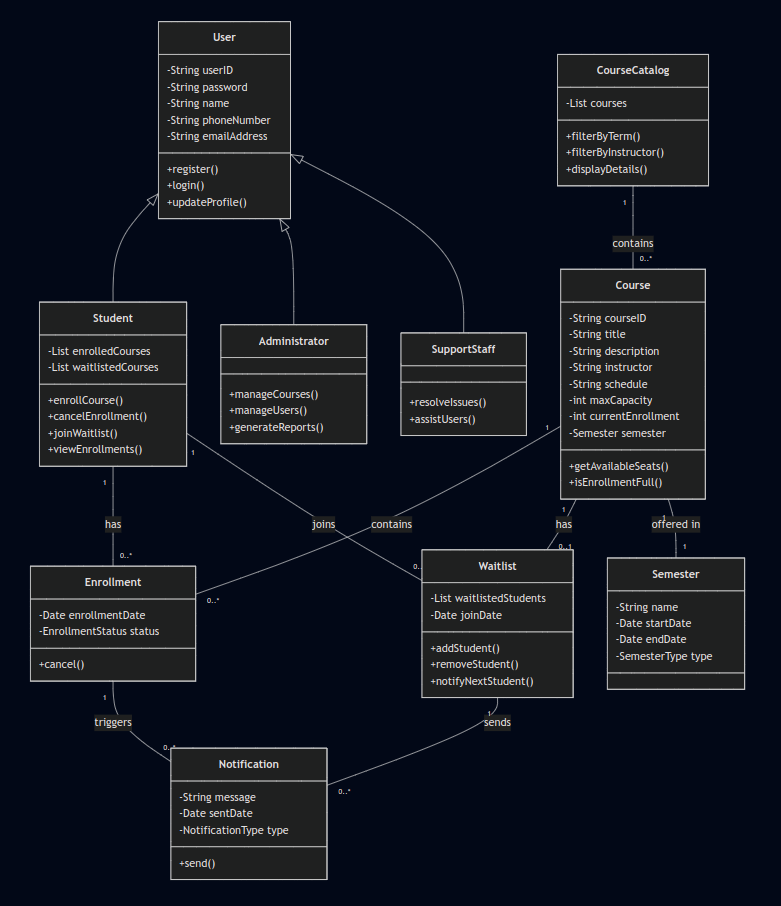
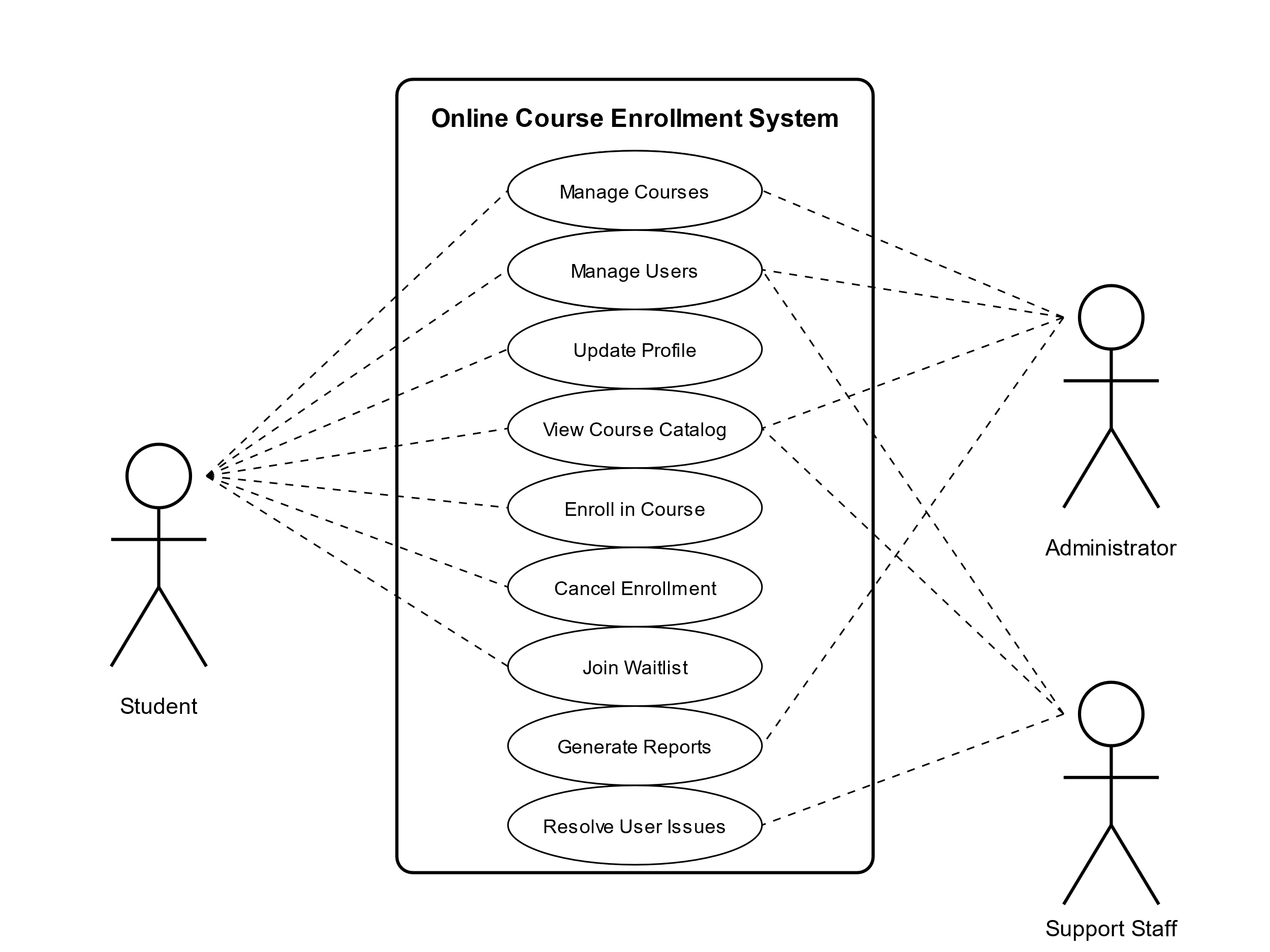
**Class diagrams (static structure)**



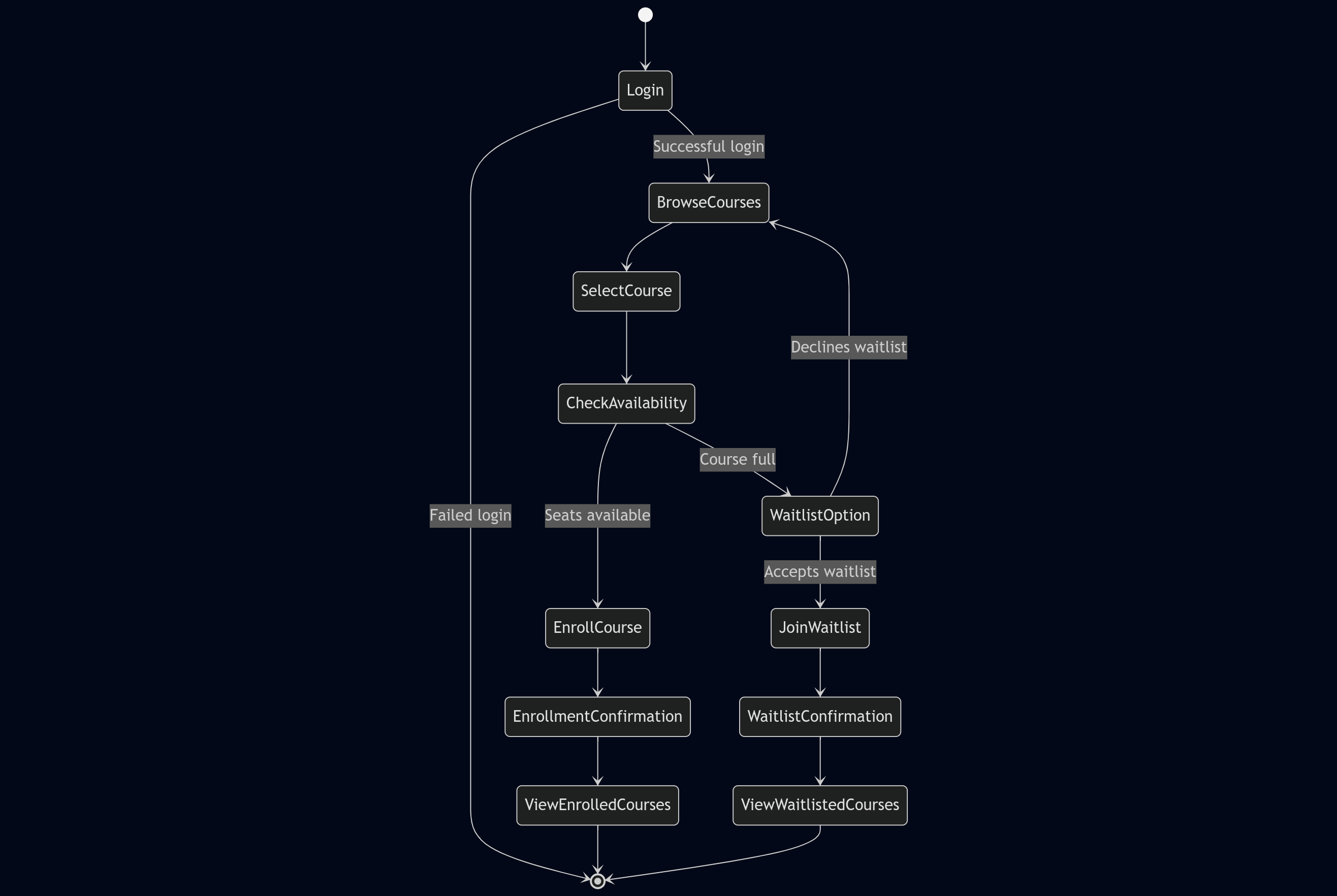
*Figure 1: Class diagram*

**Use case diagrams (user interactions)**

****

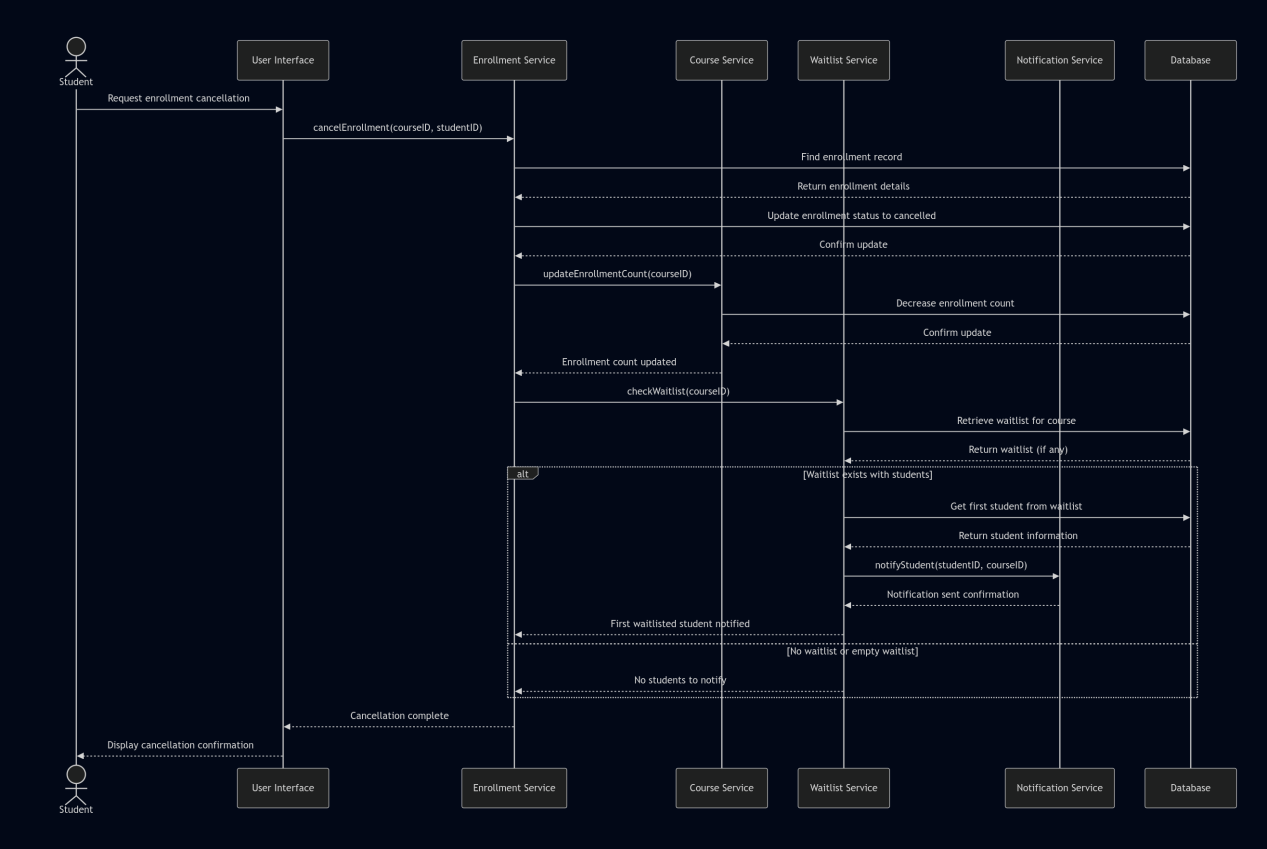
*Figure 2: Usecase diagram*

**Activity diagrams (workflow)**

****

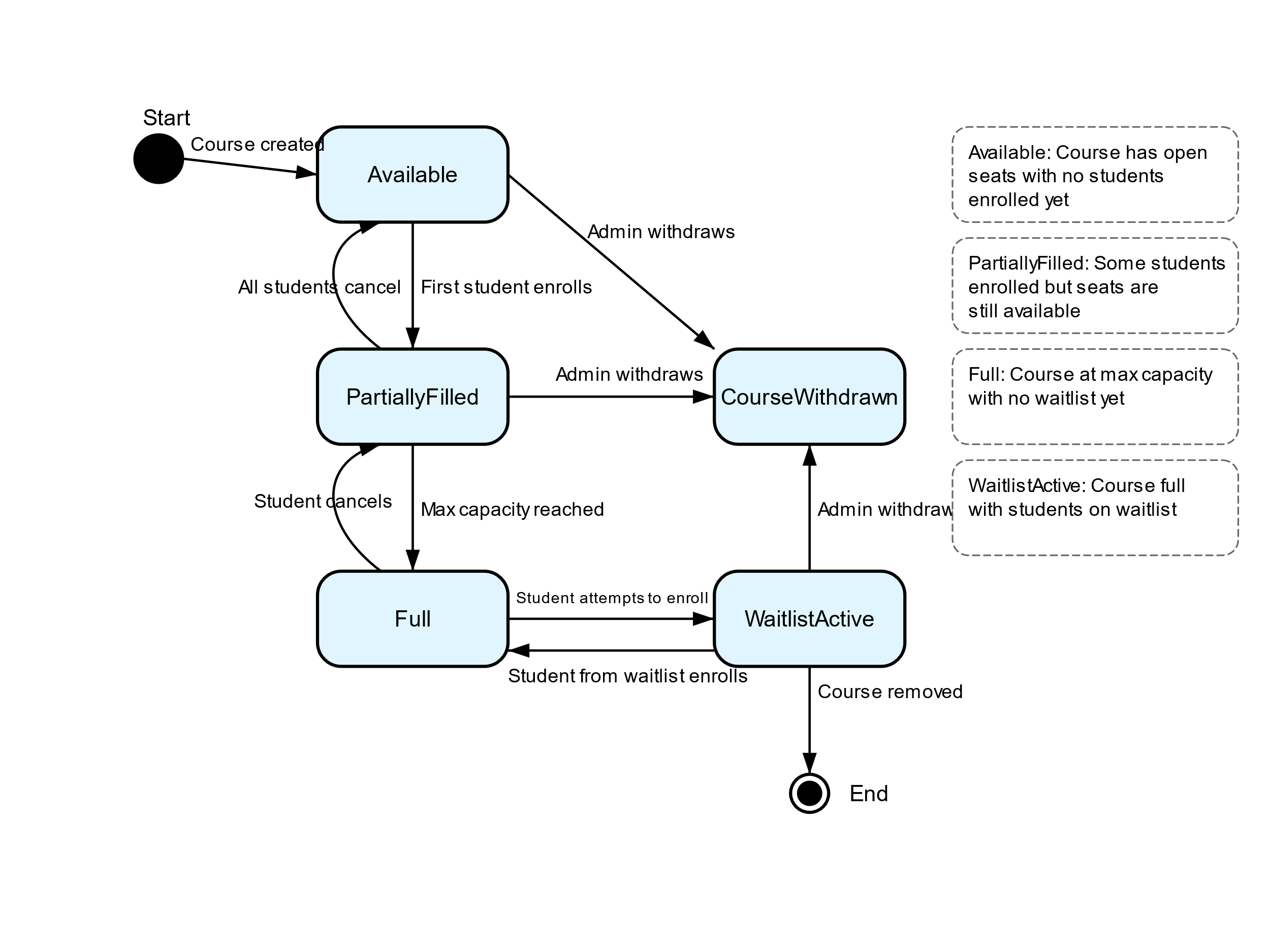
*Figure 3: Activity diagram*

**Sequence diagrams (dynamic behavior)**

****

*Figure 4: Sequence diagram*

**State diagram**

****

*Figure 5: State diagram*

UML Design Modeling for Online Course Enrollment System

Nkosinomusa Gwalla

CST499 Capstone for Computer Software

Professor Joseph Rangitsch

March 27, 2025

## Introduction

## Software design modeling serves as a critical bridge between requirement specifications and implementation, enabling developers to visualize system structure and behavior before coding begins. This paper presents a comprehensive set of Unified Modeling Language (UML) diagrams for an Online Course Enrollment System, translating the requirements from the Software Requirements Specification (SRS) document into visual representations. The system aims to streamline student registration, course enrollment, and waitlist management across academic periods. Through the creation of class diagrams, use case diagrams, sequence diagrams, activity diagrams, and state diagrams, both the static and dynamic aspects of the system are captured. Additionally, this paper examines the different levels of testing required to ensure the system's reliability, functionality, and user acceptance. By applying established design principles and testing methodologies, the Online Course Enrollment System can be developed with a clear architecture that fulfills stakeholder needs while maintaining robustness and scalabilit

## UML Models for the Online Course Enrollment System

The software design standardizes UML through its visual language to enable developers for communicating detailed system definitions and behaviors. The development of the Online Course Enrollment System required five distinct UML models that address its multiple system elements.

### Class Diagram

The designed system displays its static structure through the class diagram Figure 1 which shows the system entities alongside their attributes and methods together with all interconnecting relationships. The User class function as the main parent entity within the system to generate the three consecutive derivatives Student, Administrator, and SupportStaff. Different user types can access the system utilizing the polymorphic inheritance structure that enables them to carry out task-specific operations while behaving consistently throughout the system.

A single Student object throughout the system can connect to various Enrollment classes as well as Waitlist classes through many-to-many relationships that model the enrollment capabilities of students in multiple courses and the ability to waitlist for them. A course in the Course class establishes two many-to-one relationships by holding students as part of Enrollment and potential waiting students through Waitlist.

The Notification class serves as a bridge between both Enrollment and Waitlist classes which enables the system to generate alerts related to enrollment updates and waitlist successions. A complete class design contains proper attributes and operations that model all necessary entities for system functional requirements support.

### Use Case Diagram

The use case diagram (Figure 2) captures the interactions between the system and its external actors: Students, Administrators, and Support Staff. Each actor interacts with specific use cases that represent the functionality they can access within the system.

Students can perform actions such as registering accounts, logging in, updating profiles, viewing course catalogs, enrolling in courses, canceling enrollments, and joining waitlists. Administrators have management capabilities, including course management, user management, and report generation, while also being able to view the course catalog and log in. Support staff focus on resolving user issues and can access the system through login and view course information.

This diagram communicates the system's boundary and the various ways users interact with it, providing a high-level view of system functionality.

### Activity Diagram

The course enrollment workflow appears in Figure 3 through activity diagrams to demonstrate the order of student enrollment actions and the points of decision making before a student can select their courses. After users authenticate their login credentials the system progresses to display available courses which the user can choose from.

The student determines the next course action by checking seat availability which leads them to enrollment process when spots exist or waitlisting when courses reach full capacity. The diagram presents both main and alternative stages along with their triggered conditions which results in an extensive display of enrollment process pathways. The visual layout shows developers which sequence of operations must be implemented to create the core functionality.

### Sequence Diagram

A sequence diagram in Figure 4 demonstrates the chronological process of cancellation enrollment by showing how various system components interact (Figure 4). Students initiate cancellations that transmit via the User Interface to Enrollment Service before this service updates database enrollment records.

The diagram displays both actions of updating enrollment counts by the Course Service and actions from the Waitlist Service which identifies and informs the subsequent waitlisted student about available positions. The diagram displays the series of distributed messages sent between program elements thus developers can track the timing behavior alongside dependency relationships across this essential workflow enabling them to meet every necessary step.

### State Diagram

A course moves through its lifecycle states in the system according to what is shown in Figure 5 of the state diagram. The course starts in "Available" mode until students start enrollment which turns it into "PartiallyFilled" state then progresses to "Full" status when maximum enrollment reaches capacity and finally becomes "WaitlistActive" if students try enrolling when the course is full.

The diagram also shows how a course can transition back to previous states when students cancel enrollments or how it can move to a "CourseWithdrawn" state by administrative action. Each transition is labeled with the event that triggers it, providing clear documentation of the conditions under which a course's state changes. This model is particularly valuable for implementing the course management aspects of the system.

## Testing Levels for the Online Course Enrollment System

Testing the Online Course Enrollment System requires a complete testing plan to guarantee system excellence and user contentment and operational dependability. The system will undergo four testing levels at different phases for validating distinct system elements.

### Component Testing

According to Sommerville (2016) component testing examines solitary software components for proper functionality through its unit testing approach. The element testing of the Online Course Enrollment System examines specific classes and methods to validate correct functionality based on written documents.

Key components to be tested include:

1. User authentication methods: Testing the login() method of the User class to verify it correctly validates credentials and handles failed attempts.
2. Course availability checking: Testing the isEnrollmentFull() method of the Course class to ensure it accurately determines whether a course has reached its maximum capacity.
3. Waitlist management: Testing the addStudent() and notifyNextStudent() methods of the Waitlist class to verify they correctly manage the waitlist queue and notification process.

Component testing will primarily use white-box testing techniques, where testers have knowledge of the internal structure of the code. Automated testing frameworks appropriate to the implementation language will be employed to create and run test cases. This level of testing serves as the foundation for ensuring individual building blocks of the system work correctly before they are integrated.

### Integration Testing

Integration testing examines the interactions between integrated components or subsystems to detect interface defects (Myers et al., 2021). For the Online Course Enrollment System, integration testing will verify that components work together as expected when combined.

Key integration points to be tested include:

1. Enrollment process chain: Testing the interaction between the User Interface, Enrollment Service, Course Service, and Database components to ensure enrollment requests are processed correctly end-to-end.
2. Waitlist-to-enrollment workflow: Testing the sequence of actions that occur when a spot becomes available and a waitlisted student is notified and subsequently enrolled.
3. User authentication and authorization flow: Testing how the authentication mechanism integrates with the authorization rules to ensure users can only access features appropriate to their roles.

Two primary approaches will be used for integration testing:

Bottom-up testing: The testing process begins with lower-level components (known as Database access layer) before adding progressively higher-level components.

Top-down testing:System testers follow the method of top-down testing which starts with User Interfaces first while substituting lower-level components with simulation elements until all components merge.

Integration testing is crucial for identifying issues in the component interfaces and ensuring data flows correctly throughout the system.

### System Testing

System testing serves to verify that the complete integrated system fulfills its specified requirements as mentioned in Sommerville (2016). System testing of the Online Course Enrollment System will check every aspect of the complete application including its functionality and execution performance alongside security measures and user experience.

Key aspects to be covered in system testing include:

1. Functional testing: Verifying all use cases work correctly from end to end.
2. Performance testing: Ensuring the system can handle the specified load of 500 concurrent users during peak enrollment
3. Security testing: Validating that the system protects sensitive student information,
4. Usability testing: Assessing the user interface for ease of use and accessibility
5. Database testing: Verifying data integrity, transaction handling, and proper storage and retrieval of information.

The goal of system testing is to assess the functionality of the system through black-box analysis without examining its code base. System testing should exist at this level due to its importance for verifying the system's complete integration as well as compliance with all specified requirements.

### Acceptance Testing

A system qualifying process through acceptance testing ensures that all prescribed acceptance standards and delivery readiness to end users has been achieved (Myers et al., 2021). The acceptance testing process for the Online Course Enrollment System focuses exclusively on the system users including students and both administrative personnel and support personnel.

Two primary forms of acceptance testing will be conducted:

1. Alpha testing: Conducted in a controlled environment with selected users performing specific tasks while developers observe and note issues. This will include testing:
   * Student workflows for enrollment and waitlist management
   * Administrative workflows for course and user management
   * Support staff workflows for issue resolution
2. Beta testing: A limited public release to a broader group of users who will use the system in their own environments and report issues. This provides valuable feedback on how the system performs under real-world conditions.

Acceptance criteria will be directly derived from the SRS document, with particular attention to:

Functional completeness

Performance requirements

Usability requirements

Security requirements

Acceptance testing is the final validation before system deployment and serves as confirmation that the system meets stakeholder expectations and is ready for production use.

## Conclusion

UML design models created for the Online Course Enrollment System deliver complete illustrations of system static elements and runtime operations. The top-layer design foundation for the system rests on the class diagram and the use case diagram precisely defines user interactions. The designs use activity and sequence diagrams to show important system processes and the state diagram tracks a course's operational timeline in the system. The diagrams share a unified approach to transform specification requirements from the SRS document into a design framework for implementation.

The testing method consists of four testing levels which provide step-by-step validation for system components and integrated application as a whole. The execution of component testing verifies basic elements function properly and integration testing confirms components collaborate properly while system testing checks that all components match requirements and acceptance testing verifies system fulfillment of stakeholder expectations. The methodical testing process enables the team to detect and resolve problems at each step so they create an excellent working system that meets specification requirements.

This paper establishes design models together with testing approach which provides an effective base for building the Online Course Enrollment System to fulfill technical criteria along with user expectations during successful implementation.

## References

*Myers, G. J., Sandler, C., & Badgett, T. (2021).* The art of software testing *(3rd ed.). John Wiley & Sons.*

*Sommerville, I., Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (2019). Edition: Software Engineering. Instructor.*

*Bourque, P., Dupuis, R., Abran, A., Moore, J. W., & Tripp, L. (2002). The guide to the software engineering body of knowledge. IEEE software, 16(6), 35-44.*