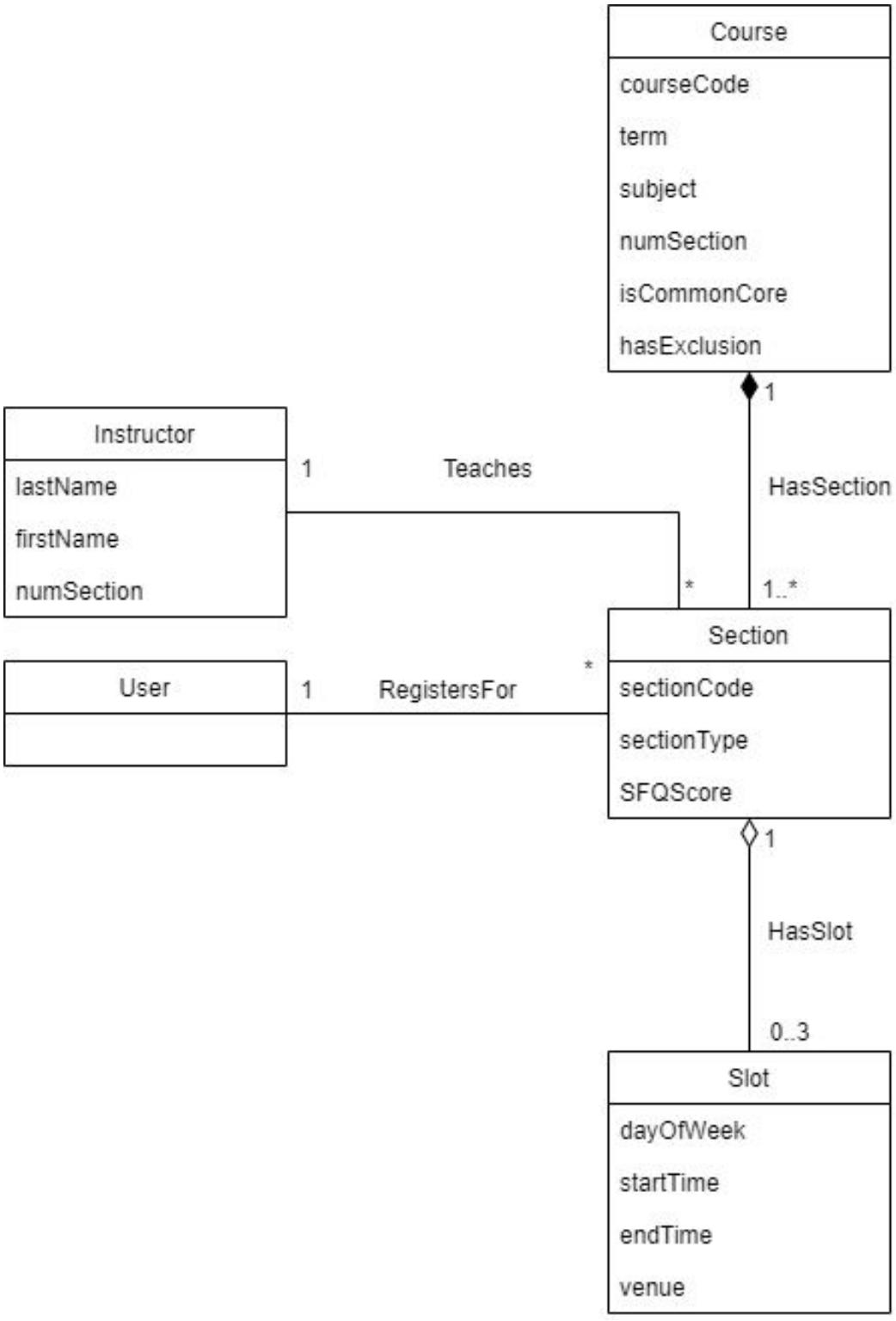
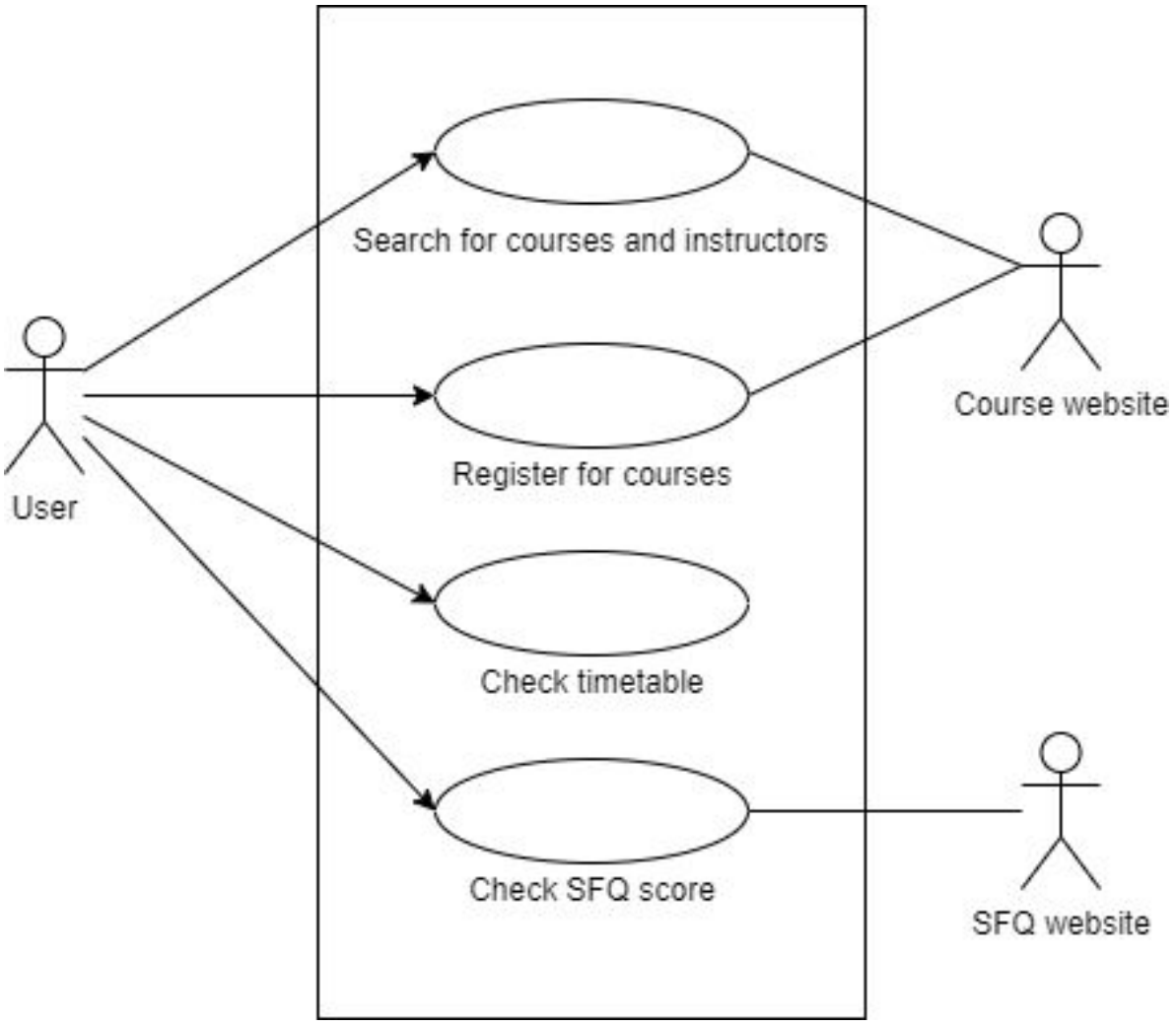
COMP3111
Software Engineering Team Project
Course Scraper

Team Code	**T-24**
Team Members	CHONG Cheuk Hei (Tasks 1 & 5) LAW Kai Yuet (Tasks 2 & 3) LI Lok Yin (Tasks 4 & 6)

Data Model Diagram



Use-case Diagram



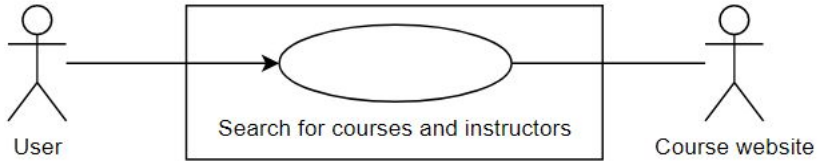
Initial Use-case Specification

Search for Courses and Instructors

Brief Description

This use-case describes the process by which a user searches for courses. The process includes the function of searching courses from a course website, to display course sections and number of courses.

Use-case Diagram



Flow of Events

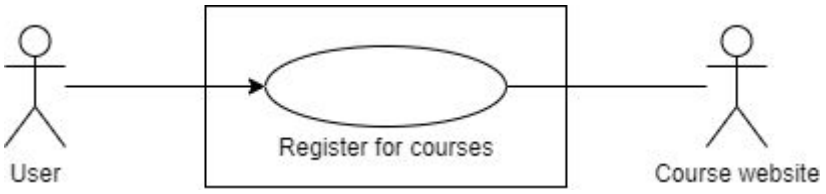
1. Fetch Courses Information from Course Website

Register for Courses

Brief Description

This use-case describes the process by which a user registers for a course. The process includes capacities to filter displayed course sections, and to change enrollment status for course sections.

Use-case Diagram



Flow of Events

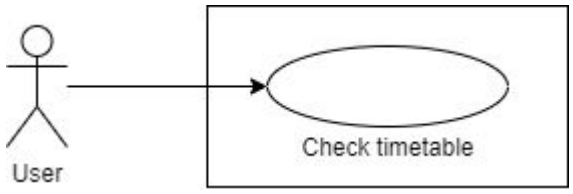
1. Filter Displayed Courses Sections
2. Manage Enrollment Status for Course Sections

Check Timetable

Brief Description

This use-case describes how a user views a timetable representing the distribution of slots of their enrolled course sections.

Use-case Diagram



Flow of Events

1. View Timetable

Check SFQ Scores

Brief Description

This use-case describes the process by which a user views the SFQ scores of courses and instructors. The process includes the capacities to fetch and view unadjusted SFQ scores of the user's enrolled courses, and to view unadjusted SFQ scores of all instructors.

Use-case Diagram

Flow of Events

1. View Enrolled Courses' SFQ Scores
2. View Instructors' SFQ Scores

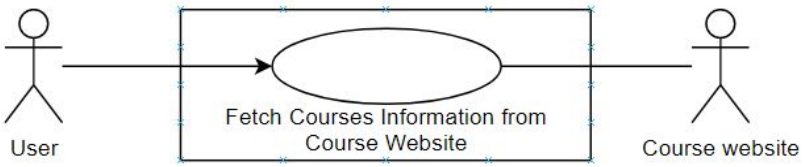
Use-case Detailed Specification

Use-case: Fetch Courses Information from Course Website

Brief Description

This use-case describes how a user searches the courses that match the user input in the textbox in the “Main” tab, or searches all subject courses in “All Subject Search” tab.

Use-case Diagram



Basic Flows

1. This use-case begins when the User actor clicks the “Main” tab.
2. The system displays some textbox for users to input Base URL, Terms, and Subject.

{Select Search Type}

3. The User actor clicks the “Search” button in “Main” tab to start searching the specific subject courses based on the user actor input above.

{Begin Searching Courses}

4. The system loads the URL based on the input and fetches the course information.

{Display the Course Results}

5. The system displays the total number of unique Sections based on this search in the console.
6. The system displays the total number of Courses that have at least one Section based on this search in the console.

{Display the Instructor Results}

7. The system sorts the order of the Instructor's name obtained in the condition that those Instructors only have no teaching assignment at Tu 3:10pm ascendingly according to the alphabetical order.
8. The system displays the sorted Instructors' name list with format LAST_NAME, First_name in the console.
9. The use-case ends here.

Alternative Flows

A1: All Subject Search

At **{Select Search Type}**,

1. User actor clicks the “All Subject Search” tab.
2. The user actor clicks the “All Subject Search” button to start searching all subjects based on the user input above.
3. The system displays the total number of categories/code prefix in the console.
4. The user actor clicks the “All Subject Search” button again.
5. The system scraped one subject and print “SUBJECT is done” in the system console every time.
6. The system updates the progress bar by the fraction 1/ ALL_SUBJECT_COUNT.
7. The system displays the total number of courses fetched in the console.

8. The system updates the results in “Filter”, “List” and “Backend” Tab.

A2: Error 404

At **{Begin Searching Courses}**,

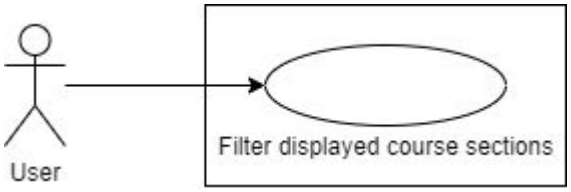
1. The system displays an appropriate message on the screen to notify the users the page is not found [Error404].
2. The flow of events resumes at **{Select Search Type}**.

Use-case: Filter Displayed Course Sections

Brief Description

This use-case describes how a user filters the fetched list of Courses to show only the Course Sections that match the selected criteria in the “Main” and “List” tab.

Use-case Diagram



Basic Flows

1. This use-case begins when the User actor clicks the “Filter” tab.
2. The system displays an interface of checkboxes representing the filtering criteria and a SELECT ALL / DE-SELECT ALL button.
3. While the User selects or de-selects a criterion

   {Initialize Displays}

   3.1. The system clears the console in the “Main” tab.

   3.2. The system clears the displayed Sections in the “List” tab.

   {Select Filters}

   3.3. If the button SELECT ALL is clicked

   3.3.1. The system checks all checkboxes in the current interface.

   3.3.2. The system changes the text of this button from SELECT ALL to DE-SELECT ALL.

   3.4. If button DE-SELECT ALL is clicked

   3.4.1. The system unchecks all checkboxes in the current interface.

   3.4.2. The system changes the text of this button from DE-SELECT ALL to SELECT ALL.

   3.5. If any checkbox TIME OF DAY {AM, PM} is checked

   3.5.1. The system applies the filter “show only Sections with at least one Slot which span the selected TIME OF DAY” to the list of Sections.

   3.6. If any checkbox DAY OF WEEK {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday} is checked

   3.6.1. The system applies the filter “show only Sections with at least one Slot which is on the selected DAY OF WEEK” to the list of Sections.

   3.7. If the checkbox COMMON CORE is checked

3.7.1. The system applies the filter “show only Sections of Courses which are 4Y CC” to the list of Sections.

3.8. If the checkbox NO EXCLUSION is checked

3.8.1. The system applies the filter “show only Sections of Courses which do not define exclusions” to the list of Sections.

3.9. If the checkbox WITH LABS OR TUTORIALS is checked

3.9.1. The system applies the filter “show only Sections of Courses which have labs or tutorials” to the list of Sections.

3.10. If any checkbox is unchecked

3.10.1. The system removes the filter represented by the unchecked checkbox from the list of Sections.

3.11. The system combines the selected filters with AND logic.

{Update Displays}

3.12. The system displays Sections which satisfy the combined filters in the console in the “Main” tab.

3.13. The system displays Sections which satisfy the combined filters in the interface of the “List” tab.

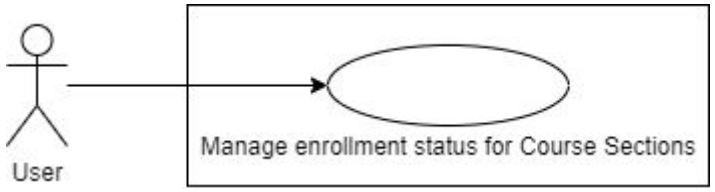
4. The use-case ends.

Use-case: Manage Enrollment Status for Course Sections

Brief Description

This use-case describes how a User views a list of Sections which match the filters specified in the “Filter” tab, and how a User enrolls in or withdraws from Course Sections.

Use-case Diagram



Basic Flows

1. This use-case begins when the User actor clicks the “List” tab.
2. If there exists any Section in the User’s “enrolled Course list” but not in the list of Sections

   {Make Enrolled Course Persist}

   2.1. The system duplicates this Section and stores it in the list of Sections.
3. The system displays the interface consisting of a table of Sections which match the filters specified in the “Filter” tab, and a checkbox ENROLL next to each displayed Section.
4. While the User selects or deselects an enrollment

   {Initialize Console}

   4.1. The system clears the console in the “Main” tab.

   4.2. The system displays Sections which satisfy filters specified in the “Filter” tab in the console in the “Main” tab.

   {Change Enrollment Status}

   4.3. If any checkbox ENROLL is checked

4.3.1. The system duplicates the selected Section and stores it in the User's "enrolled Course list".

4.4. If any checkbox ENROLL is unchecked

4.4.1. The system removes the de-selected Section from the User's "enrolled Course list".

{Update Timetable}

4.5. The system updates the interface of the "Timetable" tab to display the Slots of Sections in the User's "enrolled Course list".

4.5.1 For each Sot of enrolled Section

4.5.1.1 Assign a block color for the Section.

4.5.1.2 Create a block of the assigned color at the time slot of the Section.

4.5.1.3 Print the course code and section code inside the block in two lines.

4.5.1.4 If time class clash occurs, color the overlapping part with a new color. This Color is based on the color of the overlapping courses.

{Update Console}

4.6. The system displays the User's "enrolled Course list" in the console of the "Main" tab.

5. The use-case ends.

Alternative Flows

A1: Enrolled Course List is unimplement

At **{Update Displays}** if the User's "enrolled Course list" is unimplemented

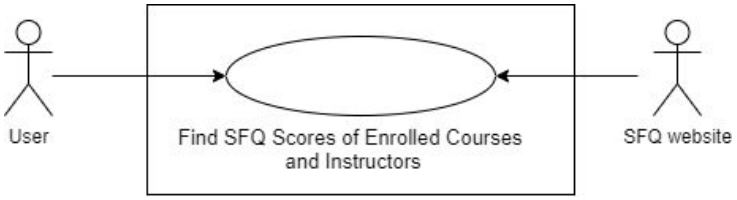
1. The system updates the interface of the "Timetable" tab to display the Slots of the first 5 Course Sections in the list of Course Sections.
2. The flow of events is resumed at **{Update Console}**.

Use-Case: Find SFQ Scores of Enrolled Courses and Instructors

Brief Description

This use-case describes how a user obtains SFQ data of their enrolled courses or the average SFQ score of their course instructors by providing the URL of the SFQ in the "SFQ" tab.

Use-Case Diagram



Basic Flows

1. This use-case begins when the User actor clicks on the "SFQ" tab.
2. The system shows a text input box for the SFQ URL and two buttons: Find SFQ with my enrolled courses, and List instructors' average SFQ

   {Obtain course SFQ Data}
3. While the User has an action to perform

   3.1. If the button "Find SFQ with my enrolled courses" is clicked

   3.2. If the Course has one Section

   3.2.1. Print the unadjusted SFQ data in the console.

3.3. If the Course has multiple Sections

3.3.1. Obtain the simple average of unadjusted SFQ data

3.3.2. Print the averaged SFQ data in the console

{Obtain instructor's average SFQ}

3.4. If the button "List instructors' average SFQ" is clicked

3.4.1. While SFQ data of each instructor is processed

3.4.1.1. If the Instructor taught one Course

3.4.1.1.1. Print the instructor's unadjusted SFQ score in the console

3.4.1.2. If the Instructor taught multiple Sections or multiple Courses

3.4.1.2.1. Add up all unadjusted SFQ scores of the instructor

3.4.1.2.2. Divide the SFQ score by the number of sections taught by him/her

3.4.1.2.3. Print the divided SFQ score in the console

4. The use-case ends