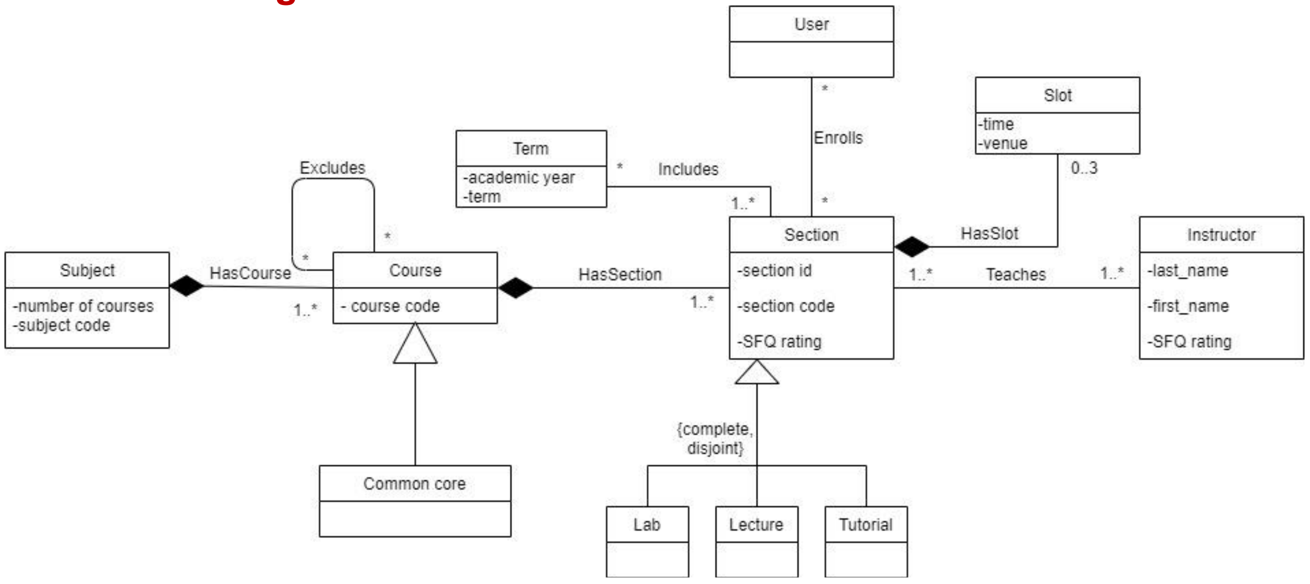
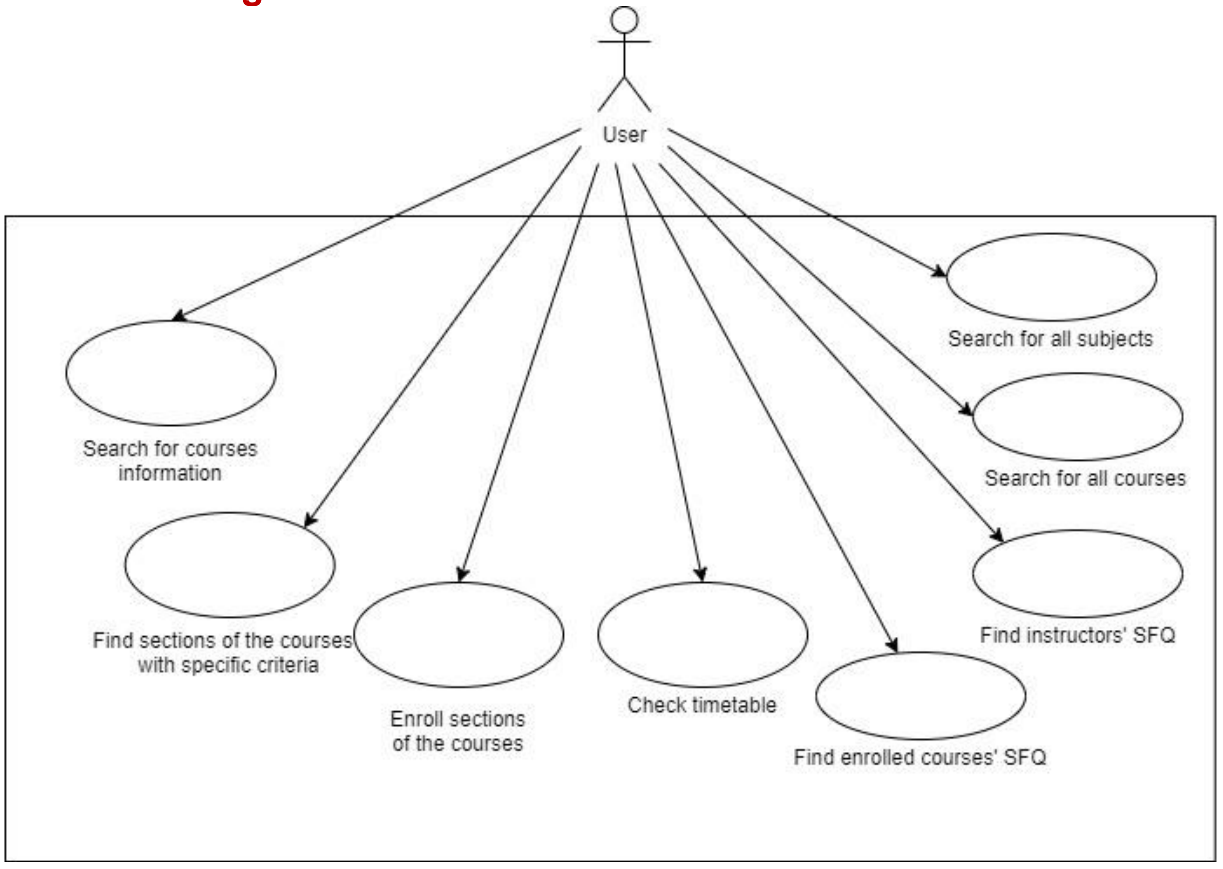
# ***Group 22 Activity 1 – System Requirement Specification***

(Yuen Zhikun, Xie Yulong, Lam Tsz Chun)

## Data Model Diagram



## Use-case Diagram


User
Search for all subjects
Search for courses information
Search for all courses
Find sections of the courses with specific criteria
Find instructors' SFQ
Enroll sections of the courses
Check timetable
Find enrolled courses' SFQ

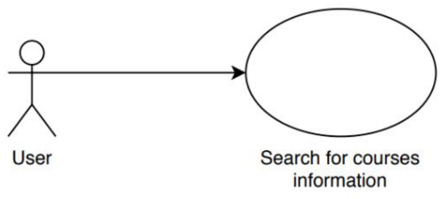
# Use-case Detailed Specification

## Use Case: Search for courses information

### Brief Description

This use case describes how a user searches for information of all courses from the selected subject.

### Use-case Diagram



### Basic Flow

1. The use case begins when the user goes into the Main Tab.
2. The system displays the interface for searching for the courses.
3. The system cleans up the console.

**{Enter URL Information}**

4. The user indicates the base URL of the "Class Schedule & Quota" website, the term and subject of the courses he wants to search for.
5. The user clicks the "Search" button.

**{Scrape Information}**

6. The system retrieves the information of all courses of the indicated subject, course code of all CC, course codes of all courses that do not have exclusion and names of all instructors.
7. The system scrapes all course subjects from the given base URL and term.
8. If a list called allSubjects that contain all course subjects exists
   8.1. The system replaces all the elements in allSubjects list that contain all course subjects with the newly scraped subjects.
9. If a list called allSubjects that contain all course subjects does not exist
   9.1. The system creates a new list called allSubjects list and stores the subjects scraped.
10. The user clicks the "Show" button.
11. The system displays the list of courses with section codes.
12. The system counts the total number of different sections and total number of courses.
13. The system displays the total number of different sections and total number of courses.
14. The system retrieves the names of instructors who have teaching assignment this term but does not need to teach at Tu 3:10 pm.
15. They system sorts the names of instructors retrieved ascendingly according to the alphabetical order of names.
16. The system displays the names of instructors who have teaching assignment this term but does not need to teach at Tu 3:10 pm ascendingly according to the alphabetical order of names.
17. The use case ends.

## Alternative Flow

***A1: Page Not Found***

At **{Scrape Information}** if the page is not found

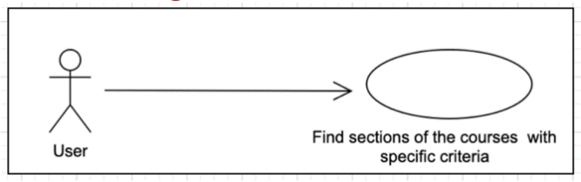
1. Display a message to notify users of the 404 page not found error
2. The flow of events is resumed at **{Enter URL Information}**

## Use case: Find sections of the courses with specific criteria

### Brief Description

This use case describes how a user find specific sections of the courses with the filter

### Use-case Diagram



### Basic flow

1. The use case begins when the user clicks the boxes/presses the "Select All" button with different criteria.
2. When the user press the only button
   - 2.1. If "Select All" button is clicked
     - 2.1.1. All the boxes on this tab are checked.

       **{Filter sections of the courses}**
     - 2.1.2. All the content in the console will be cleared and display all sections of the courses which fulfil the all the criteria of the checked boxes (AND logic).
     - 2.1.3. The text of the button will be changed to "De-select All".
   - 2.2. If "De-select All" button is clicked
     - 2.2.1. All boxes on this tab are unchecked.
     - 2.2.2. All the content in the console will be cleared.
     - 2.2.3. The text of the button will be changed back to "Select All".

   **{check all boxes}&{uncheck all boxes}**
3. When the user changes the status of any box on this tab (checked or unchecked)

   **{Filter sections of the courses}**
   - 3.1. The system clears all the content in the console and displays all sections of the courses which fulfil the all the criteria of the checked boxes (AND logic).
     - 3.1.1. If AM (or PM) is checked,
       - a. The system displays only all sections of the courses which has a slot in AM (or in PM).
     - 3.1.2. If both AM and PM are checked,
       - a. 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.
     - 3.1.3. If any boxes of the days of the week (Monday, Tuesday, …)
       - a. Display only all sections of the courses that has slots on the selected boxed.
     - 3.1.4. If Common Core is clicked

        a. The system displays only all sections of the courses that are 4Y CC
    3.1.5. If No Exclusion is clicked
        a. The system displays only all sections of the courses that does not define exclusion.
    3.1.6. If With Labs or Tutorial is clicked
        a. The system displays only all sections of the courses that has labs or tutorials
4. The use case ends.

**Alternative Flows**

***A1: No section of the courses fulfils***
At **{Filter sections of the courses}** if no any section of the courses fulfils the checked boxes
1. The system shows an empty console.

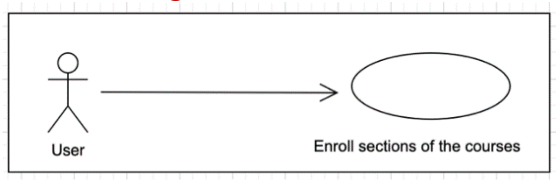
***A2: Check all boxes***
At **{check all boxes}** if all the boxes are checked
1. The system changes the text of the button from "Select All" to "De-select All".

***A3: Uncheck all boxes***
At **{uncheck all boxes}** if all the checked boxes are unclicked after the "Select All" button is clicked
1. The system changes the text of the button from "De-select All" back to "Select All".

## Use case: Enroll courses

**Brief Description**
This use case describe how a user enrolls sections from a list of filtered sections of the courses.

**Use-case diagram**



**Basic flow**
1. The use case begins when the user filters the section of the courses
2. The system clears the previous table (or the table is empty before)
   **{Result from filter}**
3. The system receives a result list/vector from the result of the filter.
4. The system fills the information of the sections of the courses (Courses Code, Lecture Section, Course Name, Instructor and the status of Enrolment with a checkbox) in a table from the result of the filter, all columns except the "Enroll" column are non-editable.
   **{Display Enrolled sections}**
5. The console will display "The following sections are enrolled:" followed by a list of enrolled sections.
   **{Enroll sections of the courses}**
6. When the status of the checkbox on this tab is changed

    6.1. When the checkbox is checked
        6.1.1. The enrolment status of this section is enrolled persistently, which means this section will be appended to the list of enrolled sections.
        6.1.2. The console will be cleared, and the new list of enrolled sections will be displayed in the console
    6.2. When the checkbox is unchecked
        6.2.1. The enrolment status of this section will be erased, which means the section will be deleted from the list of enrolled sections.
        6.2.2. The console will be cleared, and the new list of enrolled sections will be displayed in the console

**{Finish enrollment}**

7. The use case ends.

### Alternative flow

***A1: no result from filter***

At **{Result from filter}** if no result return from the filter

1. The system displays an empty table

***A2: no sections are enrolled***

At **{Display Enrolled sections}** if the user has not enrolled any sections

1. The system displays an empty list (no any section is showed)

***A3: Another lecture/tutorial/lab of this course has been enrolled***

At **{Enroll sections of the courses}** if another section of this course is enrolled before

1. The system rejects his enrolment request and show a warning "another section has been registered" in the console, then unchecked the "Enroll" checkbox of the rejected section.

***A4: Change the criteria in the filter***

Between **{Result from filter}** and **{Finish enrollment}**

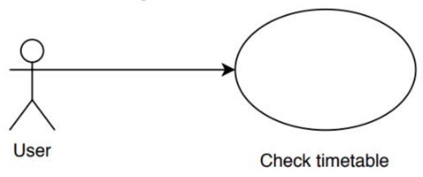
1. The user can add any new criteria/delete any existing criteria in the filter or select other tabs.
2. The system saves the enrolment status persistently until the user checks/unchecks the checkbox.

## Use Case: Check timetable

### Brief Description

This use case describes how a user checks timetable.

### Use-case Diagram



### Basic Flow
1. This use case begins when the user actor goes into the Timetable Tab.
2. The system looks for the section(s) enrolled by the user.

**{Retrieve Section}**

3. The system creates blocks for each time slot of the enrolled section.

**{Display Timetable}**

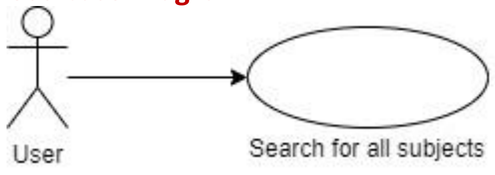
4. The system displays the timetable.
5. The use case ends.

### Alternative Flows

#### *A1: No Section Enrolled*
At **{Retrieve Section}** if there is no section enrolled by the user.
1. The system retrieves the first five sections of the scrapped data.
2. The flow of events is resumed at **{Retrieve Section}.**

#### *A2: Time Clash*
At **{Display Timetable}** if there is time clash (overlapped area exists).
1. The system gives a different color to the overlapped area.
2. The flow of events is resumed at **{Display Timetable}.**

## Use Case: Search for all subject
### Brief Description
This use case describes how the system response when the user clicks the “All Subject Search” button in the “All Subject Search” tab.

### Use-case Diagram


### Basic Flow
1. The use case begins when the User actor clicks the “All Subject Search” button in the “All Subject Search” tab.
2. The system cleans up the console.

**{Scrape Subjects}**

3. The system scrapes all course subjects from the given base URL and term.
4. If a list called allSubjects that contain all course subjects exists
   4.1. The system replaces all the elements in allSubjects list that contain all course subjects with the newly scraped subjects.
5. If a list called allSubjects that contain all course subjects does not exist
   5.1. The system creates a new list called allSubjects list and stores the subjects scraped.
6. The system records the total number of course subjects scraped.
7. The system displays the total number of course subjects scraped.

8. The system enables the "All Courses Search" button.
9. The use case ends.

**Alternate Flow**

***A1: Invalid URL***

At **{Scrape Subjects}** if the user-inputted URL is invalid,

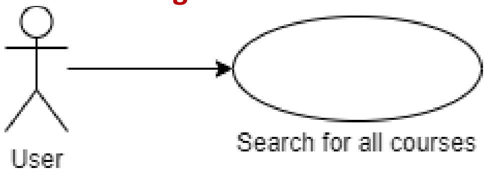
1. The systems informs the user that the URL is invalid.
2. The use case ends.

## Use Case: Search for all courses

**Brief Description**

This use case describes how the system response when the user clicks the "All Courses Search" button in the "All Subject Search" tab.

**Use-case Diagram**



**Basic Flow**

1. The use case begins when the User actor clicks the "All Courses Search" button in the "All Subject Search" tab.
2. The system cleans up the console and the progress bar status.

**{Scrape Subjects}**

3. The system scrapes all course subjects from the given base URL and term.
4. If a list called allSubjects that contain all course subjects exists
   4.1. The system replaces all the elements in allSubjects list that contain all course subjects with the newly scraped subjects.
5. If a list called allSubjects that contain all course subjects does not exist
   5.1. The system creates a new list called allSubjects list and stores the subjects scraped.

**{Scrape Courses}**

6. The system creates an empty list called allCourses.
   6.1. The system scrapes all courses of all subjects appear in the allSubjects list from the given base URL and term.
   6.2. The system performs the followings after all courses in each subject is scraped
      6.2.1. The system appends the allCourses list with the courses scraped.
      6.2.2. The system prints the message "SUBJECT is done" on the system console.
      6.2.3. The systems updates the progress bar by the fraction 1/(total number of subjects).
   6.3. The system displays the total number of courses.

**{Select All Filter}**

7. The system calls the "Select All" function in "Filter" tab.
8. The system passes the allCourses list to and calls the "Search" function in "Main" tab.
9. The system changes the text input in "Subject" in "Main" tab to '(All Subjects)'.
10. The system enables the "Find SFQ with my enrolled courses" button in the "SFQ" tab.

11. The use case ends.

**Alternate Flow**

***A1: Invalid URL***

At **{Retrieve Subjects}** if the user-inputted URL is invalid,

1. The system informs the user that the URL is invalid.
2. The use case ends.

***A2: No Subjects***

At **{Scrape Courses}** if the number of total number of course subjects is 0,

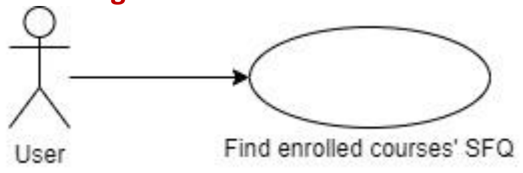
1. The system displays the total number of courses to be 0.
2. The flow of events is resumed at **{Select All Filter}.**

## Use Case: Find enrolled courses' SFQ

**Brief Description**

This use case describes how the system response when the user click the "Find SFQ with my enrolled courses" button in the "SFQ" tab.

**Use-case Diagram**



**Basic Flow**

1. The use case begins when the User actor clicks the "Find SFQ with my enrolled courses" button, and have clicked the "Search" button in "Main" tab or the "All Subject Search" button in "All Subject Search" tab once before.
2. The system cleans up the console.

**{Get Enrolled Courses}**

3. The system retrieves data of enrolled courses from the user-inputted SFQ URL.

**{Scrape SFQ Data}**

4. The system scrapes the unadjusted SFQ ratings of all the enrolled courses.
   - 4.1. If the user-inputted SFQ URL does not contain an enrolled course
     - 4.1.1. The system informs the user that the course was not available in the corresponding semester.
   - 4.2. If an enrolled course has only one section
     - 4.2.1. The system displays the unadjusted SFQ rating.
   - 4.3. If an enrolled course has multiple sections
     - 4.3.1. The system takes the average unadjusted SFQ ratings from different sections of the same course.
     - 4.3.2. The system displays the average unadjusted SFQ rating.
5. The use case ends.

## Alternate Flow

### *A1: Invalid SFQ URL*

At **{Get Enrolled Courses}** if the user-inputted SFQ URL is invalid,

1. The systems informs the user that the SFQ URL is invalid.
2. The use case ends.

### *A2: Enrolment not correctly implemented*

At **{Get Enrolled Courses}** if the enrolment is not correctly implemented in "List" tab,

1. If the number of sections in the scrapped data is more than or equal to 5
   1.1. The system enrolls the first 5 sections of the scrapped data.
2. If the number of sections in the scrapped data is less than 5
   1.1. The system enrolls all sections of the scrapped data.
3. The flow of events is resumed at **{Scrape SFQ Data}**

### *A3: No enrolled course*

At **{Get Enrolled Courses}** if there is no enrolled course,

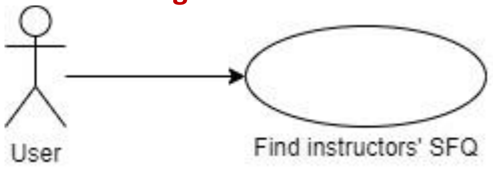
1. The system informs the user that there is no enrolled course.
2. The use case ends.

## Use Case: Finds instructors' SFQ

### Brief Description

This use case describes how the system response when the user click the "List instructors' average SFQ" button in the "SFQ" tab.

### Use-case Diagram



### Basic Flow

1. The use case begins when the User actor clicks the "List instructors' average SFQ" button.
2. The system cleans up the console.

**{Get Instructors' SFQ}**

3. The system scrapes the unadjusted SFQ ratings of instructors in every sections
   3.1 If an instructor shows up the first time
      3.1.1 The system stores the instructor's name and the corresponding SFQ rating in a list.
      3.1.2 The systems sets the number of courses teaches by that instructor to 1.
   3.2 If an instructor shows up again in another sections
      3.2.1 The system increments the number of courses teaches by the instructor by 1.
      3.2.2 The system calculates and stores the average unadjusted SFQ ratings.
4. The system displays the average unadjusted SFQ ratings of all the instructors in alphabetical order in the format of "LAST_NAME, FIRST_NAME".

5. The use case ends.

## Alternate Flow

### *A1: Invalid SFQ URL*

At **{Get Instructors’ SFQ}** if the user-inputted SFQ URL is invalid,

1. The systems informs the user that the SFQ URL is invalid.
2. The use case ends.