Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and Artificial Intelligence

CSAI 203 – Fall 2025

Introduction to Software Engineering

Course Registration System

Software Requirements Specification

Team Number: 23

Team Members:

Hazem Ahmed Refaat – 202401389

Youssof Amgad Abdelrahman – 202400370

Maya Sameh Aboamer – 202401080

Yaseen Mohamed Bilal – 202400463

Representative contact info:

s-hazem.refaat@zewailcity.edu.eg

21/10/2025

Document Outline

1. Introduction

1.1 The purpose of this system is to make an easy-to-use, organized and reliable way for students and professors to manage their courses.

1.2 The Course Registration System allows students to add/drop courses, view their grades and view their registered courses and allows professors to grade courses, remove/admit students and view all registered students. Admins can add/remove courses, assign professors for courses and modify student and professor accounts.

1.3

System: Refers to the Course Registration System being developed.

UI (User Interface): The part of the system the user interacts with and navigates.

Backend: The part of the system that contains the logic and implementation of the required functionalities.

Flask: A lightweight Python web framework used for developing the system’s backend.

HTML (HyperText Markup Language): The standard language used to make the structure of a webpage.

CSS (Cascading Style Sheets): A language used to style and organize HTML elements.

1.4

Flask Documentation: https://flask.palletsprojects.com/

Python 3 Documentation: https://docs.python.org/3/

1.5 The rest of the document describes the requirements for the Course Registration System as follows:

Section 2 provides an overall description of the system, its target users and constraints on how it will be implemented.

Section 3 provides a more detailed description of how the system functions and how it should be implemented.

Section 4 provides additional information that was not mentioned in the above sections.

2. Overall Description

2.1 The Course Registration System is designed to replace manual course management with a centralized platform.

2.2

Student Functions: Register and manage their accounts. Add or drop available courses within the registration period. View registered courses and grades.

Professor Functions: Manage assigned courses. View all students registered in their courses. Assign and update student grades.

Administrator Functions: Add or remove courses from the system. Assign courses to professors. Manage student and professor accounts (create, modify, or delete).

2.3 The system contains three user classes: Student which registers for and manages their courses, Professor which manages course content and gives grades, and administrator which oversees all accounts and can adjust all system aspects.

2.4

Backend environment: Python 3, Flask and database.

Frontend environment: Any modern browser, HTML and CSS.

2.5 The system will be implemented using Python (Flask) for backend and HTML/CSS for the frontend. The UI must be intuitive, and the website will use cookies to keep longer login sessions.

2.6 The system will include a user guide with step-by-step instructions for students, professors, and administrators to perform their functions and cover logging in and out and navigating the interface.

2.7 Users will have stable internet access and basic computer skills. The system will depend on Flask and its libraries, updates and changes to it may affect the system.

3. Specific Requirements

3.1

Functional requirement	Description	Scenario
Register course	The student user selects a course from the list of offered courses and adds it to their registered courses list.	The user logs in as a student and goes to the offered courses list, selects a course and the software checks that the student is not already registered in that course and if the student is not already registered the course is added to their registered courses list and registered credit hours are changed accordingly.

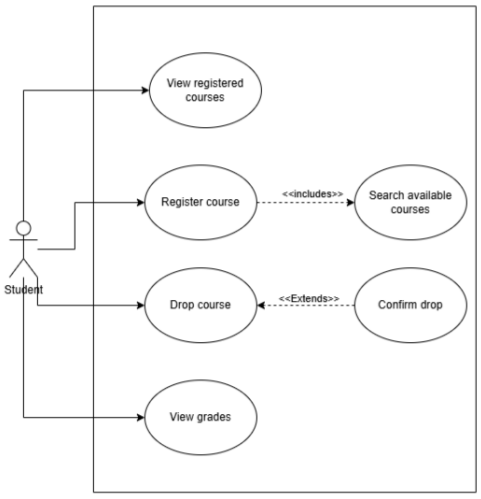
Drop course	The student user selects an already registered course and drops it from their registered courses list.	The user logs in as student and goes to the registered courses list. There He/ She can choose the course they want to drop and successfully remove it from the registered courses. Subsequently the number of registered credit hours for the students changes according to the newly dropped course.
View grades	The student can view a transcript with their GPA and individual grades for each course graded by a professor.	The user logs in as a student and goes to the transcript page which views all their grades and the calculated GPA based on those grades and the courses credit hours.
View registered courses	The student can view the list that includes all their currently registered courses and their details including the professor teaching, the course period (Spring, Fall, Summer) and the course's credit hours. The student can also view the total credit hours registered.	The user logs in as a student and goes to the registered courses page where they can view the registered courses' names, they select or hover over a course and its details appear.
Remove/admit students	Professor users can approve an admission request or remove a registered student.	Professor logs in and goes to the requests page, there He/ She can view any requests for admission to the course and approve them. He/ She can also remove any of the existing students.

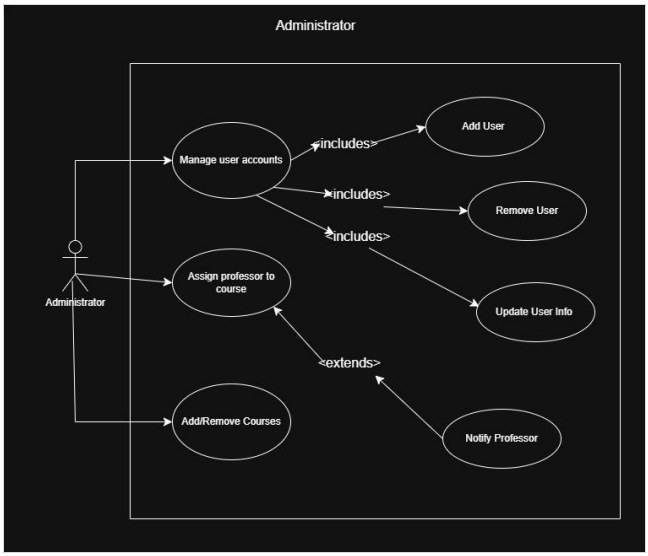
View all registered students	Simple function and its only purpose is for the instructor to be able to view all registered students in a course.	Instructor can log in to his account, go to the courses he is teaching and view all students registered in each course
Grade	Professor user can select a course they are teaching and give letter grades to all the students registered in that course.	The user logs in as a professor and views all the courses they teach, they select a course and view all the registered students in that course and then they assign each student a letter grade from ‘A’ to ‘F’.
Add/remove available courses	The admin adds a new course to the courses offered list or removes an already offered course.	The user logs in as an admin and then goes to the offered courses list to add or remove a course in the list, to add the course the admin provides the details required for the course and then the software checks if the course is already offered and if not, then the course gets added.
Modify accounts	The admin chooses any account registered whether it's a student or a professor and modifies or removes them.	The user logs into the software as an admin and then goes to the accounts registered list and he can select any account whether it's a student or an instructor/professor and the admin can modify the details of the account or even remove it from the system completely.
Assign professors for courses	The admin chooses which professor to assign to a designated course that's in the offered courses list.	The user logs in as an admin and then goes to the courses offered list and then the admin will select any course and assign the instructor/professor,

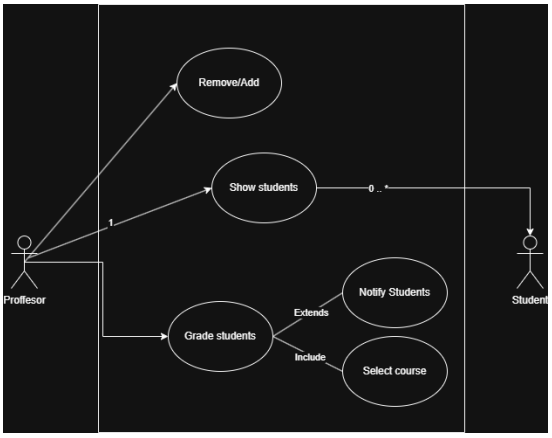
		and that applies for every course in the list.

3.2

3.2.1







3.2.2

Use Case ID	UC-01
Use Case Name	Register course
Primary Actor	Student

Stakeholders and interests	Student: Needs a method to register for courses to add credit hours and get graded. Professor: Needs students to be able to register for their courses so they can grade them.
Preconditions	The student has not been already registered for the selected course.
Postconditions	The student adds the course to their registered courses list and the course's professor can see and grade them in that course (success). Course not added with an error message stating the reason (fail)
Trigger	Student selects a course from the offered courses list and clicks register
Main Success Scenario	1. Student goes to the offered courses list 2. Selects the course they want to register for 3. System checks if student is not already registered for that course 4. The system adds the course to the student's registered courses and adds the student to the course's registered students.
Include	System executes "Search available course
Special requirements	None
Priority	High
Frequency of use	Every registration period (Summer/Fall/Spring)

Use Case ID	UC-02
Use Case Name	View registered courses
Primary Actor	Student
Stakeholders and interests	Student: Needs to view their courses in an organized way.

Preconditions	The student has registered courses.
Postconditions	The student can view all courses they are registered in with their details (success). The student gets notified that they do not have any registered courses (fail)
Trigger	Student goes to the registered courses page.
Main Success Scenario	1. Student goes to the registered courses page 2. Selects a course they want to view 3. Course details become visible to the student
Special requirements	None
Priority	High
Frequency of use	Every day

Use Case ID	UC-03
Use Case Name	Drop course
Primary Actor	Student
Stakeholders and interests	Student: Needs a method to drop from a course they have previously registered for.
Preconditions	The user has registered courses
Postconditions	The selected course is removed from the student's registered courses list, and the credit hours are reduced accordingly, the professor will no longer be able to see the student in that course.
Trigger	Student selects a course from the registered courses list and clicks drop.
Main Success Scenario	1. Student goes to their registered courses list 2. Selects a course 3. Clicks drop
Special requirements	None

Priority	Medium
Frequency of use	Irregular

Use Case ID	UC-04
Use Case Name	View grades
Primary Actor	Student
Stakeholders and interests	Student: Needs to be able to view their grades for each course after completion and their calculated GPA. Professor: Needs a method so that their students can see the grades they have been given.
Preconditions	The user has graded courses
Postconditions	The student has a view of all their graded courses with their letter grades and the calculated GPA (success). An error message appears telling the student that no courses have been graded (fail).
Trigger	Student goes to the grades transcript page.
Main Success Scenario	1. Student goes to the grades transcript page. 2. Views all the courses with their letter grades, completed credit hours and the calculated GPA.
Special requirements	None
Priority	High
Frequency of use	End of every period (Summer/Fall/Spring)

Use Case ID	UC-05
Use Case Name	Add/Remove course
Primary Actor	Administrator
Stakeholders and Interests	**Admin:** needs to manage all the courses offered on the website by adding new ones or removing existing ones. **Professors:** need to know which courses are available and if any are canceled or newly added. **Students:** need to see the correct list of available courses to register in and to know if any course gets removed.
Preconditions	The admin is logged into the system and on the “Offered Courses” page.
Postconditions	Success: The course is either added or removed successfully from the offered courses list, and the update is reflected immediately across the system. Fal: If the course already exists (in case of adding) or cannot be removed (in case of locked or graded courses), the system shows an error message.
Trigger	The admin clicks on either “Add Course” or “Remove Course” from the offered courses list.
Main Success Scenario	1. The admin logs in as an admin. 2. Goes to the offered courses page. 3. Selects to either **Add** or **Remove** a course. 4. If added, the admin fills in the required details for the new course (course code, title, credits, etc.). 5. The system checks if the course already exists. 6. If it doesn’t, the course is added to the offered list.

	7. If removing, the admin selects the course and confirms removal. 8. The system removes the course from the offered list and updates all related data.
Special Requirements	The admin must have full permission to modify the course list.
Priority	High
Frequency of Use	Every semester or when course offerings need to be updated.

Use Case ID	UC-06
Use Case Name	Assign professor to course
Primary Actor	Administrator
Stakeholders and Interests	**Admin:** assigns professors to the correct courses to prepare the schedule. **Professors:** need to know which courses they will teach. **Students:** should be able to see who will instruct each course.
Preconditions	The admin is logged in and both the professor and the course exist in the system.
Postconditions	The selected professor is successfully assigned to the course and notified (success).
Trigger	The admin selects a course from the offered courses list and clicks “Assign Professor.”
Main Success Scenario	1. The admin logs in to the system. 2. Goes to the offered courses list. 3. Selects a course to assign a professor. 4. Chooses a professor from the available list. 5. The professor gets assigned to the course successfully.
Extends	The use case extends **“Notify professor”** to send a notification that they have been assigned to a course.

Priority	High
Frequency of Use	Every semester or whenever courses are being assigned.

Use Case ID	UC-07
Use Case Name	Manage user accounts
Primary Actor	Administrator
Stakeholders and Interests	**Admin:** needs to have full control over all accounts. **Students and Professors:** need their accounts to stay up-to-date and accurate. **System:** must maintain valid and secure user data.
Preconditions	The admin is logged in and on the “User Accounts” page.
Postconditions	The selected user account is modified, updated, or removed successfully (success). If information is invalid, the operation fails and an error message is shown.
Trigger	The admin selects a user account (student or professor) and chooses to modify or remove it.
Main Success Scenario	1. The admin logs in as an admin. 2. Goes to the accounts list page. 3. Selects any account from the list (student or instructor). 4. Chooses to modify, update, or remove the account. 5. Makes the changes and confirms. 6. The system updates or deletes the account accordingly.
Special Requirements	Admin must have full access permissions.
Priority	High
Frequency of Use	Frequently used throughout the semester whenever account changes are needed

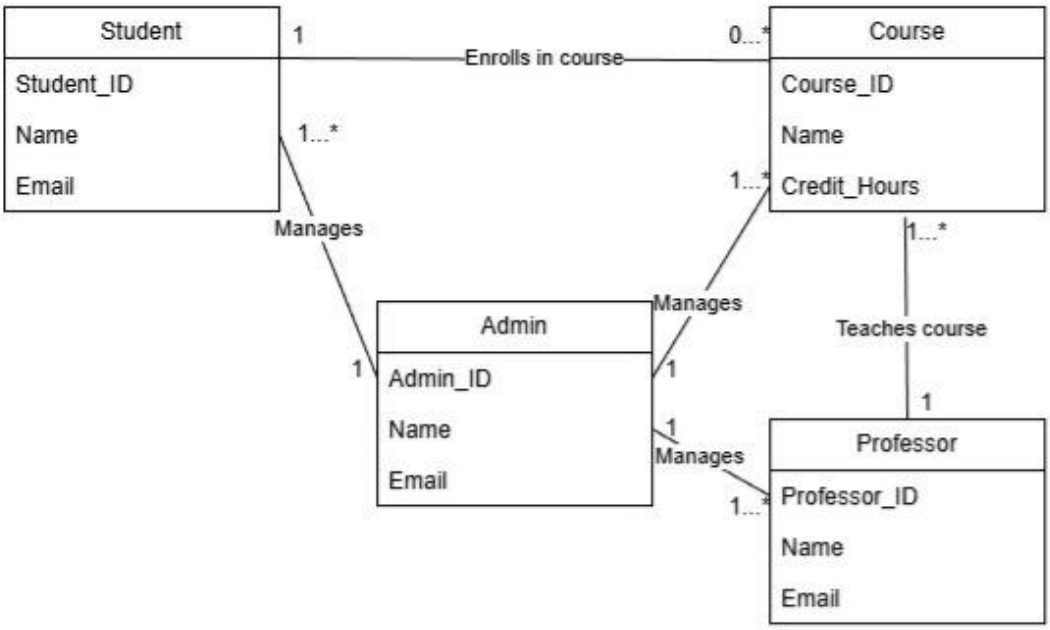
Use Case ID	UC-08
Use Case Name	Grade Students
Primary Actor	Professor
Stakeholders and Interests	Professor Needs to assign and update grades for students in their course. **Students:** Have an interest in receiving accurate and timely grades for their work.
Preconditions	The Professor is logged into the system, is viewing the course they are teaching.
Postconditions	**Success:** The grade is entered or updated in the system and is visible to the student. **Fail:** If the grade is invalid.
Trigger	The Doctor/Instructor selects the "Enter Grades" or "Grade Students" function for a specific course assignment or the final course grade.
Main Success Scenario	1. The Professor logs in. 2. Navigates to the course they are teaching. 3. Selects the "Grade Students" function. 4. The system displays the list of enrolled students. 5. The instructor enters or updates the grade for each student. 6. The instructor submits the grades. 7. The system validates and saves the grades.
Includes	The use case includes "**Select course**" to select the course needed.
Extends	The use case extends "**Notify student**" to alert them that a new grade has been posted.
Special Requirements	Grades, once finalized and submitted after the deadline, should be locked and require admin override to change. All grade changes must be logged.
Priority	High
Frequency of Use	Periodically, after end of the semester.

Use Case ID	UC-09
Use Case Name	View All Registered Students
Primary Actor	Professor
Stakeholders and Interests	Professor **:** Needs to see the list of all students registered in their course for tracking, communication, and grading purposes. **System:** Must provide accurate and real-time student data.
Preconditions	The Doctor/Instructor is logged into the system and has selected a course they are teaching.

Postconditions	The system displays a complete and accurate list of all students registered in the selected course.
Trigger	The Doctor/Instructor navigates to a specific course and selects the "View Registered Students" option.
Main Success Scenario	1. The Doctor/Instructor logs in. 2. Navigates to the 'My Courses' section. 3. Selects a specific course. 4. Clicks on "View Registered Students". 5. The system retrieves and displays the list of all enrolled students.
Special Requirements	None
Priority	High
Frequency of Use	Very frequently, throughout the semester.

Use Case ID	UC-10
Use Case Name	Admit/Remove Students
Primary Actor	Doctor/Instructor
Stakeholders and Interests:	Doctor/Instructor: Needs to manage student roster Students: Need course access or removal System: Maintain accurate enrollment records
Preconditions	Doctor is logged in and viewing their course
Postconditions	Student is admitted/removed; error shown if operation fails
Trigger	Selecting "Admit Student" or "Remove Student" function
Main Success Scenario	Doctor logs in .1 Navigates to teaching course .2 Selects admit/remove function .3 Selects student and confirms action .4 System updates course .5
Special Requirements	None
Priority	High
Frequency of Use	Frequently during add/drop periods

3.3 - 3.3.1



3.3.2

Student	Student_ID, Name, Email	Enrolls In Course	1..* per course	“name”, “ID number”, “Email”
Professor	Professor_ID, Name, Email	Teaches Course	1 per course	“name”, “ID number”, “Email”
Course	Course_ID, Name, Credit_Hours	Has Multiple Students and a Professor	0..* per student, 1..* per professor	“name”, “ID number”, “Credit Hours”

Admin	Admin_ID, Name, Email	Manages Courses, Students, and Professors	1 manages Courses, Students, and professors	“name”, “ID number”, “Email”

3.4

Non-functional requirement	Method of testing	Success criteria
Intuitive user interface	Ask testers who have not used the software before to perform each of the functional requirements without giving them steps or helping them navigate the website.	If at least 90% of the testers were able to perform all the functional requirements successfully within a reasonable time and did not report any trouble navigating the website.
Long term login session	We will test it by leaving the website open for a period of time, also by closing the website for a few days and then reopen it again.	If the user doesn't get logged out of the website by the tests we mentioned then the only way the user could log out of the site is by manually doing so.

3.5

3.5.1 The UI will include a login page, navigation menu, tables and menus that will include the courses and students.

3.5.2 The system will require a computer or a mobile device with an internet connection and can run a modern browser.

3.5.3 The system will need a database for data storage, other than that it is standalone for the functionality it serves.

3.5.4 Server communication is done via HTTPS over a stable internet connection.

4. Appendices

4.1

Student_ID, Professor_ID, Admin_ID: Unique identifiers for each user (integer).

Course_ID: Unique identifier for each course (string).

Grade: Letter given as a grade to each student in each course (char).

4.2

GPA: Grade Point Average, calculated given each course’s credit hours and its grade.

Administrator: User who has authority to manage system data and permissions.