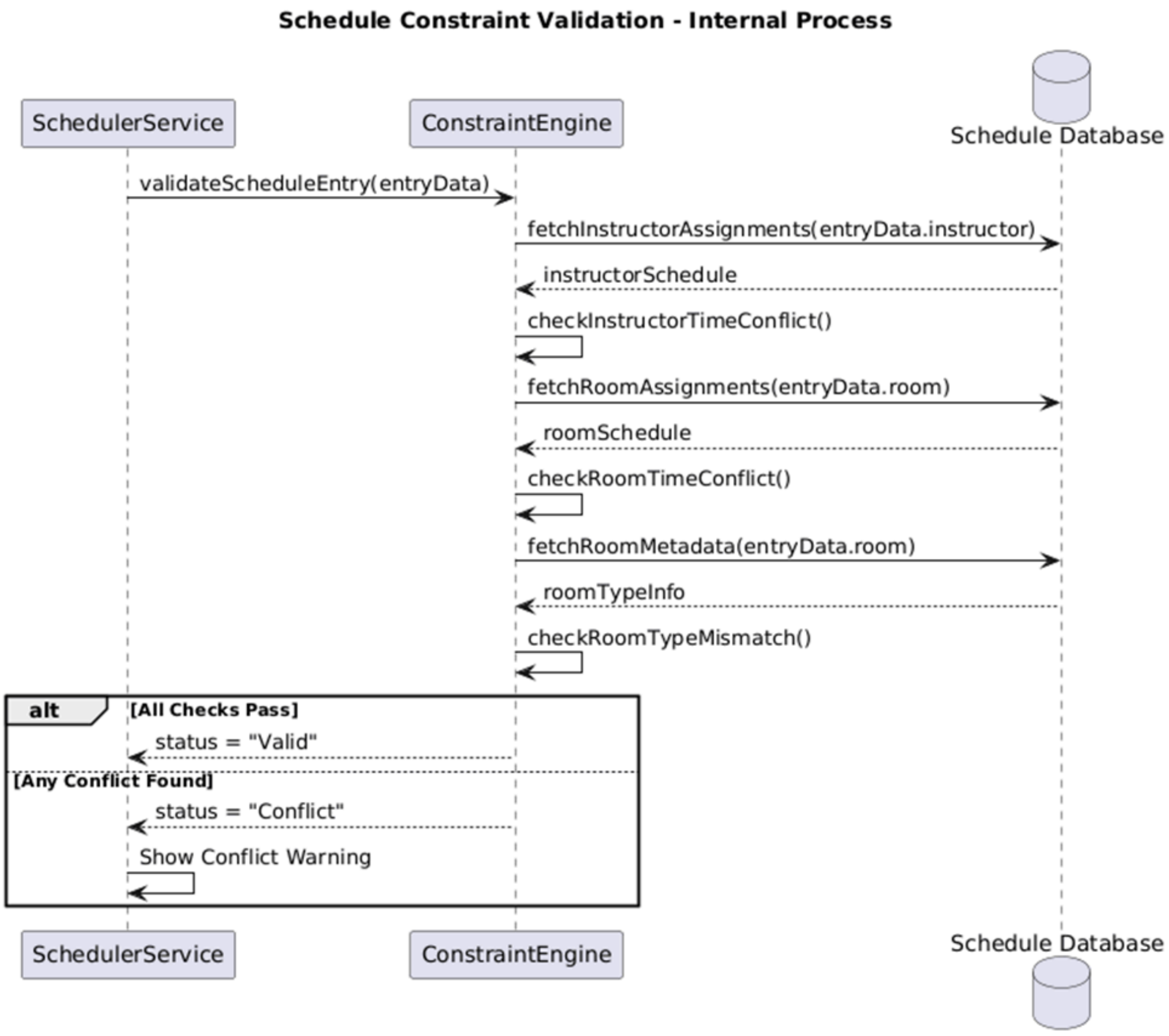
3.3.2:


Add Course
Administrator
Course Management UI
Course Service
Course Database
Navigate to Manage Course
requestAddCourseForm()
displayAddCourseForm()
Input Course(ID, Name, Type, Capacity)
Click 'Save'
submitCourseData(courseData)
validateInput()
alt
[Input Valid]
saveNewCourse(courseData)
saveSuccess
showSuccessConfirmation()
Display "Course added successfully"
[Input Invalid]
showErrorMessage(details)
Display validation errors
Administrator
Course Management UI
Course Service
Course Database

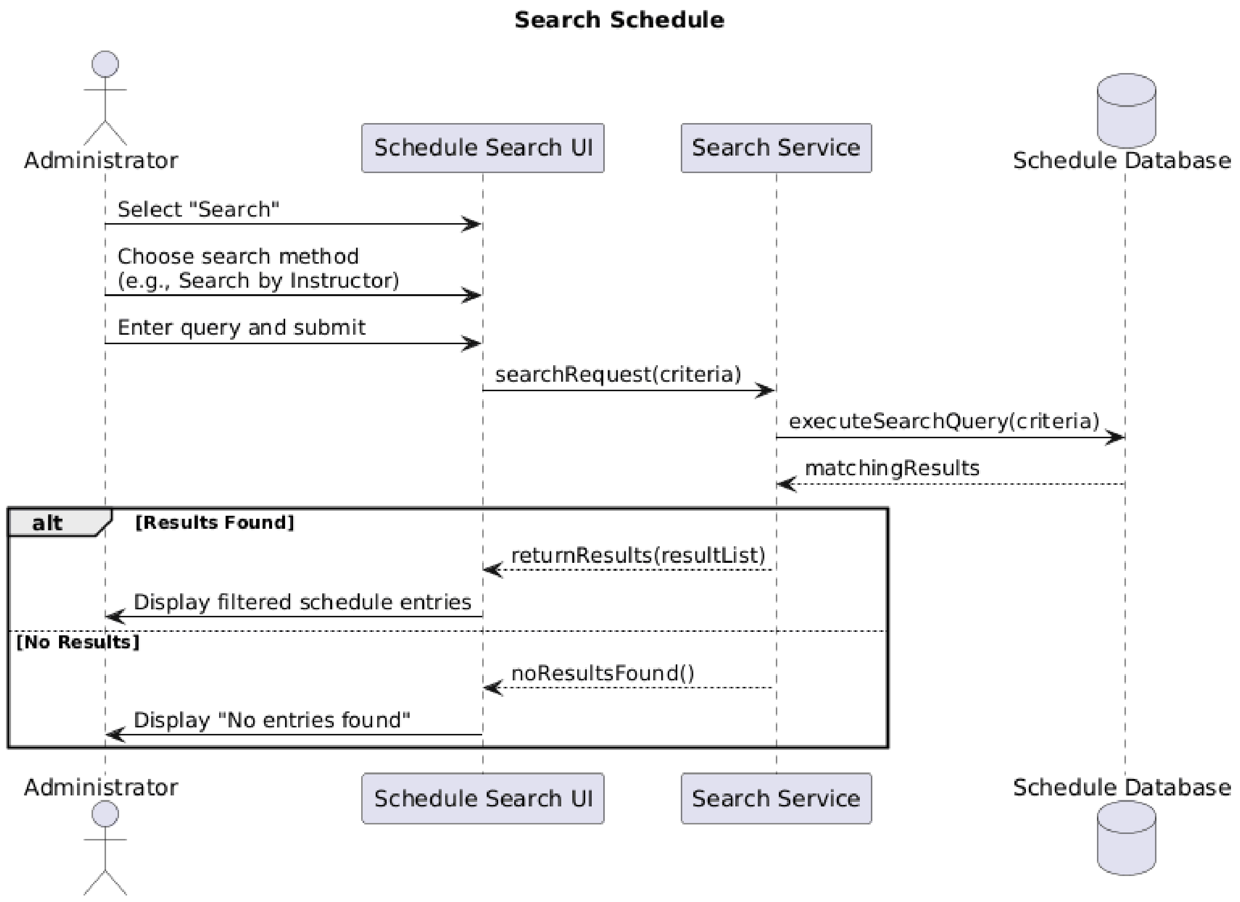
3.3.3:


Create Schedule Entry
Administrator
Schedule UI
Scheduler Service
Constraint Engine
Schedule Database
Select "Create Schedule Entry"
Display entry form
Select Course, Section, Instructor, Timeslot, Room
Submit entry
submitScheduleEntry(entryData)
Schedule Constraint Validation
fetchExistingSchedules()
fetchInstructorAssignments()
fetchRoomAssignments()
scheduleData
validationResult
alt
[Entry Valid (No Conflict)]
saveNewScheduleEntry(entryData)
saveSuccess
showSuccessConfirmation()
Display "Schedule Entry Created"
[Conflict Detected]
Show Conflict Warning
Display conflict details
Request adjustment

3.3.4:


Schedule Constraint Validation - Internal Process
SchedulerService
ConstraintEngine
Schedule Database
validateScheduleEntry(entryData)
fetchInstructorAssignments(entryData.instructor)
instructorSchedule
checkInstructorTimeConflict()
fetchRoomAssignments(entryData.room)
roomSchedule
checkRoomTimeConflict()
fetchRoomMetadata(entryData.room)
roomTypeInfo
checkRoomTypeMismatch()
alt
[All Checks Pass]
status = "Valid"
[Any Conflict Found]
status = "Conflict"
Show Conflict Warning
SchedulerService
ConstraintEngine
Schedule Database

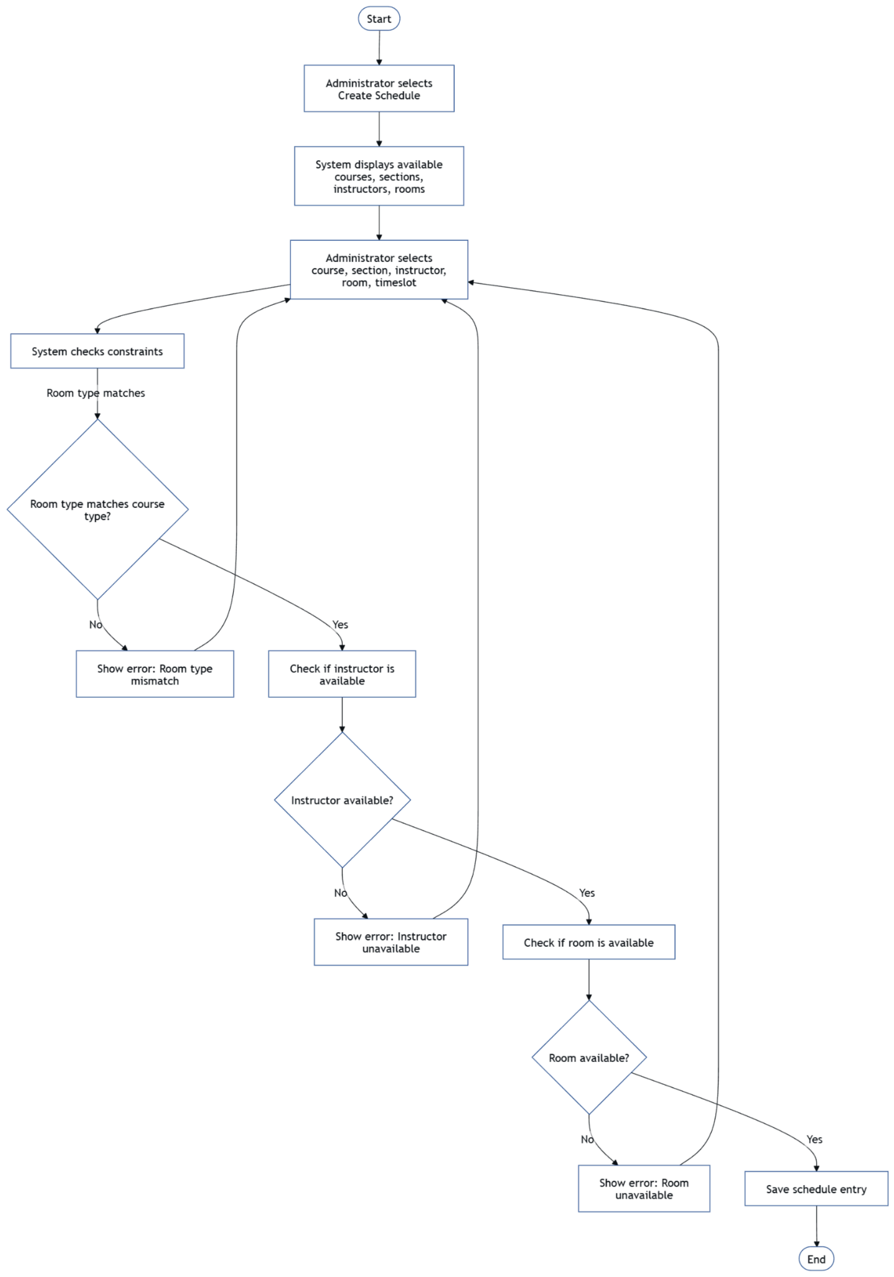
3.3.5:


Search Schedule
Administrator
Schedule Search UI
Search Service
Schedule Database
Select "Search"
Choose search method
(e.g., Search by Instructor)
Enter query and submit
searchRequest(criteria)
executeSearchQuery(criteria)
matchingResults
alt
[Results Found]
returnResults(resultList)
Display filtered schedule entries
[No Results]
noResultsFound()
Display "No entries found"

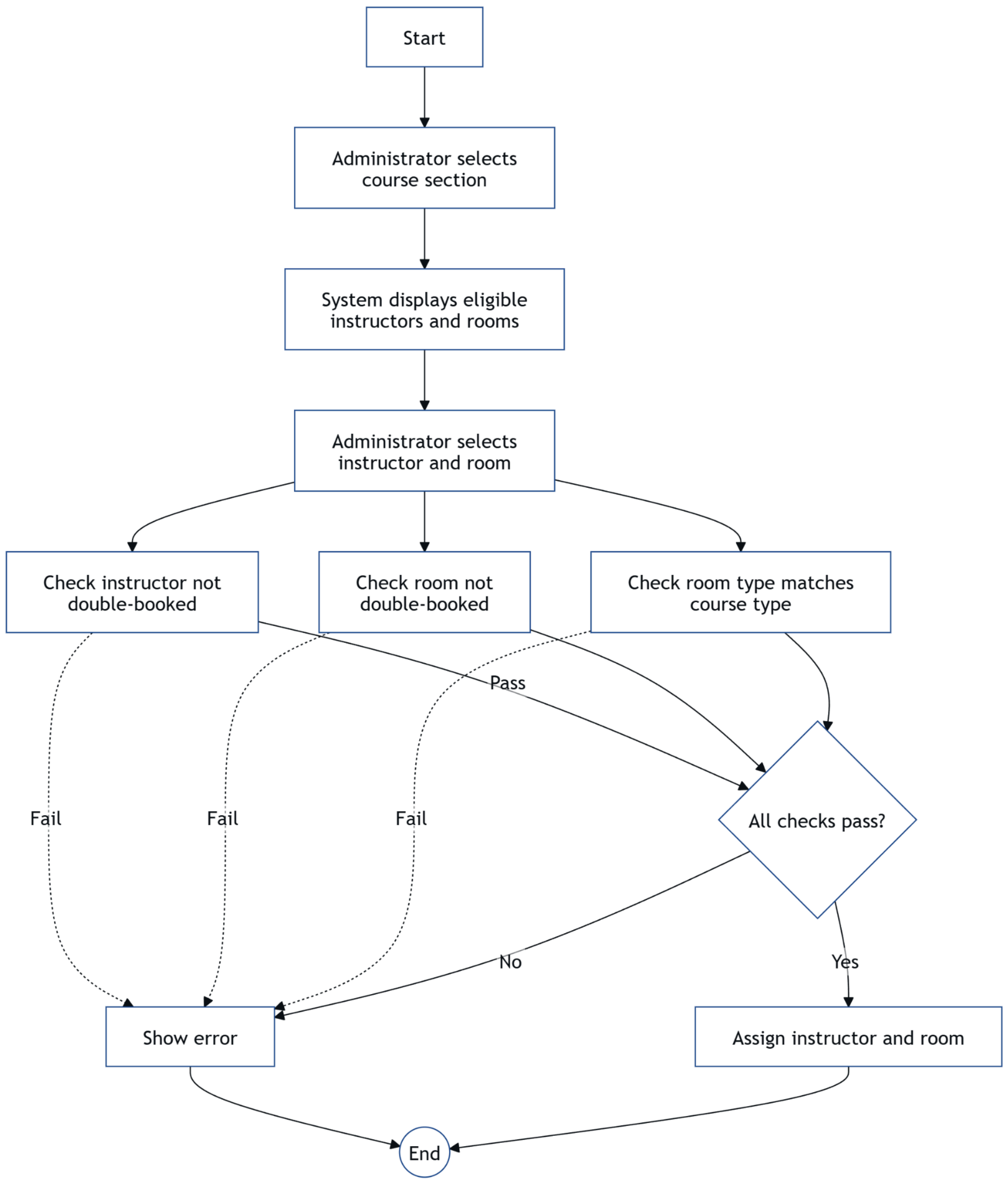
3.4. Activity Diagrams

This section has been covered by Nafis Karim (2221342642)

Class Schedule Management

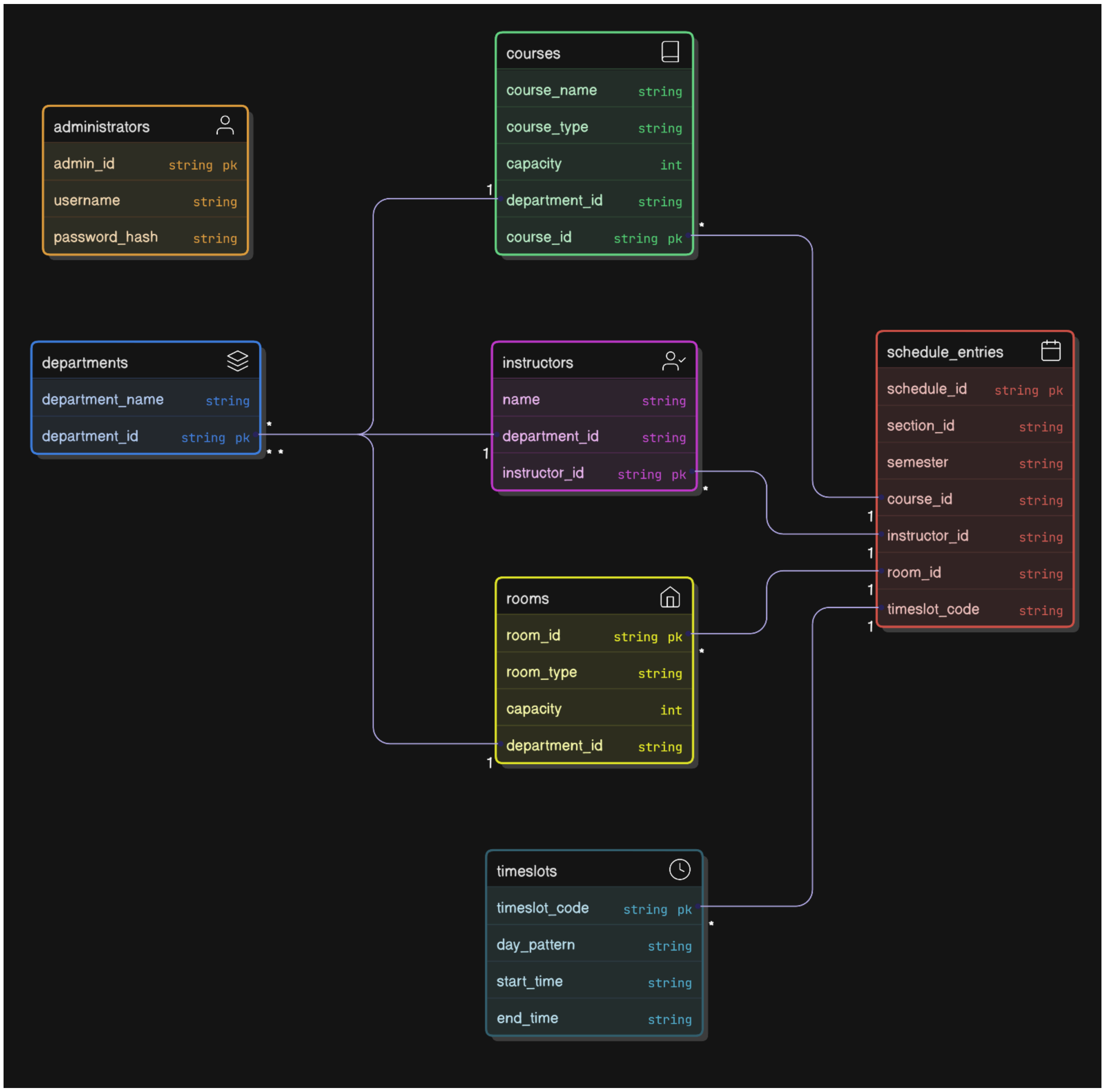

Start
Administrator selects Create Schedule
System displays available courses, sections, instructors, rooms
Administrator selects course, section, instructor, room, timeslot
System checks constraints
Room type matches
Room type matches course type?
No
Yes
Show error: Room type mismatch
Check if instructor is available
Instructor available?
No
Yes
Show error: Instructor unavailable
Check if room is available
Room available?
No
Yes
Show error: Room unavailable
Save schedule entry
End

Course Selection


Start
Administrator selects course section
System displays eligible instructors and rooms
Administrator selects instructor and room
Check instructor not double-booked
Check room not double-booked
Check room type matches course type
Pass
Fail
Fail
Fail
All checks pass?
No
Yes
Show error
Assign instructor and room
End

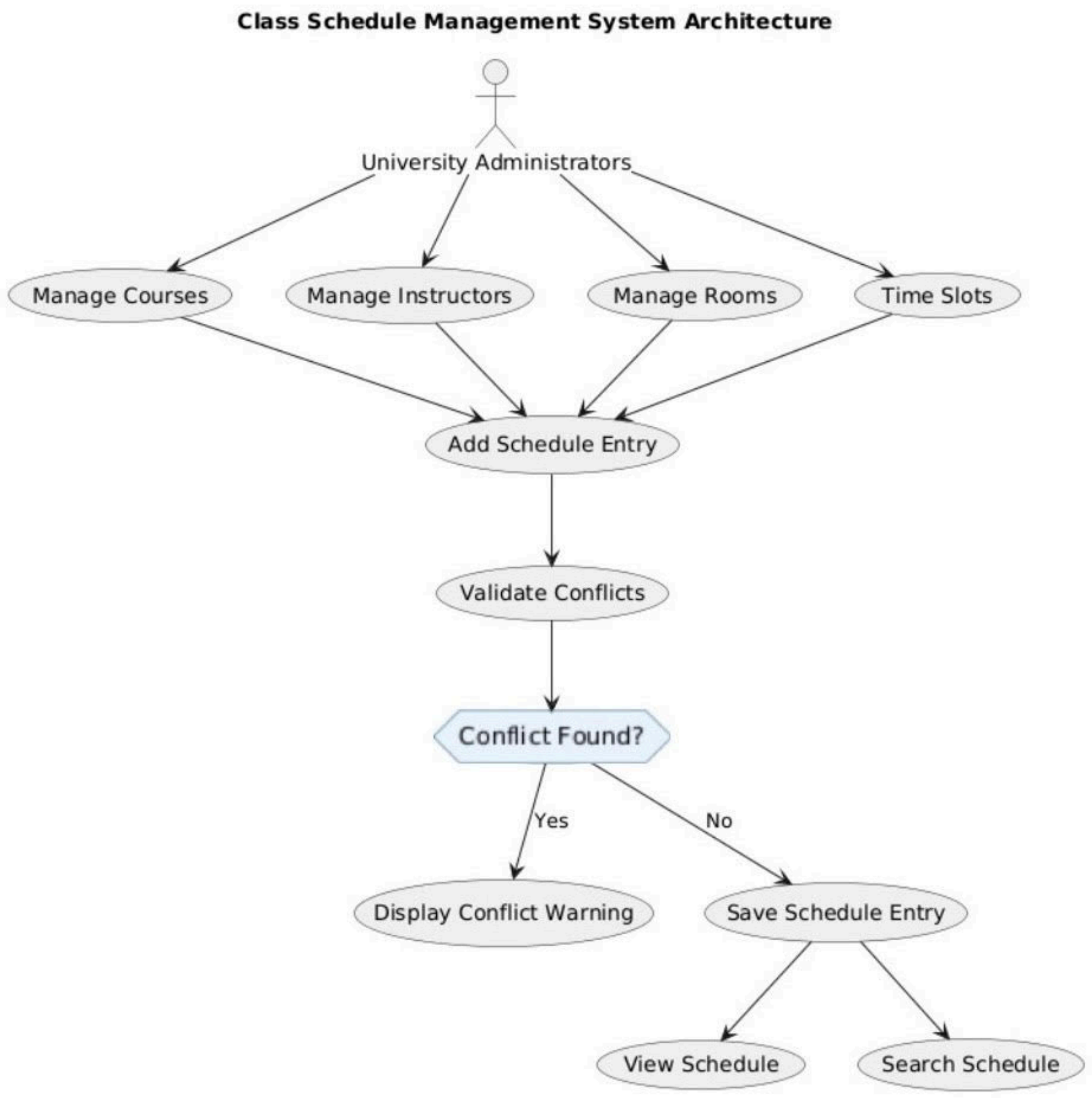
3.5: Class Diagram

This section has been covered by Susmita Goswami (2221409042)



3.6: High-Level Architecture Diagram

This section has been covered by Zinat Imteaz (2222004642)



4. Timeslot Rules and Scheduling Constraints

Timeslot Rules

Fixed Timeslot Patterns

The system uses five predefined, non-overlapping timeslots with 10-minute breaks between them to prevent scheduling conflicts and ensure adequate transition time between classes.

Timeslot ID	Label	Days	Start Time	End Time	Duration
TS1	ST1	Sunday & Tuesday	8:00 AM	9:30 AM	90 min
TS2	ST2	Sunday & Tuesday	9:40 AM	11:10 AM	90 min
TS3	MW1	Monday & Wednesday	11:20 AM	12:50 PM	90 min
TS4	MW2	Monday & Wednesday	1:00 PM	2:30 PM	90 min
TS5	RA1	Thursday & Saturday	2:40 PM	4:10 PM	90 min

Timeslot Characteristics

- **Fixed Scheduling**: Timeslot patterns are predefined and immutable; administrators cannot create custom timeslots.
- **Class Repetition**: A class scheduled in a given timeslot pattern repeats on both days of that pattern .
- **Break Time**: A 10-minute break separates consecutive timeslots to allow instructor and student movement between classrooms.
- **No Overlaps**: Timeslots are explicitly designed to have no time overlaps, ensuring clean scheduling and no time conflicts.

Scheduling Constraints

Instructor Availability Constraint

- **Rule**: No instructor can be assigned to teach more than one course section in the same timeslot.
- **Rationale**: An instructor cannot physically teach two classes simultaneously.

- **Violation Detection**: If an administrator attempts to assign an instructor to a second course in a timeslot where they already teach, the system flags this as a conflict.

Room Availability Constraint

- **Rule**: No room can host more than one class in the same timeslot.
- **Rationale**: A physical room has finite space and can only accommodate one class at a time.
- **Violation Detection**: If an administrator attempts to schedule two courses in the same room during an overlapping timeslot, the system detects and prevents the conflict.

Room Type Constraint (Course-Room Matching)

- **Rule**: Lab courses must be scheduled only in LAB-type rooms; theory courses must be scheduled only in THEORY-type rooms.
- **Course Types**:
    - **THEORY**: Lecture-based courses (e.g., Algorithms, Intro to Programming).
    - **LAB**: Hands-on laboratory or practical courses (e.g., Data Structures Lab, Circuits Lab).
- **Room Types**:
    - **THEORY**: Standard classrooms with lecture facilities.
    - **LAB**: Equipped with laboratory equipment, computers, or experimental apparatus.
- **Violation Detection**: If a LAB course is assigned to a THEORY room (or vice versa), the system detects this mismatch and flags it as a conflict.

Section Uniqueness Constraint

- **Rule**: Each course can have multiple sections (A, B, C, etc.), and each section is treated as a separate schedule entry.
- **Example**: CSE101 may have sections A, B, and C, each with its own instructor, room, and timeslot.
- **Implication**: Two sections of the same course can be taught by different instructors in different rooms and timeslots, but the same course code + section combination cannot appear twice.

Schedule Entry Composition Constraint

- **Rule**: A valid schedule entry consists of exactly five components:
    1. **Course** (course code, e.g., CSE101)
    2. **Section** (e.g., A, B, C)

    3. **Instructor** (assigned teacher, e.g., Dr. Rahman)
    4. **Timeslot** (predefined pattern, e.g., ST1)
    5. **Room** (physical classroom, e.g., Room 101)
- **Implication**: All five components must be specified; a schedule entry is incomplete without any of them.

No Duplicate Entries Constraint

- **Rule**: The same combination of (Course + Section + Instructor + Timeslot + Room) cannot exist twice in the schedule.
- **Rationale**: Prevents accidental duplicate scheduling and maintains data integrity.

Capacity Constraint (Optional, for future enhancement)

- **Rule**: The number of students enrolled in a course section should not exceed the room's seating capacity.
- **Current Status**: Not enforced in the current frontend prototype but should be implemented when room capacity data is added to the system.

Conflict Detection Rules

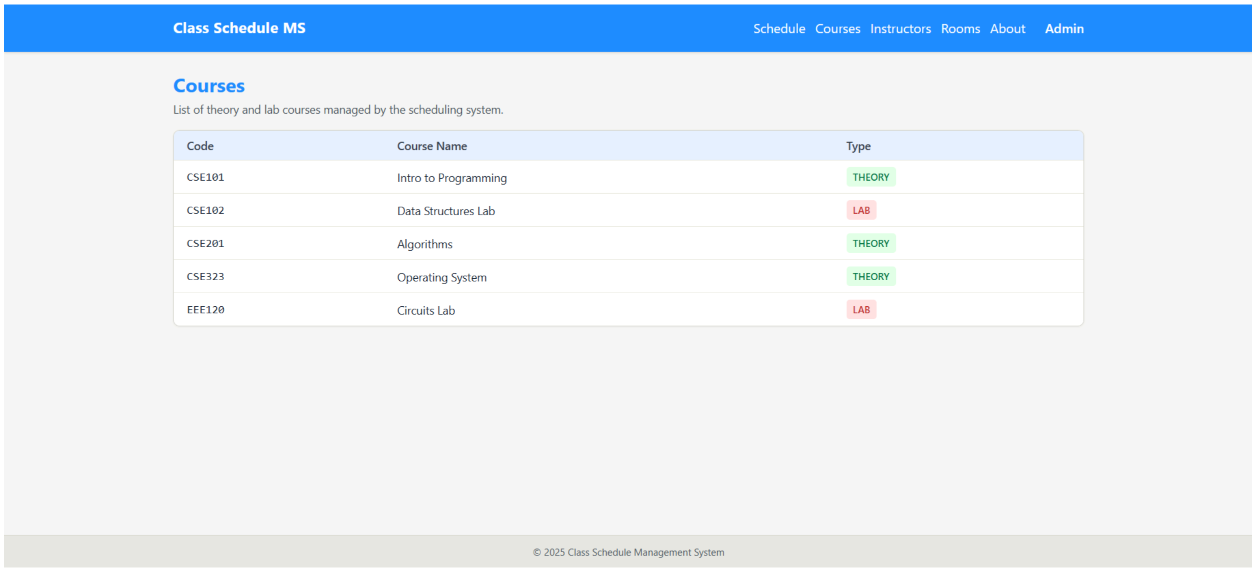
The system detects conflicts when an administrator attempts to add or modify a schedule entry. A conflict exists if:

1. **Instructor Clash**: The selected instructor is already teaching a different course in the chosen timeslot.
2. **Room Clash**: The selected room is already booked for a different course in the chosen timeslot.
3. **Room Type Mismatch**: A LAB course is assigned to a THEORY room, or a THEORY course is assigned to a LAB room.
4. **Incomplete Data**: Required fields (course, section, instructor, timeslot, room) are missing.

When conflicts are detected, the system:

- Displays conflict messages to the administrator in the Conflict Status panel**.**
- Allows the administrator to either:
    - Modify the entry to resolve conflicts and re-submit.
    - Save as Draft to store the conflicted entry for later review and resolution in the Conflicts page.

5. Prototype Demonstration:



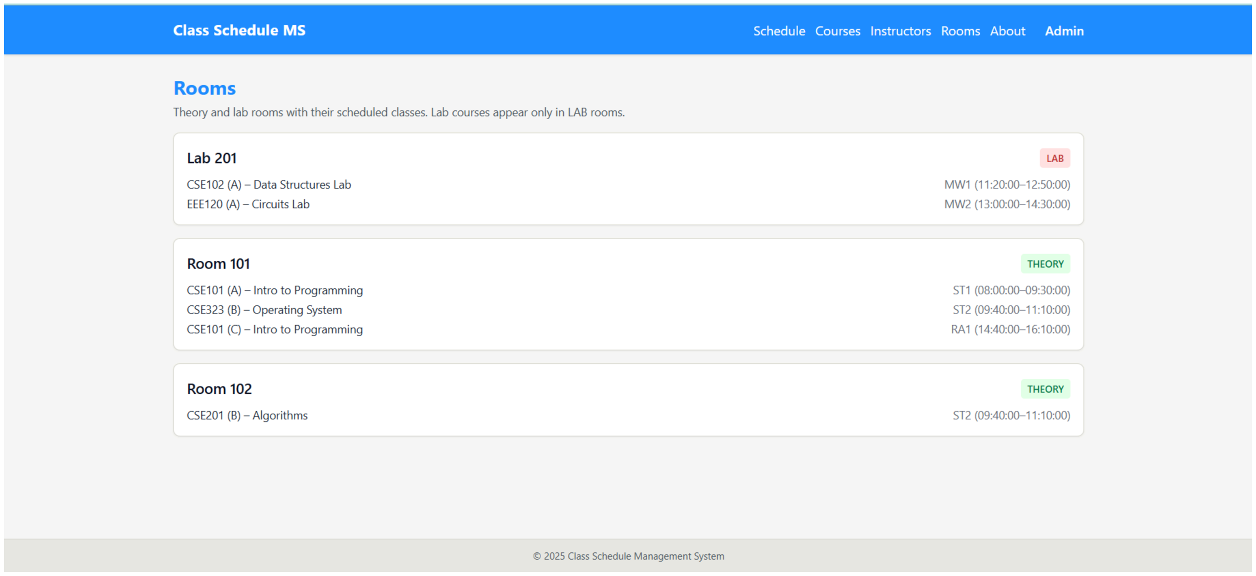
Class Schedule MS — Schedule Courses Instructors Rooms About Admin

Courses

List of theory and lab courses managed by the scheduling system.

Code	Course Name	Type
CSE101	Intro to Programming	THEORY
CSE102	Data Structures Lab	LAB
CSE201	Algorithms	THEORY
CSE323	Operating System	THEORY
EEE120	Circuits Lab	LAB

© 2025 Class Schedule Management System



Class Schedule MS — Schedule Courses Instructors Rooms About Admin

Rooms

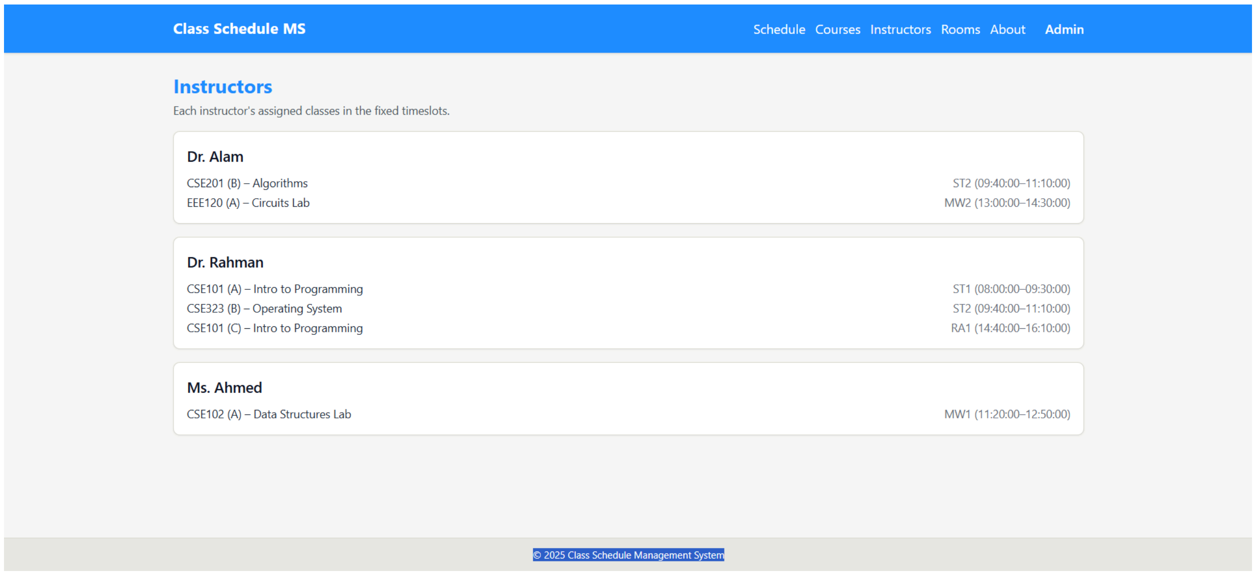
Theory and lab rooms with their scheduled classes. Lab courses appear only in LAB rooms.

Lab 201 — LAB
- CSE102 (A) – Data Structures Lab — MW1 (11:20:00–12:50:00)
- EEE120 (A) – Circuits Lab — MW2 (13:00:00–14:30:00)

Room 101 — THEORY
- CSE101 (A) – Intro to Programming — ST1 (08:00:00–09:30:00)
- CSE323 (B) – Operating System — ST2 (09:40:00–11:10:00)
- CSE101 (C) – Intro to Programming — RA1 (14:40:00–16:10:00)

Room 102 — THEORY
- CSE201 (B) – Algorithms — ST2 (09:40:00–11:10:00)

© 2025 Class Schedule Management System



Class Schedule MS

Schedule Courses Instructors Rooms About Admin

Instructors

Each instructor's assigned classes in the fixed timeslots.

Dr. Alam

CSE201 (B) – Algorithms	ST2 (09:40:00–11:10:00)
EEE120 (A) – Circuits Lab	MW2 (13:00:00–14:30:00)

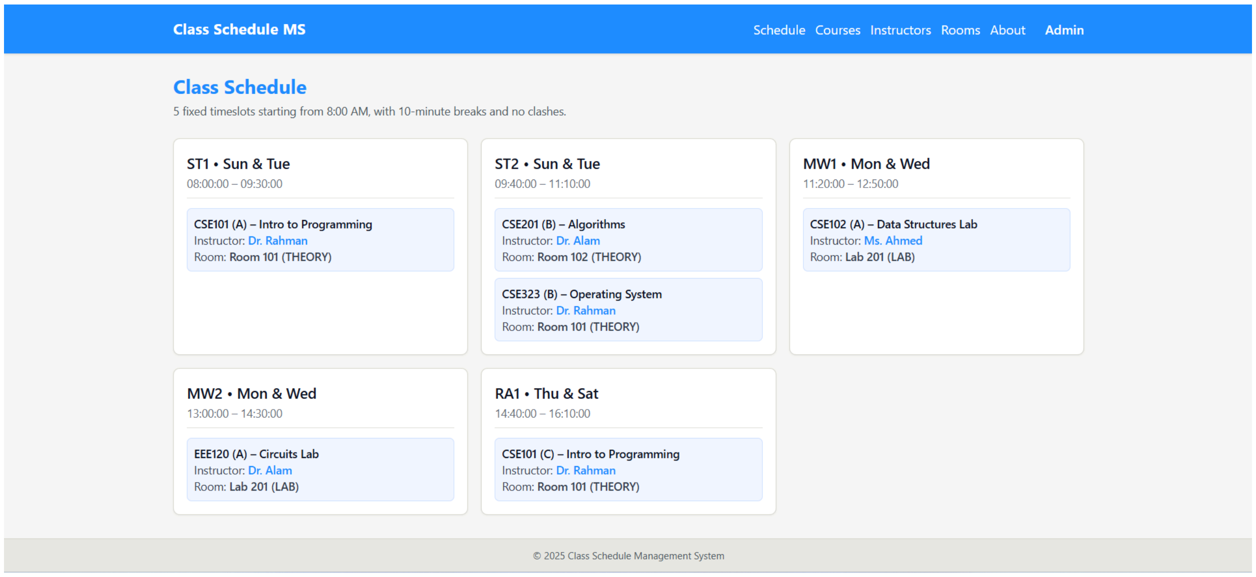
Dr. Rahman

CSE101 (A) – Intro to Programming	ST1 (08:00:00–09:30:00)
CSE323 (B) – Operating System	ST2 (09:40:00–11:10:00)
CSE101 (C) – Intro to Programming	RA1 (14:40:00–16:10:00)

Ms. Ahmed

CSE102 (A) – Data Structures Lab	MW1 (11:20:00–12:50:00)

© 2025 Class Schedule Management System



Class Schedule MS

Schedule Courses Instructors Rooms About Admin

Class Schedule

5 fixed timeslots starting from 8:00 AM, with 10-minute breaks and no clashes.

ST1 • Sun & Tue
08:00:00 – 09:30:00

CSE101 (A) – Intro to Programming
Instructor: Dr. Rahman
Room: Room 101 (THEORY)

ST2 • Sun & Tue
09:40:00 – 11:10:00

CSE201 (B) – Algorithms
Instructor: Dr. Alam
Room: Room 102 (THEORY)

CSE323 (B) – Operating System
Instructor: Dr. Rahman
Room: Room 101 (THEORY)

MW1 • Mon & Wed
11:20:00 – 12:50:00

CSE102 (A) – Data Structures Lab
Instructor: Ms. Ahmed
Room: Lab 201 (LAB)

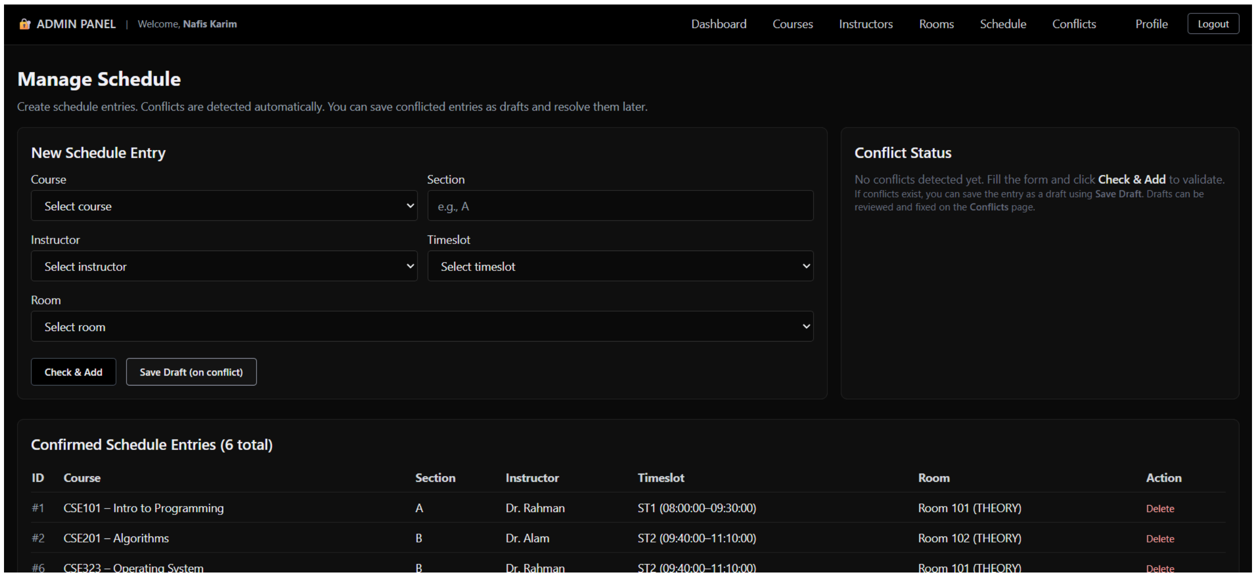
MW2 • Mon & Wed
13:00:00 – 14:30:00

EEE120 (A) – Circuits Lab
Instructor: Dr. Alam
Room: Lab 201 (LAB)

RA1 • Thu & Sat
14:40:00 – 16:10:00

CSE101 (C) – Intro to Programming
Instructor: Dr. Rahman
Room: Room 101 (THEORY)

© 2025 Class Schedule Management System


ADMIN PANEL | Welcome, **Nafis Karim**

Dashboard Courses Instructors Rooms Schedule Conflicts Profile Logout

Manage Schedule

Create schedule entries. Conflicts are detected automatically. You can save conflicted entries as drafts and resolve them later.

New Schedule Entry

Course: Select course

Section: e.g., A

Instructor: Select instructor

Timeslot: Select timeslot

Room: Select room

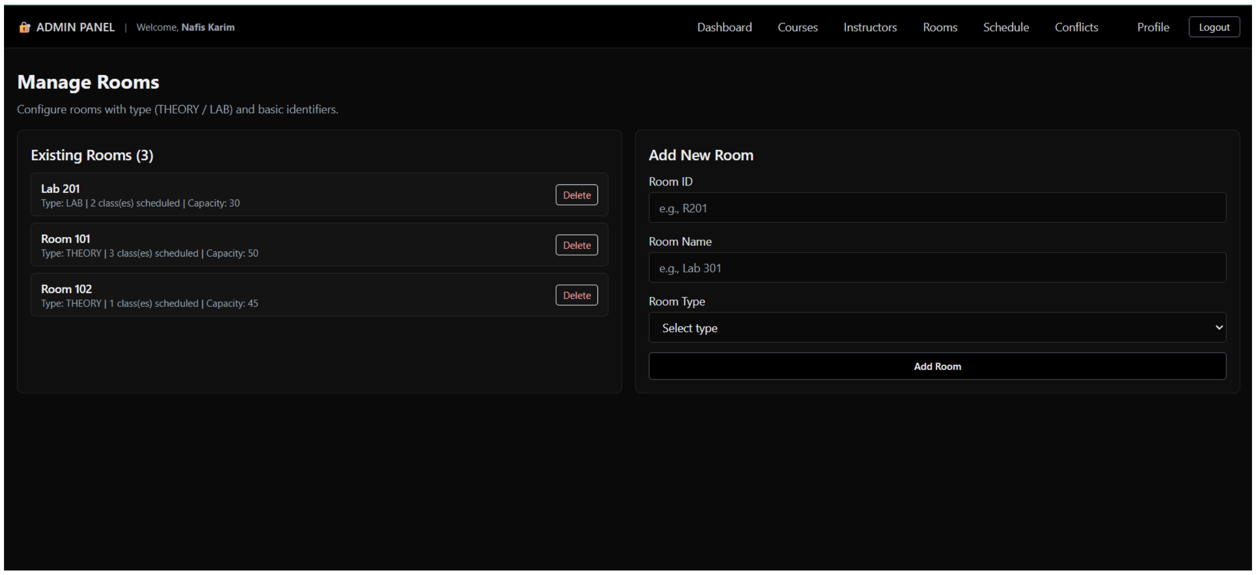
Check & Add | Save Draft (on conflict)

Conflict Status

No conflicts detected yet. Fill the form and click **Check & Add** to validate.
If conflicts exist, you can save the entry as a draft using **Save Draft**. Drafts can be reviewed and fixed on the **Conflicts** page.

Confirmed Schedule Entries (6 total)

ID	Course	Section	Instructor	Timeslot	Room	Action
#1	CSE101 – Intro to Programming	A	Dr. Rahman	ST1 (08:00:00–09:30:00)	Room 101 (THEORY)	Delete
#2	CSE201 – Algorithms	B	Dr. Alam	ST2 (09:40:00–11:10:00)	Room 102 (THEORY)	Delete
#6	CSE323 – Operating System	B	Dr. Rahman	ST2 (09:40:00–11:10:00)	Room 101 (THEORY)	Delete


ADMIN PANEL | Welcome, **Nafis Karim**

Dashboard Courses Instructors Rooms Schedule Conflicts Profile Logout

Manage Rooms

Configure rooms with type (THEORY / LAB) and basic identifiers.

Existing Rooms (3)

Lab 201
Type: LAB | 2 class(es) scheduled | Capacity: 30 — Delete

Room 101
Type: THEORY | 3 class(es) scheduled | Capacity: 50 — Delete

Room 102
Type: THEORY | 1 class(es) scheduled | Capacity: 45 — Delete

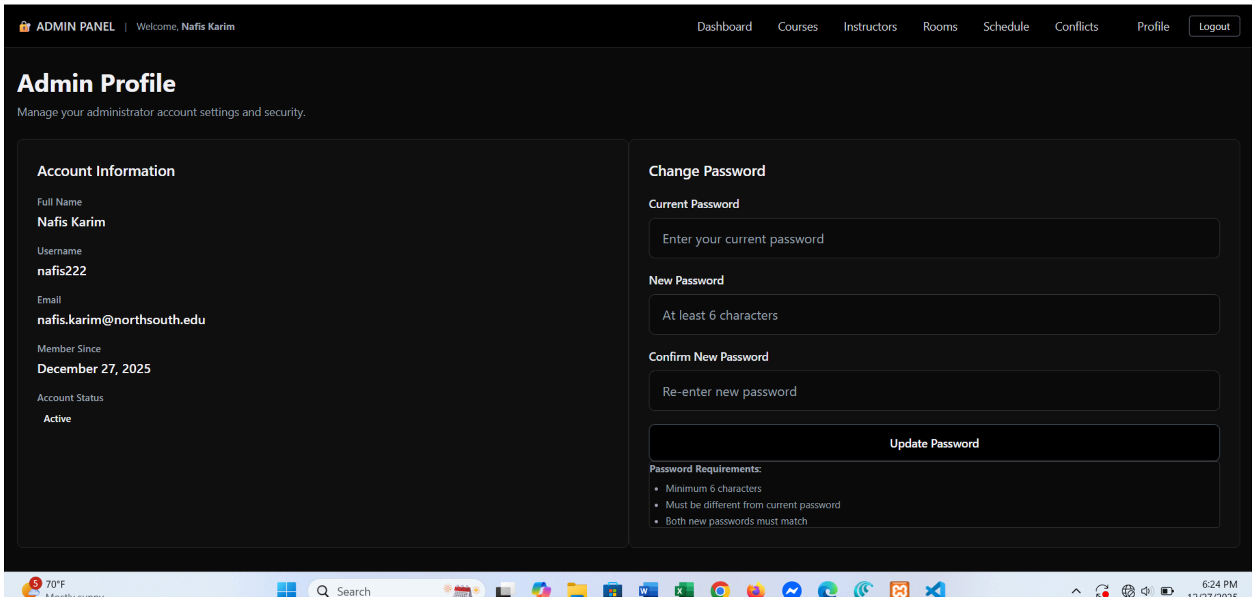
Add New Room

Room ID: e.g., R201

Room Name: e.g., Lab 301

Room Type: Select type

Add Room



ADMIN PANEL | Welcome, **Nafis Karim**

Dashboard Courses Instructors Rooms Schedule Conflicts Profile Logout

Admin Profile

Manage your administrator account settings and security.

Account Information

Full Name
Nafis Karim

Username
nafis222

Email
nafis.karim@northsouth.edu

Member Since
December 27, 2025

Account Status
Active

Change Password

Current Password
Enter your current password

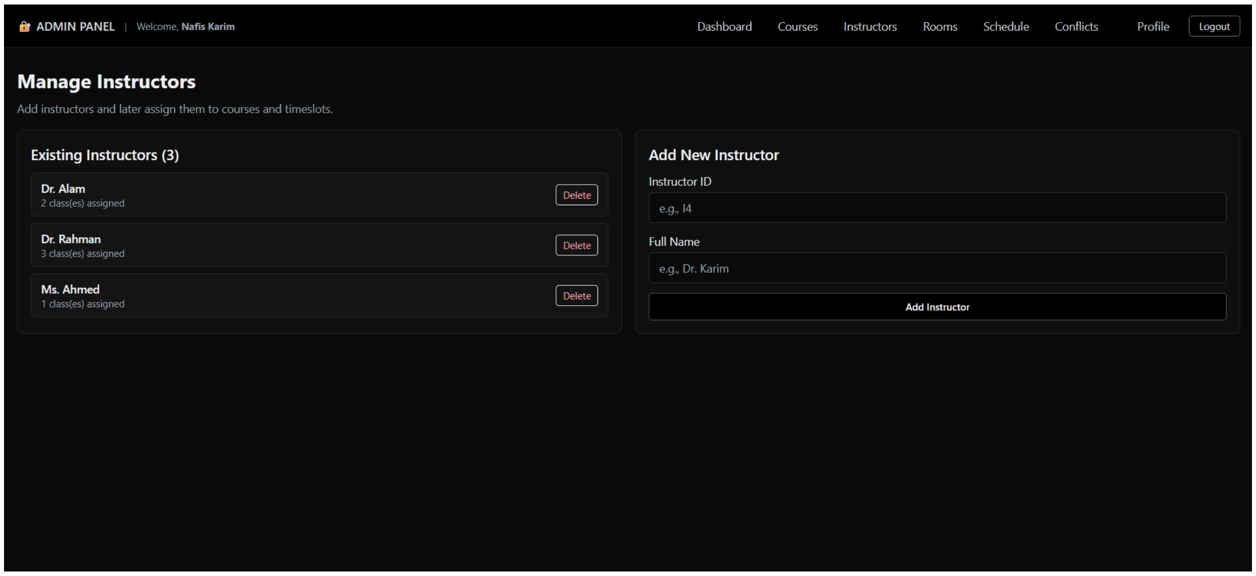
New Password
At least 6 characters

Confirm New Password
Re-enter new password

Update Password

Password Requirements:
- Minimum 6 characters
- Must be different from current password
- Both new passwords must match

70°F Search 6:24 PM



ADMIN PANEL | Welcome, **Nafis Karim**

Dashboard Courses Instructors Rooms Schedule Conflicts Profile Logout

Manage Instructors

Add instructors and later assign them to courses and timeslots.

Existing Instructors (3)

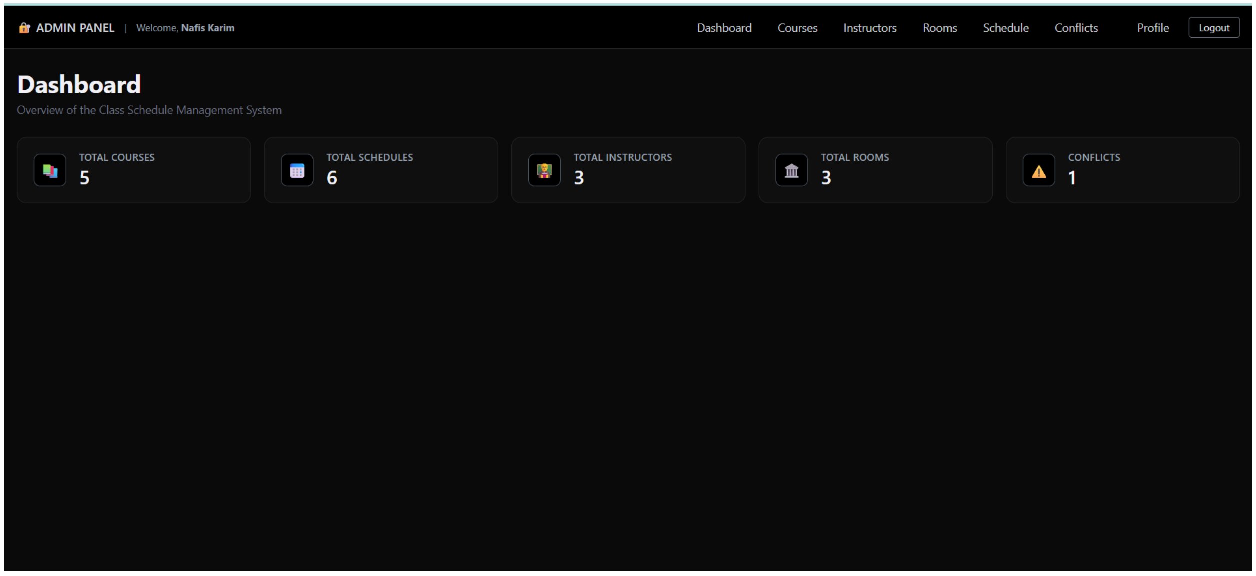
Dr. Alam 2 class(es) assigned	Delete
Dr. Rahman 3 class(es) assigned	Delete
Ms. Ahmed 1 class(es) assigned	Delete

Add New Instructor

Instructor ID
e.g., I4

Full Name
e.g., Dr. Karim

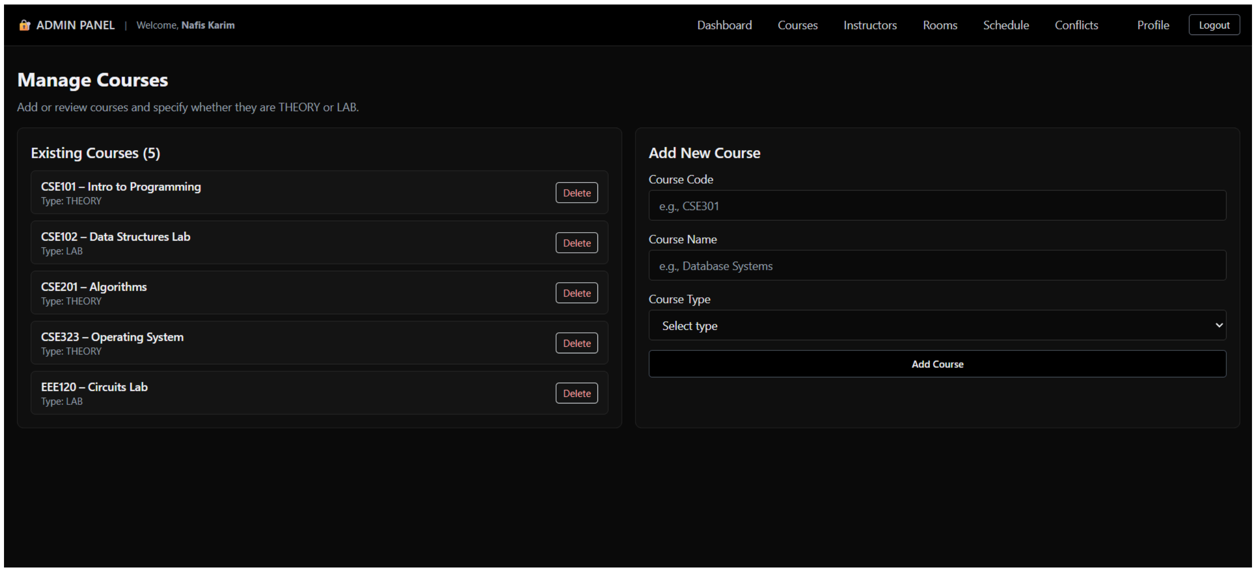
Add Instructor



Dashboard

Overview of the Class Schedule Management System

TOTAL COURSES	5
TOTAL SCHEDULES	6
TOTAL INSTRUCTORS	3
TOTAL ROOMS	3
CONFLICTS	1



Manage Courses

Add or review courses and specify whether they are THEORY or LAB.

Existing Courses (5)

CSE101 – Intro to Programming
Type: THEORY
Delete

CSE102 – Data Structures Lab
Type: LAB
Delete

CSE201 – Algorithms
Type: THEORY
Delete

CSE323 – Operating System
Type: THEORY
Delete

EEE120 – Circuits Lab
Type: LAB
Delete

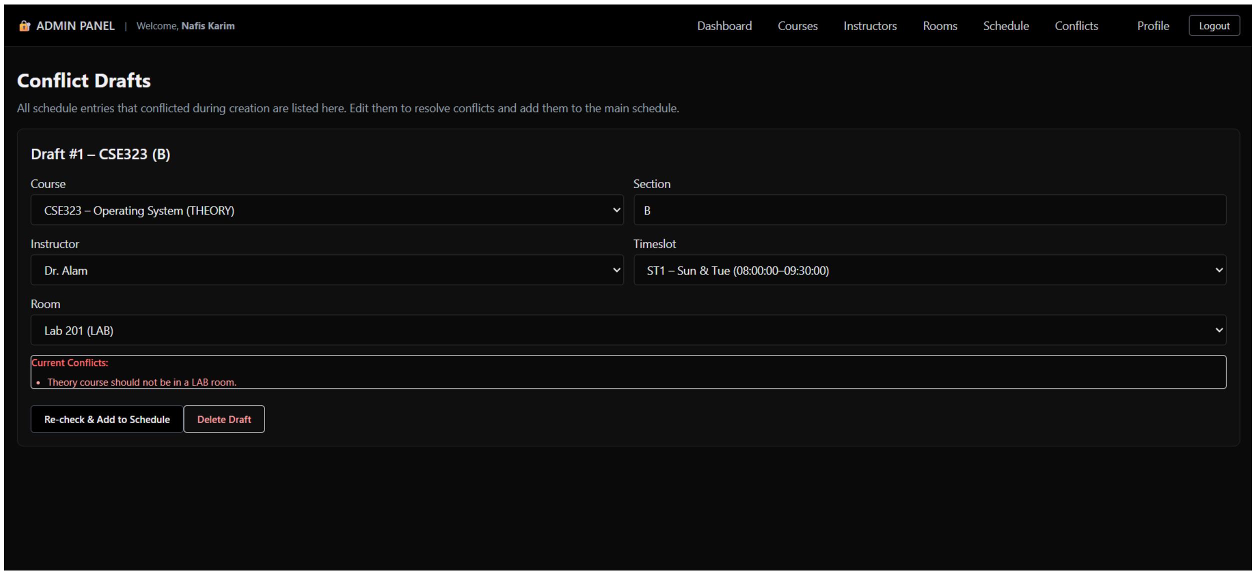
Add New Course

Course Code: e.g., CSE301

Course Name: e.g., Database Systems

Course Type: Select type

Add Course



Conflict Drafts

All schedule entries that conflicted during creation are listed here. Edit them to resolve conflicts and add them to the main schedule.

Draft #1 – CSE323 (B)

Course: CSE323 – Operating System (THEORY)

Section: B

Instructor: Dr. Alam

Timeslot: ST1 – Sun & Tue (08:00:00–09:30:00)

Room: Lab 201 (LAB)

Current Conflicts:

- Theory course should not be in a LAB room.

6. Testing Summary

Testing was carried out in multiple phases to ensure the Class Schedule Management System (CSMS) works correctly:

Unit Testing:
Tested individual components such as course management, instructor availability checks, room assignments, and conflict detection to ensure correctness.

Integration Testing:
Verified that multiple modules worked together properly—for example, creating a schedule entry and checking for conflicts simultaneously.

System Testing:
Conducted end-to-end tests to ensure the full system operates correctly, including schedule creation, validation, conflict detection, and UI interactions.

Test Cases

Test Case 1: Add a valid class schedule (no conflicts).

- **Expected:** Schedule added successfully.
- **Result:** Passed.

Test Case 2: Attempt to add a schedule with a double-booked instructor.

- **Expected:** Conflict warning displayed.
- **Result:** Passed.

Test Case 3: Assign a non-lab course to a lab room.

- **Expected:** Error message displayed.
- **Result:** Passed.

Test Case 4: Assign the same room to two different courses at the same timeslot.

- **Expected:** Conflict warning displayed.
- **Result:** Passed.

Test Case 5: Search schedule by instructor or room.

- **Expected:** Correct filtered results displayed.
- **Result:** Passed.

Test Case 6: Admin login with invalid credentials.

- **Expected:** Login error displayed; no access granted.
- **Result:** Passed.

7. Challenges Faced

Developing the Class Schedule Management System (CSMS) involved several technical and design-related challenges. These challenges arose mainly due to the complexity of academic scheduling rules, system constraints, and the need to ensure data integrity while maintaining usability. The key challenges faced during the development process are described below.

7.1 Handling Complex Scheduling Constraints

One of the major challenges was accurately implementing and enforcing multiple scheduling constraints simultaneously. The system needed to ensure that:

- An instructor is not assigned to more than one class in the same timeslot.
- A room is not double-booked for overlapping timeslots.
- Course types (LAB or THEORY) strictly match corresponding room types.

Designing a validation mechanism that checks all these constraints in real time, without affecting system performance, requires careful logical structuring and efficient database queries.

7.2 Timeslot Pattern Management

Another challenge was managing predefined timeslot patterns (ST, MW, RA) and ensuring that scheduling logic correctly interpreted repeated days. Since each timeslot applies to multiple days, conflict detection had to consider recurring schedules rather than single-day assignments. Implementing this logic while keeping the timeslots immutable added complexity to the scheduling engine.

7.3 Maintaining Data Consistency and Integrity

Ensuring data consistency across interconnected entities such as courses, sections, instructors, rooms, and schedule entries was challenging. Any update or deletion in one entity could potentially affect multiple schedule records. Proper use of foreign keys, validation rules, and controlled delete/update operations was required to prevent orphan records and data mismatches.

7.4 Conflict Detection and User Feedback

Providing clear, understandable conflict messages to administrators was another difficulty. Instead of generic error messages, the system had to explain exactly why a schedule entry failed (e.g., instructor conflict or room type mismatch). Designing meaningful, human-readable feedback while maintaining technical accuracy required careful error-handling logic.

7.5 Balancing Usability with Functionality

The system targets administrators who may not have advanced technical knowledge. Therefore, designing an interface that is simple, intuitive, and efficient- while still supporting complex scheduling operations- was challenging. Dropdown-based selections, clear navigation, and minimal clicks were incorporated to improve usability without reducing system capabilities.

7.6 Local Deployment and Testing Limitations

Since the system was deployed and tested on a localhost environment, certain real-world deployment considerations such as multi-user concurrency, live server load, and cloud-based scalability could not be fully tested. This limitation required careful simulation of possible conflicts and edge cases during development.

8. Conclusion

The Class Schedule Management System (CSMS) successfully addresses the challenges associated with manual academic scheduling by providing a structured, automated, and conflict-aware solution for university administrators. The system effectively manages courses, sections, instructors, rooms, and timeslots while enforcing essential scheduling constraints such as instructor availability, room availability, and course-room type compatibility.

Through a modular design and clearly defined functional and non-functional requirements, the CSMS ensures reliability, maintainability, and ease of use. Real-time conflict detection, clear error feedback, and role-based access control enhance both system accuracy and security. The use of modern web technologies such as HTML5, Tailwind CSS, PHP, and MySQL allows the system to remain responsive, scalable, and easy to extend.

Although the current implementation is limited to a localhost environment and focuses primarily on administrative scheduling, the system provides a strong foundation for future enhancements. Potential extensions include student enrollment integration, automatic room capacity enforcement, advanced optimization algorithms for scheduling, and deployment on a live university server.