# Registration Service

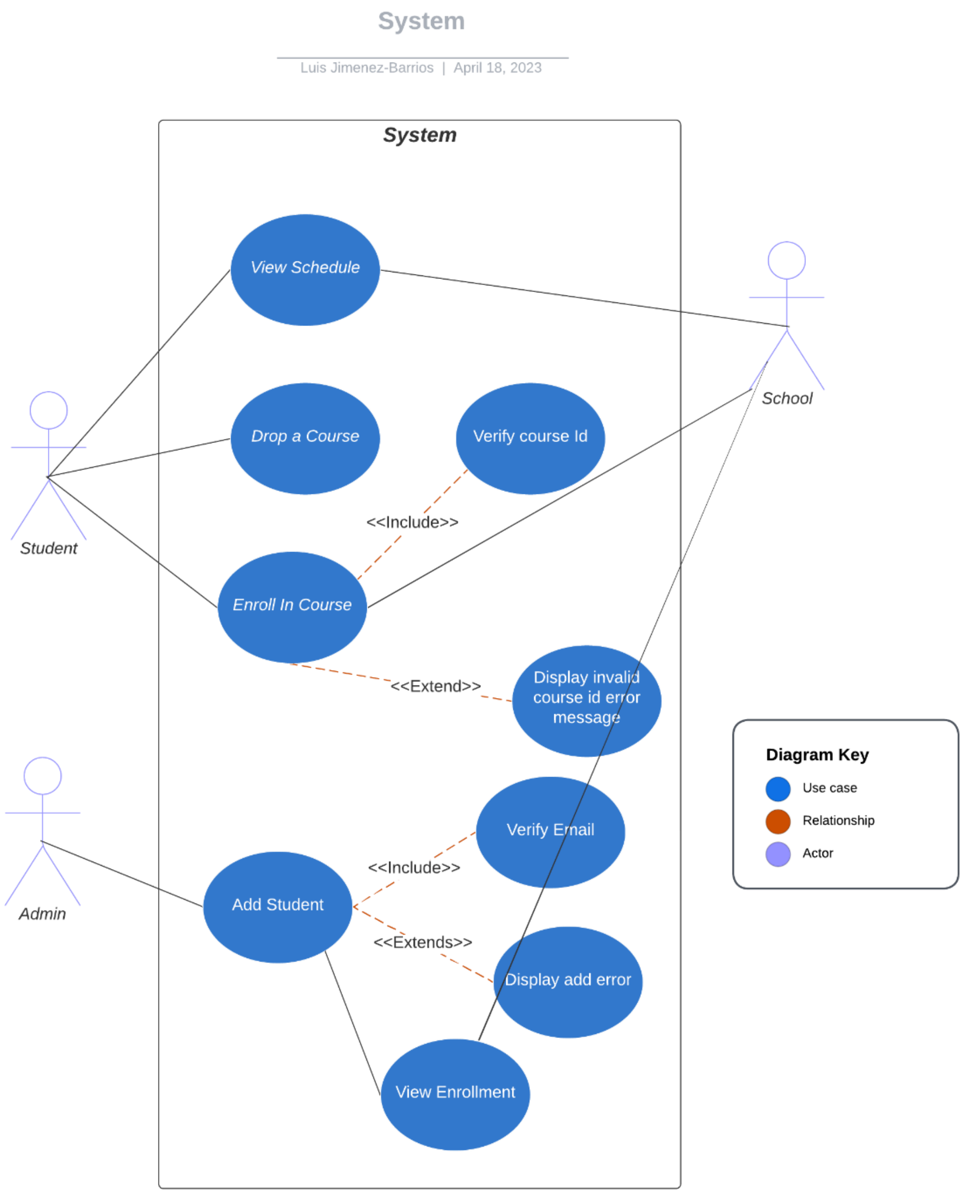
Luis Jimenez

Partner: Warren Ngoun

**CST 438 – Software Engineering**

**Version: 1** **Date: (04/18/2023)**

2.1 System Environment showing UML use case diagrams for your service.



3.2 Functional Requirements

3.2.1 Add Student

| | |
|---|---|
| **Use Case Name** | Add Student |
| **XRef** | |
| **Trigger** | The Administrator selects to add a new student to the database |
| **Precondition** | The Editor has accessed the Course Registration main screen. |
| **Basic Path** | 1. The system presents empty fields to enter the student information<br>2. The Administrator enters the student's information and submits the form<br>3. The system checks that the email address and name fields are not blank, and that the email entered doesn't exist in the database. Then updates the database |
| **Alternative Paths** | If in step 3, the email is already registered, the administrator receives a message stating "Student NOT added" and is expected to enter an unregistered email |
| **Postcondition** | The student has been added to the database |
| **Exception Paths** | The Administrator may abandon the operation at any time. |
| **Other** | The student's information includes the student's name, email address, hold status, and hold status code. |

3.2.2 Student views their schedule

| | |
|---|---|
| **Use Case Name** | Get Schedule |
| **XRef** | |
| **Trigger** | The student selects to view their schedule |
| **Precondition** | The Editor has accessed the registration service main screen. |
| **Basic Path** | 1. The system retrieves and displays the student’s schedule for the selected term. |
| **Alternative Paths** | |
| **Postcondition** | The schedule for the term is displayed to the student. |
| **Exception Paths** | The student may abandon the operation at any time. |
| **Other** | |

### 3.2.3 A student enrolls in a course

| | |
|---|---|
| **Use Case Name** | Add course |
| **XRef** | |
| **Trigger** | The student selects to add a new course to their schedule. |
| **Precondition** | The student has accessed the registration service main screen and selected a term. |
| **Basic Path** | 1. The system presents a form to enter the course id which is to be added. |

| | |
|---|---|
| | 2. The system updates the schedule for the student's schedule. |
| **Alternative Paths** | If in step 1, if the course id does not exist then the user receives a message such as "Course id invalid or student not allowed to register for the course" |
| **Postcondition** | The Course has been added to the student's schedule |
| **Exception Paths** | The student may abandon the operation at any time. |
| **Other** | |

3.2.4 A student drops a course.

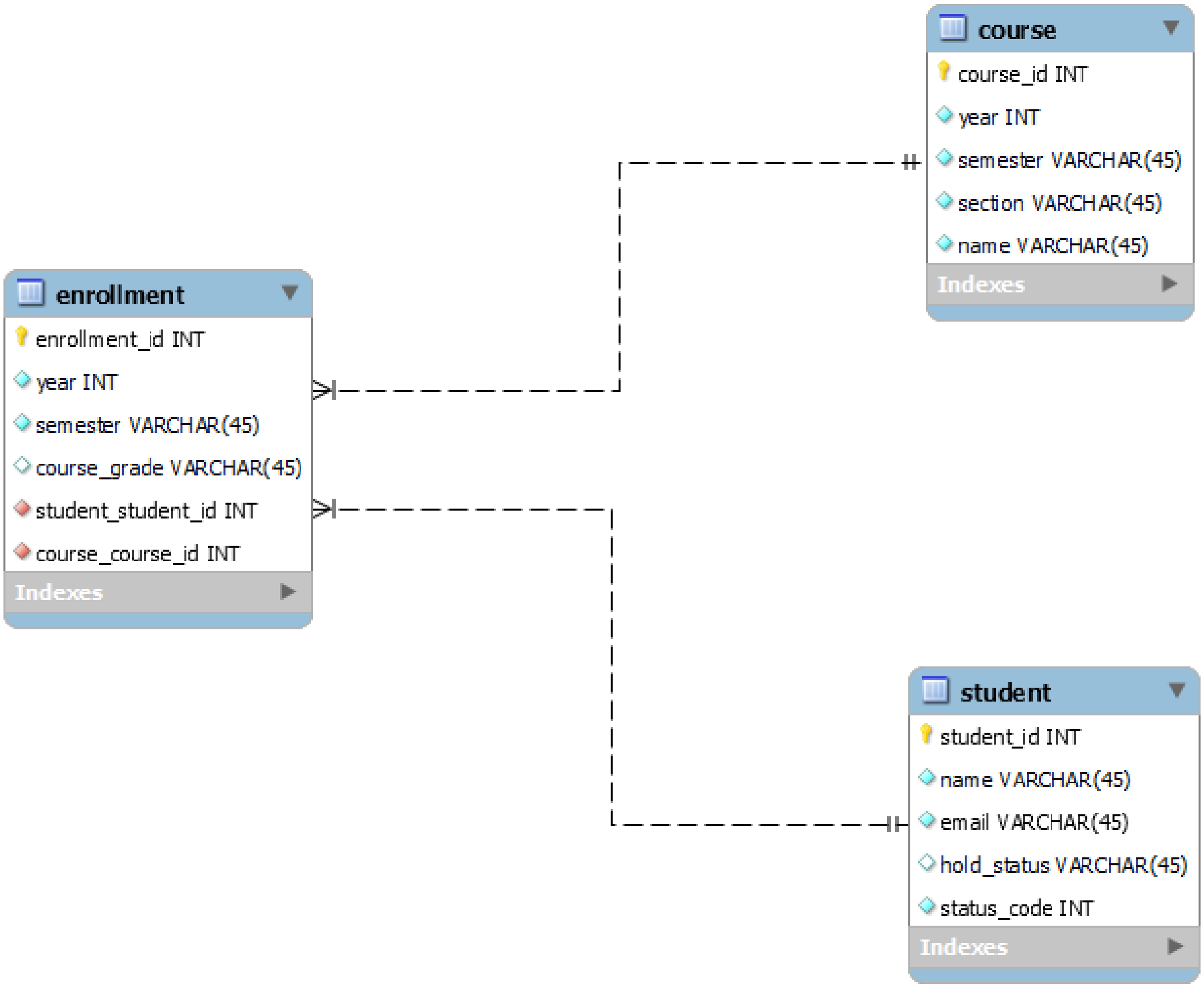
| | |
|---|---|
| **Use Case Name** | Drop course |
| **XRef** | |
| **Trigger** | The student enters the course id for the course that is to be dropped. |
| **Precondition** | The Editor has accessed the registration service main screen. |
| **Basic Path** | 1. The system requests the course id of the course.<br>2. The system updates the database which drops the course from the student's schedule. |
| **Alternative Paths** | If in step 1, the course id is blank or invalid then the user receives a message such as "Course id is invalid" |
| **Postcondition** | The course is dropped from the student's schedule |
| **Exception Paths** | The student may abandon the operation at any time. |

| | |
|---|---|
| Other | |

3.3 Briefly (3-sentences) discuss non-functional requirements.

The service should use an infrastructure that can maintain high-performance, ensuring that the data displayed is always up to date and can handle multiple requests. Security measures should be implemented in order to protect sensitive information of the students or administrators such as OAuth 2 authentication. The service should be compatible with modern browsers which should provide a responsive and seamless user experience.

3.4 Logical database diagram and accompanying details on tables, keys and attributes



The registration service is designed to allow students to register and manage their course schedule for the appropriate semester and term. Along with allowing admins to register new students and manage their schedules. This structure allows for efficient querying and updating records related to a student’s class schedule making it an easy and efficient way for managing a school’s registration service.