**Software Requirements and Design Document**

**For**

**Group R1**

Version 1.0

**Authors**:

Aaron Gallon

Corso Montuori

Jeremiah Daniels

Noelle Hartman

# Overview (5 points)

The system we are creating resembles that of a Canvas page. It is to be used by both Teachers and students. Features currently implemented include: enrolling and updating a student, searching for students, creating and updating a course, and listing students and courses. A grouping function is in the works that will automatically group students into random groups when selected. After implementation, there will be a feature to show the groups created. The system can take in data, store it, and illustrate it to the user. Eventually, the system will be implemented with Blazor to provide an aesthetic user interface.

# Functional Requirements (10 points)

1. The system shall take user input to select a menu option (high).
2. The system shall re-present menu options when user input is invalid (low).
3. Upon adding a student, prompts will arise to gather data (med).
4. The system shall list students by ID, followed by name then classification(med).
5. Each student search will query based off name(high).
6. Upon adding a course, prompts will arise to gather data(med).
7. The system shall list courses by code, name, then description(med).
8. Each course search will query based off course code(high).

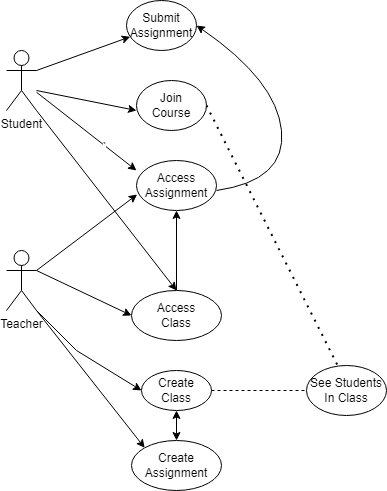
# Non-functional Requirements (10 points)

1. Duplicate students and courses should not be allowed.
2. Classification options for students are not case sensitive.
3. Results should be presented within 2 seconds of completion.

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



# Class Diagram and/or Sequence Diagrams (15 points)

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

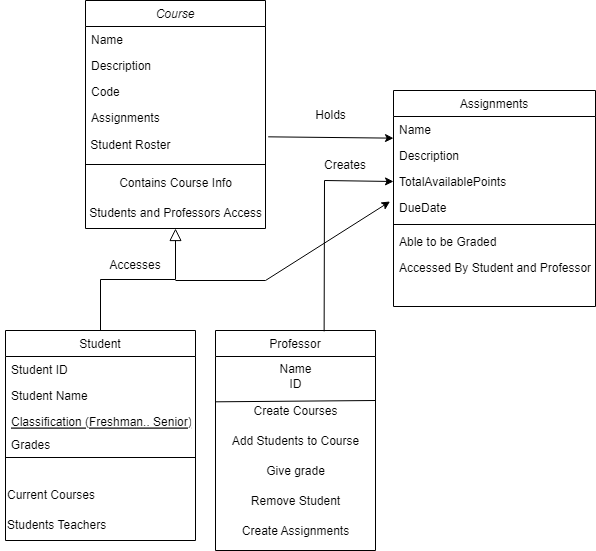
*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

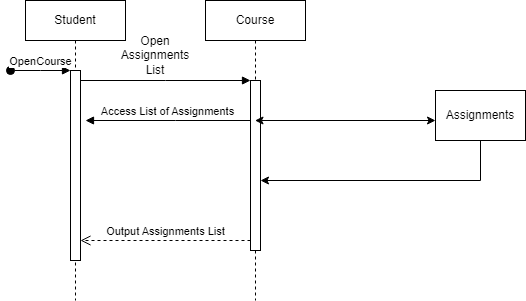
***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

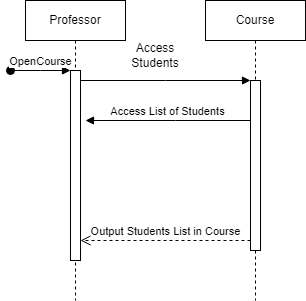
***Class Diagram***



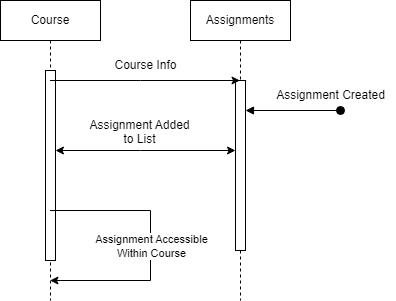
***Student Course Sequence***



***Professor Course Sequence***



***Course Assignments Sequence***



# Operating Environment (5 points)

*Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.*

The operating environment will include using a local machine while the software is utilized on a web-based application. The software will operate with the use of .NET and will be compliant with the Common Language Interface allowing C# to have a high-level syntax.

# Assumptions and Dependencies (5 points)

It is assumed that there will be no duplicate courses, each with their own course ID. Students may share the same name but will have different student IDs. Backend components have been reused and modified from the Full Stack Application’s Project. Features such as add a student and a course is derived from there. This system is dependent on Blazor for user interface and .NET for backend development.