Software Requirements Specification (SRS)

Smart Course Registration System (SCRS)

Course: Software Engineering Project – Phase 2
Instructor: Mohameds Sami Rakha

Prepared by:

Team Member Name	Student ID	Student Email
Mariam Ahmed Maher	202400733	s-mariam.maher@zewailcity.edu.eg
Farida Emad	202401764	s-farida.fouad@zewailcity.edu.eg
Elaliaa Kotb	202402473	s-elaliaa.kotb@zewailcity.edu.eg
Rahma Ali	202400365	s-rahma.anwar@zewailcity.edu.eg

1. Introduction

1.1 Purpose

This document serves to identify the software requirements of the Smart Course Registration System (SCRS). The system aims to make course this makes registration easier and more efficient for university students. It allows students register and drop courses view their weekly schedule receive email reminders of forthcoming lectures and tutorials. This document defines the functional and non-functional requirements describe the main features of the system and explain the interactions between different users and the system.

1.2 Scope

The Smart Course Registration System, SCRS, is system that will make it easier to register to courses , view and manage them.

The major functions ofthe system include:

1. Registration for available courses.
2. Dropping registered courses
3. Receiving email reminders automatically for upcoming classes.
4. Enabling instructors to list course information and schedules.
5. Permitting administrators to manage the system database, user accounts, and course offerings in general.

The SCRS will reduce administrative workload and improve communication

between students and instructors, and admins.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
SCRS	Smart Course Registration System
UI	User Interface
DB	Database
Admin	System Administrator
UML	Unified Modeling Language
Use Case	Description of system behavior from the user's perspective

1.4 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications.
- Project Guidelines and Phase 2 Instructions Document (provided by course instructor).
- flask/html/css documentation

1.5 Intended Audience

The SCRS is mainly designed for all of the university student body who are involved in the course registration process:

1. Students: Can choose courses available accordingly with their major and can add and drop courses fast without long trips to offices
2. Instructors: Can edit information about a course , add announcements and assignments and hande out grades
3. Admins : They can add new members of the university and assign them a role(TA, instructor or student)
4. Teaching Assistant: Communicate most with the students regarding grading, projects, submissions, and can receive permission from the instructor to grade assignments

1.6 OverView

This SRS will explain what the system will do, who will use it, and how it will function. It covers both the functional and non-functional requirements, along with diagrams and models that will help in the next design and implementation phases.

2. Overall Description

2.1 Product Perspective

SCRS is a web-system interacting with the database where user, course, and schedule data. It will be developed using Flask,HTML, and CSS. It will have three main user interfaces — Student, Instructor, and Admin.

The system architecture will follow a client-server model, where:

1. The frontend is accessed through a web browser.
2. Data processing and interaction with the database are done on the backend.
3. Information about courses, users, and schedules is stored in the database.

2.2 Product Functions

The following are the main functions of SCRS:

1. Log in to the system.
2. View all courses open for registration and other information like how many seats are left and how many credit hours it is
3. Drop/Add registered courses before the deadline.
4. Receive automated email reminders of upcoming sessions

5. Instructors can view and edit their courses.
6. Instructors can also upload a general annoncement to all students enrolled in his course
7. Instructors can also upload an assignment and grade it
8. Instructors can give TAs permission to view student work and grade them
9. TA , instructor and admin can view all students enrolled in a course but TA and Instructors can only see for courses they teach, admins have access to all courses
10. Admins can add/remove courses, add/remove users

2.3 User Classes and Characteristics

Role	**Main Functionalities**	**Sensitivity**
Student	Registers, drops, and views courses.	Basic
Teaching assitant	Grades assignments, views student info, post annoncements and assignments	basic
Instructor	Manages course content, and related updates.	Medium
Admin	Manages users, courses, and system configurations.	High

2.4 Operating Environment

The program can run on any device using:

- **Operating system : windows/linux**
- **Backend: Flask**
- **Frontend: Html/css**
- **Browser: Chrome/Edge/Firefox**

2.5 Design and Implementation Constraints

- Project interface is implemented using HTML/CSS and Flask and no database(sql or similar) is used.
- Once program is closed all information is erased since it is stored in python per session
- The program will follow client-server architecture

2.6 User Documentation

The system will include a simple help section for users explaining how to:

- Sign Up / log in.
- Enroll in and drop courses.
- View schedules.

- Manage account settings.

2.7 Assumptions and Dependencies

1. A valid university email for notification purposes is a requirement.
2. Roles are assigned by admins are assigned by the university system.
3. The system requires an internet connection to work.
4. This email reminder service depends on a third-party email server(Gmail)
5. In the future the system will be tied to a stable database to support large volumes of data

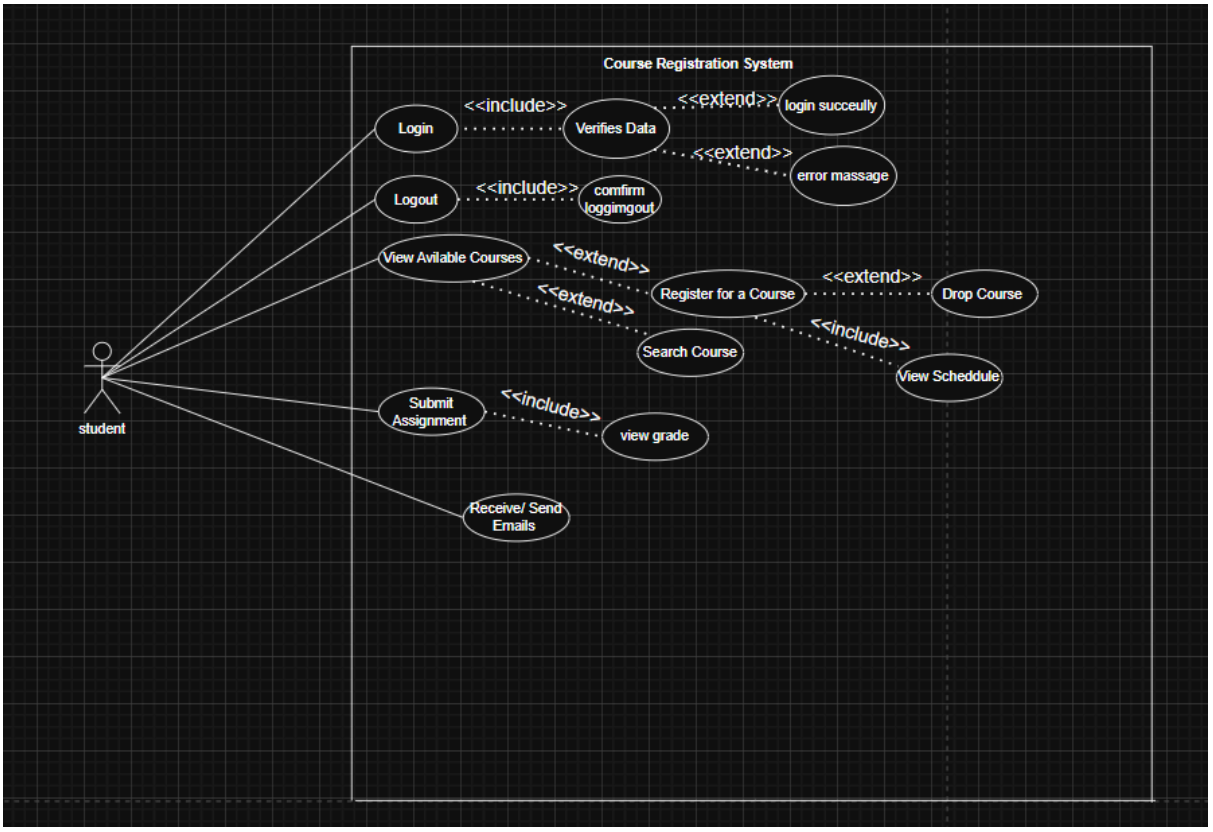
3. Specific Requirements

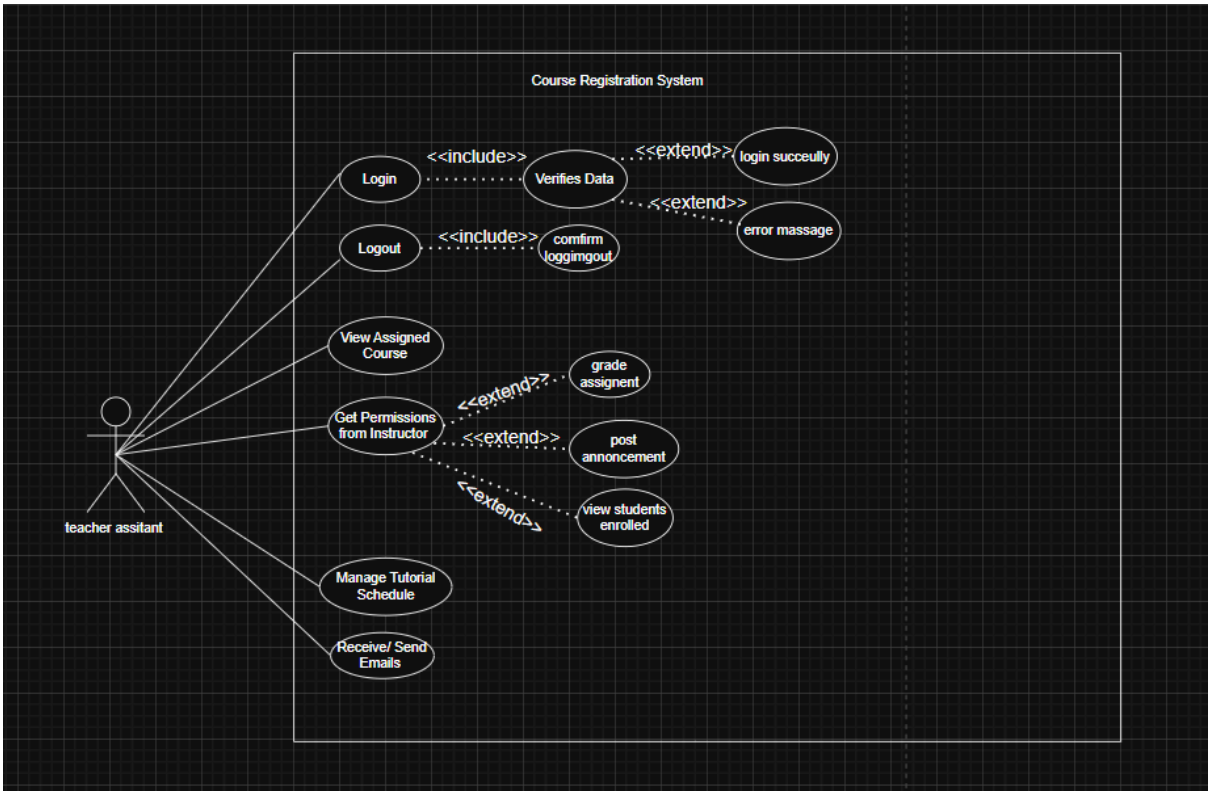
3.1 Functional Requirements

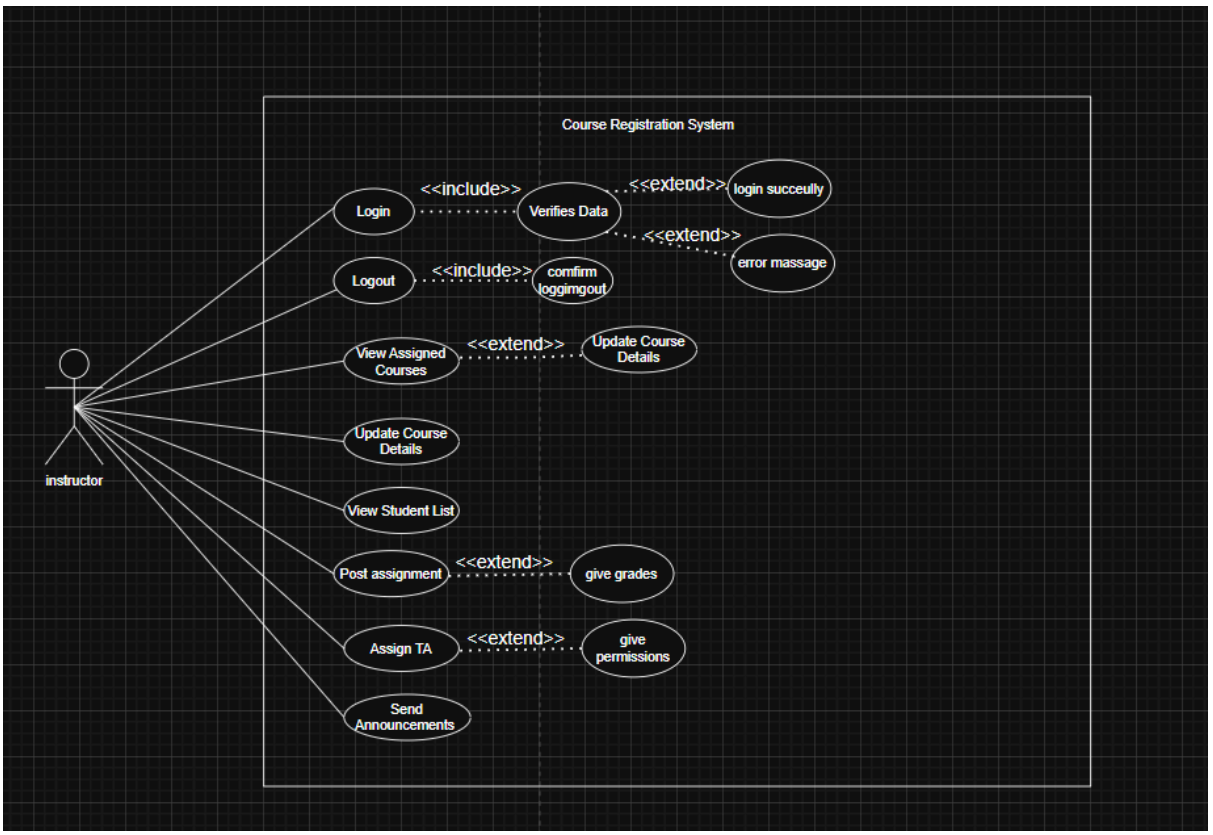
Functional Requirement	Description
User Authentication	Authenticates the user login(checks if user is registered in the system and loads the appropriate page according to his role)
Course Managment	Admins can add/ delete/ edit courses and their info
Course Registration and Dropping	Students can register for courses or drop them during a given time (add/drop period) specified by the admins
Email Notification	System uses 3rd party mail software (gmail) to send daily reminders to students of their schedule today according to the courses they are registered too
System Administration	Admins manage access and roles to users and also add/edit courses in the system
View Course Details	All members can course name/ code/ seats left/ timings
Search and Filter Courses	All members can search all courses in the system and filter them according to department the course belongs too

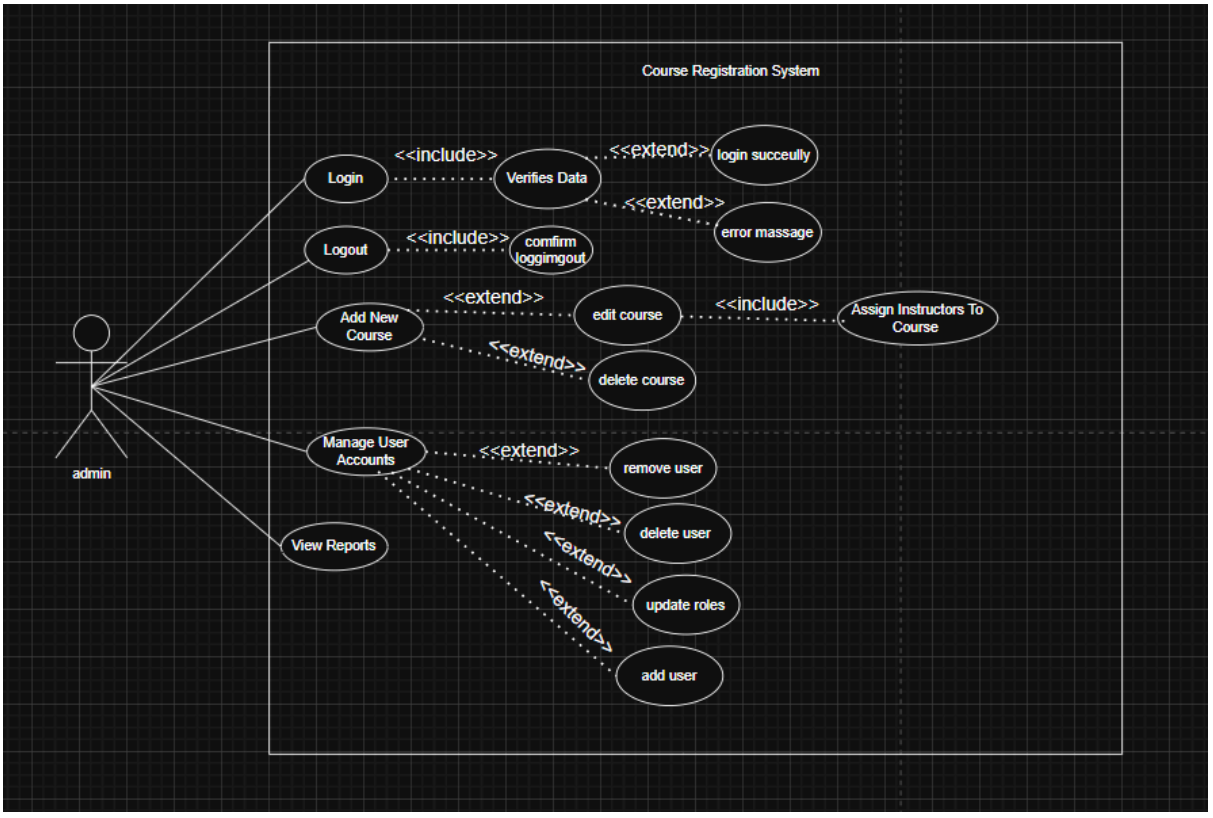
Detecting Conflicts	The system automatically flags courses with the same timings

3.2 Use Case Model









3.2.2 Use Case Descriptions (IEEE Template Example)

Student

Use case ID	UC-01
Use Case Name	login
Primary Action	Authenticate and access the student dashboard.
Stakeholders and intrests	Students
Preconditions	Student has a registered account and is not already logged in.
Postconditions	Student is authenticated and redirected to their dashboard; session is initiated.
trigger	Enters login info and press submit
Main success scenario	1. Student opens the system. 2. Student enters username/email and password. 3. System validates credentials. 4. System establishes a session and loads the student dashboard.
Special requirments	Already registered
Priority	high
Frequency of use	high

Use case ID	UC-02
Use Case Name	Register for course
Primary Action	Enroll in an available course that fits schedule and prerequisites.
Stakeholders and intrests	Students can register for the course and instructors and TAs can see who enrolled

Preconditions	Student is in the same major the course is offered in and registration period is open and no conflicts
Postconditions	Course is added to student's schedule and added to the lost of enrolled students to the course and the number of course seats decreases
trigger	Select course and press register
Main success scenario	1. Student navigates to 'Available Courses'. 2. System displays list of courses with capacity, schedule, and prerequisites displayed on each 3. Student selects a course to register. 4. System verifies seat availability and conflicts. 5. System confirms enrollment and updates the student schedule.
Special requirments	Already registered user and course is in same major student is enrolled in
Priority	high
Frequency of use	high

Use case ID	UC-03
Use Case Name	View Course Content
Primary Action	View Assignments and posts from course
Stakeholders and intrests	Students registered for the course and instructors and TAs can post announcements and assignments
Preconditions	Student is logged in and enrolled in the course.
Postconditions	Requested course content is displayed.
trigger	Opening course

Main success scenario	1. Student opens 'My Courses' and selects a course. 2. System shows posts or assignments . 3. Student opens a specific announcement 4. System rendersthe full post
Special requirments	Already registered to the course
Priority	high
Frequency of use	high

Use case ID	UC-04
Use Case Name	Submit assignment
Primary Action	Student submits assignments
Stakeholders and intrests	Students submits and Instructor or TA grades the submission
Preconditions	Student is logged in and enrolled in the course and assignment is published
Postconditions	Submission is stored with timestamp
trigger	Clicking submit on assignment
Main success scenario	1. Open assignment 2. Upload submission 3. Press submit 4. Submission stored
Special requirments	Submission is within type constraints (pdf and size constraint)
Priority	high
Frequency of use	high/per assignment

Use case ID	UC-05
Use Case Name	View Gades
Primary Action	Student views grades
Stakeholders and intrests	Students gets grade, Instructor/TA send grade
Preconditions	Assignment is posted and has submission
Postconditions	Students views grade in grades tab
trigger	TA/Instructor sends grades
Main success scenario	1.Teacher sends grade 2.Student opens course 3. Opens grades tabs in course 4.view grades
Special requirments	Sent a submission
Priority	high
Frequency of use	high/per assignmnet

Instructor

Use case ID	UC-06
Use Case Name	login
Primary Action	Authenticate and access the instructor dashboard.
Stakeholders and intrests	instructor
Preconditions	instructorhas a registered account and is not already logged in.
Postconditions	Instructor is authenticated and redirected to their dashboard; session is initiated.

trigger	Enters login info and press submit
Main success scenario	1. Instructor opens the system. 2. Instructor enters username/email and password. 3. System validates credentials. 4. System establishes a session and loads the student dashboard.
Special requirments	Already registered
Priority	high
Frequency of use	high

Use case ID	UC-07
Use Case Name	Manage Courses
Primary Action	Instructor can see all courses they teach, TAs under them, course timings,students enrolled in each course
Stakeholders and intrests	Instructor, TAs, students
Preconditions	Logged in
Postconditions	Instructor can preform actions per course
trigger	Instructor opens Courses Tab
Main success scenario	1. Instructor opens the courses tab 2. Can now view everything
Special requirments	Teaches the course
Priority	high
Frequency of use	high/everyday

Use case ID	UC-08
Use Case Name	Create assignment
Primary Action	Posts Assignment
Stakeholders and intrests	Instructor, TAs and students
Preconditions	Instructor is logged in and teaches the course
Postconditions	Assignment is posted and visible on wall
trigger	Write description and press post
Main success scenario	1. Instructor opens course “wall” 2. Press create assignment writes its description 3. Post assignment
Special requirments	Already teaches the course
Priority	high
Frequency of use	medium/weekly

Use case ID	UC-09
Use Case Name	Create post
Primary Action	Posts announcement
Stakeholders and intrests	Instructor, TAs and students
Preconditions	Instructor is logged in and teaches the course
Postconditions	announcement is posted and visible on wall
trigger	Write description and press post

Main success scenario	1. Instructor opens course “wall” 2. Press create post , writes its description 3. Post announcement
Special requirments	Already teaches the course
Priority	high
Frequency of use	medium/weekly

Use case ID	UC-10
Use Case Name	Grade assignment
Primary Action	Posts Assignment grade to student
Stakeholders and intrests	Instructor, TAs and students and admins
Preconditions	Instructor is logged in and teaches the course
Postconditions	Grade is published to student and visible by them
trigger	Give grade and press send
Main success scenario	1. Instructor opens course “wall” 2. Opens assignment 3. Opens submissions 4. Send grade
Special requirments	Already teaches the course
Priority	high
Frequency of use	medium/weekly

Teaching Assistant

Use case ID	UC-11
Use Case Name	login
Primary Action	Authenticate and access the Assistant dashboard.
Stakeholders and intrests	Assistant
Preconditions	Assistant has a registered account and is not already logged in.
Postconditions	Assistant is authenticated and redirected to their dashboard; session is initiated.
trigger	Enters login info and press submit
Main success scenario	1. Assistant opens the system. 2. Assistant enters username/email and password. 3. System validates credentials. 4. System establishes a session and loads the student dashboard.
Special requirments	Already registered
Priority	high
Frequency of use	high

Use case ID	UC-12
Use Case Name	Grade assignment
Primary Action	Posts Assignment grade to student
Stakeholders and intrests	Instructor, TAs and students and admins
Preconditions	Instructor is logged in and teaches the course

Postconditions	Grade is published to student and visible by them
trigger	Give grade and press send
Main success scenario	1. Instructor opens course “wall” 2. Opens assignment 3. Opens submissions 4. Send grade
Special requirments	Already teaches the course,and has permission from instructor
Priority	high
Frequency of use	medium/weekly

Use case ID	UC-13
Use Case Name	Create post
Primary Action	Posts announcement
Stakeholders and intrests	Instructor, TAs and students
Preconditions	Assistant is logged in and teaches the course
Postconditions	announcement is posted and visible on wall
trigger	Write description and press post
Main success scenario	1. Instructor opens course “wall” 2. Press create post , writes its description 3. Post announcement
Special requirments	Already teaches the course,and has permission from instructor
Priority	high

Frequency of use	medium/weekly

Use case ID	UC-14
Use Case Name	Create assignment
Primary Action	Posts Assignment
Stakeholders and intrests	Instructor, TAs and students
Preconditions	Assitant is logged in and teaches the course
Postconditions	Assignment is posted and visible on wall
trigger	Write description and press post
Main success scenario	1. Instructor opens course “wall” 2. Press create assignment writes its description 3. Post assignment
Special requirments	Already teaches the course,and has permission from instructor
Priority	high
Frequency of use	medium/weekly

Admin

Use case ID	UC-15
Use Case Name	login
Primary Action	Authenticate and access the admin dashboard.

Stakeholders and intrests	admin
Preconditions	admin has a registered account and is not already logged in.
Postconditions	admin is authenticated and redirected to their dashboard; session is initiated.
trigger	Enters login info and press submit
Main success scenario	1. admin opens the system. 2. admin enters username/email and password. 3. System validates credentials. 4. System establishes a session and loads the student dashboard.
Special requirments	Already registered
Priority	high
Frequency of use	high

Use case ID	UC-16
Use Case Name	Manage users
Primary Action	View all users, create/edit user
Stakeholders and intrests	admin and all users
Preconditions	admin is already logged in.
Postconditions	admin is directed to their dashboard; and can view all the users and all actions for each
trigger	Opens “manage users “ tab
Main success scenario	1. Admin Searchs for a user 2. Edits their info

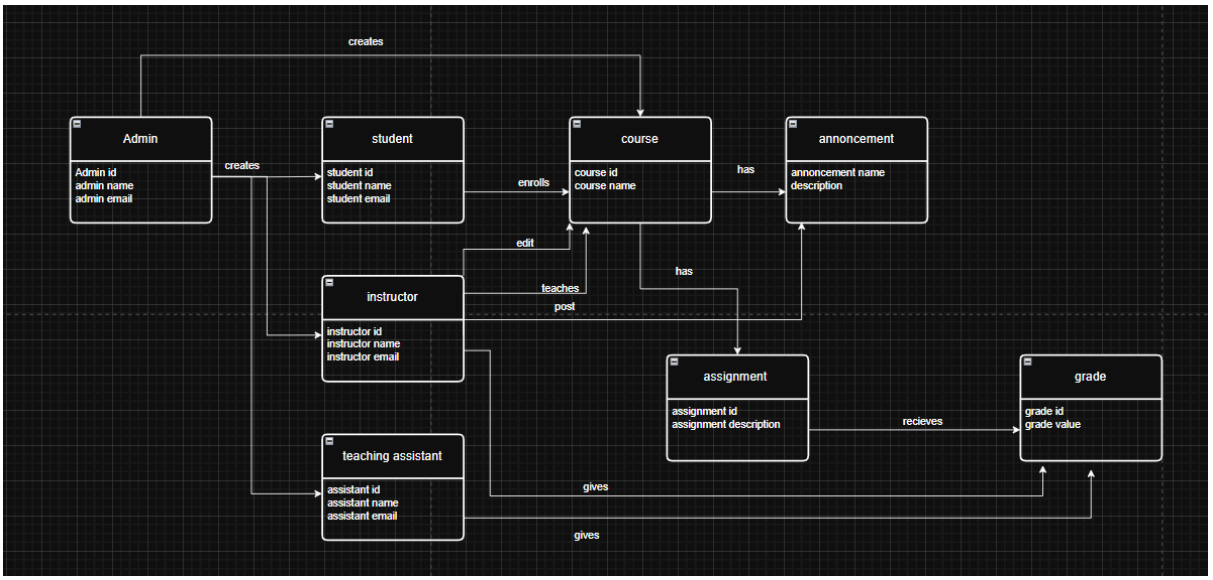
	Or 1.Admin press create user 2. Assign them a role and press create
Special requirments	Already registered with admin role
Priority	high
Frequency of use	high/daily

Use case ID	UC-17
Use Case Name	Manage courses
Primary Action	View all courses , create courses
Stakeholders and intrests	admin and all users
Preconditions	admin is already logged in.
Postconditions	admin is directed to their dashboard; and opens the course search bar
trigger	Opens “courses “ tab
Main success scenario	1. Admin Searchs for a course 2. Can view all students enrolled in the course 3. Can view grades of each student enrolled per course Or 1.Admin press create course 2. Assign them a instructor and TA Or 1. Admin Searchs for a course 2. Press edit course

Special requirments	Already registered with admin role
Priority	high
Frequency of use	high/daily

3.3 Domain Model

3.3.1 Conceptual class diagram



3.3.2 Class Descriptions

Class	Description
User	Base class for all system users.
Student	Can register/drop courses and view schedules.
Instructor	Can update course schedules, post announcements, post assisgnments, give grades, give TAs permissions
Teaching Assistant(TA)	Can view all students, grade their assignments and communicate directly with students and instructors
Admin	Manages all users and courses.
Course	Contains course details and schedule.
Registration	Represents a student's enrollment record makes sure a student can only enroll to courses in his same major
EmailReminder	Handles notification scheduling.

3.4 Non-Functional Requirements

Type	Description	Testing Method	Success Criteria
Usability	The interface must be simple and easy to navigate.	User testing	90% of users perform tasks without help.
Performance	The system should load any page within 3 seconds.	Load testing	Response time ≤ 3s.
Security	Passwords must be encrypted.	Code review, Penetration testing	No plain-text passwords found.
Reliability	System uptime ≥ 95%.	Monitoring	Less than 5% downtime per month.
Portability	Runs on major browsers.	Cross-browser testing	Works on Chrome, Edge, Firefox.

3.5 External Interface Requirements

3.5.1 User Interface

1. Simple, responsive web interface.
2. Login and registration pages with validation messages.
3. Tabs for everyone:
    a. Notifications
    b. Settings
    c. Send an email to…
4. Tabs for students:
    a. Courses Registered
        i. Drop course
        ii. View details
        iii. View assignments
            1. Submit assignment
        iv. Grades (in each course)
    b. Search for Courses
        i. Apply filters
        ii. Add course
5. Tabs for Instructor:
    a. Courses taught
        i. Edit Course
        ii. Add TA
            1. Give TA permissions
        iii. View Enrolled students

    b. Post Assignment
        i. View submissions
            1. Give grade
    c. Post Announcment
6. Tabs for Admins:
    a. View All courses
    b. Add Course
        i. Assign instructor
    c. Add user
        i. Assign role
        ii. Edit user
        iii. Delete user
    d. View All students
        i. View transcript
7. Tabs for TA:
    a. Courses Taught
        i. View all students
        ii. Post assignments
            1. View submissions
            2. Give grades
        iii. Post announcements

3.5.2 Hardware Interface

- Works on any standard computer or mobile device with internet access.

3.5.3 Software Interface

- HTML/CSS for frontend.
- Flask for backend
- Email API for sending notifications.

3.5.4 Communication Interface

- All data sent via HTTPS protocol.

4. Appendices

4.1 Appendix A: Data Dictionary

Data Item	Type	Description
userID	**Integer**	**Unique ID for user**
Name	**string**	**User's full name**
Email	**String**	**University email**
Password	**String**	**Encrypted password**

courseId	**String**	**Unique ID for course**
schedule	**string**	**Lecture days/times**
reminderTime	**DateTime**	**Time for sending reminders**

4.2 Appendix B: Glossary

Term	Definition
Schedule	Weekly timetable showing class times.
Reminder	Email notification before a class.
Registration	The process of enrolling in a course.
Admin	System manager responsible for configurations.
Instructor	Faculty member teaching a course.

Conclusion:

The full functional and non-functional requirements are defined in this SRS document.

for the Smart Course Registration System (SCRS). It provides a clear view of

what our system will perform, and how various users will interact with it. This

This document will serve as the base for the design and implementation phases of

the project