Software Requirements

Specification

1. **User Authentication and Role-Based Access(Chen)**
    - Description: The system must authenticate users (students, faculty, administrators) and provide appropriate access based on their role. This ensures system security and proper functionality segregation.
    - Validation: All user actions are properly restricted based on role, and unauthorized access attempts are prevented.

2. **Course Registration Processing**
    - Description: Students must be able to register for courses, with the system handling prerequisites, schedule conflicts, and enrollment limits automatically.
    - Validation: Successful registration with proper error handling for various scenarios (full courses, conflicts, etc.).

3. **Data Persistence**
    - Description: All system data (course catalog, student registrations, waitlists) must be persistently stored in files and properly managed to prevent data loss.
    - Validation: Data integrity is maintained across system restarts and file operations.

4. **Waitlist Management(Chen)**
    - Description: The system must automatically manage waitlists for full courses, including adding students, updating positions, and notifying when spots become available.
    - Validation: Waitlists function correctly with proper position tracking and notifications.

5. **Report Generation(Andrew)**
    - Description: The system must generate various reports (enrollment, waitlists, class rosters) and export them in different formats (CSV, plain text).
    - Validation: All required report types can be generated accurately and exported in specified formats.

6. **Course Catalog Management**
    - Description: Administrators must be able to manage the course catalog,

including adding, modifying, and deactivating courses.
- Validation: Course catalog can be successfully managed with all changes properly persisted.

7. Search Functionality(Andrew)

- Description: Users must be able to search the course catalog using various criteria (course code, name, department).
- Validation: Search function returns accurate results based on different search criteria.

8. Faculty Course Management(Anthony)

- Description: Faculty must be able to view assigned courses, class rosters, and update course syllabi.
- Validation: Faculty can successfully perform all required course management tasks.

9. External System Interfaces

- Description: The system must interface with external university systems through file exchanges for billing, grades, and student information.
- Validation: All required data exchanges with external systems function correctly.

10. Error Handling and Logging(Anthony)

- Description: The system must properly handle and log all errors, providing user-friendly error messages and maintaining an error log for troubleshooting.
- Validation: All error scenarios are handled gracefully with proper user feedback and logging.

1. Course

- Responsibilities:
  - Maintains course information (code, name, description, credits)
  - Manages enrollment limits and current enrollment
  - Handles prerequisites checking
- Attributes:
  - courseID: String
  - courseName: String

        - description: String
        - credits: int
        - enrollmentLimit: int
        - currentEnrollment: int
        - prerequisites: List<Course>
        - waitlist: WaitList

2. Student

- Responsibilities:
    - Manages student information and enrollment
    - Handles course registration and schedules
- Attributes:
    - studentID: String
    - name: String
    - enrolledCourses: List<Course>
    - waitlistedCourses: List<Course>
    - academicRecord: AcademicRecord

3. UserManager

- Responsibilities:
    - Handles user authentication and authorization
    - Manages user sessions and roles
- Attributes:
    - users: Map<String, User>
    - activeUsers: Set<String>
    - userRoles: Map<String, UserRole>

4. RegistrationManager

- Responsibilities:
    - Processes course registration requests
    - Manages waitlists and enrollment updates
- Attributes:
    - courseRegistry: Map<String, Course>
    - registrationRules: List<RegistrationRule>
    - waitlistManager: WaitlistManager

5. FileDataManager

- Responsibilities:
    - Handles all file I/O operations
    - Manages data persistence and retrieval
- Attributes:
    - dataFiles: Map<String, File>
    - fileFormats: Map<String, DataFormat>
    - backupManager: BackupManager

Revision History

Date	Revision	Description	Author
09/16/2024	1.0	Initial Version	Chen Li
09/23/2024	1.0	Adding Specific Requirements, External and Internal Interface Requirements	Chen Li
09/25/2024	1.0	Adding Use Cases	Chen Li
9/26/2024	1.0	Worked on privacy/security and performance requirements	Andrew Tang
9/27/2024	1.0	Added use case	Andrew Tang
9/27/2024	1.0	Added use case 5.6 & 5.7	Connor McMillan
9/28/2024	1.0	Added class diagrams	Connor McMillan
9/28/2024	1.0	Worked on Environmental/Performance Requirements	Anthony Kungo
9/28/2024	1.0	Worked on Use Case Specification Document	Anthony Kungo
9/29/2024	1.0	Added UML Use Case Diagram	Anthony Kungo
9/29/2024	1.0	Add sequence diagrams	Andrew Tang
9/30/2024	1.0	Add sequence diagram	Andrew Tang
9/30/2024	1.0	Added sequence diagram 8.6 & 8.7	Connor McMillan
9/30/2024	1.0	Added use case 5.8-5.15	Chen Li
10/2/2024	1.0	Added sequence diagrams	Andrew Tang

Table of Contents

1. Purpose

This document outlines the software requirements for a University Class Enrollment System. The system aims to streamline the process of course registration, manage class schedules, and facilitate communication between students, faculty, and administrators.

1.1. Scope

The system will handle course catalog management, student registration, faculty course assignments, and basic reporting. It will integrate with existing university systems for student records and billing.

1.2. Definitions, Acronyms, Abbreviations

UI: User Interface

API: Application Programming Interface

DBMS: Database Management System

1.3. References

Use Case Specification Document – Step 2 in assignment description

UML Use Case Diagrams Document – Step 3 in assignment description

Class Diagrams – Step 5 in assignment description

Sequence Diagrams – Step 6 in assignment description

1.4. Overview

The University Class Enrollment System is a Java-based application designed to manage the course enrollment process for a university. It provides functionality for students to register for courses, faculty to manage their classes, and administrators to oversee the entire process. The system operates without web technologies or databases, instead utilizing file-based storage for all data persistence.

2. Overall Description

2.1. Product Perspective

The University Class Enrollment System will be a standalone Java application accessible to students, faculty, and administrators on their local machines. It will use file-based storage for all data.

2.2. Product Architecture

The system will be organized into 6 major modules: the Course catalog management module, the Student course registration and scheduling module, and the Faculty course assignment module, the Waitlist management module, the Report generation module and User authentication and authorization module

2.3. Product Functionality/Features

- Course catalog management
- Student course registration and scheduling
- Faculty course assignment
- Waitlist management
- Report generation
- User authentication and authorization

2.4. Constraints

Technology Constraints:

- The system must be developed entirely in Java.
- No web technologies or databases can be used.
- The application must run on JVM 8 or higher.

Data Storage Constraints:

- All data must be stored in local files.
- File formats must be designed to ensure data integrity and easy parsing.

Network Constraints:

- The system must operate within a local network environment.
- No internet connectivity should be required for core functionalities.

User Interface Constraints:

- The GUI must be developed using Java Swing or JavaFX.
- The interface must be accessible to users with basic computer skills.

2.5. Assumptions and Dependencies

2.5.1 Assumptions:

User Assumptions:

- All users have basic computer literacy and can operate a Java application.
- Users have appropriate permissions to run Java applications on their machines.

System Assumptions:

- The local network is reliable and has sufficient bandwidth to handle file operations.
- All user computers meet the minimum hardware requirements to run the Java

application smoothly.

Data Assumptions:

- The initial course catalog and user data will be provided in a compatible format.
- Regular backups of data files will be maintained outside the system.

Operational Assumptions:

- The university has a process in place to distribute and update the application as needed.
- There is a designated system administrator to manage file permissions and backups.

2.5.2 Dependencies

Java Runtime Environment:

- The system is dependent on the availability of JRE 8 or higher on all user machines.

File System:

- The application depends on read and write access to specific directories for data storage.

Network File Sharing:

- Proper configuration of network file sharing is required for multi-user functionality.

Printing Services:

- Integration with local or network printers is necessary for report generation.

External Tools:

- The system may depend on external Java libraries for specific functionalities (e.g., PDF generation for reports).

University Policies:

- The system's operations are dependent on adherence to university enrollment policies and academic calendar.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Users should be allowed to log in using their issued id and pin, both of

which are alphanumeric strings between 6 and 20 characters in length.

3.1.1.2 The system should provide Java swing or Java FX based help pages on each screen that describe the purpose of each function within the system.

3.1.1.3 All modules should implement proper error handling and display user-friendly error messages.

3.1.1.4 Each module should have a consistent look and feel using Java Swing or JavaFX components.

3.1.1.5 All data modifications should be immediately saved to the appropriate files to ensure data integrity.

3.1.2. Course Catalog Management Module Requirements:

3.1.2.1 The system shall allow administrators to add new courses to the catalog.

3.1.2.2 Administrators should be able to edit existing course information, including course code, name, description, credits, and prerequisites.

3.1.2.3 The system shall provide a search function to find courses based on various criteria (e.g., course code, name, department).

3.1.2.4 Administrators should be able to set the maximum enrollment capacity for each course.

3.1.2.5 The system shall allow administrators to mark courses as active or inactive for the current semester.

3.1.3. Student Course Registration and Scheduling Module Requirements:

3.1.3.1 Students should be able to view the course catalog and filter courses based on various criteria.

3.1.3.2 The system shall allow students to add courses to their schedule, provided they meet the prerequisites and there are available slots.

3.1.3.3 Students should be able to drop courses from their schedule within the allowed add/drop period.

3.1.3.4 The system shall prevent schedule conflicts when students attempt to register for courses.

3.1.3.5 Students should be able to view their current schedule, including course times and locations.

3.1.4. Faculty Course Assignment Module Requirements:

3.1.4.1 Administrators should be able to assign faculty members to specific courses.

3.1.4.2 Faculty members should be able to view the courses they are assigned to teach.

3.1.4.3 The system shall allow faculty to access and download class rosters for their assigned courses.

3.1.4.4 Faculty should be able to set or modify course-specific details such as syllabus and course materials.

3.1.4.5 The system shall prevent the assignment of faculty to courses with conflicting schedules.

3.1.5. Waitlist Management Module Requirements:

3.1.5.1 The system shall automatically add students to a waitlist when they attempt to register for a full course.

3.1.5.2 Students should be able to view their position on waitlists for courses they've attempted to join.

3.1.5.3 The system shall automatically notify the next student on the waitlist when a spot becomes available in a course.

3.1.5.4 Administrators should be able to view and manage waitlists for all courses.

3.1.5.5 The system shall allow administrators to manually add or remove students from waitlists.

3.1.6. Report Generation Module Requirements:

3.1.6.1 The system shall generate enrollment reports showing the number of students registered for each course.

3.1.6.2 Administrators should be able to generate reports on course waitlists, including lengths and movement.

3.1.6.3 The system shall provide functionality to export reports in various formats (e.g., CSV, plain text).

3.1.6.4 Faculty should be able to generate and print class rosters for their assigned courses.

3.1.6.5 The system shall allow administrators to generate and view historical enrollment data and trends.

3.1.7. User Authentication and Authorization Module Requirements:

3.1.7.1 The system shall securely store user credentials, with passwords being hashed and salted.

3.1.7.2 Users should be able to reset their PIN through a secure process that involves verification of their identity.

3.1.7.3 The system shall implement role-based access control, restricting access to functionalities based on user type (student, faculty, administrator).

3.1.7.4 The system shall log all login attempts, both successful and failed, for security auditing purposes.

3.1.7.5 After a configurable number of failed login attempts, the system shall temporarily lock the user account and notify the administrator.

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, term 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 must provide an interface to the University's student information system to retrieve and update student demographic information. The interface will be a comma-separated text file containing the following fields: student id, name, address, phone number, email. This file will be exported weekly and will contain all current student records.

3.2.3 The system must provide an interface to the University's course catalog system to retrieve updated course information. The interface will be a comma-separated text file containing the following fields: course id, course name, department, credits, description. This file will be imported at the beginning of each term and will contain all active courses for the upcoming term.

3.2.4 The system must provide an interface to the University's faculty management system to retrieve current faculty information. The interface will be a comma-separated text file containing the following fields: faculty id, name, department, office location, contact information. This file will be imported monthly and will contain all current faculty records.

3.3. Internal Interface Requirements

3.3.1 The system must process a data-feed from the grading system such that student grades are stored along with the historical student course enrollments. Data feed will be in the form of a comma-separated interface file that is exported from the grading system nightly.

3.3.2 The system must process a data-feed from the University billing system that contains new student records. The feed will be in the form of a comma-separated text file and will be exported from the billing system nightly with new student records. The fields included in the file are student name, student id, and student pin number.

3.3.3 The system must maintain an internal interface between the Course Catalog Management module and the Student Course Registration module to ensure that students can only register for courses that are currently active and have available slots.

3.3.4 The system must maintain an internal interface between the Waitlist Management module and the Student Course Registration module to automatically enroll students from the waitlist when spots become available in a course.

3.3.5 The system must maintain an internal interface between the User Authentication and Authorization module and all other modules to ensure that users can only access functionalities appropriate to their role (student, faculty, or administrator).

3.3.6 The system must process an internal data-feed from the Student Course Registration module to the Report Generation module to provide up-to-date enrollment statistics for generating reports.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 The system must encrypt user login info when user logs in.
4.1.2 The user must be securely logged out when they want to.
4.1.3 The system should automatically logout inactive users when noticed

that they have been inactive for too long.
4.1.4 Upon at least 5 failed login attempts, the account will be locked.
4.1.5 Only the bare minimum of user data should be collected when needed.

4.2. Environmental Requirements

4.2.1 College Enrollment System cannot require that any software other than a web browser be installed on user computers.
4.2.2 System must make use of Universities' existing csv file implementation for its databases for schedules, classes, and courses

4.3. Performance Requirements

4.3.1 It must take no longer than 5 seconds to do actions such as enrolling or joining waitlists.
4.3.2 It should be able to handle multiple users at the same time.
4.3.3 It should be able to support multiple universities.
4.3.4 It should be able to support multiple devices.

5.Use Case Specification Document

5.1 System Login

Use Case ID: UC-SL-001
Use Case Name: Logging in to the system
Relevant Requirements: 3.1.3.1, 3.1.3.2
Primary Actor: Student, Administrator, Faculty

Pre-conditions:

- User has login information

Post-conditions:

- User is logged in

Basic Flow or Main Scenario:

1. User enters username, password, and school code
    - Students will then have access to their class schedule, and will be able to enroll/drop classes.
    - Administrators will be able to access courses, classes, student information, and student enrollment.
    - Faculty will be able to update their course and class information, and submit grades.

Extensions or Alternate Flows:

- None

Exceptions:

- Username doesn’t exist
- Password doesn’t match given username
- School code is wrong

Related Use Cases:

- Student Course Registration
- Faculty Grade Submission
- Administrator Course Management
- Student Course Drop
- Administrator Creates Schedule
- User Management

5.2 Student Course Registration

Use Case ID: UC-SCR-001
Use Case Name: Register for a Course
Relevant Requirements: 3.1.3.1, 3.1.3.2, 3.1.3.4
Primary Actor: Student

Pre-conditions:

- Student is logged into the system
- Student has met all prerequisites for the desired course
- The course has available slots

Post-conditions:

- Student is enrolled in the course
- Course enrollment count is updated
- Student's schedule is updated

Basic Flow or Main Scenario:

2. Student navigates to the course registration page
3. Student searches for the desired course
4. System displays course details and availability
5. Student selects the course and clicks "Register"
6. System checks for schedule conflicts and prerequisites
7. System enrolls the student in the course
8. System displays confirmation message
9. System updates student's schedule and course enrollment count

Extensions or Alternate Flows:

- 4a. Course is full:
    1. System adds student to the course waitlist
    2. System displays waitlist confirmation
- 5a. Schedule conflict detected:
    1. System displays conflict warning
    2. Student can choose to cancel registration or proceed with conflict

Exceptions:

- Student does not meet course prerequisites
- System experiences a file I/O error during enrollment update

Related Use Cases:

- View Course Catalog
- Student Course Drop
- System Login

5.3 Faculty Grade Submission

Use Case ID: UC-FGS-001
Use Case Name: Submit Student Grades
Relevant Requirements: 3.1.4.2, 3.1.4.3
Primary Actor: Faculty

Pre-conditions:

- Faculty is logged into the system
- Faculty is assigned to the course
- The grading period is open

Post-conditions:

- Student grades are recorded in the system
- Grade submission is marked as complete for the course

Basic Flow or Main Scenario:

1. Faculty navigates to the grade submission page
2. System displays list of assigned courses
3. Faculty selects a course
4. System displays the class roster with grade input fields
5. Faculty enters grades for each student
6. Faculty submits the completed grade sheet
7. System validates the entered grades
8. System saves the grades and marks submission as complete
9. System displays confirmation message

Extensions or Alternate Flows:

- 6a. Faculty saves grades as draft:
    1. System saves current progress without finalizing
    2. Faculty can return later to complete submission
- 7a. Invalid grade detected:
    1. System highlights invalid entries
    2. Faculty corrects errors and resubmits

Exceptions:

- System experiences a file I/O error during grade saving

- Grading period closes before submission is complete

Related Use Cases:

- View Class Roster
- Update Course Syllabus
- System Login

5.4 Administrator Course Management

Use Case ID: UC-ACM-001
Use Case Name: Create New Course
Relevant Requirements: 3.1.2.1, 3.1.2.2, 3.1.2.4
Primary Actor: Administrator

Pre-conditions:

- Administrator is logged into the system
- The course does not already exist in the catalog

Post-conditions:

- New course is added to the course catalog
- Course is available for student registration (if set as active)

Basic Flow or Main Scenario:

1. Administrator navigates to the course management page
2. Administrator selects "Create New Course" option
3. System displays course creation form
4. Administrator enters course details (code, name, description, credits, prerequisites)
5. Administrator sets maximum enrollment capacity
6. Administrator sets course status (active/inactive)
7. Administrator submits the new course information
8. System validates the entered information
9. System adds the new course to the catalog
10. System displays confirmation message

Extensions or Alternate Flows:

- 8a. Duplicate course code detected:
    1. System displays warning message
    2. Administrator can modify code or cancel creation
- 6a. Administrator sets course as inactive:
    1. Course is added to catalog but not available for registration

Exceptions:

- System experiences a file I/O error during course creation
- Invalid data format detected during validation

Related Use Cases:

- Edit Existing Course

- Set Course Prerequisites
- System Login

5.5 Student Course Drop

Use Case ID: UC-SCD-001
Use Case Name: Student Drops Course
Relevant Requirements: 3.1.1.5, 3.1.3.3, 3.1.4.2
Primary Actor: Student

Pre Conditions:

- Student is logged into system
- Student is in the course

Post Condition:

- Student is dropped from course
- Related Faculty is notified of student dropped
- System updates the number of students in course

Basic Flow or Main Scenario:

1. Student navigates to drop course page
2. Student chooses the course to drops and submits
3. Student is dropped from said course
4. Related faculty is notified of the student dropping the course
5. System updates to account for the dropped student
6. Display confirmation message

Extension or Alternate Flows:

- Detected that the student will go below full-time
    1. Warning message will appear on screen
    2. Will ask for another confirmation to drop course

Exceptions:

- File I/O error when updating file for course and student

Related Use Case:

- Student Course Registration
- Student Waitlist
- System Login

5.6 Administrator Creates Schedule

Use Case ID: UC-ACS-001
Use Case Name: Administrator creates a schedule
Relevant Requirements: 3.1.3.4, 3.1.3.5

Primary Actor: Administrator

Pre Conditions:

- Administrator is logged into the system

Post Condition:

- A schedule is created for one particular course

Basic Flow or Main Scenario:

1. Administrator logs into the system
2. Administrator clicks on ‘Create a schedule’ button
3. Administrator selects a course to apply the schedule to
4. Administrator selects the days of the week and time slots
5. Administrator clicks ‘Finish’
6. System adds the schedule for the course into the database

Extension or Alternate Flows:

- Detected that there exists a schedule for that particular course
    1. Warning message will appear on screen
    2. Will suggest to modify the existing schedule

Exceptions:

- File I/O error when updating file for schedule

Related Use Case:

- Student Course Registration
- Student Waitlist
- System Login

5.7 User Management

Use Case ID: UC-UM-001
Use Case Name: Administrator creates, deletes, or modifies a user
Relevant Requirements: 3.1.1.1
Primary Actor: Administrator

Pre Conditions:

- Administrator is logged into the system

Post Condition:

- A user is created, deleted, or modified

Basic Flow or Main Scenario:

1. Administrator logs into the system
2. To create a user:
    a. Administrator clicks on ‘Add user’ button

    b. Administrator fills out a basic form containing the following fields
        i. Username
        ii. Password
        iii. Institution Code
        iv. Drop down: Student or Administrator
    c. Administrator clicks ‘Finish’
    d. System adds the user into the database
3. To edit a user:
    a. Administrator clicks on ‘Edit user’ button
    b. Administrator can modify the following fields
        i. Username
        ii. Password
        iii. Institution Code
        iv. Drop down: Student or Administrator
    c. Administrator clicks ‘Finish’
    d. System reinserts the user into the database
4. To delete a user:
    a. Administrator clicks on ‘Remove user’ button
    b. Administrator selects the username(s) to delete
    c. System removes the user(s) from the database

Extension or Alternate Flows:

- Detected that there are empty fields for adding or modifying users
    - Warning message will appear on screen
- Detected that there are no users selected to delete
    - Warning message will appear on screen
- Detected that the only administrator account is being deleted
    - Error message will appear on the screen. You cannot delete the only registered administrator account.

Exceptions:

- File I/O error when updating a user

Related Use Case:

- System Login

5.8 View Class Roster

Use Case ID: UC-VCR-001 Use Case Name: View Class Roster Relevant Requirements: 3.1.4.3 Primary Actor: Faculty

Pre-conditions:

- Faculty is logged into the system
- Faculty is assigned to at least one course

Post-conditions:

- Class roster is displayed for the selected course

Basic Flow or Main Scenario:

1. Faculty navigates to the "My Courses" section
2. System displays a list of courses assigned to the faculty
3. Faculty selects a specific course
4. System retrieves the class roster from the course data file
5. System displays the roster with student names, IDs, and enrollment status

Extensions or Alternate Flows:

- 2a. No courses assigned:
    1. System displays message indicating no courses are currently assigned
- 5a. Faculty requests to export roster:
    1. System provides option to export as CSV or plain text file
    2. Faculty selects format and initiates export
    3. System generates and saves the exported file

Exceptions:

- System cannot access course data file
- Course assignment data is corrupted or incomplete

Related Use Cases:

- Submit Student Grades
- Generate Course Report

5.9 Update Course Syllabus

Use Case ID: UC-UCS-001 Use Case Name: Update Course Syllabus Relevant Requirements: 3.1.4.4 Primary Actor: Faculty

Pre-conditions:

- Faculty is logged into the system
- Faculty is assigned to the course
- Course exists in the system

Post-conditions:

- Course syllabus is updated in the system
- Update timestamp is recorded

Basic Flow or Main Scenario:

1. Faculty navigates to course management section
2. System displays list of assigned courses
3. Faculty selects a course to update
4. System loads current syllabus content
5. Faculty modifies syllabus content
6. Faculty submits updated syllabus
7. System validates the content
8. System saves the updated syllabus
9. System confirms successful update

Extensions or Alternate Flows:

- 5a. Faculty saves draft:
    1. System saves current progress without publishing
    2. Faculty can return later to complete updates
- 7a. Content exceeds size limit:
    1. System displays warning
    2. Faculty must reduce content before submitting

Exceptions:

- File system error during save operation
- Concurrent update attempt by another user

Related Use Cases:

- View Class Roster
- Create New Course

5.10 View Course Catalog

Use Case ID: UC-VCC-001 Use Case Name: View Course Catalog Relevant Requirements: 3.1.3.1 Primary Actor: Student

Pre-conditions:

- User is logged into the system
- Course catalog data is available

Post-conditions:

- Course catalog is displayed to the user

Basic Flow or Main Scenario:

1. User navigates to the course catalog section
2. System retrieves course catalog data
3. System displays list of all available courses
4. User can scroll through the catalog
5. System provides filtering and sorting options

Extensions or Alternate Flows:

- 3a. User applies filters:
    1. User selects filtering criteria (department, course level, etc.)
    2. System updates display with filtered results
- 3b. User uses search function:
    1. User enters search terms
    2. System displays matching courses

Exceptions:

- Course catalog data is unavailable

- System encounters error while filtering/sorting

Related Use Cases:

- Register Course
- Search Courses

5.11 Generate Enrollment Report

Use Case ID: UC-GER-001 Use Case Name: Generate Enrollment Report Relevant Requirements: 3.1.6.1, 3.1.6.2, 3.1.6.3, 3.1.6.5 Primary Actor: Administrator

Pre-conditions:

- Administrator is logged into the system
- Enrollment data is available in the system

Post-conditions:

- Requested report is generated and available for viewing/export

Basic Flow or Main Scenario:

1. Administrator navigates to the reporting section
2. System displays available report types
3. Administrator selects "Enrollment Report"
4. Administrator configures report parameters (term, department, etc.)
5. Administrator selects report format (CSV, plain text)
6. System generates the report
7. System displays report preview
8. Administrator can export or print the report

Extensions or Alternate Flows:

- 4a. Administrator selects historical data:
    1. System retrieves archived enrollment data
    2. Report includes historical trends
- 6a. Report generation takes longer than expected:
    1. System displays progress indicator
    2. Administrator can cancel report generation

Exceptions:

- Required data files are corrupted or unavailable
- System lacks sufficient memory to generate large reports

Related Use Cases:

- View Class Roster
- Generate Waitlist Report

5.12 Search Courses

Use Case ID: UC-SC-001 Use Case Name: Search Courses Relevant Requirements: 3.1.2.3 Primary Actor: Student/Faculty/Administrator

Pre-conditions:

- User is logged into the system
- Course catalog is available

Post-conditions:

- Search results are displayed to the user

Basic Flow or Main Scenario:

1. User navigates to course search function
2. System displays search interface with various criteria options
3. User enters search criteria (course code, name, department)
4. User initiates search
5. System queries course catalog data
6. System displays matching courses
7. User can select a course for more details

Extensions or Alternate Flows:

- 6a. No courses match criteria:
    1. System displays "No results found" message
    2. Suggests broadening search criteria
- 6b. Too many results:
    1. System suggests refining search criteria
    2. Provides option to view all results

Exceptions:

- Search function fails due to data access error
- Invalid search criteria format

Related Use Cases:

- View Course Catalog
- Register Course

5.13 View Waitlist

Use Case ID: UC-VW-001 Use Case Name: Administrator View Waitlist Relevant Requirements: 3.1.5.4 Primary Actor: Administrator

Pre-conditions:

- Administrator is logged into the system
- At least one course has an active waitlist

Post-conditions:

- Waitlist information is displayed to the administrator

Basic Flow or Main Scenario:

1. Administrator navigates to the waitlist management section
2. System displays a list of courses with active waitlists
3. Administrator selects a specific course
4. System retrieves the waitlist data for the selected course
5. System displays detailed waitlist information including:
    - Student names and IDs
    - Timestamp of waitlist entry
    - Current position in waitlist

Extensions or Alternate Flows:

- 2a. No active waitlists:
    1. System displays message indicating no current waitlists
- 3a. Administrator applies filters:
    1. Administrator selects filtering criteria (e.g., department, course level)
    2. System updates display with filtered results

Exceptions:

- System cannot access waitlist data file
- Waitlist data is corrupted or incomplete

Related Use Cases:

- Generate Waitlist Report
- Modify Course Waitlist

5.14 Generate Waitlist Report

Use Case ID: UC-GWR-001 Use Case Name: Generate Waitlist Report Relevant Requirements: 3.1.6.2, 3.1.6.3 Primary Actor: Administrator

Pre-conditions:

- Administrator is logged into the system
- Waitlist data is available in the system

Post-conditions:

- Waitlist report is generated and available for export

Basic Flow or Main Scenario:

1. Administrator navigates to the reporting section
2. System displays available report types
3. Administrator selects "Waitlist Report"
4. Administrator configures report parameters:
    - Time period
    - Specific courses or departments
    - Include historical waitlist movement

5. Administrator selects output format (CSV, plain text)
6. System generates the report
7. System displays report preview
8. Administrator exports or prints the report

Extensions or Alternate Flows:

- 6a. Large report generation:
    1. System displays progress bar
    2. Administrator can cancel report generation
- 7a. Administrator requests data visualization:
    1. System generates graphs showing waitlist trends
    2. Graphs are included in the exported report

Exceptions:

- Report generation fails due to data access error
- Insufficient system resources for large reports

Related Use Cases:

- View Waitlist
- Generate Enrollment Report

5.15 Modify Course Waitlist

Use Case ID: UC-MCW-001 Use Case Name: Modify Course Waitlist Relevant Requirements: 3.1.5.5 Primary Actor: Administrator

Pre-conditions:

- Administrator is logged into the system
- Course waitlist exists
- Student records are accessible

Post-conditions:

- Course waitlist is updated
- Affected students are notified of changes

Basic Flow or Main Scenario:

1. Administrator navigates to waitlist management section
2. System displays list of courses with waitlists
3. Administrator selects a course
4. System displays current waitlist for the course
5. Administrator chooses to add or remove a student
6. If adding: a. Administrator searches for student by ID or name b. System verifies student eligibility c. Administrator selects position for new entry
7. If removing: a. Administrator selects student from waitlist b. Administrator confirms removal
8. System updates waitlist
9. System generates notifications for affected students

Extensions or Alternate Flows:

- 6b. Student already on waitlist:
    1. System displays warning message
    2. Administrator can cancel or move student
- 7a. Administrator moves student position:
    1. Administrator selects new position
    2. System reorders waitlist accordingly

Exceptions:

- Student record not found
- System fails to update waitlist file

Related Use Cases:

- View Waitlist
- Generate Waitlist Report

5.16 View Personal Waitlist Position

Use Case ID: UC-VWP-001 Use Case Name: View Personal Waitlist Position Relevant Requirements: 3.1.5.2 Primary Actor: Student

Pre-conditions:

- Student is logged into the system
- Student is on at least one course waitlist

Post-conditions:

- Student views their current waitlist positions

Basic Flow or Main Scenario:

1. Student navigates to "My Waitlists" section
2. System retrieves student's waitlist data
3. System displays list of courses where student is waitlisted:
    - Course name and code
    - Current position on waitlist
    - Estimated time/chance of enrollment
    - Option to remove self from waitlist
4. Student can select a specific course for more details
5. System displays detailed waitlist information for selected course

Extensions or Alternate Flows:

- 2a. No active waitlists:
    1. System displays message indicating student is not on any waitlists
    2. Provides link to course registration
- 4a. Student removes self from waitlist:
    1. System prompts for confirmation
    2. Upon confirmation, removes student and updates waitlist

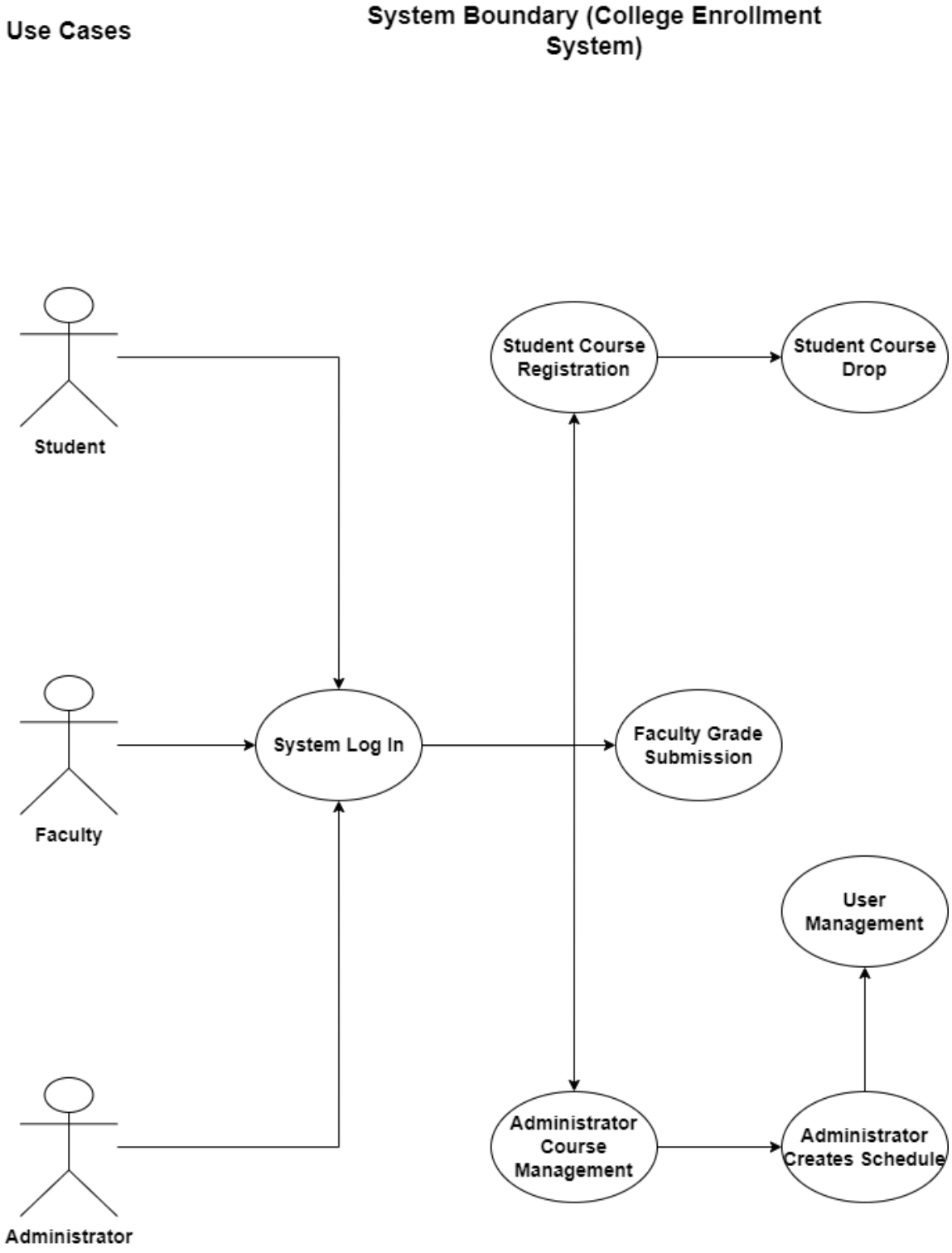
Exceptions:

- System cannot access waitlist data
- Waitlist position calculation error

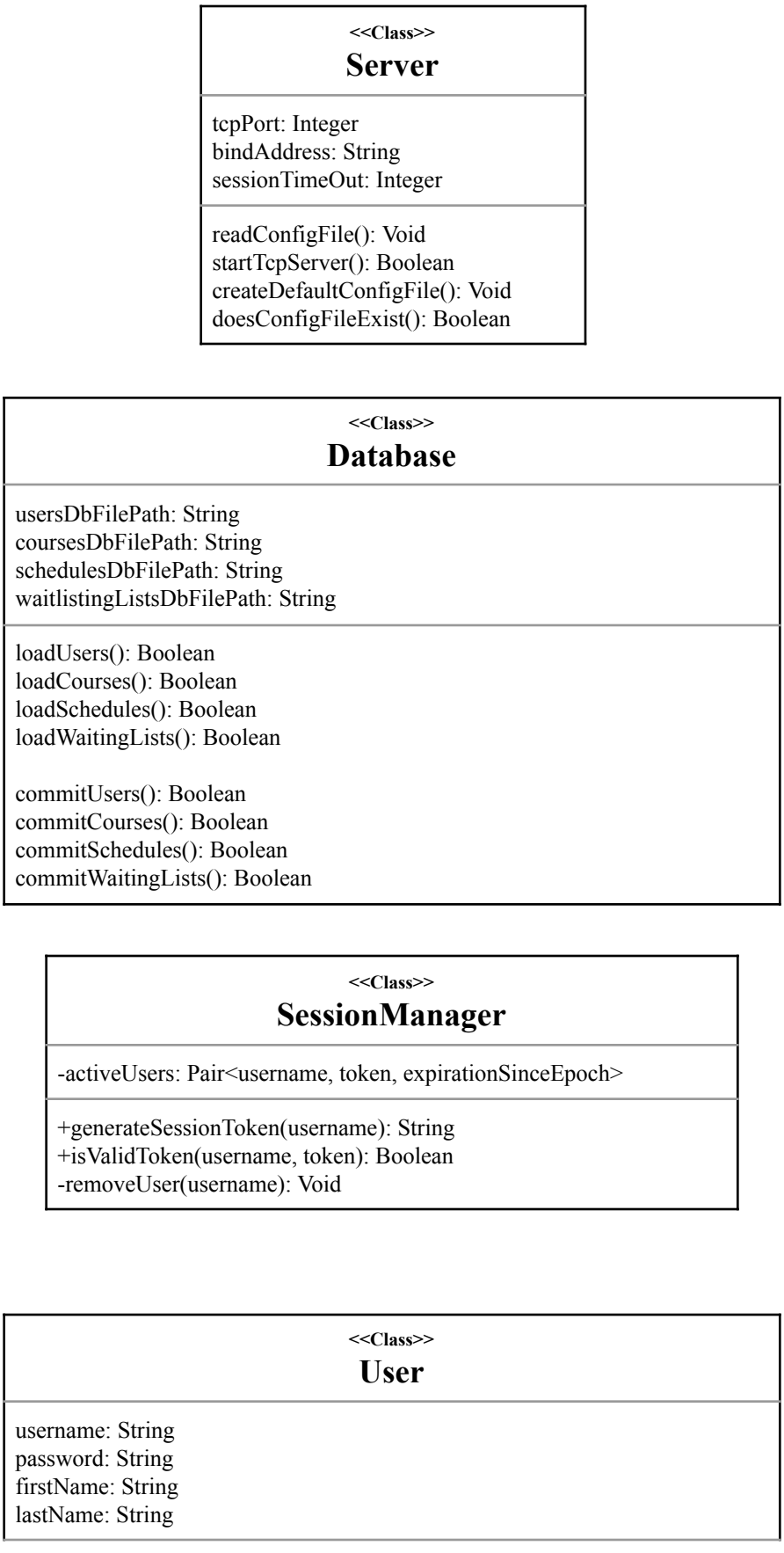
Related Use Cases:

- Register for a Course
- View Course Catalog

6.UML Use Case Diagrams(Chen,Anthony, Connor, Andrew)



7.Class Diagrams(Connor)


<<Class>>
Server
tcpPort: Integer
bindAddress: String
sessionTimeOut: Integer
readConfigFile(): Void
startTcpServer(): Boolean
createDefaultConfigFile(): Void
doesConfigFileExist(): Boolean
<<Class>>
Database
usersDbFilePath: String
coursesDbFilePath: String
schedulesDbFilePath: String
waitlistingListsDbFilePath: String
loadUsers(): Boolean
loadCourses(): Boolean
loadSchedules(): Boolean
loadWaitingLists(): Boolean
commitUsers(): Boolean
commitCourses(): Boolean
commitSchedules(): Boolean
commitWaitingLists(): Boolean
<<Class>>
SessionManager
-activeUsers: Pair<username, token, expirationSinceEpoch>
+generateSessionToken(username): String
+isValidToken(username, token): Boolean
-removeUser(username): Void
<<Class>>
User
username: String
password: String
firstName: String
lastName: String

institutionID: String
accountType: AccountType

getUsername(): String
getPassword(): String
getFirstName(): String
getLastName(): String
getInstitutionId(): String
getAccountType(): AccountType

setUsername(username): Boolean
setPassword(password): Boolean
setFirstName(firstName): Boolean
setLastName(lastName): Boolean
setInstitutionId(id): Boolean
setAccountType(AccountType): Boolean

<<Class>>

Users

users: User[]

addUser(user: User): Boolean
doesUsernameExist(username): Boolean
getUsersByInstitution(id): User[]
isUsernameValid(username): Boolean
removeUser(username): Boolean
isValidLogin(username, password): Boolean

commitToDb(): Void

<<Class>>

Course

class: String
description: String
institutionID: String
notes: String
addConsent: ConsentType
level: LevelType
section: String
academicProgram: AcademicProgramType
units: Float
grading: GradingType
instructionMode: InstructionModeType
location: String
campus: String
room: String
size: Integer
prerequisites: String[] of Course.class

getClass(): String
getInstitutionID(): String
getDescription(): String
getNotes(): String
getAddConsent(): ConsentType
getLevel(): LevelType
getSection(): String
getAcademicProgram(): AcademicProgramType
getUnits(): Float
getGrading(): GradingType
getInstructionMode(): InstructionModeType
getLocation(): String
getCampus(): String
getRoom(): String
getSize(): Integer
getPrerequisites(): Strings[]

setClass(): Boolean
setInstitutionID(id): Boolean
setDescription(description): Boolean
setNotes(notes): Boolean
setAddConsent(ConsentType): Boolean
setLevel(LevelType): Boolean
setSection(section): Boolean
setAcademicProgram(AcademicProgramType): Boolean
setUnits(units: Float): Boolean
setGrading(GradingType): Boolean
setInstructionMode(InstructionModeType): Boolean
setLocation(location): Boolean
setCampus(campus): Boolean
setRoom(room): Boolean
setSize(size): Boolean
setWaitlistSize(size): Boolean
setPrerequisites(prerequisites: String[] of Course.class): Boolean

<<Class>>
Courses

courses: Course[]

addCourse(course: Course): Boolean
getCoursesByInstitution(id): Course[]
removeCourse(course: Course): Boolean

commitToDb(): Void

<<Class>>
Schedule

class: String
institutionID: String
section: String
instructor: String
days: Days[]
endDate: String
startDate: String
startTime: Time
endTime: Time

getClass(): String
getInstitutionID(): String
getSection(): String
getInstructor(): String
getDays(): Days[]
getEndDate(): String
getStartDate(): String
getStartTime(): Time
getEndTime(): Time

setClass(name): Boolean
setInstitutionID(id): Boolean
setSection(section): Boolean
setInstructor(name): Boolean
setDays(days: Days[]): Boolean
setEndDate(date): Boolean
setStartDate(date): Boolean
setStartTime(time: Time): Boolean
setEndTime(time: Time): Boolean

doesThisScheduleConflict(schedule: Schedule): Boolean

<<Class>>

Schedules

schedules: Schedule[]

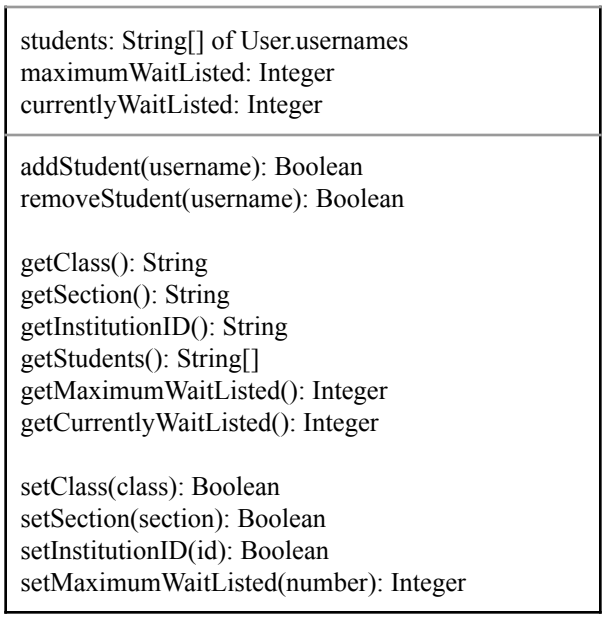
addSchedule(schedule: Schedule): Boolean
getSchedules(): Schedule[]
getSchedulesByInstitution(id): Schedule[]
removeSchedule(schedule: Schedule): Boolean

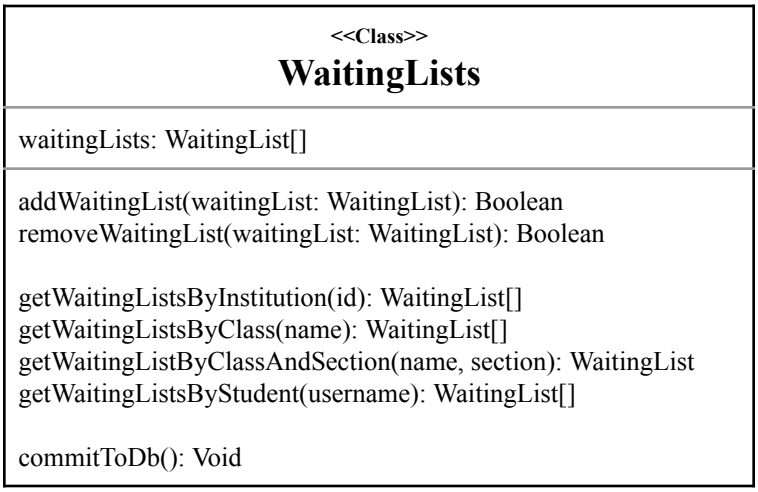
commitToDb(): Void

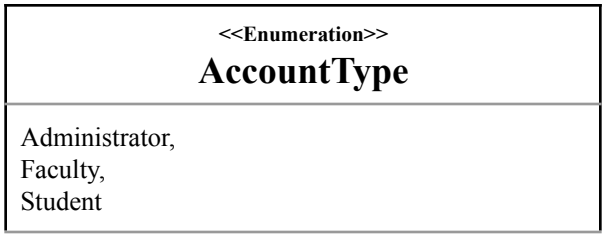
<<Class>>

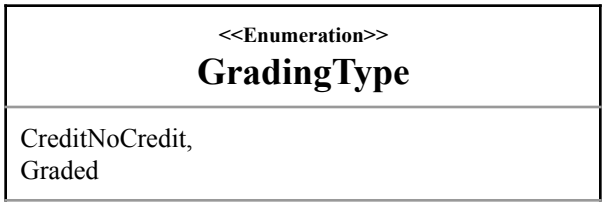
WaitingList

class: String
section: String
institutionID: String


students: String[] of User.usernames
maximumWaitListed: Integer
currentlyWaitListed: Integer
addStudent(username): Boolean
removeStudent(username): Boolean
getClass(): String
getSection(): String
getInstitutionID(): String
getStudents(): String[]
getMaximumWaitListed(): Integer
getCurrentlyWaitListed(): Integer
setClass(class): Boolean
setSection(section): Boolean
setInstitutionID(id): Boolean
setMaximumWaitListed(number): Integer


<<Class>>
WaitingLists
waitingLists: WaitingList[]
addWaitingList(waitingList: WaitingList): Boolean
removeWaitingList(waitingList: WaitingList): Boolean
getWaitingListsByInstitution(id): WaitingList[]
getWaitingListsByClass(name): WaitingList[]
getWaitingListByClassAndSection(name, section): WaitingList
getWaitingListsByStudent(username): WaitingList[]
commitToDb(): Void


<<Enumeration>>
AccountType
Administrator,
Faculty,
Student


<<Enumeration>>
GradingType
CreditNoCredit,
Graded

<<Dictionary>>

InstructionModeType

InPerson, “In Person”
Hybrid, “Hybrid”
Online, “Online Synchronous”
OnlineAS, “Online Asynchronous”

<<Dictionary>>

AcademicProgramType

UGM, “Undergraduate Matriculated”
NM, “Non-Matriculated”

<<Enumeration>>

Days

Monday,
Tuesday,
Wednesday,
Thursday,
Friday,
Saturday,
Sunday

<<Dictionary>>

ConsentType

Dept, “Requires Department Consent”
None, “No Consent Required”

<<Dictionary>>

LevelType

Lower, “Lower Division”
Upper, “Upper Division”

<<Dictionary>>

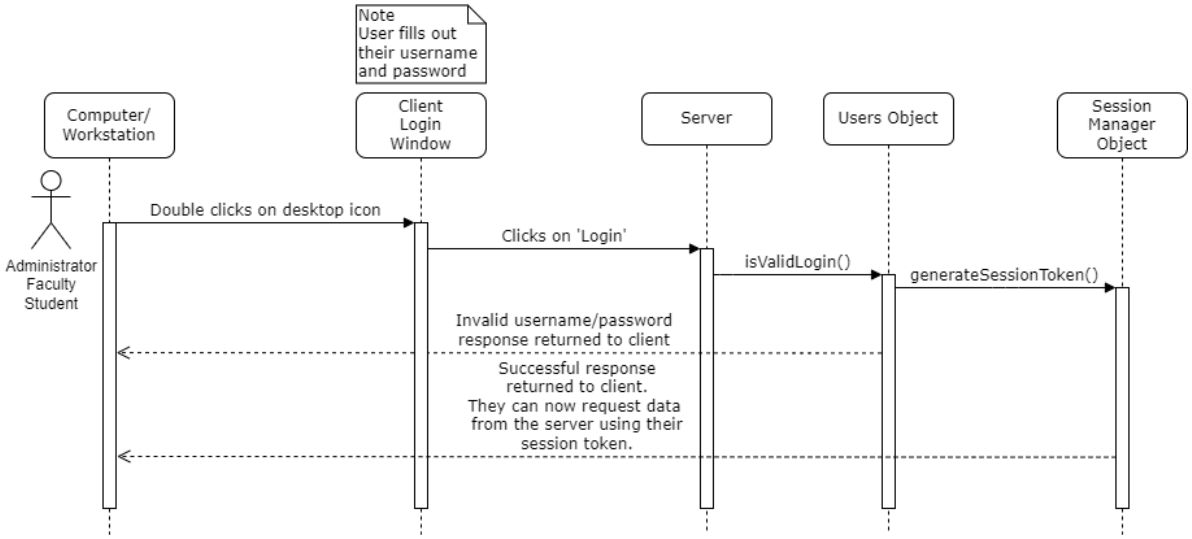
Time

700, “7:00 AM”
715, “7:15 AM”
730, “7:30 AM”
745, “7:45 AM”
800, “8:00 AM”

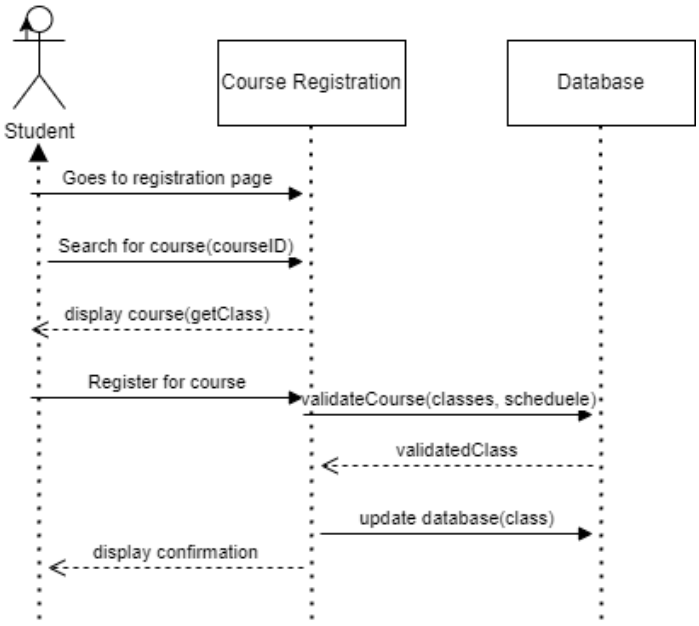
815, “8:15 AM”
830, “8:30 AM”
845, “8:45 AM”
900, “9:00 AM”
915, “9:15 AM”
930, “9:30 AM”
945, “9:45 AM”
1000, “10:00 AM”
1015, “10:15 AM”
1030, “10:30 AM”
1045, “10:45 AM”
1100, “11:00 AM”
1115, “11:15 AM”
1130, “11:30 AM”
1145, “11:45 AM”
1200, “12:00 PM”
1215, “12:15 PM”
1230, “12:30 PM”
1245, “12:45 PM”
1300, “1:00 PM”
1315, “1:15 PM”
1330, “1:30 PM”
1345, “1:45 PM”
1400, “2:00 PM”
1415, “2:15 PM”
1430, “2:30 PM”
1445, “2:45 PM”
1500, “3:00 PM”
1515, “3:15 PM”
1530, “3:30 PM”
1545, “3:45 PM”
1600, “4:00 PM”
1615, “4:15 PM”
1630, “4:30 PM”
1700, “5:00 PM”
1715, “5:15 PM”
1730, “5:30 PM”
1745, “5:40 PM”
1800, “6:00 PM”
1815, “6:15 PM”
1830, “6:30 PM”
1845, “6:45 PM”
1900, “7:00 PM”
1915, “7:15 PM”
1930, “7:30 PM”
1945, “7:45 PM”
2000 , “8:00 PM”
2015, “8:15 PM”
2030, “8:30 PM”
2045, “8:45 PM”
2100, “9:00 PM”
2115, “9:15 PM”
2130, “9:30 PM”
2145, “9:45 PM”
2200, “10:00 PM”

8.Sequence Diagrams(Connor, Andrew)

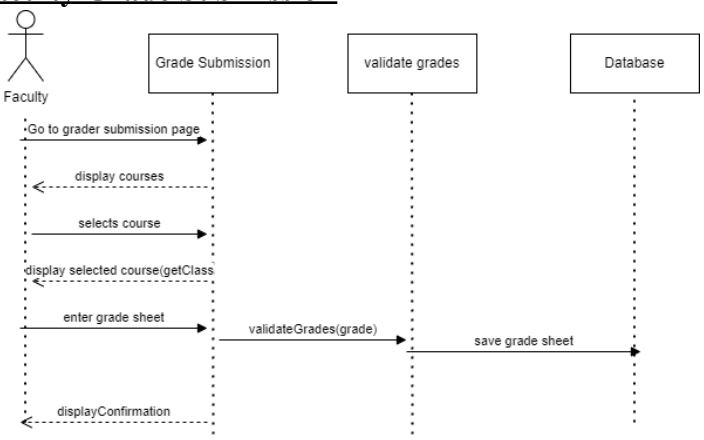
8.1 System Login



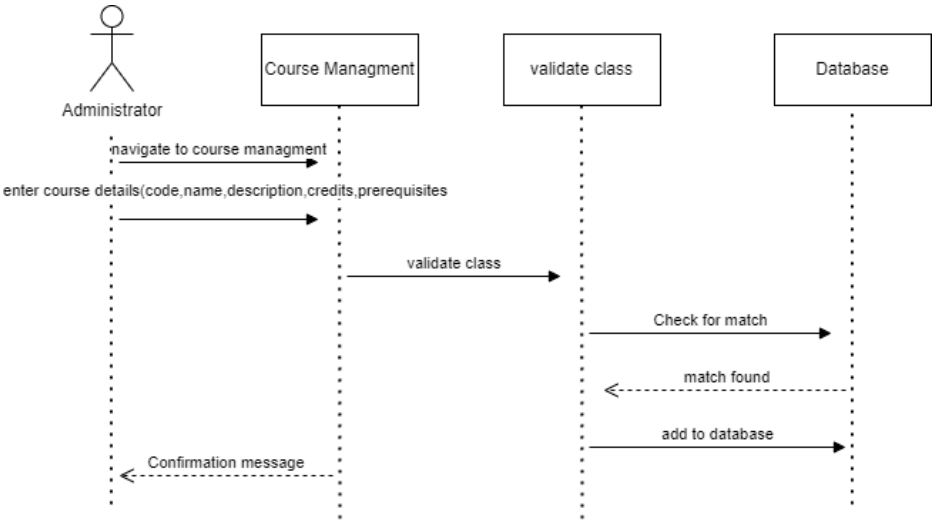
8.2 Student Course Registration



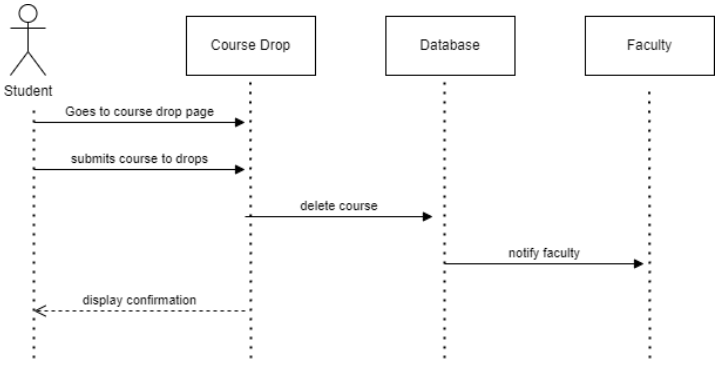
8.3 Faculty Grade Submission



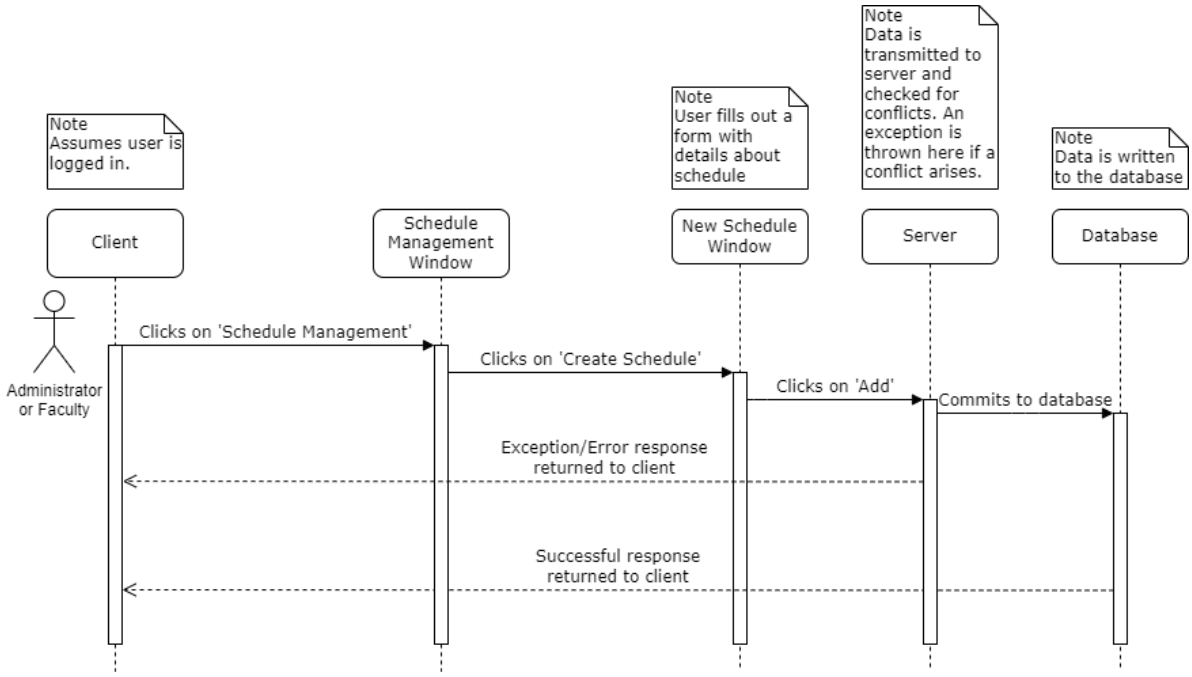
8.4 Administrator Course Management



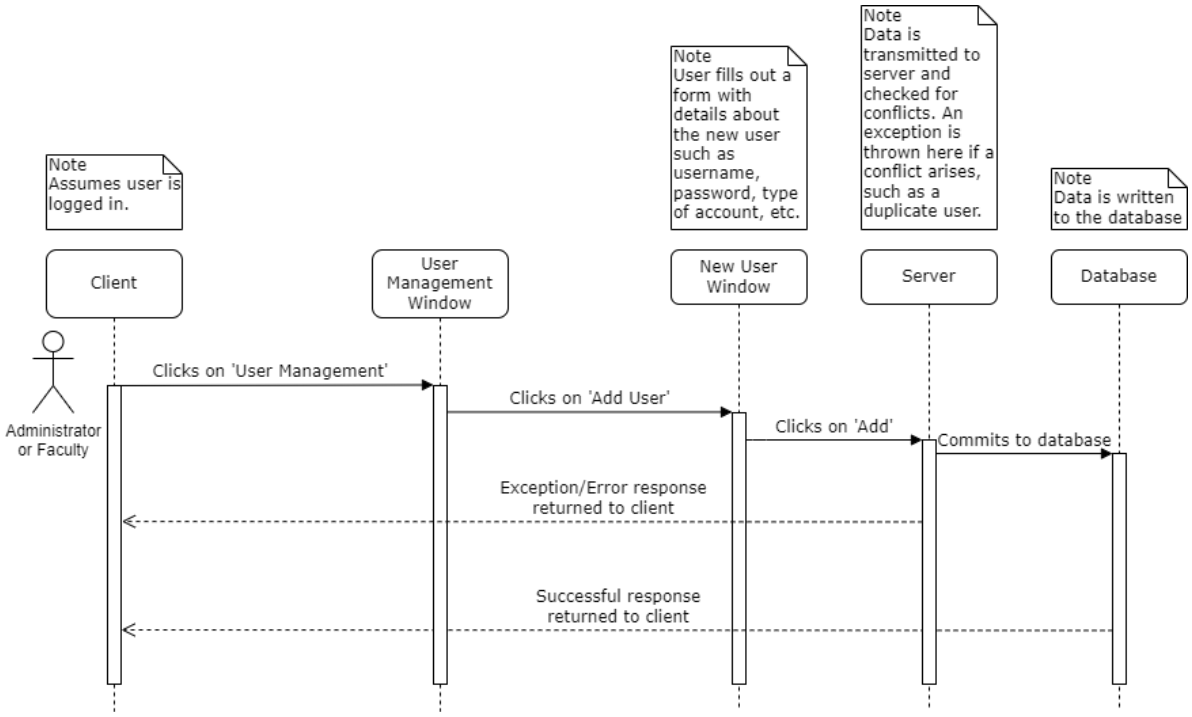
8.5 Student Course Drop



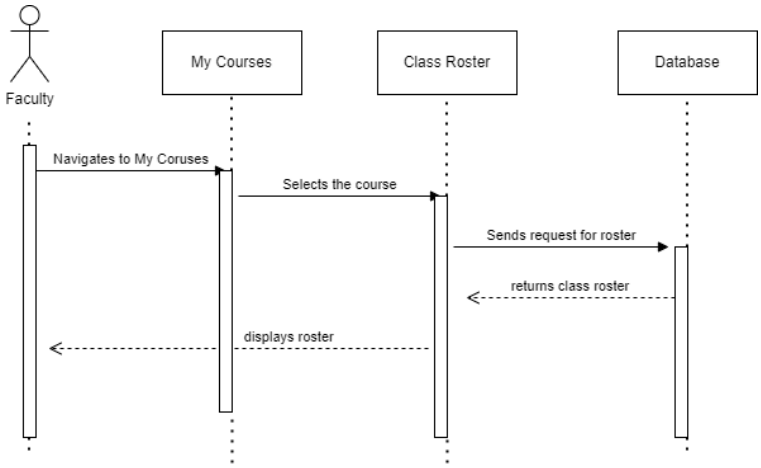
8.6 Administrator Creates Schedule



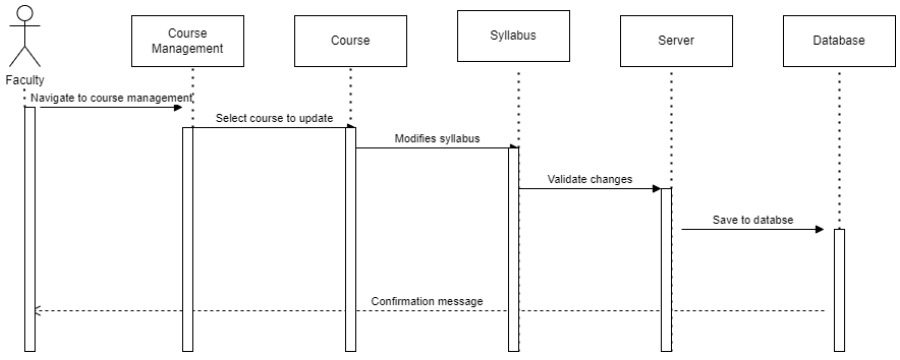
8.7 User Management



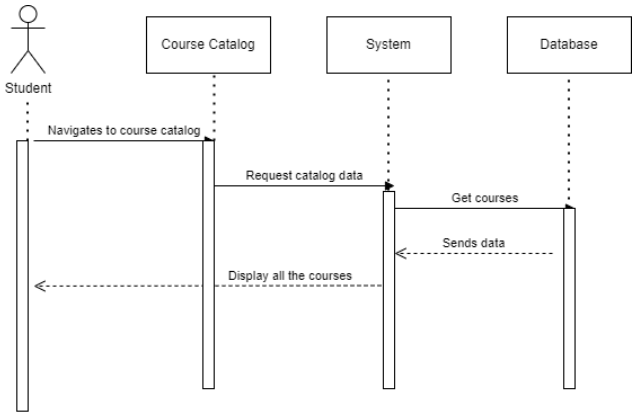
5.8 View Class Roster



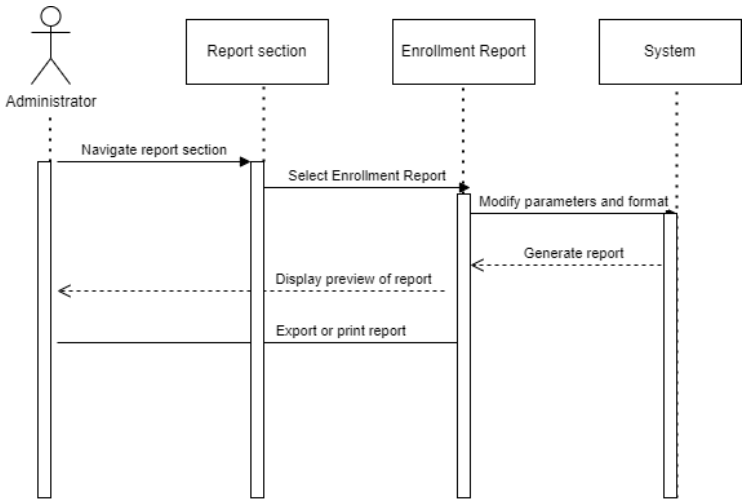
5.9 Update Course Syllabus



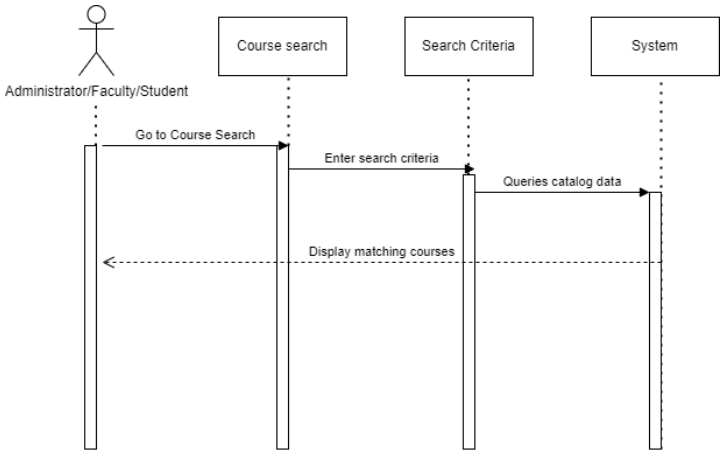
5.10 View Course Catalog



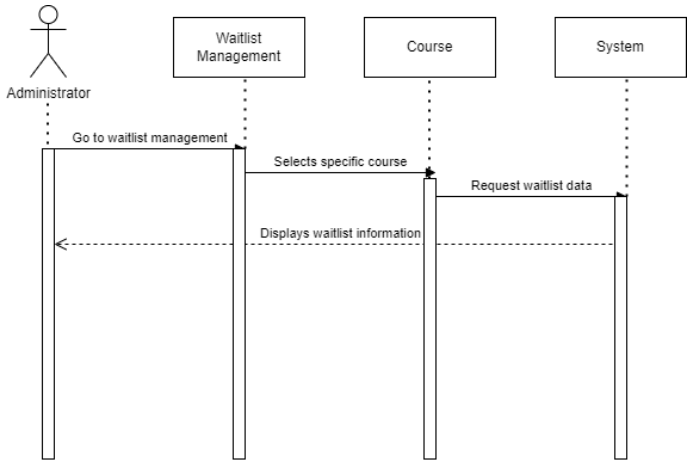
5.11 Generate Enrollment Report



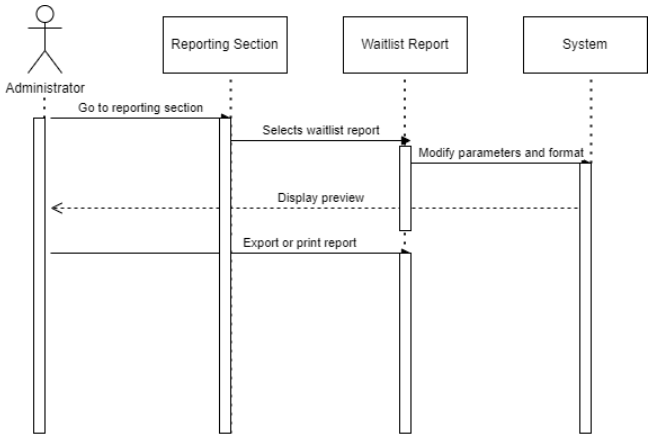
5.12 Search Courses



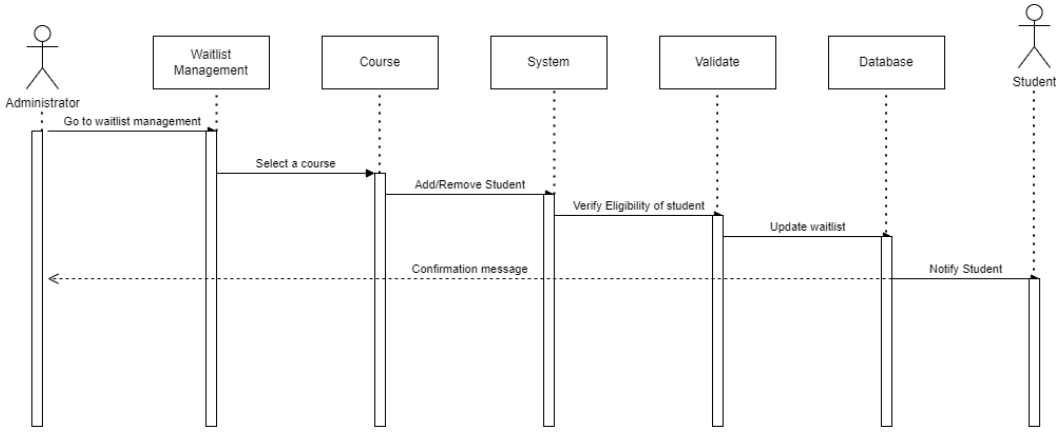
5.13 View Waitlist



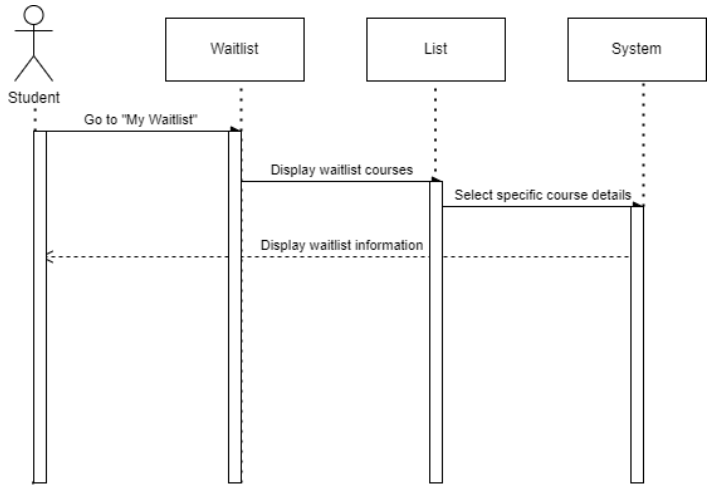
5.14 Generate Waitlist Report



5.15 Modify Course Waitlist



5.16 View Personal Waitlist Position


Student
Waitlist
List
System
Go to "My Waitlist"
Display waitlist courses
Select specific course details
Display waitlist information