RISE-EDU

Software Design Specification

Phase 2

Revision History

Date	Revision	Description	Author
10/15/2025	1.0	Initial Version	Wail Mohammed
10/15/2025	1.1	Purpose, Scope, Definitions Updated	Wail Mohammed
10/15/2025	1.2	Updated class candidates 01,02,03,04,05	Wail Mohammed
10/17/2025	1.3	Updated class candidates 06,07,08,09,10	Wail, Yesenia, Emmanuel, Shichang
10/23/2025	1.4	Updated Class Diagram and Sequence diagrams	Wail Mohammed
10/25/2025	1.5	Updated all class candidates, added class 11	Wail, Yesenia, Emmanuel
10/26/2025	1.6	Updated the System Architecture	Emmanuel
10/27/2025	1.7	Updated Course Prerequisites Check	Shichang Wang
10/27/2025	1.8	Updated Project Description, Product Architecture.	Wail Mohammed
10/27/2025	1.9	Updated class candidates 1 to 11. Added description and new diagrams.	Wail Mohammed
10/28/2025	2.0	Added Client, Server, Client Handler and Message classes	Wail Mohammed
10/28/2025	2.1	Updated Product Design Definition to include facade design pattern	Wail Mohammed
10/28/2025	2.2	Updated Class 13 description and product design description	Wail Mohammed
10/28/2025	2.3	Updated Class 16 GUI manager	Shichang Wang
10/28/2025	2.4	Update Report diagram	Shichang Wang
10/28/2025	2.5	describing actor and description for the use cases part	Shichang Wang

Table of Contents

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 and Design Pattern

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 is consolidated into a single System Manger class which follows a facade design pattern to provide a simple and unified interface to all complex sub classes within the system.

2.3.2 System Manager: All requests from the client handler (server side) will use the System Manager to coordinate with other core sub classes to preform client requests. The system manager will manage data classes to create objects such as users, universities, courses, as well as processes the core enrollment , drop , and withdrawal requests by delegating the work to the other specialized sub classes. The responsibilities of the System Manager facade will include the following:

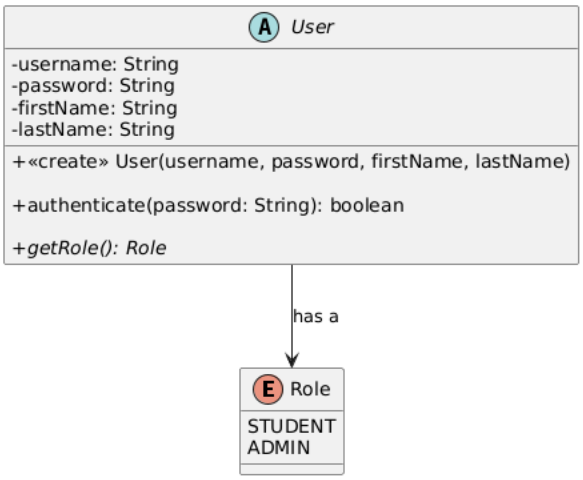
- 2.3.2.1 User Management: Once a user object is created and found, this will handle user authentication using a predefined users data file, by examining the password.
- 2.3.2.2 Course Management Once a course is created and found, this will handle updating, deleting, editing, and updating of all course information that include class size, waitlist size, and prerequisites.
- 2.3.2.3 Enrollment Management: To handle student course enrollment , course drop , course withdrawal , and checking for waitlist size. This will check for enrollment course limits and check prerequisites.
- 2.3.2.4 Report Management: 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.
- 2.3.2.5 Data Management: To handle all system data. It will use a locally created database files to load and save data objects of users, courses, universities. As well as handle enrollment actions such as enroll, drop, waitlist, and withdraw.

2.3.3 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

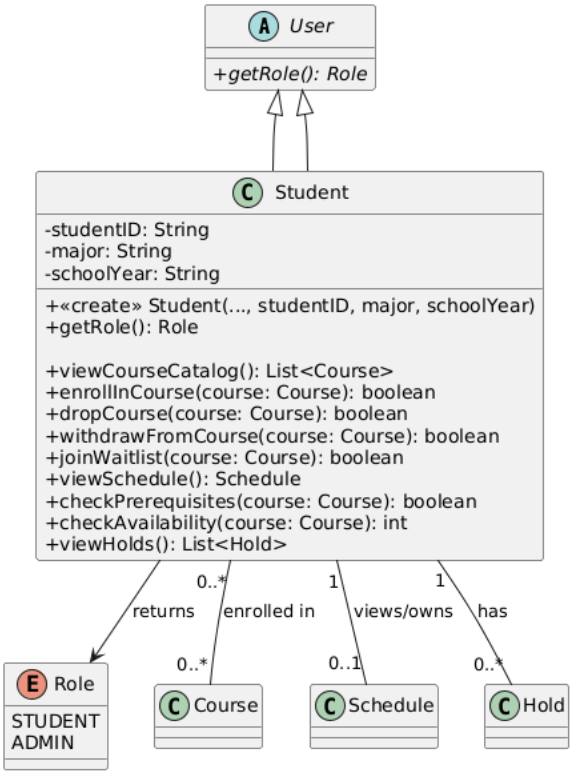
Class 1: User

Description : Abstract user class which prevents the creation of a new User object.



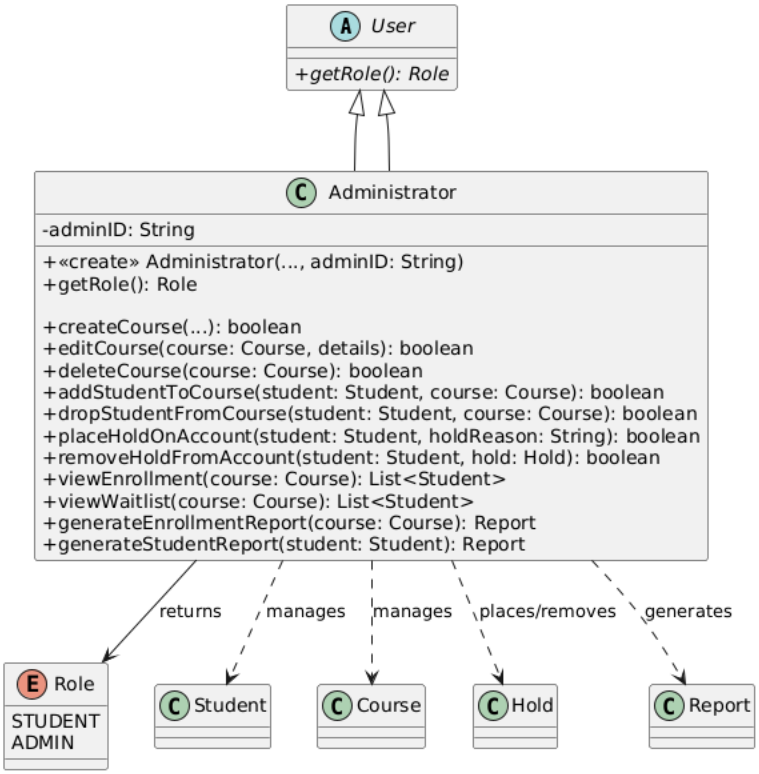
Class 2: Student

Description: This class represents a student user. It inherits all the properties of User (like username, password) and adds student-specific methods such as viewCourse, enrollInCourse, viewSchedule, etc..)



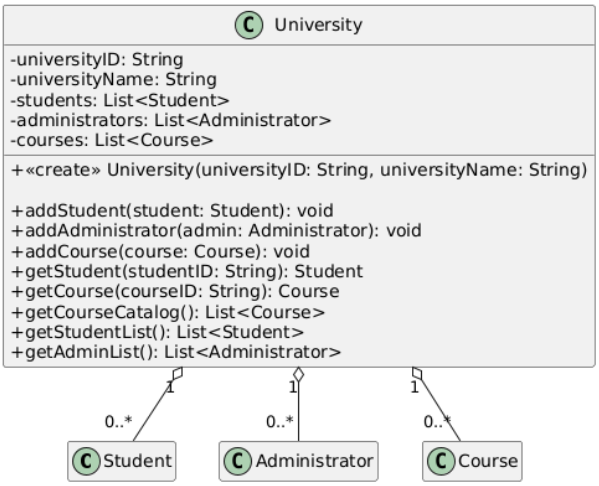
Class 3: Administrator

Description: This class represents an admin user. It inherits all the properties of User (like username, password) and adds admin-specific methods such as addStudent, createCourse, placeHoldOnAccount, generateEnrollmentReport, etc..)



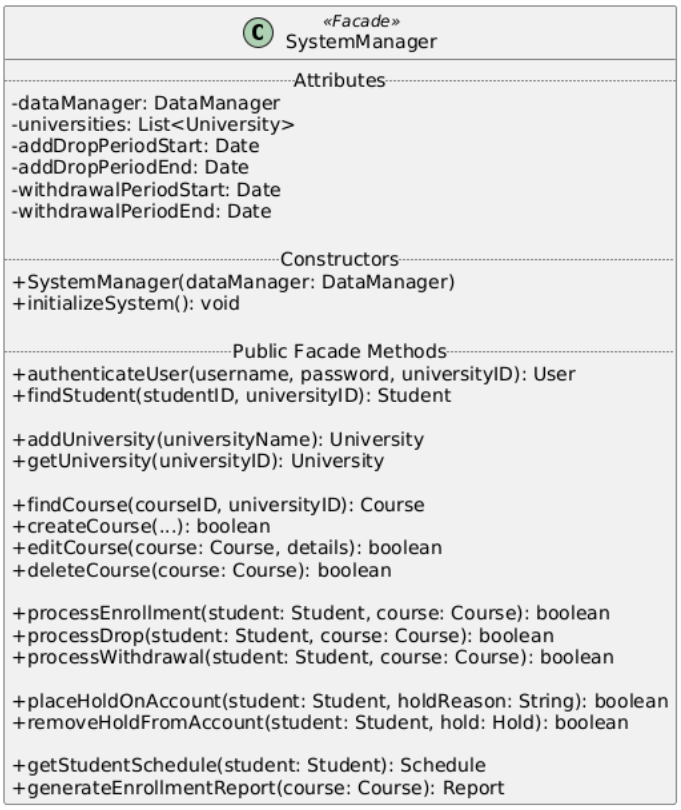
Class 4: University

Description: This class represents a university that contains all students, admins and courses per university. The system manager (next class) will first use this class to first get the correct university, then get all the students or courses dealing with that university.



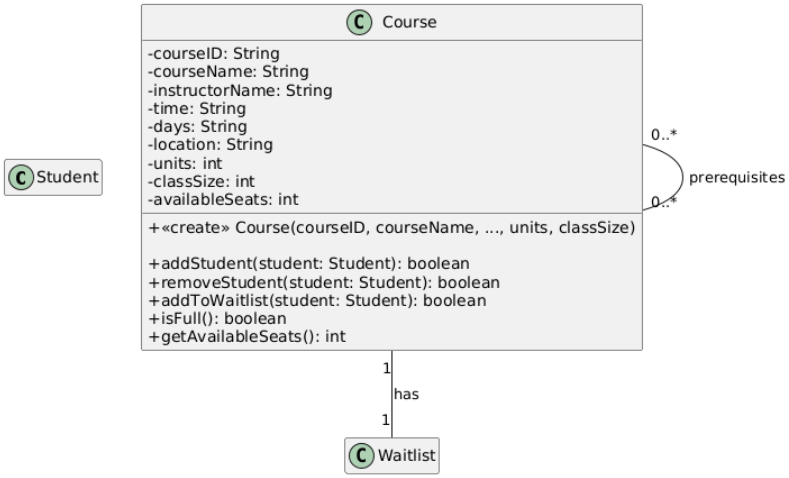
Class 5: System Manager

Description: This class represents the system manager, which handles the core business logic of the application. It acts as a facade for the system by hiding all complex interactions between University class, User class, Student, Course, Data Manager, etc... Therefore providing a single high-level interface which the ClientHandler will use to manage all requests and access system core functions.



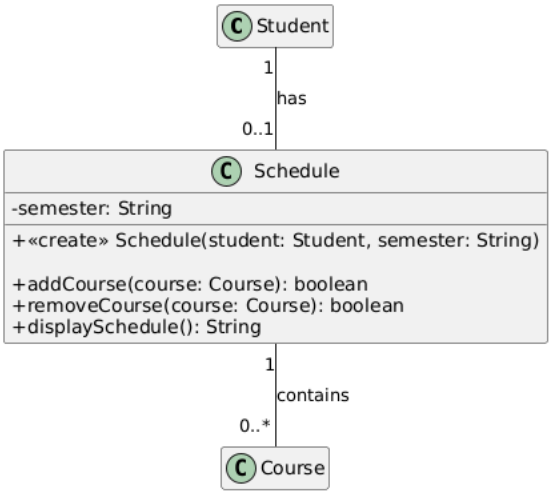
Class 6: Course

Description: This class holds the data for a single course such as course name, time, location, instructor, class size, units, etc. It also manages students being added or removed. It also holds prerequisites if needed.



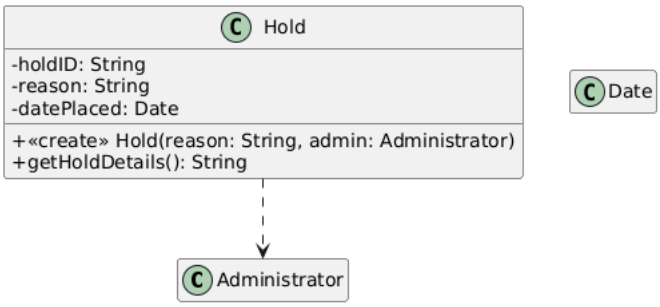
Class 7: Schedule (List of courses)

Description: This class is to hold a list of Course object for a specific student in a specific semester. For instance, if the Student class calls student.viewSchedule() it will create and return a Schedule object.



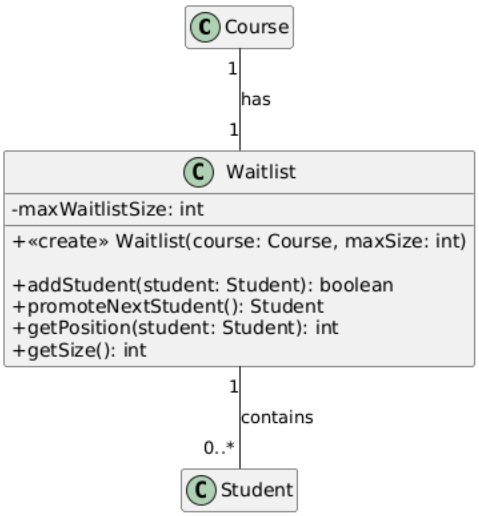
Class 8: Hold

Description: This is a data class that holds any student holds. The Adminstrator class creates an instance of this class whenever an admin needs to place a hold on a particular student.



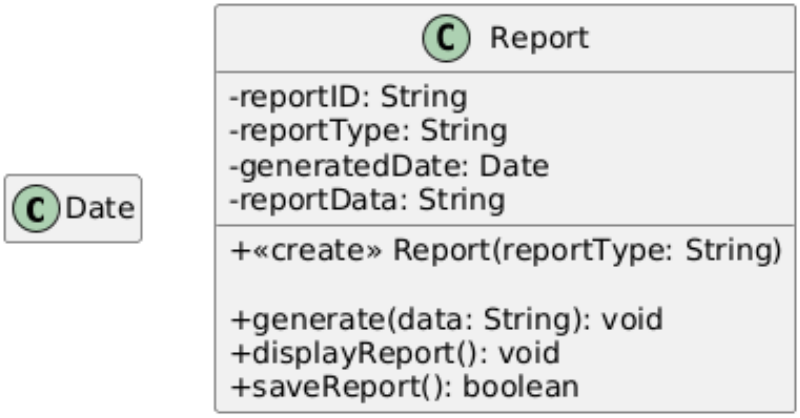
Class 9: Waitlist

Description: This is a data class that holds and manages the waitlist for a particular course. A Course class will create only one Waitlist object. The Waitlist class will then be used to determine the correct order, promote the correct student to join the class or get the current position of a particular student in the waitlist.



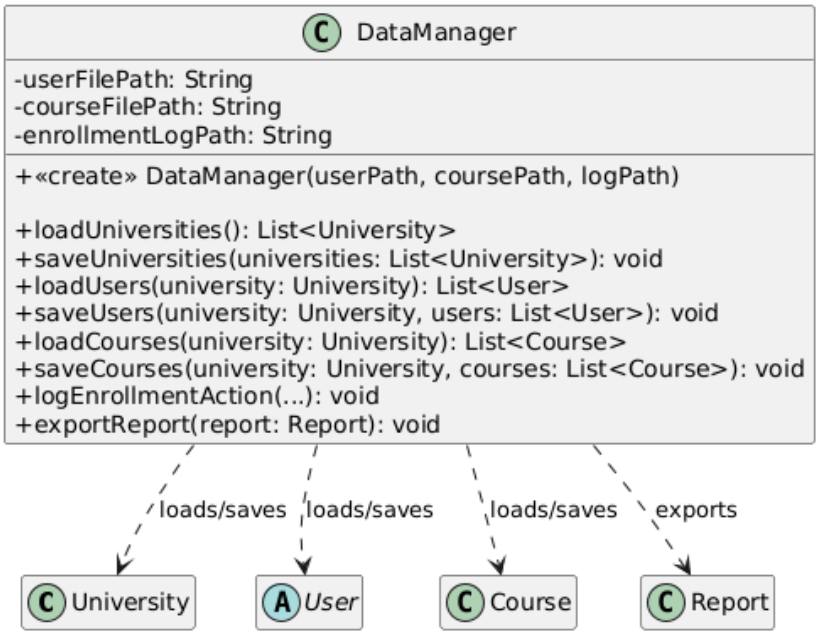
Class 10: Report:

Description: This is a data class that holds the logs which the Administrator class will create an object to pass in data and generate and save reports.



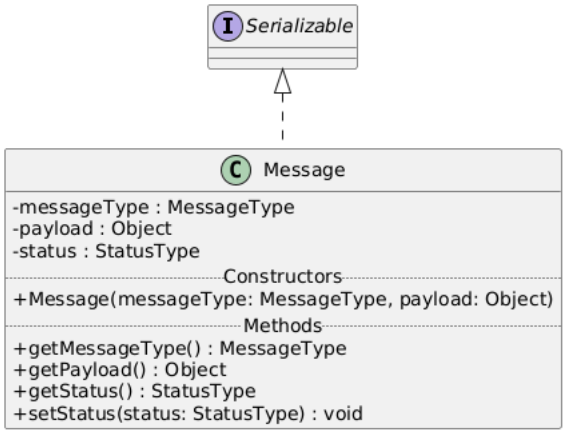
Class 11: Data Manager:

Description: This class will handle the data management, saving and loading of universities, users and courses to and from local file which helps to keep file-handling logic in one place.



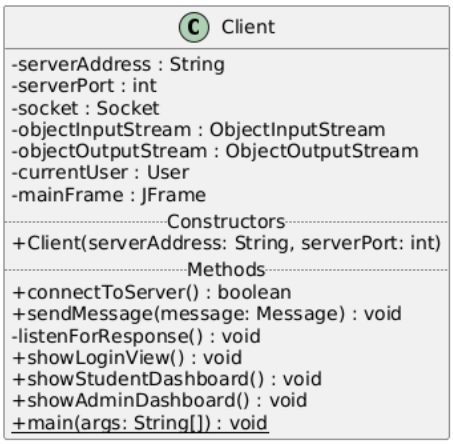
Class 12: Message

Description: This class will act as a serializable class to be sent over a network, which will be used to send request/response back and forth between the client and server. It will also use Enums to encapsulate the MessageType when sending or receiving requests and responses.



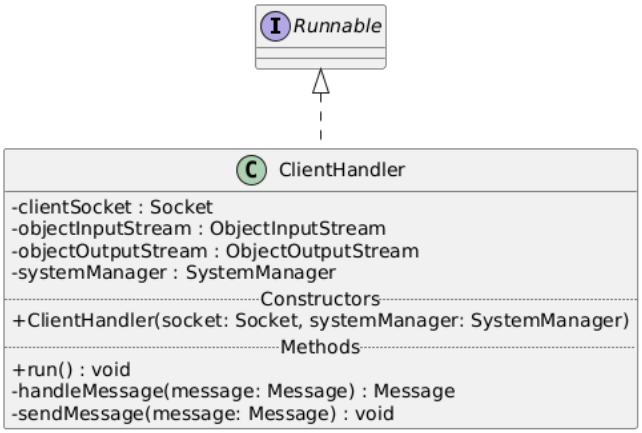
Class 13: Client

Description: This class will act as the interface and point of access to the user to log in and send commands to the server through Message objects. It will contain the GUI for the user and provide two views one for Student user and another for Administrator user.



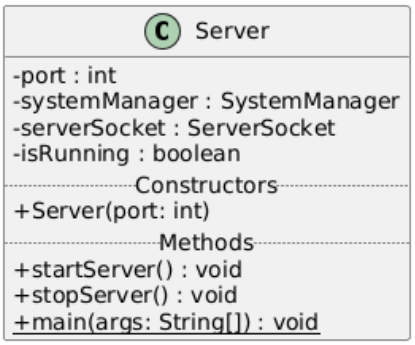
Class 14: Client Handler

Description: This class will allow the system to use a multithreaded client-server design pattern, where the server after connecting to a client, will have this class handle the request/response while the server handles other connections before passing them to this class.

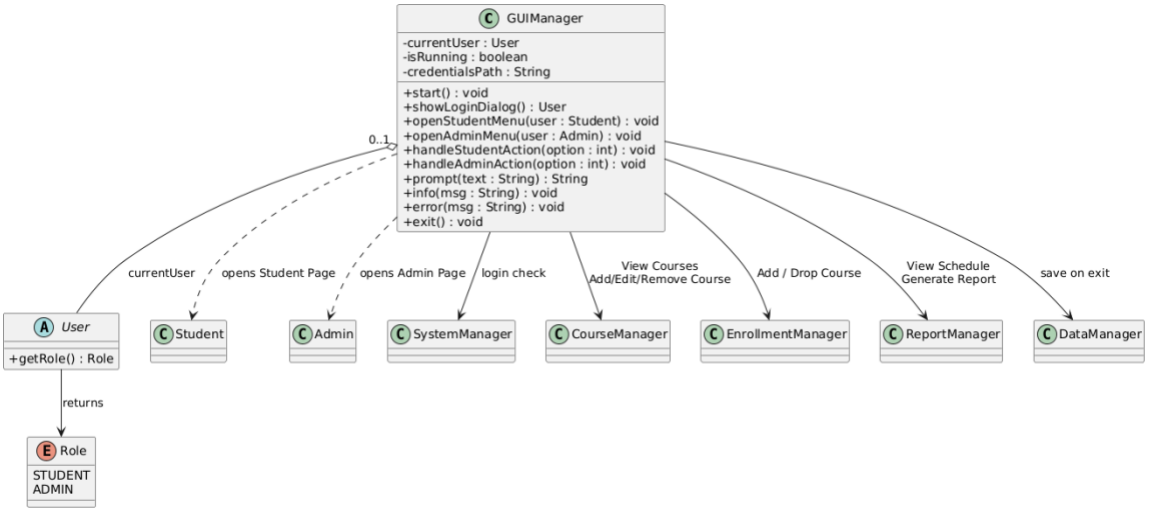


Class 15: Server

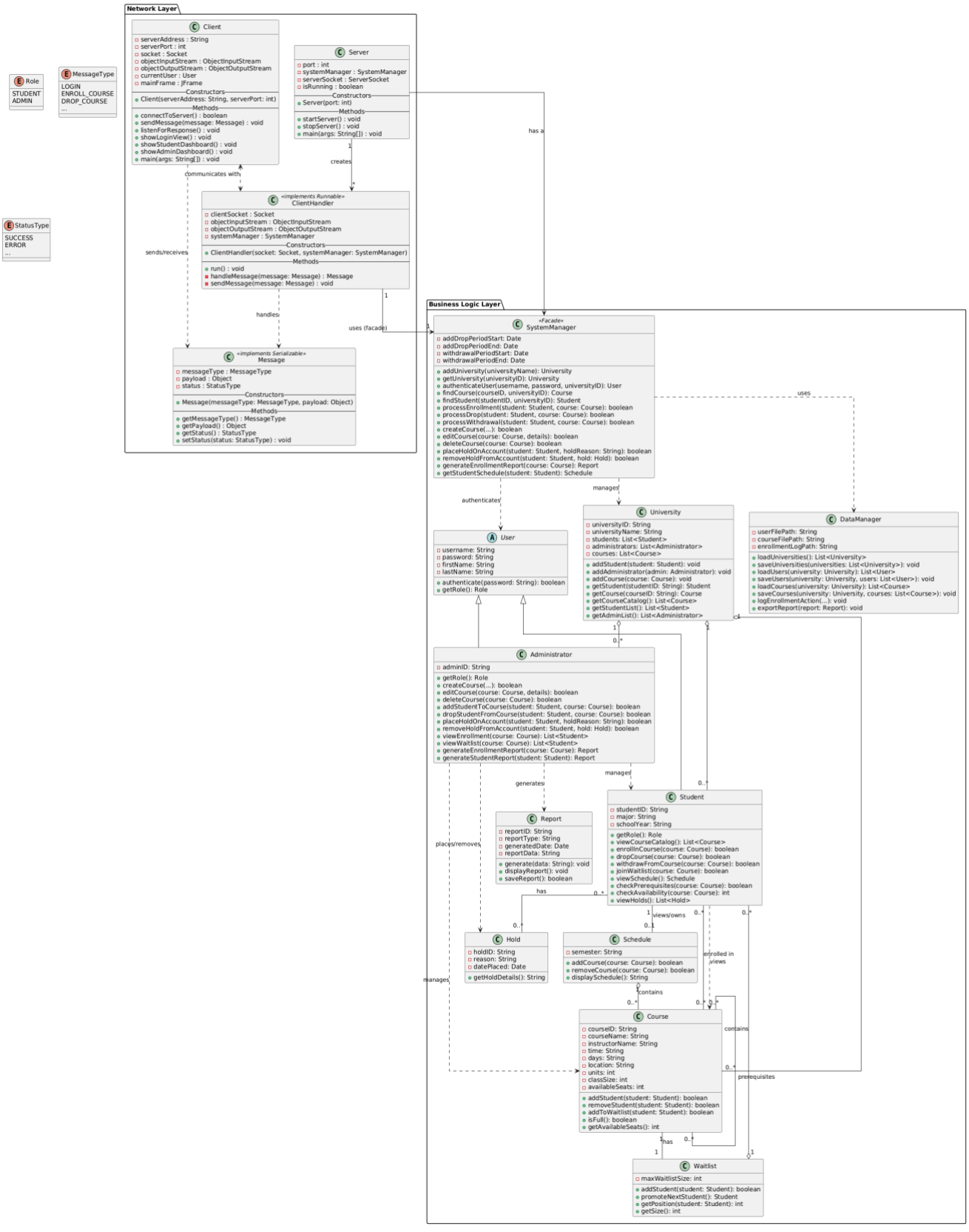
Description: This class is the server class, that will handle the core business logic and data validation in the system. It will listen to new client connections and will pass them to the clientHandler class once a connection is established.



Class 16: GUI Manager
Description:
This class is the main graphical user interface controller of the system. It manages all user interactions through Swing dialog boxes (JOptionPane) and coordinates transitions between the Login, Student, and Administrator menus. The GUIManager reads user credentials, verifies roles, and routes actions to the corresponding manager classes—SystemManager, CourseManager, EnrollmentManager, ReportManager, and DataManager. It also maintains the current user session, controls menu loops, displays messages, and saves data when the user exits.



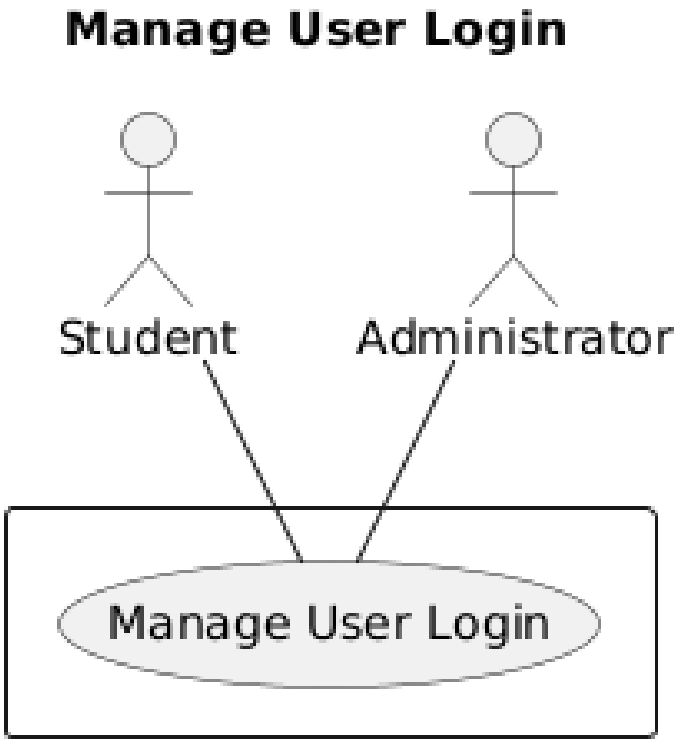
4. Class Diagram



6. Use Cases

6.1 UC01 : Manage User Login

Actor: Student, Administrator
Description:
Students and Administrators to log into the system using their username and password. The system verifies the entered credentials from the stored data file (credentials.txt). If the information matches, the user is authenticated and redirected to their respective interface—Student or Administrator. Invalid inputs trigger an error message, prompting the user to try again.

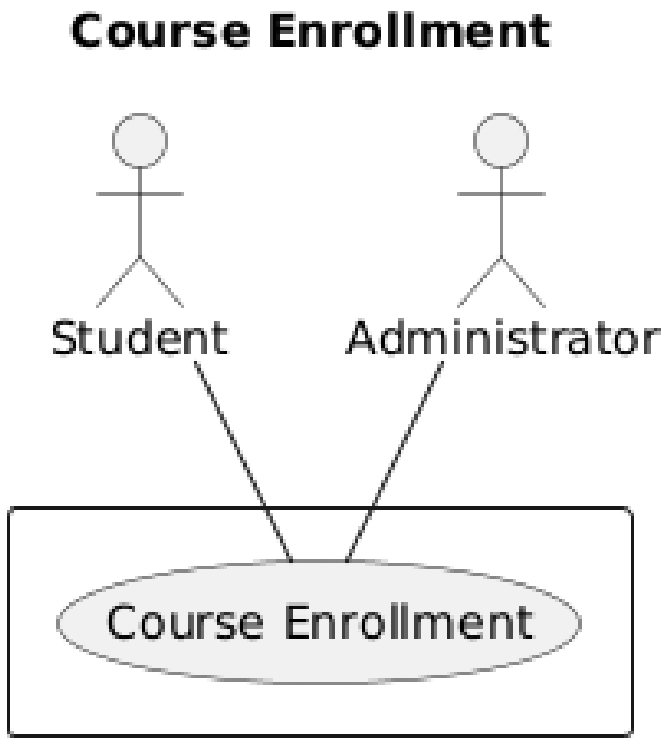


6.2 UC02: Course Enrollment

Actor: Student, Administrator

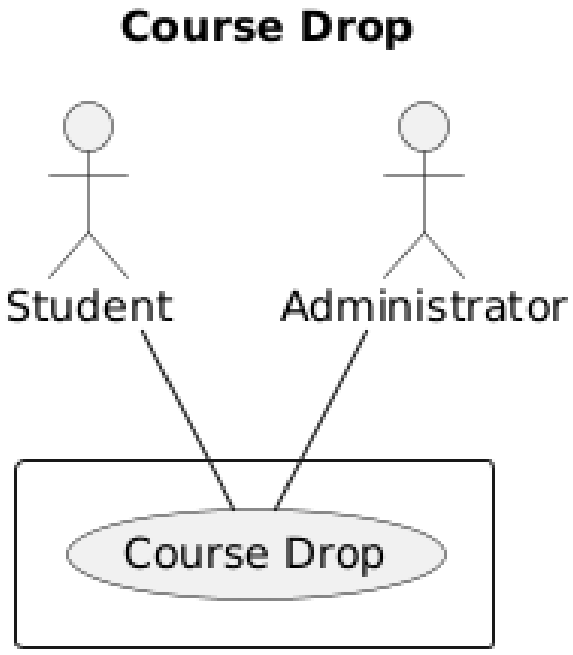
Description:

This use case allows Students to enroll in or drop courses and enables Administrators to manage enrollment records.



6.3 UC03: Course Drop

Actor: Student, Administrator
Description:
This use case allows Students to drop courses they are enrolled in and lets Administrators update or approve course removals. The system validates enrollment records and updates the database accordingly.

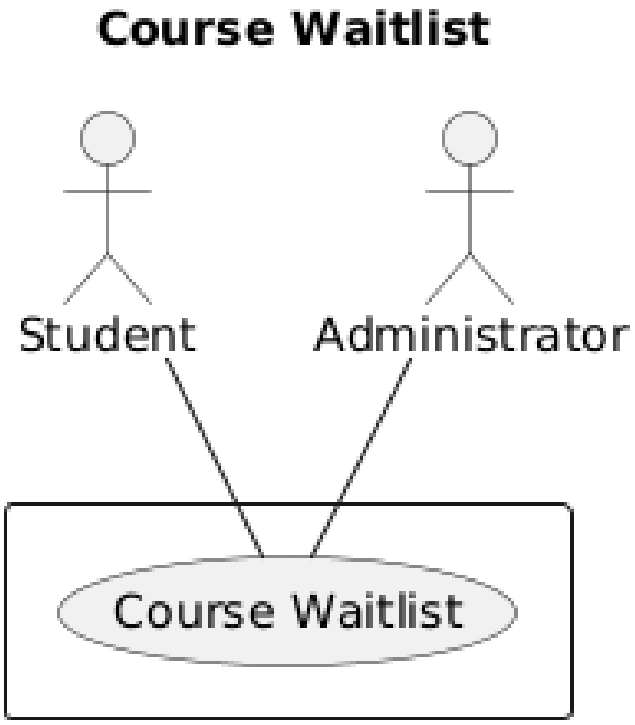


6.4 UC04: Course Waitlist

Actor: Student, Administrator

Description:

This use case allows Students to join a course waitlist when a class is full and enables Administrators to manage waitlisted students. The system records the request, tracks waitlist order, and automatically updates availability when a spot opens. Administrators can also manually adjust or clear the waitlist as needed.

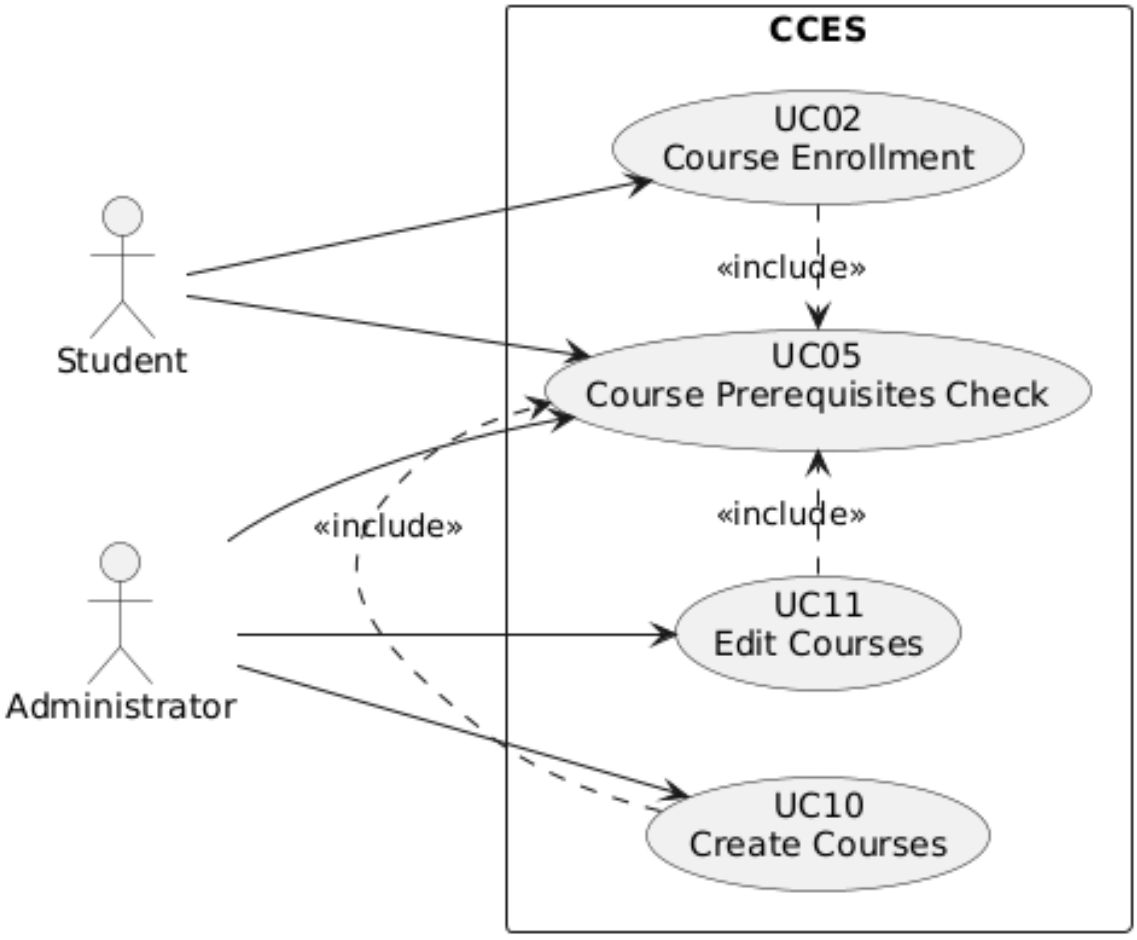


6.5 UC05: Course Prerequisites Check

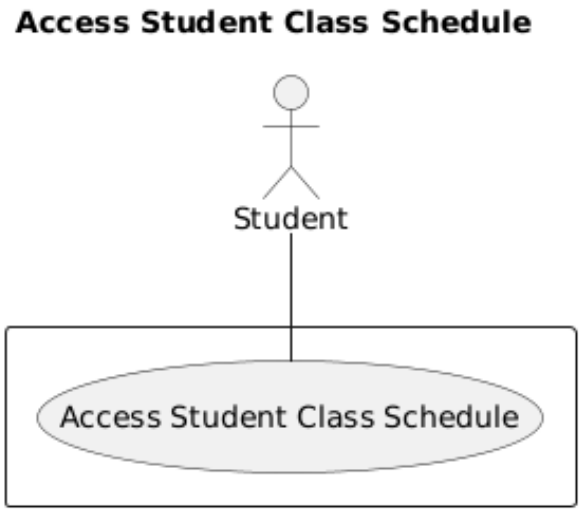
Actor: Student, Administrator

Description:

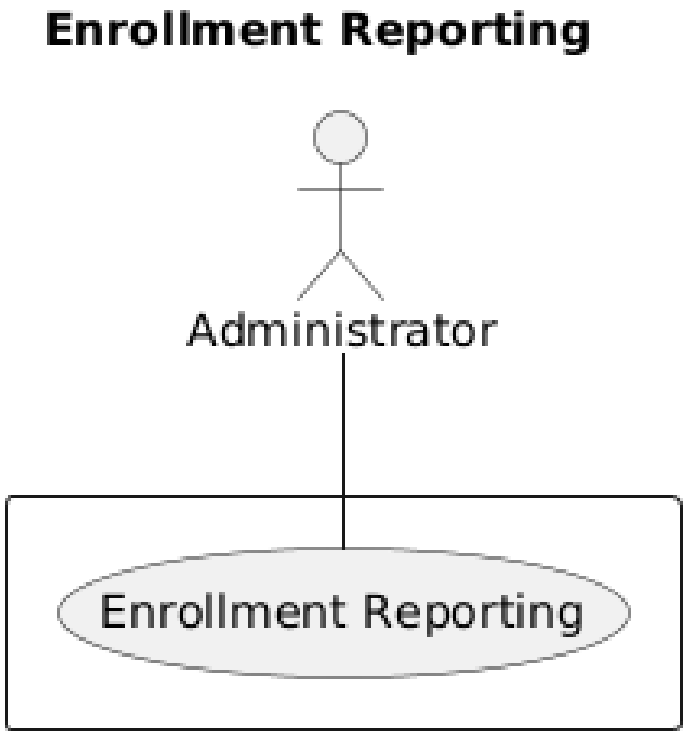
This use case illustrates the interaction between Students and Administrators within the Course Creation and Enrollment System (CCES). Students can enroll in courses, triggering a prerequisite check before registration is confirmed. Administrators can create and edit courses, both of which include the prerequisite validation process to ensure course integrity. The system manages all dependencies automatically and updates course data accordingly.



6.6 UC06: Access Schedule
Actor: Student
Description:
This use case allows Students to view their current class schedule, including enrolled courses, meeting times, and instructors. The system retrieves schedule information from stored enrollment data and displays it in an organized format for the student's reference.

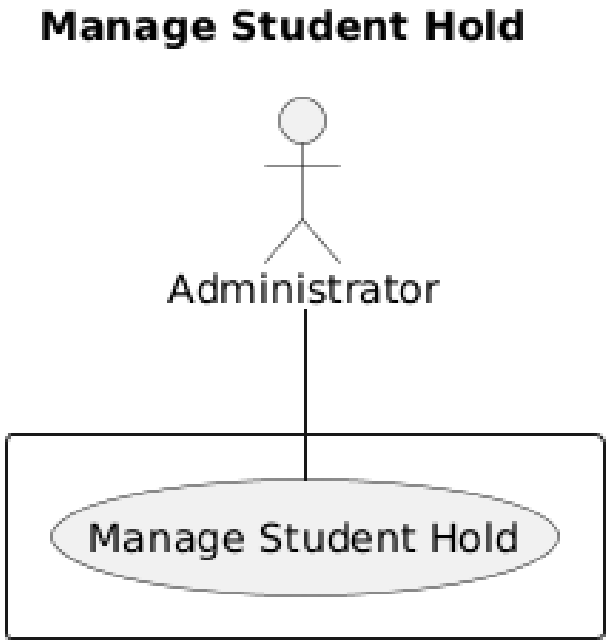


6.7 UC07: Enrollment Reporting
Actor: Administrator
Description:
This use case allows Administrators to generate and view enrollment reports that summarize student registration data. The system compiles course and enrollment information, formats it into a report, and provides options to display or save the results for administrative review.



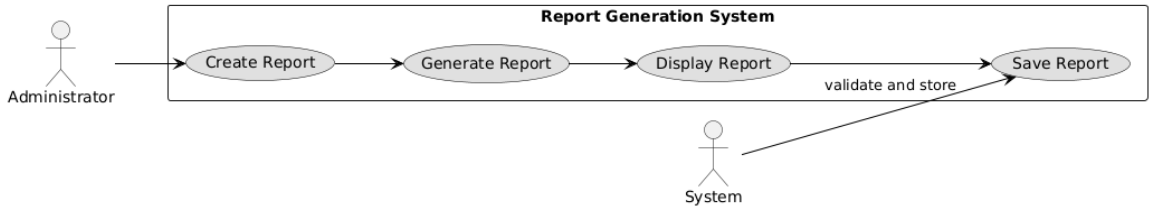
6.8 UC08: Manage Hold

Actor: Administrator
Description:
This use case allows Administrators to view, add, or remove student holds that restrict course registration or other system access. The system updates the student's record accordingly and ensures that hold information is reflected in enrollment and reporting modules.

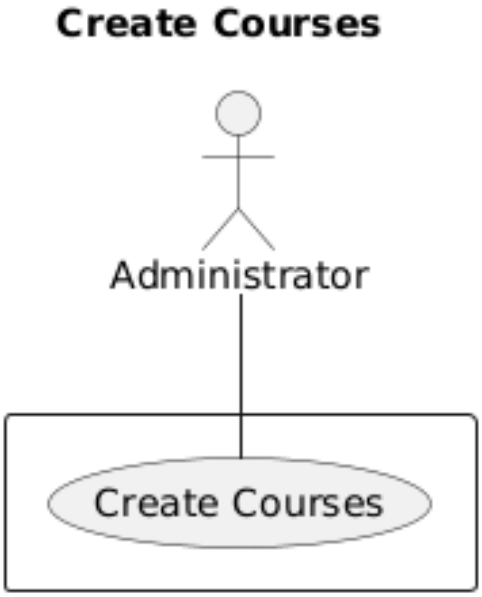


6.9 UC9: Update Changes Report

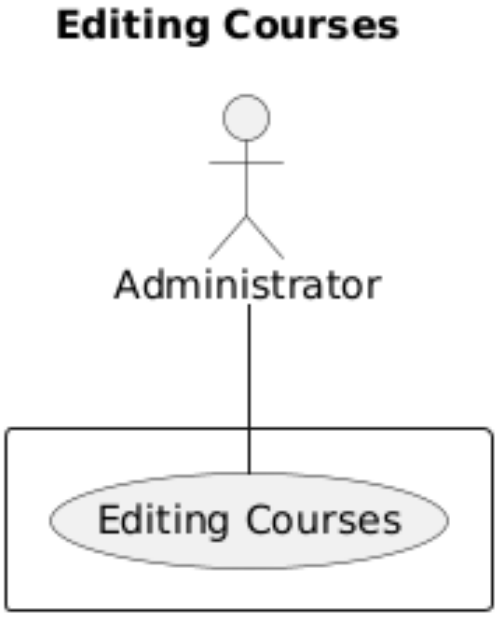
Actor: Administrator, System
Description:
This use case allows the Administrator to create, generate, and display updated reports reflecting recent system changes. The System validates the data and stores the final report for future access. The process includes creating a new report, generating updated content, displaying the output for review, and saving it into the report database.



6.10UC10: Create Courses
Actor: Administrator
Description:
This use case allows the Administrator to create new courses by entering course details such as course ID, title, subject, schedule, and assigned instructor. The system validates the information, checks for duplicate course IDs, and stores the new course in the course database for student access and enrollment.

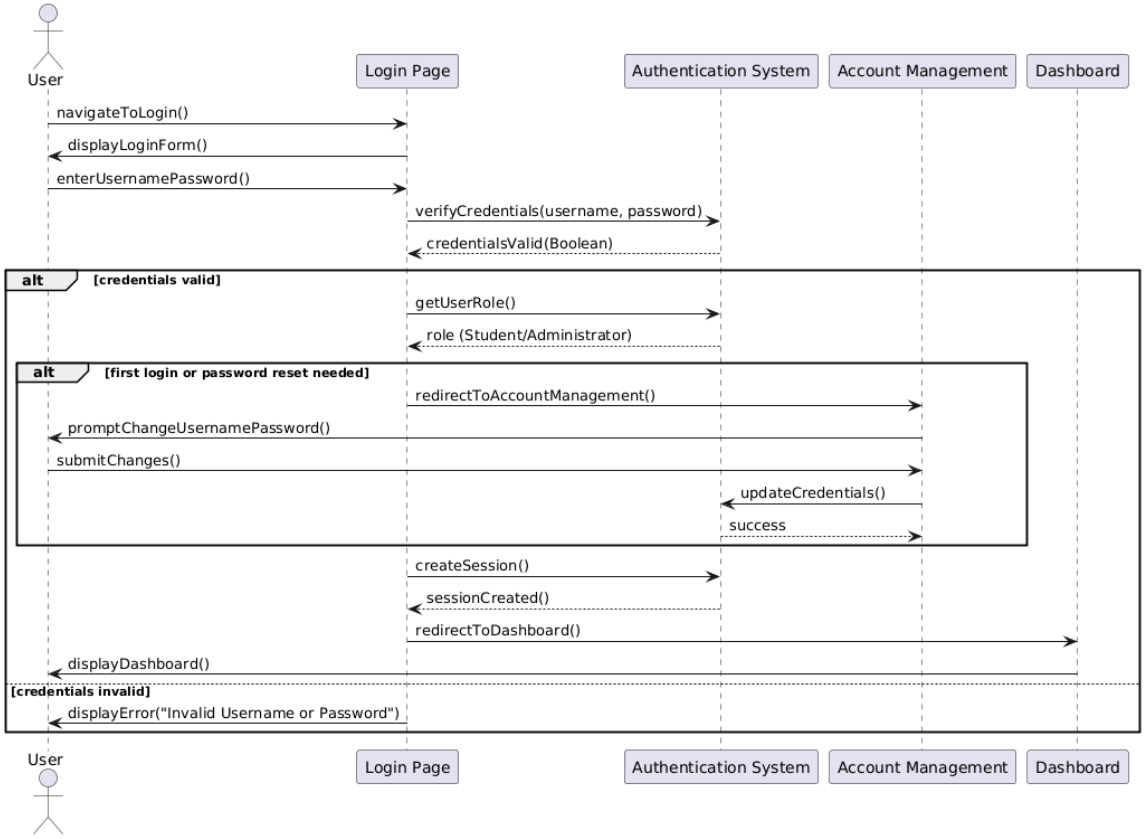


6.11UC11: Edit Courses
Actor: Administrator
Description:
This use case allows the Administrator to modify existing course information such as course title, instructor, schedule, or capacity. The system verifies that the course exists, validates all updated details, and saves the modifications to ensure accurate course and enrollment records.

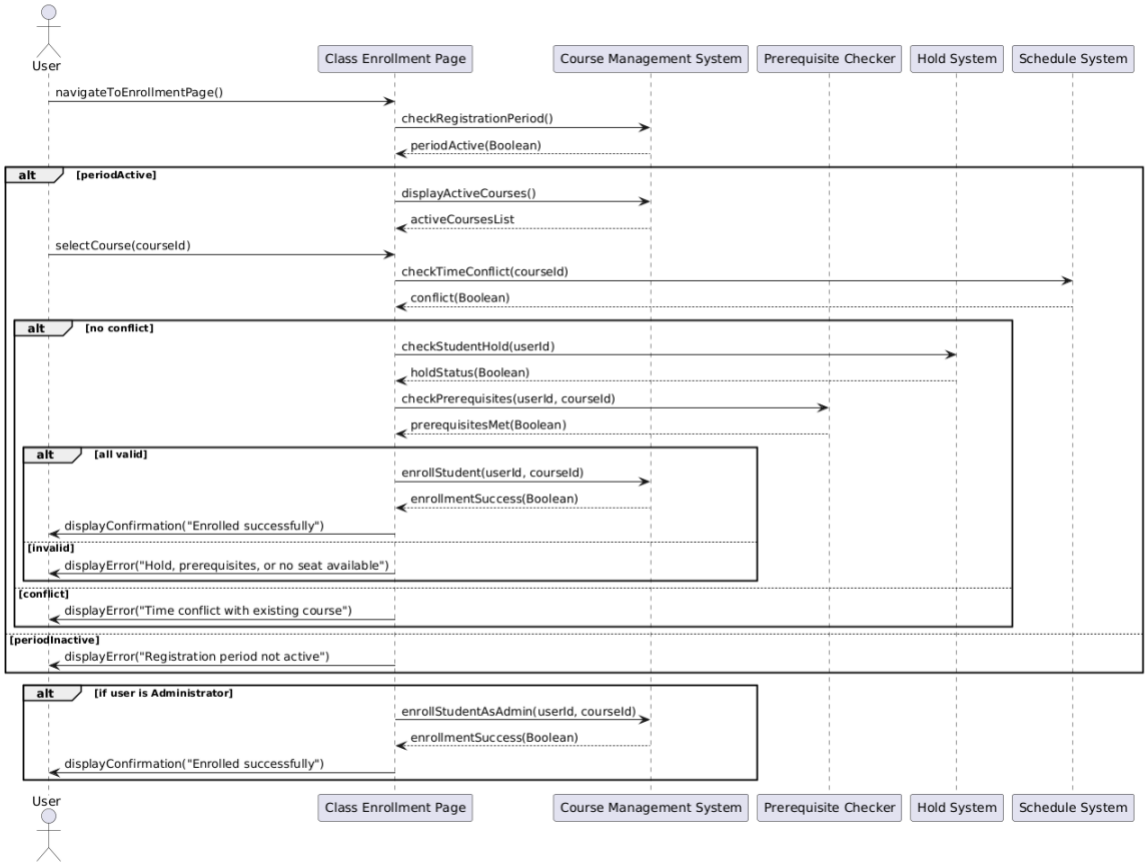


7. Sequence Diagrams

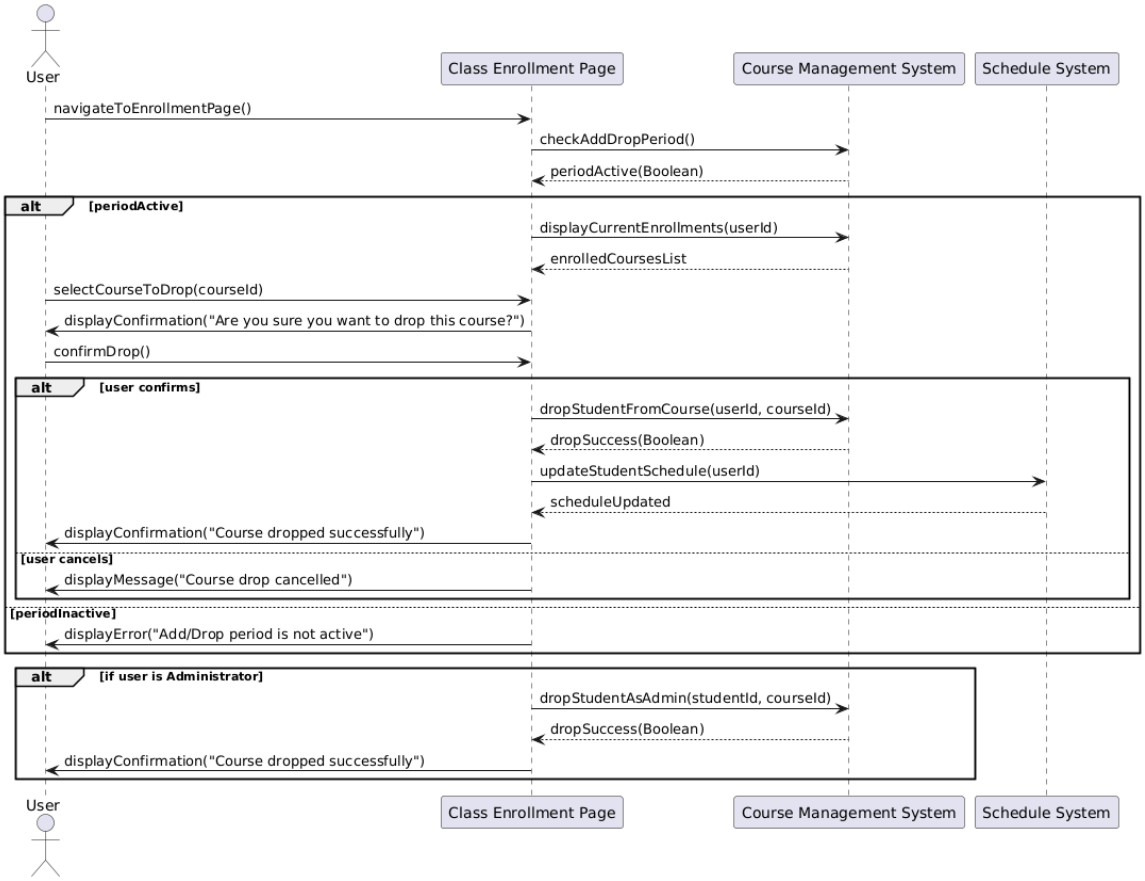
Use Case 1: Manage User Login



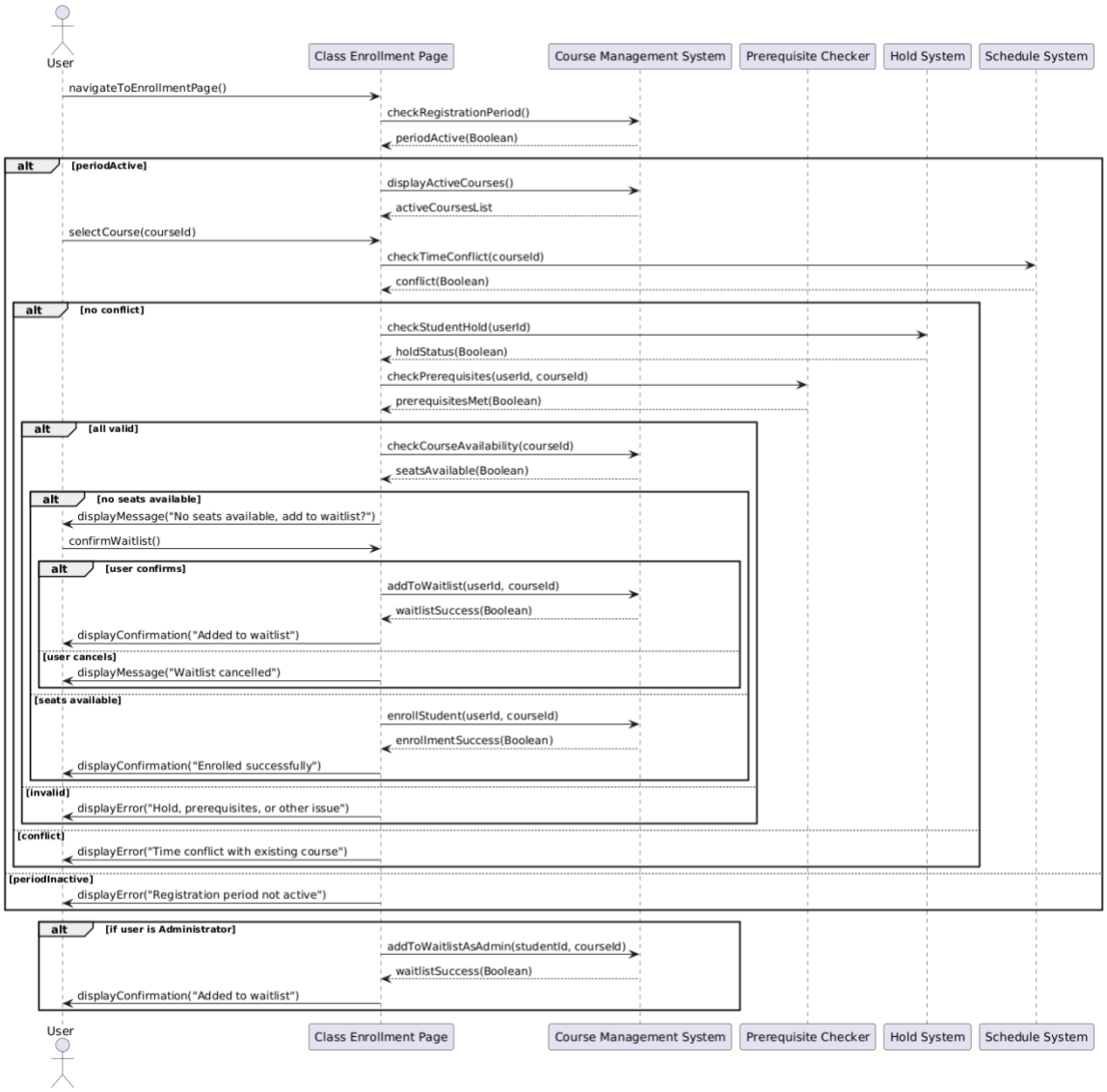
Use Case 2: Course Enrollment



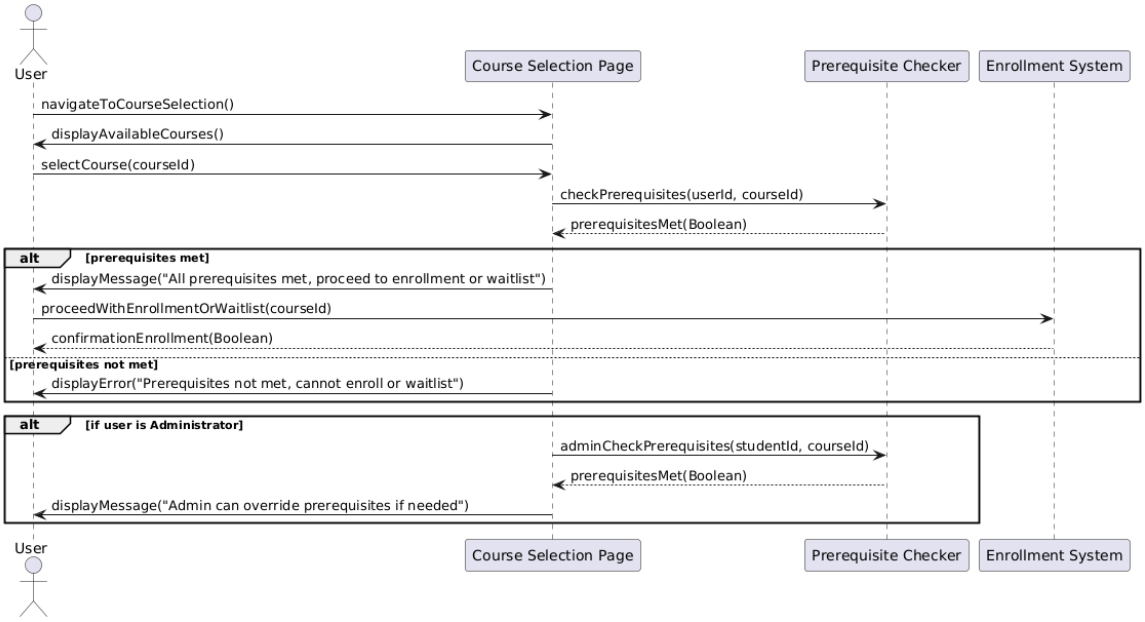
Use Case 3: Course Drop



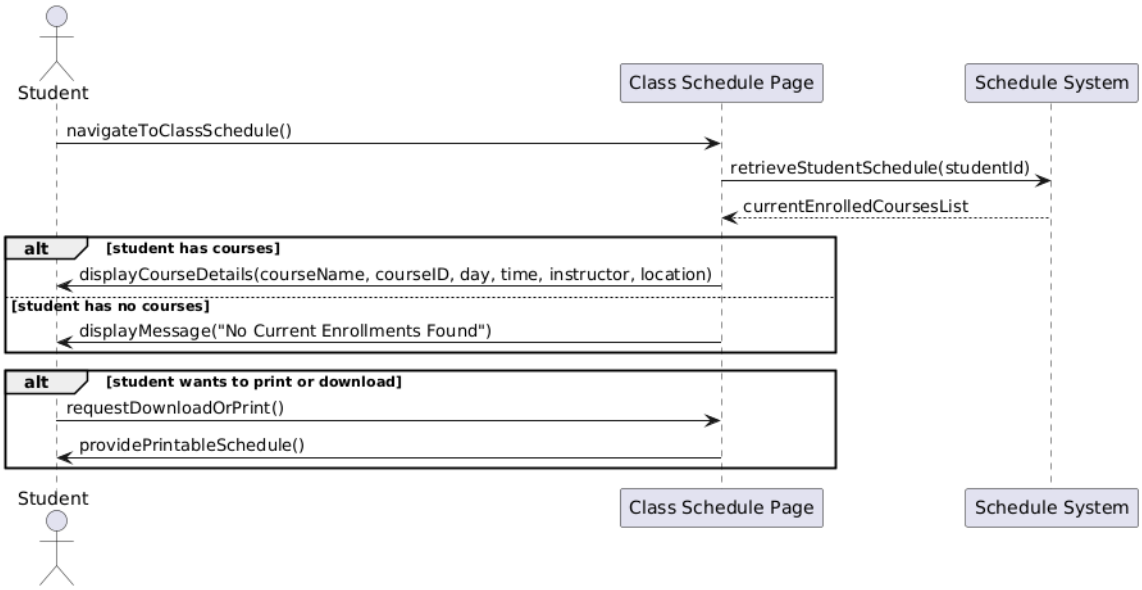
Use Case 4: Course Waitlist



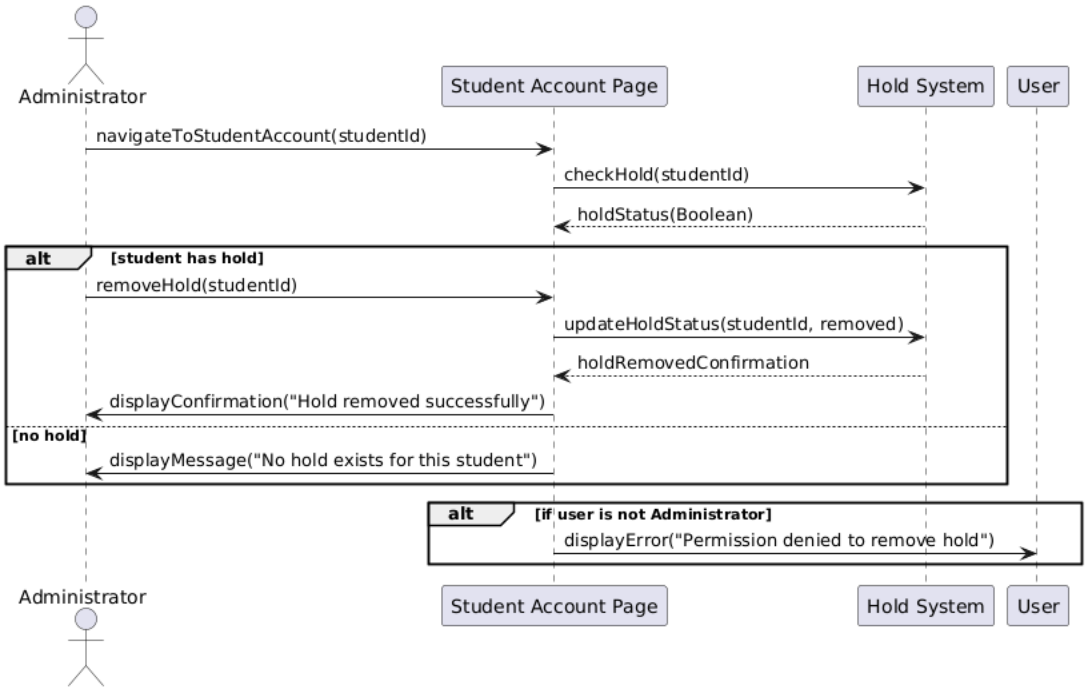
Use Case 5: Course Prerequisites Check



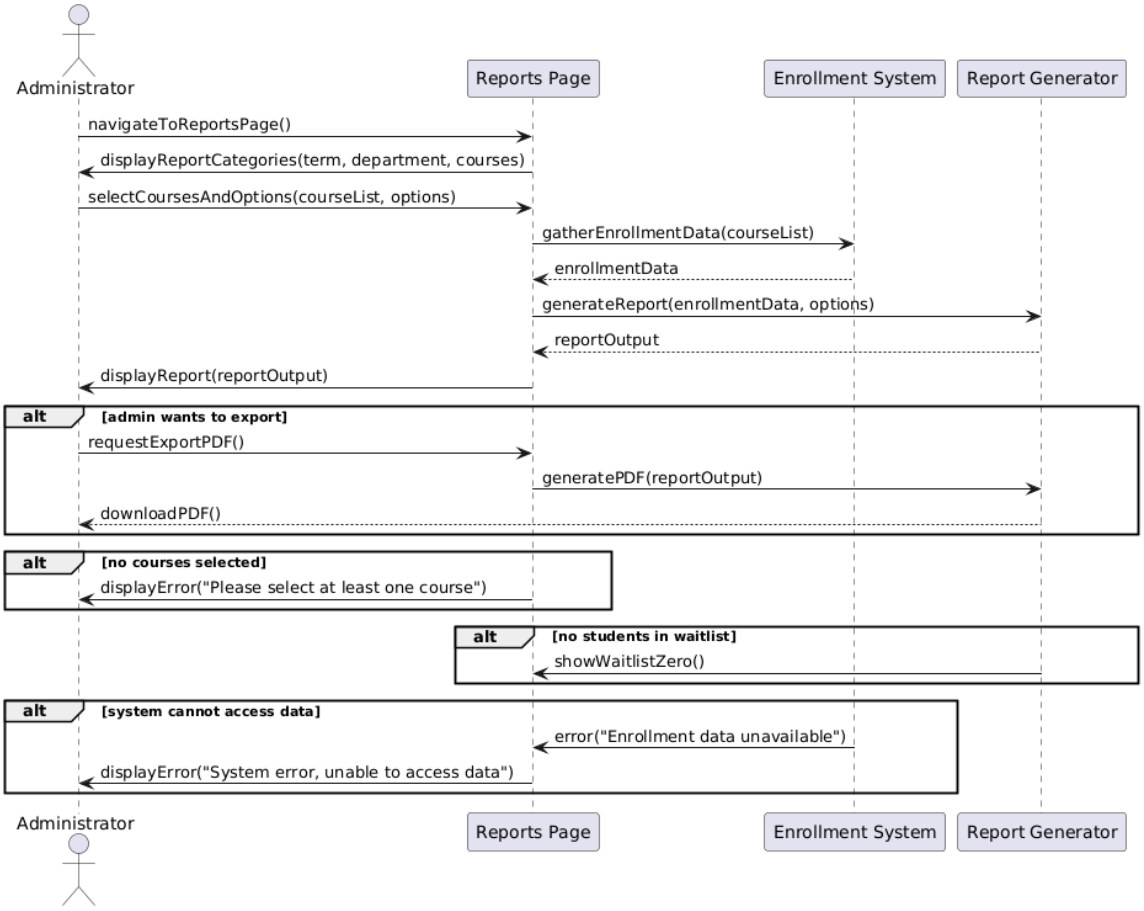
Use Case 6: Access Student Class Schedule



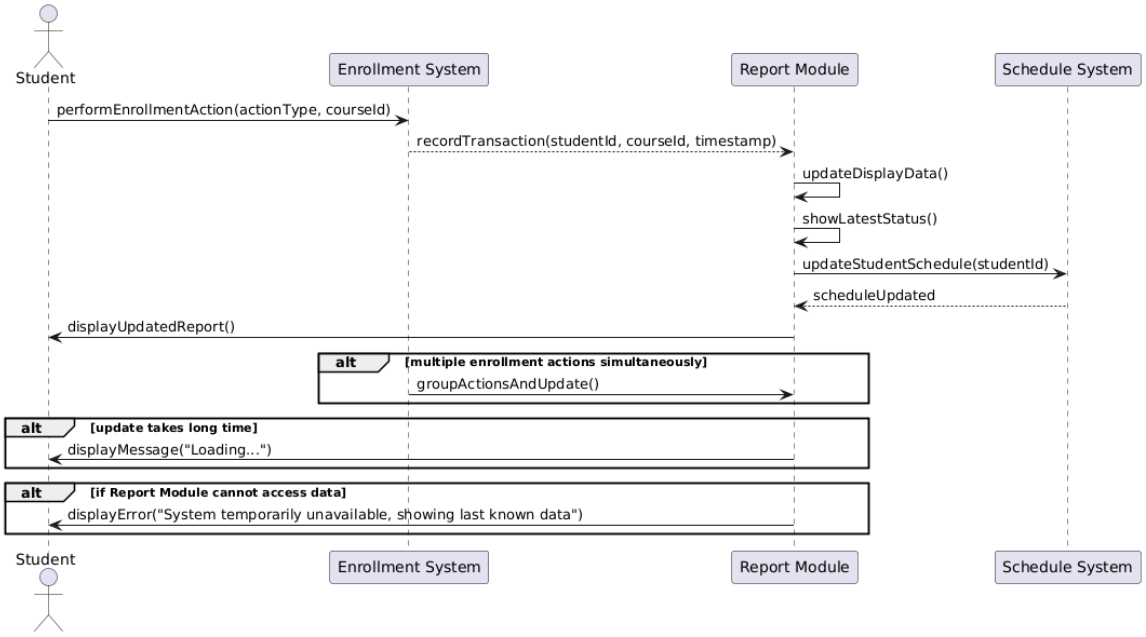
Use Case 7: Manage Student Hold



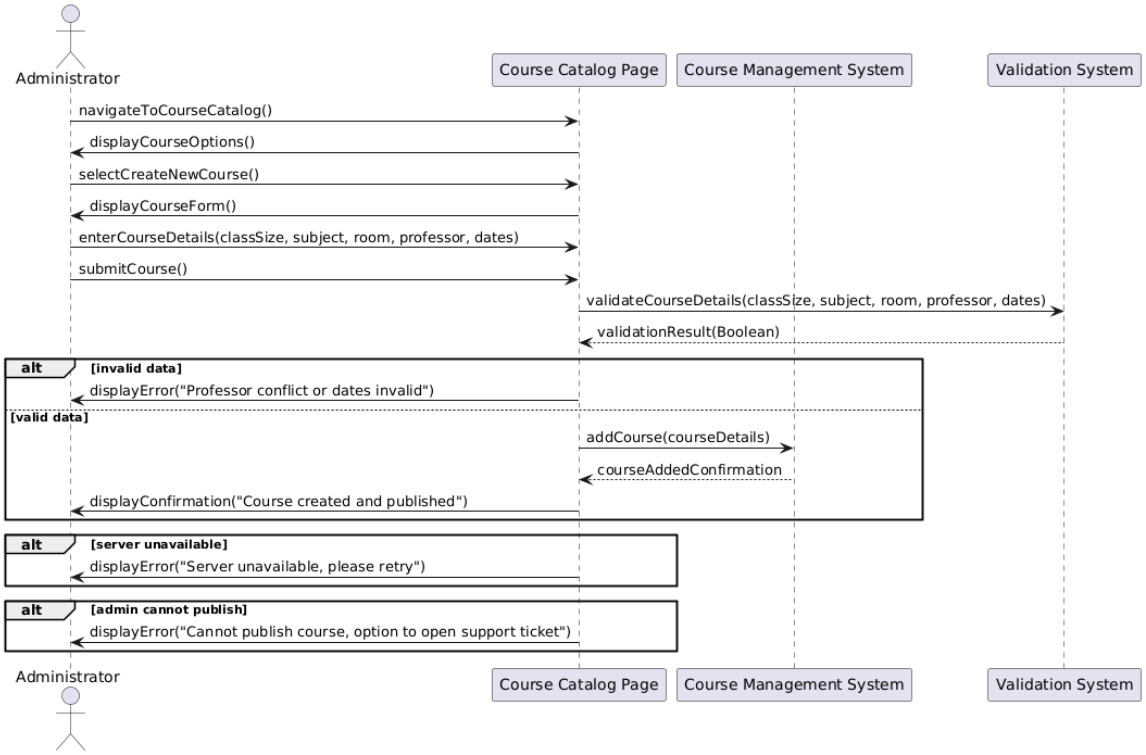
Use Case 8: Enrollment Reporting



Use Case 9: Update Changes Report



Use Case 10: Create courses



Use Case 11: Editing courses