# OBJECT-ORIENTED SOFTWARE ENGINEERING (SE204n)
# LAB FILE

**Subject Code: SE204n**
Subject Name: Object-Oriented Software Engineering
Branch: Software Engineering
Year: 2nd Year/4th Semester

Submitted by:
**ASTHA AGARWAL**
**23/SE/184**

Submitted to:
**Prof. Ruchika Malhotra**
**Head of Department**
**Department of Software Engineering**
**Delhi Technological University**

Delhi Technological University
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

# Vision and Mission of Department

## Vision

Department of Software Engineering to be a leading world class technology department playing its role as a key node in national and global knowledge network, thus empowering the computer science industry with the wings of knowledge and power of innovation.

## Mission

- To nurture talent of students for research, innovation and excellence in the field of computer engineering starting from Under graduate level.

- To develop highly analytical and qualified computer engineers by imparting training on cutting edge technology.

- To produce socially sensitive computer engineers with professional ethics.

- To focus on R&D environment in close partnership with industry and foreign universities.

- To produce well-rounded, up to date, scientifically tempered, design oriented engineers and scientists capable of lifelong learning.

# Program Educational Objectives

- **PEO 1**: To acquire in-depth knowledge of software and hardware techniques, which provide a strong foundation to pursue continuing education and nurture the talent for innovation and research.
- **PEO 2**: To nurture the talent in leadership qualities, at an appropriate level in order to address the issues in a responsive, ethical and innovative manner.
- **PEO 3**: To excel in careers by being a part of the success and growth of an organization with whom they will be associated.
- **PEO 4**: To inculcate the ability for lifelong learning by active participation in self-study courses, seminars, research projects.

# Program Specific Outcome (PSOs)

- **PSO1**: Design, analyze and develop engineering problems.
- **PSO2**: Specify, design, develop, test and maintain usable systems that behave reliably and efficiently and satisfy all the requirements that customers have defined for them.
- **PSO3**: Develop software systems that would perform tasks related to Research, Education and Training and/or E-governance.

# Program Outcomes

- **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

- **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

- **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

- **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

- **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

- **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

- **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

- **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

- **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

- **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

# INDEX

# EXPERIMENT-1

AIM: Draft the Problem Statement for the course management system

## Description:

The software should enable efficient management of all the courses a student has opted for online. The students should be able to view their attendance,view and download course material, check their grades and upload submissions. A faculty can upload their assignments, view grades and accept submissions with the help of the system.

Limitations of conventional course management system:

1. Lack of Attendance Visibility for Students:
   In conventional course management systems, students do not have access to their attendance records. This lack of transparency makes it difficult for them to track their attendance status, potentially leading to missed opportunities for improvement and accountability.
2. Limited Networking Opportunities:
   Traditional course management systems do not provide details about classmates, which hinders networking and collaboration among students. The absence of a shared platform for interaction makes it challenging for students to connect, discuss course materials, or form study groups.
3. Absence of Overall Grade Overview:
   Students are often unable to view their overall grades and performance summaries in conventional systems. This limitation prevents them from assessing their progress throughout the course and identifying areas that require additional effort.
4. Lack of Performance Analytics for Teachers:
   In existing systems, teachers do not have access to comprehensive performance analytics, such as overall grade distributions and average scores. As a result, it is difficult to identify trends and address common areas of difficulty.

Features of the proposed system:

1. Transparent Attendance Tracking for Students:
   The proposed system will provide students with real-time access to their attendance records. This feature will enhance transparency and allow students to monitor their attendance status, enabling them to stay informed and take necessary actions to maintain required attendance levels.
2. Enhanced Networking and Collaboration:
   The new system will include a dedicated platform where students can view details of their classmates, enhancing peer-to-peer learning.
3. Comprehensive Grade Overview for Students:
   The proposed system will offer students an overall view of their grades and

performance trends across various assessments. This feature will help them track their academic progress, set personal goals, and work on areas needing improvement.

4. Advanced Performance Analytics for Teachers:
   Teachers will have access to detailed performance analytics, including overall grade distributions and average score trends. This feature will provide valuable insights into class performance, enabling educators to make data-driven decisions and offer targeted support to students.

# EXPERIMENT-2

AIM: Draft the Initial requirements document for the course management system

| | |
|---|---|
| Title of the Project | Course Management System |
| Stakeholders involved in capturing requirements | Student, Instructor, Admin |
| Techniques used for requirement capturing | Brainstorming |
| Name of the persons along with designation | Vasavi Taneja - developer<br>Astha Agarwal - developer |
| Date | January, 2025 |
| Version | 1.0 |

**Consolidated list of requirements**

1. A system is to be implemented which can run on the user's LAN.
2. The system should be able to generate and maintain the login ID and password of all possible users.
3. There are two types of members in the course management system - students and faculty.
4. The administrator should be able to maintain the details of all the members of the course management system.
5. The administrator should be able to maintain the details of all the courses.
6. The faculty should be able to upload assignments, accept and grade submissions.
7. The faculty should be able to upload course material.
8. The faculty should be able to view student details.
9. The faculty should be able to update attendance of students.
10. The faculty should be able to calculate the average performance of the class/
11. The maximum number of files that can be uploaded for each submission is 3.
12. The student should be able to view their attendance for each course.
13. The student should be able to view courses they have registered for.
14. The student should be able to view and edit profile details.
15. The student should be able to download course material.
16. The student should be able to make submissions and check grades.
17. The student should be able to view details of classmates (email ID) and network with other students.
18. The system should be able to generate reports like:
    i) Details of all students and the courses they have enrolled in
    ii) Attendance details of all students for a given course to the faculty
    iii) Details of submissions made by students

# EXPERIMENT-3

# Software Requirements Specification Document for Library Management System

## Problem Statement

A software is to be developed to enable efficient management of all the courses a student has opted for online. The students should be able to view their attendance,view and download course material, check their grades and upload submissions. A faculty can upload their assignments, view grades and accept submissions with the help of the system.

The Course Management System performs the following functions:

### 1. Course Material and Assignment Management

- Upload Course Material (Faculty):
    - Faculty can browse and upload files to specific courses.
    - System validates course ID and checks file size constraints.
    - A deadline for assignment submission can be set.
    - Faculty have the option to undo uploads if needed.
- Download Course Material (Student):
    - Students can download materials uploaded for their registered courses.
    - System validates the course ID and ensures only enrolled students access the content.
    - Only materials uploaded up to the last lecture are available.
- Upload Assignments (Faculty):
    - Faculty can upload assignment files to the portal for students.
    - Upload process includes file validation and deadline setting.
    - Unauthorized faculty (invalid membership) are blocked from uploading.

### 2. Student Submissions and Grading

- Make Submissions (Student):
    - Students can upload assignment files for respective courses.
    - System enforces file size limits and submission deadlines.
    - Invalid course ID or late submissions are rejected with an error message.
    - Students can undo submissions before the deadline.
- Grade Submissions (Faculty):
    - Faculty can access a list of student submissions per course.
    - Submissions can be viewed and graded directly through the portal.
    - Defaulters (non-submitters) are also listed separately.
    - Invalid course ID or student ID results in an error and restart of the process.

### 3. Academic Monitoring and Access

- View Attendance (Student):

    - Students can view attendance records for their registered courses.
    - Attendance is shown up to the last recorded lecture.
    - Invalid course ID or premature exit from the process is handled appropriately.
- View Classmate Details (Student):
    - Students can access contact details (e.g., college email) of classmates in a course.
    - Details are managed and uploaded by the administrator.
    - Access is restricted to valid course registrations only.
- Calculate Average Performance (Faculty):
    - Faculty can compute the average grade performance of a class.
    - The system calculates based on the grades uploaded.
    - If no grades are uploaded or course ID is invalid, the operation fails gracefully.

### 4. Maintain Details (Student, Faculty, and Course Information)

- Maintain Student Details:
    - The system stores comprehensive student records including Name, student ID, email, registered courses, and submission history.
    - Administrators can add, update, or delete student records.
    - Updates ensure synchronization with current course enrollments and grade records.
- Maintain Faculty Details:
    - Faculty records include name, faculty ID, department, and assigned courses.
    - Faculty membership is verified for upload and grading permissions.
    - Admins can update faculty information or revoke access (e.g., on expiry of membership).
- Maintain Course Details:
    - Each course includes course ID, course name, assigned faculty, and list of enrolled students.
    - Courses can be added or modified by administrators.
    - All course-related operations (uploads, downloads, grades, attendance) are linked through validated course IDs.

## Contents

**Introduction**

**Overall Description**

**Specific Requirements**

# 1.Introduction

The purpose of this document is to outline the software requirements for the Course Management System (CMS). The CMS aims to streamline and automate academic course-related activities for students, faculty, and administrators, including course enrollment, assignment submission, material distribution, attendance management, and report generation.

## 1.1 Purpose

The software should enable efficient management of all the courses a student has opted for online. The students should be able to view their attendance,view and download course material, check their grades and upload submissions. A faculty can upload their assignments, view grades and accept submissions with the help of the system

## 1.2 Scope

This system will support academic institutions in managing student and faculty activities effectively over a secure LAN network. Key functionalities include:

- User authentication and role-based access
- Uploading and managing course materials and assignments
- Student submissions and faculty grading
- Attendance management
- Performance calculations
- Viewing and managing course, student, and faculty details
- Report generation for analytics and record keeping

## 1.3 Definitions and Acronyms

- CMS: Course Management System
- UI: User Interface

- LAN: Local Area Network
- SRS: Software Requirements Specification
- DB: Database

### 1.4 References

(a) Software Engineering by K.K. Aggarwal & Yogesh Singh

(b) Software Engineering by Ruchika Malhotra & Yogesh Singh

(c) IEEE Recommended Practice for Software Requirements Specifications—IEEE Std. 830-1998.

## 2. Overall Description

### 2.1 Product Perspective

The CMS is a standalone application deployed on an institution's internal LAN. It interacts with a central database and provides role-based interfaces for students, faculty, and administrators. The system is designed to be scalable and adaptable to different educational institutions.

### 2.2 System Interfaces

- DBMS for persistent storage (e.g., MySQL/PostgreSQL)
- Authentication service (for login/password validation)
- Local institutional servers for deployment

### 2.3 User Interfaces

- Student Dashboard: View courses, submissions, grades, attendance, and download materials
- Faculty Dashboard: Upload assignments/materials, manage attendance, grade submissions
- Admin Dashboard: Manage users, courses, and reports
  All interfaces are web-based with responsive design for accessibility across devices.

### 2.4 Hardware Interfaces

- Server: Intel i5/i7 processor, 16 GB RAM, 1 TB HDD
- Clients: Any modern device with a browser (PC, tablet, etc.)

### 2.5 Software Interfaces

- Backend: Python
- Database: MySQL
- Frontend: HTML5, CSS, JS

### 2.6 Communication Interfaces

- System runs on LAN; communication over TCP/IP
- Secure transmission via HTTPS for client-server interaction

### 2.7 Memory Constraints

- Server should support at least 100 concurrent users with 16 GB RAM
- Disk space requirements scale with course material and user count

### 2.8 Operations

- System backup daily at midnight

- Maintenance window scheduled weekly for updates
- Logs maintained for user activities

### 2.9 Site Adaptation Requirements

- System adaptable to other LAN environments with minimal configuration
- Site-specific data like institution name, logo, etc., configurable via admin panel

### 2.10 Product Functions

The Course Management System will support the following key functions:

- **Authentication:** Secure login for students, faculty, and administrators.
- **User Management:** Add, update, and delete user profiles.
- **Course Management:** Create and maintain course details.
- **Material Upload:** Faculty can upload lecture notes and other materials.
- **Assignment Management:** Upload assignments, accept submissions, and assign grades.
- **Attendance Management:** Faculty can update and students can view attendance.
- **Report Generation:** Generate detailed reports for admin and faculty on student enrollment, attendance, and submissions.
- **Peer Networking:** Students can view basic classmate details to foster communication.

### 2.11 User Characteristics

- **Students:** Basic computer and internet usage skills. Will primarily use the system to view/download materials, submit assignments, and track academic progress.
- **Faculty:** Moderate technical knowledge. Will upload content, grade submissions, manage attendance, and view reports.
- **Administrators:** Technically proficient users responsible for managing users, courses, and generating institutional-level reports.

### 2.12 Constraints

- The system must operate over a secure institutional LAN.
- Users must not upload more than three files per assignment submission.
- All user passwords must be stored in encrypted form.
- The system should be operable only on devices with an internet browser.
- Limited to English language UI in the first release.

## 3. Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

- Role-based dashboards

- Form-based inputs with validations

- Report generation interface with filters

#### 3.1.2 Hardware Interfaces

- Compatible with standard server and client hardware

### 3.1.3 Software Interfaces

- Compatible with standard DBMS

### 3.1.4 Communication Interfaces

- All modules interact through secure internal APIs
- Encrypted data transmission over LAN

### 3.2 Functional Requirements

1. Authentication

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to login to the system |
| **Actors:** Student/faculty/Admin |
| **Preconditions:** The user must have valid login ID and password |
| **Postconditions:** if the use case is successful, the user will be able to login and use the system. |
| **Event Flow**<br>**Basic Flow**<br>1. The user enters valid Login Id<br>2. The user enters valid password<br>3. The user confirms login |
| **Alternate flows:**<br>Alternative Flow 1: Unauthorized user<br>If the system does not validate the login details, then the user is prompted to retry.<br>Alternative Flow 2: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** None |

2. Download course material

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to download course material |
| **Actors:** Student |
| **Preconditions:** The student must be logged onto the system before the use case begins |
| **Postconditions:** if the use case is successful, the student will be able to view their course registration details, otherwise the system remains unchanged. |
| **Event Flow**<br>**Basic Flow** |

| |
|---|
| 1. The student selects the option to download course material for the specified course.<br>2. The course material up till the date of the last lecture uploaded by the faculty is displayed. |
| **Alternate flows:**<br>Alternative Flow 1: Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 2: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

3. Upload course material

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to upload course material |
| **Actors:** Faculty |
| **Preconditions:** The faculty must be logged onto the system before the use case begins |
| **Postconditions:** If the use case is successful, the faculty will be able to upload their respective course material to the system for students to view, else the system remains unchanged |
| **Event Flow**<br>**Basic Flow**<br>1. The faculty selects the option to upload assignments for the specified course.<br>2. The interface allows the faculty to browse contents of the desktop to upload files.<br>3. The faculty selects the file to be submitted and uploads the file. The faculty also sets the deadline for assignment submission.<br>4. The faculty selects the “Upload” option.<br>5. The database of course material is updated.<br>6. The faculty still has the choice to undo the material upload. |
| **Alternate flows:**<br>**Alternate flow 1:** File uploaded is too large<br>The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.<br>Alternative Flow 2: Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 3: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

4. Make submissions

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to make submissions |

**Actors:** Student

**Preconditions:** The student must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the student will be able to submit assignments which will be saved to the system, otherwise the system remains unchanged.

**Event Flow**
**Basic Flow**

1. The student selects the option to make submissions material for the specified course.
2. The interface allows the student to browse contents of the desktop to upload files.
3. The student selects the file to be submitted and uploads the file.
4. The student selects the "Submit" option.
5. The database of submissions is updated.
6. The student still has the choice to undo the submission, till the deadline.

**Alternate flows:**
**Alternate flow 1:** File uploaded is too large
The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.

**Alternate flow 2:** The student tries to submit after the deadline
The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.

**Alternate flow 3:** Invalid courseID
If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.

**Special requirements:** None

**Associated use cases:** Login

5. Accept and grade submission

**Introduction:** This use case documents the steps that must be followed in order to accept and grade submissions

**Actors:** Faculty

**Preconditions:** The faculty must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the faculty will be able to view and edit the assignments submitted by the students to the system and also grade them.

**Event Flow**
**Basic Flow**

1. The faculty selects the option to accept and grade submissions
2. List of students and their submissions are displayed. In a separate list, the faculty can also see the set of defaulters.
3. The faculty can open the files (submissions) and enter the grade for each student.
4. The database of grades is updated

**Alternate flows:**
Alternative Flow 1: Invalid courseID

If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 2: Invalid studentID
If the system does not validate the studentID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 3: User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

6. Upload assignments

**Introduction:** This use case documents the steps that must be followed in order to upload assignments

**Actors:** Faculty

**Preconditions:** The faculty must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the faculty will be able to upload assignments to the system for students to view, else the system remains unchanged

**Event Flow**
**Basic Flow**

1. The faculty selects the option to upload assignments for the specified course.
2. The interface allows the faculty to browse contents of the desktop to upload files.
3. The faculty selects the file to be submitted and uploads the file.
4. The faculty selects the “Upload” option.
5. The database of submissions is updated.
6. The faculty still has the choice to undo the assignment upload, till the deadline.

**Alternate flows:**
**Alternate flow 1:** File uploaded is too large
The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.
Alternative Flow 1: Invalid courseID
If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 2: User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

7. Calculate average performance

**Introduction:** This use case documents the steps that must be followed in order to calculate average performance

**Actors:** Faculty

**Preconditions:** The faculty must be logged onto the system before the use case begins

| |
|---|
| **Postconditions:** If the use case is successful, the faculty will be able to update attendance and save it to the database, else the system remains unchanged |
| **Event Flow**<br>**Basic Flow**<br>1. The faculty selects the option to calculate average performance of the class<br>2. The average performance of the class is calculated and displayed to the faculty. |
| **Alternate flows:**<br>**Alternate flow 1:** No grades have been uploaded<br>Since grades have not been uploaded, average performance cannot be calculated. Zero value is returned.<br>Alternative Flow 2:Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 3: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

8. View attendance

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to view attendance |
| **Actors:** Student |
| **Preconditions:** The student must be logged onto the system before the use case begins |
| **Postconditions:** if the use case is successful, the student will be able to view their attendance details, otherwise the system remains unchanged. |
| **Event Flow**<br>**Basic Flow**<br>1. The student selects the option to view their attendance for the specified course.<br>2. The attendance of the student up till the date of the last lecture is displayed. |
| **Alternate flows:**<br>Alternative Flow 1: Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 2: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

9. View classmate details

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to view classmate details |

| |
|---|
| **Actors:** Student |
| **Preconditions:** The student must be logged onto the system before the use case begins |
| **Postconditions:** if the use case is successful, the student will be able to view the contact information of their classmates, otherwise the system remains unchanged. |
| **Event Flow**<br>**Basic Flow**<br>1. The student selects the option to view classmate details for the specific course(such as college email ID).<br>2. The classmate details uploaded by the administrator are displayed. |
| **Alternate flows:**<br>Alternative Flow 1: Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 2: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

10. Generate reports

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to generate reports |
| **Actors:** Administrator |
| **Preconditions:** The administrator must be logged onto the system before the use case begins |
| **Postconditions:** If the use case is successful, the admin will be able to generate reports, else the system remains unchanged |
| **Event Flow**<br>**Basic Flow:**<br>1. The admin selects the option to generate reports.<br>2. The portal displays several choices. To generate reports of attendance/grades/submissions/student details/faculty details |
| **Alternate flows:**<br>Alternative Flow 1: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

11. Maintain student details

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to maintain student details. |
| **Actors:** Administrator |

**Preconditions:** The administrator must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the admin will be able to update the details of students save it to the database, else the system remains unchanged

**Basic Flow** This use case starts when the administrator wishes to add, update, delete, or view student information from the system.

1. The system requests that the administrator specify the function he/she would like to perform (either add a student, update a student record, delete a student record, or view a student record).
2. Once the administrator provides the requested information, one of the subflows is executed.

**Basic Flow 1: Add a Student** The system requests that the administrator enter the user information. This includes:

- Student ID
- Name
- Class
- Section
- Phone
- Address
- Mother’s Name
- Father’s Name
- Email

Once the administrator provides the requested information, the system checks that the student ID is unique. The student is added to the system.

**Basic Flow 2: Update a Student**

1. The system requests that the administrator enter the student’s ID.
2. The administrator enters the student’s ID.
3. The system retrieves and displays the student’s information.
4. The administrator makes the desired changes to the student information. This includes any of the details specified in the **Add a Student** subflow.
5. Once the administrator updates the necessary information, the system updates the student record with the new details.

**Basic Flow 3: Delete a Student**

1. The system requests that the administrator specify the student ID of the student to be deleted.
2. The administrator enters the student ID. The system retrieves and displays the student information.
3. The system prompts the administrator to confirm the deletion of the student record.
4. The administrator verifies the deletion.
5. The system deletes the record.

**Basic Flow 4: View a Student**

1. The system requests that the administrator specify the student ID.
2. The system retrieves and displays the student information.

**Alternate flows:**

**Alternative Flow 1: Invalid Entry**

If in the **Add a Student** or **Update a Student** flow, the administrator enters invalid **Student ID/Name/Class/Section/Phone/Address/Mother's Name/Father's Name/Email** or leaves any required field empty, the system displays an appropriate error message. The administrator returns to the basic flow and may reenter the invalid entry.

**Alternative Flow 2: Student Already Exists**

If in the **Add a Student** flow, a student with the specified student ID already exists, the system displays an error message. The administrator returns to the basic flow and may reenter the student information.

**Alternative Flow 3: Student Not Found**

If in the **Update a Student**, **Delete a Student**, or **View a Student** flow, the student information with the specified student ID does not exist, the system displays an error message. The administrator returns to the basic flow and may reenter the student ID.

**Alternative Flow 4: User exits**
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

12. Maintain faculty details

**Introduction:** This use case documents the steps that must be followed in order to maintain faculty details.

**Actors:** Administrator

**Preconditions:** The administrator must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the admin will be able to update the details of students save it to the database, else the system remains unchanged

**Basic Flow** This use case starts when the administrator wishes to add, update, delete, or view faculty information from the system.

1. The system requests that the administrator specify the function he/she would like to perform (either add a faculty member, update a faculty record, delete a faculty record, or view a faculty record).
2. Once the administrator provides the requested information, one of the subflows is executed.
    - If the administrator selects **"Add Faculty"**, the **Add Faculty** subflow is executed.
    - If the administrator selects **"Update Faculty"**, the **Update Faculty** subflow is executed.
    - If the administrator selects **"Delete Faculty"**, the **Delete Faculty** subflow is executed.
    - If the administrator selects **"View Faculty"**, the **View Faculty** subflow is executed.

**Basic Flow 1: Add Faculty** The system requests that the administrator enter faculty information. This includes:

- Faculty ID
- Name
- Department
- Designation
- Phone
- Address
- Email

Once the administrator provides the requested information, the system checks that the faculty ID is unique. The faculty member is added to the system.

**Basic Flow 2: Update Faculty**

1. The system requests that the administrator enter the faculty ID.
2. The administrator enters the faculty ID.
3. The system retrieves and displays the faculty information.
4. The administrator makes the desired changes to the faculty information. This includes any of the details specified in the **Add Faculty** subflow.
5. Once the administrator updates the necessary information, the system updates the faculty record with the new details.

**Basic Flow 3: Delete Faculty**

1. The system requests that the administrator specify the faculty ID of the faculty member to be deleted.
2. The administrator enters the faculty ID. The system retrieves and displays the faculty information.
3. The system prompts the administrator to confirm the deletion of the faculty record.
4. The administrator verifies the deletion.
5. The system deletes the record.

**Basic Flow 4: View Faculty**

1. The system requests that the administrator specify the faculty ID.
2. The system retrieves and displays the faculty information.

**Alternative Flow 1: Invalid Entry**

If in the **Add Faculty** or **Update Faculty** flow, the administrator enters invalid **Faculty ID/Name/Department/Designation/Phone/Address/Email** or leaves any required field empty, the system displays an appropriate error message. The administrator returns to the basic flow and may reenter the invalid entry.

**Alternative Flow 2: Faculty Already Exists**

If in the **Add Faculty** flow, a faculty member with the specified faculty ID already exists, the system displays an error message. The administrator returns to the basic flow and may reenter the faculty information.

**Alternative Flow 3: Faculty Not Found**

If in the **Update Faculty**, **Delete Faculty**, or **View Faculty** flow, the faculty information with the specified faculty ID does not exist, the system displays an error message. The administrator returns to the basic flow and may reenter the faculty ID.

**Alternative Flow 4: User exits**
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

13. Maintain course details

**Introduction:** This use case documents the steps that must be followed in order to maintain course details.

**Actors:** Administrator

**Preconditions:** The administrator must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the admin will be able to update the details of students save it to the database, else the system remains unchanged

**Basic Flow** This use case starts when the administrator wishes to add, update, delete, or view course information from the system.

1. The system requests that the administrator specify the function he/she would like to perform (either add a course, update a course record, delete a course record, or view a course record).
2. Once the administrator provides the requested information, one of the subflows is executed.
   - If the administrator selects **"Add Course"**, the **Add Course** subflow is executed.
   - If the administrator selects **"Update Course"**, the **Update Course** subflow is executed.
   - If the administrator selects **"Delete Course"**, the **Delete Course** subflow is executed.
   - If the administrator selects **"View Course"**, the **View Course** subflow is executed.

**Basic Flow 1: Add Course** The system requests that the administrator enter course information. This includes:

- Course ID
- Course Name
- Department
- Credits
- Instructor
- Schedule

Once the administrator provides the requested information, the system checks that the course ID is unique. The course is added to the system.

**Basic Flow 2: Update Course**

1. The system requests that the administrator enter the course ID.
2. The administrator enters the course ID.
3. The system retrieves and displays the course information.
4. The administrator makes the desired changes to the course information. This includes any of the

|  |
|---|
| details specified in the **Add Course** subflow.<br>5. Once the administrator updates the necessary information, the system updates the course record with the new details.<br><br>**Basic Flow 3: Delete Course**<br><br>1. The system requests that the administrator specify the course ID of the course to be deleted.<br>2. The administrator enters the course ID. The system retrieves and displays the course information.<br>3. The system prompts the administrator to confirm the deletion of the course record.<br>4. The administrator verifies the deletion.<br>5. The system deletes the record.<br><br>**Basic Flow 4: View Course**<br><br>1. The system requests that the administrator specify the course ID.<br>2. The system retrieves and displays the course information. |
| **Alternative Flow 1: Invalid Entry** If in the **Add Course** or **Update Course** flow, the administrator enters invalid **Course ID/Course Name/Department/Credits/Instructor/Schedule** or leaves any required field empty, the system displays an appropriate error message. The administrator returns to the basic flow and may reenter the invalid entry.<br><br>**Alternative Flow 2: Course Already Exists** If in the **Add Course** flow, a course with the specified course ID already exists, the system displays an error message. The administrator returns to the basic flow and may reenter the course information.<br><br>**Alternative Flow 3: Course Not Found** If in the **Update Course**, **Delete Course**, or **View Course** flow, the course information with the specified course ID does not exist, the system displays an error message. The administrator returns to the basic flow and may reenter the course ID.<br><br>**Alternative Flow 4: User exits**<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

### 3.3 Performance Requirements

- The system should handle up to 1000 concurrent users efficiently
- Attendance and performance reports should generate in less than 5 seconds

### 3.4 Design Constraints

- Must support modular and scalable architecture
- System must be browser compatible (Chrome, Firefox, Edge)

### 3.5 Software System Attributes

#### 3.5.1 Reliability

- 99.9% uptime during working hours
- Fault-tolerant design for critical operations

### 3.5.2 Availability

- System available on LAN 24/7 except during maintenance

### 3.5.3 Security

- Role-based access
- Password encryption
- SQL injection prevention and input sanitization

### 3.5.4 Maintainability

- Modular codebase with inline documentation
- Admin panel for configuration and logs

### 3.5.5 Portability

- Deployable on any Linux/Windows server
- Frontend accessible on all standard browsers

## 3.6 Logical Database Requirements

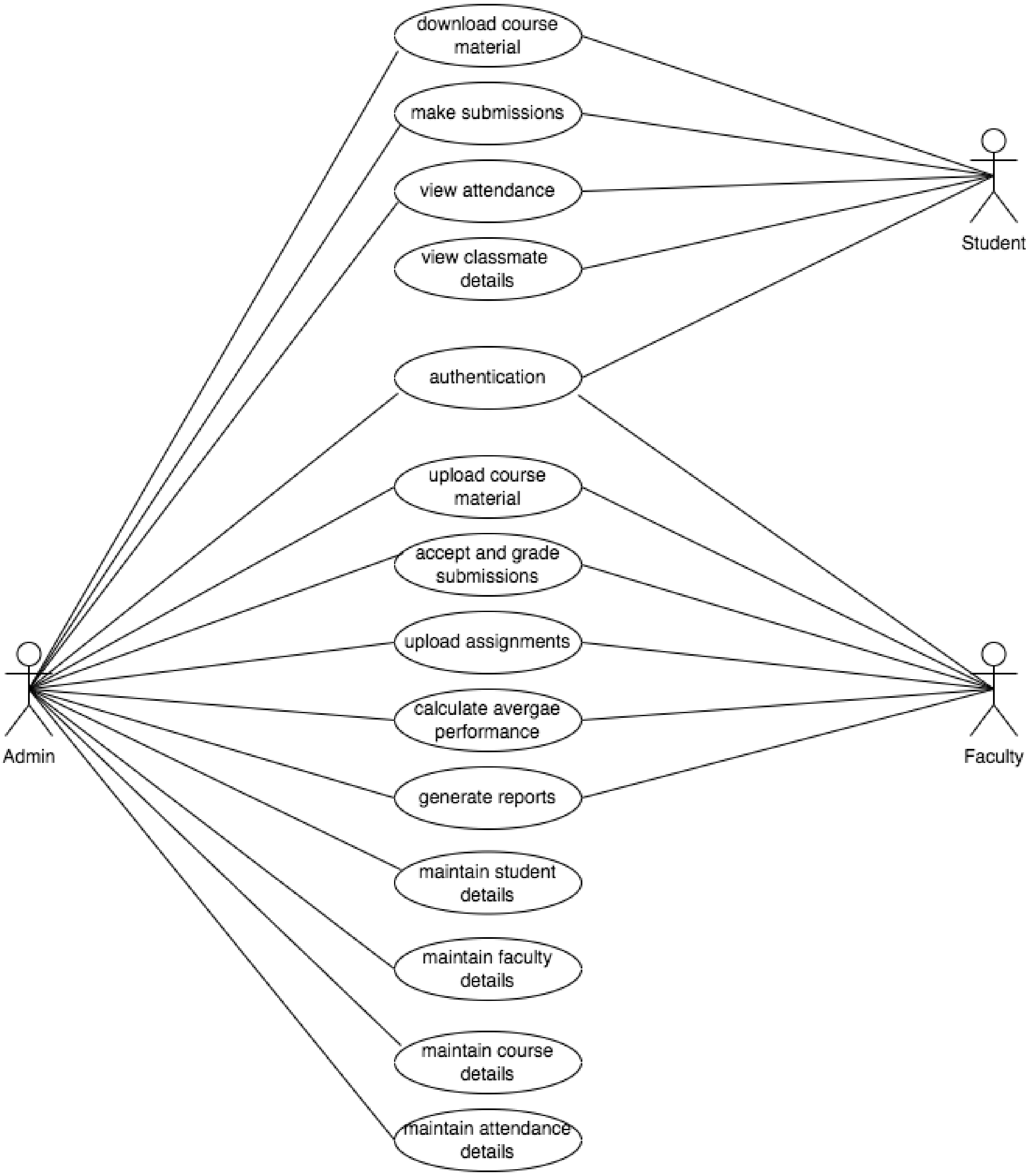
- User table (students, faculty, admins)
- Courses, Assignments, Submissions, Grades
- Attendance records
- Relational integrity with foreign key constraints

## 3.7 Other Requirements

- Maximum file upload per submission: 3 files
- Submissions must be timestamped

# EXPERIMENT-4

AIM: Draft the Use Case Diagram for the Course Management System



# EXPERIMENT-5

AIM: Draft the Use Case Descriptions for the Course Management System

**Introduction:** This use case documents the briefly introduces the use case of a Course Management System

**Actors:** Administrator, faculty,student

**Preconditions:** The administrator/student/faculty must be logged onto the system before the use case begins

**Postconditions: I**f the use cases are successful, the student/admin/faculty will be able to view and edit the details in the system, otherwise the system remains unchanged.

**Event Flow**

**Basic Flow**

1. The sub flow use case for authentication is used.
2. The sub flow use case for downloading course material is used.
3. The sub flow use case for uploading course material is used.
4. The sub flow use case for making submissions is used.
5. The sub flow use case for accepting and grading submissions is used.
6. The sub flow use case for uploading assignments is used.
7. The sub flow use case for calculating average performance is used.
8. The sub flow use case for viewing attendance is used.
9. The sub flow use case for viewing classmate details is used.
10. The sub flow use case for generating reports is used.
11. The sub flow use case for maintaining attendance details is used
12. The sub flow use case for maintaining student details is used
13. The sub flow use case for maintaining faculty details is used.
14. The sub flow use case for maintaining course details is used.

**Alternate flows:**

**Alternate flow 1:** Invalid login credentials
The use case ends and returns to the beginning of the basic flow.

**Alternate flow 2:** User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to login to the system

**Actors:** Student/faculty/Admin

**Preconditions:** The user must have valid login ID and password

**Postconditions:** if the use case is successful, the user will be able to login and use the system.

**Event Flow**

**Basic Flow**

1. The user enters valid Login Id
2. The user enters valid password
3. The user confirms login

**Alternate flows:**

Alternative Flow 1: Unauthorized user

If the system does not validate the login details, then the user is prompted to retry.

Alternative Flow 2: User exits

This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** None

**Introduction:** This use case documents the steps that must be followed in order to download course material

**Actors:** Student

**Preconditions:** The student must be logged onto the system before the use case begins

**Postconditions:** if the use case is successful, the student will be able to view their course registration details, otherwise the system remains unchanged.

**Event Flow**

**Basic Flow**

1. The student selects the option to download course material for the specified course.
2. The course material up till the date of the last lecture uploaded by the faculty is displayed.

**Alternate flows:**

Alternative Flow 1: Invalid courseID

If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.

Alternative Flow 2: User exits

This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to upload course material

**Actors:** Faculty

**Preconditions:** The faculty must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the faculty will be able to upload their respective course material to the system for students to view, else the system remains unchanged

**Event Flow**
**Basic Flow**

1. The faculty selects the option to upload assignments for the specified course.
2. The interface allows the faculty to browse contents of the desktop to upload files.
3. The faculty selects the file to be submitted and uploads the file. The faculty also sets the deadline for assignment submission.
4. The faculty selects the “Upload” option.
5. The database of course material is updated.
6. The faculty still has the choice to undo the material upload.

**Alternate flows:**
**Alternate flow 1:** File uploaded is too large
The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.
Alternative Flow 2: Invalid courseID
If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 3: User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to make submissions

**Actors:** Student

**Preconditions:** The student must be logged onto the system before the use case begins

| |
|---|
| **Postconditions:** If the use case is successful, the student will be able to submit assignments which will be saved to the system, otherwise the system remains unchanged. |
| **Event Flow**<br>**Basic Flow**<br>1. The student selects the option to make submissions material for the specified course.<br>2. The interface allows the student to browse contents of the desktop to upload files.<br>3. The student selects the file to be submitted and uploads the file.<br>4. The student selects the “Submit” option.<br>5. The database of submissions is updated.<br>6. The student still has the choice to undo the submission, till the deadline. |
| **Alternate flows:**<br>**Alternate flow 1:** File uploaded is too large<br>The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.<br><br>**Alternate flow 2:** The student tries to submit after the deadline<br>The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.<br><br>**Alternate flow 3:** Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow. |
| **Special requirements:** None |
| **Associated use cases:** Login |

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to accept and grade submissions |
| **Actors:** Faculty |
| **Preconditions:** The faculty must be logged onto the system before the use case begins |
| **Postconditions:** If the use case is successful, the faculty will be able to view and edit the assignments submitted by the students to the system and also grade them. |
| **Event Flow**<br>**Basic Flow**<br>1. The faculty selects the option to accept and grade submissions<br>2. List of students and their submissions are displayed. In a separate list, the faculty can also see the set of defaulters.<br>3. The faculty can open the files (submissions) and enter the grade for each student.<br>4. The database of grades is updated |
| **Alternate flows:** |

Alternative Flow 1: Invalid courseID
If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 2: Invalid studentID
If the system does not validate the studentID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 3: User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to upload assignments

**Actors:** Faculty

**Preconditions:** The faculty must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the faculty will be able to upload assignments to the system for students to view, else the system remains unchanged

**Event Flow**
**Basic Flow**

1. The faculty selects the option to upload assignments for the specified course.
2. The interface allows the faculty to browse contents of the desktop to upload files.
3. The faculty selects the file to be submitted and uploads the file.
4. The faculty selects the “Upload” option.
5. The database of submissions is updated.
6. The faculty still has the choice to undo the assignment upload, till the deadline.

**Alternate flows:**
**Alternate flow 1:** File uploaded is too large
The upload is unsuccessful. Error message is displayed. Use case ends and returns to the beginning of the basic flow.
Alternative Flow 1: Invalid courseID
If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.
Alternative Flow 2: User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to

| |
|---|
| calculate average performance |
| **Actors:** Faculty |
| **Preconditions:** The faculty must be logged onto the system before the use case begins |
| **Postconditions:** If the use case is successful, the faculty will be able to update attendance and save it to the database, else the system remains unchanged |
| **Event Flow**<br>**Basic Flow**<br>1. The faculty selects the option to calculate average performance of the class<br>2. The average performance of the class is calculated and displayed to the faculty. |
| **Alternate flows:**<br>**Alternate flow 1:** No grades have been uploaded<br>Since grades have not been uploaded, average performance cannot be calculated. Zero value is returned.<br>Alternative Flow 2:Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 3: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to view attendance |
| **Actors:** Student |
| **Preconditions:** The student must be logged onto the system before the use case begins |
| **Postconditions:** if the use case is successful, the student will be able to view their attendance details, otherwise the system remains unchanged. |
| **Event Flow**<br>**Basic Flow**<br>1. The student selects the option to view their attendance for the specified course.<br>2. The attendance of the student up till the date of the last lecture is displayed. |
| **Alternate flows:**<br>Alternative Flow 1: Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 2: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |

| |
|---|
| **Special requirements:** None |
| **Associated use cases:** Login |

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to view classmate details |
| **Actors:** Student |
| **Preconditions:** The student must be logged onto the system before the use case begins |
| **Postconditions:** if the use case is successful, the student will be able to view the contact information of their classmates, otherwise the system remains unchanged. |
| **Event Flow**<br>**Basic Flow**<br>1. The student selects the option to view classmate details for the specific course(such as college email ID).<br>2. The classmate details uploaded by the administrator are displayed. |
| **Alternate flows:**<br>Alternative Flow 1: Invalid courseID<br>If the system does not validate the courseID, then an error message is flagged and the use case returns to the beginning of the basic flow.<br>Alternative Flow 2: User exits<br>This allows the user to exit at any time during the use case. The use case ends. |
| **Special requirements:** None |
| **Associated use cases:** Login |

| |
|---|
| **Introduction:** This use case documents the steps that must be followed in order to generate reports |
| **Actors:** Administrator |
| **Preconditions:** The administrator must be logged onto the system before the use case begins |
| **Postconditions:** If the use case is successful, the admin will be able to generate reports, else the system remains unchanged |
| **Event Flow**<br>**Basic Flow:**<br>1. The admin selects the option to generate reports.<br>2. The portal displays several choices. To generate reports of |

attendance/grades/submissions/student details/faculty details

**Alternate flows:**
Alternative Flow 1: User exits
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to maintain student details.

**Actors:** Administrator

**Preconditions:** The administrator must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the admin will be able to update the details of students save it to the database, else the system remains unchanged

**Basic Flow** This use case starts when the administrator wishes to add, update, delete, or view student information from the system.

1. The system requests that the administrator specify the function he/she would like to perform (either add a student, update a student record, delete a student record, or view a student record).
2. Once the administrator provides the requested information, one of the subflows is executed.

**Basic Flow 1: Add a Student** The system requests that the administrator enter the user information. This includes:

- Student ID
- Name
- Class
- Section
- Phone
- Address
- Mother’s Name
- Father’s Name
- Email

Once the administrator provides the requested information, the system checks that the student ID is unique. The student is added to the system.

**Basic Flow 2: Update a Student**

1. The system requests that the administrator enter the student's ID.
2. The administrator enters the student's ID.
3. The system retrieves and displays the student's information.
4. The administrator makes the desired changes to the student information. This includes any of the details specified in the **Add a Student** subflow.
5. Once the administrator updates the necessary information, the system updates the student record with the new details.

**Basic Flow 3: Delete a Student**

1. The system requests that the administrator specify the student ID of the student to be deleted.
2. The administrator enters the student ID. The system retrieves and displays the student information.
3. The system prompts the administrator to confirm the deletion of the student record.
4. The administrator verifies the deletion.
5. The system deletes the record.

**Basic Flow 4: View a Student**

1. The system requests that the administrator specify the student ID.
2. The system retrieves and displays the student information.

**Alternate flows:**

**Alternative Flow 1: Invalid Entry**

If in the **Add a Student** or **Update a Student** flow, the administrator enters invalid **Student ID/Name/Class/Section/Phone/Address/Mother's Name/Father's Name/Email** or leaves any required field empty, the system displays an appropriate error message. The administrator returns to the basic flow and may reenter the invalid entry.

**Alternative Flow 2: Student Already Exists**

If in the **Add a Student** flow, a student with the specified student ID already exists, the system displays an error message. The administrator returns to the basic flow and may reenter the student information.

**Alternative Flow 3: Student Not Found**

If in the **Update a Student**, **Delete a Student**, or **View a Student** flow, the student information with the specified student ID does not exist, the system displays an error message. The administrator returns to the basic flow and may reenter the student ID.

**Alternative Flow 4: User exits**
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to maintain faculty details.

**Actors:** Administrator

**Preconditions:** The administrator must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the admin will be able to update the details of students save it to the database, else the system remains unchanged

**Basic Flow** This use case starts when the administrator wishes to add, update, delete, or view faculty information from the system.

1. The system requests that the administrator specify the function he/she would like to perform (either add a faculty member, update a faculty record, delete a faculty record, or view a faculty record).
2. Once the administrator provides the requested information, one of the subflows is executed.
   - If the administrator selects **"Add Faculty"**, the **Add Faculty** subflow is executed.
   - If the administrator selects **"Update Faculty"**, the **Update Faculty** subflow is executed.
   - If the administrator selects **"Delete Faculty"**, the **Delete Faculty** subflow is executed.
   - If the administrator selects **"View Faculty"**, the **View Faculty** subflow is executed.

**Basic Flow 1: Add Faculty** The system requests that the administrator enter faculty information. This includes:

- Faculty ID
- Name
- Department
- Designation
- Phone
- Address
- Email

Once the administrator provides the requested information, the system checks that the faculty ID is unique. The faculty member is added to the system.

**Basic Flow 2: Update Faculty**

1. The system requests that the administrator enter the faculty ID.
2. The administrator enters the faculty ID.
3. The system retrieves and displays the faculty information.
4. The administrator makes the desired changes to the faculty information. This includes any of the details specified in the **Add Faculty** subflow.
5. Once the administrator updates the necessary information, the system updates the faculty record with the new details.

**Basic Flow 3: Delete Faculty**

1. The system requests that the administrator specify the faculty ID of the faculty member to be deleted.
2. The administrator enters the faculty ID. The system retrieves and displays the faculty information.
3. The system prompts the administrator to confirm the deletion of the faculty record.
4. The administrator verifies the deletion.
5. The system deletes the record.

**Basic Flow 4: View Faculty**

1. The system requests that the administrator specify the faculty ID.
2. The system retrieves and displays the faculty information.

**Alternative Flow 1: Invalid Entry**

If in the **Add Faculty** or **Update Faculty** flow, the administrator enters invalid **Faculty ID/Name/Department/Designation/Phone/Address/Email** or leaves any required field empty, the system displays an appropriate error message. The administrator returns to the basic flow and may reenter the invalid entry.

**Alternative Flow 2: Faculty Already Exists**

If in the **Add Faculty** flow, a faculty member with the specified faculty ID already exists, the system displays an error message. The administrator returns to the basic flow and may reenter the faculty information.

**Alternative Flow 3: Faculty Not Found**

If in the **Update Faculty**, **Delete Faculty**, or **View Faculty** flow, the faculty information with the specified faculty ID does not exist, the system displays an error message. The administrator returns to the basic flow and may reenter the faculty ID.

**Alternative Flow 4: User exits**
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

**Introduction:** This use case documents the steps that must be followed in order to maintain course details.

**Actors:** Administrator

**Preconditions:** The administrator must be logged onto the system before the use case begins

**Postconditions:** If the use case is successful, the admin will be able to update the details of students save it to the database, else the system remains unchanged

**Basic Flow** This use case starts when the administrator wishes to add, update, delete, or view course information from the system.

1. The system requests that the administrator specify the function he/she would like to perform (either add a course, update a course record, delete a course record, or view a course record).
2. Once the administrator provides the requested information, one of the subflows is executed.
   - If the administrator selects **"Add Course"**, the **Add Course** subflow is executed.
   - If the administrator selects **"Update Course"**, the **Update Course** subflow is executed.
   - If the administrator selects **"Delete Course"**, the **Delete Course** subflow is executed.
   - If the administrator selects **"View Course"**, the **View Course** subflow is executed.

**Basic Flow 1: Add Course** The system requests that the administrator enter course information. This includes:

- Course ID
- Course Name
- Department
- Credits
- Instructor
- Schedule

Once the administrator provides the requested information, the system checks that the course ID is unique. The course is added to the system.

**Basic Flow 2: Update Course**

1. The system requests that the administrator enter the course ID.
2. The administrator enters the course ID.
3. The system retrieves and displays the course information.
4. The administrator makes the desired changes to the course information. This includes any of the details specified in the **Add Course** subflow.

5. Once the administrator updates the necessary information, the system updates the course record with the new details.

**Basic Flow 3: Delete Course**

1. The system requests that the administrator specify the course ID of the course to be deleted.
2. The administrator enters the course ID. The system retrieves and displays the course information.
3. The system prompts the administrator to confirm the deletion of the course record.
4. The administrator verifies the deletion.
5. The system deletes the record.

**Basic Flow 4: View Course**

1. The system requests that the administrator specify the course ID.
2. The system retrieves and displays the course information.

**Alternative Flow 1: Invalid Entry** If in the **Add Course** or **Update Course** flow, the administrator enters invalid **Course ID/Course Name/Department/Credits/Instructor/Schedule** or leaves any required field empty, the system displays an appropriate error message. The administrator returns to the basic flow and may reenter the invalid entry.

**Alternative Flow 2: Course Already Exists** If in the **Add Course** flow, a course with the specified course ID already exists, the system displays an error message. The administrator returns to the basic flow and may reenter the course information.

**Alternative Flow 3: Course Not Found** If in the **Update Course**, **Delete Course**, or **View Course** flow, the course information with the specified course ID does not exist, the system displays an error message. The administrator returns to the basic flow and may reenter the course ID.

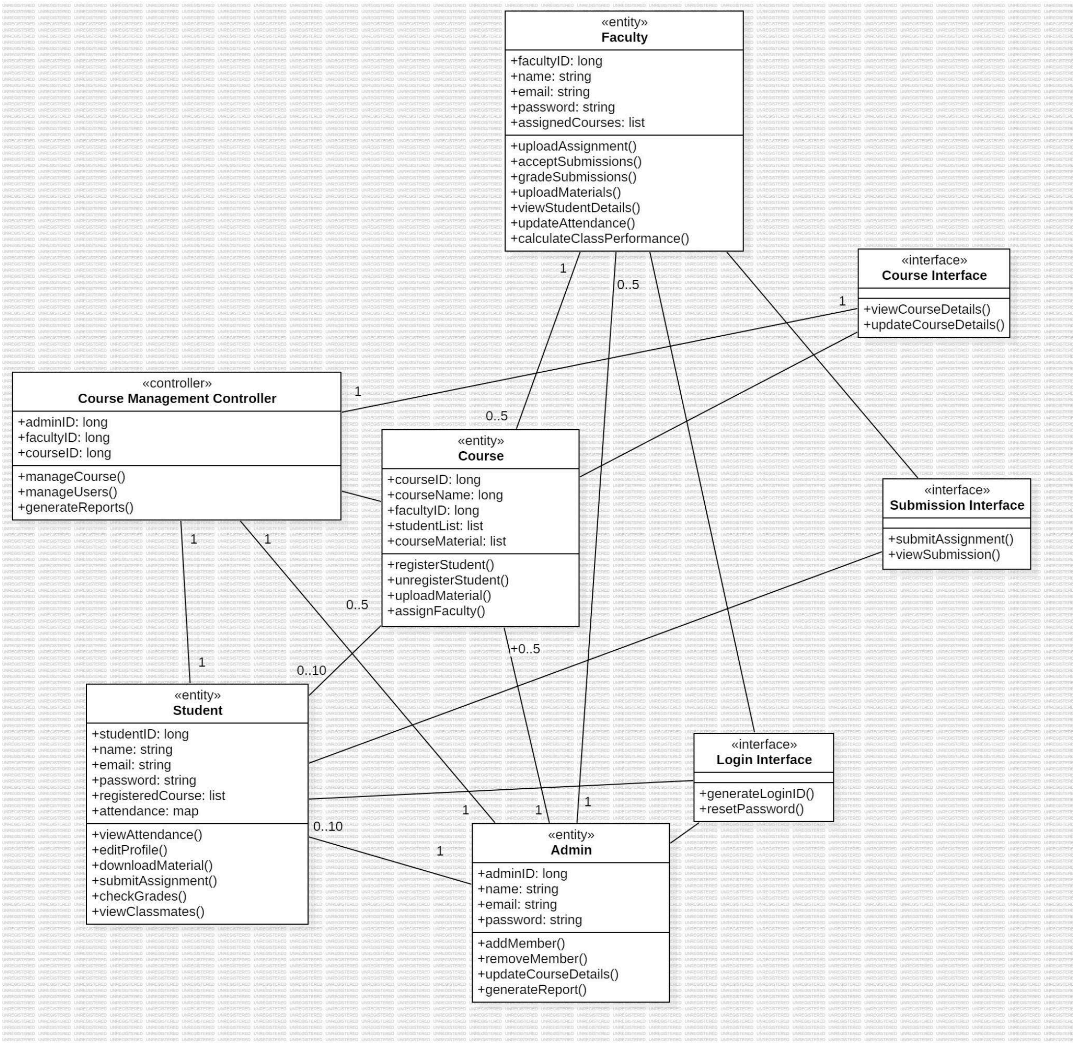
**Alternative Flow 4: User exits**
This allows the user to exit at any time during the use case. The use case ends.

**Special requirements:** None

**Associated use cases:** Login

# EXPERIMENT-6

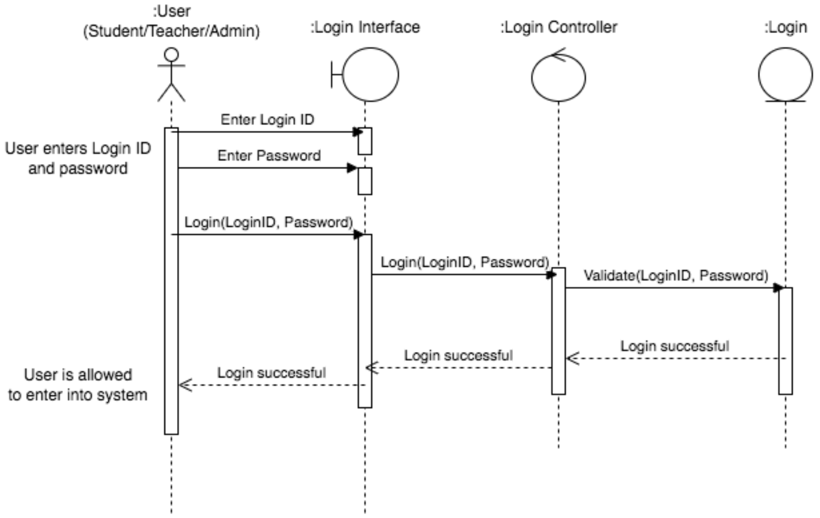
**Aim:** To design the class diagram of course management system

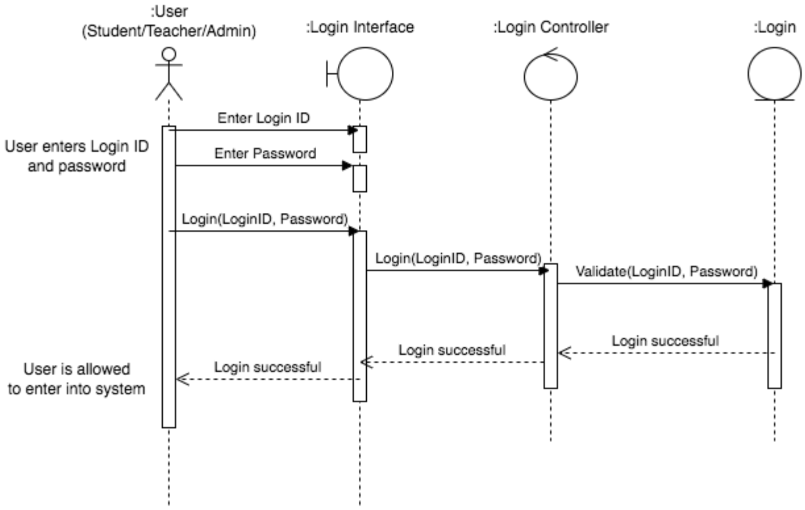


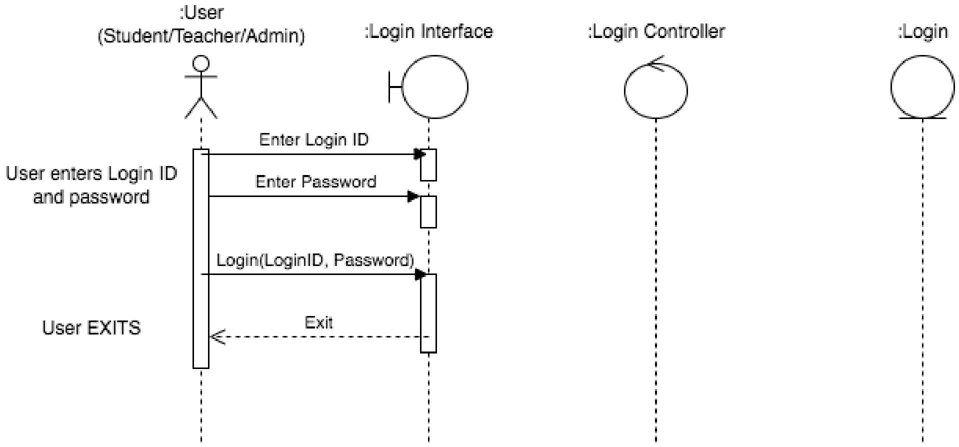
# EXPERIMENT-7

**Aim:** To design the sequence diagrams of course management systems

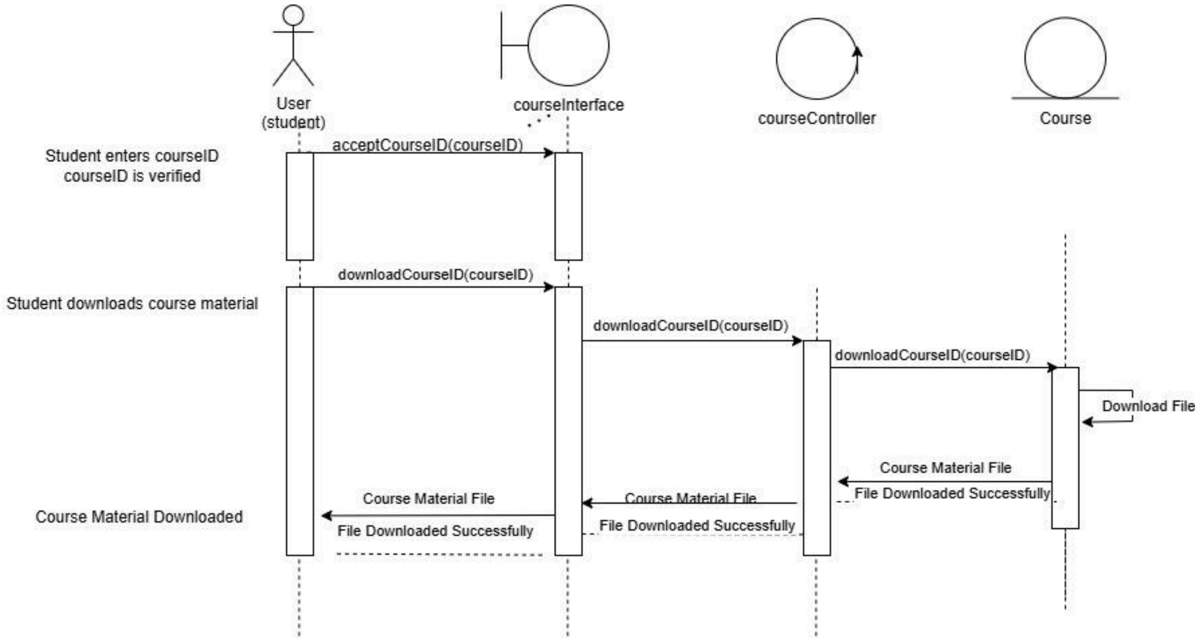
1. Login

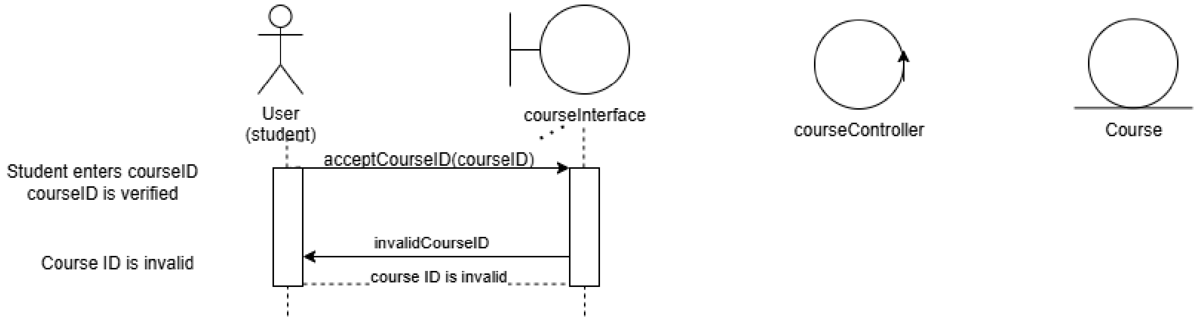




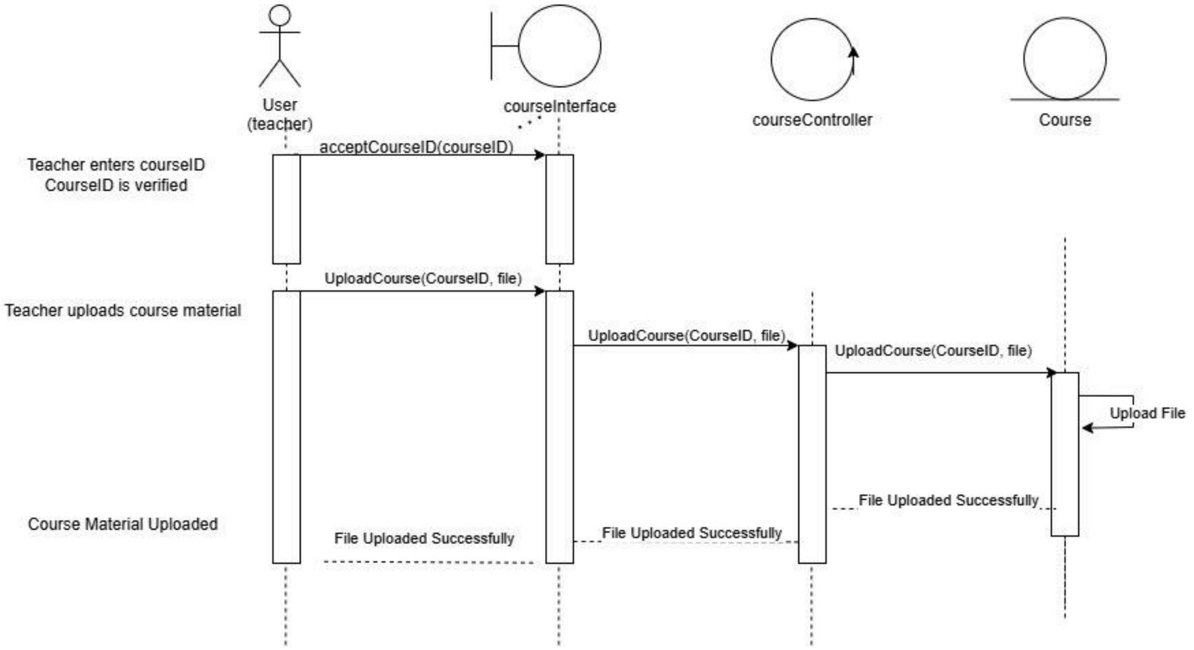


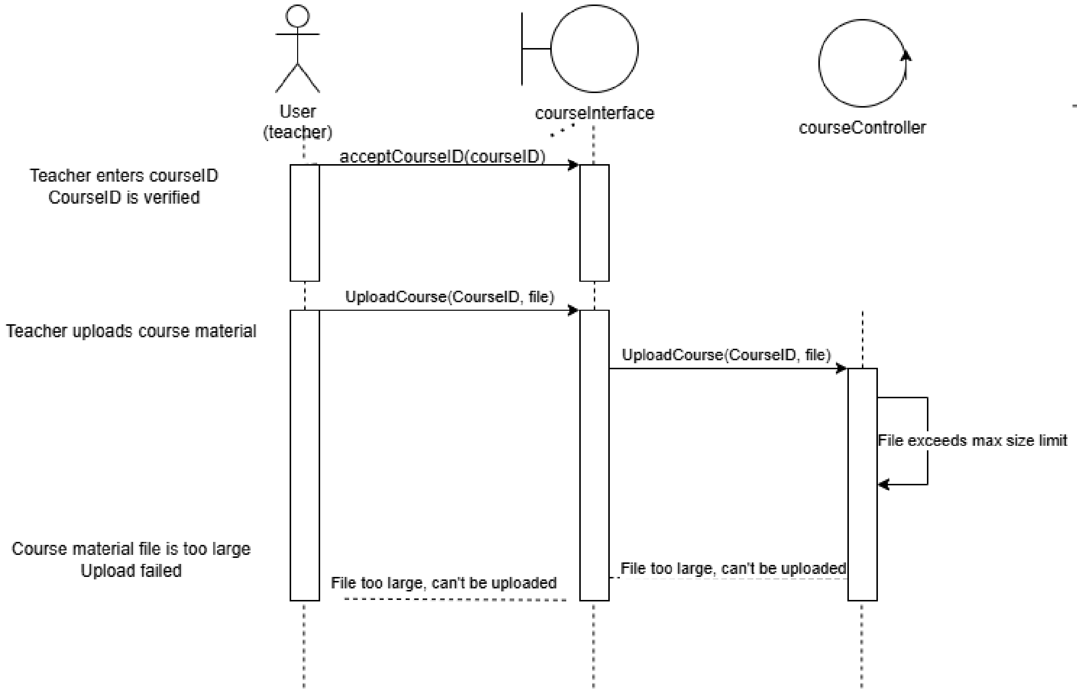
2. Downloading course material

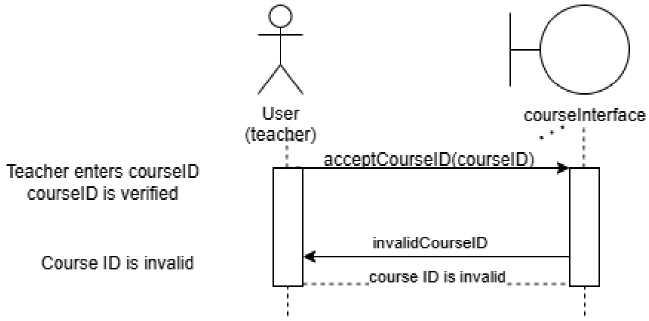

User
(student)
courseInterface
courseController
Course
Student enters courseID
courseID is verified
acceptCourseID(courseID)
Student downloads course material
downloadCourseID(courseID)
downloadCourseID(courseID)
downloadCourseID(courseID)
Download File
Course Material File
File Downloaded Successfully
Course Material File
File Downloaded Successfully
Course Material Downloaded
Course Material File
File Downloaded Successfully



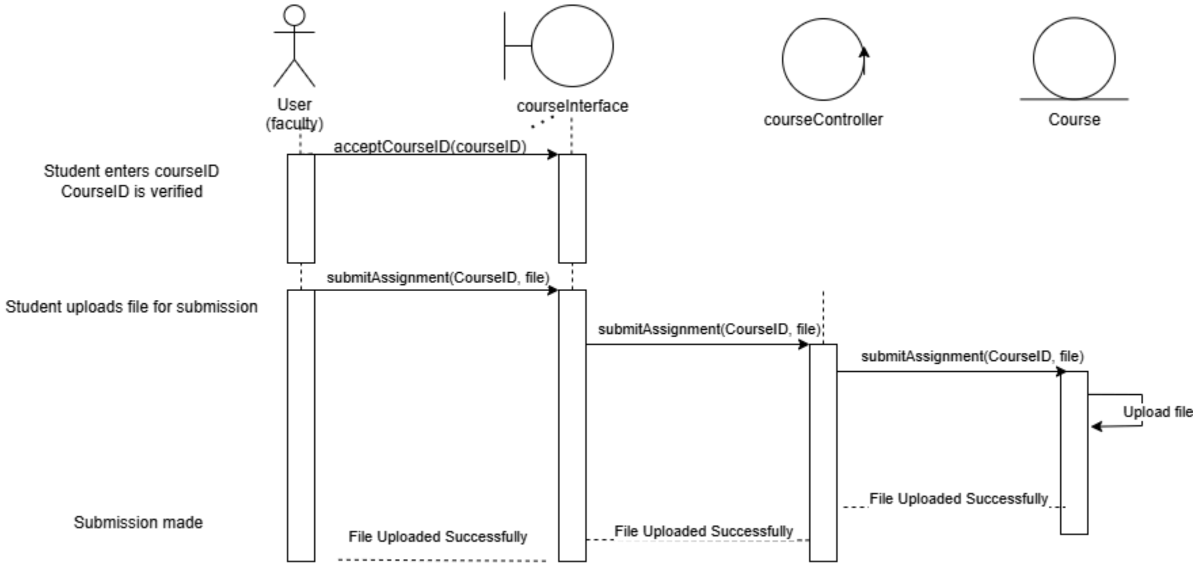
3. Uploading course material

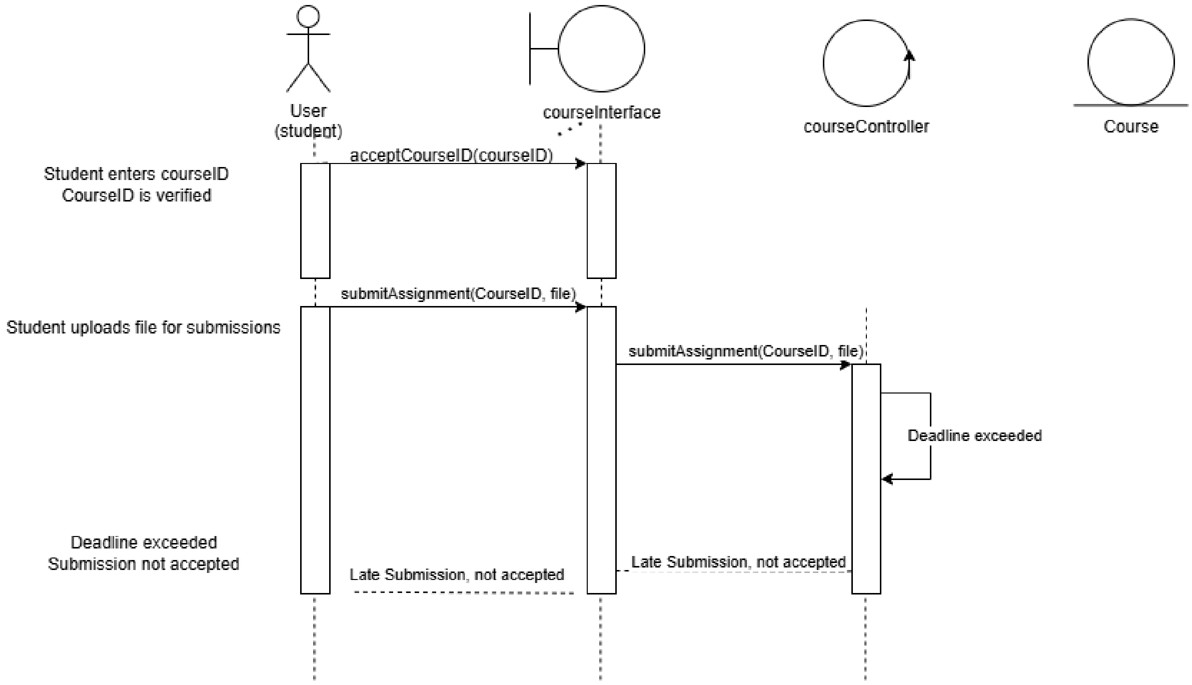

User
(teacher)
courseInterface
courseController
Course
Teacher enters courseID
CourseID is verified
acceptCourseID(courseID)
Teacher uploads course material
UploadCourse(CourseID, file)
UploadCourse(CourseID, file)
UploadCourse(CourseID, file)
Upload File
File Uploaded Successfully
File Uploaded Successfully
Course Material Uploaded
File Uploaded Successfully

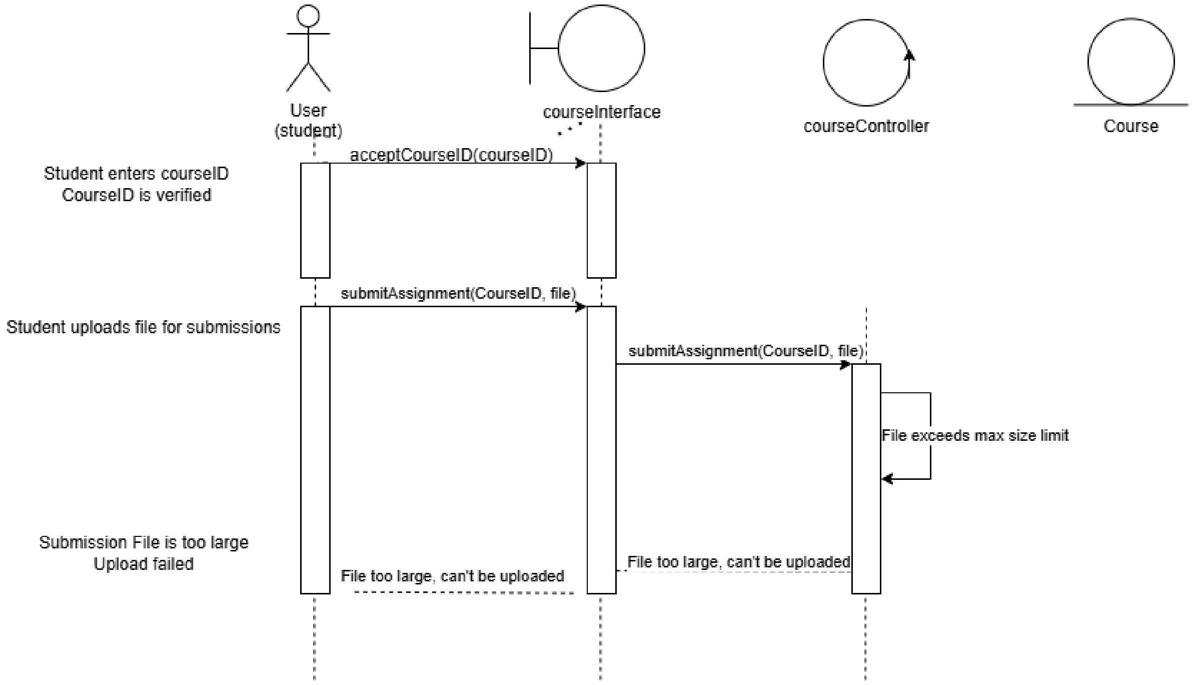

User
(teacher)
courseInterface
courseController
Course
Teacher enters courseID
CourseID is verified
acceptCourseID(courseID)
Teacher uploads course material
UploadCourse(CourseID, file)
UploadCourse(CourseID, file)
File exceeds max size limit
Course material file is too large
Upload failed
File too large, can't be uploaded
File too large, can't be uploaded

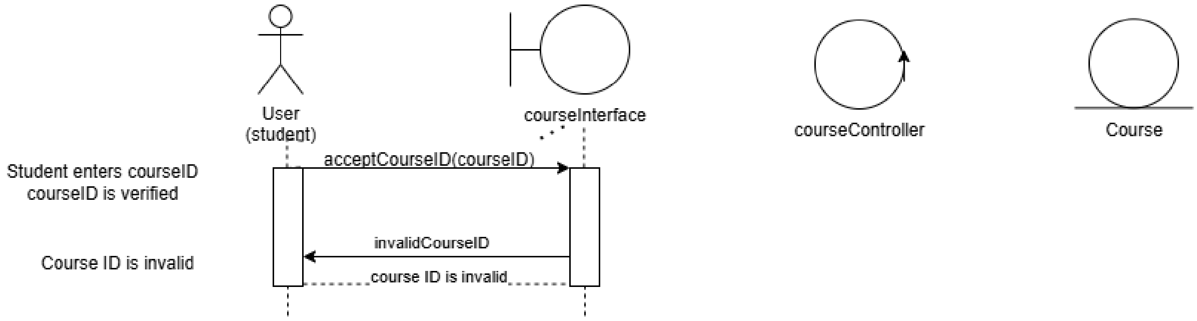


4. Make submission

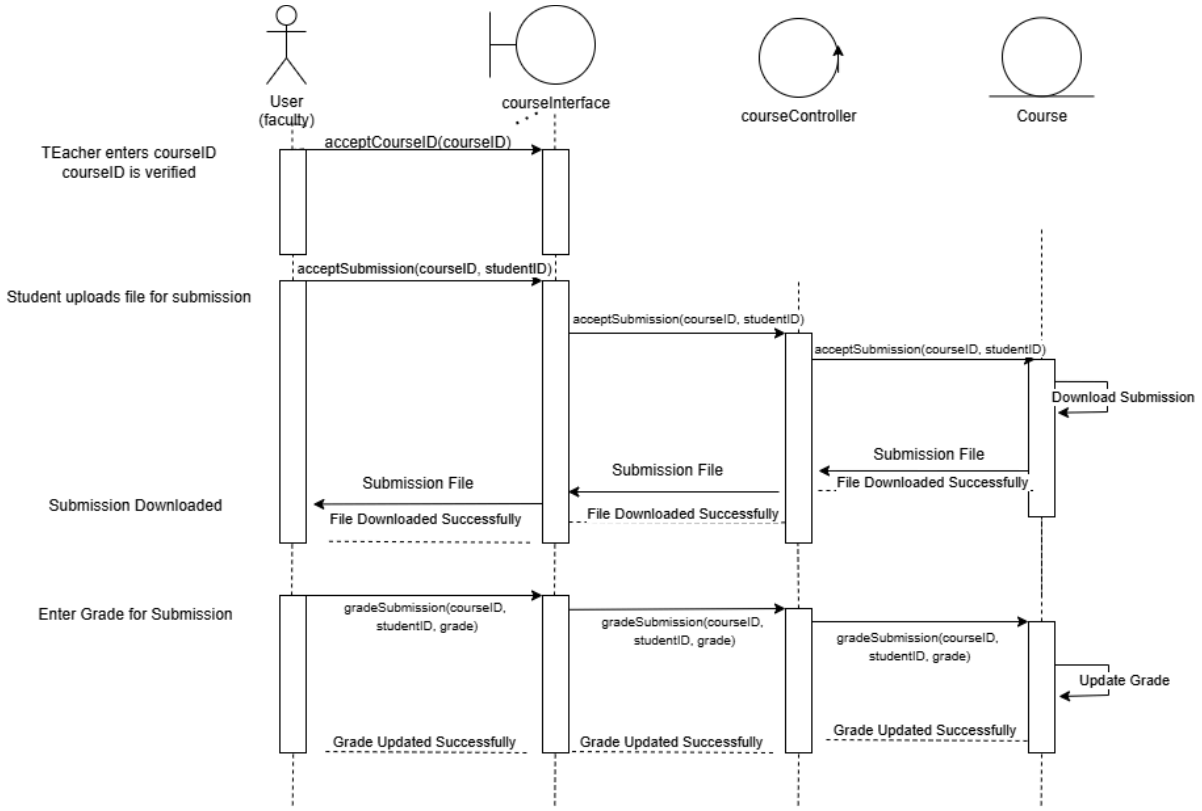

User
(faculty)
courseInterface
courseController
Course
Student enters courseID
CourseID is verified
acceptCourseID(courseID)
Student uploads file for submission
submitAssignment(CourseID, file)
submitAssignment(CourseID, file)
submitAssignment(CourseID, file)
Upload file
File Uploaded Successfully
Submission made
File Uploaded Successfully
File Uploaded Successfully

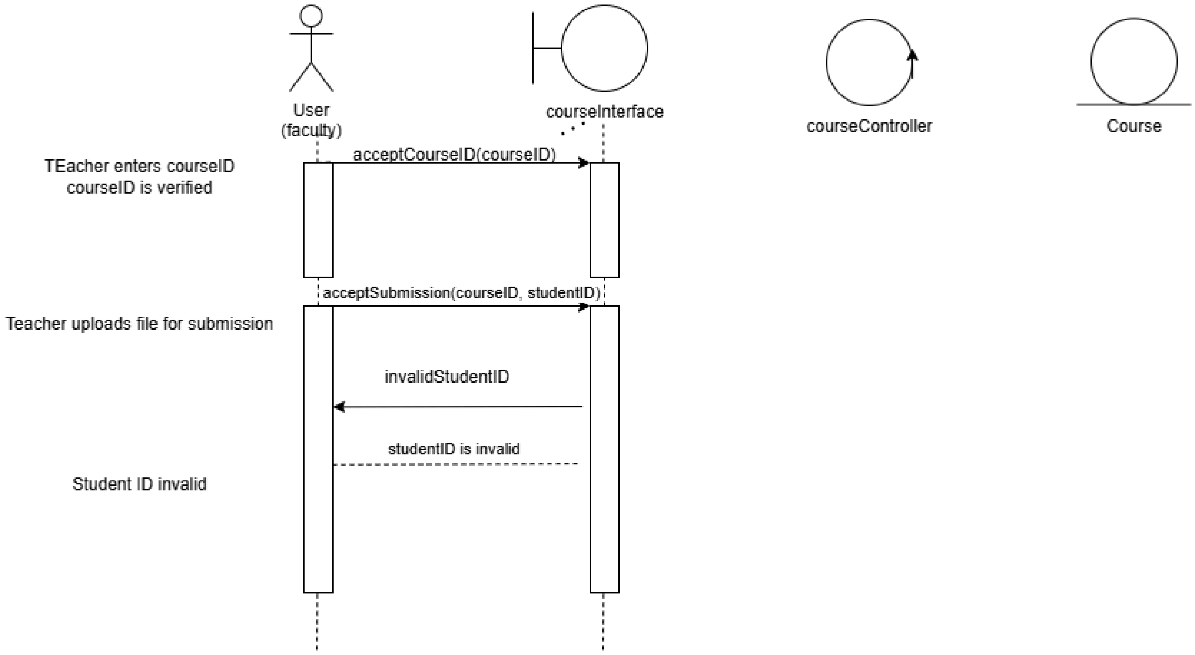

User (student)
courseInterface
courseController
Course
Student enters courseID
CourseID is verified
acceptCourseID(courseID)
submitAssignment(CourseID, file)
Student uploads file for submissions
submitAssignment(CourseID, file)
Deadline exceeded
Deadline exceeded
Submission not accepted
Late Submission, not accepted
Late Submission, not accepted

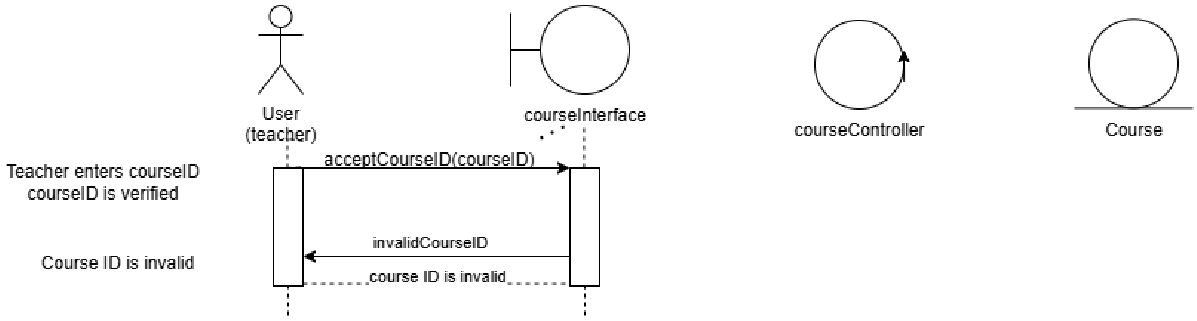

User (student)
courseInterface
courseController
Course
Student enters courseID
CourseID is verified
acceptCourseID(courseID)
submitAssignment(CourseID, file)
Student uploads file for submissions
submitAssignment(CourseID, file)
File exceeds max size limit
Submission File is too large
Upload failed
File too large, can't be uploaded
File too large, can't be uploaded


User (student)
courseInterface
courseController
Course
Student enters courseID
courseID is verified
acceptCourseID(courseID)
invalidCourseID
Course ID is invalid
course ID is invalid

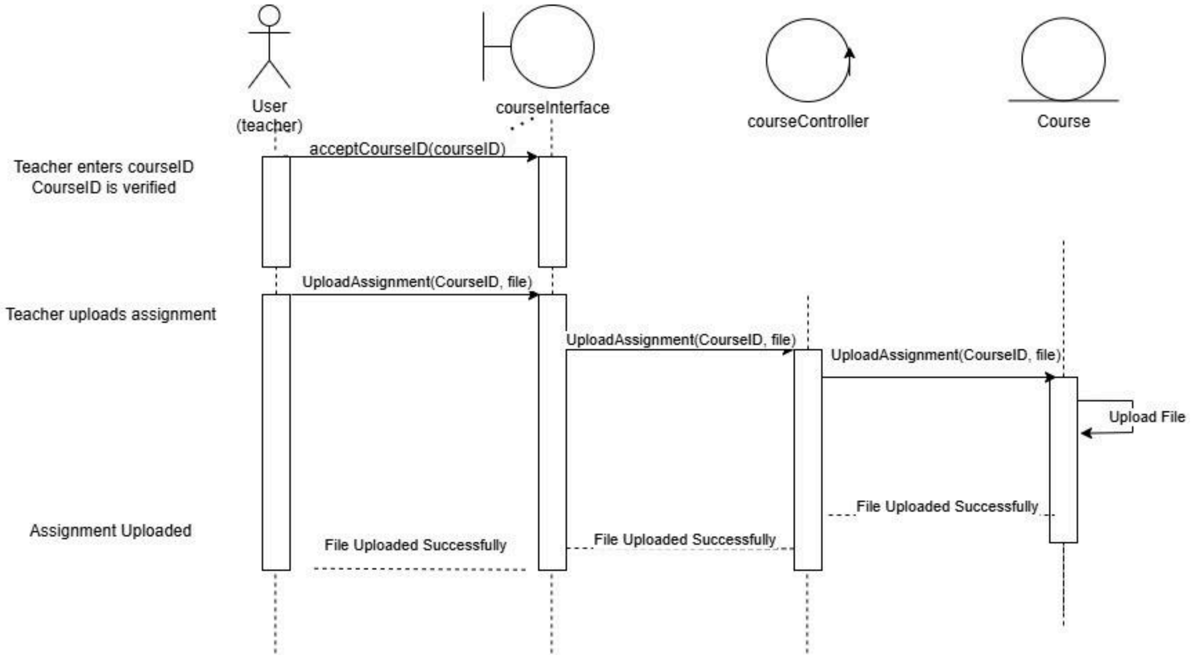
5. Accept and grade submissions

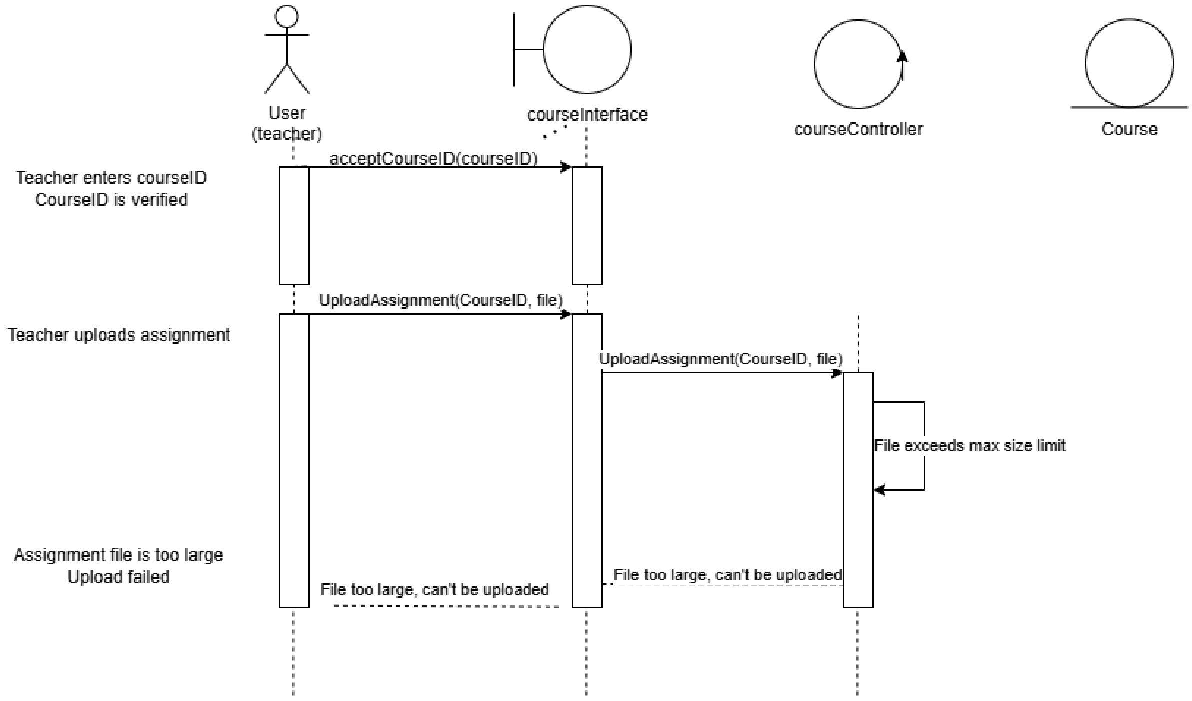

User
(faculty)
courseInterface
courseController
Course
TEacher enters courseID
courseID is verified
acceptCourseID(courseID)
acceptSubmission(courseID, studentID)
Student uploads file for submission
acceptSubmission(courseID, studentID)
acceptSubmission(courseID, studentID)
Download Submission
Submission File
File Downloaded Successfully
Submission File
File Downloaded Successfully
Submission File
File Downloaded Successfully
Submission Downloaded
Enter Grade for Submission
gradeSubmission(courseID, studentID, grade)
gradeSubmission(courseID, studentID, grade)
gradeSubmission(courseID, studentID, grade)
Update Grade
Grade Updated Successfully
Grade Updated Successfully
Grade Updated Successfully

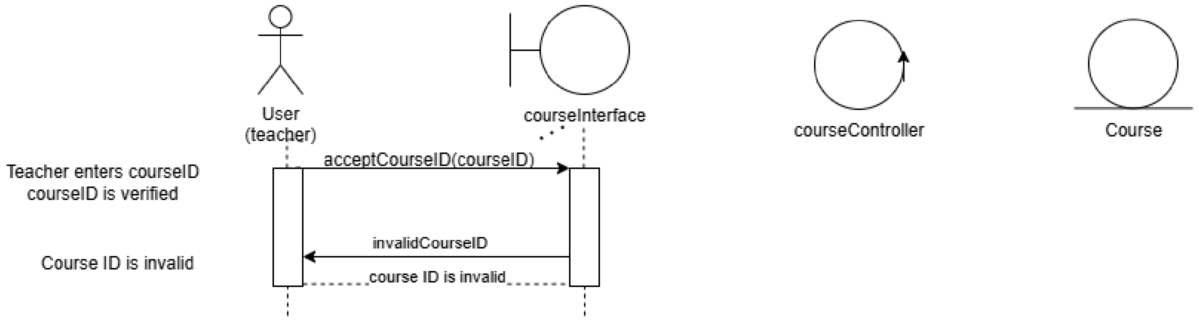




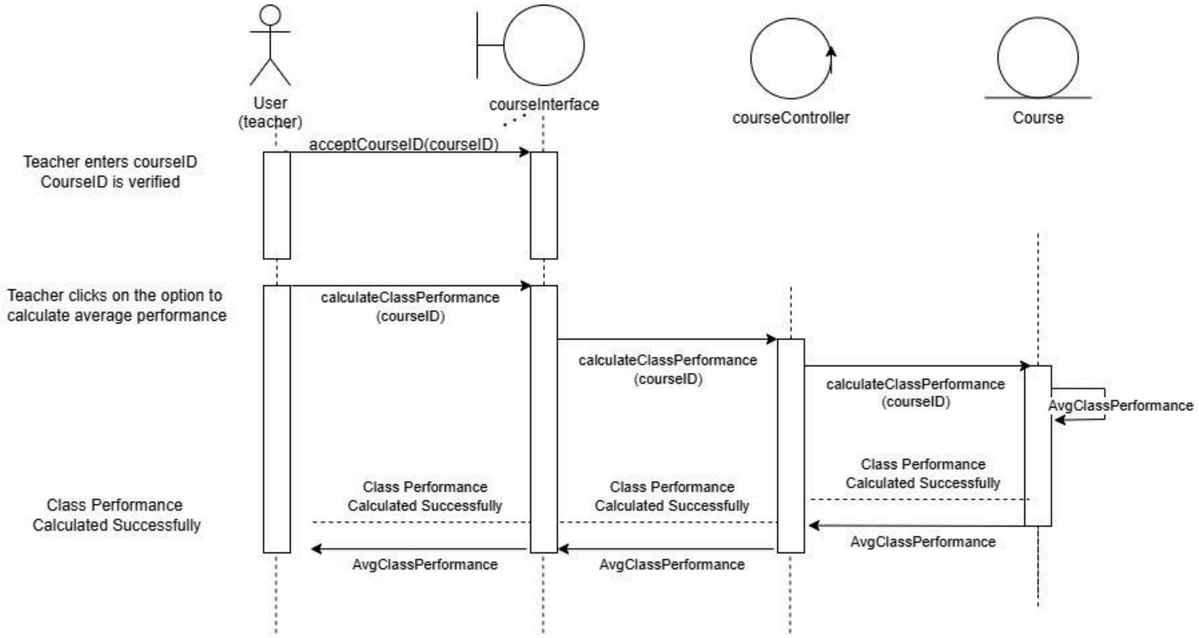
6. Upload assignments

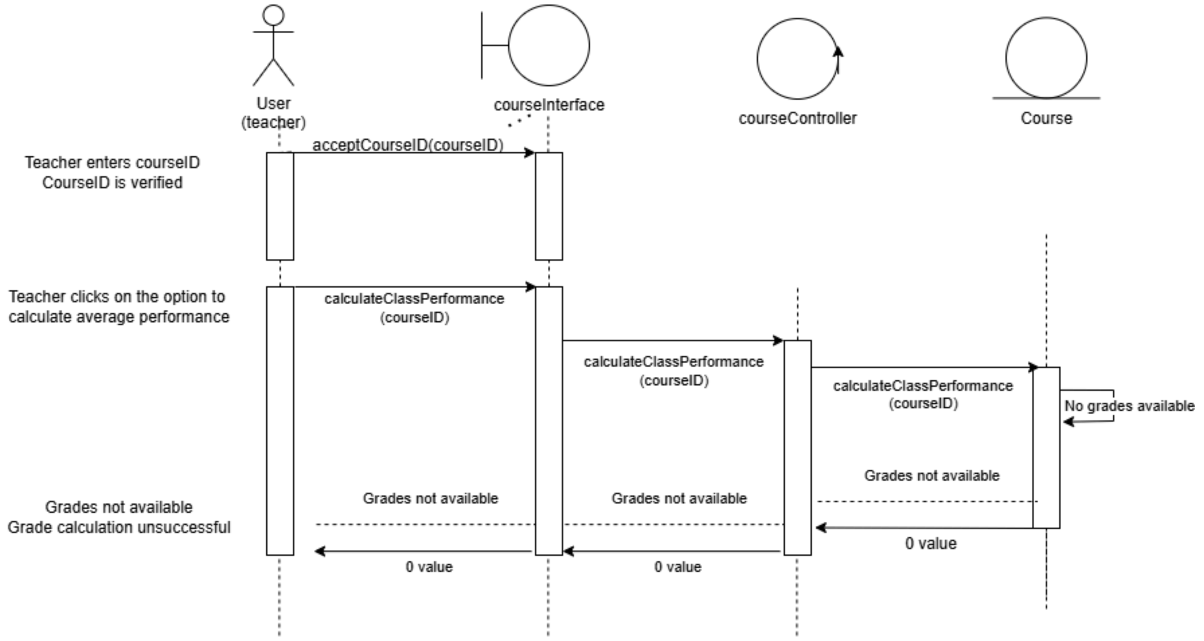

User
(teacher)
courseInterface
courseController
Course
acceptCourseID(courseID)
Teacher enters courseID
CourseID is verified
UploadAssignment(CourseID, file)
Teacher uploads assignment
UploadAssignment(CourseID, file)
UploadAssignment(CourseID, file)
Upload File
File Uploaded Successfully
File Uploaded Successfully
Assignment Uploaded
File Uploaded Successfully

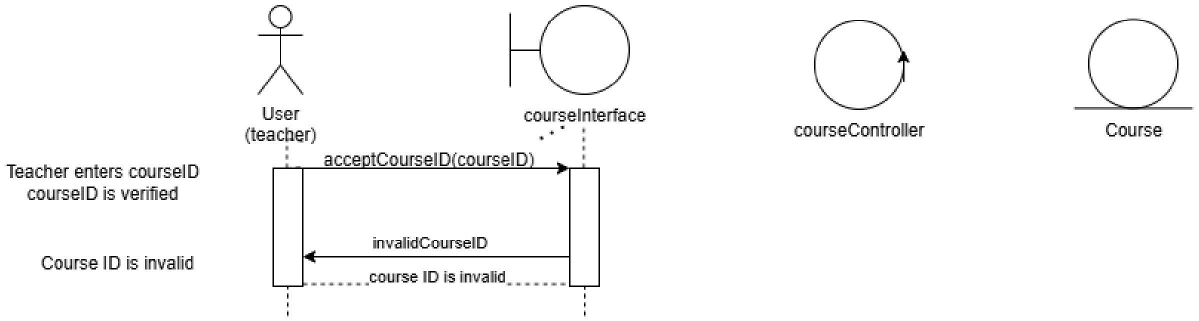

User
(teacher)
courseInterface
courseController
Course
acceptCourseID(courseID)
Teacher enters courseID
CourseID is verified
UploadAssignment(CourseID, file)
Teacher uploads assignment
UploadAssignment(CourseID, file)
File exceeds max size limit
Assignment file is too large
Upload failed
File too large, can't be uploaded
File too large, can't be uploaded



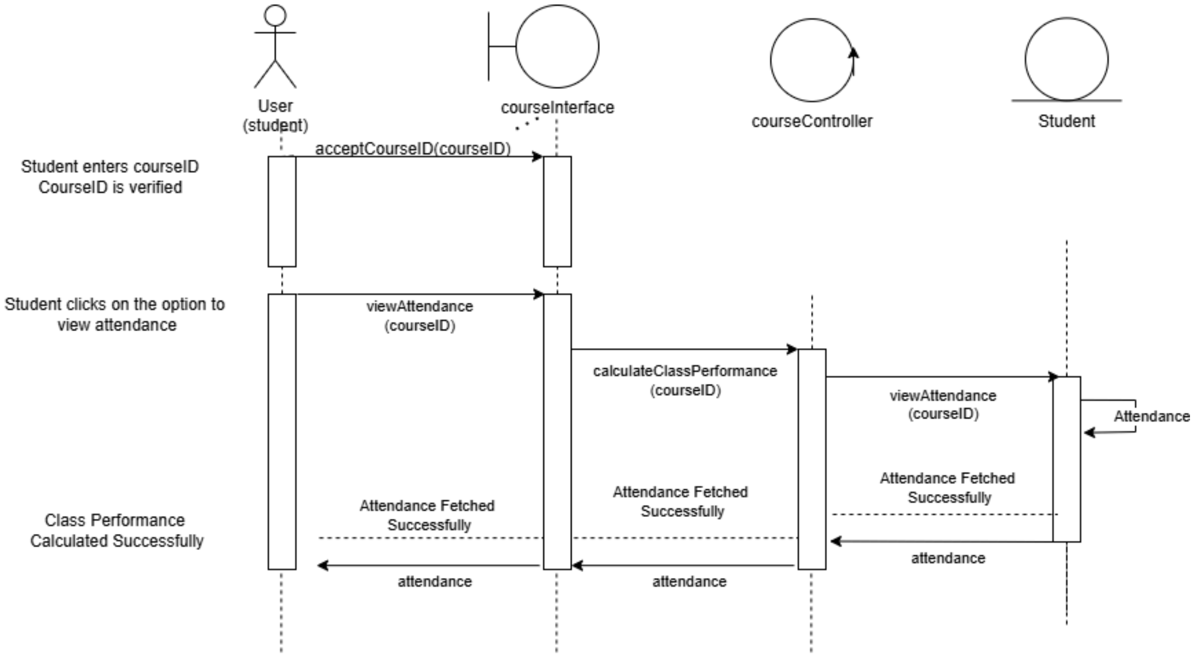
## 7. Calculate average performance

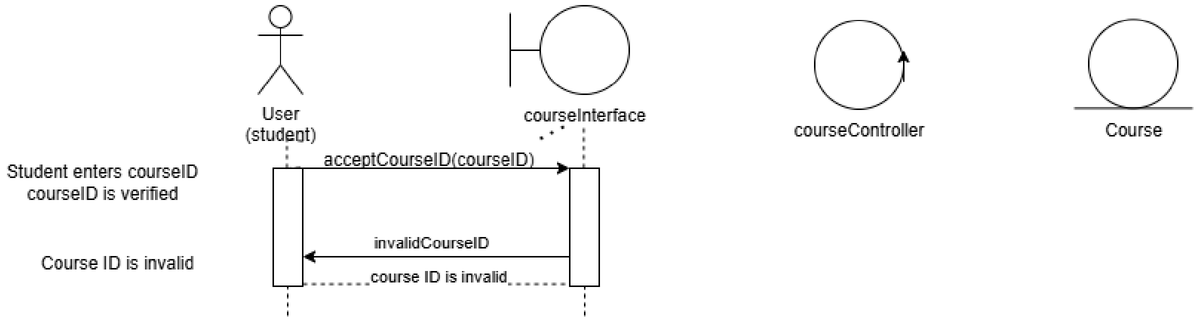




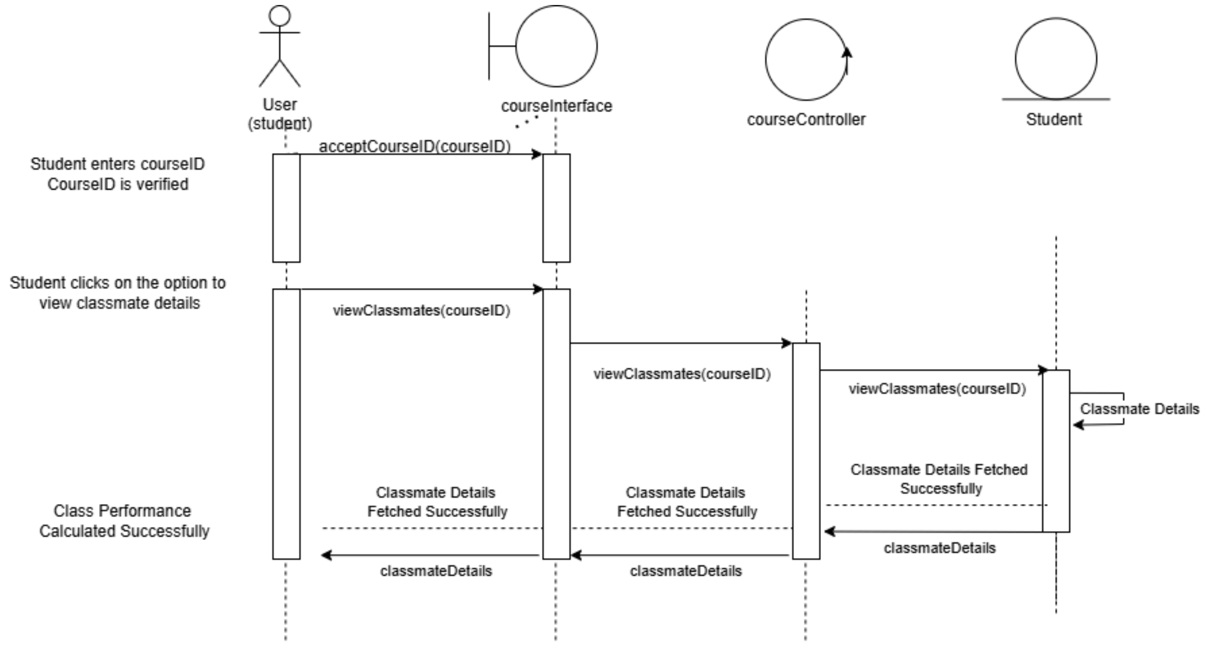


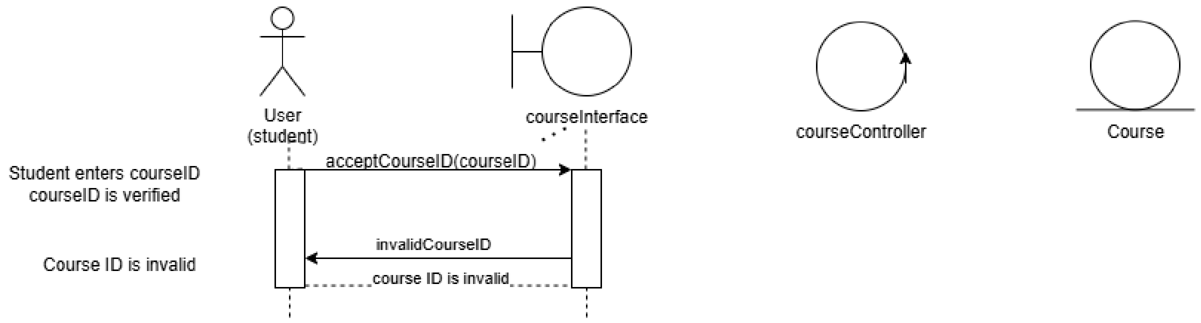
8. View attendance



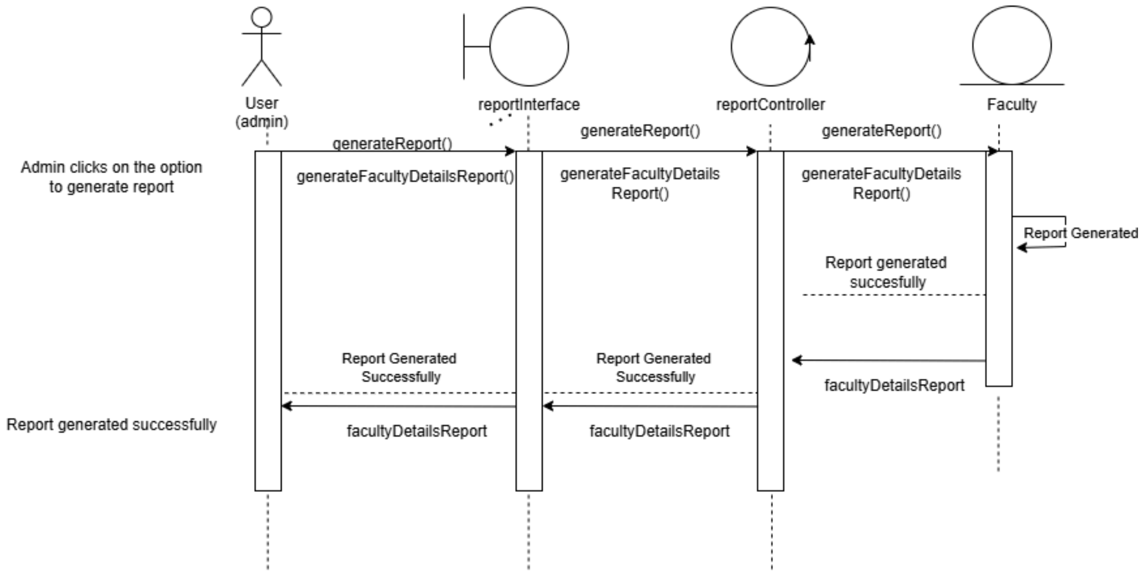


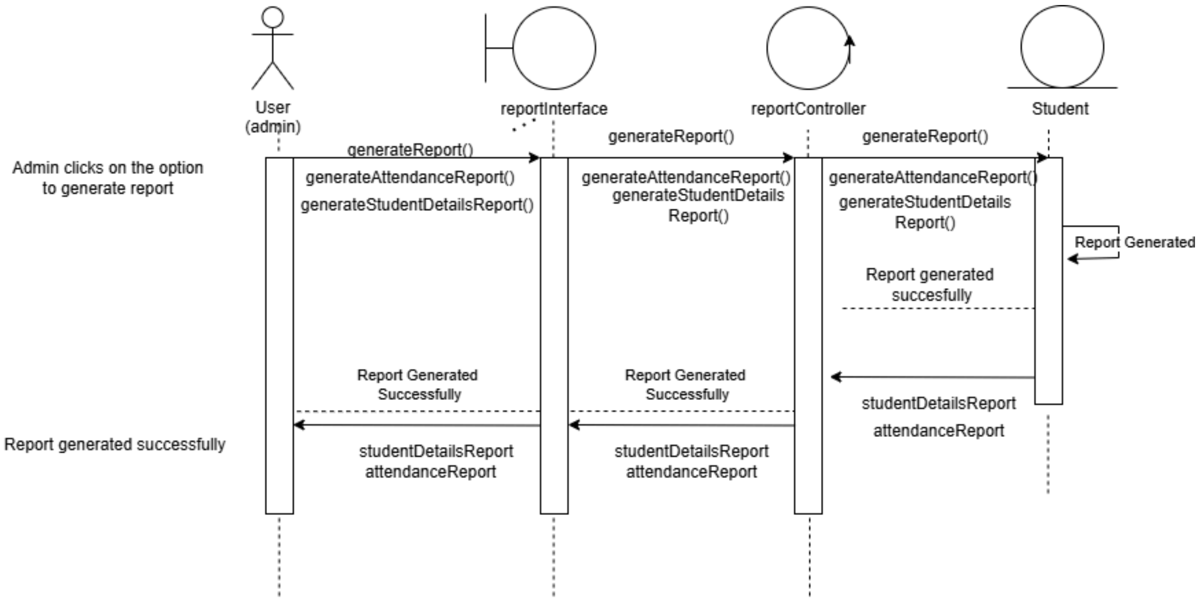
9. View classmate details

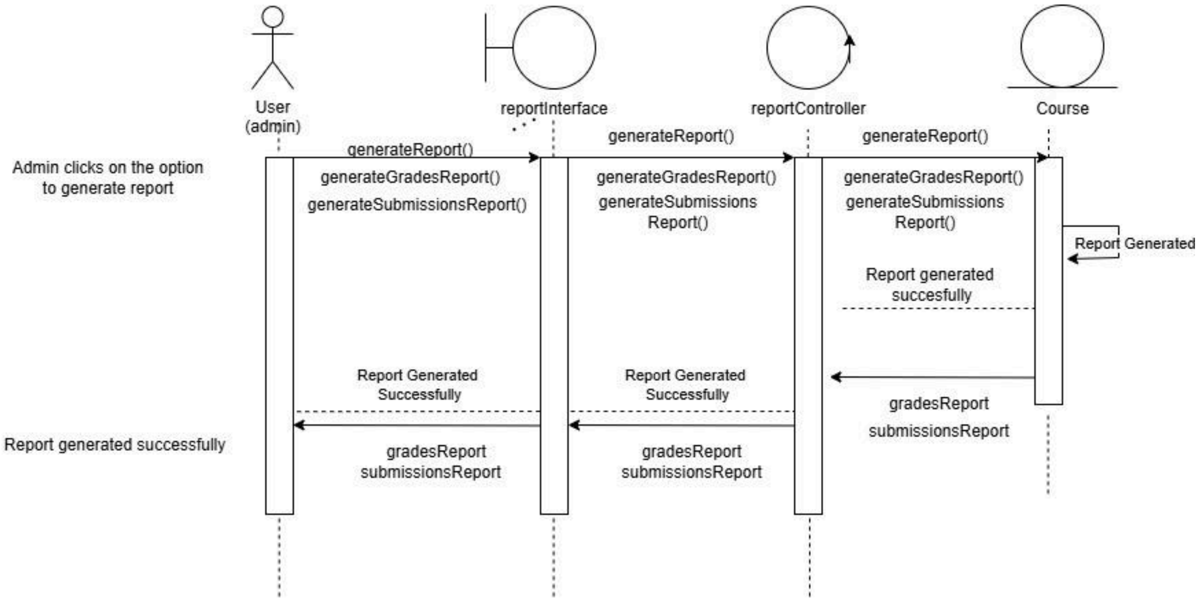




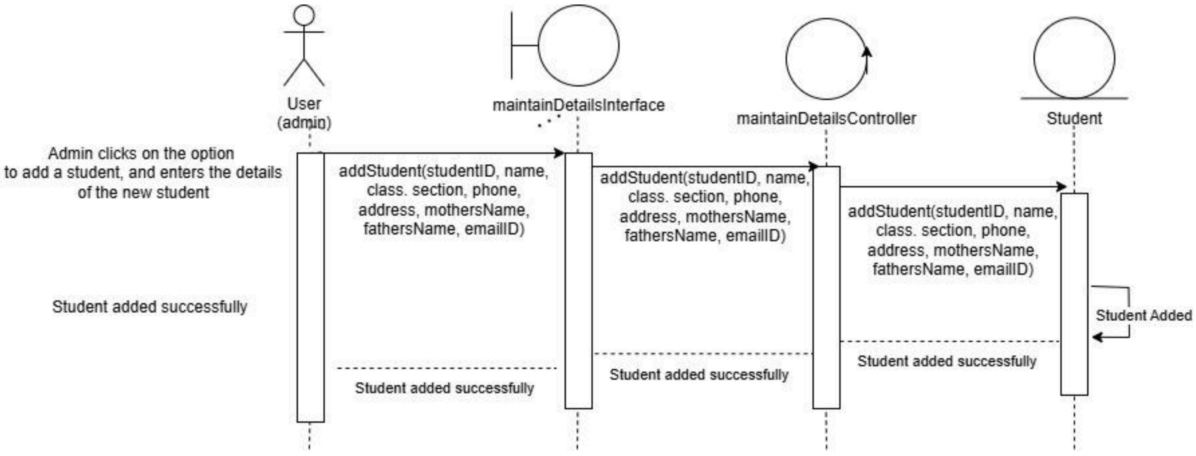
10. Generate reports

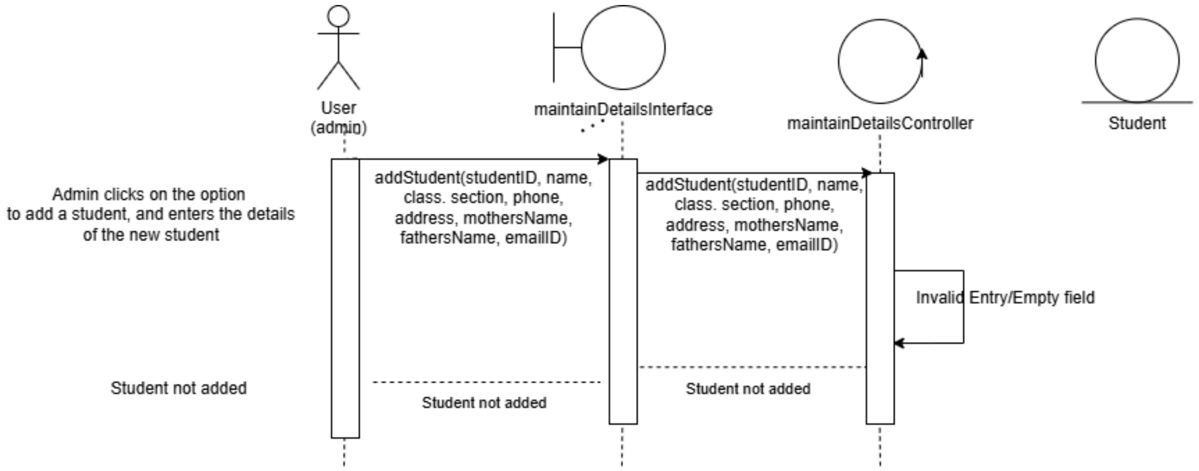


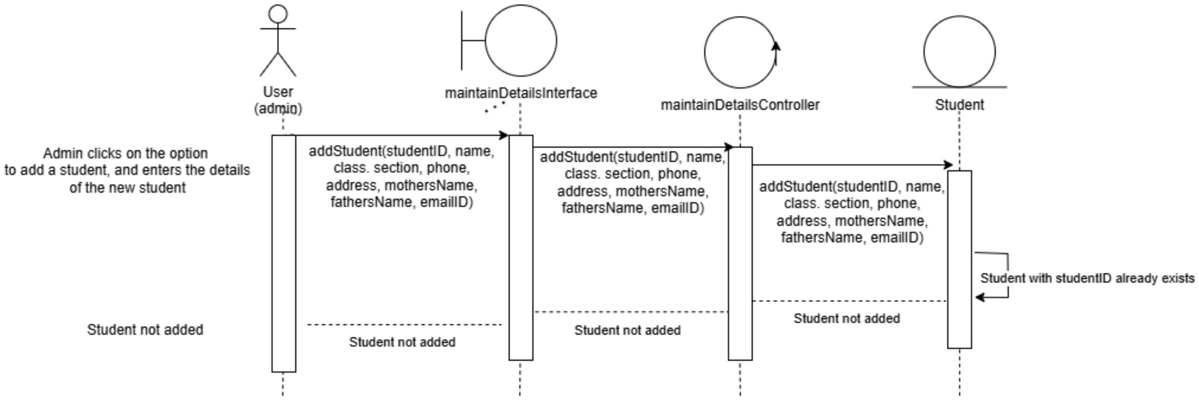


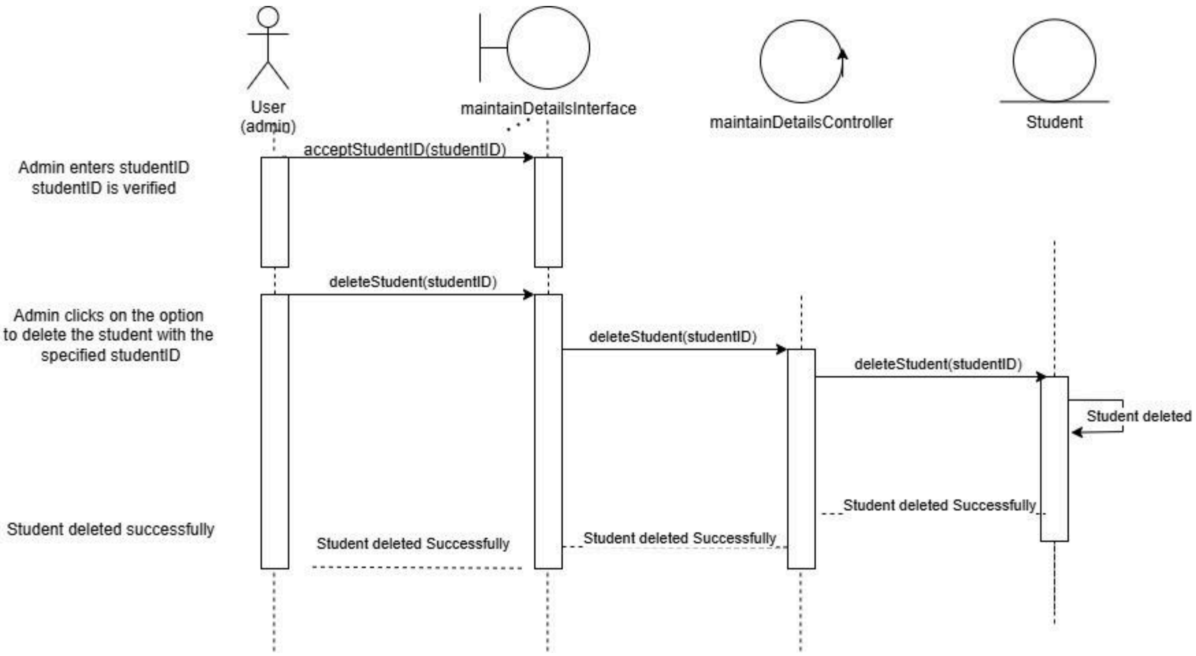


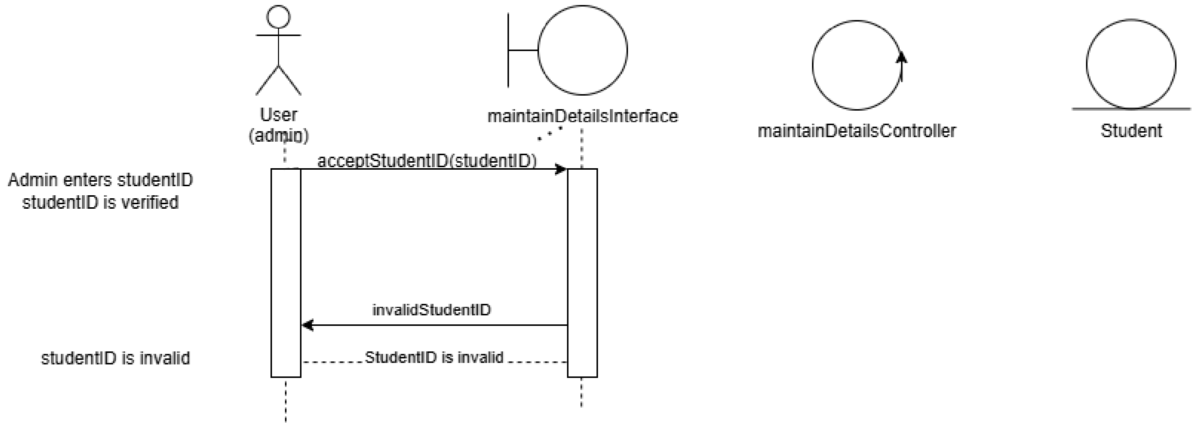
## 11. Maintain student details

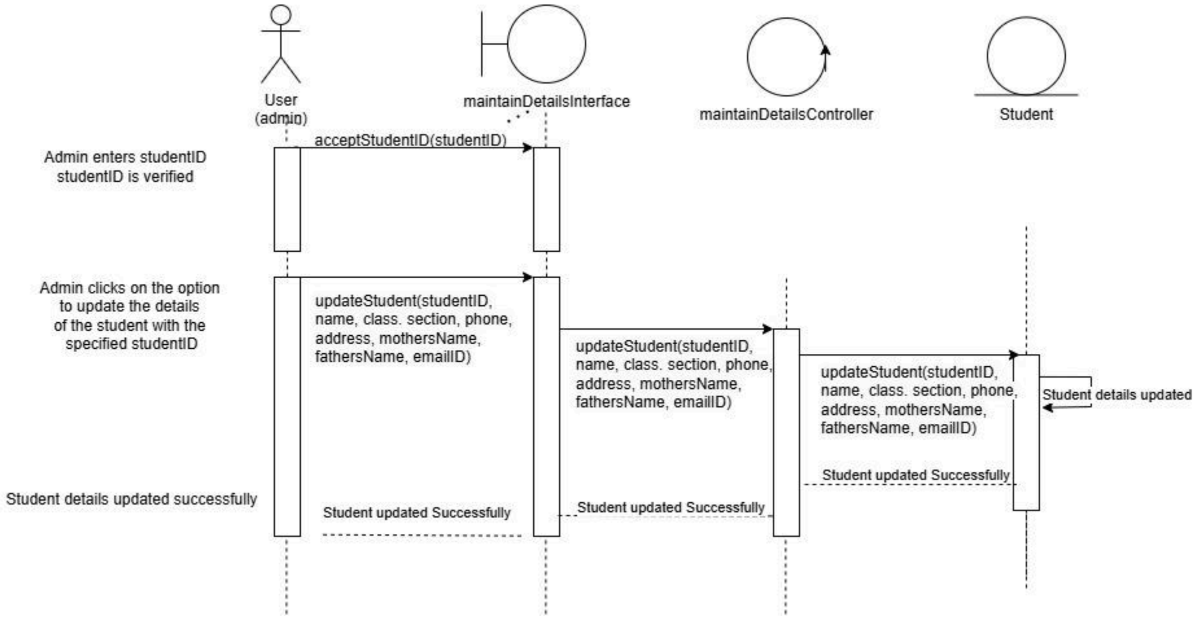

User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin clicks on the option
to add a student, and enters the details
of the new student
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
Student Added
Student added successfully
Student added successfully
Student added successfully
Student added successfully

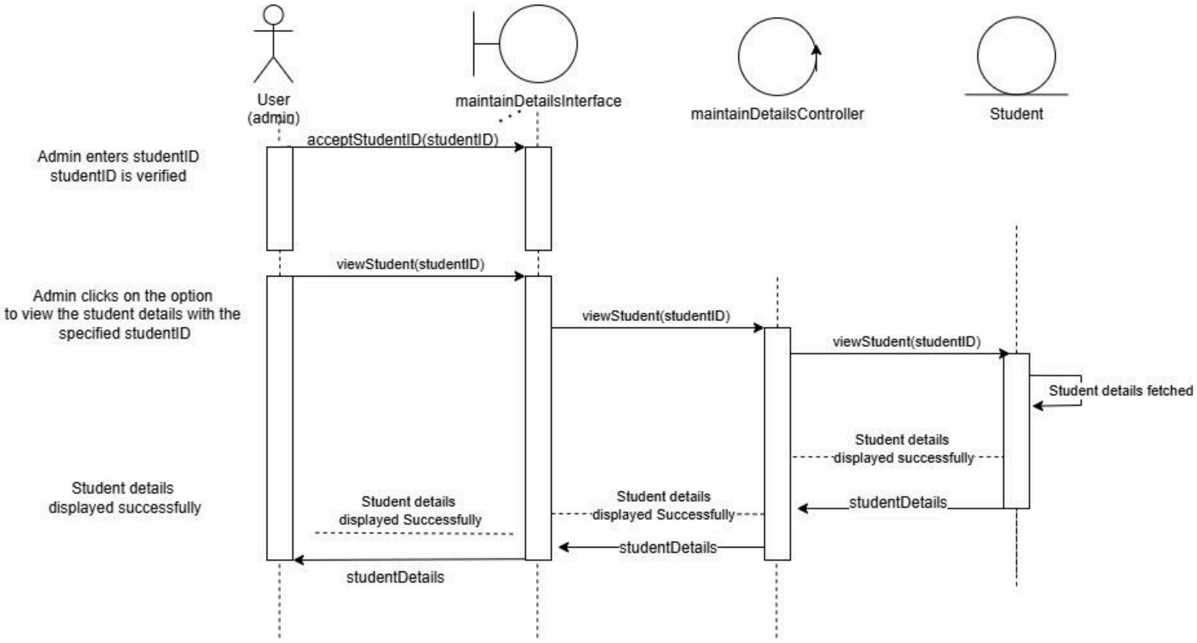

User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin clicks on the option
to add a student, and enters the details
of the new student
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
Invalid Entry/Empty field
Student not added
Student not added
Student not added


User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin clicks on the option
to add a student, and enters the details
of the new student
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
addStudent(studentID, name,
class. section, phone,
address, mothersName,
fathersName, emailID)
Student with studentID already exists
Student not added
Student not added
Student not added
Student not added

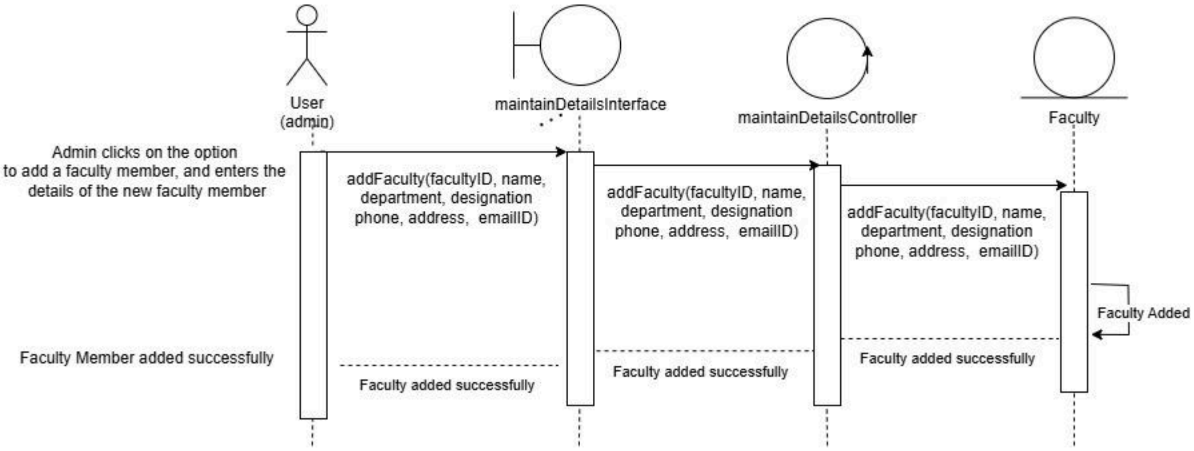

User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin enters studentID
studentID is verified
acceptStudentID(studentID)
deleteStudent(studentID)
Admin clicks on the option
to delete the student with the
specified studentID
deleteStudent(studentID)
deleteStudent(studentID)
Student deleted
Student deleted Successfully
Student deleted Successfully
Student deleted successfully
Student deleted Successfully

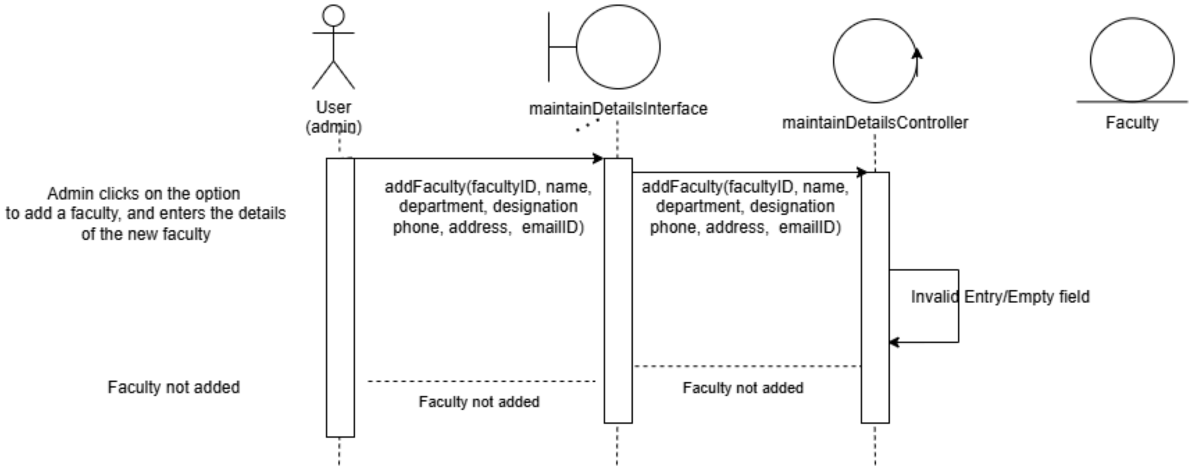

User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin enters studentID
studentID is verified
acceptStudentID(studentID)
invalidStudentID
studentID is invalid
StudentID is invalid

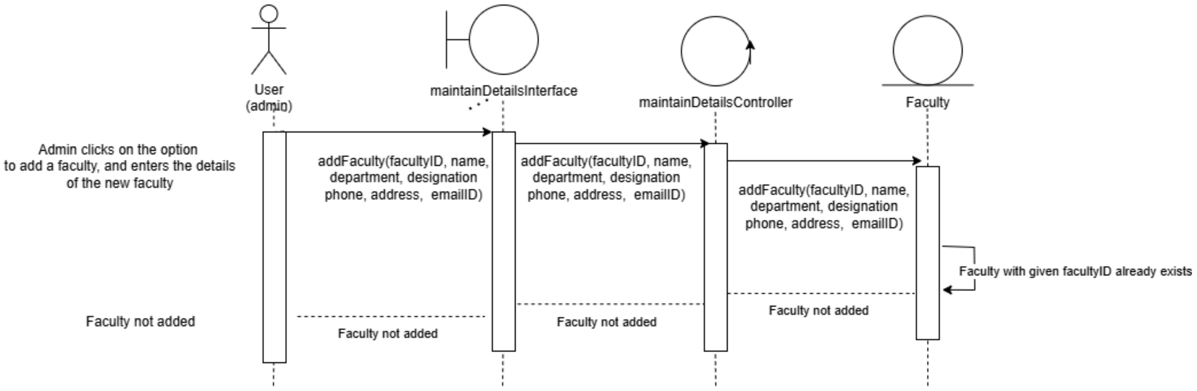

User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin enters studentID
studentID is verified
acceptStudentID(studentID)
Admin clicks on the option
to update the details
of the student with the
specified studentID
updateStudent(studentID,
name, class. section, phone,
address, mothersName,
fathersName, emailID)
updateStudent(studentID,
name, class. section, phone,
address, mothersName,
fathersName, emailID)
updateStudent(studentID,
name, class. section, phone,
address, mothersName,
fathersName, emailID)
Student details updated
Student updated Successfully
Student updated Successfully
Student details updated successfully
Student updated Successfully

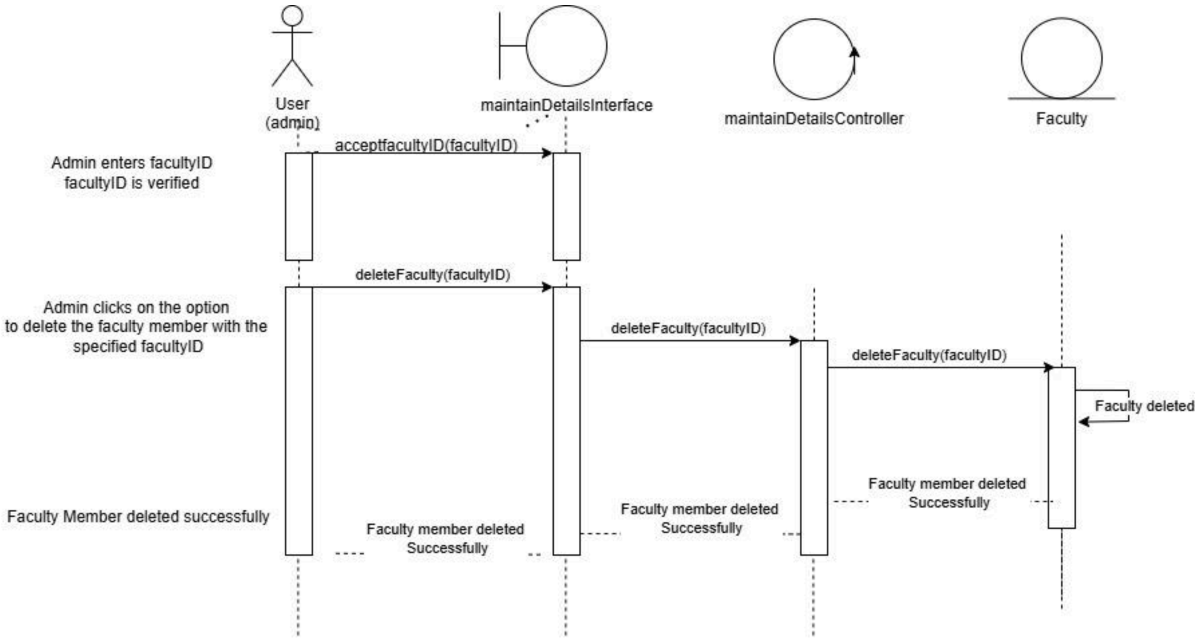

User
(admin)
maintainDetailsInterface
maintainDetailsController
Student
Admin enters studentID
studentID is verified
acceptStudentID(studentID)
viewStudent(studentID)
Admin clicks on the option
to view the student details with the
specified studentID
viewStudent(studentID)
viewStudent(studentID)
Student details fetched
Student details
displayed successfully
Student details
displayed successfully
Student details
displayed Successfully
Student details
displayed Successfully
studentDetails
studentDetails
studentDetails

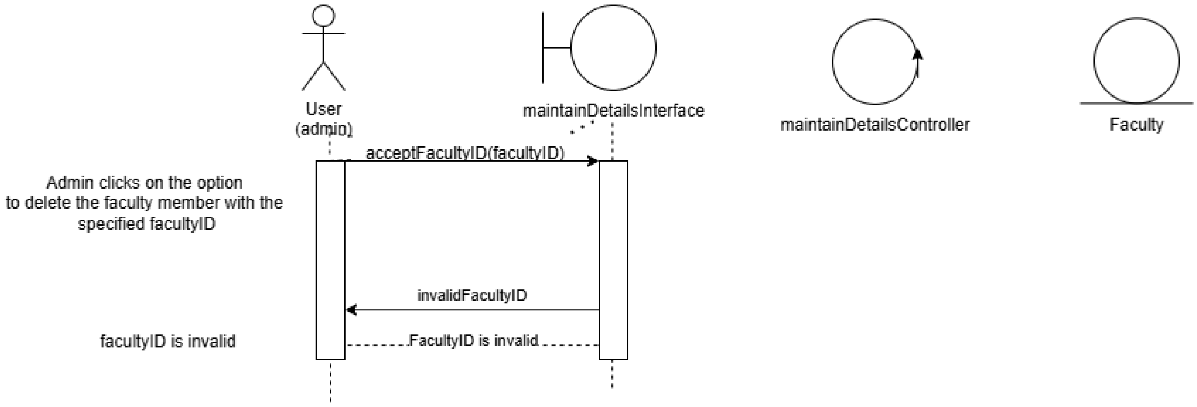
## 12. Maintain faculty details

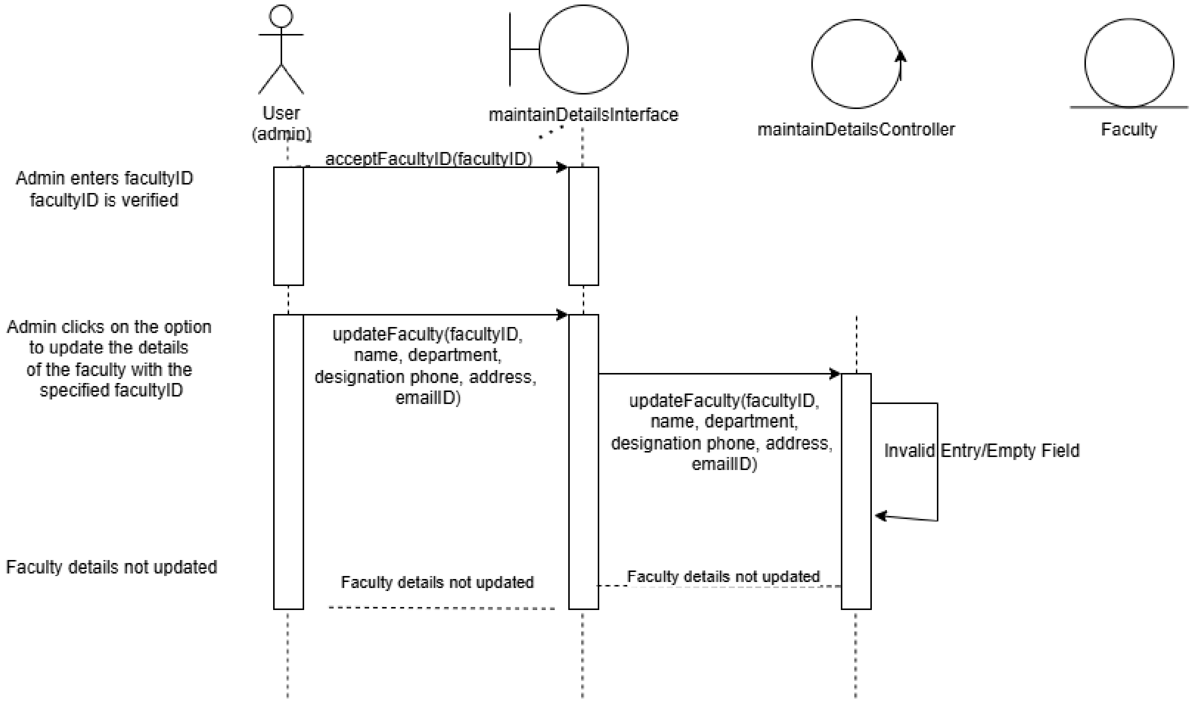

User
(admin)
maintainDetailsInterface
maintainDetailsController
Faculty
Admin clicks on the option
to add a faculty member, and enters the
details of the new faculty member
addFaculty(facultyID, name,
department, designation
phone, address, emailID)
addFaculty(facultyID, name,
department, designation
phone, address, emailID)
addFaculty(facultyID, name,
department, designation
phone, address, emailID)
Faculty Added
Faculty Member added successfully
Faculty added successfully
Faculty added successfully
Faculty added successfully

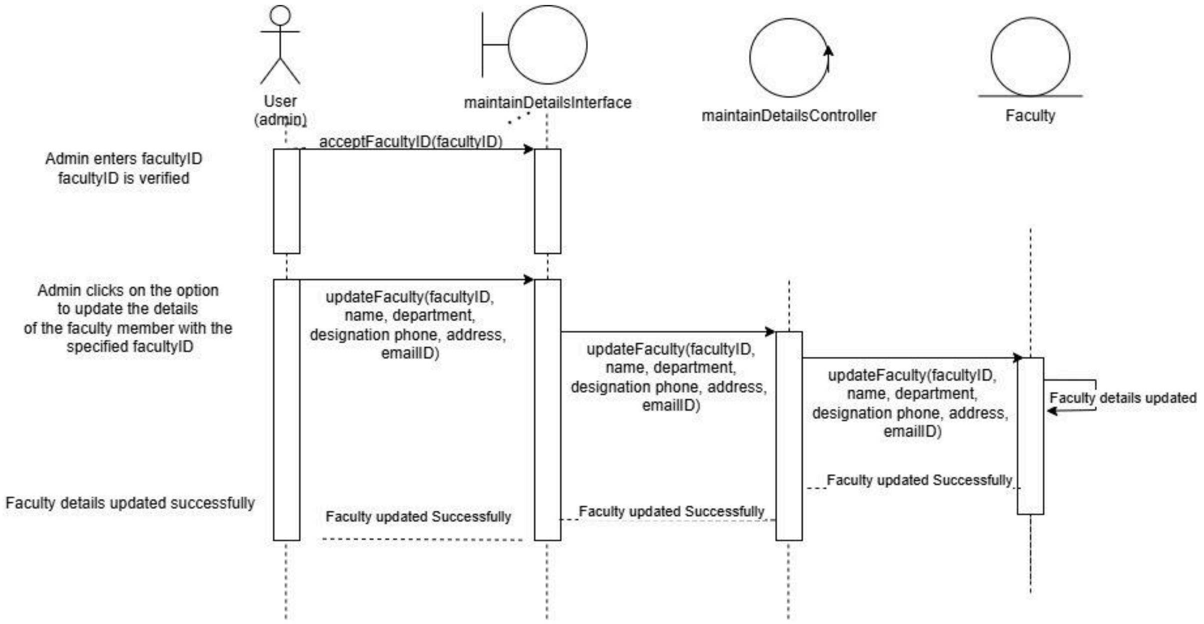


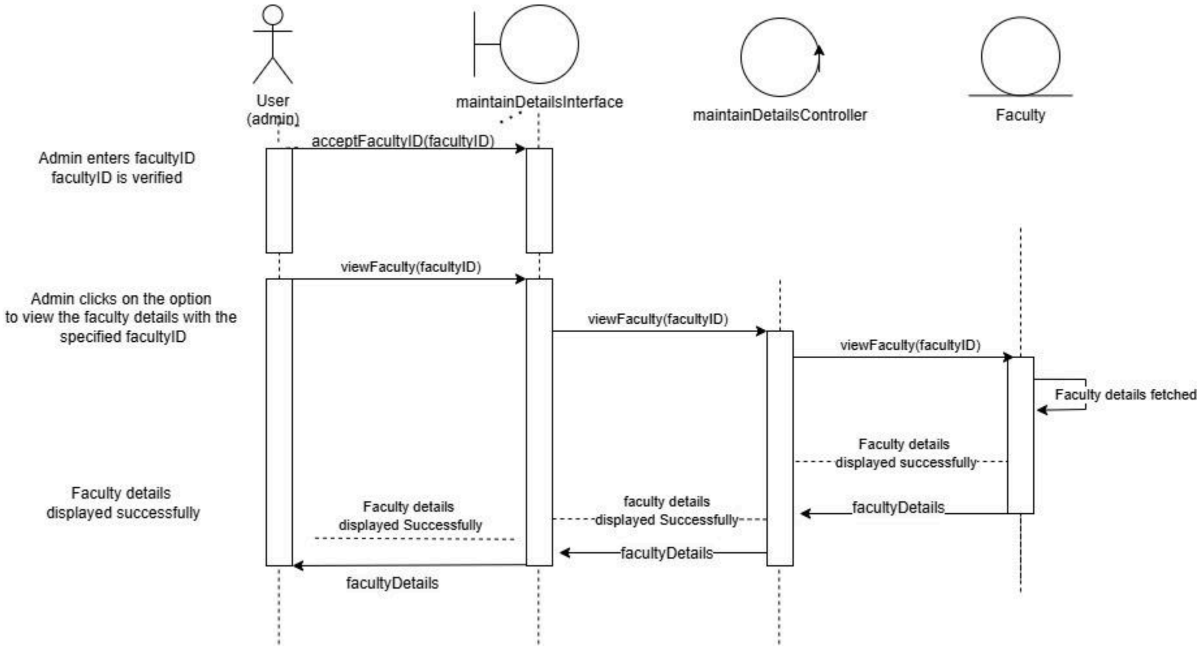



User
(admin)
maintainDetailsInterface
maintainDetailsController
Faculty
acceptfacultyID(facultyID)
Admin enters facultyID
facultyID is verified
deleteFaculty(facultyID)
Admin clicks on the option
to delete the faculty member with the
specified facultyID
deleteFaculty(facultyID)
deleteFaculty(facultyID)
Faculty deleted
Faculty member deleted
Successfully
Faculty member deleted
Successfully
Faculty Member deleted successfully
Faculty member deleted
Successfully

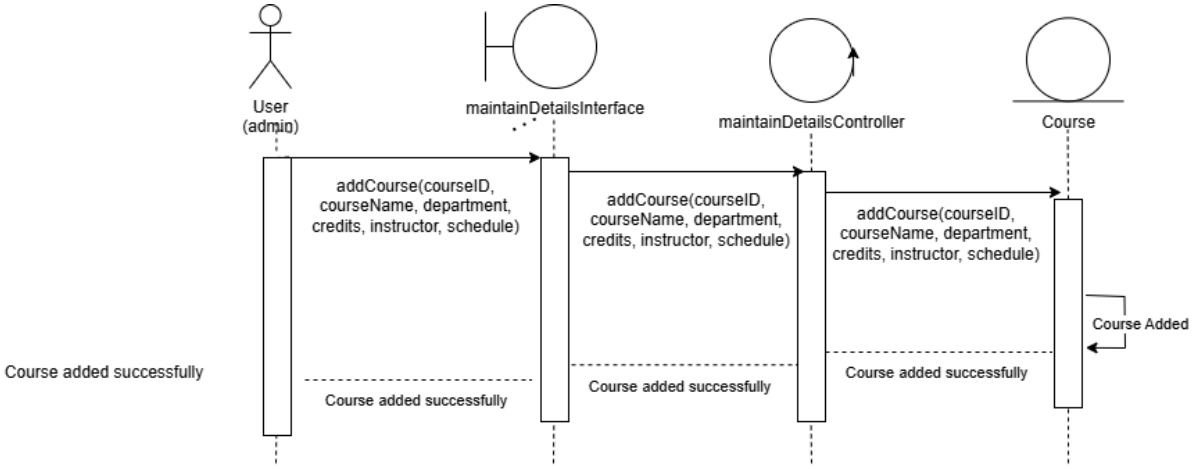

User
(admin)
maintainDetailsInterface
maintainDetailsController
Faculty
acceptFacultyID(facultyID)
Admin clicks on the option
to delete the faculty member with the
specified facultyID
invalidFacultyID
facultyID is invalid
FacultyID is invalid

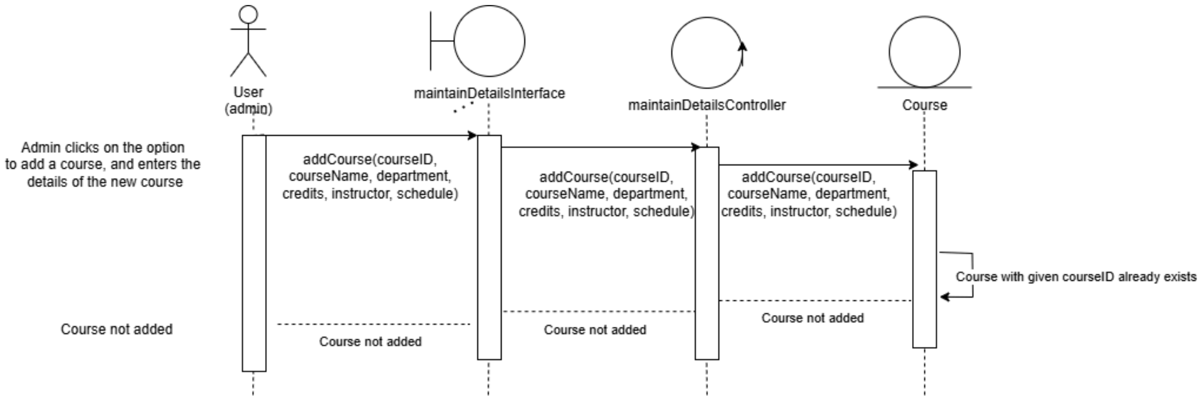

User
(admin)
maintainDetailsInterface
maintainDetailsController
Faculty
acceptFacultyID(facultyID)
Admin enters facultyID
facultyID is verified
Admin clicks on the option
to update the details
of the faculty with the
specified facultyID
updateFaculty(facultyID,
name, department,
designation phone, address,
emailID)
updateFaculty(facultyID,
name, department,
designation phone, address,
emailID)
Invalid Entry/Empty Field
Faculty details not updated
Faculty details not updated
Faculty details not updated

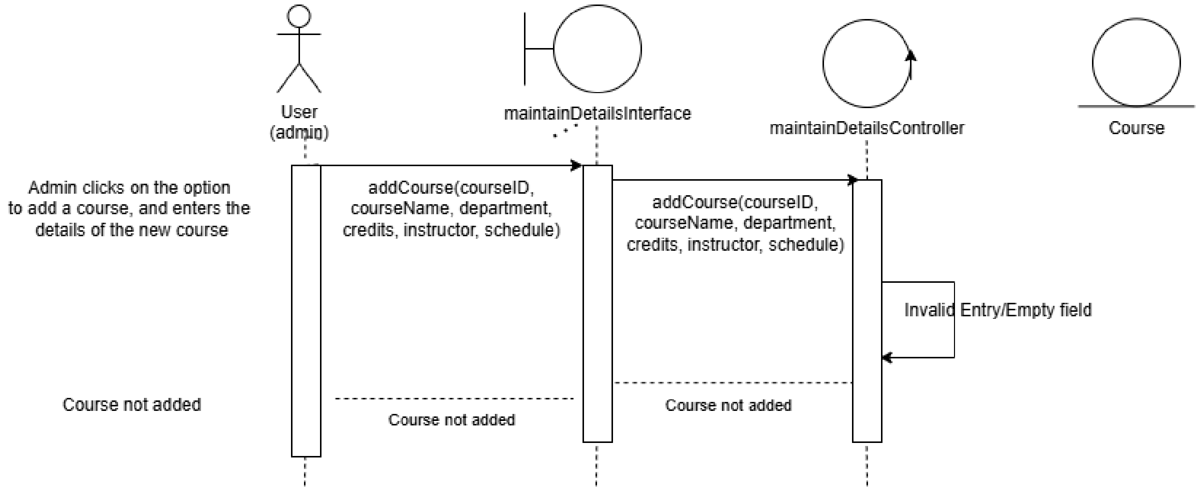

User
(admin)
maintainDetailsInterface
maintainDetailsController
Faculty
acceptFacultyID(facultyID)
Admin enters facultyID
facultyID is verified
Admin clicks on the option
to update the details
of the faculty member with the
specified facultyID
updateFaculty(facultyID,
name, department,
designation phone, address,
emailID)
updateFaculty(facultyID,
name, department,
designation phone, address,
emailID)
updateFaculty(facultyID,
name, department,
designation phone, address,
emailID)
Faculty details updated
Faculty updated Successfully
Faculty details updated successfully
Faculty updated Successfully
Faculty updated Successfully

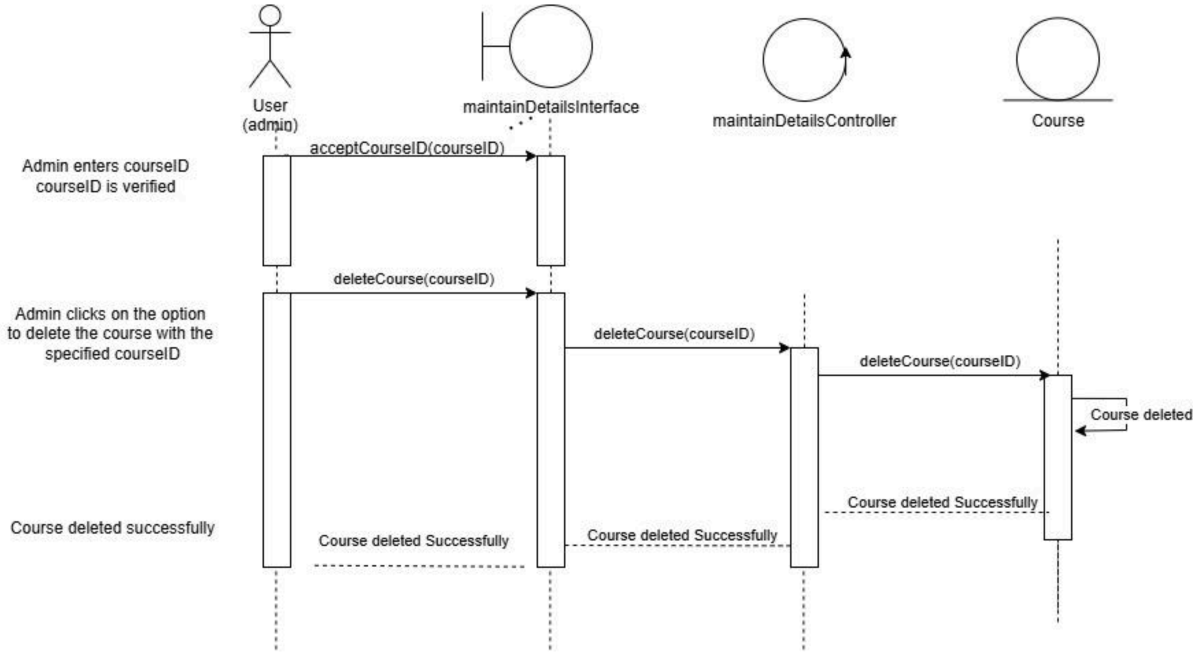


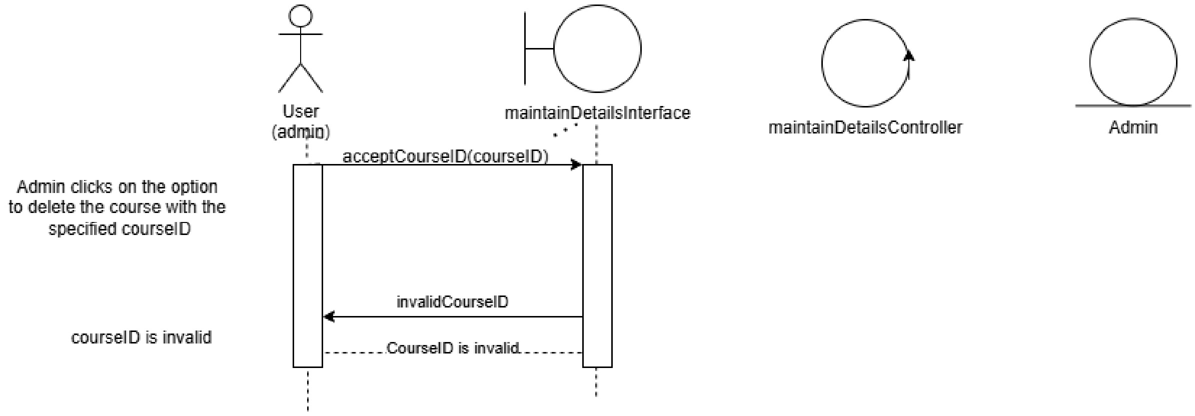
## 13. Maintain course details

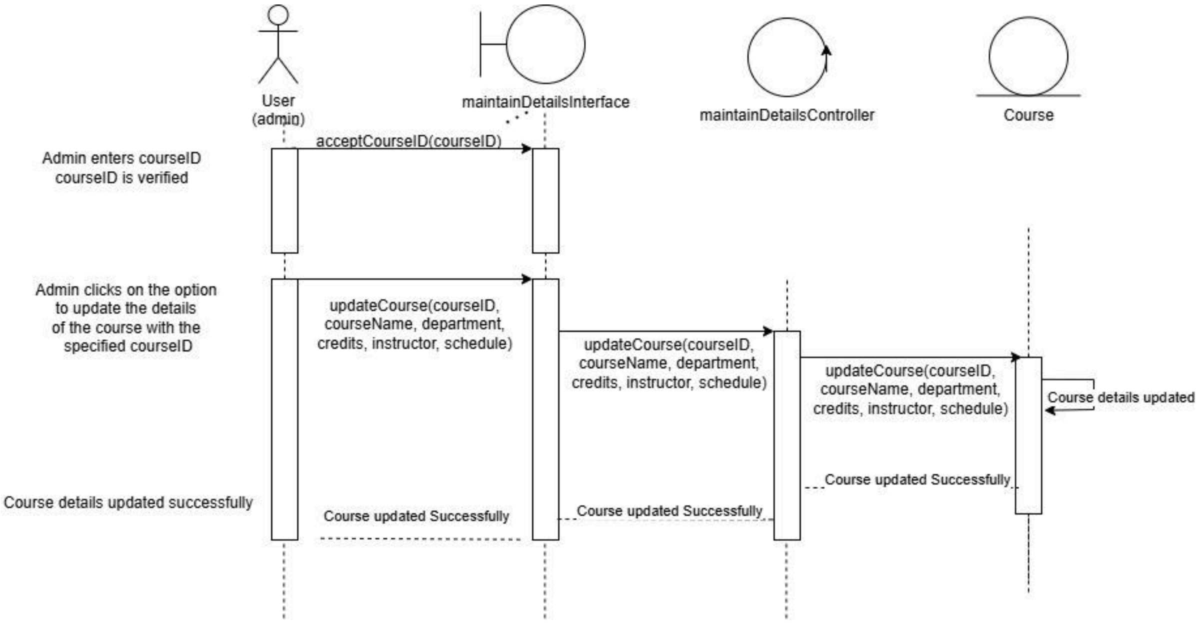

User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
Course Added
Course added successfully
Course added successfully
Course added successfully
Course added successfully

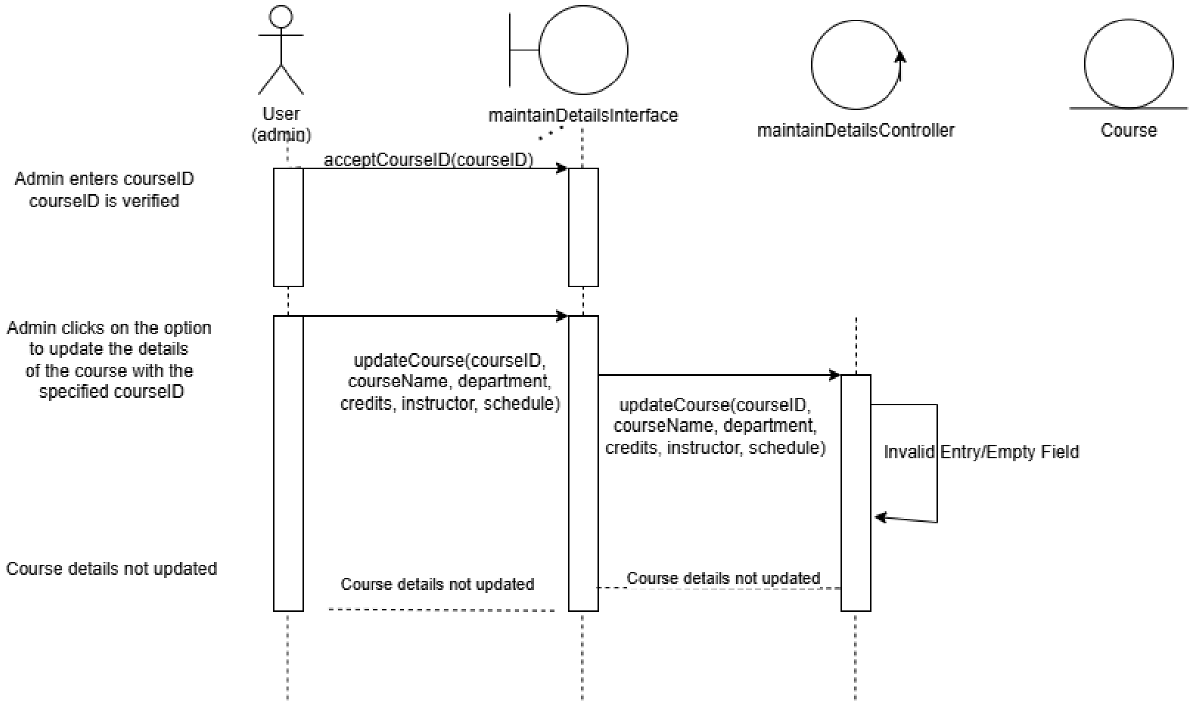

User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
Admin clicks on the option
to add a course, and enters the
details of the new course
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
Course with given courseID already exists
Course not added
Course not added
Course not added
Course not added

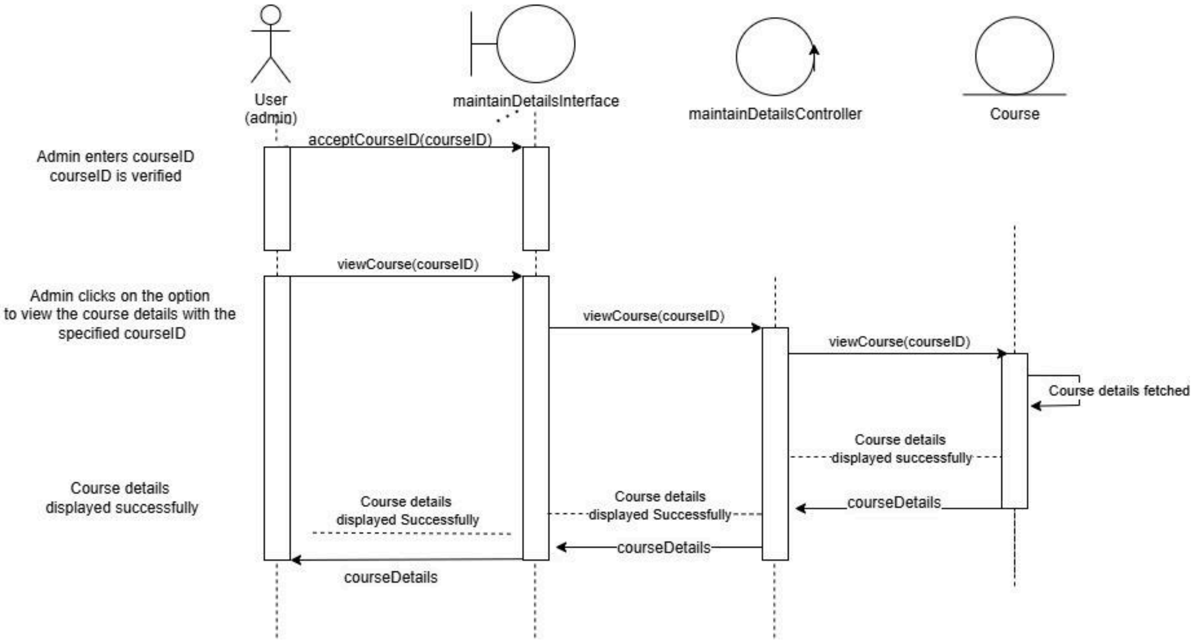

User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
Admin clicks on the option
to add a course, and enters the
details of the new course
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
addCourse(courseID,
courseName, department,
credits, instructor, schedule)
Invalid Entry/Empty field
Course not added
Course not added
Course not added


User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
acceptCourseID(courseID)
Admin enters courseID
courseID is verified
deleteCourse(courseID)
Admin clicks on the option
to delete the course with the
specified courseID
deleteCourse(courseID)
deleteCourse(courseID)
Course deleted
Course deleted Successfully
Course deleted successfully
Course deleted Successfully
Course deleted Successfully


User
(admin)
maintainDetailsInterface
maintainDetailsController
Admin
acceptCourseID(courseID)
Admin clicks on the option
to delete the course with the
specified courseID
invalidCourseID
courseID is invalid
CourseID is invalid

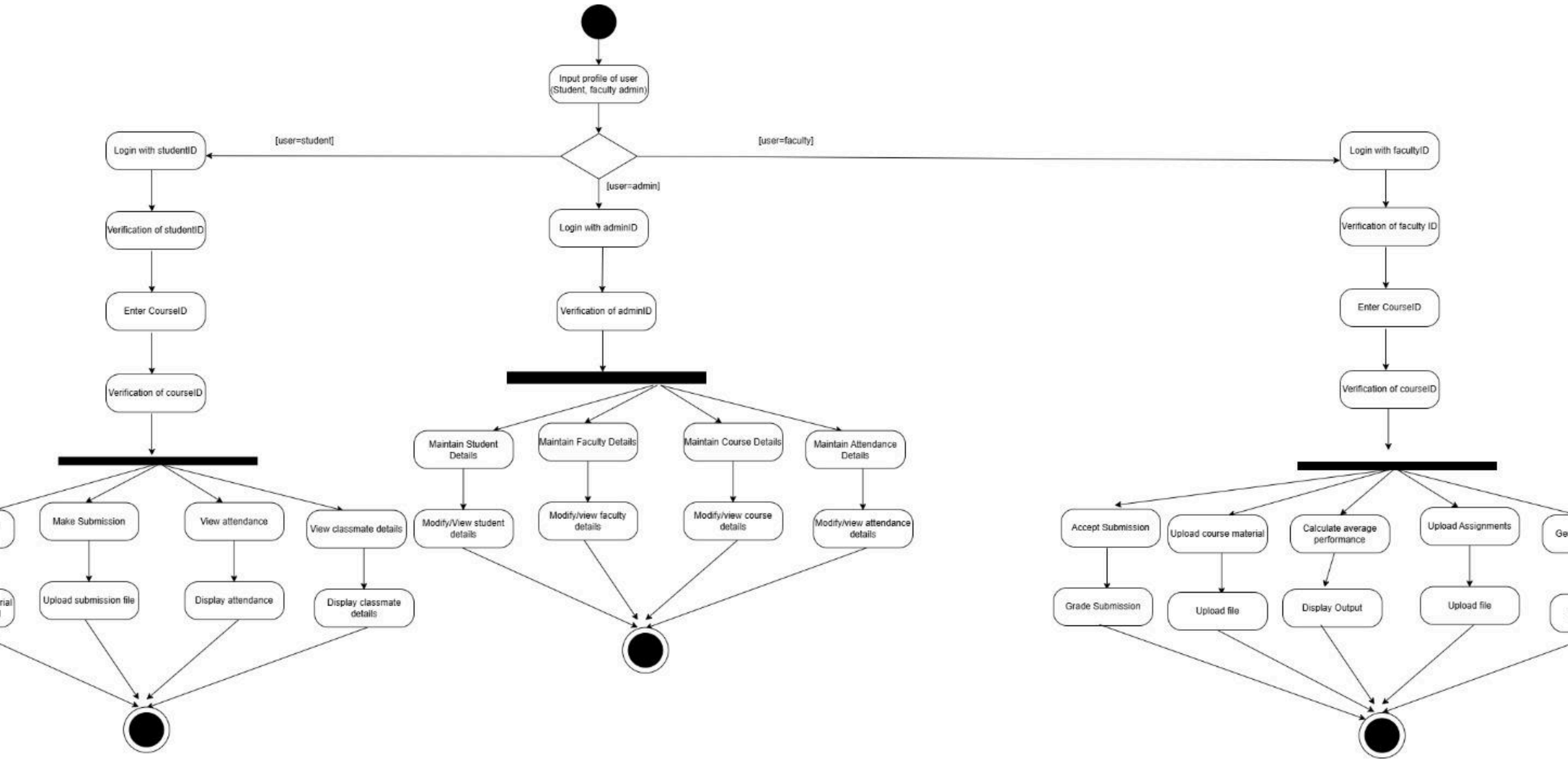

User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
acceptCourseID(courseID)
Admin enters courseID
courseID is verified
Admin clicks on the option
to update the details
of the course with the
specified courseID
updateCourse(courseID,
courseName, department,
credits, instructor, schedule)
updateCourse(courseID,
courseName, department,
credits, instructor, schedule)
updateCourse(courseID,
courseName, department,
credits, instructor, schedule)
Course details updated
Course updated Successfully
Course updated Successfully
Course updated Successfully
Course details updated successfully


User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
Admin enters courseID
courseID is verified
acceptCourseID(courseID)
Admin clicks on the option
to update the details
of the course with the
specified courseID
updateCourse(courseID,
courseName, department,
credits, instructor, schedule)
updateCourse(courseID,
courseName, department,
credits, instructor, schedule)
Invalid Entry/Empty Field
Course details not updated
Course details not updated
Course details not updated


User
(admin)
maintainDetailsInterface
maintainDetailsController
Course
Admin enters courseID
courseID is verified
acceptCourseID(courseID)
viewCourse(courseID)
Admin clicks on the option
to view the course details with the
specified courseID
viewCourse(courseID)
viewCourse(courseID)
Course details fetched
Course details
displayed successfully
Course details
displayed Successfully
courseDetails
Course details
displayed successfully
Course details
displayed Successfully
courseDetails
courseDetails

# EXPERIMENT-8

**Aim:** To design the activity diagram of the course management system



# EXPERIMENT-9

**Aim:** To design the test case matrices of course management system

1. Maintain student details

| **Test Case ID** | **Scenario and Description** | **Student ID** | **Name** | **Class** | **Section** | **Phone** | **Address** | **Mother's Name** | **Father's Name** | **Email** | **Update Confirmed** | **Deletion Confirmed** | **Expected Result** | **Remarks (if any)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TC$_1$ | Add a Student (All valid inputs) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Student is added successfully | - |
| TC$_2$ | Add a Student (Invalid Student ID) | Invalid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Student ID format incorrect or missing |
| TC$_3$ | Add a Student (Invalid Name) | Valid | Invalid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Name format incorrect or missing |
| TC$_4$ | Add a Student (Invalid Class) | Valid | Valid | Invalid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Class field missing or incorrect |
| TC$_5$ | Add a Student (Invalid Section) | Valid | Valid | Valid | Invalid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Section field missing or incorrect |
| TC$_6$ | Add a Student (Invalid Phone) | Valid | Valid | Valid | Valid | Invalid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Phone number format incorrect |
| TC$_7$ | Add a Student (Invalid Address) | Valid | Valid | Valid | Valid | Valid | Invalid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Address field missing |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TC$_8$ | Add a Student (Invalid Mother’s Name) | Valid | Valid | Valid | Valid | Valid | Valid | Invalid | Valid | Valid | n/a | n/a | Error message displayed | Mother’s name missing |
| TC$_9$ | Add a Student (Invalid Father’s Name) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Invalid | Valid | n/a | n/a | Error message displayed | Father’s name missing |
| TC$_{10}$ | Add a Student (Invalid Email) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Invalid | n/a | n/a | Error message displayed | Email format incorrect |
| TC$_{11}$ | Add a Student (Student Already Exists) | Existing Student ID | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Student ID already exists |
| TC$_{12}$ | Update a Student (All valid inputs) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Yes | n/a | Student details updated successfully | - |
| TC$_{13}$ | Update a Student (Invalid Student ID) | Invalid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Student ID format incorrect or missing |
| TC$_{14}$ | Update a Student (Student Not Found) | Non-existent Student ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Student ID |
| TC$_{15}$ | Update a Student (Update Cancelled) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Valid | No | n/a | Main screen appears | Update operation cancelled |
| TC$_{16}$ | Delete a Student | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Yes | Student record deleted | - |
| TC$_{17}$ | Delete a Student (Student Not Found) | Non-existent Student ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Student ID |
| TC$_{18}$ | Delete a Student (Delete Cancelled) | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | No | Main screen appears | Delete operation cancelled |  |
| TC$_{19}$ | View a Student | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Student details displayed | - |
| TC$_{20}$ | View a Student (Student Not Found) | Non-existent Student ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Student ID |

## 2. Maintain faculty details

| **Test Case ID** | **Scenario and Description** | **Faculty ID** | **Name** | **Department** | **Designation** | **Phone** | **Address** | **Email** | **Update Confirmed** | **Deletion Confirmed** | **Expected Result** | **Remarks (if any)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TC$_1$ | Add a Faculty Member (All valid inputs) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Faculty member is added successfully | - |
| TC$_2$ | Add a Faculty Member (Invalid Faculty ID) | Invalid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Faculty ID format incorrect or missing |
| TC$_3$ | Add a Faculty Member (Invalid Name) | Valid | Invalid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Name format incorrect or missing |
| TC$_4$ | Add a Faculty Member (Invalid Department) | Valid | Valid | Invalid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Department field missing or incorrect |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TC$_{5}$ | Add a Faculty Member (Invalid Designation) | Valid | Valid | Valid | Invalid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Designation field missing or incorrect |
| TC$_{6}$ | Add a Faculty Member (Invalid Phone) | Valid | Valid | Valid | Valid | Invalid | Valid | Valid | n/a | n/a | Error message displayed | Phone number format incorrect |
| TC$_{7}$ | Add a Faculty Member (Invalid Address) | Valid | Valid | Valid | Valid | Valid | Invalid | Valid | n/a | n/a | Error message displayed | Address field missing |
| TC$_{8}$ | Add a Faculty Member (Invalid Email) | Valid | Valid | Valid | Valid | Valid | Valid | Invalid | n/a | n/a | Error message displayed | Email format incorrect |
| TC$_{9}$ | Add a Faculty Member (Faculty Already Exists) | Existing Faculty ID | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Faculty ID already exists |
| TC$_{10}$ | Update a Faculty Member (All valid inputs) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | Yes | n/a | Faculty details updated successfully | - |
| TC$_{11}$ | Update a Faculty Member (Invalid Faculty ID) | Invalid | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Faculty ID format incorrect or missing |
| TC$_{12}$ | Update a Faculty Member (Faculty Not Found) | Non-existent Faculty ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Faculty ID |
| TC$_{13}$ | Update a Faculty Member (Update Cancelled) | Valid | Valid | Valid | Valid | Valid | Valid | Valid | No | n/a | Main screen appears | Update operation cancelled |
| TC$_{14}$ | Delete a Faculty Member | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Yes | Faculty record deleted | - |
| TC$_{15}$ | Delete a Faculty Member (Faculty Not Found) | Non-existent Faculty ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed |
| TC$_{16}$ | Delete a Faculty Member (Delete Cancelled) | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | No | Main screen appears | Delete operation cancelled |
| TC$_{17}$ | View a Faculty Member | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Faculty details displayed | - |
| TC$_{18}$ | View a Faculty Member (Faculty Not Found) | Non-existent Faculty ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Faculty ID |

## 3. Maintain course details

| Test Case ID | Scenario and Description | Course ID | Course Name | Department | Credits | Instructor | Schedule | Update Confirmed | Deletion Confirmed | Expected Result | Remarks (if any) |
|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TC$_1$ | Add a Course (All valid inputs) | Valid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Course is added successfully | - |
| TC$_2$ | Add a Course (Invalid Course ID) | Invalid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Course ID format incorrect or missing |
| TC$_3$ | Add a Course (Invalid Course Name) | Valid | Invalid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Course Name format incorrect or missing |
| TC$_4$ | Add a Course (Invalid Department) | Valid | Valid | Invalid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Department field missing or incorrect |
| TC$_5$ | Add a Course (Invalid Credits) | Valid | Valid | Valid | Invalid | Valid | Valid | n/a | n/a | Error message displayed | Credits field missing or incorrect |
| TC$_6$ | Add a Course (Invalid Instructor) | Valid | Valid | Valid | Valid | Invalid | Valid | n/a | n/a | Error message displayed | Instructor field missing or incorrect |
| TC$_7$ | Add a Course (Invalid Schedule) | Valid | Valid | Valid | Valid | Valid | Invalid | n/a | n/a | Error message displayed | Schedule field missing or incorrect |
| TC$_8$ | Add a Course (Course Already Exists) | Existing Course ID | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Course ID already exists |
| TC$_9$ | Update a Course (All valid inputs) | Valid | Valid | Valid | Valid | Valid | Valid | Yes | n/a | Course details updated successfully | - |
| TC$_{10}$ | Update a Course (Invalid Course ID) | Invalid | Valid | Valid | Valid | Valid | Valid | n/a | n/a | Error message displayed | Course ID format incorrect or missing |
| TC$_{11}$ | Update a Course (Course Not Found) | Non-existent Course ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Course ID |
| TC$_{12}$ | Update a Course (Update Cancelled) | Valid | Valid | Valid | Valid | Valid | Valid | No | n/a | Main screen appears | Update operation cancelled |
| TC$_{13}$ | Delete a Course | Valid | n/a | n/a | n/a | n/a | n/a | n/a | Yes | Course record deleted | - |
| TC$_{14}$ | Delete a Course (Course Not Found) | Non-existent Course ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Course ID |
| TC$_{15}$ | Delete a Course (Delete Cancelled) | Valid | n/a | n/a | n/a | n/a | n/a | n/a | No | Main screen appears | Delete operation cancelled |
| TC$_{16}$ | View a Course | Valid | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Course details displayed | - |
| TC$_{17}$ | View a Course (Course Not Found) | Non-existent Course ID | n/a | n/a | n/a | n/a | n/a | n/a | n/a | Error message displayed | No record found with this Course ID |

## 4. Login

| Test Case ID | Scenario and Description | Login ID | Password | Login Confirmed | Expected Result | Remarks (if any) |
|---|---|---|---|---|---|---|
| TC$_1$ | Successful Login (Valid credentials) | Valid | Valid | Yes | User logs in successfully | - |
| TC$_2$ | Unsuccessful Login (Invalid Login ID) | Invalid | Valid | Yes | Error message displayed | Invalid Login ID entered |
| TC$_3$ | Unsuccessful Login (Invalid Password) | Valid | Invalid | Yes | Error message displayed | Incorrect password entered |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| TC$_4$ | Unsuccessful Login (Both credentials invalid) | Invalid | Invalid | Yes | Error message displayed | Incorrect Login ID and password |
| TC$_5$ | Unsuccessful Login (Empty Login ID) | Empty | Valid | Yes | Error message displayed | Login ID field is empty |
| TC$_6$ | Unsuccessful Login (Empty Password) | Valid | Empty | Yes | Error message displayed | Password field is empty |
| TC$_7$ | Unsuccessful Login (Both fields empty) | Empty | Empty | Yes | Error message displayed | Both fields are empty |
| TC$_8$ | User exits before confirming login | Any | Any | No | System returns to the main screen | User chose not to proceed |

5. Downloading course material

| Test Case ID | Scenario Name and Description | Input 1: Course ID | Expected Output | Remarks (if any) |
|---|---|---|---|---|
| TC$_1$ | Scenario 1 — Download course material with valid input | Valid input | Course material displayed up to last lecture | Course ID exists and is registered by the student |
| TC$_2$ | Scenario 2 — Download alternative flow: Invalid course ID format | Invalid input | Error message: Invalid course ID | Course ID format is invalid |
| TC$_3$ | Scenario 2 — Download alternative flow: Course ID not registered | Valid input | Error message: Course not found | Course ID does not exist in student’s registrations |
| TC$_4$ | Scenario 3 — Download alternative flow: User exits before operation completion | n/a | User exits the system; no data downloaded | Operation aborted by user |

6. Upload course material

| Test Case ID | Scenario Name and Description | Input 1: Course ID | Input 2: File Size | Input 3: Deadline | Expected Output | Remarks (if any) |
|---|---|---|---|---|---|---|
| TC$_1$ | Scenario 1 — Upload course material with valid inputs | Valid input | Valid input | Valid input | File uploaded successfully | Course material updated in the database |
| TC$_2$ | Scenario 2 — Upload alternative flow: File too large | Valid input | Invalid input | Valid input | Error: File size exceeds allowed limit | Upload failed; reset to start |
| TC$_3$ | Scenario 3 — Upload alternative flow: Invalid course ID | Invalid input | Valid input | Valid input | Error: Invalid course ID | Course ID not found in faculty's registered courses |
| TC$_4$ | Scenario 4 — Upload alternative flow: User exits | n/a | n/a | n/a | Upload canceled by user | Upload process aborted by faculty |

7. Make submissions

| Test Case ID | Scenario Name and Description | Input 1: Course ID | Input 2: File Size | Input 3: Deadline Passed | Expected Output | Remarks (if any) |
|---|---|---|---|---|---|---|
| TC$_5$ | Scenario 1 — Submit assignment with valid inputs | Valid input | Valid input | No | Submission successful | File stored in submission database |
| TC$_6$ | Scenario 2 — Submission alternative flow: File too large | Valid input | Invalid input | No | Error: File too large | Upload canceled and reset |
| TC$_7$ | Scenario 3 — Submission alternative flow: Submission after deadline | Valid input | Valid input | Yes | Error: Deadline has passed | Deadline enforcement failed submission |
| TC$_8$ | Scenario 4 — Submission alternative flow: Invalid course ID | Invalid input | Valid input | No | Error: Invalid course ID | Course not available to student |

8. Accept and grade submissions

| Test Case ID | Scenario Name and Description | Input 1: Course ID | Input 2: Student ID | Input 3: Grade | Expected Output | Remarks (if any) |
|---|---|---|---|---|---|---|
| TC$_9$ | Scenario 1 — Grade submissions with valid inputs | Valid input | Valid input | Valid input | Grades updated successfully | Submission record modified with grade |
| TC$_{10}$ | Scenario 2 — Grading alternative flow: Invalid course ID | Invalid input | Valid input | Valid input | Error: Invalid course ID | Course ID not in faculty's allocation |
| TC$_{11}$ | Scenario 3 — Grading alternative flow: Invalid student ID | Valid input | Invalid input | Valid input | Error: Invalid student ID | Student not enrolled in course |
| TC$_{12}$ | Scenario 4 — Grading alternative flow: User exits | n/a | n/a | n/a | Action canceled; returns to main menu | Faculty opted out of grading process |

9. Upload assignment

| Test Case ID | Scenario Name and Description | Input 1: Faculty ID | Input 2: Course ID | Input 3: File Size | Expected Output | Remarks |
|---|---|---|---|---|---|---|
| TC$_1$ | Upload assignment – basic flow | Valid input | Valid input | Valid input | Assignment uploaded | Faculty is allowed to upload |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | successfully | assignment |
| TC$_2$ | Upload assignment – invalid faculty | Invalid input | Valid input | Valid input | Faculty not authorized | Membership not valid |
| TC$_3$ | Upload assignment – invalid courseID | Valid input | Invalid input | Valid input | Course ID invalid | Course does not exist |
| TC$_4$ | Upload assignment – file too large | Valid input | Valid input | Invalid input | Upload failed: file too large | File exceeds allowed size |
| TC$_5$ | Upload assignment – user exits | Valid/Invalid input | Valid/Invalid input | Valid/Invalid input | User exits the process | System unchanged |

10. Calculate average performance

| Test Case ID | Scenario Name and Description | Input 1: Faculty ID | Input 2: Course ID | Expected Output | Remarks |
|---|---|---|---|---|---|
| TC$_1$ | Calculate average – basic flow | Valid input | Valid input | Average displayed | Average computed from uploaded grades |
| TC$_2$ | Calculate average – no grades uploaded | Valid input | Valid input | Cannot calculate average | No data in gradebook |
| TC$_3$ | Calculate average – invalid courseID | Valid input | Invalid input | Course ID invalid | Course not found |
| TC$_4$ | Calculate average – user exits | Valid/Invalid input | Valid/Invalid input | User exits the process | System unchanged |

11. View attendance

| Test Case ID | Scenario Name and Description | Input 1: Student ID | Input 2: Course ID | Expected Output | Remarks |
|---|---|---|---|---|---|
| TC$_1$ | View attendance – basic flow | Valid input | Valid input | Attendance displayed | Shows records up to last lecture |
| TC$_2$ | View attendance – invalid courseID | Valid input | Invalid input | Course ID invalid | Course does not exist |
| TC$_3$ | View attendance – user exits | Valid/Invalid input | Valid/Invalid input | User exits the process | No data displayed |

12. View classmate details

| Test Case ID | Scenario Name and Description | Input 1: Student ID | Input 2: Course ID | Expected Output | Remarks |
|---|---|---|---|---|---|
| TC$_1$ | View classmates – basic flow | Valid input | Valid input | Classmate details displayed | Email IDs and names shown |

| | | | | | |
|---|---|---|---|---|---|
| TC$_2$ | View classmates – invalid courseID | Valid input | Invalid input | Course ID invalid | Cannot fetch classmate list |
| TC$_3$ | View classmates – user exits | Valid/Invalid input | Valid/Invalid input | User exits the process | No action taken |

13. Generate reports

| Test Case ID | Scenario Name and Description | Input 1: Admin ID | Input 2: Report Type | Expected Output | Remarks |
|---|---|---|---|---|---|
| TC$_1$ | Generate reports – basic flow | Valid input | Valid input | Report generated | Attendance, grades, or other selected data |
| TC$_2$ | Generate reports – unauthorized access | Invalid input | Valid input | Unauthorized admin | Admin ID not validated |
| TC$_3$ | Generate reports – user exits | Valid/Invalid input | Valid/Invalid input | User exits | No report generated |