CSAI 203 - Fall 2025

EduTrack

Software Requirements Specification

Team Number:29

Team Members:3

- *Habiba Mohammed-202402213*
- *Ahmed Mahmoud-202401491*
- *Mohamed Essam-202401171*

Representative Contact: *s-ahmed.abda@zewailcity.edu.eg*

Date: *8/11/2025*

Table of Contents

1. Introduction

1.1 Purpose

The goal of this Software Requirement Specification (SRS) is to specify all functional and non-functional requirements for EduTrack - Smart Classroom. EduTrack is designed to improve communication between students and teachers by providing an all-in-one solution that allows teachers to upload lectures, track attendance, assign grades, and communicate with students through a built-in chat system. It will allow students to view materials, monitor their performance, and communicate efficiently.

1.2 Scope

The features provided by EduTrack include:

- Authentication and role based access (Admin, Teacher, Student)
- Course Lifecycles (Create, enroll, manage)
- Upload lectures (PDF/video link) and download files
- Post assignments or upload assignments, submissions, grades and feedback
- Take attendance on a per session basis
- Course chat
- Dashboards and notifications

The system is intended for university/college courses and runs in web browsers which uses a Flask backend and SQL database.

1.3 Definitions, Acronyms, and Abbreviations

Flask	Lightweight Python web framework for backend development.
MVC	Model–View–Controller pattern separating logic, interface, and data.
CRUD	Create, Read, Update, Delete – core database operations.
UI	User Interface – what the user interacts with visually.
DB	Database used for storing application data.

1.4 References

- Course Project Guidelines (Fall 2025), Dr. Mohamed Sami Rakha.
- IEEE 830-1998 Standard for Software Requirements Specification.
- UML IEEE Use Case Description Template (lecture slides).

1.5 Overview

This document contains a high-level overview, detailed functional requirements complete with scenarios, a use case model and IEEE-style use case descriptions, a domain model (conceptual class diagram), non-functional requirements with test plans, an external interfaces section, and appendices (data dictionary + glossary).

2. Overall Description

2.1 Product Perspective

EduTrack is a web app, developed as a standalone application and using MVC as its architecture. There is a backend (Flask) that exposes the routes for both pages and page actions. The Models serve as maps to DB tables while the Views are simply HTML templates. There is no JavaScript used in the project so pages will refresh on POST actions.

2.2 Product Functions

- Authentication and user management (register, login, passwords recovery - etc).
- Course management (create, update, delete, join via code).
- Upload/access lecture materials (class recordings via PDF files or video links).
- Assignment creation, submission, grading, feedback etc.
- Record attendance for each session and date for each class.
- Class chat (save messages in database, so page will refresh).
- Dashboards for teachers and students, simulated notifications that look like banners.
- Admin panel for managing users/courses/logs etc.

2.3 User Classes and Characteristics

Admin	has total control (can create/delete users, see logs).
Teacher	creates/manages courses, uploads lectures, takes attendance, and grades.

Student	signs up for courses, views lectures, submits assignments, sees grades and attendance, uses chat

2.4 Operating Environment

Python, Flask, SQL. Deploy to a server. Browsers: Chrome, Firefox, Edge..

2.5 Design and Implementation Constraints

Flask + HTML + CSS

MVC is mandatory.

Limited file upload (e.g., <= 10MB).

2.6 User Documentation

Quick-start guide: install, run server, create admin, create course, upload lecture, mark attendance, create assignment, grade, use chat.

Basic user help pages embedded.

2.7 Assumptions and Dependencies

Users have internet and browsers. The server environment supports Flask and database. File storage available on server.

3. Specific Requirements

3.1 Functional Requirements

Each requirement includes description, scenarios, alternative flows, and success criteria.

FR1 _ User Registration & Authentication Description: Users can register an account and login. Role assigned: student, teacher, admin (admin account may be seeded). Passwords stored hashed. Main scenario: Register → validate info → create record → login → dashboard. Alternative: wrong email/weak password → error. Success: hashed password stored, login success. Test: register valid/invalid user, dump DB, check login.
FR2 _ Admin User Management Description:Admin can create/delete/edit users and reset passwords for users. Main Scenario:An admin logs in → goes to the Admin Panel → selects to create a new Teacher user or edits an existing one by providing an email and name → the system creates the user in the backend. Alternate:An admin attempts to delete a user that doesn't exist → the system displays an error. Test:Create test users; delete users; check users have been deleted.

FR3 _ Course Management

Description: Teachers create their own courses (with title, code, description), can manage who is a participant and close course completions.

Main Scenario: Teacher logs in → clicks "Create Course".

They fill in their title, description, code (it will generate a code if none is provided).

The course will not display until it is saved. The teacher will be the owner of the course.

Students will join by using the generated code.

Alternate:The information for the course can be edited, a course can be deleted (but the teacher will have to confirm).

Test:Create Course, student uses course code to join.

FR4 _ Lecture Upload and Management

Description: The teacher will upload files (PDF) and/or paste links to videos for the students to view as part of the lecture.

Main Scenario:The teacher clicks the "Upload Lecture" button in the course page.

The teacher uploads a PDF file *(<10MB)* or enter a link to a video.

The system saves the uploaded lecture information and a file path for future access purposes.The students see the lecture in the course materials.

Alternate: for unsupported file types-the system will reject; for files that fail to upload-the user is able to re-upload again.

Testing: Upload valid file; attempt to upload a file that exceeds size limitations; verify that student can download.

FR5_ Assignment Construction and Submission

Description: The teacher will construct an assignment title, description, due date, and maximum assignment grade to be submitted by the student files before the due date.

Main Scenario:Teacher posts an assignment in the course with a due date and maximum assignment grade.Student visits the assignment page → uploads their assignment file (PDF) to the assignment page and submits.

Alternate: The assignment will allow either no late submissions or be able to allow or block late submissions in accordance with the policy; assignment will allow submission if enabled.

Testing: Submit assignment before the due date and after the due date; verify a timestamp appears; teacher views list of submissions.

FR6_ Grading and Feedback

Description: Using this feature, the Teacher can evaluate students submissions by assigning grades and textual feedback. The students then receive and can view their respective results.

Main scenario:Teacher can view and access submissions → Teacher selects a submission for a specific Student → Teacher provides a grade and written feedback → Teacher can save results of the submission.

The student receives a notification that a new grade was added. The student views their grade on their dashboard and assignment page.

Alternate: Teacher opens a student's submission → Teacher enters the grade but leaves the feedback field empty (or vice versa) → Teacher attempts to save the results → The system displays an error message indicating that both the grade and feedback are required → Teacher completes the missing field(s) and saves again → The system successfully saves the results and the student receives a notification as in the main scenario

Test: Teacher completes grading of submission; student sees grade updated; the grade remains when the page is refreshed.

FR7_Attendance Reporting

Description: Teacher will mark attendance for each course session (date). Attendance status can be marked Present/Absent/Excused.

Main scenario:Teacher opens course → clicks "Record Attendance" → system lists enrolled students.Teacher marks Present/Absent → saves.Attendance records are saved per student per date.

Extensions: Marking all students as present on one click; export report of student attendance.

Alternate:Teacher forgets to mark attendance for one or more students → system shows an alert that all students must have a status → Teacher completes missing marks → saves successfully.

Test: Mark attendance of the class, check history of student's attendance and total percentage of presence.

FR8 — Course Chat

Description: Chat functionality at the course level where any user currently enrolled can post messages.

Main scenario:User opens chat → correctly shows list of messages in timestamp order.User types message → submits form → server saves message to DB → user is redirected to the same page with messages refreshed.

Alternate: Message submits blank then error; there is an issue with the DB, message tries to send again.

Test: Post messages first from a student then teacher, ensuring messages save and show.

FR9_Dashboard & Notifications

Description: Role-based dashboard with courses, upcoming assignments, unread notifications, attendance summary, and average grade.

Main scenario:User logs in → dashboard shows courses, assignments, notifications, attendance, and grades → user clicks a notification → system opens relevant page.

Alternate:Some dashboard data fails to load → system shows a placeholder → user refreshes → data loads successfully

Test: Create assignment & grade → check notification; simulate upcoming due.

FR10 — File Download & Storage

Description: Students or teachers download lecture files

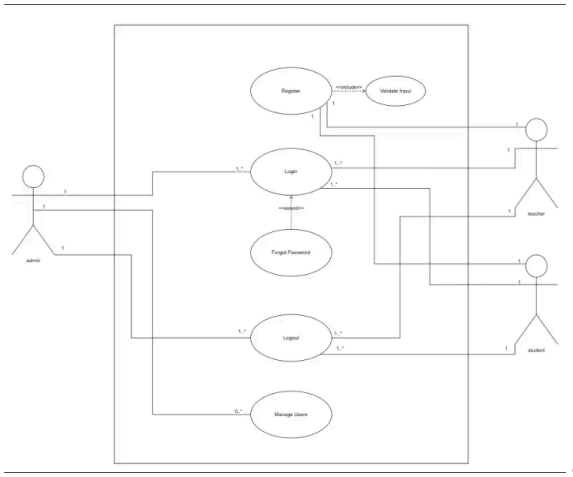
Main scenario:User opens course → selects a file → click download → file downloads successfully.

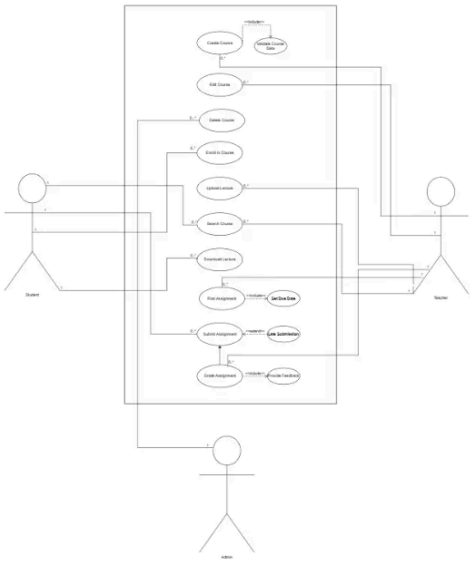
Alternate:User selects a file → network or server error occurs → system displays “Download failed” → user retries → file downloads successfully.

Test:enrolled students can download file.

3.2 Use Case Model

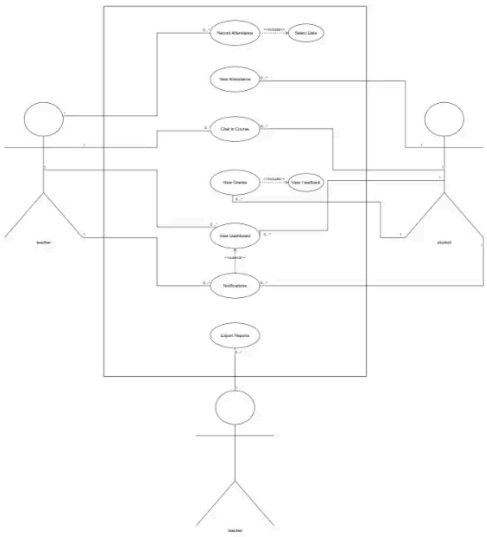
3.2.1 Use Case Diagrams





use case diagram 1

use case diagram 2



use case diagram 3

3.2.2 Use Case Descriptions (IEEE template)

Template Section	Purpose / Notes
Use Case ID	UC1
Use Case Name	User Login
Primary Actor	Student / Teacher / Admin
Stakeholders & Interests	Users need secure and reliable access to their dashboards.Admin requires authentication logs and system security.
Preconditions	- User has a valid registered account.- Server is running and reachable.
Postconditions	**Success:** User session created and redirected to appropriate dashboard.**Failure:** No session created; error message displayed.
Main Success Scenario	1. The user opens the login page. 2. The user enters email and password. 3. The system validates credentials . 4. The system creates session and redirects user to the appropriate dashboard (based on role).
Extensions (Alternative Flows)	3a. Invalid credentials → system displays an error message and allows retry. 3b. Account inactive → system shows “Contact Admin” message.
Includes / Extends	**Includes:** Validate Input.**Extends:** Forgot Password.
Special Requirements	- Passwords stored securely. - Login attempts are limited.
Assumptions	Stable network connection available.
Frequency of Use	Multiple times daily by each active user.

Use Case ID	**UC2**
Use Case Name	Create Course
Primary Actor	Teacher
Stakeholders & Interests	Teachers want to create structured courses. Students want correct course info.
Preconditions	Teacher authenticated.
Postconditions	Course record stored; teacher assigned as owner.
Main Success Scenario	Teacher opens "Create Course".Inputs title, description, optional code.Submits; system validates and saves.System shows success and course page.
Extensions	Duplicate title → warning.Invalid data → show validation errors.
Includes / Extends	Includes: Validate Course Data.Extends: Send Notification (to enrolled students when updated).
Special Requirements	Course code is unique.
Assumptions	The teacher has permission.
Frequency	Weekly or per semester.

Use Case ID	**UC3**
Use Case Name	Upload Lecture
Primary Actor	Teacher
Stakeholders & Interests	Teachers upload materials; students download.
Preconditions	Teacher authenticated and course exists.
Postconditions	Lecture record & file stored; visible to enrolled students.

Main Success Scenario	Teacher selects course → Upload Lecture.Chooses file or enters video URL.System validates file type/size, stores file/link.Success notification.
Extensions	File too large → reject.Unsupported file type → reject.
Includes / Extends	Includes: File Validation.Extends: Notify Students.
Special Requirements	Max upload 10MB. Virus-scan in future versions.
Assumptions	Storage space available.
Frequency	Per lecture (weekly).

Use Case ID	**UC4**
Use Case Name	Post Assignment
Primary Actor	Teacher
Stakeholders & Interests	Teachers publish tasks; students submit.
Preconditions	Teacher authenticated; course exists.
Postconditions	Assignment stored with due date.
Main Success Scenario	Teacher creates assignment with title, description, due date → system saves → students notified (simulated).
Extensions	Missing fields → show error.Past due date → warn.
Includes / Extends	Includes: Validate Assignment Data.Extends: Send Notification.
Special Requirements	Date format fixed (YYYY-MM-DD).

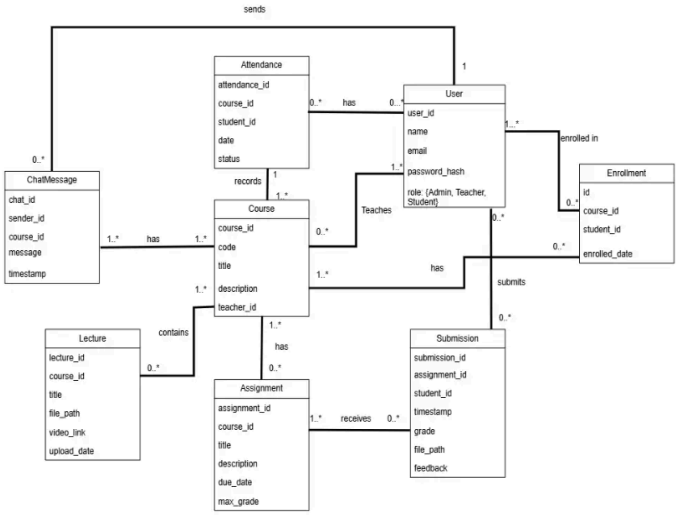
Use Case ID	**UC5**
Use Case Name	Submit Assignment
Primary Actor	Student
Stakeholders & Interests	Student wants to submit on time; teacher needs to grade.
Preconditions	Student authenticated & enrolled; assignment open.
Postconditions	Submission stored with timestamp.
Main Success Scenario	Student uploads file → system records timestamp & file link.
Extensions	Late submission handled based on course policy; resubmission allowed if enabled.
Includes / Extends	Includes: File Validation.Extends: Late Submission policy.
Special Requirements	Accept only allowed file types (pdf/doc).

Use Case ID	**UC6**
Use Case Name	**Record Attendance**
Primary Actor	**Teacher**
Stakeholders	**Students want attendance record; admin may need reports.**
Preconditions	**Course exists, students enrolled.**
Postconditions	**Attendance record saved for date.**

Main Flow	**Teacher selects date → marks present/absent → save.**
Extensions	**Bulk mark all present/absent; correct previous entries.**

3.3 Domain Model

3.3.1 Conceptual Class Diagram



class diagram

Main classes & key attributes:

- **User**(user_id, name, email, password_hash, role)
- **Course**(course_id, title, code, description, teacher_id)
- **Enrollment**(id, course_id, student_id, enrolled_date)
- **Lecture**(lecture_id, course_id, title, file_path, video_link, upload_date)
- **Assignment**(assignment_id, course_id, title, description, due_date, max_grade)
- **Submission**(submission_id, assignment_id, student_id, file_path, timestamp, grade, feedback)
- **Attendance**(attendance_id, course_id, student_id, date, status)
- **ChatMessage**(chat_id, course_id, sender_id, message, timestamp)

Multiplicity examples:

- 1..* ***Teachers*** *(User with role=teacher) — creates → 0..** **Course**.
- 1..* **Course** — has 0..* **Lecture**, 0..* **Assignment**, 1..* **ChatMessage**.
- 1..* **Student** — may have 0..* **Submission** and 0..* **Attendance** records.

3.3.2 Class Descriptions

1**. User**

- **Attributes:** user_id, name, email, password_hash, role
- **Responsibilities:** Represents a system user (Student, Teacher, or Admin). Manages authentication and role-based access.
- **Constraints:** email must be unique; role defines access permissions.

2. Course

- **Attributes:** course_id, title, code, description, teacher_id
- **Responsibilities:** Represents a course offered in the system. Links to a Teacher and enrollments.

  Constraints: code must be unique; teacher_id references a User with role=Teacher.

3. Enrollment

- **Attributes:** id, course_id, student_id, enrolled_date
- **Responsibilities:** Tracks which students are enrolled in which courses.
- **Constraints:** A student can only enroll once per course.

4. Lecture

- **Attributes:** lecture_id, course_id, title, file_path, video_link, upload_date
- **Responsibilities:** Represents course materials (files/videos) uploaded by the teacher.

  Constraints: Each lecture must belong to a valid course.

5. Assignment

- **Attributes:** assignment_id, course_id, title, description, due_date, max_grade
- **Responsibilities:** Represents tasks assigned to students for evaluation.
- **Constraints:** Each assignment belongs to one course; due_date cannot be in the past at creation.

6. Submission

- **Attributes:** submission_id, assignment_id, student_id, file_path, timestamp, grade, feedback
- **Responsibilities:** Represents a student's submission for an assignment, including evaluation results.
- **Constraints:** Each student can submit once per assignment (or multiple if allowed by system rules).

7. Attendance

- **Attributes:** attendance_id, course_id, student_id, date, status
- **Responsibilities:** Tracks attendance for each student per course session.
- **Constraints:** One record per student per course per date; status can be Present, Absent, or Excused.

8. ChatMessage

- **Attributes:** chat_id, course_id, sender_id, message, timestamp
- **Responsibilities:** Represents a message sent in a course chat by a user.
- **Constraints:** sender_id must be enrolled in the course; messages are timestamped.

3.4 Non-Functional Requirements

1. **Usability**
    - *Requirement:* Interface must be intuitive; typical new user can perform basic tasks (login, view course) within 5 minutes.
    - *Test:* 3 test users follow a checklist; record success/time.
    - *Success Criteria:* ≥ 80% of tasks completed by 3 users within 5 minutes.
2. **Performance**
    - *Requirement:* Core pages (dashboard, course page) load under 2 seconds under normal load.
    - *Test:* Manual timing with a cold server + simple load test.
    - *Success Criteria:* Average page load ≤ 2s.

3. **Security**
    - *Requirement:* Passwords stored hashed; session protected.
    - *Test:* Code review & try to login with wrong creds; verify hashed password in DB.
    - *Success Criteria:* No plaintext passwords; invalid login blocked; sessions expire after inactivity (configurable).
4. **Reliability**
    - *Requirement:* System must handle continuous usage for at least 1 hour without crash.
    - *Test:* Leave server running and perform representative operations for 1 hour.
    - *Success Criteria:* No application crashes; no unhandled exceptions.
5. **Maintainability**
    - *Requirement:* Code organized in modules (blueprints) and documented.
    - *Test:* Code walkthrough and measure Cyclomatic complexity roughly (manual review).
    - *Success Criteria:* Project split into logical modules: auth, courses, assignments, attendance, chat, admin.
6. **Storage Constraint**
    - *Requirement:* File uploads limited to 10MB per file.
    - *Test:* Upload files of various sizes.
    - *Success Criteria:* Files >10MB rejected; <=10MB accepted.

3.5 External Interface Requirements

3.5.1 User Interface

HTML templates: Login, Register, Dashboard (student/teacher), Course page (materials/lectures), Assignment page (post/submit), Attendance page, Chat page, Admin panel. Screens are simple forms & tables.

3.5.2 Hardware Interface

Standard web server any PC. No special hardware.

3.5.3 Software Interface

Flask application using SQL. File system for lecture/assignment storage.

3.5.4 Communication Interface

HTTP (Flask routes) over localhost; production may use HTTPS.

---

4. Appendices

4.1 Appendix A: Data Dictionary

Entity	**Attribute**
User	user_id
User	name
User	email
User	password_hash
Course	course_id
Course	title
Course	code

Enrollment	id
Assignment	assignment_id
Submission	submission_id
Attendance	attendance_id
ChatMessage	chat_id

4.2 Appendix B: Glossary

- **Assignment:** Task posted by teacher with due date.
- **Submission:** Student’s uploaded answer for an assignment.
- **Attendance record:** Present/Absent per student per date.