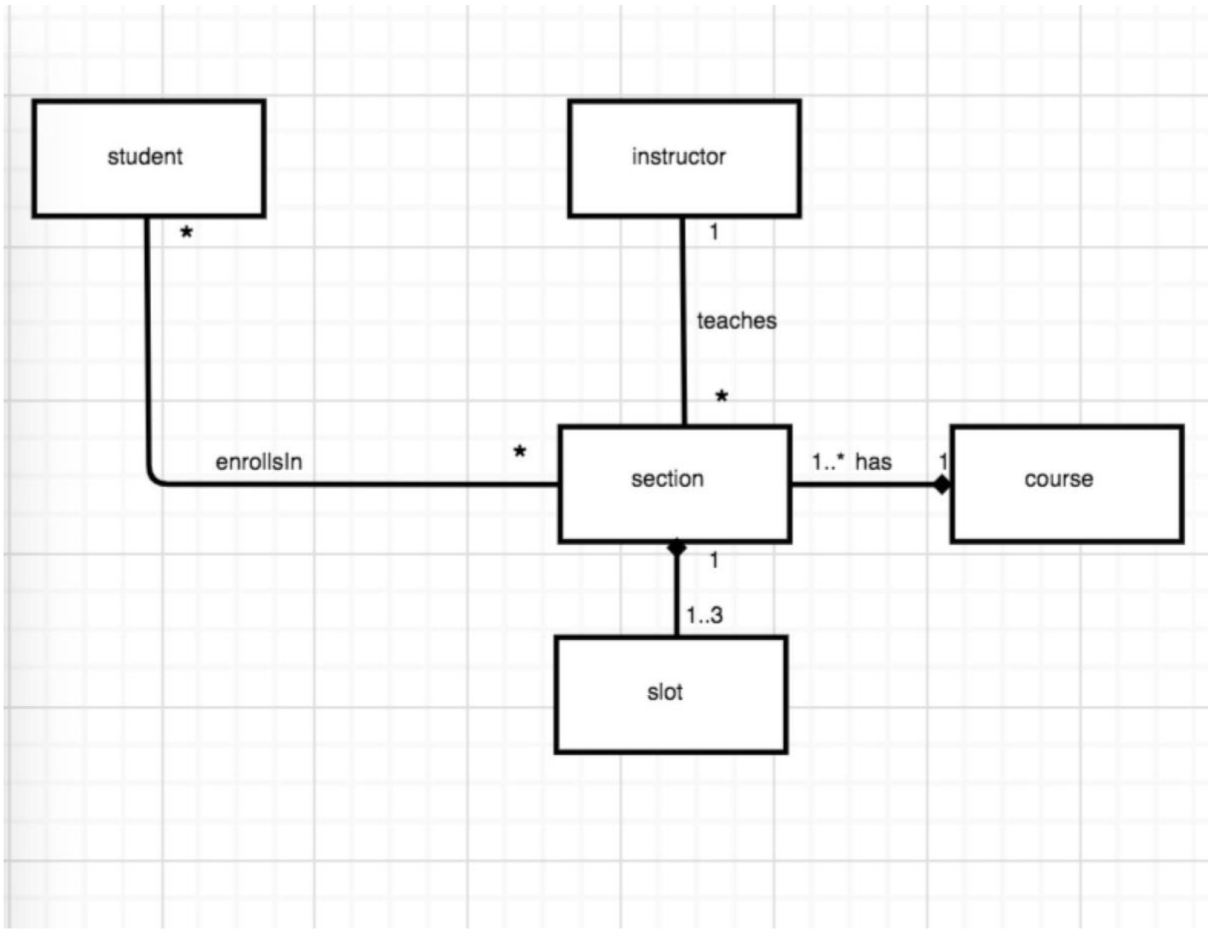
Team Name: 😂

Project Github: https://github.com/Jeff-HOU/COMP3111

COMP3111 Activity 1
Leader: HOU, Kaijun
Member1: ZHANG, xxx
Member2: XIA, xxx

Model:



```
Student
-------------------------------------------------------------
-enrolledCourses: Course[]
-------------------------------------------------------------
+SearchAllSubject()
+searchCourse(baseUrl: String, term: String, subject: String)
+filterCourse(checkbox: Boolean[])
+enrollInSection(courseCode: String, sectionCode: String)
+dropSection(courseCode: String, sectionCode: String)
+getEnrolledCourse(): Course[]
+searchForSFQ(courseCode: String, insturctorName: String, url: String)
```

```
Instructor
-------------------------------------------------------------
-instructorName: String
-teachingSection: Section[]
-------------------------------------------------------------
+getName(): String
+getSection(): Section[]
```

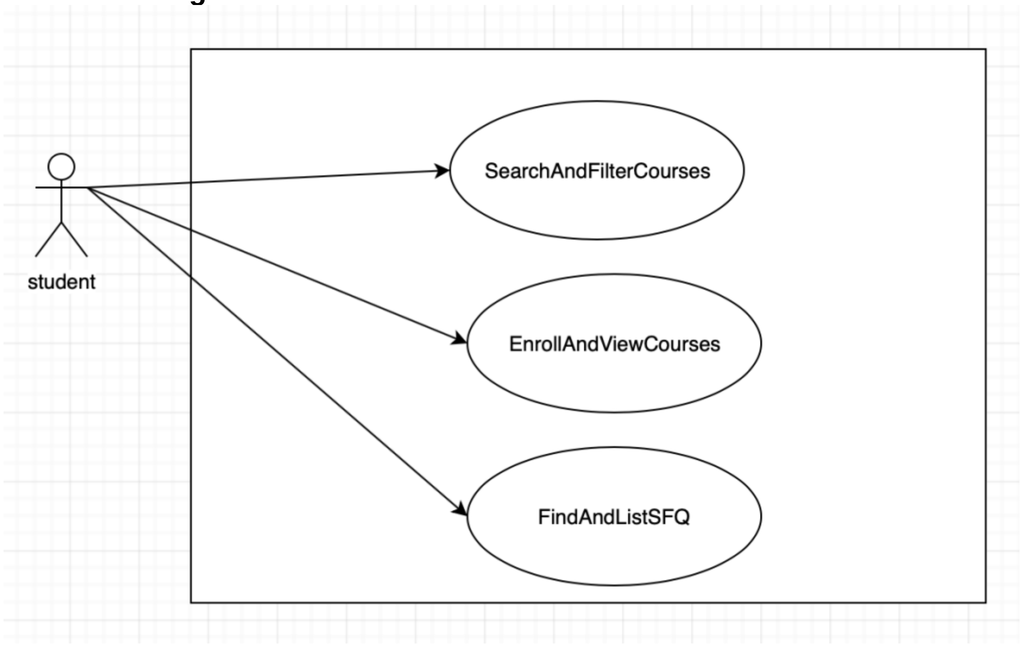
```
Course
-------------------------------------------------------------
-subject: String
-courseCode: String
-courseColor: String
-exlusiveCourse: Course[]
-labsAndTutorial: Section[]
-offeringSections: Section[]
-------------------------------------------------------------
+ifBelongsToCommonCourse(courseCode: String): Boolean
+getColor(): String
+setColor(color: String)
```

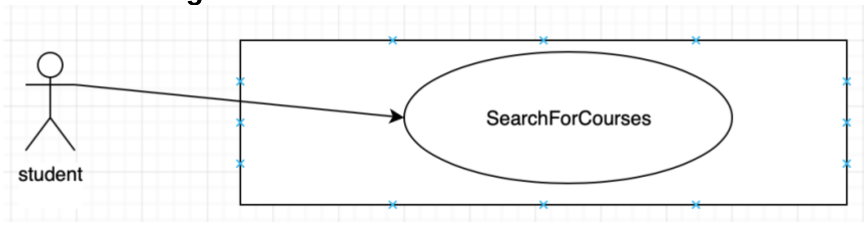
```
Section
-------------------------------------------------------------
-subject:String
-courseCode: String
-sectionCode: String
-Term: String
-offeringSlots: Slot[]
-instructor: Instructor
-------------------------------------------------------------
+getCourseCode(): String
+getSectionCode(): String
+getSlot(): Slot
+getInstructor(): Instructor
+getTerm(): String
```

```
Slot
-------------------------------------------------------------
-courseCode: String
-sectionCode: String
```

-venue: String -time: String
+getVenue(): String +gettime(): Date

Use Case Diagram:



Specifications:
Task 1:
Use Case: searchFor courses
Brief Description
This use case describes how a student searches for courses.
Use-case Diagram


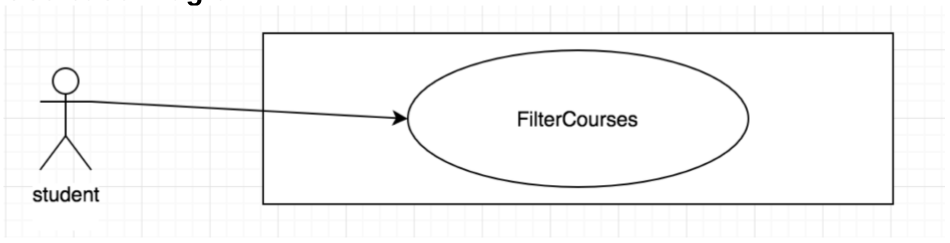
Basic Flow
1. The use case begins when the student actor begins to search for courses in the "Main" page.
{Enter Base Url}
2. The user enters the base url it wants to search courses on.
{Enter Term}
3. The user enters the term it wants to search courses on.
{Enter Subject}
4. The user enters the subject it wants to search courses that belong to.
{Search for Courses}
5. The user clicks the "Search" button. The system will display the following information in the console: all courses that satisfies the term and subject specified by the user on the base url with their course code, course title, number of units, slots, number of enrollment and sections; Total Number of different sections in this search: NUMBER_OF_SECTIONS; Total Number of Course in this search: NUMBER_OF_COURSES; Instructors who has teaching assignment this term but does not need to teach at Tu 3:10pm: INSTRUCTOR_NAME1, INSTRUCTOR_NAME2, INSTRUCTOR_NAME3,... where the names are sorted by ascending alphabetical order and displayed as "LAST_NAME, First_Name"
6. The use case ends

Alternative flows
A1: Invalid Base Url
At {Search for Courses} if the entered Base Url is invalid.
1. The system displays "404 not found" on the screen.
2. The flow of events is resumed at {Base Url}

A2: Invalid Term
At {Search for Courses} if the entered Term is invalid.
1. The system displays "Term entered is invalid" on the screen.
2. The flow of events is resumed at {Enter Term}

A3: Invalid Subject
At {Search for Courses} if the entered Subject is invalid.
1. The system displays "Subject entered is invalid" on the screen.
2. The flow of events is resumed at {Enter Subject}

Task 2:
Use Case: Filter Courses
Brief Description:
This use case describes how a student select attributes to filter the courses
Use-case Diagram



Preconditions
The student entered Base URL, Terms and Subject on the main tab
Basic Flow:

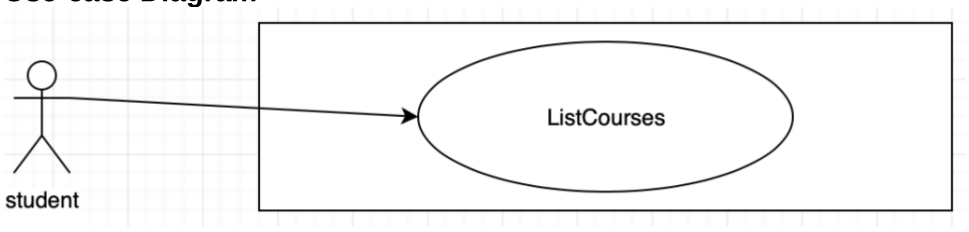
1. The use case begins when the student choose to filter a course
2. The system displays the interface for selecting attribute to filter and the system displays courses information on the console

{Select attributes}

3. If the student has an attribute to be used for filtering courses
    - 3.1 If the student clicked Select All.
        - 3.1.1 The system checked All boxes on this tab.
        - 3.1.2 The system change the text of that button to De-select All
            - 3.1.2.1 If the student clicked De-select All.
                - 3.1.2.1.1 The system changes the text of that button to Select All
                - 3.1.2.1.2 The system uncheck All boxes on this tab.
    - 3.2 While the status of any box on this tab is changed (checked or unchecked)
      {Change attributes}
        - 3.2.1 If any box(es) among AM or/and PM is checked
          {Check AM or/and PM}
            - 3.2.1.1 If the student checks one of the boxes between AM or PM
                - 3.2.1.1.1 The system display only all sections of the courses which has a slot in AM (or in PM).
            - 3.2.1.2 if the student checks both AM and PM
                - 3.2.1.2.1 The system display only all sections of the courses that has a slot that starts at AM and ends at PM, or a section has a slot in AM and another slot in PM
        - 3.2.2 If any boxes of days of the week (Monday, Tuesday, ...) is checked
            - 3.2.2.1 The system display only all sections of the courses that has slots on the selected boxes.
        - 3.2.3 If Common Core is checked
            - 3.2.3.1 The system display only all sections of the courses that are 4Y CC.
        - 3.2.4 If No Exclusion is checked
            - 3.2.4.1 The system display only all sections of the courses that does not define exclusion.
        - 3.2.5 If With Labs or Tutorial is checked
            - 3.2.5.1 The system display only all sections of the courses that has labs or tutorials.
4. The use case ends

Alternative flows
A1: No box checked
At {Change attributes} if no box is checked in the given interface
1. The system display all courses
2. The flow of events is resumed at {Select attributes}

Task 3:
Use Case: list courses

Brief Description
This use case lists the courses filtered out in "filter course".
Use-case Diagram


Basic Flow
1. The use case begins when the student actor updates checkbox in "filter" page.
{Update table}
2. The table is updated when the any checkbox in "filter" page is modified. The table has columns of Course Code, Lecture Section, Course Name, Instructor and Enroll. There will be a checkbox in each row in the Enroll column. The checkbox shows as checked if the student has indicated its enrollment, or unchecked otherwise.
3. The use case ends.

Alternative Flow
A1: No course is filtered
If there is no courses to select from after some checkbox is updated in "filter" page.
1. The system displays "No content in table".
2. The flow of events is resumed at {Update table}

A2: Select course
At any point in the whole flow:
{Select Course}

```
1. While there is a course to select
    1.1 If the course is selected
        {Begin enrolling course}
        1.1.1 The checkbox is checked.
        1.1.2 The system updates the enrollment status stored somewhere.
        1.1.3 The system updates the console “The following sections are enrolled:”
        {End enrolling course}
    1.2 If the course is unselected
        {Begin dropping course}
        1.2.1 The checkbox is unchecked.
        1.2.2 The system updates the enrollment status stored somewhere.
        1.2.3 The system updates the console  “The following sections are enrolled:”
        {End dropping course}
```

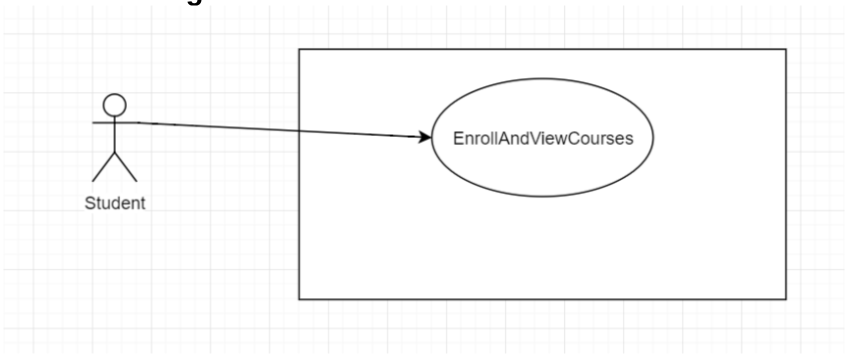
Task 4

Use Case: EnrollAndView courses

Brief Description

This use case describes how a student enrolls in courses and view the courses he/she enrolled in a time table.

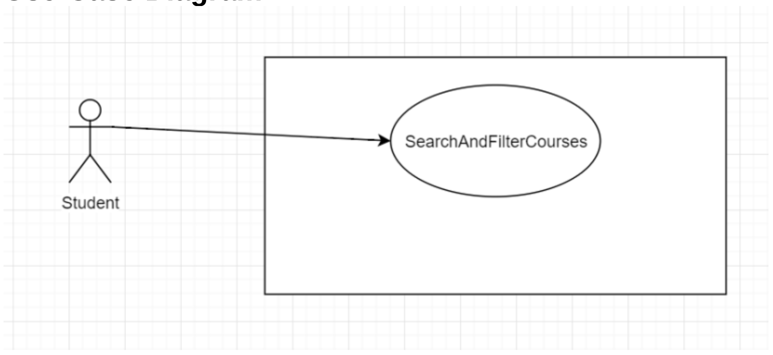
Use-case Diagram



Basic Flow

```
1. The use case begins when the student actor chooses to enrolls in the courses he wants to enroll in.
2. The system displays the interface for enrollment.
{enrolls in}
3. While the student tick the box of a course
        3.1 enrolls in the course
        3.2 if the course has not been enrolled
                {update time table}
                3.2.1  assign a new color for this course
                3.2.2  if the time slots have conflict with other enrolled courses
                        3.2.2.1 store the conflict time period with a mixing of the two colors used by the two
                        conflicting courses
                3.2.3 store the color and corresponding time in the time table
        3.3 if the course has been enrolled
                {update time table}
                3.3.1 use the color for this course and store the new slots in the time table
{view the time table}
4. while the student click the time table button to view the time table
        {display time table}
        4.1 display the updated time table according to the data stored in it
5. the use case ends
```

Task 5
Use Case: SearchAndFiltercourses
Brief Description
This use case describes how a student search for all subjects.

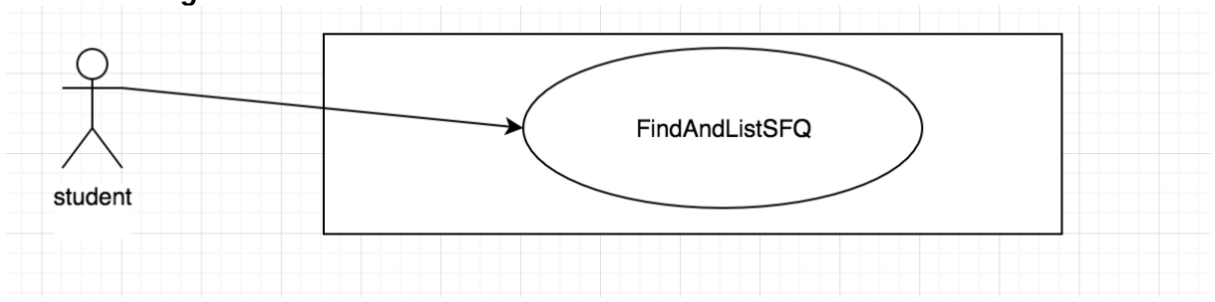
Use-Case Diagram


Basic Flow

```
1. The use case begin when the user chooses to search for all subject.
2. the system display the interface of main
{search}
3. while the student click search
    3.1 if the subject field is empty
                {display the search result}
                3.1.1 display  Total Number of Categories/Code Prefix: ALL_SUBJECT_COUNT
    3.2 if the subject field is not empty
    {perform specific subject search}
4. the system displays the interface of allSubjectSearch
    {allSubjectSearch}
    4.1 while the student click allSubjectSearch
        4.1.1 display Total Number of Categories/Code Prefix: ALL_SUBJECT_COUNT
        4.1.2 while there are subjects left
                {while there are subjects left}
                4.1.2.1 search for all courses
                4.1.2.2 display all the courses information under this subject
                4.1.2.3 if one subject is done
                    4.1.2.3.1 the system print “SUBJECT IS DONE” on the console
                    4.1.2.3.2 update the progress bar by the fraction 1/ALL_SUBJECT_COUNT
        4.1.3 when all subjects are done
                4.1.3.1 display Total Number of Courses Fetched: TOTAL_NUMBER_OF_COURSES
5. the use case ends
```

Alternative flows
A1: No Term is inputed
At {Search} if there is no term inputed.
1. The field displays "e.g 1830(19: yr 18-19; 30: Spring)".
2. The flow of events is resumed at {search}

Task 6
Use Case: FindAndListSFQ
Brief Description:
This use case describes how a student search for SFQ
Use-case Diagram



Basic Flow:

1. The use case begins when the student chooses to find the SFQ
2. The system displays the interface for enrolled courses or instructors

{Set the interface}

3. The system disable Find SFQ with my enrolled courses
4. If search or All Subject search has been clicked
   4.1 the system make Find SFQ with my enrolled courses enabled

{Enter URL}

5. The student enter the URL for the SFQ website

{Search for SFQ for courses}

6. While the student wants to get SFQ
   {Grab Data from URL}
   6.1 If the student clicked Find SFQ with my enrolled courses
      {Show Courses' SFQ}
      6.1.1 Print unadjusted SFQ data of enrolled courses on console
   6.2 If the student clicked List instructors' average SFQ
      {Show Instructors' SFQ}
      6.2.1 The system print all instructors' name and their unadjusted SFQ score on console
7. The use case ends

Alternative flows
A1: Invalid URL
At {Grab Data from URL} if the entered URL is invaild
1. The system informs the student that the URL is invalid
2. The flow of events is resumed at {Enter URL}

A2: Multiple Sections for Enrolled Course
At {Grab Data from URL} if multiple section are available for a course
1.The system take the average unadjusted SFQ data and print it with the other enrolled courses' SFQ on console
2. The flow of events is resumed at {Search for SFQ for courses}

A3: Multiple Sections for instructor(s)
At {Grab Data from URL} if multiple sections' SFQ are available for instructor(s)
1.The system add all unadjusted SFQ score of the sections taught by him/her and divided by number of sections for each of those instructors and save the results
2. The system print the name and calculated results of those instructors and print all other instructors' name and their unadjusted SFQ score on console.
3. The flow of events is resumed at {Search for SFQ for courses}

A4: Enrollment not implemented
At{Show Courses' SFQ} if enrollment is not correctly implemented
1. The system enrolls the first five sections of the scraped data (all sections if the total number of section is less than 5)
2. The flow is resumed at {Search for SFQ for courses}