RISE-EDU

Software Requirements Specification

Revision History

Date	Revision	Description	Author
09/17/2025	1.0	Initial Version	Wail Mohammed
09/23/2025	1.1	Purpose, Scope, Definitions Updated	Wail, Emmanuel, Shichang, Yesenia
09/24/2025	1.2	Assumptions and Internal/External Requirements Update	Wail
09/25/2025	1.3	Assumptions and Course Planner Non-Functional Environmental Requirement Update	Shichang
09/25/2025	1.4	Functional Student Module Requirements Non-Functional Performance Requirements	Emmanuel
09/28/2025	1.5	Added a Use Case	Emmanuel
9/29/2025	1.6	Added Requirements	Yesenia Ruiz
09/29/2025	1.7	Updated Two Use Cases	Wail Mohammed
09/29/2025	1.8	Updated Two Use Cases-Course Planner/Reporting	Shichang Wang
09/29/2025	1.9	Updated UCs: 01,02,03,04,05,06,07,08,09	Wail Mohammed
9/30/2025	1.10	Added Use Cases 12 and 13	Yesenia Ruiz
09/30/2025	1.11	Added Use Case:10, 11-Report module	Shichang Wang
10/01/2025	1.12	Formatting and final edits	Yesenia Ruiz
10/01/2025	1.13	Updated, scope, project perspective, constraints, common requirements, student, administrator and reporting modules. Updated use cases depending on updated scope.	Wail Mohammed

Table of Contents

1. Purpose

This document outlines the requirements for the College Course Enrollment System Project.

1.1. Scope

This document will catalog the user, system, and hardware requirements for the CCES (College Course Enrollment) system. It will not, however, document how these requirements will be implemented. It will allow school administrators to create and manage the school's course schedule, while also providing students with tools to enroll, drop, and withdraw from courses. The system will also manage prerequisites for courses and allow users to waitlist if class sizes are full.

1.2. Definitions, Acronyms, Abbreviations

1.2.1 CCES: College Course Enrollment System

1.2.2 Student User: User that will be able to enroll, drop, waitlist and withdraw from classes.

1.2.3 Administrator: School admin user that will be responsible for managing course listings.

1.2.4 TCP/IP: A piece of software suite that will allow communication between client and server.

1.2.5 Client: Users interact with the client entity to be able to send and receive information from the server.

1.2.6 Server: The server is responsible for listening to and interacting with multiple clients at the same time, and managing information being received from the clients.

1.2.7 Add/Drop period is the same as registration period.

1.2.8 Withdrawal period is another period when student will be allowed to withdraw from a course.

1.3. References

Use Case Specification Document

UML Use Case Diagrams Document

Class Diagrams

Sequence Diagrams

1.4. Overview

The CCES (College Course Enrollment System) allows for the creation of college course schedules by administrators and allows students to enroll in these courses. The system will support class sizes, waiting lists, prerequisites, and reports. This system supports a network of universities, students, and Administrators. This is a Java application with a GUI that operates over TCP/IP. This system requires a server application and a client application. There is no web or HTML component.

2. Overall Description

2.1. Product Perspective

The CCES system is a platform designed for university students and administrators. Administrators can control the number of courses, class size, waiting list size, prerequisites list, and issue reports. Students can enroll in courses, drop courses, and withdraw from courses. The logging module enables administrators to issue reports and for students to display their class schedules.

2.2. Product Architecture

The system will be organized into three major modules: the student module, the administrator module, and the Reporting module.

2.3. Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

2.3.1 The enrollment system must have an authentication, authorization, and account creation system that differentiates between school admins and students based on their existing student or admin IDs.

2.3.2 Students will be able to access their online accounts and input their name, email address, major, and school year. Similarly, admins will be able to access their online accounts and input their name, email address.

2.3.3 Students must be able to view college courses' attributes, such as class IDs, names, times, days, instructor names, number of units, credit/no credit, number of available seats, and available waitlists.

2.3.4 School admin users must be able to access a student user’s class information and add holds or update priority registration to that particular student object instance.

2.3.5 The system must allow admins to store and update a list of courses with all course class attributes.

2.3.6 The system must allow students to view courses, check if students are eligible to enroll (no holds), drop, waitlist, and/or withdraw from courses. The system must not give any students access to other students’ accounts.

2.3.7 The system must allow admins only to add/edit and delete courses and drop students if needed.

2.3.8 The system must be available to all users (students and admins) during course enrollment periods and take into account earlier registration for students who have a higher priority.

2.3.9 The system must be able to update course info as well as students' records in real time. Admins must also be able to see the up-to-date student and class records.

2.3.10 The system must be large enough to handle and be able to store student account data, the list of all active courses, and the list of all waitlisted courses.

2.4. Constraints

2.4.1. The system will use TCP/IP but will not be accessible by web-browser.

2.4.2. The data will be managed and stored in a text file, not in a database.

2.4.3 No payments will be handled in this project scope.

2.4.4. The application will only be based on Java programming language.

2.4.5 The application will not use JSON style formats to send and receive data between client and server.

2.5. Assumptions and Dependencies

2.5.1. All users will have network access to connect to the system.

2.5.2. It is assumed that there will be two user roles within the system. The school administrator and the student role.

2.5.3. It is assumed that school administrators will maintain course addition, deletion, and maintain prerequisite requirements for courses that need it.

2.5.4. It is assumed that students are associated with one university.

2.5.5. It is assumed that each school has unique courses.

2.5.6 It is assumed that the maximum number of users at a given time is 15,000.

2.5.7 students and administrators will not share their login information.

2.5.8 Students will be responsible for checking their enrollment status, if the student is on the waitlist for the course.

2.5.9 Student accounts created based on correct personal information.

2.5.10 Students will be notified of the enrollment start date.

2.5.11 Students will complete their enrollment within the deadlines.

2.5.12 The reporting module will only generate reports based on data in the system.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Users will be given a standardized login credentials for students and administrators.

3.1.1.2 Student user's login info will be the uppercase letter 'S' along with their first name and last name and last 2 digits of their year of birth, e.g. firstnamelastnameYY.

3.1.1.3 Administrator user's login info will be the uppercase letter 'A' along with their first name and last name and last 2 digits of their year of birth, e.g. firstnamelastnameYY.

3.1.1.4 All Users are given a password that is be 12 characters long. It will start with 1 uppercase letter, followed by 10 digits and 1 symbol.

3.1.1.5 The system will recognize and differentiate between student users and administrator users using their first character of their login username.

3.1.1.6 The system will store and maintain the user login credentials in a text document.

3.1.2. The Student Module Requirements:

3.1.2.1 Students shall be allowed access into the school's system only by entering their provided username and password.

3.1.2.2 Students shall be allowed to look up courses by Course names, Course ID, course time, instructor name, class location, class size the semester those courses are offered.

3.1.2.3 The system shall enforce all course prerequisites are satisfied before successfully enrolling a student user in a course.

3.1.2.4 The system shall not allow students users to enroll in a course if a hold is applied on their account.

3.1.2.5 The system shall allow students to check the availability of seats in the class during course registration periods.

3.1.2.6 The system shall ask student user if they want to add a class or drop a class during add/drop period.

3.1.2.7 The system shall ask student user if they want to be placed on waitlist when class size is full, during add/drop period.

3.1.2.8 The system shall ask student user if they want to withdraw from a class during withdrawal period.

3.1.3. The School Administrator Module Requirements:

3.1.2.1 The administrator is the only user that will be allowed to manage courses stored in a course list.

3.1.2.2 The administrator will be allowed to add courses, delete courses, and edit courses stored in a course list.

3.1.2.3 The administrator will be allowed to enroll a student in a course, drop a student from a course, as well as withdraw a student from a course only during add/drop period or withdrawal period.

3.1.2.4 The administrator will be allowed to place a hold on a student's account.

3.1.2.5 The administrator will be allowed to see which students are enrolled in the course and who is on a waiting list.

3.1.4. Reporting Module Requirements:

3.1.4.1 The system should keep track of every enrollment action that student user makes, such as add, drop, waitlist, withdraw for each course.

3.1.4.2 The system will save the updated courses info associated with every student's account with timestamps.

3.1.4.3 The system shall allow school administrators to display and save a report that show how many students are enrolled in a specific course and how many are placed on waitlist.

3.1.4.4 The system shall allow school administrators to display a student's past course enrollments.

3.1.4.5 The system shall allow all users to view a confirmation message after every successful change.

3.1.4.6 The system also needs to display error messages whenever students make invalid input.

3.1.4.7 The system will generate a general report that includes all previous and current course enrollments for a specific student upon user request.

3.2. External Interface Requirements

3.2.1 The system must provide an interface to the University billing system administered by the Bursar's office so that students can be automatically billed for the courses in which they have enrolled. The interface is to be in a comma-separated text file containing the following fields: student id, course id, semester id, action. Where "action" is whether the student has added or dropped the course. The file will be exported nightly and will contain new transactions only.

3.2.2 The system interface must be clear, intuitive and provide a consistent user experience.

3.2.3 The system must provide error messages when invalid actions take place.

3.3. Internal Interface Requirements

3.3.1 The system must use a client-server communication via TCP/IP network.

3.3.2 The system must process a data-feed from the enrollment system such that the student updated course records are stored in the form of a comma-separated interface file that is exported from the enrollment system upon user's demand.

3.3.3 The data feed from the enrollment system must have the following fields included in the file; student name, student id, major, school name, name and number of all courses enrolled and withdrawn.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 The system must encrypt data being transmitted over the Internet.
4.1.2 The system must not allow students to view each other's data, meaning only the student should be able to view their own data.
4.1.2 The system must allow administrators to only view necessary student information, not where they live and other data like that.

4.2. Environmental Requirements

4.2.1 The system shall run on the university's existing Linux-based server infrastructure.
4.2.2 The system shall operate properly with the university's existing network and power infrastructure.
4.2.3 The system shall not require high-performance machines; any standard desktop used by students or administrators will be sufficient to run the client.
4.2.4 The system shall be deployable on existing university servers without requiring hardware upgrades.
4.2.5 The system shall remain compatible with future minor upgrades to the university's operating systems.

4.3. Performance Requirements

4.3.1 The system shall accommodate 15,000 users during the peak usage time window of 8:00 am to 12 pm local time.
4.3.3 The system shall display the status of course enrollments, dropped courses, waitlisted positions after the operation is carried out.
4.3.4 The system shall show a list of available courses within 5 seconds when a student tries to search through.

5. Use Case Specifications Document

Use Case ID: UC01

Use Case Name: Manage User Login

Relevant Requirements:

- Requirements Document ID: 2.3.1, 3.1.1, 4.1

Primary Actor:

- Student
- School Administrator

Pre-conditions:

- The server is online, and the user client is connected to the server.
- The user must have an existing account and be active.
- The user credentials are stored in a data file.
- The client has access to the user list.

Post-conditions:

- User credentials are checked.
- Session is created.
- Appropriate user roles and access are given to students and Administrators.

Basic Flow or Main Scenario:

1) The user navigates to the login page.
2) The user is prompted to enter the username and password.
3) The user enters the username and password, then hits submit.
4) The system verifies the credentials.
5) The system assigns a role (Administrator or student) based on the account type provided in the login credential document
6) If this is the first time the user logs in or if a password reset is needed:
    a. The system redirects to the account management page.
    b. The user must either change their default username and/or password per system security requirements.
7) The system validates the changes and updates the user records.
8) The system creates an authenticated session.
9) The system redirects the user to the main dashboard.

Extensions or Alternate Flows:

- Invalid input at log in:
  If the user enters an invalid current credentials, per the system's security requirements, the system displays an error message, "Invalid Username or password", and asks the user to retry.
- Invalid input at update page:
  If the user enters an invalid username or password per the system's security requirements, the system displays an error message "Invalid Username or Password" and asks the user to retry.

Exceptions:

- If the authentication system or server is unavailable, an error message is displayed to the user.

Related Use Cases:

- UC02 Course Enrollment.

Use Case ID: UC02
Use Case Name: Course Enrollment
Relevant Requirements:

- Requirements Document ID: 2.3.6

Primary Actor:

- Student.
- Administrator

Pre-conditions:

- The server is online, and the user client is connected to the server.
- The user must be logged into an active session in the system.
- Course registration add/drop period is active.

Post-conditions:

- The student is enrolled in selected active courses.

Basic Flow or Main Scenario:

1) The user navigates to the "Class Enrollment" page.
2) The system checks if the course registration date is active.
3) The system displays the current active classes for the current semester.\
4) The user selects the course to be enrolled in.
5) If the selected course day and time is not equal to another existing enrolled course day and time:
    a. The system allows the user to proceed to the next step.
    b. Otherwise, the system prompts the user with a time conflict message and redirects back to the previous step(3).
6) The system checks for students' holds, prerequisite courses, and available spots.
7) If system checks are all valid:
    a. The student is enrolled, and a confirmation message is shown to the user.
    b. Otherwise, the system prompts the user with an error message and redirects back to the previous step(3).

Extensions or Alternate Flows:

- If the user is a school Administrator, the system allows the same basic flow.

Exceptions:

- If the class management system or server fails, an error message is displayed to the user.

Related Use Cases:

- UC01 User Login

Use Case ID: UC03
Use Case Name: Course Drop
Relevant Requirements:

- Requirements Document ID: 2.3.6.

Primary Actor:

- Student.
- Administrator

Pre-conditions:

- The server is online, and the user client is connected to the server.
- The user must be logged into an active session in the system.
- Course registration add/drop period is active.
- The student must be enrolled in at least one class.

Post-conditions:

- The student is dropped from selected active courses.

Basic Flow or Main Scenario:

1) The user navigates to the "Class Enrollment" page.
2) The system checks if the course registration add/drop period is active.
3) The system displays the student's current active enrolled classes for the current semester.
4) The user selects the course to be dropped.
5) The system displays a confirmation message before dropping the class.
6) If the user confirms, the student is dropped from selected class or classes, and a confirmation message is shown to the user.

Extensions or Alternate Flows:

- If the user is a school Administrator, the system allows the same basic flow.

Exceptions:

- If the class management system or server fails, an error message is displayed to the user.

Related Use Cases:

- UC01 User Login
- UC02 Course Enrollment

Use Case ID: UC04
Use Case Name: Course Waitlist
Relevant Requirements:

- Requirements Document ID: 3.1.4.1, 3.1.4.3, 4.3.3

Primary Actor:

- Student.
- Administrator

Pre-conditions:

- The server is online, and the user client is connected to the server.
- The user must be logged into an active session in the system.
- Course registration add/drop period is active.
- Available spots in a selected course/courses exceed the maximum class size.

Post-conditions:

- The student is added to a waiting list for the selected active course/courses.

Basic Flow or Main Scenario:

1) The user navigates to the “Class Enrollment” page.
2) The system checks if the course registration add/drop period is active.
3) The user selects the course to be added.
4) The system checks for students' holds, prerequisite courses, and available spots.
5) The system detects that available slots exceed the maximum class size.
6) The system displays an error message and prompts the user to be waitlisted.
7) If the user confirms, the student is added to the waitlist for the selected class or classes, and a confirmation message is shown to the user.

Extensions or Alternate Flows:

- If the user is a school Administrator, the system allows the same basic flow.

Exceptions:

- If the class management system or server fails, an error message is displayed to the user.

Related Use Cases:

- UC01 User Login
- UC02 Course Enrollment

Use Case ID: UC05
Use Case Name: Course Prerequisites Check
Relevant Requirements:

- Requirements Document ID: 3.1.12.3

Primary Actor:

- Student.
- Administrator

Pre-conditions:

- The server is online, and the user client is connected to the server.
- The user must be logged into an active session in the system.
- Course registration add/drop period is active.
- Selected course/courses have prerequisites.

Post-conditions:

- The user is allowed or denied to sign up or waitlist courses.

Basic Flow or Main Scenario:

1) The user selects the course to be added.
2) The system checks the student's completed course list.
3) The system checks the required prerequisites list for the selected course.
5) If prerequisites are found in the student's completed course list:
    a) The system displays "All prerequisites have been met for this course" and allows the user to proceed with UC03 and UC05.
    b) The system displays an error message and prompts the user to exit.

Extensions or Alternate Flows:

- If the user is a school Administrator, the system allows the same basic flow.

Exceptions:

- If the class management system or server fails, an error message is displayed to the user.

Related Use Cases:

- UC01 User Login
- UC02 Course enrollment
- UC04 Course Waitlist

Use Case ID: UC06
Use Case Name: Access Student Class Schedule
Relevant Requirements:

- Requirements Document ID: 3.1.4.3, 3.1.4.4, 3.1.4.7

Primary Actor:

- Student.

Pre-conditions:

- The server is online, and user client is connected to server.
- The user must have an existing account.
- The student must be logged into an active session in the system.

Post-conditions:

- The student can view current enrolled classes for the semester at any time.The student must be logged in to the system.
- The student can view course name, course day and time, instructor name, location.

Basic Flow or Main Scenario:

1) The student navigates to the "Class Schedule" page.
2) The system retrieves the student's current enrolled classes if any, for the current semester.
3) The system displays the course name, course ID, course day and time, instructor name, class location.

Extensions or Alternate Flows:

- If the student is not enrolled in any classes, the system displays a message to the user "No Current Enrollments Found".
- The student can download or print the course schedule.

Exceptions:

- If the class management system or server fails, an error message is displayed to the user.

Related Use Cases:

- UC01 User Login
- UC03 Course Drop
- UC04Waitlist Management

Use Case ID: UC07
Use Case Name: Manage Student Hold
Relevant Requirements:

- Requirements Document ID: 2.3.4, 3.1.2.4

Primary Actor:

- Administrator

Pre-conditions:

- The server is online, and the user client is connected to the server.
- The user must be logged into an active session in the system.
- The student account must exist and have a hold.

Post-conditions:

- The student's account hold is removed.

Basic Flow or Main Scenario:

1) The administrator navigates to the student's account.
3) The administrator removes the hold from the student's account.
5) The system sends the user a confirmation message, and the student account hold is removed.

Extensions or Alternate Flows:

- If the user is a student and tries to remove an account hold, an error message is displayed to let the user know of the error.

Exceptions:

- If the class management system or server fails, an error message is displayed to the user.

Related Use Cases:

- UC01 User Login
- UC02 Course enrollment
- UC04 Course Waitlist

Use Case ID: UC08
Use Case Name: Enrollment Reporting
Relevant Requirements:

- Requirements Document ID: FR 3.1.4.1 , FR 3.1.4.2 , FR 3.1.4.4 , FR 3.1.4.6

Primary Actor:

- Administrator

Pre-conditions:

- The Administrator is already logged in to the system.
- Course Enrollment data exists.

Post-conditions:

- The report is displayed on the screen.
- The Administrator can export the report.

Basic Flow or Main Scenario:

1) The Administrator navigates to the Reports page.
2) The system shows some categories like term, department, or course selection for the admin to pick.
3) The Administrator selects courses and chooses to generate reports.
4) The system gathers enrollment and waitlist data.
5) The system puts the data together and displays the report.
6) The Administrator chooses the Export Report.
7) The System generates a PDF of the report for download.

Extensions or Alternate Flows:

- If the Administrator didn't choose any course, the system will prompt the Administrator to choose at least one course.
- If there is no student in the waiting list, the report still shows a waiting list with 0.

Exceptions:

- If the system can't access the enrollment data, the system will display the error message and record the error.

Related Use Cases:

- UC01 Manage User Login
- UC02 Course Enrollment
- UC03 Course Drop
- UC06 Course Waitlist

Use Case ID: UC09
Use Case Name: Real Time Report Update
Relevant Requirements:

- Requirements Document ID: FR 3.1.4.1 ， FR 3.1.4.2 ， FR 3.1.4.3 ， FR 3.1.4.4

Primary Actor:

- Report module

Pre-conditions:

- The server is online, and the system is running normally.
- At least one student made an action.

Post-conditions:

- Report modules update immediately to display the latest status.
- Student's schedule updates immediately.

Basic Flow or Main Scenario:

1) Students make an enrollment action (add, drop and waiting list).
2) The system records the transaction with student information (ID, course number and timestamp).
3) The report module will display the updated data and show the new status of the action.
4) On the student's schedule page, the latest status updates.

Extensions or Alternate Flows:

- If more than one enrollment action made at same time, the system will group them and update together.
- If the update takes a long time, we will display a "loading…" message until the update is finished.

Exceptions:

- If the Report module can't reach enrollment data, the page keeps the last known information and displays an error message "System is temporarily unavailable".

Related Use Cases:

- UC01 User Login
- UC02 Course Enrollment
- UC03 Course Drop
- UC04 Course Waitlist
- UC9 Enrollment Reporting
- UC07 Hold Management

Use Case ID: UC10
Use Case Name: Create courses
Relevant Requirements:
- Requirements Document ID: 2.3.7, 3.1.2.2

Primary Actor:
- Administrator

Pre-conditions:
- The server is online, and the user client is connected to the server.
- The administrator must have access to the system.
- The administrator is registered in the system as an administrator.

Post-conditions:
- A new course is added into the course catalog.

Basic Flow or Main Scenario:
1. The administrator navigates to the catalog of courses page.
2. The administrator selects to create a new course.
3. The administrator enters in class size, subject, room number, professor's name, and the date and times the class will take place.
4. The administrator submits the new course information.
5. The administrator publishes the course after revision.
6. The administrator is redirected by the system to view the published course.

Extensions or Alternate Flows:
- Invalid information for new course - professor already teaching class
  If the administrator enters a professor's name but the professor is already teaching a diff course at the same time then the system will highlight the professor's name and the time the course is in red.
- Invalid information for new course - dates don't align with semester schedule.
  If the administrator enters a date that does not match with the semester dates, then the system will highlight those dates in red.

Exceptions:
- If the server is unavailable, an error message is displayed to the user and is prompted to retry again.
- If the system doesn't allow the administrator to publish the course, then the administrator will be given the option to open a ticket for support.

Related Use Cases:
- UC01 Login

Use Case ID: UC11
Use Case Name: Editing courses
Relevant Requirements:
- Requirements Document ID: 3.1.2.2

Primary Actor:
- Administrator

Pre-conditions:
- The server is online, and the user client is connected to the server.
- The administrator must have access to the system.
- The administrator is registered in the system as an administrator.

Post-conditions:
- A course has been edited in the course catalog.

Basic Flow or Main Scenario:
1. The administrator navigates to the catalog of courses page.
2. The administrator selects to edit a course.
3. The administrator can edit any of the following class size, subject, room number, professor's name, and the date and times the class will take place.
4. After the administrator is done editing the course they will submit the new course.
5. The administrator then will review the information changed and publish the course.
6. The system will redirect the administrator to the course that was just edited and is now republished in the catalog.

Extensions or Alternate Flows:
- Invalid information for editing a course - professor already teaching class
  If the administrator enters a professor's name but the professor is already teaching a diff course at the same time, then the system will highlight the professor's name and the time the course is in red.
- Invalid information for editing a course - dates don't align with the semester schedule.
  If the administrator enters a date that does not match with the semester dates, then the system will highlight those dates in red.

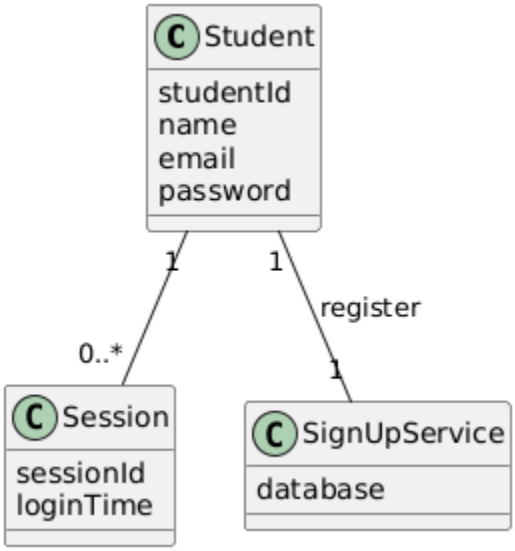
Exceptions:
- If the server is unavailable, an error message is displayed to the user and is prompted to retry again.
- If the system doesn't allow the administrator to publish the course, then the administrator will be given the option to open a ticket for support.

Related Use Cases:
- UC01 Login
- UC11 Create Course

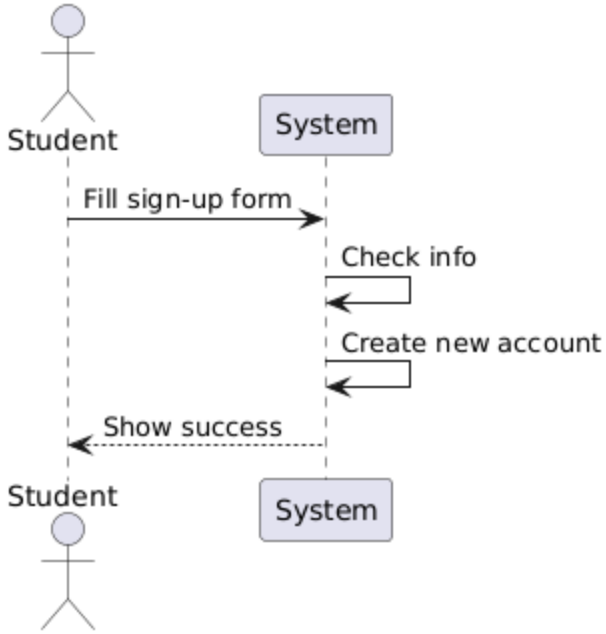
6. UML Use Case Diagrams

Use Case 1:

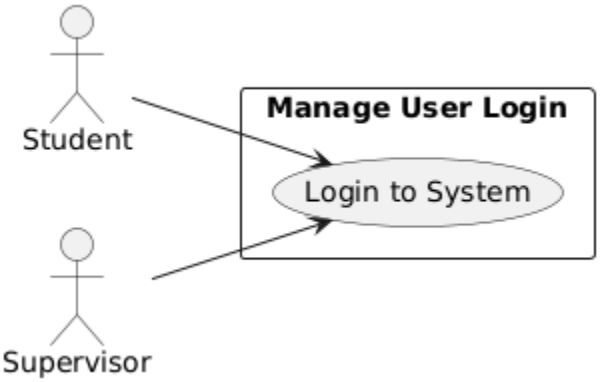
UML Diagram



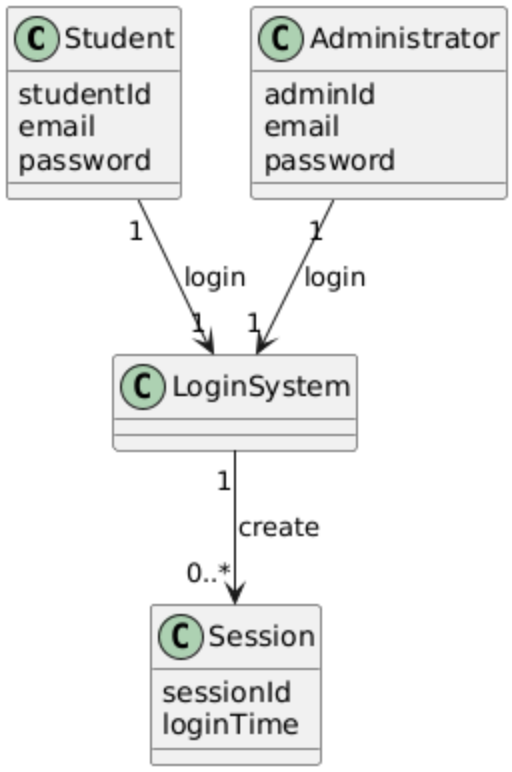
Sequence Diagram:



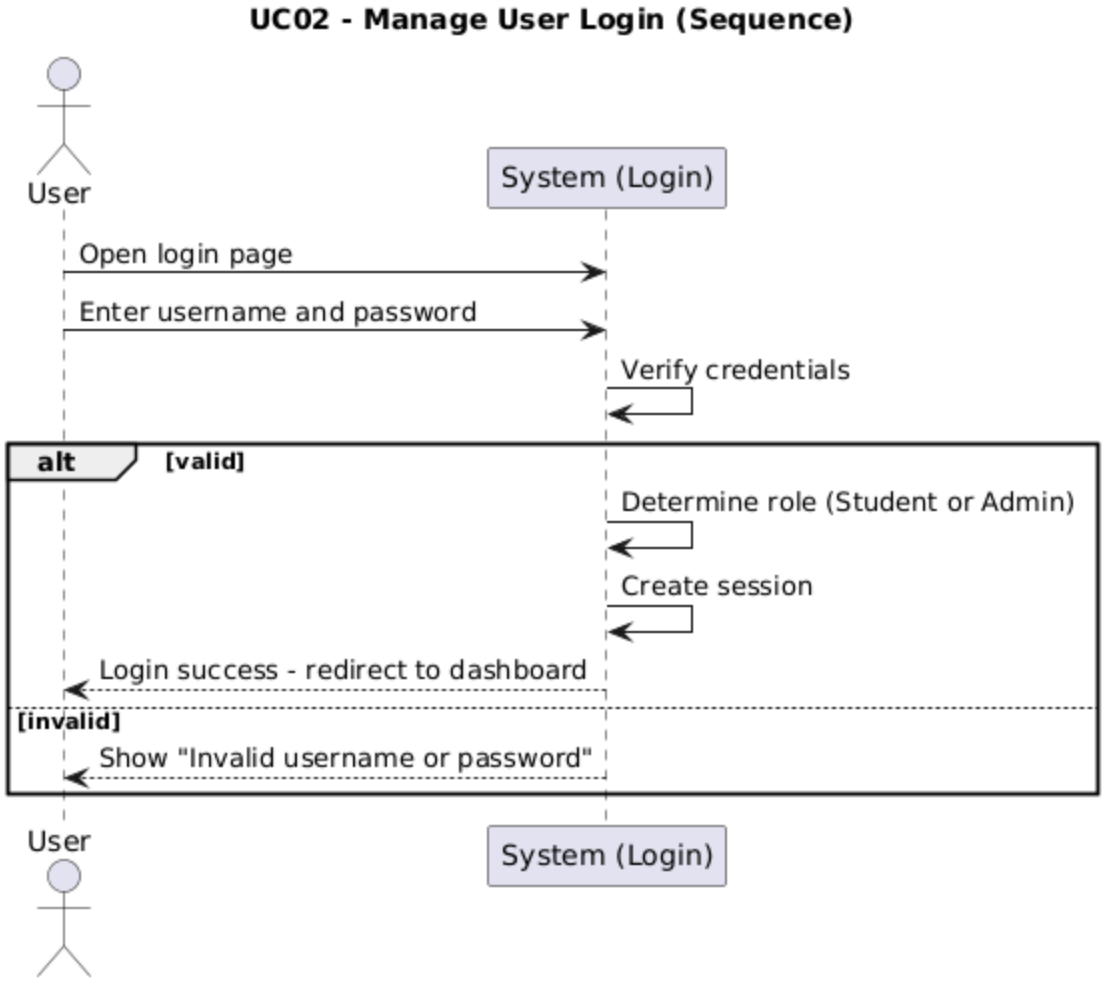
Use Case 2:



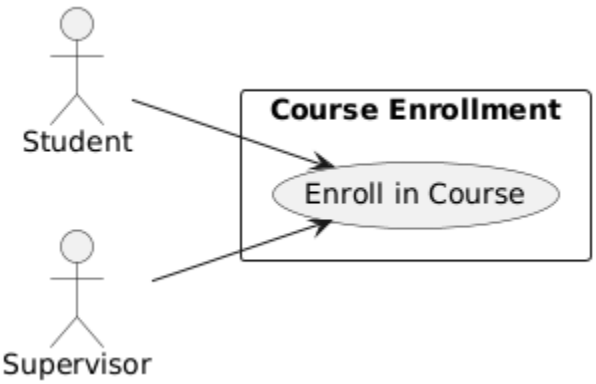
UML Diagram：



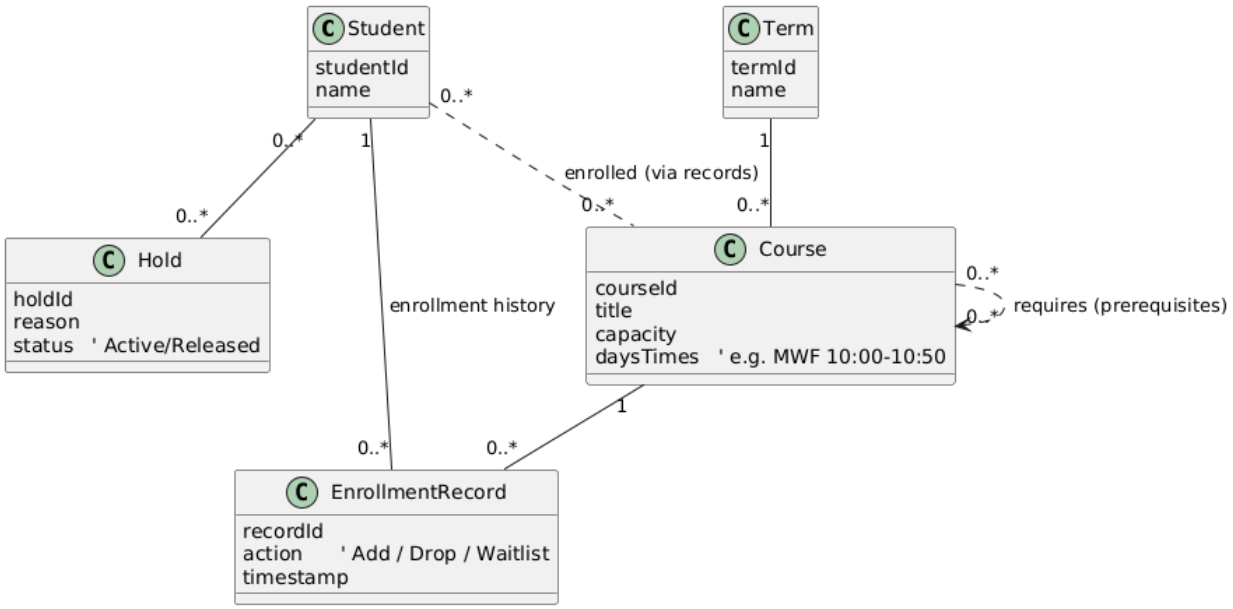
Sequence Diagram:



Use Case 3:

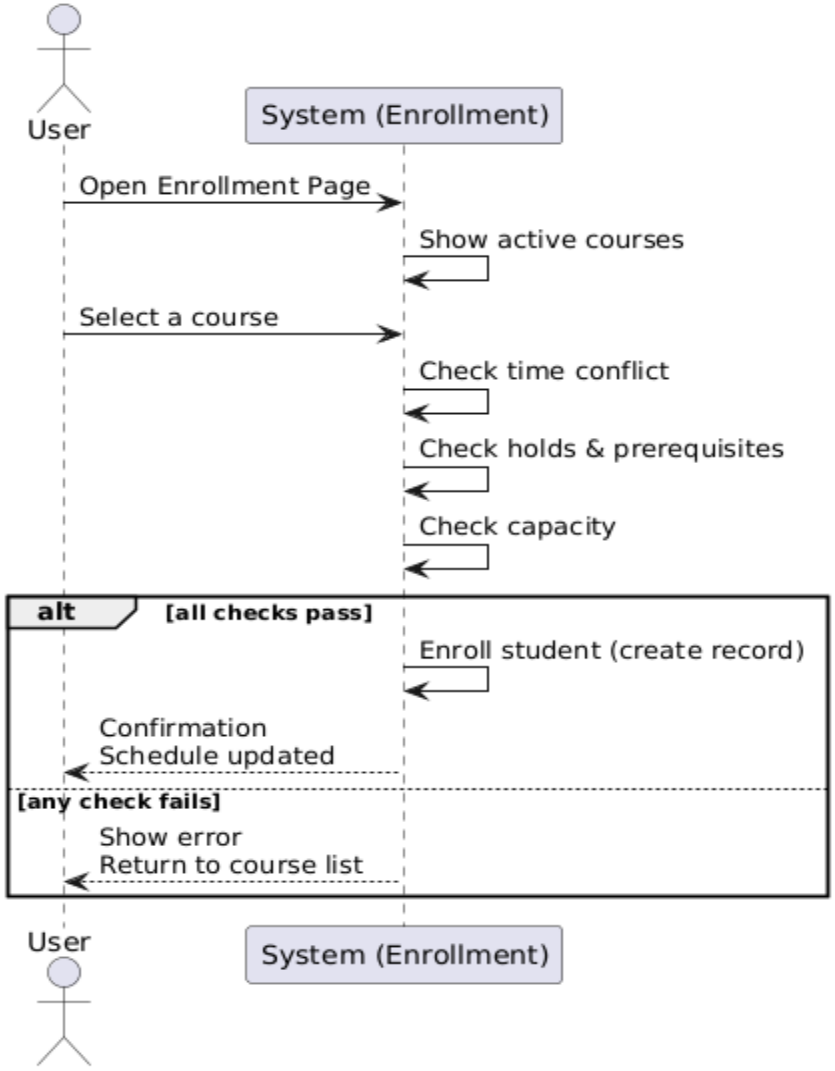


Class diagram:

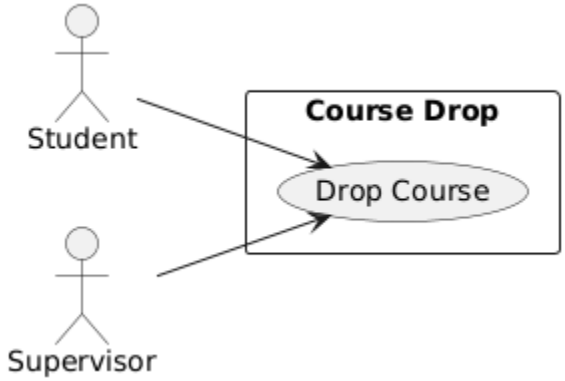


Sequence Diagram:

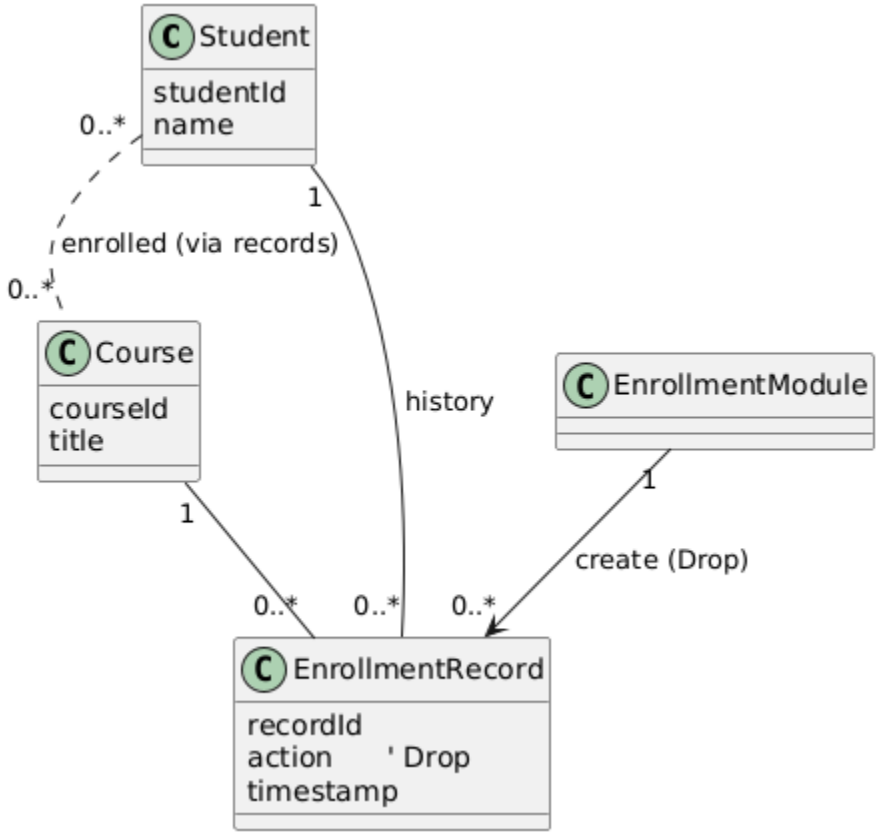
UC03 - Course Enrollment (Sequence, strict)



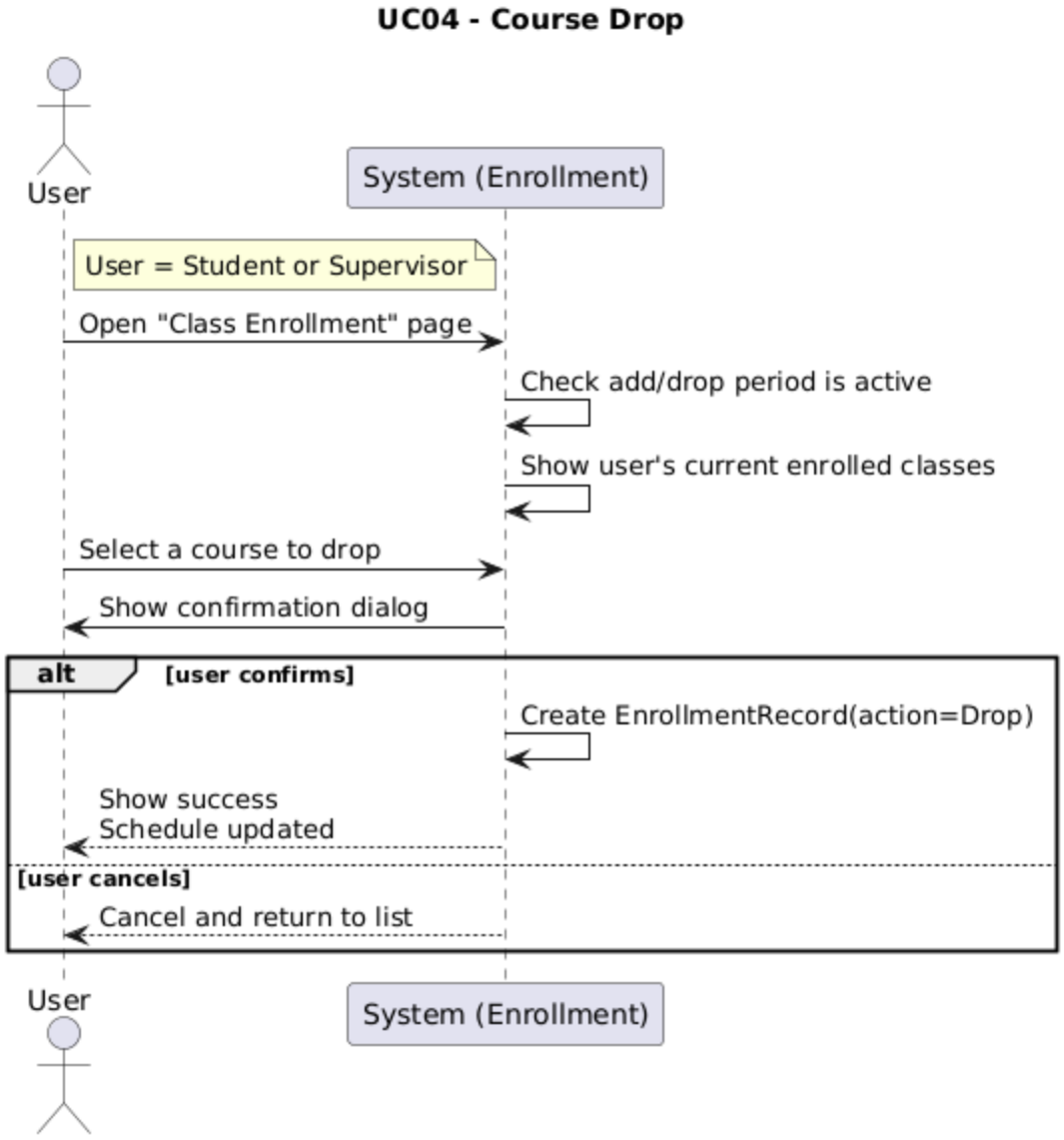
Use Case 4:



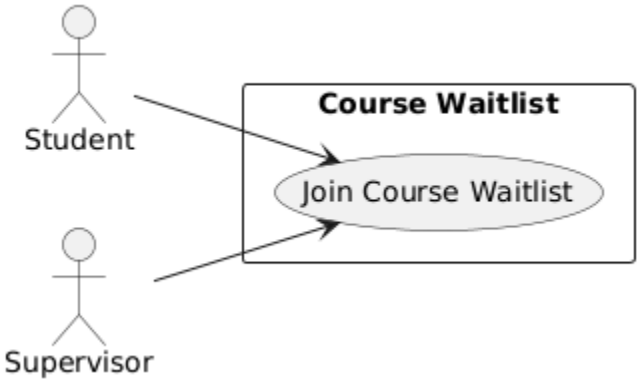
Class diagram :



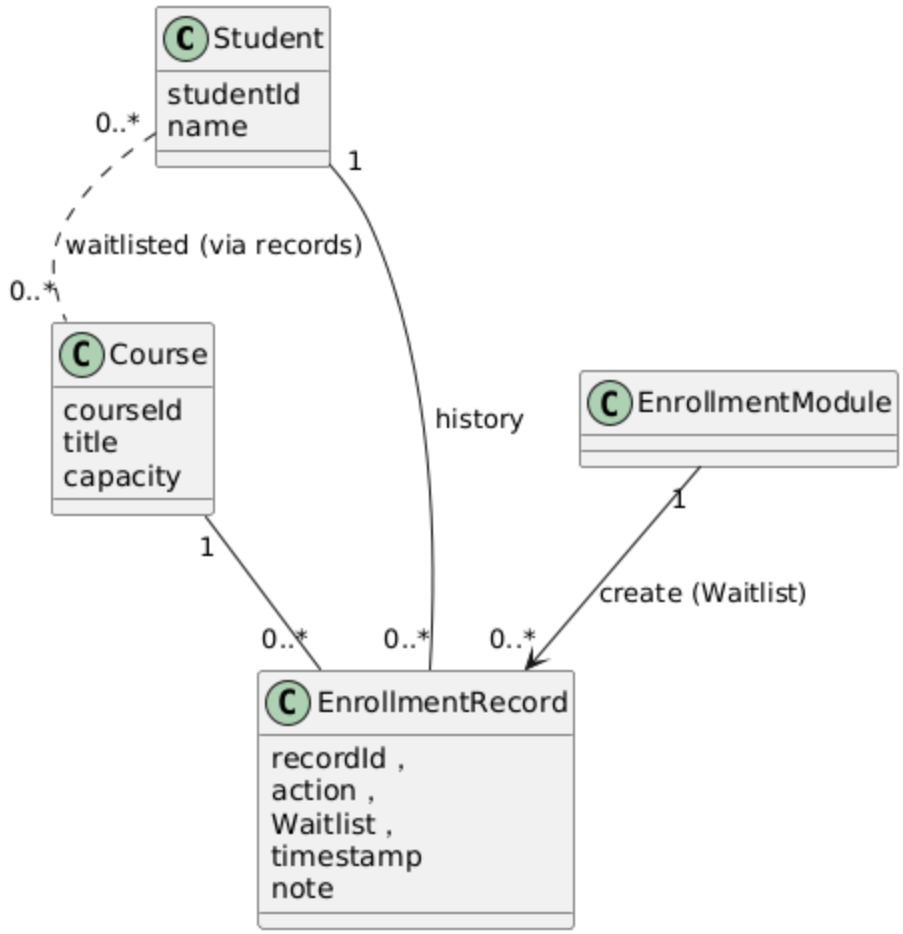
Sequence Diagram :



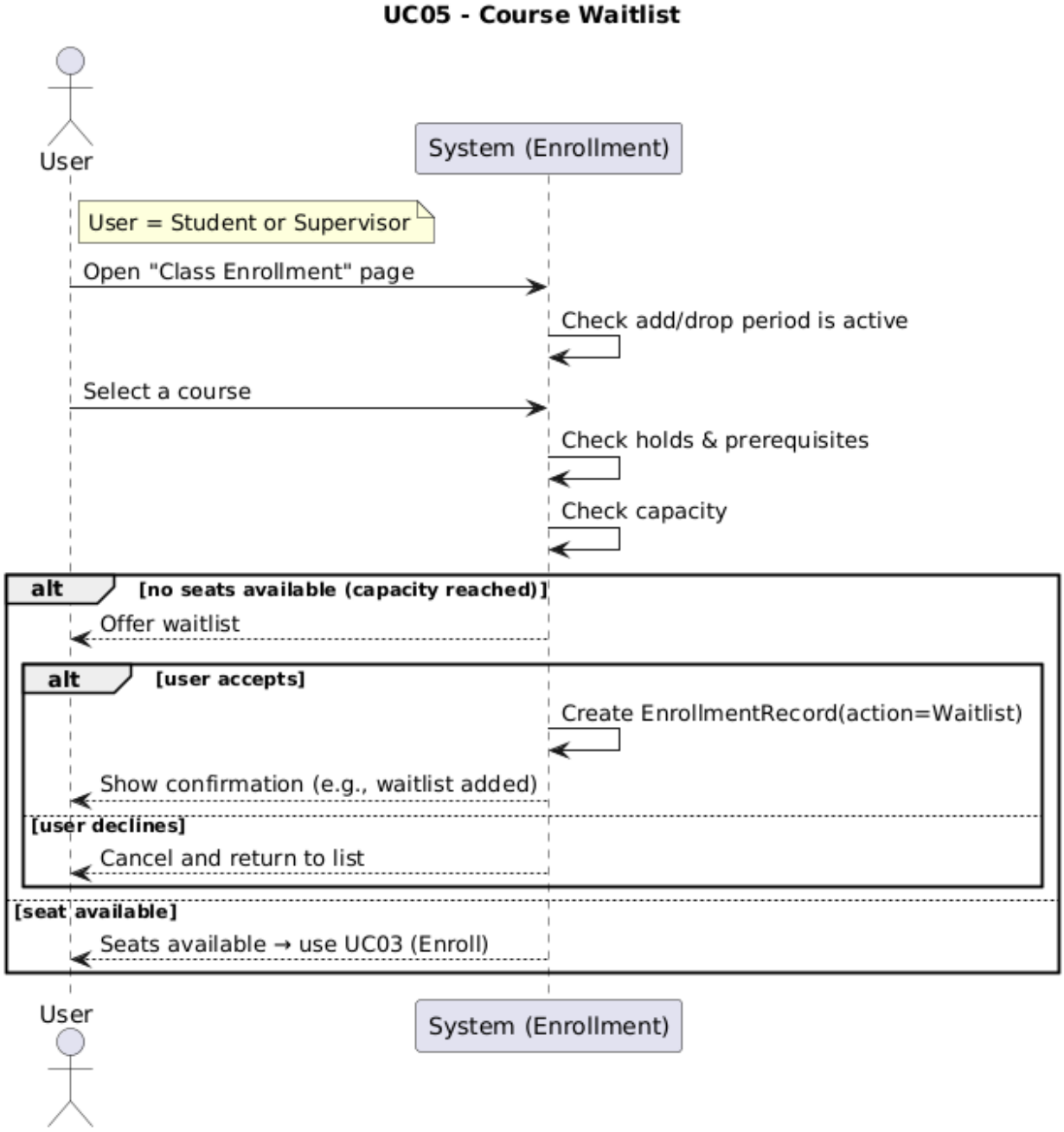
Use Case 5:



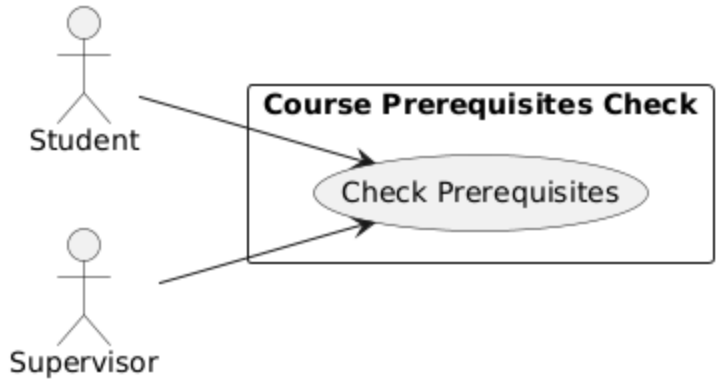
Class diagram :



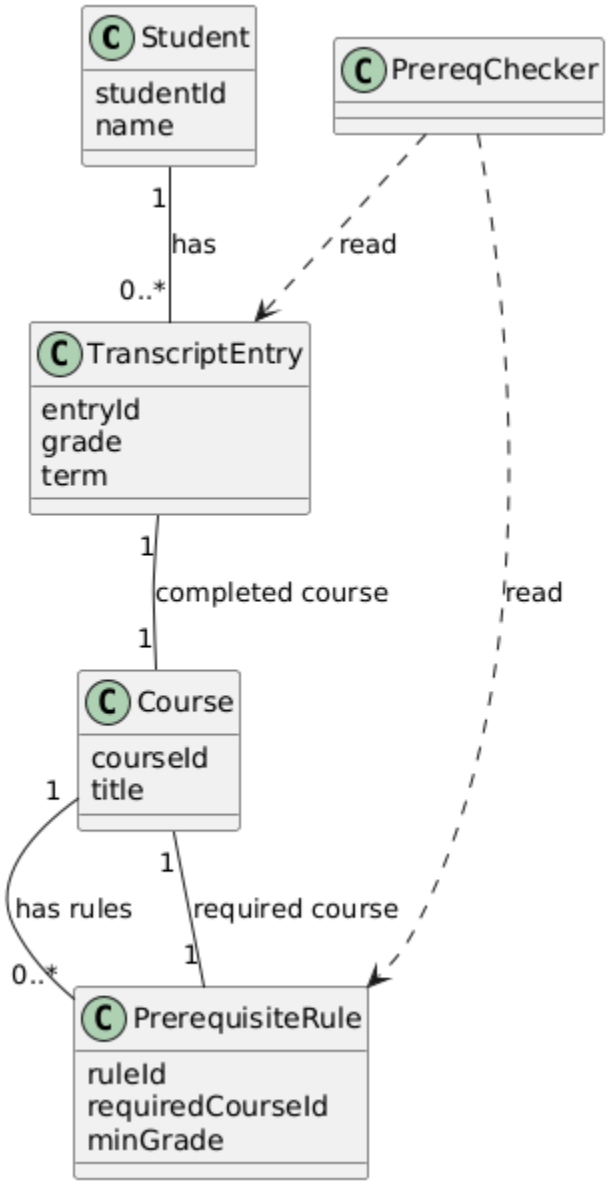
Sequence Diagram :


UC05 - Course Waitlist
User
System (Enrollment)
User = Student or Supervisor
Open "Class Enrollment" page
Check add/drop period is active
Select a course
Check holds & prerequisites
Check capacity
alt
[no seats available (capacity reached)]
Offer waitlist
alt
[user accepts]
Create EnrollmentRecord(action=Waitlist)
Show confirmation (e.g., waitlist added)
[user declines]
Cancel and return to list
[seat available]
Seats available → use UC03 (Enroll)
User
System (Enrollment)

Use Case 6:

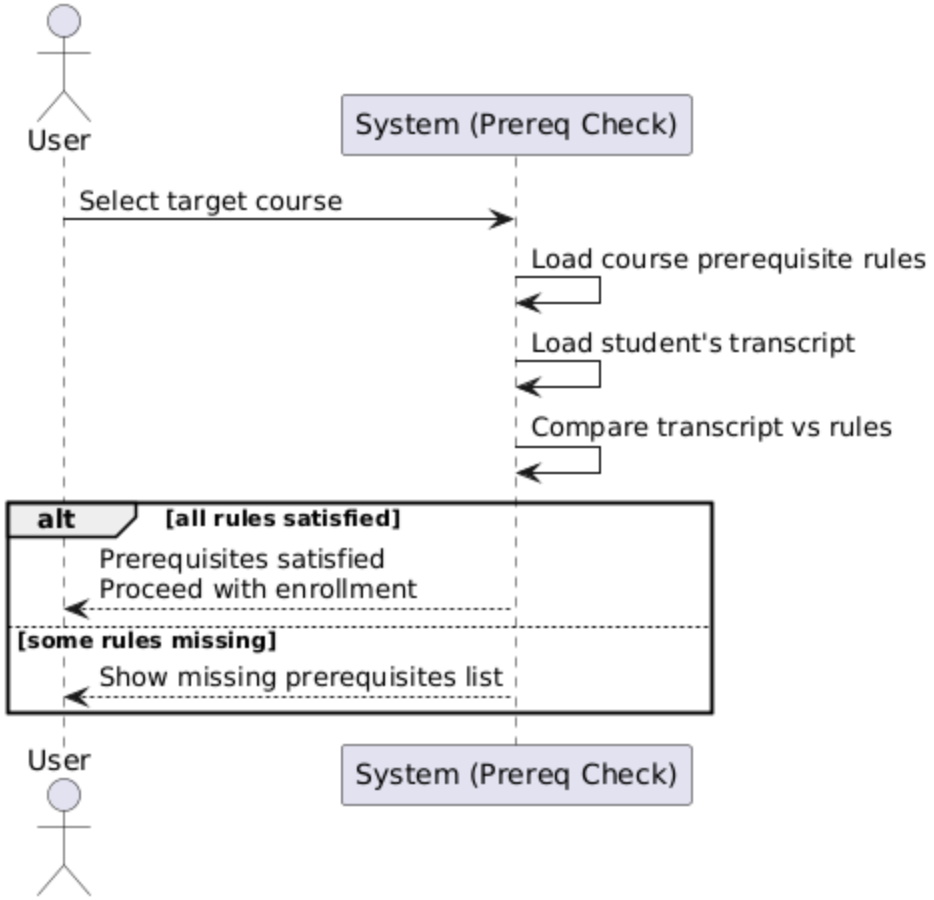


Class diagram :

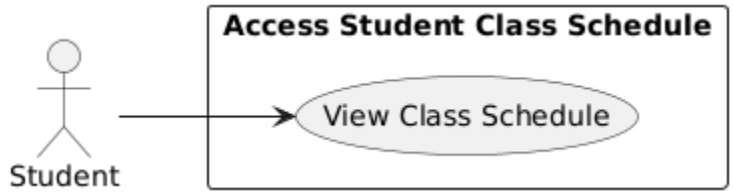


Sequence Diagram :

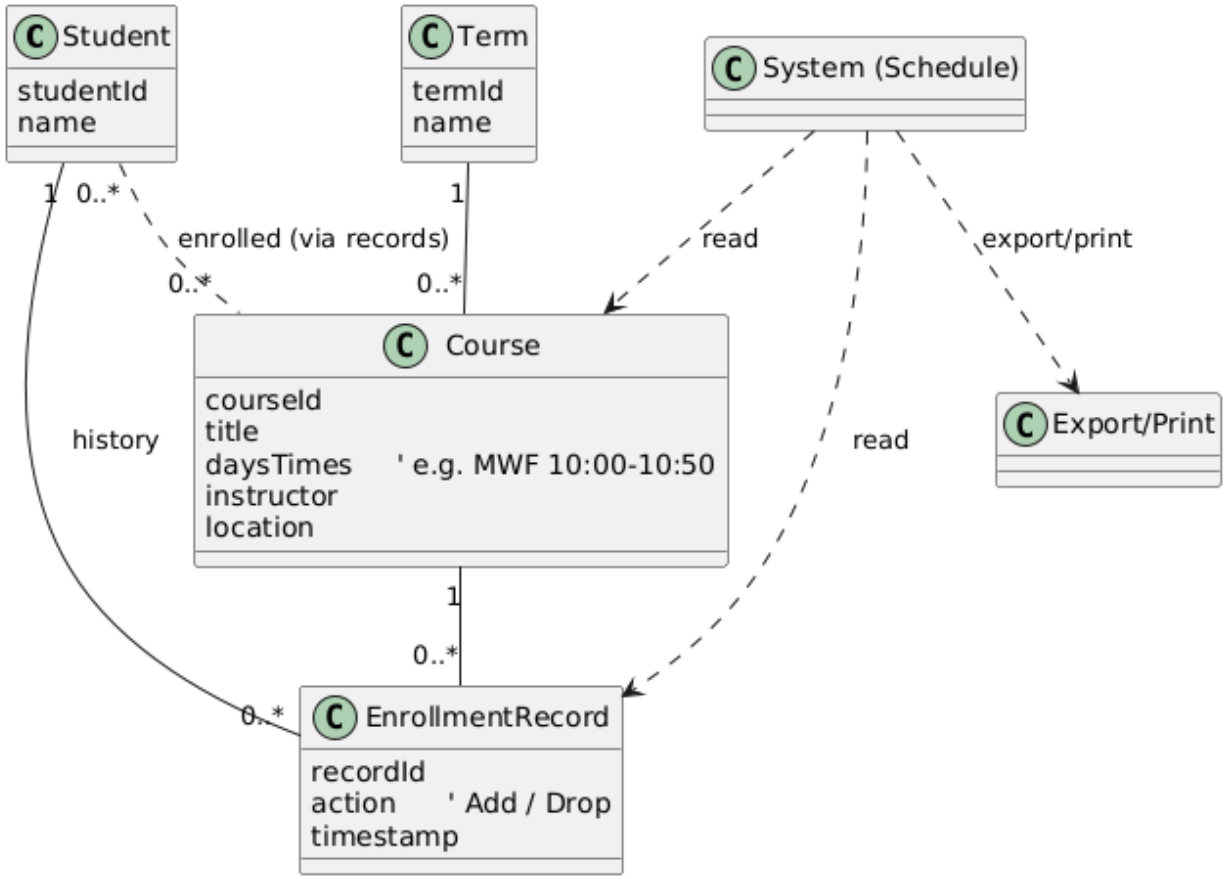
UC06 - Prerequisites Check (Sequence)



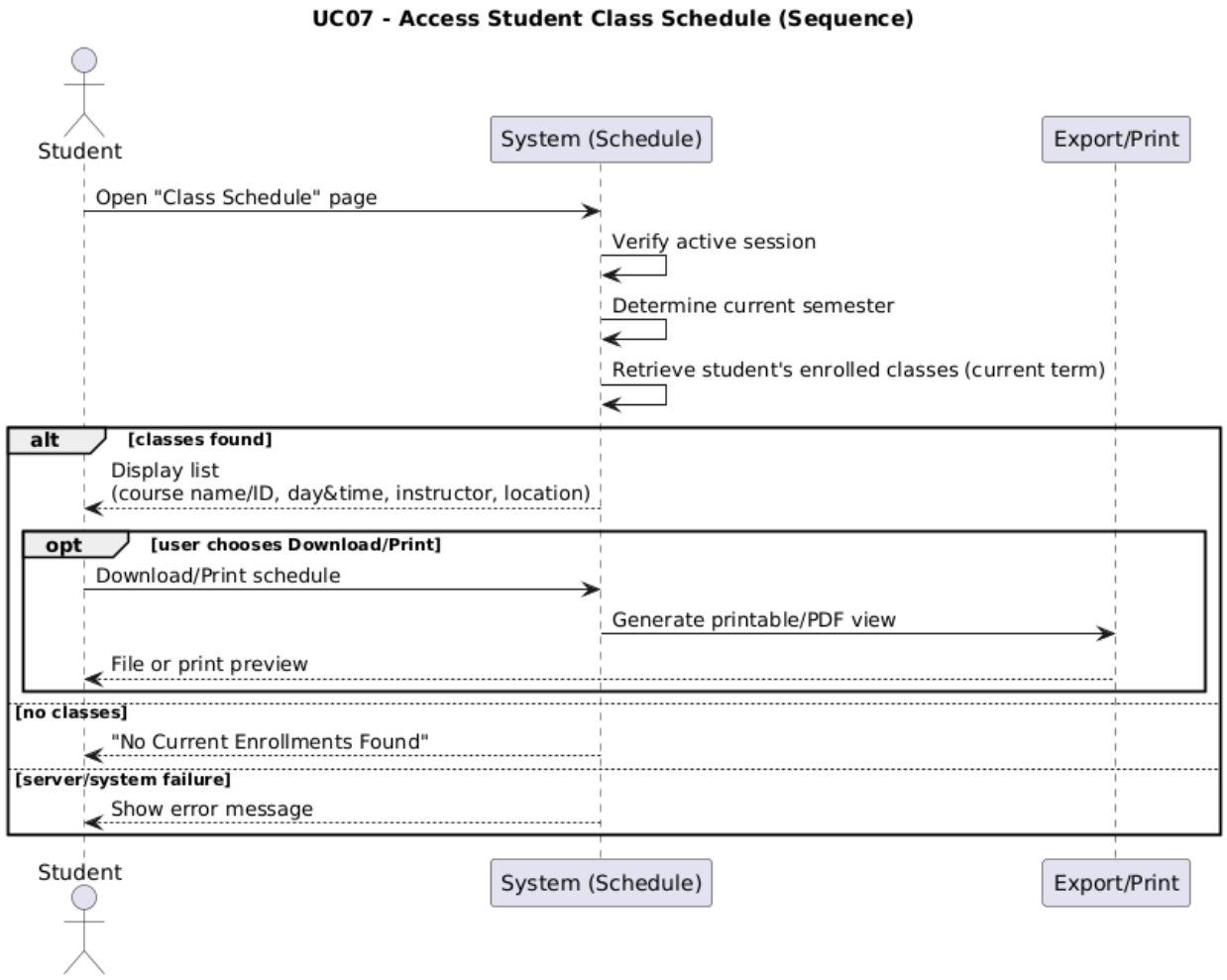
Use Case 7:



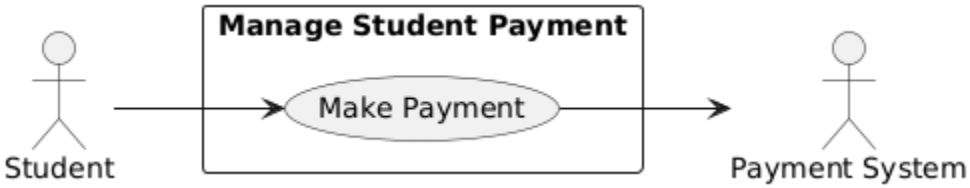
Class Diagram：

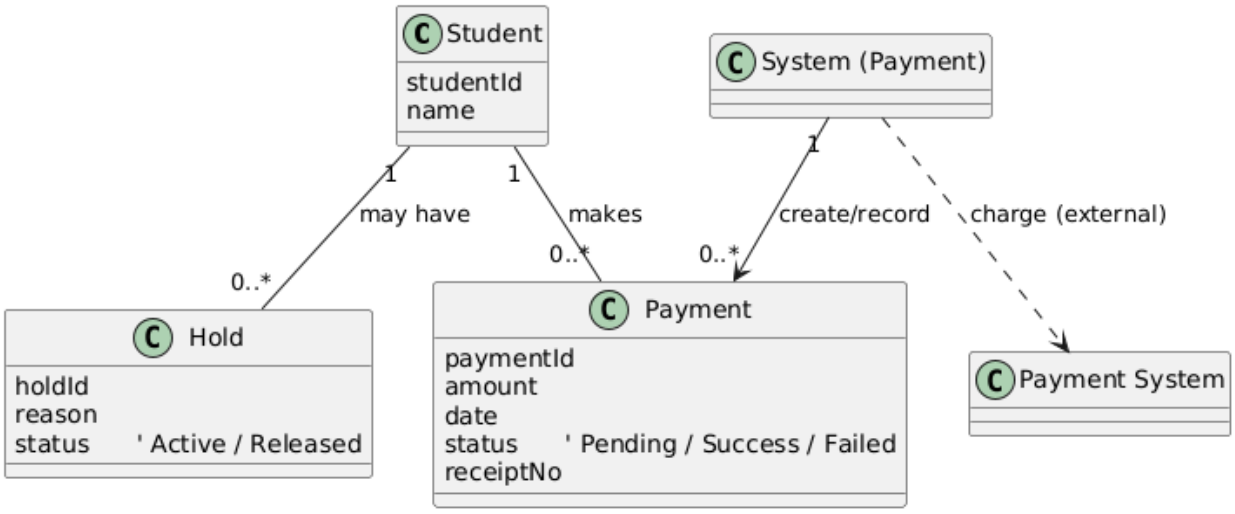


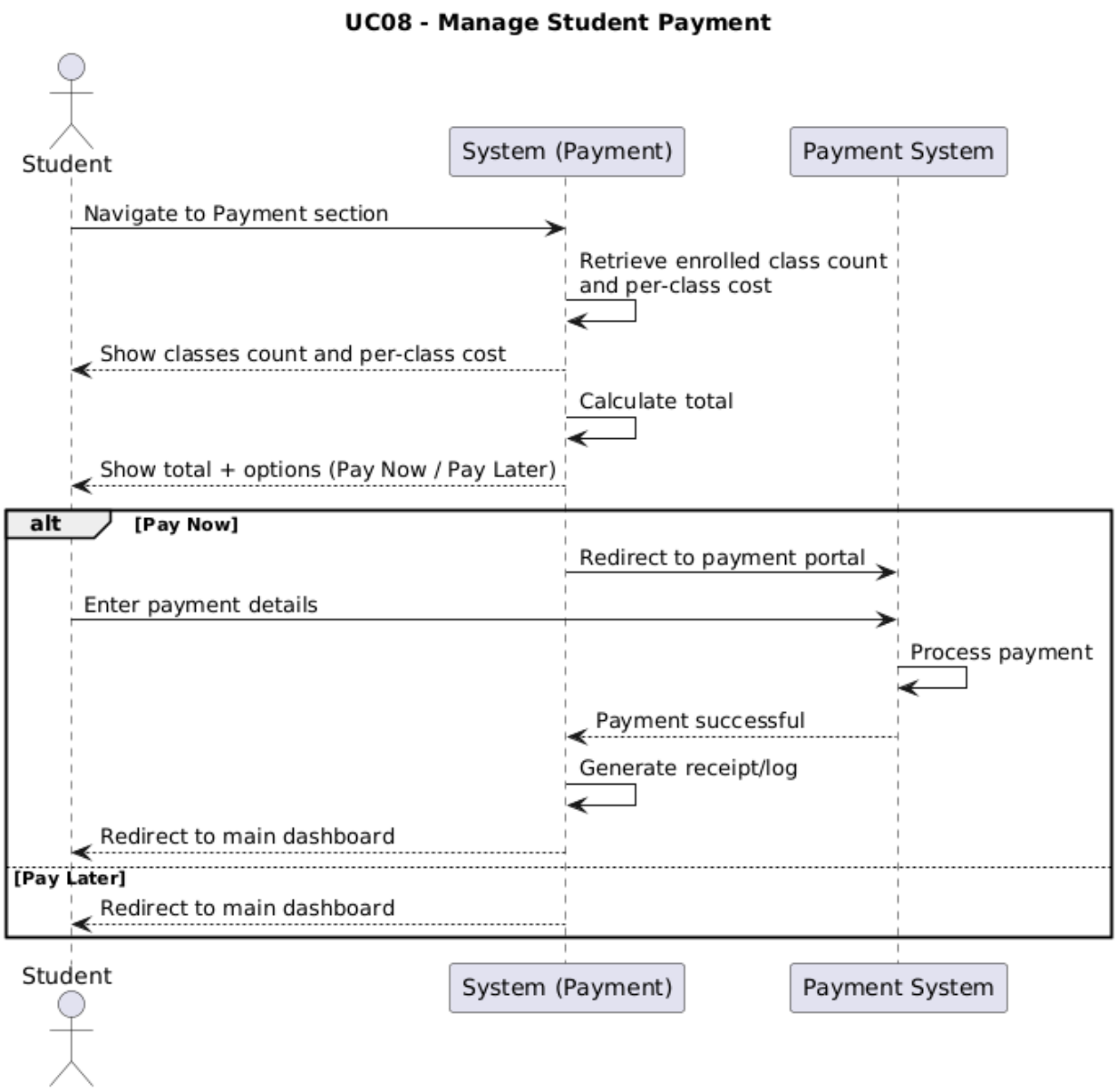
Sequence Diagram：


UC07 - Access Student Class Schedule (Sequence)
Student
System (Schedule)
Export/Print
Open "Class Schedule" page
Verify active session
Determine current semester
Retrieve student's enrolled classes (current term)
alt
[classes found]
Display list
(course name/ID, day&time, instructor, location)
opt
[user chooses Download/Print]
Download/Print schedule
Generate printable/PDF view
File or print preview
[no classes]
"No Current Enrollments Found"
[server/system failure]
Show error message
Student
System (Schedule)
Export/Print

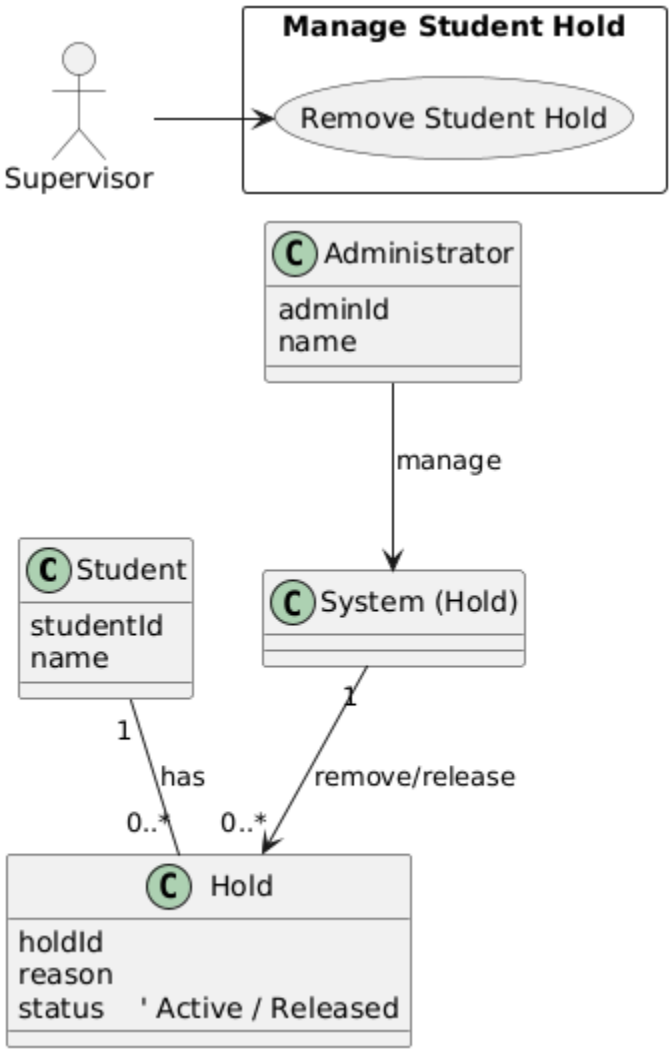
Use Case 8:

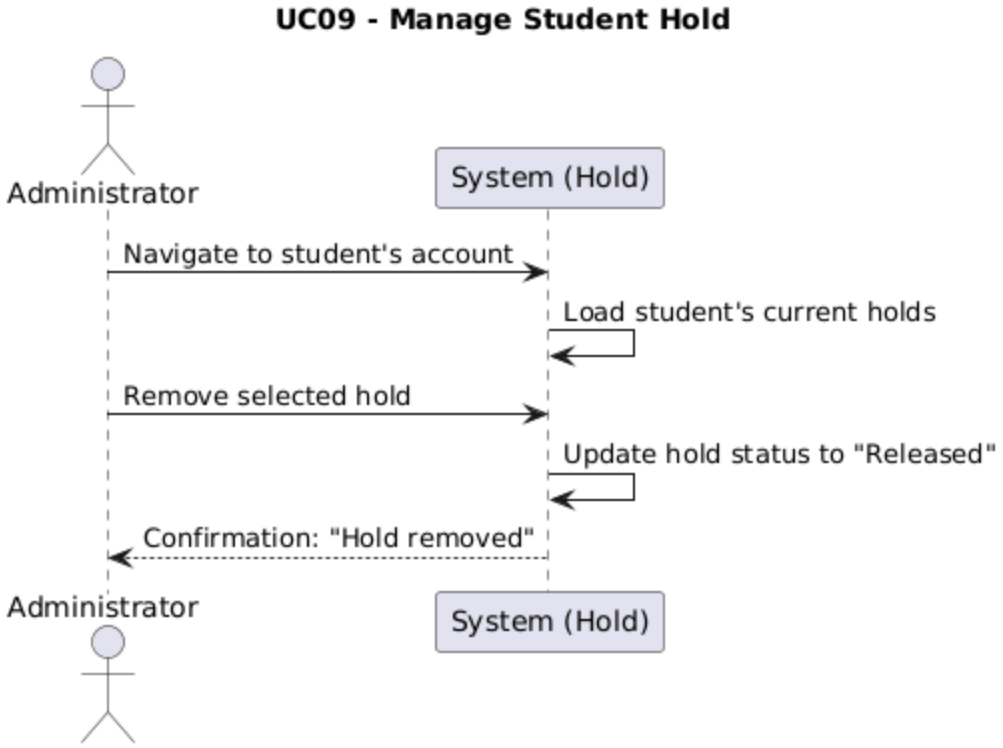

Manage Student Payment
Make Payment
Student
Payment System


Student
studentId
name
System (Payment)
1
may have
0..*
1
makes
0..*
1
create/record
0..*
charge (external)
Hold
holdId
reason
status ' Active / Released
Payment
paymentId
amount
date
status ' Pending / Success / Failed
receiptNo
Payment System

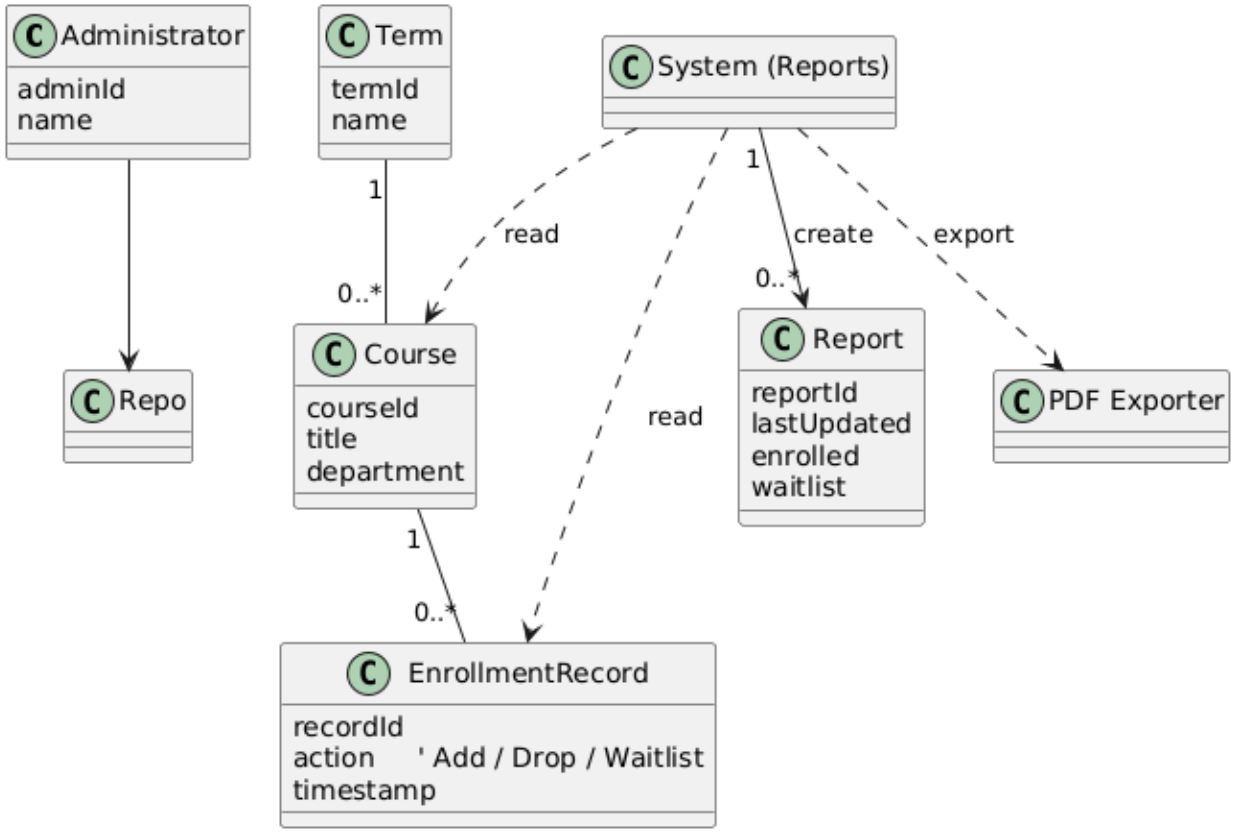

UC08 - Manage Student Payment
Student
System (Payment)
Payment System
Navigate to Payment section
Retrieve enrolled class count and per-class cost
Show classes count and per-class cost
Calculate total
Show total + options (Pay Now / Pay Later)
alt
[Pay Now]
Redirect to payment portal
Enter payment details
Process payment
Payment successful
Generate receipt/log
Redirect to main dashboard
[Pay Later]
Redirect to main dashboard
Student
System (Payment)
Payment System

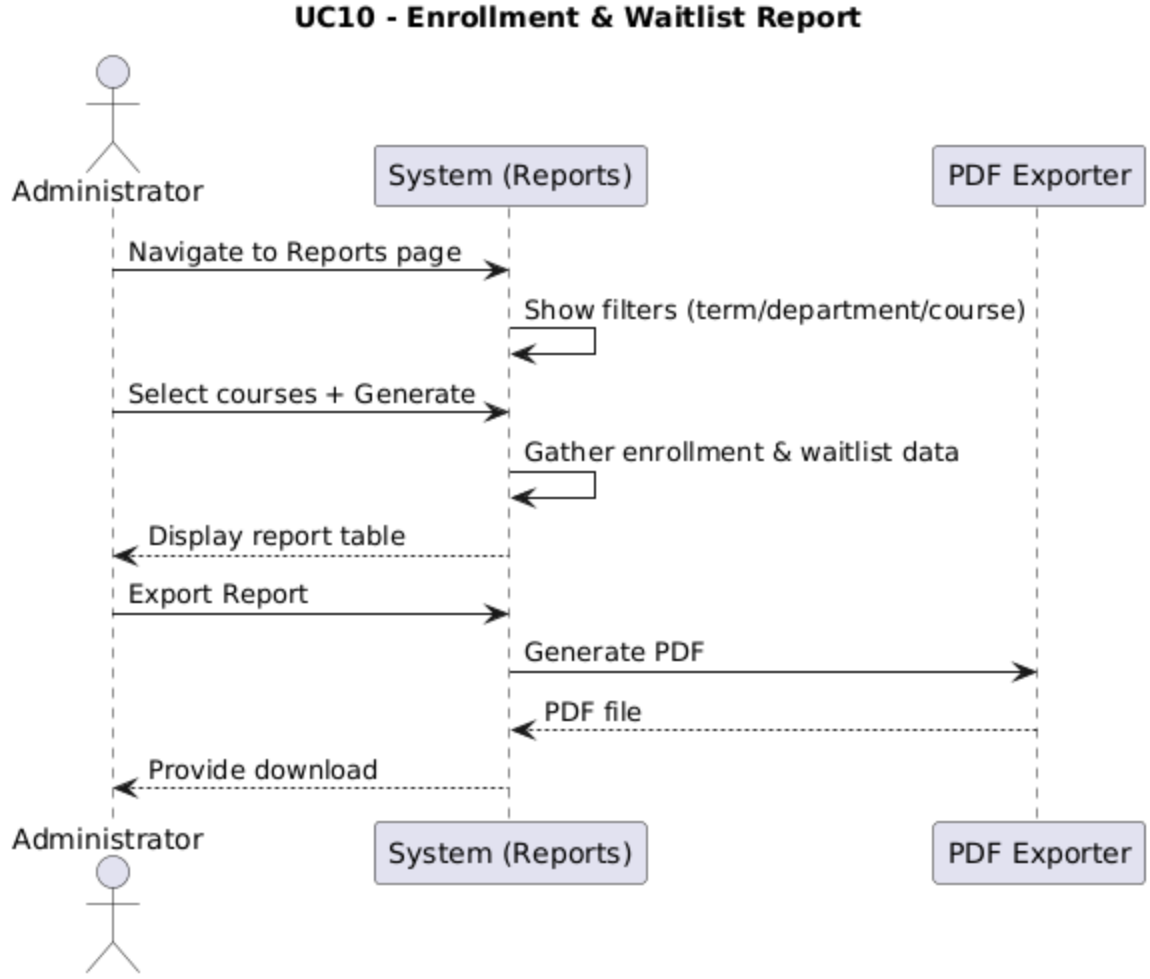
Use Case 9:


Manage Student Hold
Remove Student Hold
Supervisor
Administrator
adminId
name
manage
Student
studentId
name
System (Hold)
1
has
0..*
1
remove/release
0..*
Hold
holdId
reason
status ' Active / Released

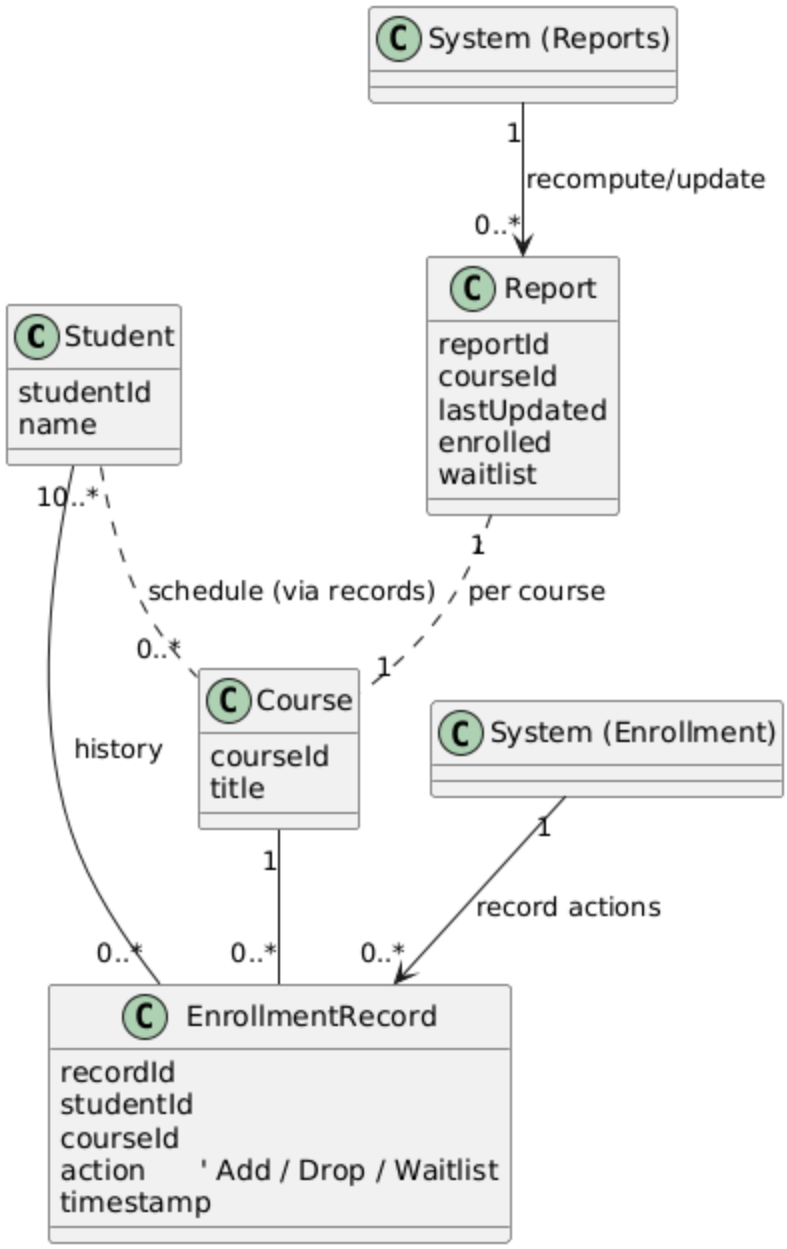

UC09 - Manage Student Hold
Administrator
System (Hold)
Navigate to student's account
Load student's current holds
Remove selected hold
Update hold status to "Released"
Confirmation: "Hold removed"
Administrator
System (Hold)

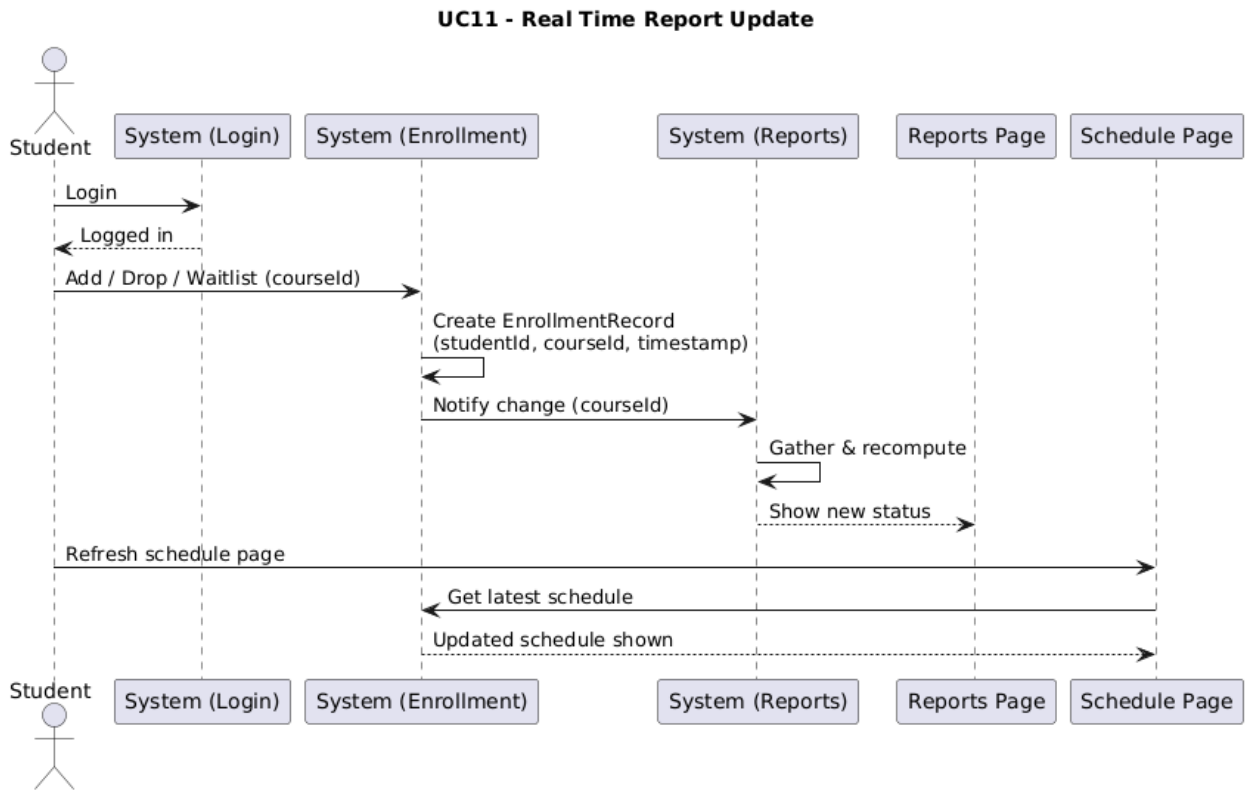
UC10:


Administrator
adminId
name
Term
termId
name
System (Reports)
Repo
Course
courseId
title
department
Report
reportId
lastUpdated
enrolled
waitlist
PDF Exporter
EnrollmentRecord
recordId
action ' Add / Drop / Waitlist
timestamp
1
0..*
read
create
export
read

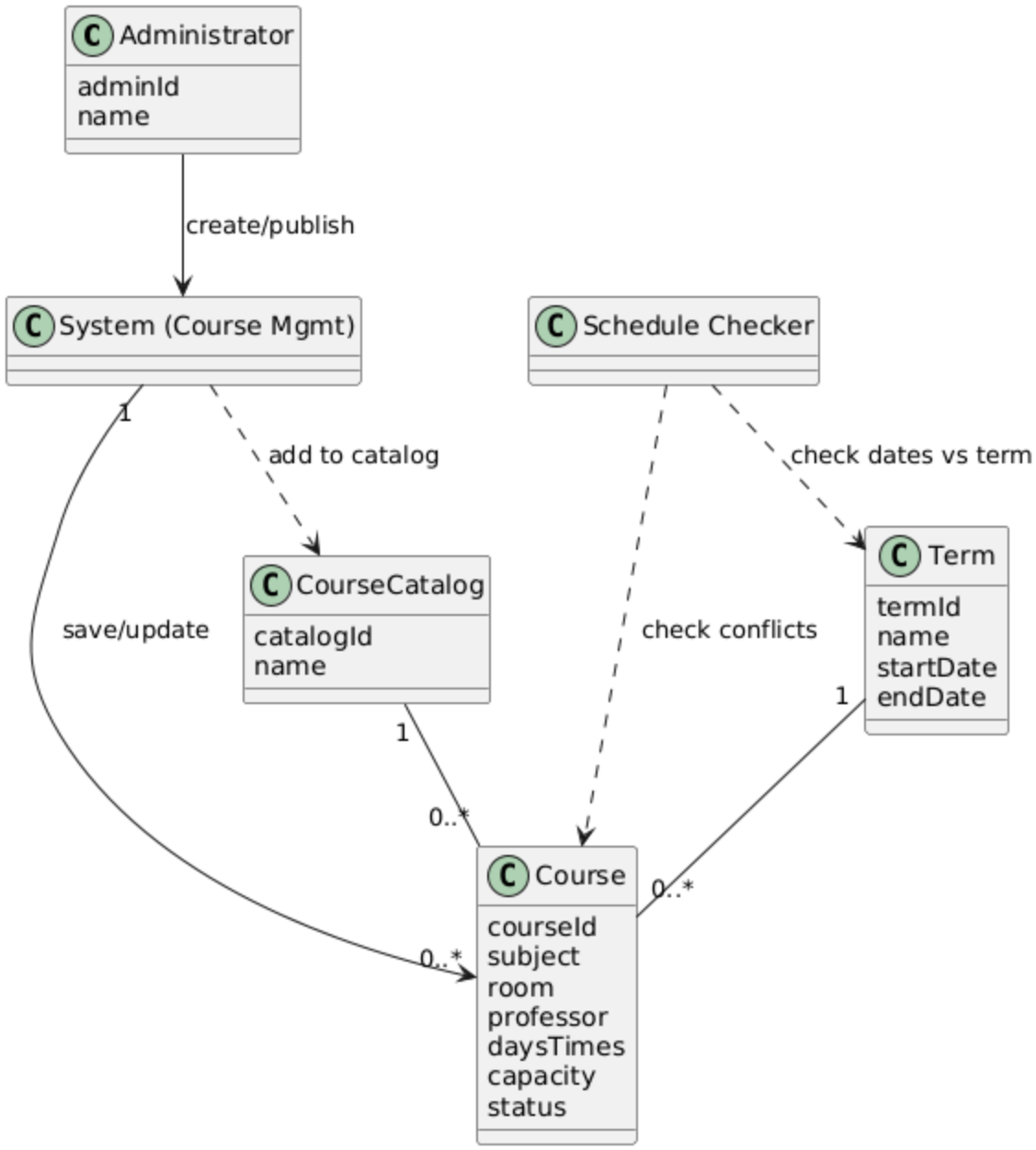

UC10 - Enrollment & Waitlist Report
Administrator
System (Reports)
PDF Exporter
Navigate to Reports page
Show filters (term/department/course)
Select courses + Generate
Gather enrollment & waitlist data
Display report table
Export Report
Generate PDF
PDF file
Provide download
Administrator
System (Reports)
PDF Exporter

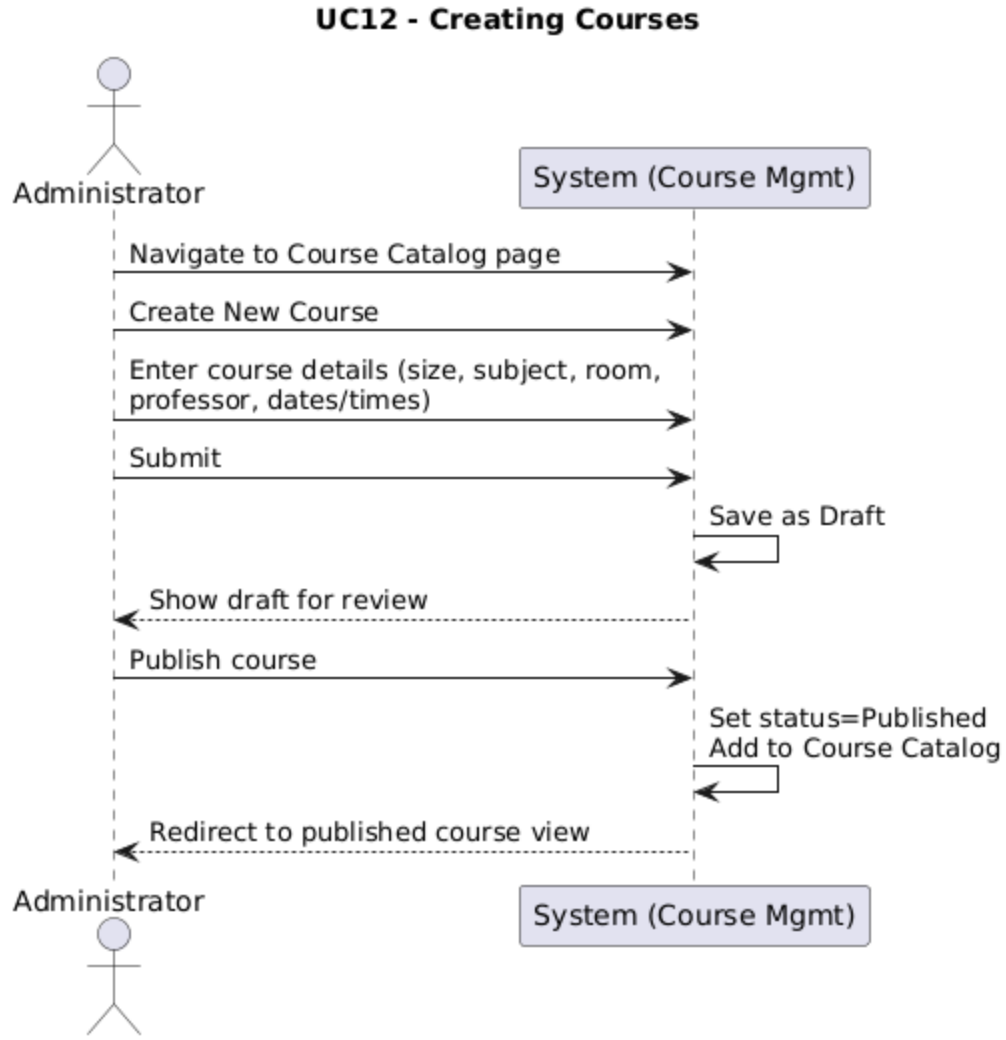
UC11:


System (Reports)
1
recompute/update
0..*
Report
reportId
courseId
lastUpdated
enrolled
waitlist
Student
studentId
name
10..*
1
schedule (via records)
per course
0..*
1
Course
courseId
title
System (Enrollment)
history
1
record actions
0..*
0..*
0..*
EnrollmentRecord
recordId
studentId
courseId
action ' Add / Drop / Waitlist
timestamp

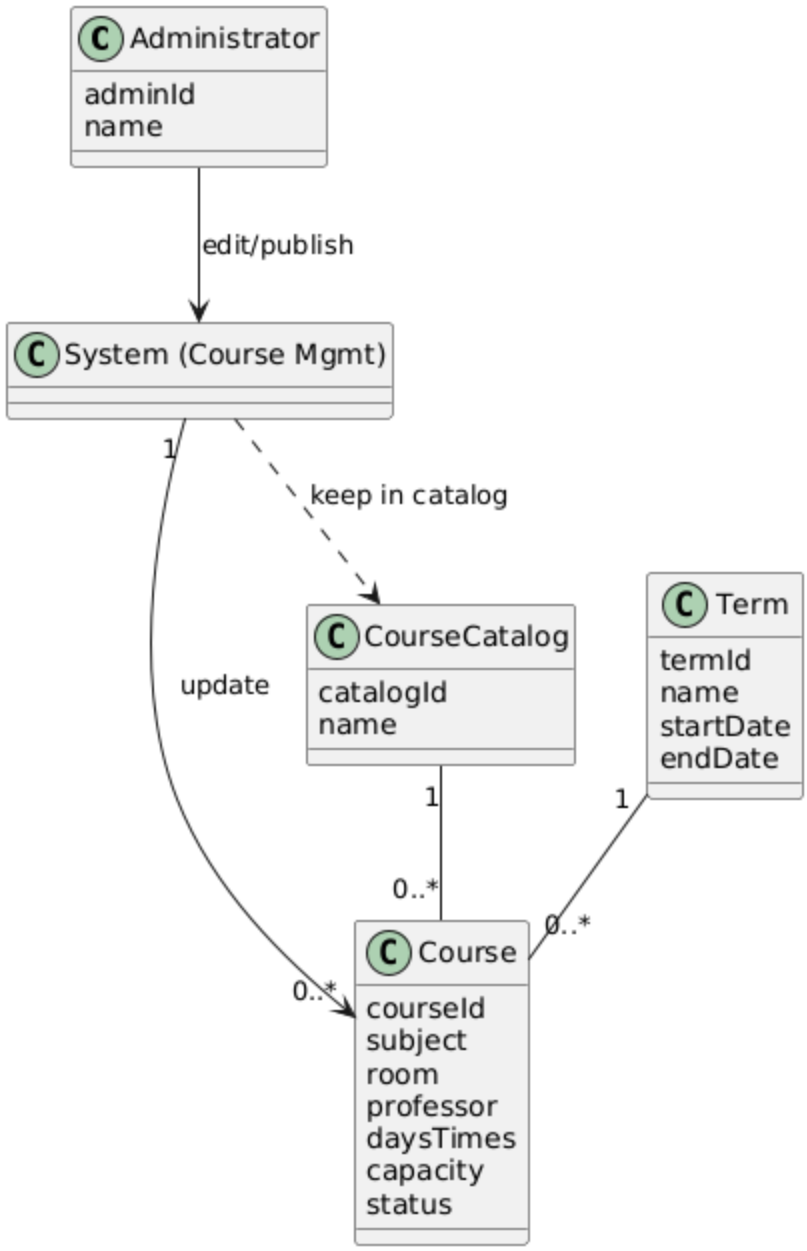

UC11 - Real Time Report Update
Student
System (Login)
System (Enrollment)
System (Reports)
Reports Page
Schedule Page
Login
Logged in
Add / Drop / Waitlist (courseId)
Create EnrollmentRecord
(studentId, courseId, timestamp)
Notify change (courseId)
Gather & recompute
Show new status
Refresh schedule page
Get latest schedule
Updated schedule shown
Student
System (Login)
System (Enrollment)
System (Reports)
Reports Page
Schedule Page

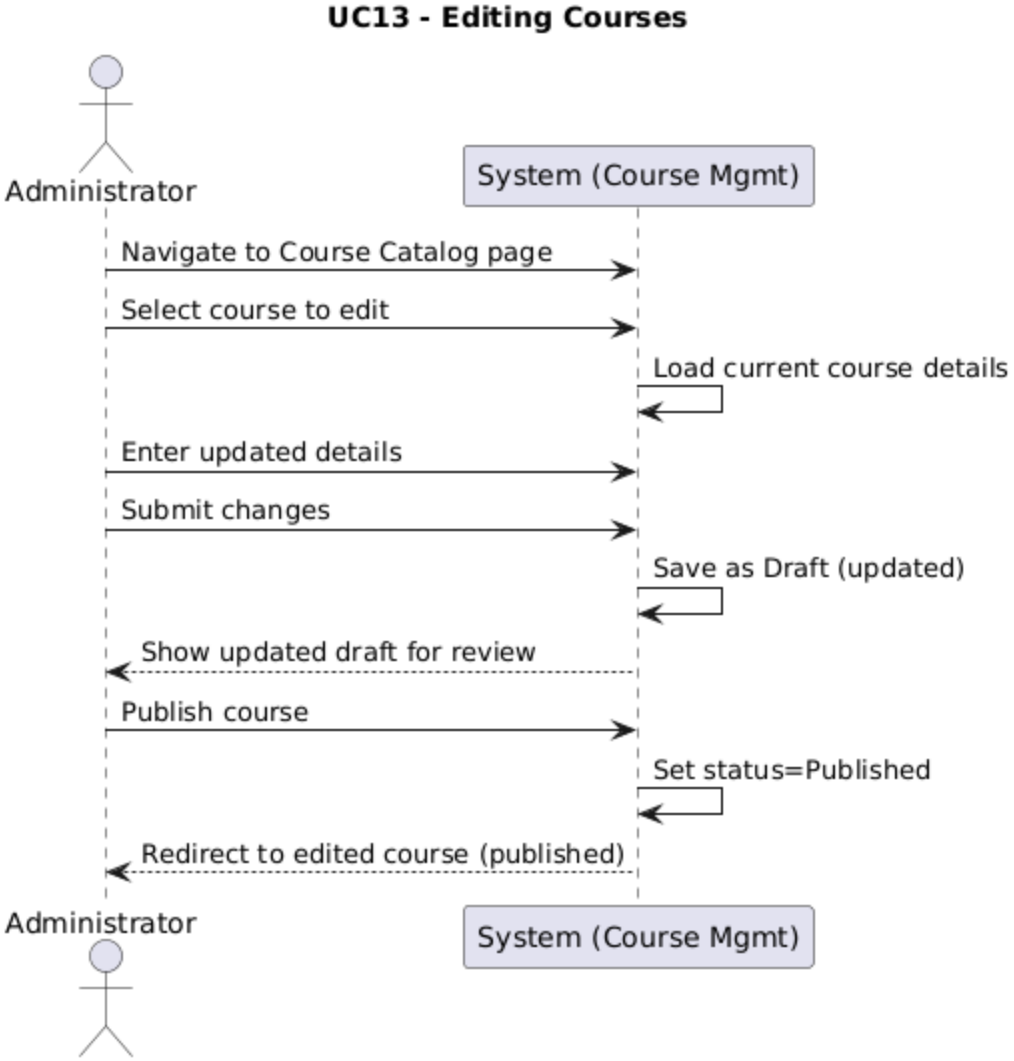
UC12:


Administrator
adminId
name
create/publish
System (Course Mgmt)
Schedule Checker
1
add to catalog
check dates vs term
CourseCatalog
catalogId
name
save/update
check conflicts
Term
termId
name
startDate
endDate
1
1
0..*
0..*
0..*
Course
courseId
subject
room
professor
daysTimes
capacity
status


UC12 - Creating Courses
Administrator
System (Course Mgmt)
Navigate to Course Catalog page
Create New Course
Enter course details (size, subject, room, professor, dates/times)
Submit
Save as Draft
Show draft for review
Publish course
Set status=Published
Add to Course Catalog
Redirect to published course view
Administrator
System (Course Mgmt)

UC13:


Administrator
adminId
name
edit/publish
System (Course Mgmt)
1
keep in catalog
update
CourseCatalog
catalogId
name
Term
termId
name
startDate
endDate
1
1
0..*
0..*
0..*
Course
courseId
subject
room
professor
daysTimes
capacity
status


UC13 - Editing Courses
Administrator
System (Course Mgmt)
Navigate to Course Catalog page
Select course to edit
Load current course details
Enter updated details
Submit changes
Save as Draft (updated)
Show updated draft for review
Publish course
Set status=Published
Redirect to edited course (published)
Administrator
System (Course Mgmt)