

# **College Enrollment System**

## *Design Specification*

## Revision History

| Date       | Revision | Description  | Author               |
|------------|----------|--|----------------------|
| 10/14/2024 | 1.0      | Initial Version Creation                                       | Janelle Kwofie       |
| 10/19/2024 | 1.2      | Added Class descriptions                                       | Janelle Kwofie       |
| 10/24/2024 | 2.0      | Added UML Diagrams   | Bach Ngo             |
| 10/25/2024 | 2.1      | Filled Section 1   | Roy Alkhoury         |
| 10/25/2024 | 2.2      | Filled Section 2   | Roy Alkhoury         |
| 10/26/2024 | 2.3      | Added GUI Design   | Roy Alkhoury         |
| 10/26/2024 | 2.4      | Added UML Class Diagram  | Anthony Josh Legrama |
| 10/27/2024 | 2.5      | Edited UML Class Diagram                                       | Bach Ngo             |
| 10/27/2024 | 2.6      | Added UML Case Diagram   | Anthony Josh Legrama |
| 10/27/2024 | 2.7      | Added UML Sequence Diagrams                                    | Anthony Josh Legrama |
| 10/28/2024 | 2.8      | Added UML Class Diagram (GUI)                                  | Anthony Josh Legrama |
| 10/28/2024 | 2.9      | Updated Class descriptions & minor edits for UML Class Diagram | Bach Ngo             |
| 10/28/2024 | 2.10     | Added UML Class Diagram for server                             | Bach Ngo             |
| 10/29/2024 | 2.11     | Updated UML Class Diagram for GUI                              | Roy Alkhoury         |
| 10/29/2024 | 2.12     | Updated 2.2  | Roy Alkhoury         |
| 10/29/2024 | 2.13     | Updated Class Descriptions                                     | Bach Ngo             |
| 10/29/2024 | 2.14     | Added UML Sequence Diagrams                                    | Anthony Josh Legrama |
| 10/30/2024 | 2.15     | Finalized Project Information                                  | Janelle Kwofie       |
|            |          |  |                      |
|            |          |  |                      |
|            |          |  |                      |

# Table of Contents

|  |           |
|--|-----------|
| <b>1. Overview.....</b>                        | <b>5</b>  |
| 1.1. Scope.....                                | 5         |
| 1.2. Definitions, Acronyms, Abbreviations..... | 5         |
| 1.3. Interfaces and Classes.....               | 5         |
| <b>2. GUI.....</b>                             | <b>6</b>  |
| 2.1. Scope.....                                | 6         |
| 2.2. Attributes.....                           | 6         |
| <b>3. University.....</b>                      | <b>10</b> |
| 3.2. Structure.....                            | 10        |
| 3.2.2. Attributes.....                         | 10        |
| <b>4. Course.....</b>                          | <b>11</b> |
| 4.2. Structure.....                            | 11        |
| 4.2.2. Attributes.....                         | 11        |
| <b>5. Administrator.....</b>                   | <b>12</b> |
| 5.2. Structure.....                            | 12        |
| 5.2.2. Attributes.....                         | 12        |
| <b>6. Student.....</b>                         | <b>13</b> |
| 6.1. Purpose.....                              | 13        |
| 6.2. Structure.....                            | 13        |
| 6.2.2. Attributes.....                         | 13        |
| <b>7. Section.....</b>                         | <b>13</b> |
| 7.2. Structure.....                            | 13        |
| 7.2.1. Constructor.....                        | 13        |
| 7.2.2. Attributes.....                         | 13        |
| <b>8. Schedule Entry.....</b>                  | <b>15</b> |
| 8.2. Structure.....                            | 15        |
| 8.2.1. Constructor.....                        | 15        |
| 8.2.2. Attributes.....                         | 15        |
| <b>9. EnrollStatus (Enumeration).....</b>      | <b>15</b> |
| 9.2. Values.....                               | 15        |
| <b>10. Account.....</b>                        | <b>16</b> |
| 10.1. Purpose.....                             | 16        |
| 10.2. Structure.....                           | 16        |
| <b>11. ClientMsg.....</b>                      | <b>16</b> |
| 11.1. Purpose.....                             | 16        |
| 11.2. Structure.....                           | 16        |
| <b>12. ServerMsg.....</b>                      | <b>17</b> |
| 12.1. Purpose.....                             | 17        |

|  |           |
|--|-----------|
| 12.2. Structure.....                         | 17        |
| <b>13. BodyXYZ.....</b>                      | <b>17</b> |
| 13.1. Purpose.....                           | 17        |
| <b>14. ClientHandler.....</b>                | <b>17</b> |
| 14.1. Purpose.....                           | 17        |
| 14.2. Structure.....                         | 17        |
| <b>15. StudentSessionHandler.....</b>        | <b>18</b> |
| 15.1. Purpose.....                           | 18        |
| 15.2. Structure.....                         | 18        |
| <b>16. AdminSessionHandler.....</b>          | <b>19</b> |
| 16.1. Purpose.....                           | 19        |
| 16.2. Structure.....                         | 19        |
| <b>17. InstructorSessionHandler.....</b>     | <b>20</b> |
| 17.1. Purpose.....                           | 20        |
| 17.2. Structure.....                         | 20        |
| <b>18. UML Diagrams.....</b>                 | <b>21</b> |
| 18.1. UML Use Case Diagram.....              | 21        |
| 18.2. UML Class Diagram.....                 | 22        |
| 18.3. UML Sequence Diagrams.....             | 25        |
| <b>19. Project Organization Content.....</b> | <b>31</b> |
| 19.1. Calendar Overview.....                 | 31        |
| 19.2. Meeting Minutes.....                   | 32        |
| 19.3. Project Timeline.....                  | 33        |
| 19.4. Project Schedule.....                  | 34        |

# 1. **Overview**

## 1.1. **Scope**

This document provides in great detail the design architecture of the college enrollment system we will implement. It includes the methods attributes, diagrams, and the GUI implementation.

## 1.2. **Definitions, Acronyms, Abbreviations**

- CES - College Enrollment System
- GUI - Graphical User Interface
- Core classes - Classes that model the real-life problem. See the first diagram in Section 18.2: UML Class Diagrams for concrete examples.

## 1.3. **Interfaces and Classes**

- [GUI](#)
- [University](#)
- [Course](#)
- [Administrator](#)
- [Student](#)
- [Section](#)
- [Schedule Entry](#)
- [UML Diagrams](#)

## **2. GUI**

### **2.1. Scope**

The GUI is encapsulated inside a JFrame, which emulates the main page of the College Enrollment System, CES. Inside the JFrame, there will be 2 central panels. The main panel goes over what will be displayed, and the options panel to the side provides options to assist the main panel, such as opening the schedule.

### **2.2. Attributes**

#### **2.2.1. Variables**

##### **2.2.1.1. JFrame frame - the base of the GUI**

**2.2.1.1.1. JPanel optionsPanel** - side panel of options to select what to display

**2.2.1.1.2. JPanel mainPanel** - the main panel displaying a specific option from optionsPanel

**2.2.1.1.3. JPanel loginScreen** - template screen if not logged in

##### **2.2.1.2. JPanel optionsPanel(new GridLayout)**

**2.2.1.2.1. JLabel Options** - uneditable label that displays “options” at the top of the optionsPanel

**2.2.1.2.2. JButton classSearch** - navigates to the mainPanel’s classSearch card

**2.2.1.2.3. JButton courseCatalog** - navigates to the mainPanel’s courseCatalog card

**2.2.1.2.4. JButton schedule** - navigates to the mainPanel’s schedule card

##### **2.2.1.3. JPanel mainPanel(new CardLayout)**

**2.2.1.3.1. JLabel templateStart** - tells the user to click an option to start

**2.2.1.3.2. JPanel schedulePanel** - the card display for the schedule

**2.2.1.3.2.1. JLabel mySchedule** - displays the user’s schedule

**2.2.1.3.2.2. JButton dropButton** - user wants to drop a class from their schedule

**2.2.1.3.3. JPanel searchClassPanel** - the card display for class searching

**2.2.1.3.3.1. JTextArea courseNameText** - where the user enters the course name to search

**2.2.1.3.3.2. JTextArea courseNumberText** - where the user enters the course number to search

**2.2.1.3.3.3. JTextArea coursePrefix** - where the user enters the course prefix to search

**2.2.1.3.3.4. JTextArea instructorText** - where the user enters the instructor name to search

- 2.2.1.3.3.5. String course\_name\_query - the course name as a string
- 2.2.1.3.3.6. String course\_number\_query - the course number as a string
- 2.2.1.3.3.7. String course\_prefix\_query - the course prefix as a string
- 2.2.1.3.3.8. String instructor\_query - the instructor as a string
- 2.2.1.3.4. JPanel courseCatalogPanel - the card display for the university's course catalog
  - 2.2.1.3.4.1. JLabel courseCatalog - displays the course catalog
- 2.2.1.4. JPanel loginScreen(new BorderLayout)
  - 2.2.1.4.1. JButton login - student clicks to login
    - 2.2.1.4.1.1. JTextArea uniBox - where the student enters uni name
    - 2.2.1.4.1.2. JTextArea loginBox - where the student enters the loginID
    - 2.2.1.4.1.3. JTextArea passwordBox - where the student enters the password
  - 2.2.1.4.2. JLabel loginMessage - displays a message to login
- 2.2.2. **Methods**
  - 2.2.2.1. **Main Methods:**
    - 2.2.2.1.1. Boolean isLoggedIn() - checks to see if the user is logged in to display templateScreen. If logged in, displays the optionPanel with mainDisplay
    - 2.2.2.1.2. Void initializeOptions() - initializes the optionPanel
    - 2.2.2.1.3. Void setOptionListeners() - creates listeners for all the optionPanel buttons
    - 2.2.2.1.4. ArrayList<Section> getCurrentSchedule() - returns the user's schedule
    - 2.2.2.1.5. ArrayList<Course> getCoursesByFilter() - returns the courses found after searching
    - 2.2.2.1.6. ArrayList <Course> getAllCourses() - returns all the courses for the course catalog
  - 2.2.2.2. **Getters:**
    - 2.2.2.2.1. String getCourseName() - gets the course name
    - 2.2.2.2.2. String getCoursePrefix() - gets the course prefix
    - 2.2.2.2.3. String getCourseNumber() - gets the course number
    - 2.2.2.2.4. String getInstructor() - gets the instructor of courses
    - 2.2.2.2.5. String getUniName() - gets the login's name of uni
    - 2.2.2.2.6. String getLoginID() - gets the login's loginID

2.2.2.2.7. String getPassword() - gets the login's password

**2.2.2.3. Setters:**

2.2.2.3.1. Void setCourseName(String course) - sets the course name

2.2.2.3.2. Void setCoursePrefix(String prefix) - sets the course prefix

2.2.2.3.3. Void setCourseNumber(String number) - sets the course number

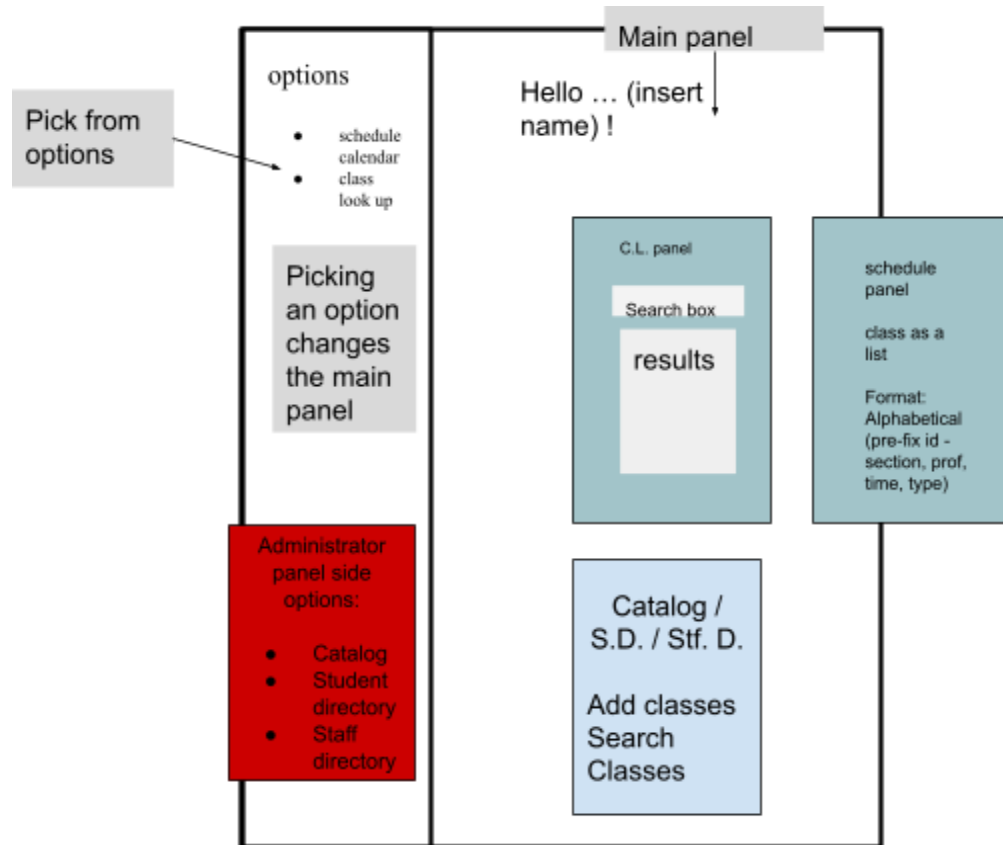
2.2.2.3.4. Void setInstructor(String instructor) - sets the instructor

**2.2.3. GUI Designs**

Comment

|                 |                               |                |        |
|-----------------|-------------------------------|----------------|--------|
| Options         | Course name:                  | Course number: | Login: |
| Courses Catalog | Course prefix:                | Instructor:    |        |
| Schedule        | Enter                         |                |        |
| Calendar        | Courses Found:                |                |        |
|                 | CS 101-(num) Professor (Name) | Course Details | Enroll |
|                 | CS 201-(num) Professor (Name) | Course Details | Enroll |





### **3. University**

#### **3.1. Purpose**

- 3.1.1. University is the main class for the CES. It houses both the information for the students, courses, and professors that belong to the University.

#### **3.2. Structure**

##### **3.2.1. Constructor**

- 3.2.1.1. University(name: String, location: String)

##### **3.2.2. Attributes**

- 3.2.2.1. String id - A unique identifier to represent the University
- 3.2.2.2. String name - The name of the University
- 3.2.2.3. String location - The location of the University
- 3.2.2.4. ArrayList<Administrator> admins - All the administrators belonging to the University
- 3.2.2.5. Map<String, Course> catalog - A mapping between the course ID and the actual Course. Contains all the courses belonging to the University
- 3.2.2.6. Map<String, Student> students - A mapping between the student ID and the actual Student. Contains all the students belonging to the University
- 3.2.2.7. Map<String, Instructor> instructors - A mapping between the instructor ID and the actual Instructor. Contains all the instructors belonging to the University

##### **3.2.3. Methods**

- 3.2.3.1. ArrayList<Course> getAllCourses() - Returns the catalog as a list
- 3.2.3.2. ArrayList<Course> getCoursesByFilter(filter: (course): boolean) - Returns a list of courses from the catalog that pass the filter
- 3.2.3.3. void addCourse(course: Course) - Takes a Course and adds it to the catalog if it does not exist
- 3.2.3.4. Course getCourseByID(courseID: String) - Returns the Course with the provided ID or null if doesn't exist
- 3.2.3.5. void delCourse(courseID: String) - Delete the course with the provided course ID
- 3.2.3.6. void editCourse(course: Course) - Edit the course with the newly provided Course
- 3.2.3.7. void addAdmin(administrator: Administrator) - Add an admin under this University
- 3.2.3.8. void addStudent(student: Student) - Add a student under this University

- 3.2.3.9. void addInstructor(instructor: Instructor) - Add an instructor under this University
- 3.2.3.10. String getName() - Get the university name
- 3.2.3.11. String getLocation() - Get the location of the university
- 3.2.3.12. ArrayList<Student> getAdmins() - Get a list of admins
- 3.2.3.13. ArrayList<Administrator> getStudents() - Get a list of students
- 3.2.3.14. ArrayList<Instructor> getInstructors() - Get a list of instructors
- 3.2.3.15. void setName(name: String) - Set the name of the university
- 3.2.3.16. void setLocation(location: String) - Set the location of the university

## 4. Course

### 4.1. Purpose

- 4.1.1. Course is a class that holds information about the course. Course has a list of Sections this is where Students get to enroll.

### 4.2. Structure

#### 4.2.1. Constructor

- 4.2.1.1. Course(prefix: String, number: String, description: String)

#### 4.2.2. Attributes

- 4.2.2.1. String id - A unique identifier to represent the Course
- 4.2.2.2. String prefix - A prefix to represent the Course subject
- 4.2.2.3. String number - A number unique to its group of prefixes to represent the Course
- 4.2.2.4. String description - The course's description
- 4.2.2.5. ArrayList<Course> prerequisites - An ArrayList<Course> of classes that must be taken before Course
- 4.2.2.6. ArrayList<Section> sections (public) - An ArrayList<Section> that has every available section for the Course

#### 4.2.3. Methods

- 4.2.3.1. void insertSection(section: Section) - Adds a Section to the Course
- 4.2.3.2. void delSection(sectionID: String) - Deletes the Section from Course if found
- 4.2.3.3. void delPrereq(courseID: String) - Deletes the prereq Course from Course if found
- 4.2.3.4. ArrayList<Course> getPrerequisites() - Returns the prerequisites for the Course
- 4.2.3.5. ArrayList<Section> getSections() - Returns the sections for the Course
- 4.2.3.6. void insertPrereq(course: Course) - Adds a Prerequisite to the Course

- 4.2.3.7. void delPrereq(courseID: String) - Removes a prerequisite based on the ID
- 4.2.3.8. void insertSection(section: Section) - Adds a Section to the Course
- 4.2.3.9. void delSection(sectionID: String) - Removes a section based on the ID
- 4.2.3.10. String getID() - Get the ID of the section
- 4.2.3.11. String getPrefix() - Get the prefix of the section
- 4.2.3.12. String getNumber() - Get the section's number
- 4.2.3.13. String getDesc() - Get the section's description
- 4.2.3.14. void setPrefix(number: String) - Set the prefix of the section
- 4.2.3.15. void setNumber(number: String) - Set the number of the section
- 4.2.3.16. void setDesc(description: String) - Set the description of the section

## **5. Administrator**

### **5.1. Purpose**

- 5.1.1. Administrator is the class that holds all administrator information and controls the actions from the administrator's perspective.

### **5.2. Structure**

#### **5.2.1. Constructor**

- 5.2.1.1. Administrator(name: String, account: Account)

#### **5.2.2. Attributes**

- 5.2.2.1. String id - A unique identifier to represent the Administrator
- 5.2.2.2. Account account - The Administrator's University account
- 5.2.2.3. String name - The name of the Administrator

#### **5.2.3. Methods**

- 5.2.3.1. String getID() - The admin's ID
- 5.2.3.2. Account getAccount() - The admin's account
- 5.2.3.3. String getName() - The admin's name

## **6. Student**

### **6.1. Purpose**

- 6.1.1.** Student is the class that holds all student information and controls the actions from the student's perspective.

### **6.2. Structure**

#### **6.2.1. Constructor**

- 6.2.1.1.** Student(name: String, account: Account)

#### **6.2.2. Attributes**

- 6.2.2.1.** String id - A unique identifier to represent the Student
- 6.2.2.2.** Account account - The Student's University account
- 6.2.2.3.** String name - The name of the Student
- 6.2.2.4.** ArrayList<Section> past\_enrollments - Every section the Student was enrolled in
- 6.2.2.5.** ArrayList<Section> enrolling - Every section the Student is currently enrolled in

#### **6.2.3. Methods**

- 6.2.3.1.** void enroll(section: Section) - Adds Student to the Section's enrollment list, waitlist, or neither depending on each capacity.
- 6.2.3.2.** void drop(sectionID: String) - Removes student from any of Section's lists if they are listed
- 6.2.3.3.** ArrayList<Section> getCurrentSchedule() - Returns the enrolling (ArrayList<Section>) for Student
- 6.2.3.4.** ArrayList<Section> getPastEnrollments() - Returns the past\_enrollments (ArrayList<Section>) for Student
- 6.2.3.5.** String getID() - Get the student's ID
- 6.2.3.6.** String getName() - Get the student's name
- 6.2.3.7.** Account getAccount() - Get the student's account

## **7. Section**

### **7.1. Purpose**

- 7.1.1.** Section is the class that holds information about each section of the course. This class is what is referenced when Students enroll/drop a course.

### **7.2. Structure**

#### **7.2.1. Constructor**

- 7.2.1.1.** Section(course: Course, cap: int, wait: int, instructor: Instructor)

#### **7.2.2. Attributes**

- 7.2.2.1.** String id - A unique identifier to represent the Section

- 7.2.2.2. Course course - The Course section is tied to
- 7.2.2.3. String number - The section's number
- 7.2.2.4. boolean active - A boolean that identifies if the Section is available
- 7.2.2.5. int max\_capacity - A number that represents the Section's maximum capacity
- 7.2.2.6. int max\_wait - A number that represents the waitlist's maximum capacity
- 7.2.2.7. Instructor instructor - The instructor for the Section
- 7.2.2.8. ArrayList<Student> enrolled - All the Students enrolled in the Section
- 7.2.2.9. ArrayList<Student> waitlisted - All the Students waitlisted for the Section
- 7.2.2.10. ScheduleEntry[] schedule - The schedule for the Section
- 7.2.3. Methods**
  - 7.2.3.1. EnrollStatus enrollStudent(student: Student) - Adds Student to the section. Returns whether the student is added to enrolled, waitlisted, or not added at all.
  - 7.2.3.2. void dropStudent(studentID: String) - Removes student from any of Section's lists if they are listed
  - 7.2.3.3. ScheduleEntry[] getSchedule() - Returns the schedule ScheduleEntry[] for the Section
  - 7.2.3.4. void setSchedule(schedule: ScheduleEntry[]) - Returns void, sets the ScheduleEntry[] for the Section
  - 7.2.3.5. boolean isFull() - Returns if both the enrollment and waitlist are full for the Section
  - 7.2.3.6. boolean isActive() - Returns if the Section is in the current catalog
  - 7.2.3.7. String getID() - Returns the section's ID
  - 7.2.3.8. Course getCourse() - Returns the course the section's tied to
  - 7.2.3.9. String getNumber() - Returns the section's number
  - 7.2.3.10. int getMaxCapacity() - Returns the section's capacity
  - 7.2.3.11. int getMaxWaitlistSize() - Returns the section's waitlist max size
  - 7.2.3.12. Instructor getInstructor() - Returns the section's instructor
  - 7.2.3.13. List<Student> getEnrolled() - Returns the enrolled students
  - 7.2.3.14. List<Student> getWaitlisted() - Returns the waitlisted students
  - 7.2.3.15. ScheduleEntry[] getSchedule() - Returns the section's schedule
  - 7.2.3.16. void setNumber(num: String) - Set the section number
  - 7.2.3.17. void setActiveState(state: boolean) - Set whether the section is active or not
  - 7.2.3.18. void setMaxCapacity(cap: int) - Sets the capacity
  - 7.2.3.19. void setMaxWaitSize(wait: int) - Sets the waitlist size

- 7.2.3.20. void setInstructor(instructor: Instructor) - Sets the instructor
- 7.2.3.21. void setSchedule(schedule: ScheduleEntry[]) - Sets the schedule

## **8. Schedule Entry**

### **8.1. Purpose**

- 8.1.1. Schedule Entry is a class that holds the Sections organization details.

### **8.2. Structure**

#### **8.2.1. Constructor**

- 8.2.1.1. ScheduleEntry(location: String, is\_sync: boolean, day\_of\_week: String, starttime: Time | null, endtime: Time | null) - starttime and endtime are ignored (can be null) if is\_sync is false

#### **8.2.2. Attributes**

- 8.2.2.1. String location - The location of the Section (either a physical classroom or online)
- 8.2.2.2. boolean is\_sync - Whether the section requires students to attend synchronous or not
- 8.2.2.3. String day\_of\_week - The day of the week the Section is in session
- 8.2.2.4. Time start\_time - The time the Section starts.
- 8.2.2.5. Time end\_time - The time the Section ends

#### **8.2.3. Methods**

- 8.2.3.1. Tuple<Time> getTime() - Returns the start and end time of the Section
- 8.2.3.2. String getLocation() - Returns the location of the Section
- 8.2.3.3. String getDayofWeek() - Returns the day(s) the Section takes place
- 8.2.3.4. boolean isSync() - Returns is\_sync

## **9. EnrollStatus (Enumeration)**

### **9.1. Purpose**

- 9.1.1. Enroll Status is an Enumeration that details the Student's current enrollment status in the Section.

### **9.2. Values**

- 9.2.1. Enrolled - The student enrolled successfully
- 9.2.2. Waitlisted - The student is put on the waitlist
- 9.2.3. Unsuccessful - The student can't be enrolled in this section

## **10. Account**

### **10.1. Purpose**

- 10.1.1. Store the account information for each user of the system.

### **10.2. Structure**

#### **10.2.1. Constructor**

- 10.2.1.1. Account(email: String, password: String)

#### **10.2.2. Attributes**

- 10.2.2.1. String email - The email of the user
- 10.2.2.2. String password - The password of the user

#### **10.2.3. Methods**

- 10.2.3.1. boolean verify(email: String, password: String) - Returns whether or not the provided email and password match this account

## **11. ClientMsg**

### **11.1. Purpose**

- 11.1.1. Represents the message sent by the client to the server.

### **11.2. Structure**

#### **11.2.1. Constructor**

- 11.2.1.1. ClientMsg(method: String, resource: String, authorID: String, body: Serializable | null)

#### **11.2.2. Attributes**

- 11.2.2.1. String method - Either GET, CREATE, DELETE, or EDIT. Tells which operation the client wants to perform.
- 11.2.2.2. String resource - Tells which resource the client wants to perform on (example: courses, schedules, etc.)
- 11.2.2.3. String authorID - Who the client is.
- 11.2.2.4. Serializable body - Any Serializable objects (such as Course or University) or null. Any additional information (depends on each request).



## **12. ServerMsg**

### **12.1. Purpose**

12.1.1. Represents the message sent by the server to the client.

### **12.2. Structure**

#### **12.2.1. Constructor**

12.2.1.1. ServerMsg(status: String, body: Serializable | null)

#### **12.2.2. Attributes**

12.2.2.1. String status - Either OK or ERR. Tells whether the client's request is successful or not.

12.2.2.2. Serializable body - Any Serializable objects or null. In the case of a GET client message, this is often the resource the client needs. In the case of an ERR message, this may contain additional information on what the issue is.

## **13. BodyXYZ**

### **13.1. Purpose**

13.1.1. Represents some niche message body format. Since not all core classes are appropriate for all use cases, specialized message body classes are created as needed. For example, no existing classes are suitable to carry course filtering information over the network. Therefore, BodySearch is created for this use case. Whenever possible, prefer using built-in and existing Serializable classes.

## **14. ClientHandler**

### **14.1. Purpose**

14.1.1. Handles messages that don't belong to a session (such as client initial request). If sufficient credentials are provided, it'll establish a SessionHandler and pass all network streams to that session until the session terminates.

### **14.2. Structure**

#### **14.2.1. Constructor**

14.2.1.1. ClientHandler(socket: Socket, universities: University[])

#### **14.2.2. Attributes**

14.2.2.1. Socket socket - The socket serving the client.

14.2.2.2. University[] universities - The list of universities, obtained from the entry point of the server code.

## **15. StudentSessionHandler**

### **15.1. Purpose**

- 15.1.1. Handles messages in a student's session until the session is terminated via the client or prematurely aborts.

### **15.2. Structure**

#### **15.2.1. Constructor**

- 15.2.1.1. StudentSessionHandler(socket: Socket, istream: ObjectInputStream, ostream: ObjectOutputStream, university: University, student: Student)

#### **15.2.2. Attributes**

- 15.2.2.1. Socket socket - The socket serving the student.
- 15.2.2.2. ObjectInputStream istream - The input stream obtained from ClientHandler.
- 15.2.2.3. ObjectOutputStream ostream - The output stream obtained from ClientHandler.
- 15.2.2.4. University university - The university the student belongs to.
- 15.2.2.5. Student student - The student in this session.

#### **15.2.3. Methods**

- 15.2.3.1. void run() - Listens on the input stream for incoming requests and passes them to appropriate private methods. Waits for private methods to finish and sends the response to the client.
- 15.2.3.2. ServerMsg logout(req: ClientMsg) - If req is valid, returns an OK ServerMsg and tells run() to terminate, or an ERR ServerMsg.
- 15.2.3.3. ServerMsg fetchCourses(req: ClientMsg) - If req is valid, returns an OK ServerMsg containing the courses being requested, or an ERR ServerMsg.
- 15.2.3.4. ServerMsg fetchSchedule(req: ClientMsg) - If req is valid, returns an OK ServerMsg containing the schedule being requested, or an ERR ServerMsg.
- 15.2.3.5. ServerMsg enroll(req: ClientMsg) - If req is valid, returns an OK ServerMsg containing the enroll status, or an ERR ServerMsg.
- 15.2.3.6. ServerMsg drop(req: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 15.2.3.7. ServerMsg fetchPastEnrollment(req: ClientMsg) - If req is valid, returns an OK ServerMsg containing the enrollment information being requested, or an ERR ServerMsg.

## **16. AdminSessionHandler**

### **16.1. Purpose**

- 16.1.1. Handles messages in an admin's session until the session is terminated via the client or prematurely aborts.

### **16.2. Structure**

#### **16.2.1. Constructor**

- 16.2.1.1. AdminSessionHandler(socket: Socket, istream: ObjectInputStream, ostream: ObjectOutputStream, university: University, admin: Administrator)

#### **16.2.2. Attributes**

- 16.2.2.1. Socket socket - The socket serving the admin.
- 16.2.2.2. ObjectInputStream istream - The input stream obtained from ClientHandler.
- 16.2.2.3. ObjectOutputStream ostream - The output stream obtained from ClientHandler.
- 16.2.2.4. University university - The university the admin belongs to.
- 16.2.2.5. Administrator admin- The admin in this session.

#### **16.2.3. Methods**

- 16.2.3.1. void run() - Listens on the input stream for incoming requests and passes them to appropriate private methods. Waits for private methods to finish and sends the response to the client.
- 16.2.3.2. ServerMsg logout(req: ClientMsg) - If req is valid, returns an OK ServerMsg and tells run() to terminate, or an ERR ServerMsg.
- 16.2.3.3. ServerMsg fetchReport(request: ClientMsg) - If req is valid, returns an OK ServerMsg containing the report or an ERR ServerMsg.
- 16.2.3.4. ServerMsg createStudent(request: ClientMsg) - If req is valid, returns an OK ServerMsg with the newly created student or an ERR ServerMsg.
- 16.2.3.5. ServerMsg addSection(request: ClientMsg) - If req is valid, returns an OK ServerMsg with the newly created section or an ERR ServerMsg.
- 16.2.3.6. ServerMsg editSection(request: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 16.2.3.7. ServerMsg delSection(request: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 16.2.3.8. ServerMsg addCourse(request: ClientMsg) - If req is valid, returns an OK ServerMsg with the newly created course or an ERR ServerMsg.

- 16.2.3.9. ServerMsg editCourse(request: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 16.2.3.10. ServerMsg delCourse(request: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 16.2.3.11. ServerMsg enrollStudent(request: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 16.2.3.12. ServerMsg dropStudent(request: ClientMsg) - If req is valid, returns an OK ServerMsg or an ERR ServerMsg.
- 16.2.3.13. ServerMsg fetchCourses(request: ClientMsg) - If req is valid, returns an OK ServerMsg containing the courses being requested, or an ERR ServerMsg.

## 17. **InstructorSessionHandler**

### 17.1. Purpose

- 17.1.1. Handles messages in an instructor's session until the session is terminated via the client or prematurely aborts.

### 17.2. Structure

#### 17.2.1. Constructor

- 17.2.1.1. InstructorSessionHandler(socket: Socket, istream: ObjectInputStream, ostream: ObjectOutputStream, university: University, instructor: Instructor)

#### 17.2.2. Attributes

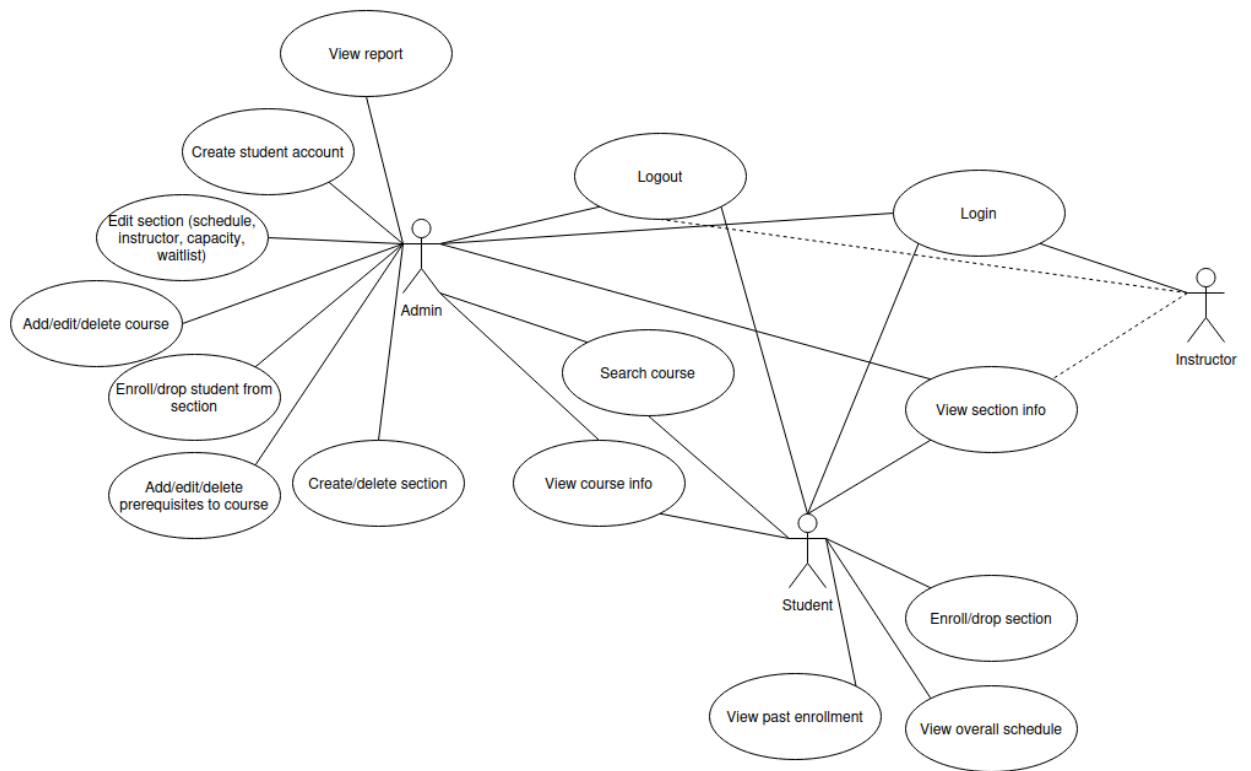
- 17.2.2.1. Socket socket - The socket serving the admin.
- 17.2.2.2. ObjectInputStream istream - The input stream obtained from ClientHandler.
- 17.2.2.3. ObjectOutputStream ostream - The output stream obtained from ClientHandler.
- 17.2.2.4. University university - The university the instructor belongs to.
- 17.2.2.5. Instructor instructor - The instructor in this session.

#### 17.2.3. Methods

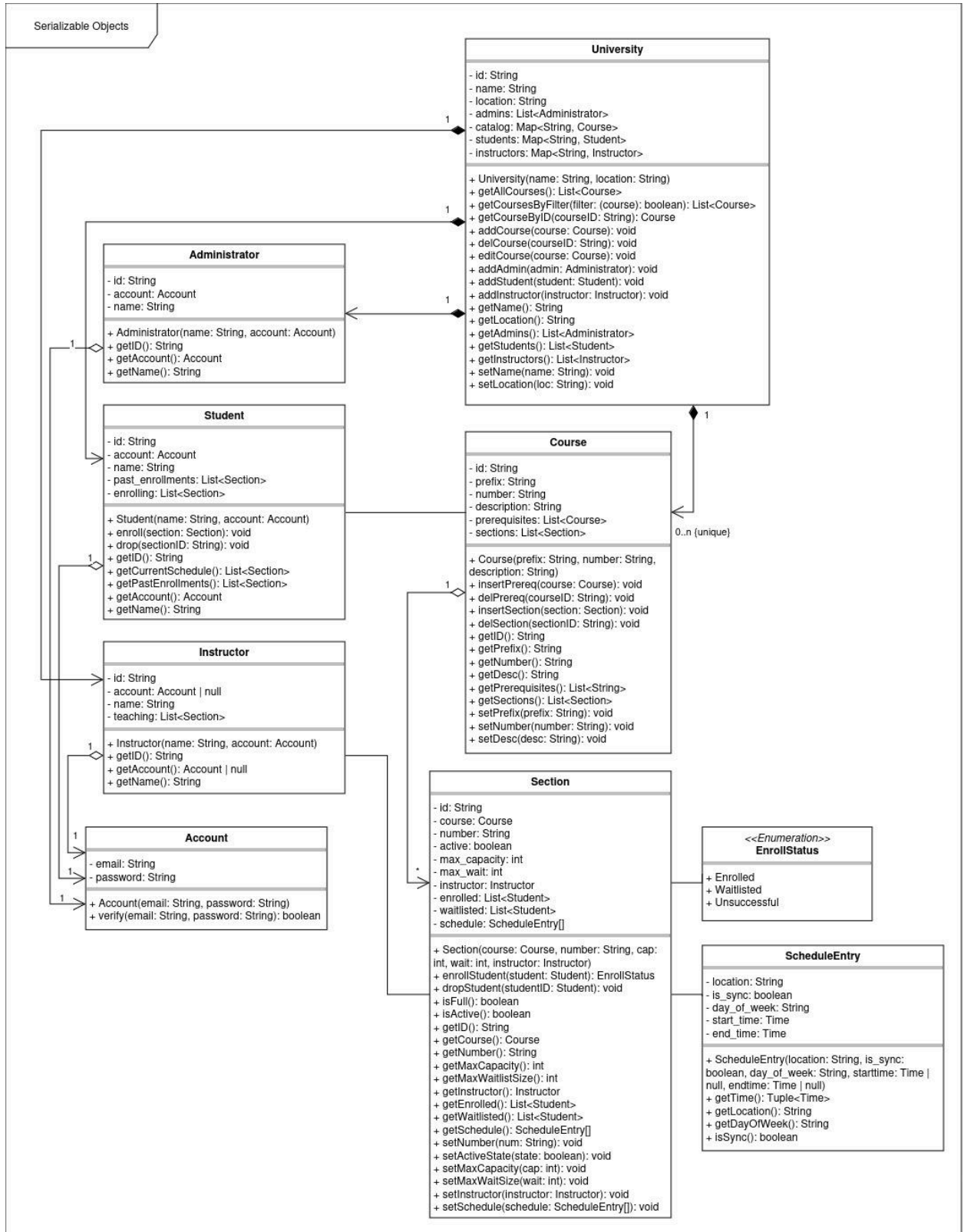
- 17.2.3.1. void run() - Listens on the input stream for incoming requests and passes them to appropriate private methods. Waits for private methods to finish and sends the response to the client.
- 17.2.3.2. ServerMsg logout(req: ClientMsg) - If req is valid, returns an OK ServerMsg and tells run() to terminate, or an ERR ServerMsg.
- 17.2.3.3. ServerMsg viewSections(req: ClientMsg) - If req is valid, returns an OK ServerMsg containing the sections being requested, or an ERR Server Msg.

## 18. UML Diagrams

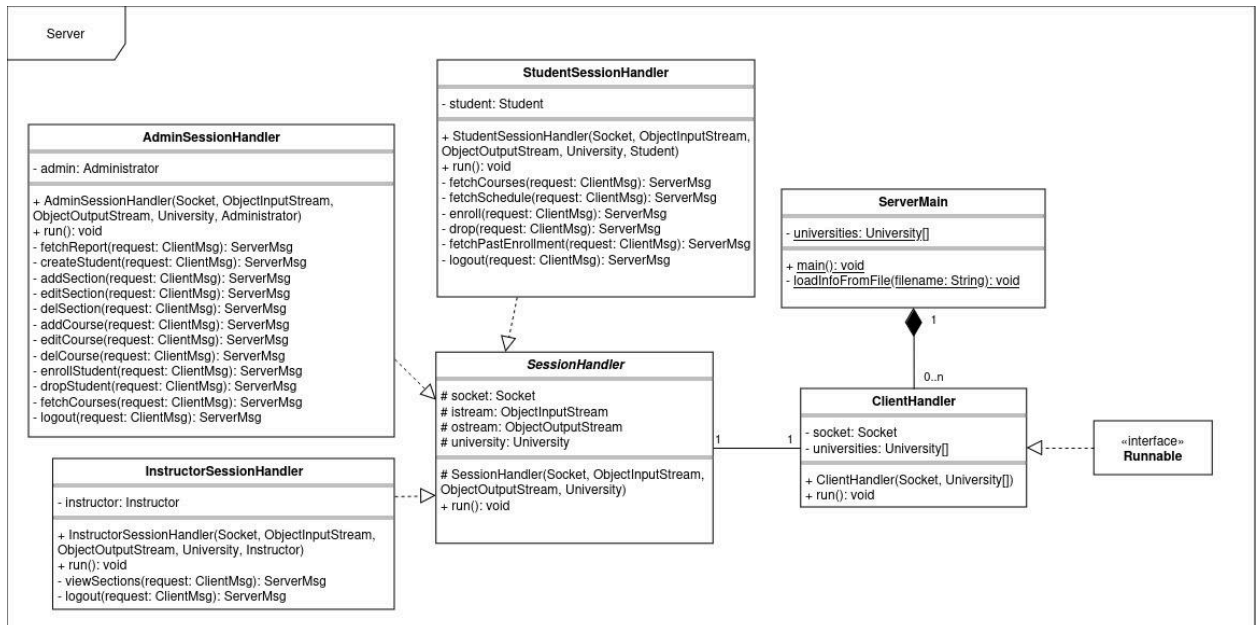
### 18.1. UML Use Case Diagram



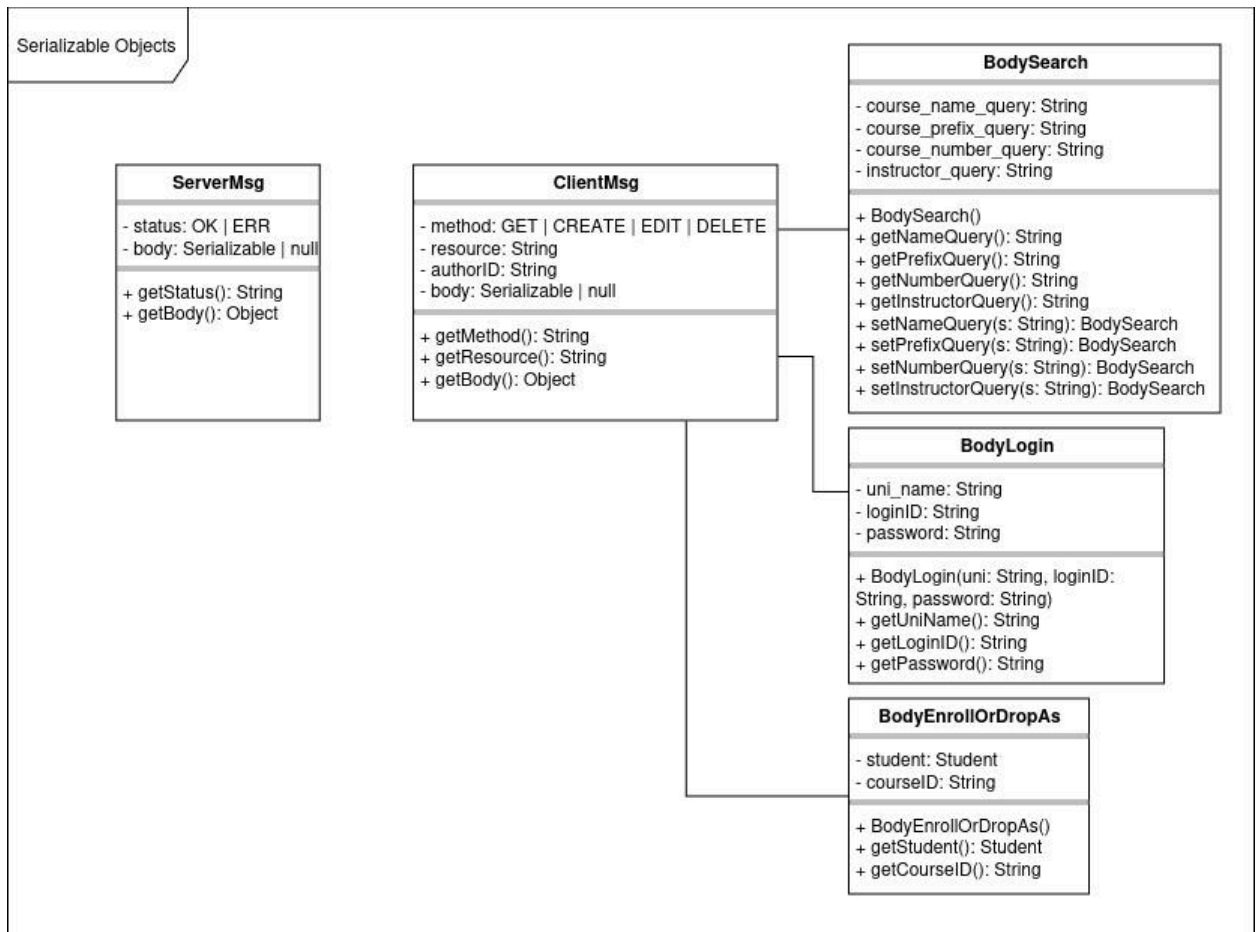
## 18.2. UML Class Diagram



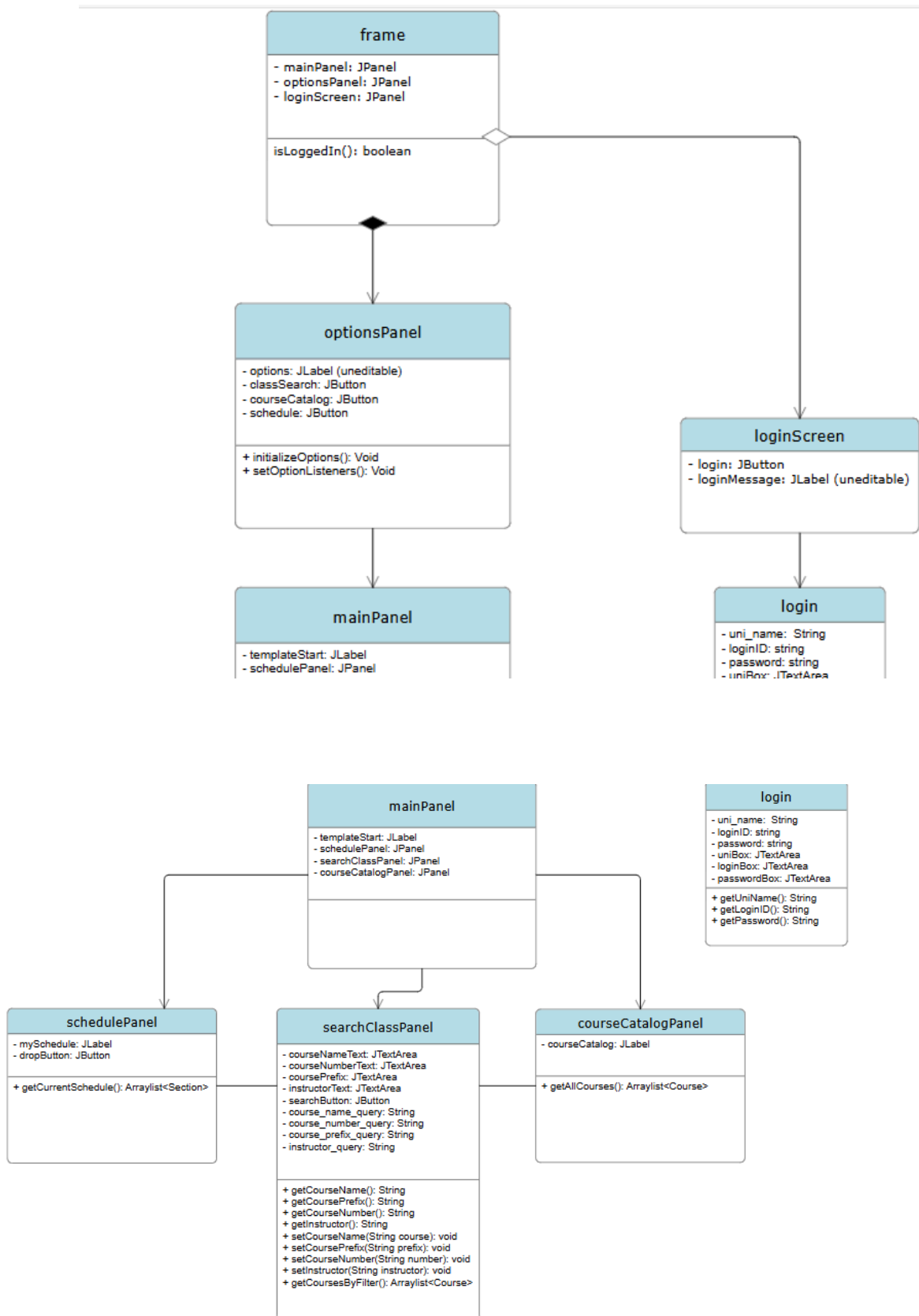
## Server:



## Network Message:



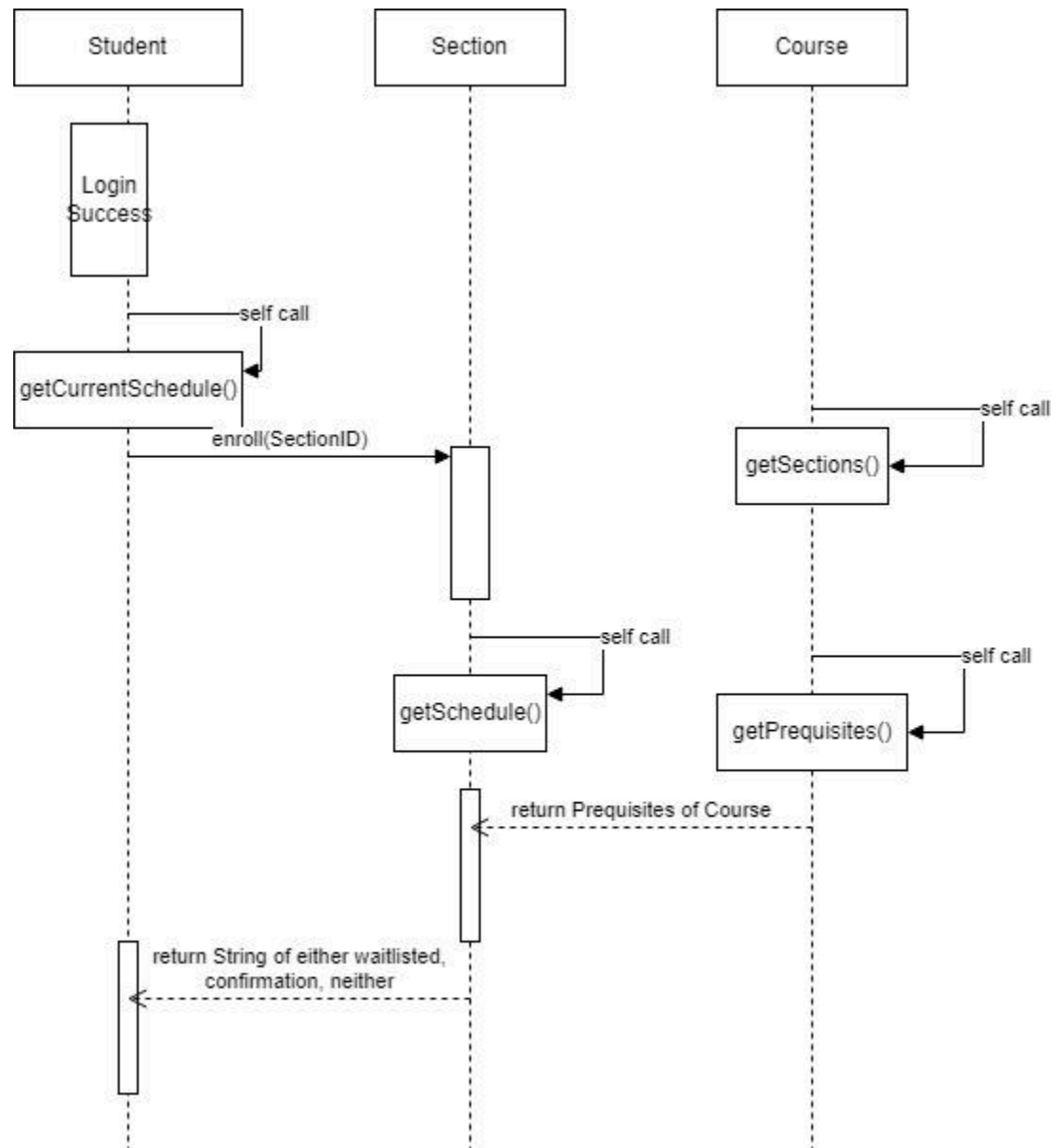
## GUI:



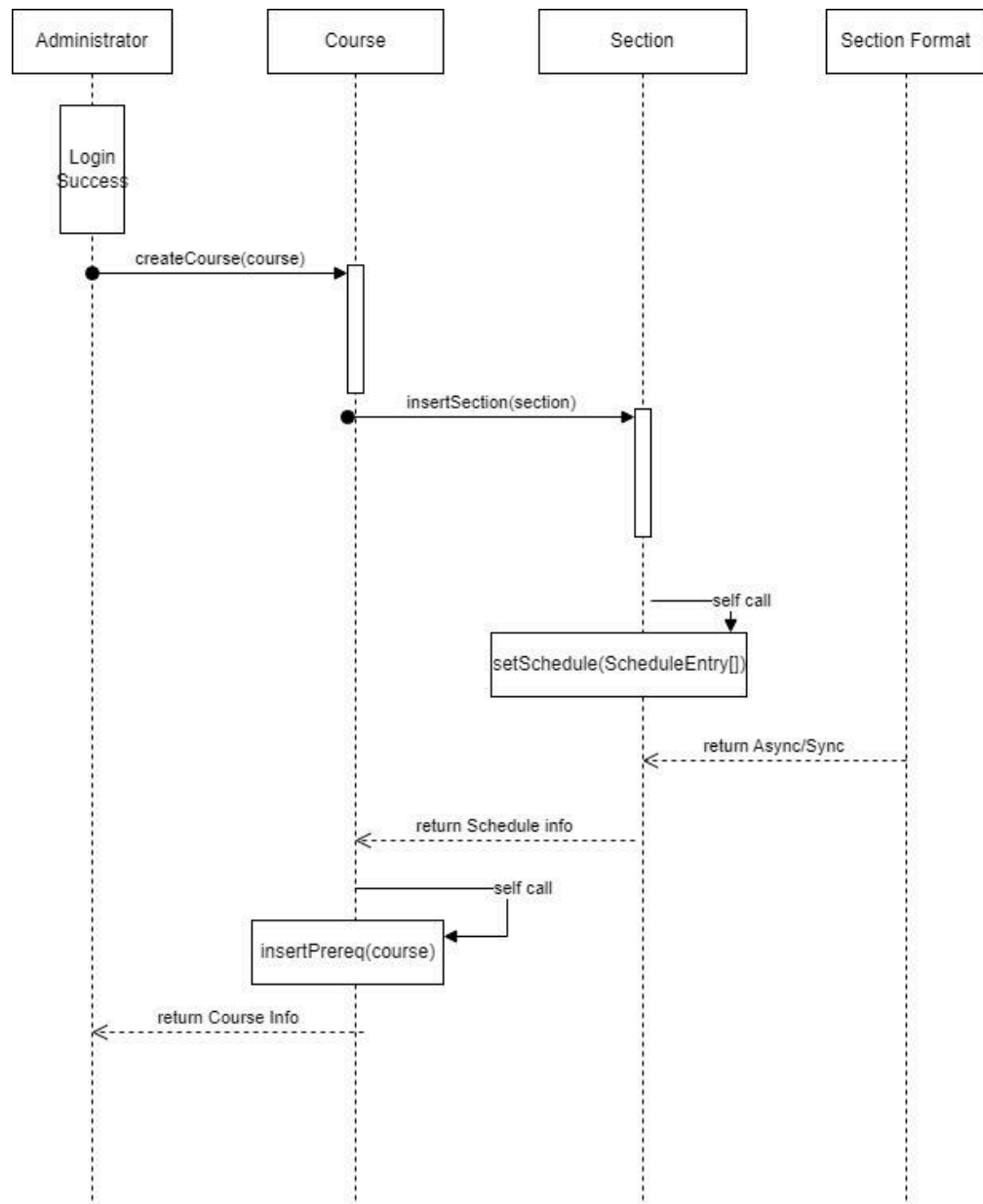


### 18.3. UML Sequence Diagrams

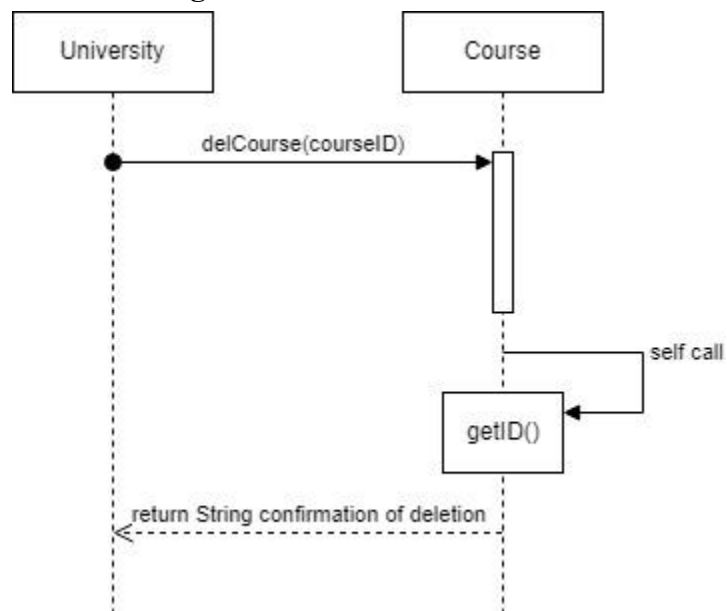
**Case: Student Enrolling in a Section**



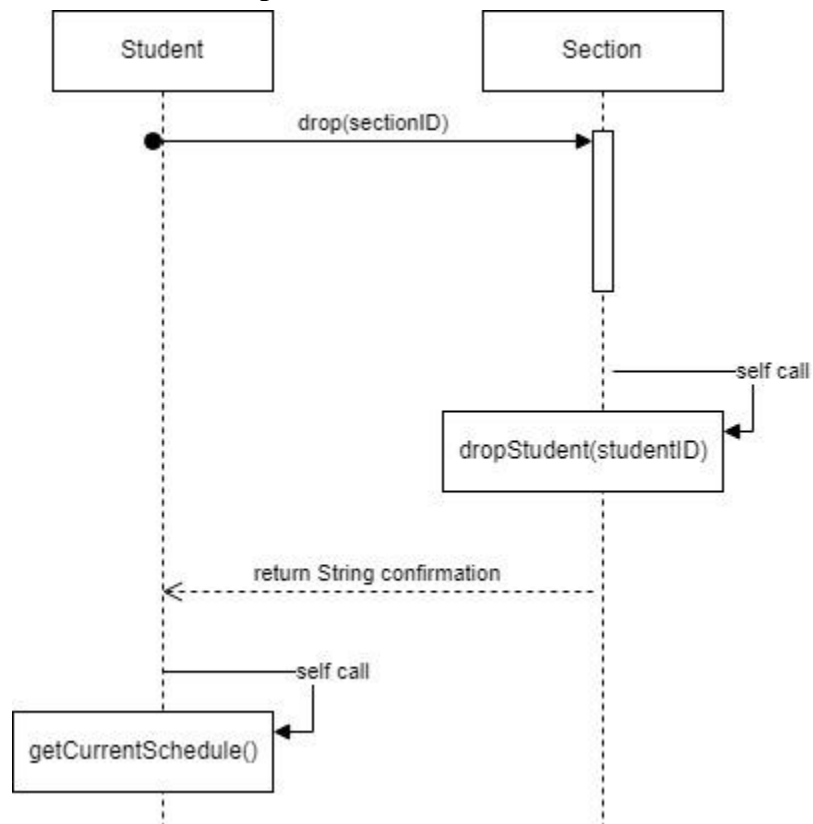
## Case: Administrator Creating a New Course



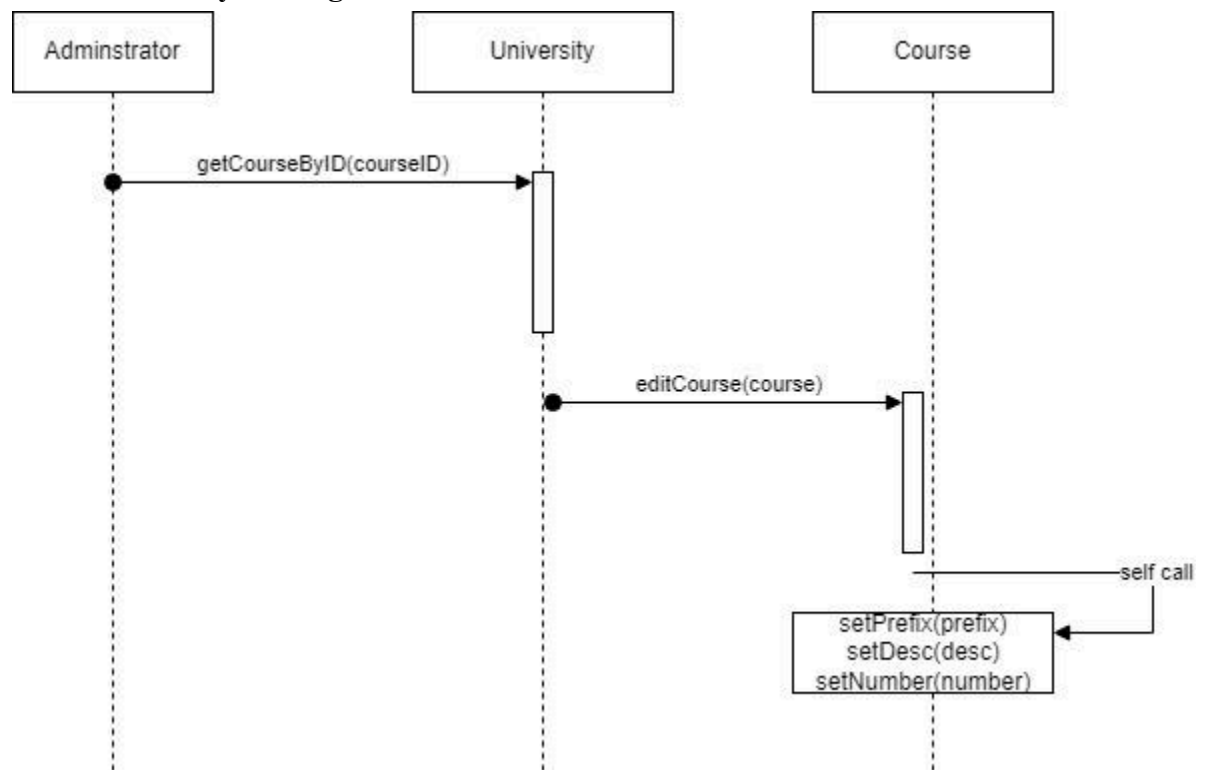
### Case: Deleting a Course



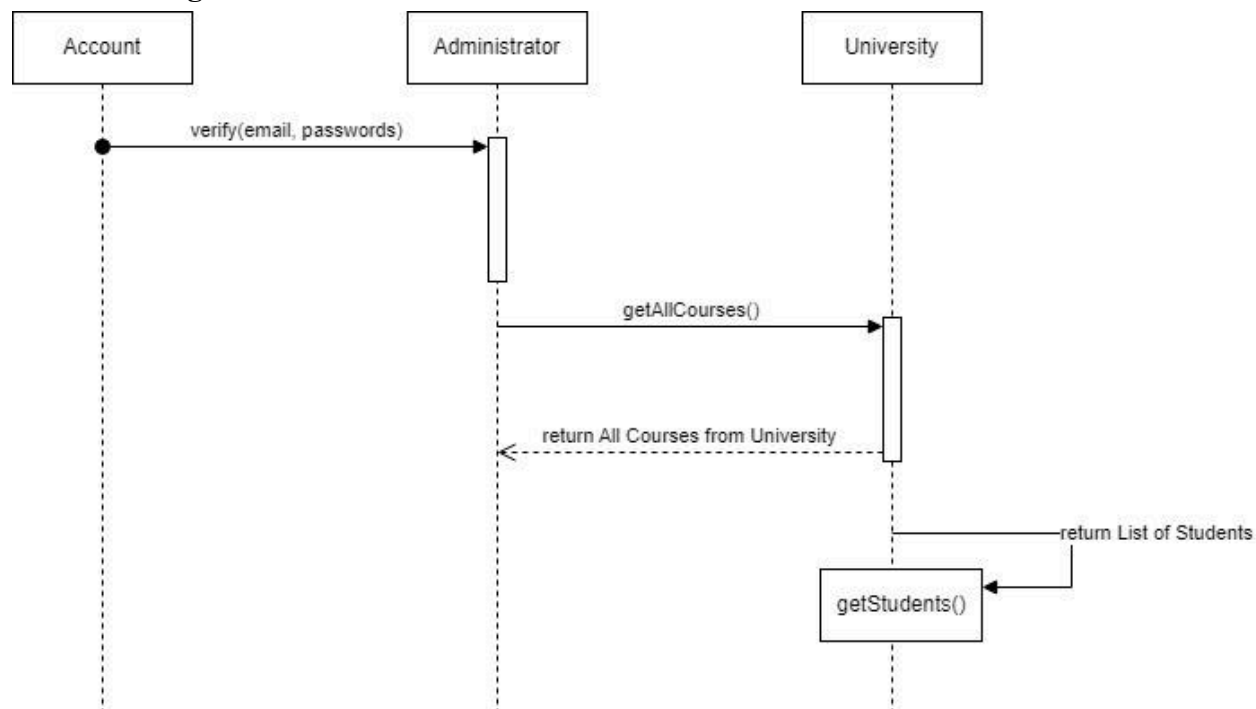
### Case: Student Drops a Section



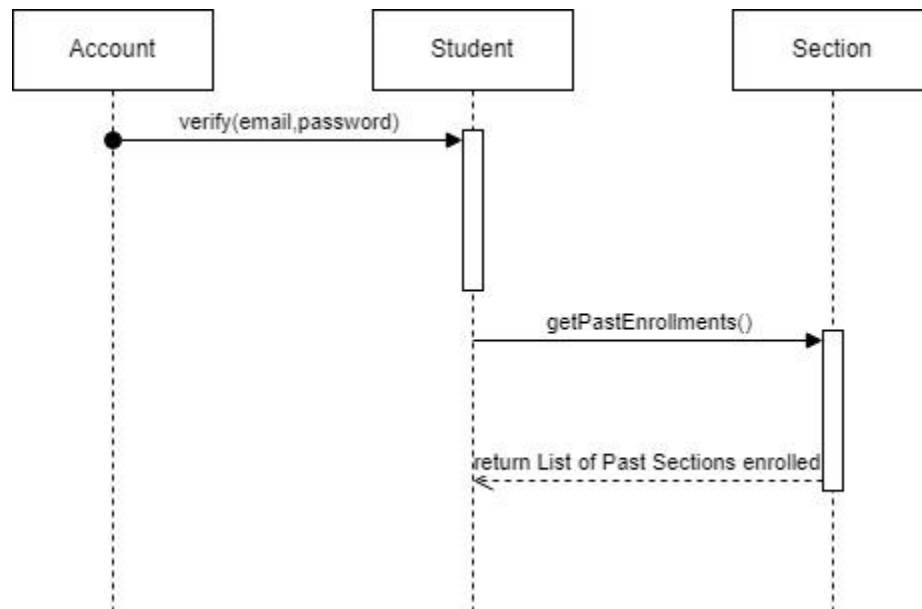
### Case: University Editing Course Information



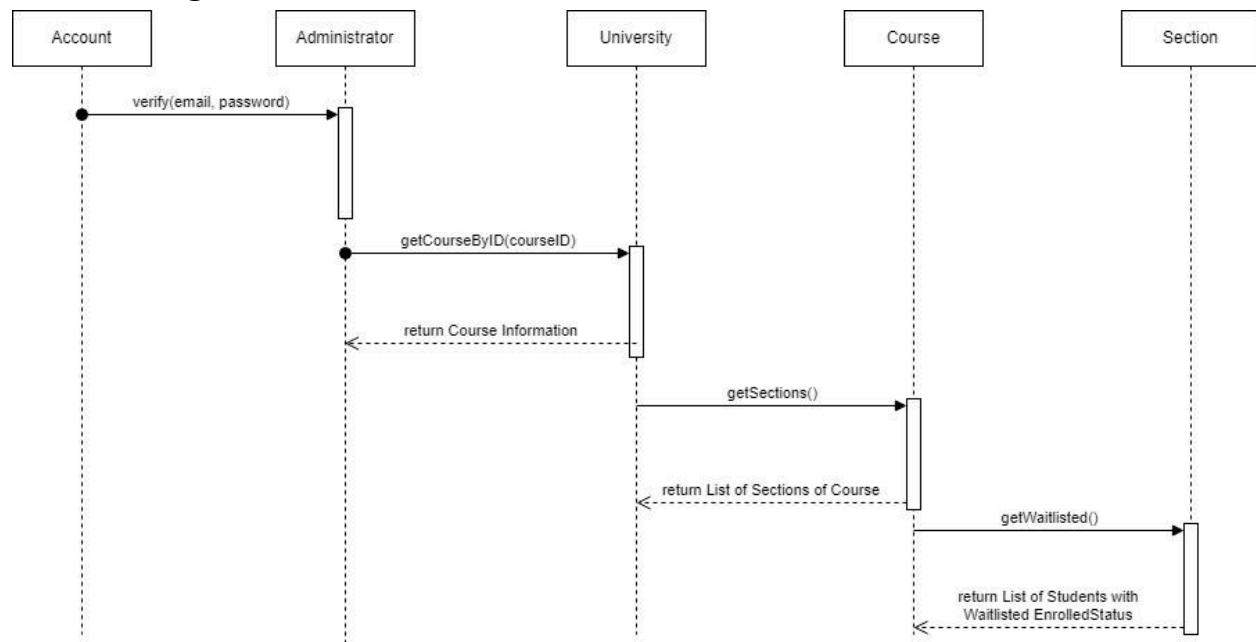
### Case: Viewing the List of Enrolled Students



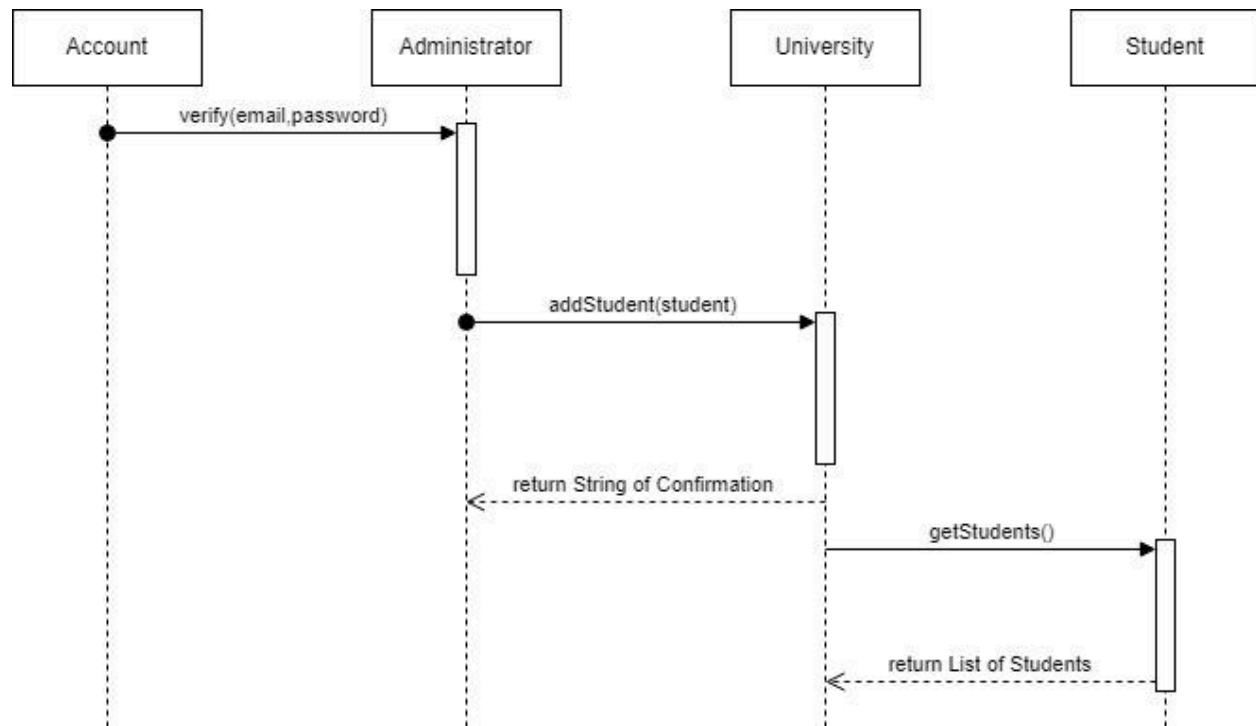
### Case: Students Accessing Past Enrollments



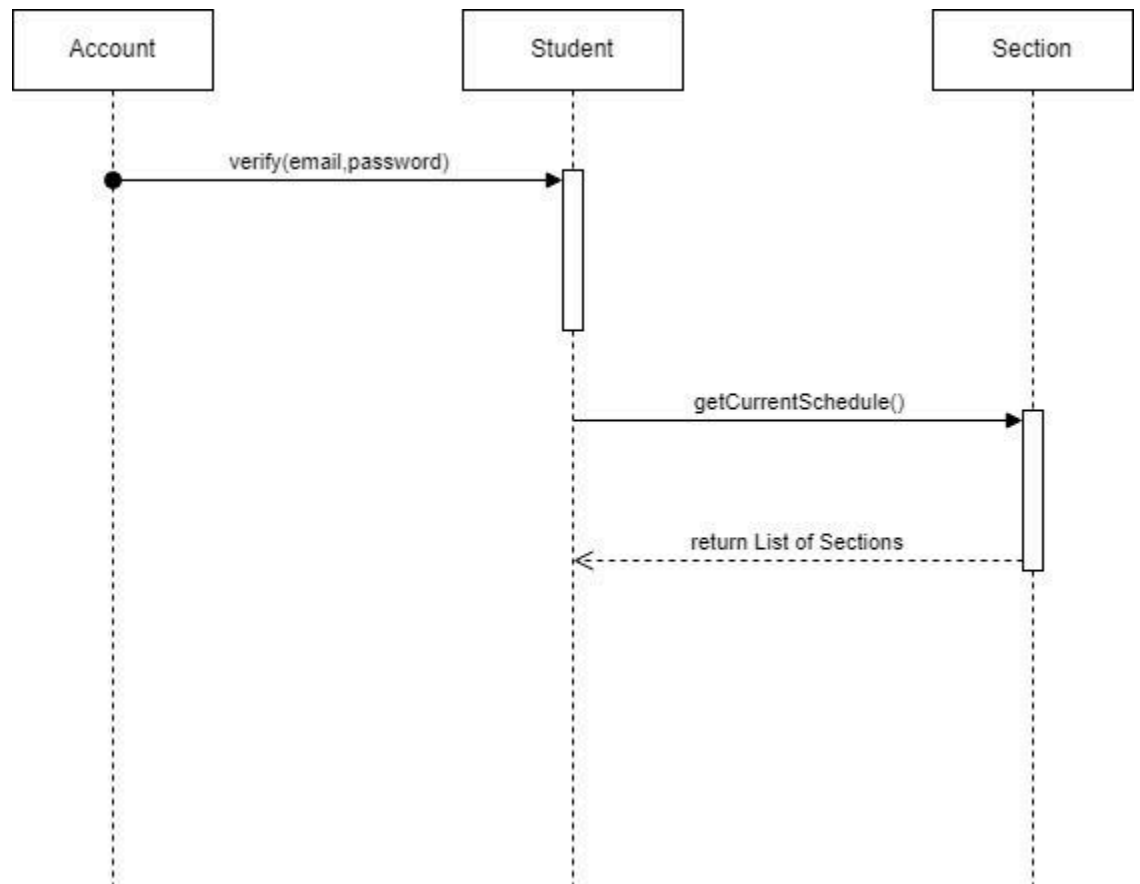
### Case: Viewing the WaitListed Students



### Case: Admin Adding Student to University



### Case: Student Viewing Current Schedule



## 19. Project Organization Content

### 19.1. Calendar Overview

| Meeting | Dates                           | Attendees  | Agenda   |
|---------|---------------------------------|--|--|
| 1.      | October 16th, 2024<br>(Discord) | Roy Alkhoury<br>Anthony Josh Legrama<br>Janelle Kwofie<br>Bach Ngo | <ul style="list-style-type: none"><li>• Discus Phase One feedback and Phase Two division of work</li></ul> |
| 2.      | October 23rd, 2024<br>(Discord) | Roy Alkhoury<br>Anthony Josh Legrama<br>Janelle Kwofie<br>Bach Ngo | <ul style="list-style-type: none"><li>• Discuss drafts for Phase Two</li></ul>                             |
| 3.      | October 28th, 2024<br>(Discord) | Roy Alkhoury<br>Anthony Josh Legrama<br>Janelle Kwofie<br>Bach Ngo | <ul style="list-style-type: none"><li>• Prepare Design Specification for submission.</li></ul>             |

## 19.2. Meeting Minutes

Project Meeting: October 16th, 2024

### NOTES:

- Went over improvements and notes given from SRS and Phase One Presentation
- Discussed the division of work for Phase Two and schedule

### TO DO's:

- Come to next meeting with ideas for how to do the design specification

### USAGE IDEA:

- College Enrollment System with main actors Student and Administrator.

### RESOURCES:

- Google Drive:  CS 401 Project
- Check email for GitHub link.

Project Meeting: October 23rd, 2024

### NOTES:

- Everyone presented their ideas for their part in Phase Two and changes that needed to be made to the SRS

### TO DO's:

- Come to next meeting with final draft of your part

### USAGE IDEA:

- College Enrollment System with main actors Student and Administrator.

### RESOURCES:

- Google Drive:  CS 401 Project
- Check email for GitHub link.

Project Meeting: October 28th, 2024

### NOTES:

- Everyone presented their final parts to be added to the Design Specification

### TO DO's:

- Presentation and Phase Two due on October 30th, 2024

### USAGE IDEA:

- College Enrollment System with main actors Student and Administrator.

### RESOURCES:

- Google Drive:  CS 401 Project
- Check email for GitHub link.



# 19.3. Project Timeline

| WBS NUMBER | TASK TITLE                              | TASK OWNER     | START DATE | DUE DATE | PCT OF TASK COMPLETE | PHASE ONE |   |   |        |   |   |        |   |   |        |   |   | PHASE TWO |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
|------------|---|----------------|------------|----------|----------------------|-----------|---|---|--------|---|---|--------|---|---|--------|---|---|-----------|---|---|--------|---|---|--------|---|---|--------|---|---|---|---|---|
|            |   |                |            |          |                      | WEEK 1    |   |   | WEEK 2 |   |   | WEEK 3 |   |   | WEEK 4 |   |   | WEEK 5    |   |   | WEEK 6 |   |   | WEEK 7 |   |   | WEEK 8 |   |   |   |   |   |
|            |   |                |            |          |                      | M         | T | W | R      | F | M | T      | W | R | F      | M | T | W         | R | F | M      | T | W | R      | F | M | T      | W | R | F | M | T |
| 1          | Phase 1 - SRS Document                  |                |            |          |                      |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.1        | Section 1                               | Janelle        | 9/5/24     | 9/30/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.2        | Section 2                               | Roy            | 9/5/24     | 9/30/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.3        | Section 3 - Functional Requirements     | Bach & Roy     | 9/5/24     | 9/30/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.4        | Section 4 - Non-Functional Requirements | Anthony        | 9/5/24     | 9/30/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.5        | Section 5 - Diagrams                    | Anthony & Bach | 9/5/24     | 9/30/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.6        | Section 6 - Use Case Specifications     | Janelle & Roy  | 9/5/24     | 10/2/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 1.7        | Section 7 & 8 - Meeting Mins & Timeline | Janelle & Bach | 9/5/24     | 10/2/24  | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2          | Phase 2 - Design                        |                |            |          |                      |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2.1        | Course Module UML                       | Bach           | 10/3/24    | 10/30/24 | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2.2        | GUI Design                              | Janelle & Roy  | 10/3/24    | 10/30/24 | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2.3        | GUI (User Module) UML                   | Roy            | 10/3/24    | 10/30/24 | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2.4        | Sequence Diagram                        | Anthony        | 10/3/24    | 10/30/24 | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2.5        | Networking Module UML                   | Bach           | 10/3/24    | 10/30/24 | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 2.6        | Class Descriptions                      | Janelle        | 10/3/24    | 10/30/24 | 100%                 |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |
| 3          | Final Phase - ???                       |                |            |          |                      |           |   |   |        |   |   |        |   |   |        |   |   |           |   |   |        |   |   |        |   |   |        |   |   |   |   |   |

#### 19.4. Project Schedule

| Date                | Description                                    | Assigned Member  | Notes              |
|---------------------|--|--|--------------------|
| October 7th, 2024   | Create Phase Two Documents                     | Janelle  | Check Google Drive |
| October 7th, 2024   | Finalize Individual Work                       | All  |                    |
| October 30th, 2024  | Presentation: Phase Two                        | All  |                    |
| October 30th, 2024  | Turn in Phase Two                              | Bach   |                    |
| November 11th, 2024 | Implementation Drafts with matching test cases | Bach - Server and Messenger<br>Roy - GUI<br>Anthony - Actor Classes<br>Janelle - Non Actor Classes |                    |
| November 18th, 2024 | Running test cases for combined program        | All  |                    |
| December 2nd, 2024  | Presentation and Project Due                   | All  |                    |