

RISE-EDU

Software Design Specification

Phase 2

Revision History

Date	Revision	Description	Author
10/15/2025	1.0	Initial Version	Wail Mohammed
<u>10/15/2025</u>	1.1	Purpose, Scope, Definitions Updated	Wail Mohammed
<u>10/15/2025</u>	1.2	Updated class candidates 01,02,03,04,05	Wail Mohammed
<u>10/17/2025</u>	1.3	Updated class candidates 06,07,08,09,10	Wail, Yesenia, Emmanuel, Shichang
<u>10/23/2025</u>	1.4	Updated Class Diagram and Sequence diagrams	Wail Mohammed
<u>10/25/2025</u>	1.5	Updated all class candidates, added class 11	Wail, Yesenia, Emmanuel
<u>10/26/2025</u>	1.6	Updated the System Architecture	Emmanuel
<u>10/27/2025</u>	1.7	Updated Course Prerequisites Check	Shichang Wang
<u>10/27/2025</u>	1.8	Updated Project Description, Product Architecture.	Wail Mohammed
	1.9		
	2.0		
	2.1		
	2.2		
	2.3		

Table of Contents

1. PURPOSE.....	5
1.1. SCOPE	5
1.2. DEFINITIONS, ACRONYMS, ABBREVIATIONS	5
1.3. REFERENCES.....	5
1.4. OVERVIEW	5
2. OVERALL DESCRIPTION.....	6
2.1. PRODUCT PERSPECTIVE	6
2.2. PRODUCT ARCHITECTURE	6
2.3. PRODUCT FUNCTIONALITY/FEATURES	6
2.4. CONSTRAINTS.....	7
2.4. ASSUMPTIONS AND DEPENDENCIES	7
3. SPECIFIC REQUIREMENTS.....	8
3.1. FUNCTIONAL REQUIREMENTS.....	8
3.1.1. Common Requirements:.....	8
3.1.2. The Student Module Requirements:.....	8
3.1.3. The School Administrator Module Requirements:.....	8
3.1.4. The Reporting Module Requirements:.....	9
3.2. EXTERNAL INTERFACE REQUIREMENTS	9
3.3. INTERNAL INTERFACE REQUIREMENTS.....	9
4. NON-FUNCTIONAL REQUIREMENTS.....	10
4.1. SECURITY AND PRIVACY REQUIREMENTS	10
4.2. ENVIRONMENTAL REQUIREMENTS	10
4.3. PERFORMANCE REQUIREMENTS	10
5. USE CASE SPECIFICATION	11
USE CASE 1: MANAGE USER LOGIN	11
USE CASE 2: COURSE ENROLLMENT.....	12
USE CASE 3: COURSE DROP.....	13
USE CASE 4: COURSE WAITLIST	14
USE CASE 5: COURSE PREREQUISITES CHECK.....	15
USE CASE 6: ACCESS STUDENT CLASS SCHEDULE	16
USE CASE 7: MANAGE STUDENT HOLD.....	17
USE CASE 8: ENROLLMENT REPORTING	18
USE CASE 9: UPDATE CHANGES REPORT	19
USE CASE 10: CREATE COURSES.....	20
USE CASE 11: EDITING COURSES	21
6. UML USE CASE DIAGRAMS.....	22
USE CASE 1: MANAGE USER LOGIN	7
USE CASE 2: COURSE ENROLLMENT.....	8
USE CASE 3: COURSE DROP.....	9
USE CASE 4: COURSE WAITLIST	10
USE CASE 5: COURSE PREREQUISITES CHECK.....	10
USE CASE 6: ACCESS STUDENT CLASS SCHEDULE	12
USE CASE 7: MANAGE STUDENT HOLD.....	13
USE CASE 8: ENROLLMENT REPORTING	14
USE CASE 9: UPDATE CHANGES REPORT	15
USE CASE 10: CREATE COURSES.....	17
USE CASE 11: EDITING COURSES	28
7. CLASS DIAGRAMS	18
8. SEQUENCE DIAGRAMS	30
USE CASE 1: MANAGE USER LOGIN	30

USE CASE 2: COURSE ENROLLMENT.....	31
USE CASE 3: COURSE DROP.....	32
USE CASE 4: COURSE WAITLIST.....	33
USE CASE 5: COURSE PREREQUISITES CHECK.....	34
USE CASE 6: ACCESS STUDENT CLASS SCHEDULE.....	35
USE CASE 7: MANAGE STUDENT HOLD.....	36
USE CASE 8: ENROLLMENT REPORTING.....	37
USE CASE 9: UPDATE CHANGES REPORT.....	38
USE CASE 10: CREATE COURSES.....	39
USE CASE 11: EDITING COURSES.....	40

1. Purpose

This document outlines the system design for the College Course Enrollment System Project. This expands on the software requirements specifications and outlines the system architecture, components, use case classes designs, communication models and the overall implementation of the system

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. Design 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

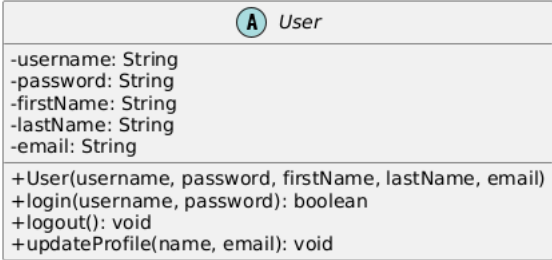
- 2.2.1 The system uses a multithreaded client-server design pattern in which one server application can handle many client applications simultaneously.
- 2.2.2 The Server Application is responsible for listening to and interacting with clients while handling the business logic and data validation.
- 2.2.3 The Client Application is responsible for the Java-based GUI. This will allow system users (students and school administrators) to interact with the system. This client application will provide two different GUI views with different functionalities depending on whether the logged-in user is a Student or an Administrator.
- 2.2.4 All communication between the client and server occurs over TCP/IP.
- 2.2.5 The GUI system is designed using Java Swing and all data will be stored using a locally created file.

2.3 Product Features

- 2.3.1 The system is divided into two primary components: the Client Application and the Server Application. The core business logic happens in the server, in which it processes all client requests. The main modules of the server business logic include:
 - 2.3.1.1 System Manager: To handle all user authorization and authentication using a predefined users data file, one for student users and another for admin users. The system manager also manages universities, courses, registration periods, and withdrawal periods. It processes the core enrollment, drop, and withdrawal requests.
 - 2.3.1.2 Course Manager: To handle creating, deleting, editing, and updating of all course information that include class size, waitlist size, and prerequisites.
 - 2.3.1.3 Enrollment Manager: To handle student course enrollment, course drop, course withdrawal, and checking for waitlist size. This will check for enrollment course limits and check prerequisites.
 - 2.3.1.4 Report Manager: To generate reports for system users. For example, administrators can use it to generate course enrollment reports while student users will use it to generate class schedules.
 - 2.3.1.5 Data Manager: To handle all system data. It will use a locally created database to load and save data on users, courses, and any enrollment actions such as enroll, drop, waitlist, and withdraw.
- 2.3.2 The system's client and server will use a request/response model. In which the client will initiate a request to the server, the server will acknowledge the request, processes it, then sends the response back to the server. The information being sent will be encapsulated in a serializable Message class.


3 Class Design

Class 1: User




Note: may be updateprofile???

Class 2: Student

 Student
-studentID: String -major: String -schoolYear: String -hasPriorityRegistration: boolean
+Student(..., studentID, major, schoolYear) +viewCourseCatalog(): List<Course> +enrollInCourse(course): boolean +dropCourse(course): boolean +withdrawFromCourse(course): boolean +joinWaitlist(course): boolean +viewSchedule(): Schedule +checkPrerequisites(course): boolean +checkAvailability(course): int +viewHolds(): List<Hold>


Note: May remove hasPriority registration

Class 3: Administrator

 Administrator
-adminID: String
+Administrator(..., adminID) +createCourse(...): boolean +editCourse(course, details): boolean +deleteCourse(course): boolean +addStudentToCourse(student, course): boolean +dropStudentFromCourse(student, course): boolean +placeHoldOnAccount(student, holdReason): boolean +removeHoldFromAccount(student, hold): boolean +viewEnrollment(course): List<Student> +viewWaitlist(course): List<Student> +generateEnrollmentReport(course): Report +generateStudentReport(student): Report


Note: may include username/password/name and email as attributes.

Class 4: System Manager

 SystemManager
-addDropPeriod: DateRange -withdrawalPeriod: DateRange
+SystemManager() +addUniversity(universityName): University +getUniversity(universityID): University +authenticateUser(username, password, universityID): User +findCourse(courseID, universityID): Course +findStudent(studentID, universityID): Student +processEnrollment(student, course): boolean +processDrop(student, course): boolean +processWithdrawal(student, course): boolean

Note: please call date attribute as Date type and then look at including university id in the class methods.

Class 5: Course

 Course
<div><div>-courseID: String</div><div>-courseName: String</div><div>-instructorName: String</div><div>-time: String</div><div>-days: String</div><div>-location: String</div><div>-units: int</div><div>-classSize: int</div><div>-availableSeats: int</div></div>
<div><div>+Course(courseID, courseName, ..., units, classSize)</div><div>+addStudent(student): boolean</div><div>+removeStudent(student): boolean</div><div>+addToWaitlist(student): boolean</div><div>+isFull(): boolean</div><div>+getAvailableSeats(): int</div></div>


Note:

Also, look at creating courseList (list of courses)


Add CourseList or ListOfCourses or CourseCatalog

Class 6: Schedule (List of courses)


Note: Lets look at this later. (may be fine)

 Schedule
-semester: String
+Schedule(student, semester) +addCourse(course): boolean +removeCourse(course): boolean +displaySchedule(): String


Class 7: Hold

 Hold
-holdID: String -reason: String -datePlaced: Date
+Hold(reason, admin) +getHoldDetails(): String


Class 8: Waitlist

 Waitlist
-maxWaitlistSize: int
+Waitlist(course, maxSize) +addStudent(student): boolean +promoteNextStudent(): Student +getPosition(student): int +getSize(): int


Class 9: Report

 Report
-reportID: String -reportType: String -generatedDate: Date -reportData: String
+Report(reportType) +generate(data): void +displayReport(): void +saveReport(): boolean

Class 10: Data Manager

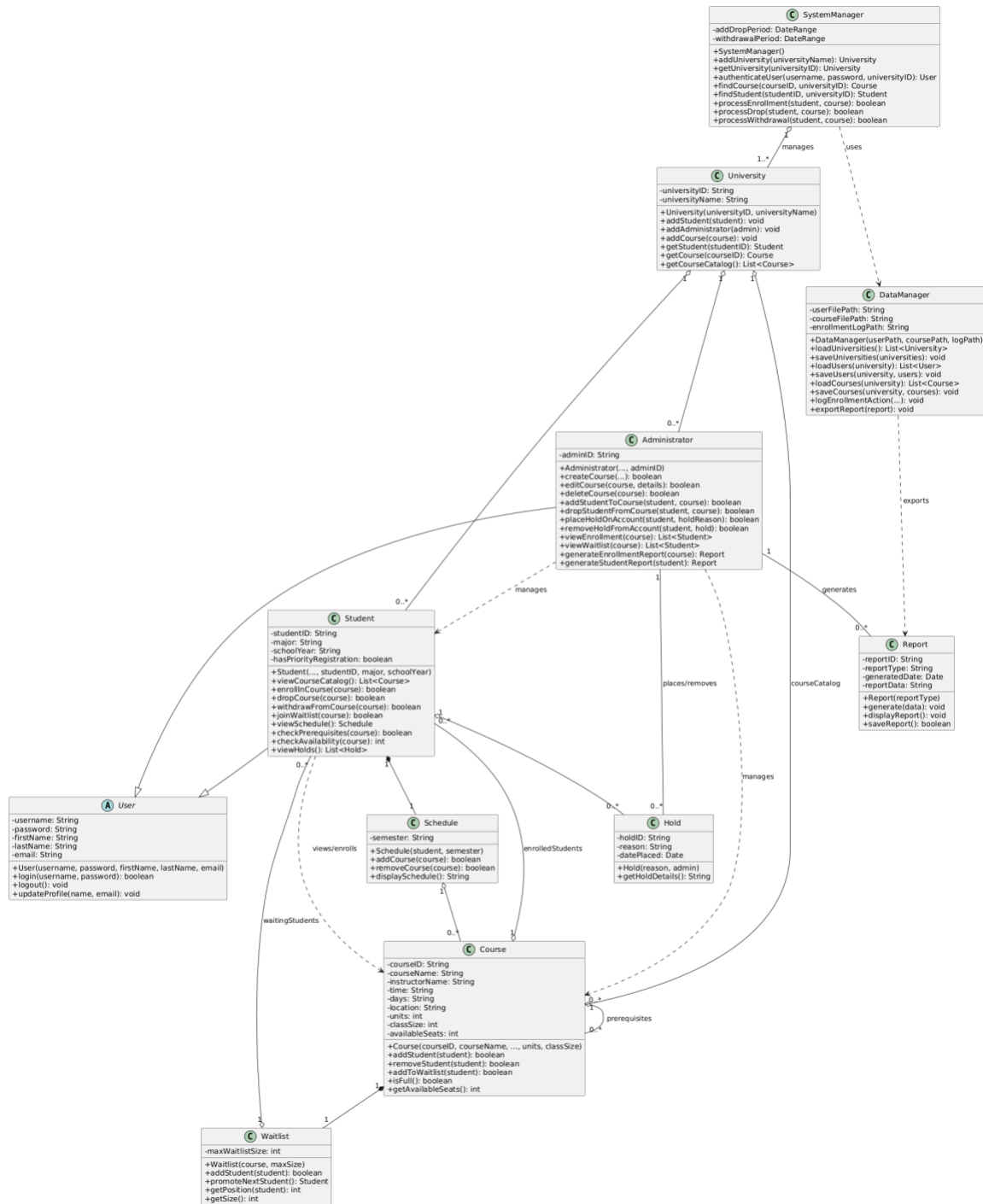
 DataManager
-userFilePath: String -courseFilePath: String -enrollmentLogPath: String
+DataManager(userPath, coursePath, logPath) +loadUniversities(): List<University> +saveUniversities(universities): void +loadUsers(university): List<User> +saveUsers(university, users): void +loadCourses(university): List<Course> +saveCourses(university, courses): void +logEnrollmentAction(...): void +exportReport(report): void

Class: 11: University

 University
-universityID: String -universityName: String
+University(universityID, universityName) +addStudent(student): void +addAdministrator(admin): void +addCourse(course): void +getStudent(studentID): Student +getCourse(courseID): Course +getCourseCatalog(): List<Course>

4 Class Diagram

RISE-EDU System Class Diagram (Multi-University)

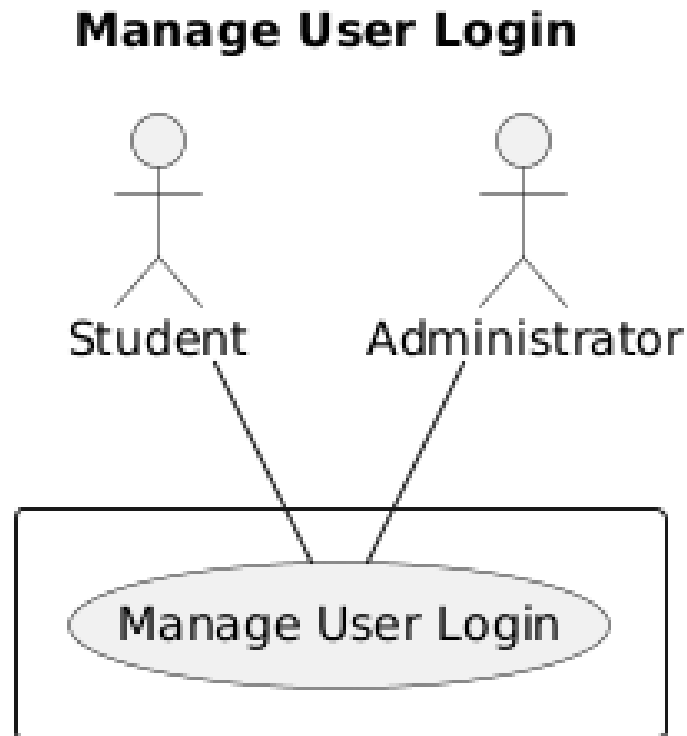


6. Use Cases

6.1 UC01 : Manage User Login

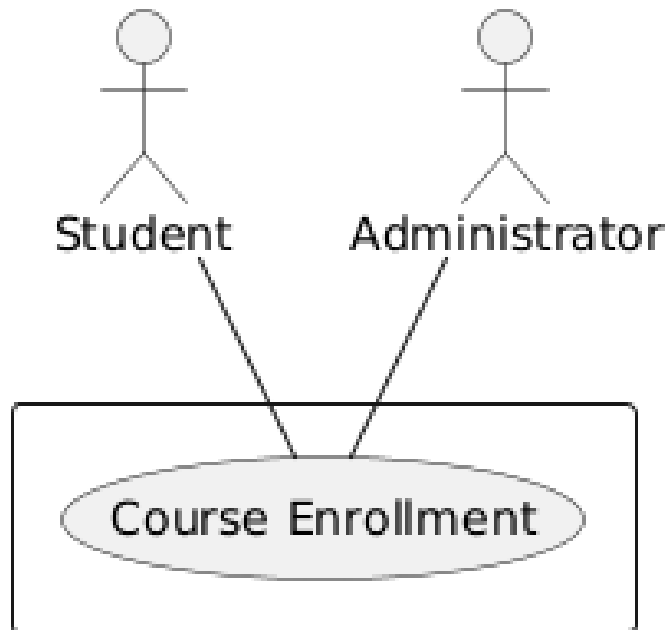
Actor : Student, Admin

Description:

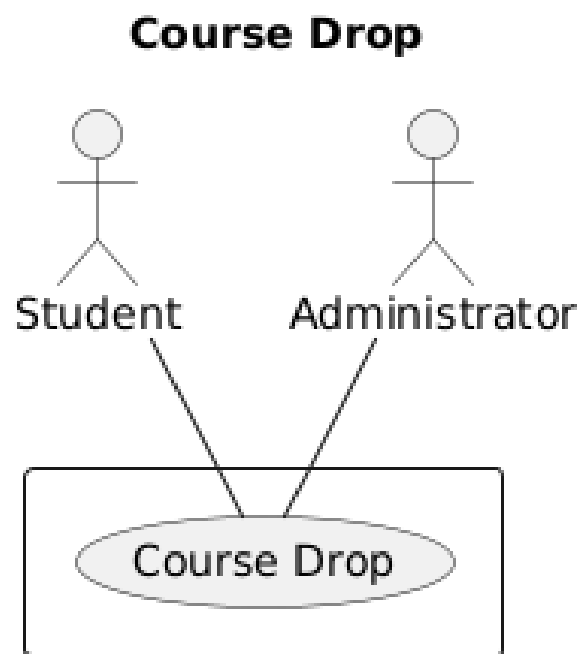


6.2 UC02: Course Enrollment

Course Enrollment

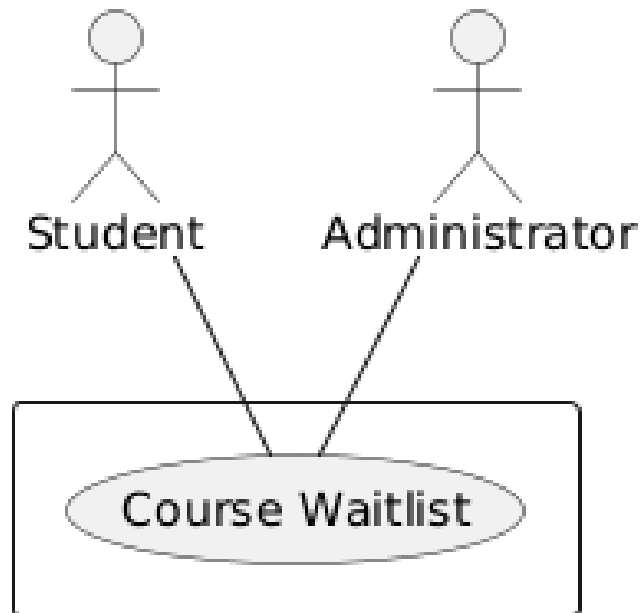


6.3 UC03: Course Drop

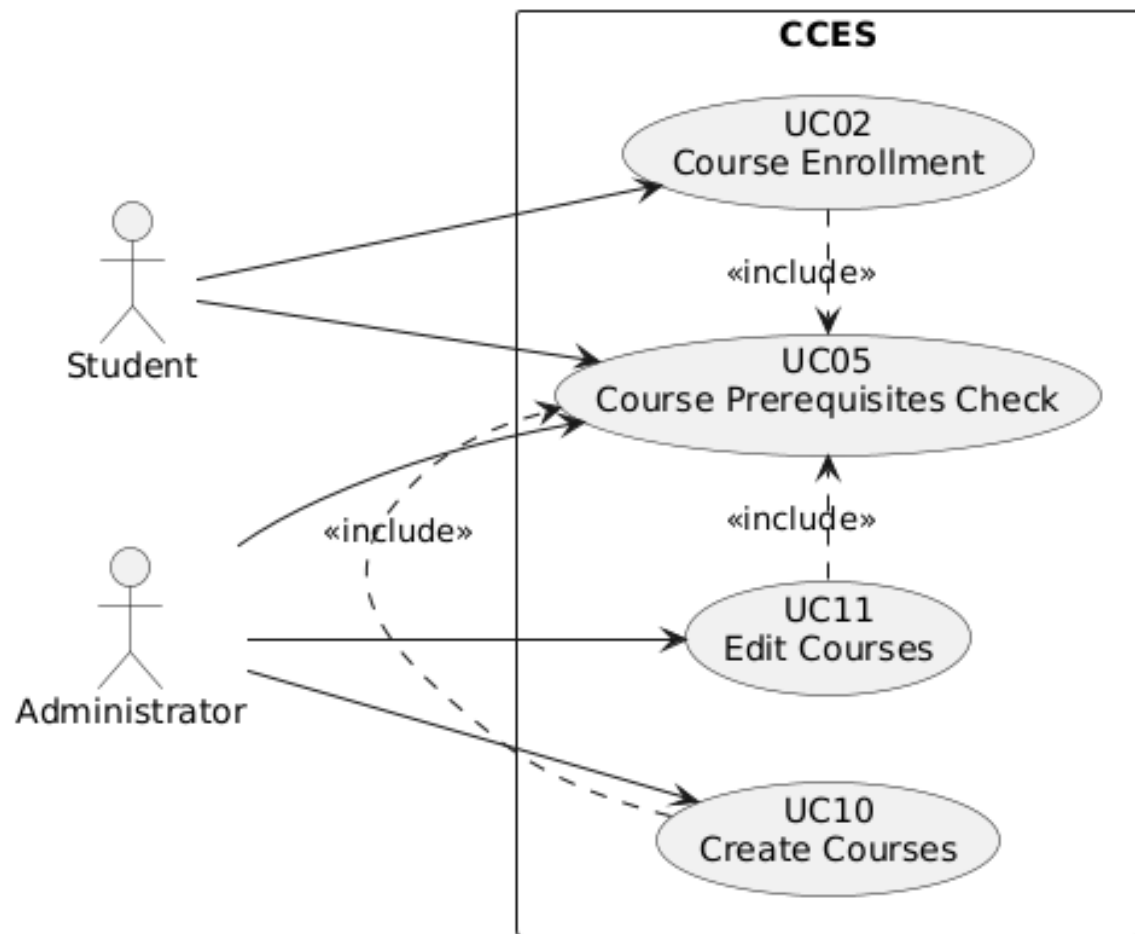


6.4 UC04: Course Waitlist

Course Waitlist

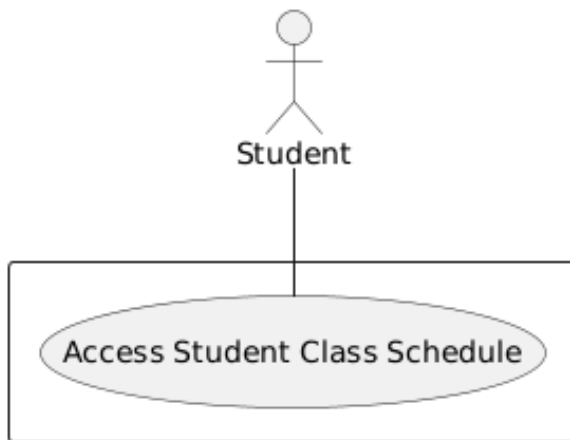


6.5 UC05: Course Prerequisites Check

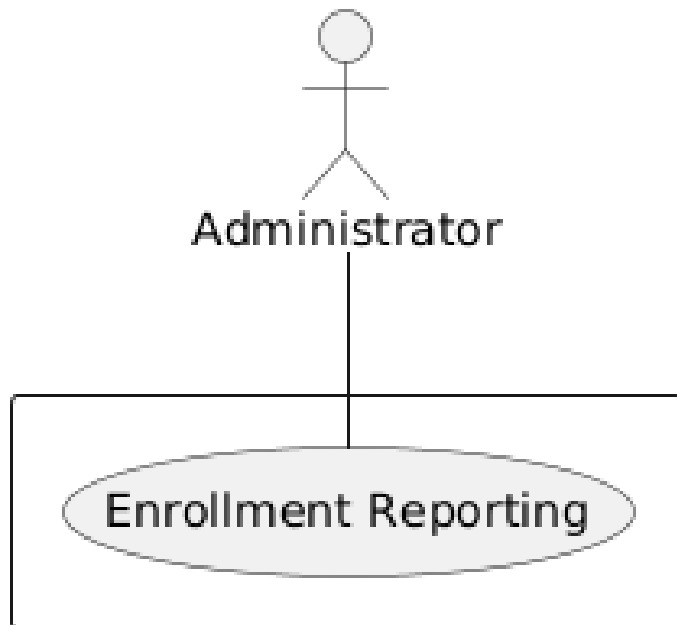


6.6 UC06: Access Schedule

Access Student Class Schedule

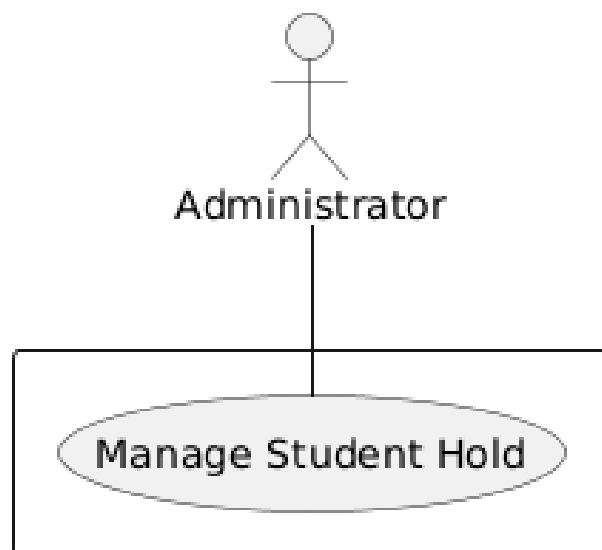


Enrollment Reporting



6.8 UC08: Manage Hold

Manage Student Hold



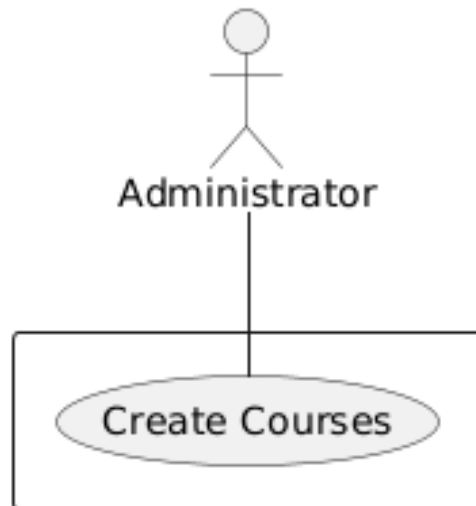
6.9 UC09: Update Changes Report

6.10UC10: Create Courses

Actor :

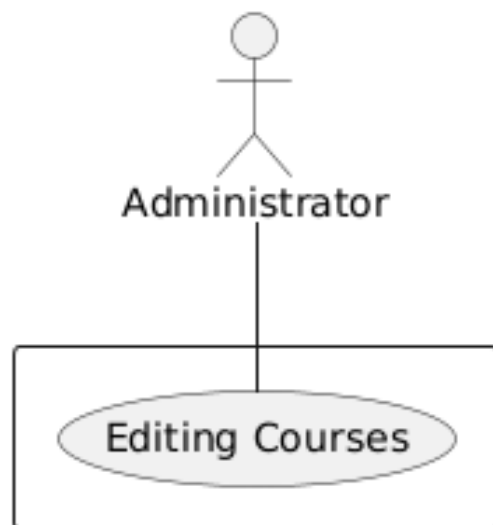
Description:

Create Courses



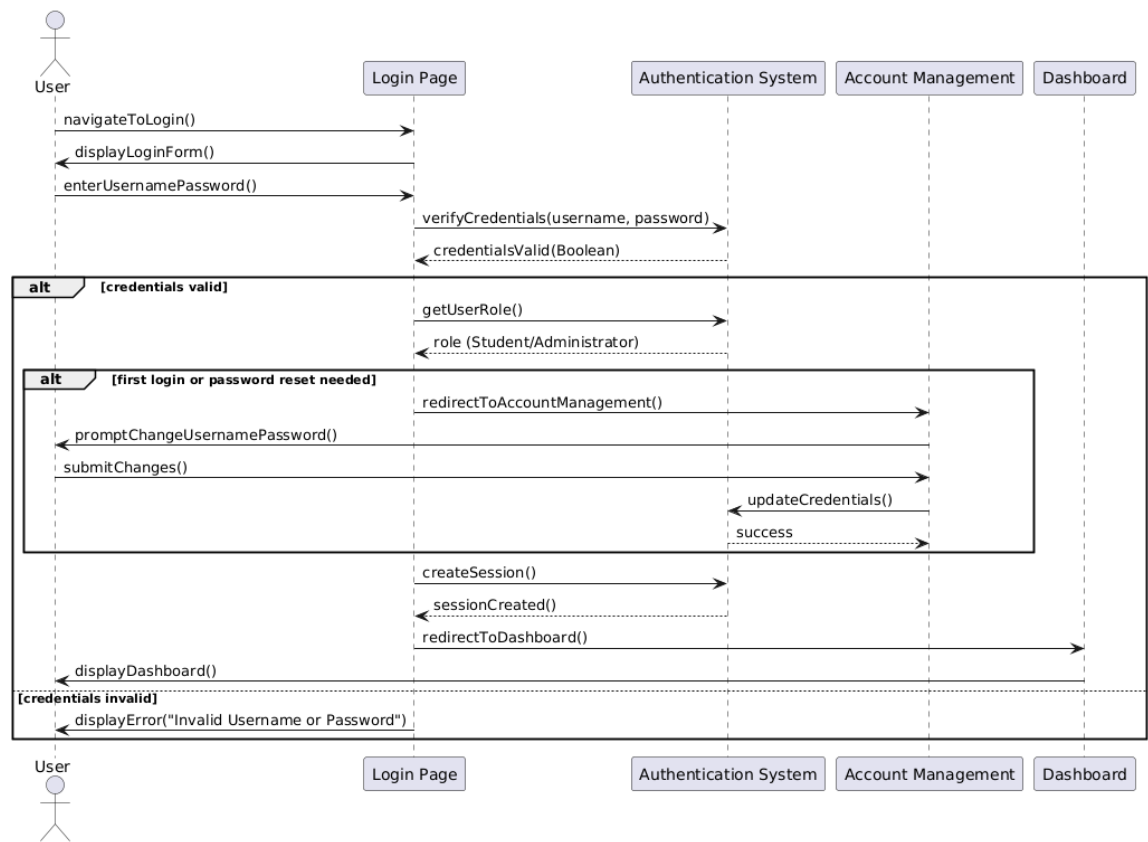
6.11UC11: Edit Courses

Editing Courses

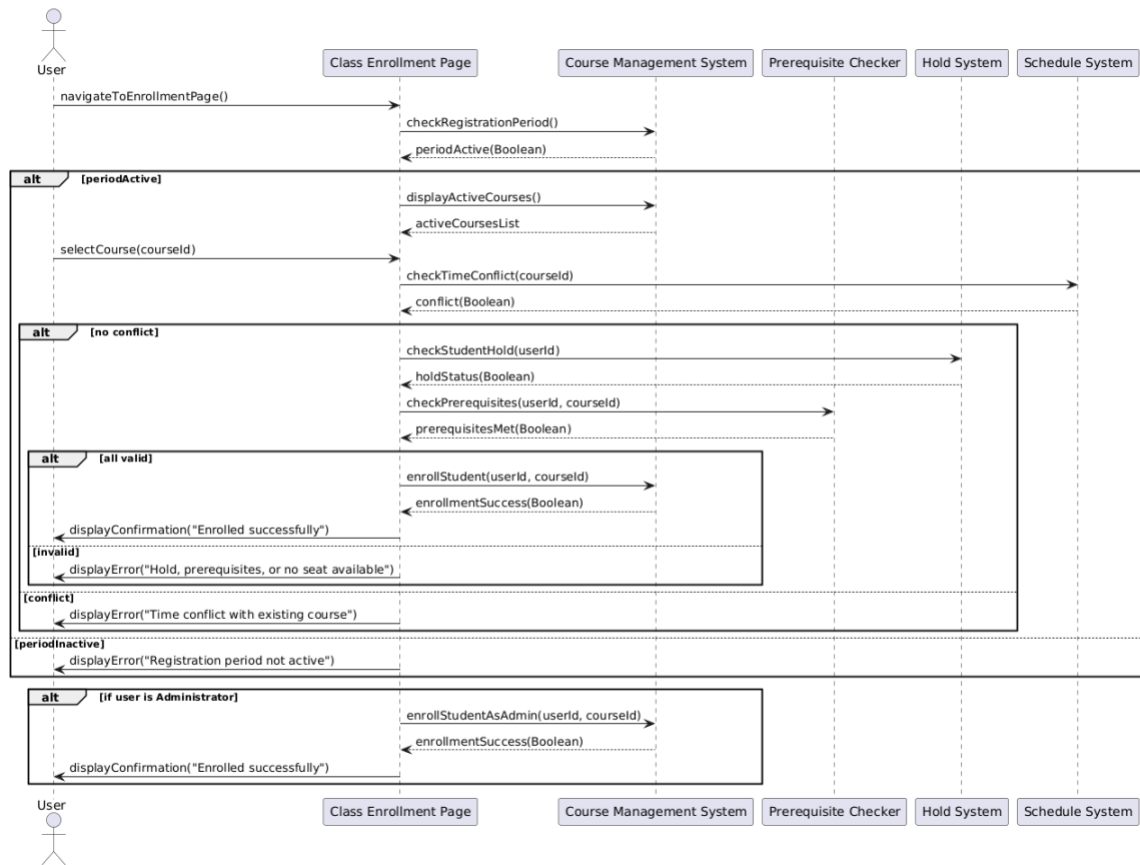


7. Sequence Diagrams

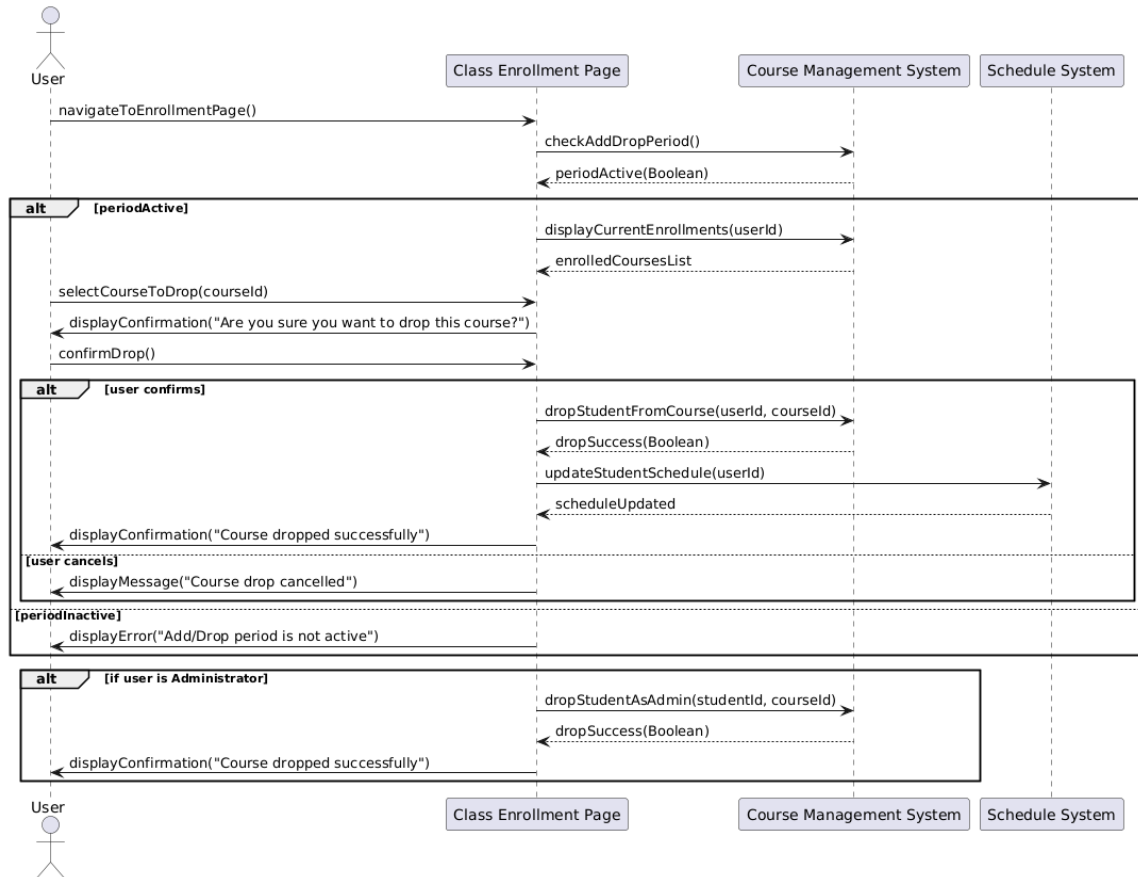
Use Case 1: Manage User Login



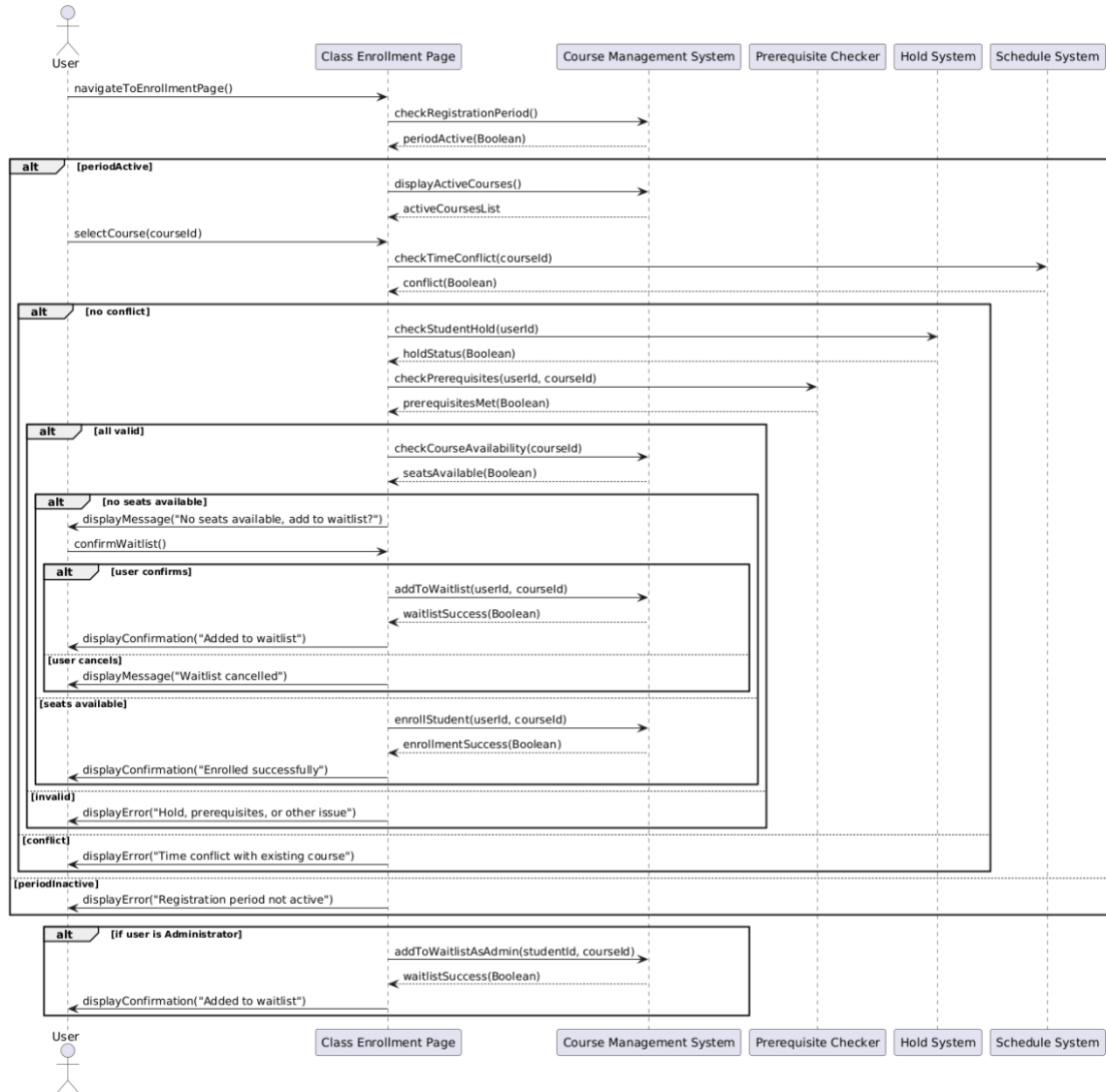
Use Case 2: Course Enrollment



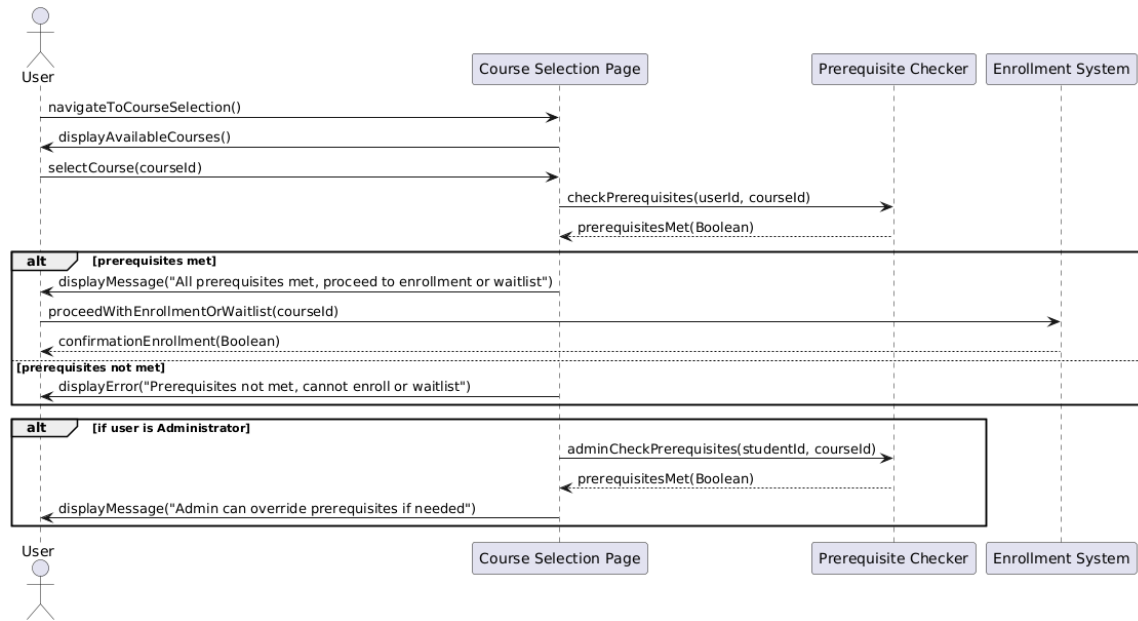
Use Case 3: Course Drop



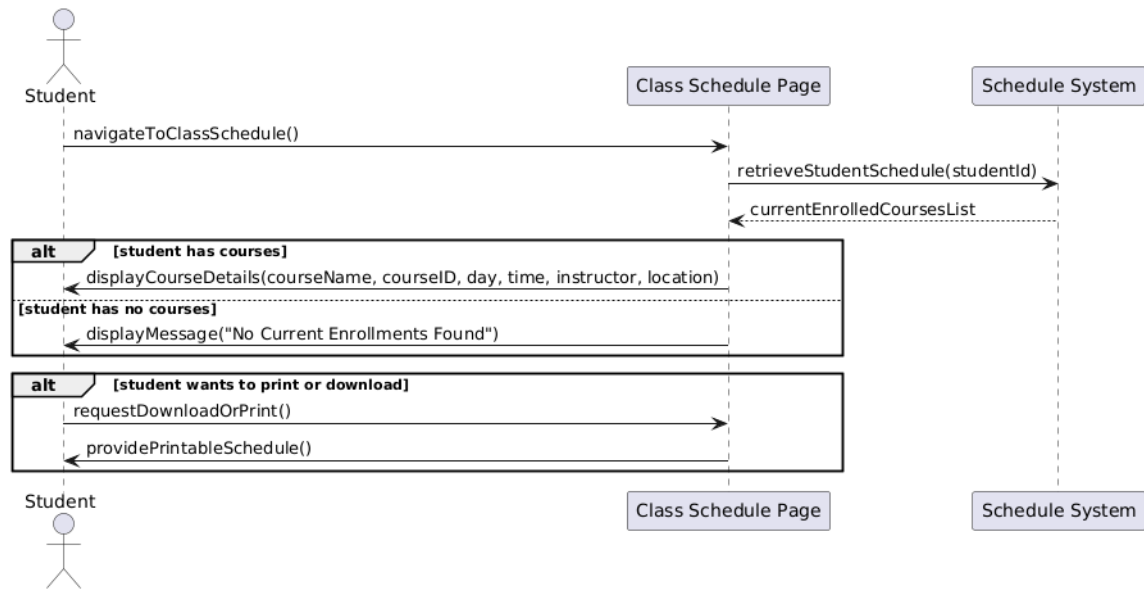
Use Case 4: Course Waitlist



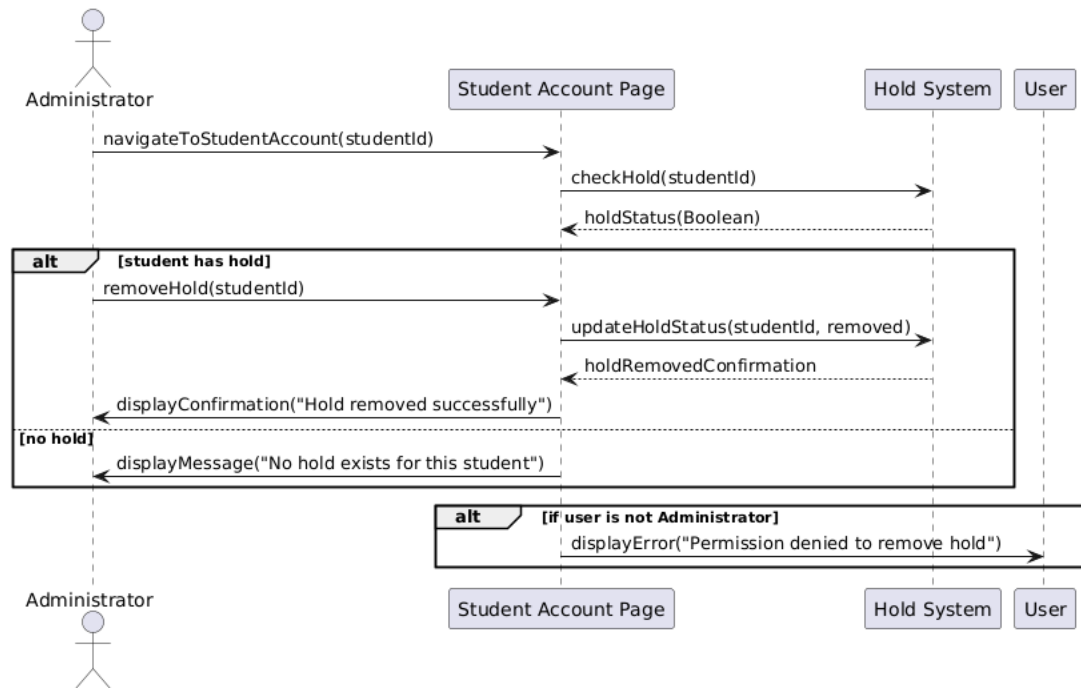
Use Case 5: Course Prerequisites Check



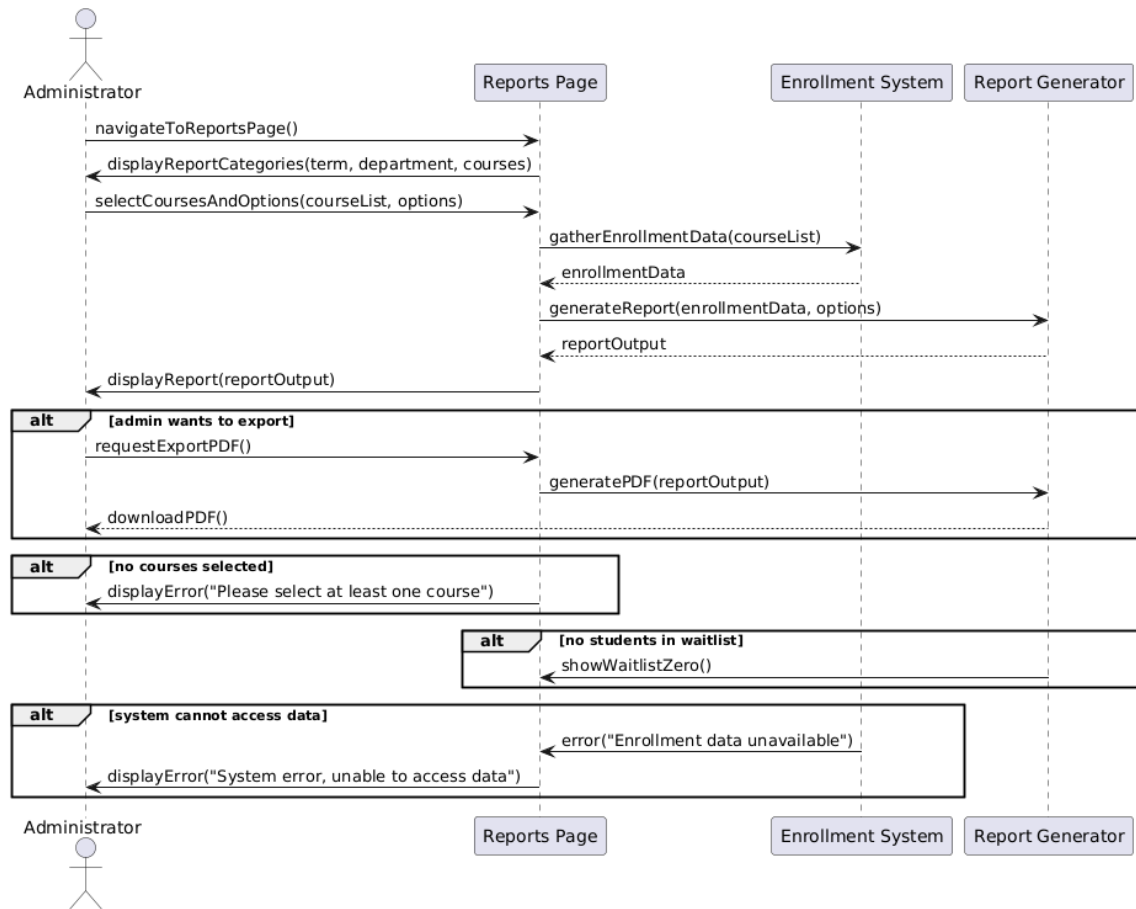
Use Case 6: Access Student Class Schedule



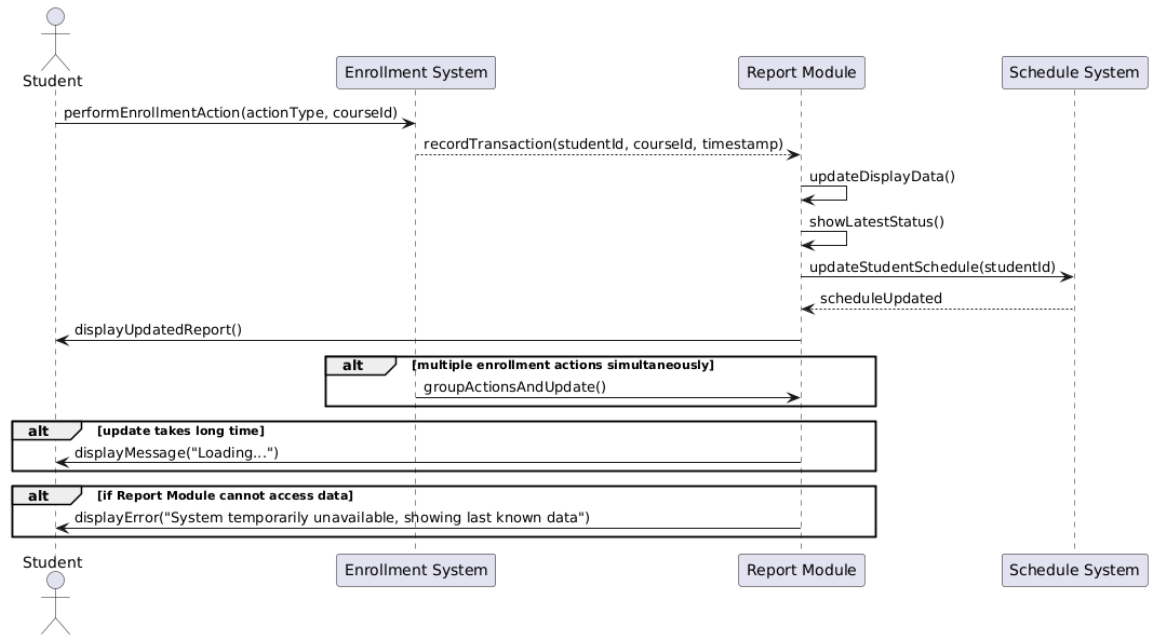
Use Case 7: Manage Student Hold



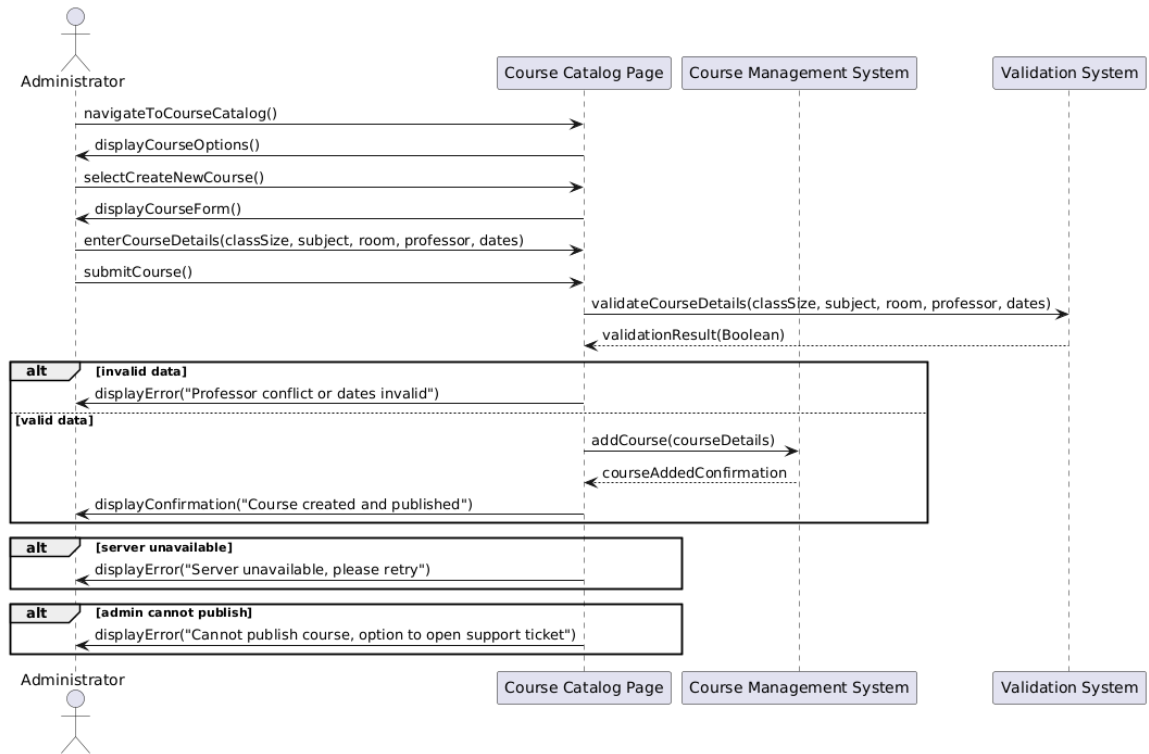
Use Case 8: Enrollment Reporting



Use Case 9: Update Changes Report



Use Case 10: Create courses



Use Case 11: Editing courses

