Syrian Private University Faculty of Engineering Department of Software & Information Systems Engineering

Online Examination System with AI Assistant

Prepared by

Ismail Abdulrahim

Ali Alainea

AbdulmajedAlkhousi

Supervised by

Eng.SaeedDiab

2025-2026

SUPERVISOR CERTIFICATION

I Certify that the preparation of this project entitled……,prepared by……,was made under my supervision at FacultyOf Computer, Informatics & Communication Engineering in partial Fulfillment of the Requirements for the Degree of Bachelors of Software and Information System Engineering.

Name:……..Signature:……..Date:……

Abstract

This paper presents the design and implementation of an intelligent online examination system titled **“Online Examination System with AI Assistant.”** The system provides a secure and structured digital environment that supports the complete examination lifecycle, including exam creation, delivery, monitoring, and evaluation. It is designed to serve administrators, teachers, and students through a role-based web platform that ensures controlled access and organized workflows.

The system incorporates authentication with email-based verification, role-based access control, and client-side anti-cheating mechanisms, including window and tab activity detection, event logging, and copy-paste restrictions during examinations. These mechanisms enable the detection and recording of suspicious behaviors, which are presented to instructors through dedicated monitoring dashboards. In addition, the platform integrates an AI-assisted question generation module powered by **Ollama**, allowing instructors to efficiently generate exam questions based on predefined criteria and existing educational content.

From a usability perspective, the system offers interactive role-based dashboards, integrated notifications, and built-in messaging and reporting features that facilitate effective communication between users. The system was developed using an incremental and iterative development methodology, enabling progressive implementation, early validation, and continuous refinement. Overall, the proposed solution combines practical exam management with intelligent content generation and behavioral monitoring, making it well-suited for modern digital assessment environments.

CONTENTS:

Chapter One: Introduction and Project Definition..... 12

1- Introduction: 13

2- Problem Definition:.....13

3- Project Objectives: 14

4-Aim Of The Project: 14

5-How Do We Solve The Problem:.....15

6-Project Management: 15

6.1-Human Resources: 16

6.2-Major Tasks:.....16

6.3-Theoretical and Reference Phase: 16

6.4-Planning Phase of the Project: 17

6.5-Analytical Study Phase: 17

6.6-System Design Phase: 18

6.7-Testing Phase: 18

Chapter Two: Theoretical Study..... 19

1-Similar Applications:.....20

4.1-Exam Website: 20

4.2- Testinvite Website: 23

4.3-FlexiQuiz Website: 24

Chapter Three: 26

1- Introduction: 28

Examination System.....28

- **PROJECT DESCRIPTION:**.....28

- **PROJECT OBJECTIVES:** 29
- **PROJECT SCOPE:** 29
- **PROJECT CONSTRAINTS:**....29
- **PROJECT TECHNOLOGY AND TOOLS:** 30

4- Software Requirements Specification (SRS): 32

4.1- Project Title: "Examination System". 33

4.2-Intruduction: 33

4.2.1-Problem Definition: 33

4.2.2-purpose: 33

4.2.4-System Description: 38

4.3-NonFunctional Requirements: 39

4.4-Functional Requirements: 37

5.1- Use Case Diagram: 5.1.1-Level use case: 39

5-Unified Modeling Language: :(UML) Diagrams: 39

5.2-Use Case Specification: 45

5.3- Activity Diagram: 62

5.4-Entity Relation Diagram: 73

5.5-Sequence Diagram: 114

6-DATA DICTIONARY: 131

7-TEST CASES: 131

8-Requirements Traceability Matrix (RTM): 89

9-SYSTEM ARCITECTURE: 134

9.1-MVT ARCHITECTURE: 143

9.2-INTRODUCTION TO MVT AND ITS IMPORTANCE: 144

Chapter Four: 147

1- Introduction: 148

2- Class Diagram: 149

3- Table Diagram: 149

4- App Interfaces: 100

Chapter Five:.....166

1- INRTODUCTION:.....167

2- TOOLS:.....167

4- TEST CASES EXCUTION: 169

5- UPDATE RTM: 175

REFERENCES:.....181

LIST OF FIGURES:

Figure 2.1-website Exam.....21

Figure 2.2-website Testinvite.....23

Figure 2.3-website FlexiQuiz.....25

Figure 3.1-WBS.....29

Figure 3.2-Use Case FOR SYSTEM-LEVEL 1.....39

Figure 3.3-Use Case FOR SYSTEM-LEVEL 2.....40

Figure 3.4-Activity Diagram for 'Login User'.....62

Figure 3.5-Activity Diagram for 'Update Account ADMIN'.....63

Figure 3.6-Activity Diagram for 'Access Status ADMIN'.....64

Figure 3.7-Activity Diagram for 'Create Test TEACHER'.....65

Figure 3.8-Activity Diagram for 'Create Group TEACHER'.....66

Figure 3.9-Activity Diagram for 'Create Question TEACHER'.....67

Figure 3.10-Activity Diagram for 'Manage Student's Status TEACHER'.....68

Figure 3.11-Activity Diagram for 'Delete Account STUDNET'.....69

Figure 3.12-Activity Diagram for 'Manage Status STUDNET'.....70

Figure 3.13-Activity Diagram for 'Send Message STUDNET'.....71

Figure 3.14-Activity Diagram for 'View Student STUDNET'.....72

Figure 3.15-Conceptual ERD.....73

Figure 3.16-Sequence Diagram 'Login User'.....74

Figure 3.17-Sequence Diagram 'Update Account ADMIN'.....75

Figure 3.18-Sequence Diagram 'Manage Access Statues ADMIN'.....76

Figure 3.19-Sequence Diagram 'Create Test TEACHER'.....77

Figure 3.20-Sequence Diagram 'Create Group TEACHER'.....78

Figure 3.21-Sequence Diagram 'Create Question TEACHER'.....79

Figure 3.22-Sequence Diagram 'Manage Student State TEACHER'.....80

Figure 3.23-Sequence Diagram 'Delete Account STUDNET'.....81

Figure 3.24-Sequence Diagram 'Send Message STUDENT'.....82

Figure 3.25-Sequence Diagram 'Delete Message STUDENT'.....83

Figure 3.26-Sequence Diagram 'View Student'.....84

Figure 3.27-MVT.....92

Figure 4.1-Class Diagram.....97

Figure 4.2-Table Diagram.....98

Figure 4.8-Login Page.....99

Figure 4.9-Dashboard ADMIN Page.....100

Figure 4.5-Manage Users ADMIN page.....100

Figure 4.10-Dashboard TEACHER Page.....101

Figure 4.11-Test Page Teacher.....101

Figure 4.12-Chat Page Teacher.....102

Figure 4.13-Dashboard Page Student.....102

Figure 4.14-Update Account Page STUDENT.....103

LIST OF TABLES:

Table 3.1-activity network.....30

Table 3.2-TABLE REQUIREMENTS.....32

Table 3.3-Use Case Specification 'Login User'
.....41

Table 3.4-Use Case Specification 'Logout ADMIN' 42

Table 3.5-Use Case Specification 'Update Account ADMIN' 42

Table 3.6-Use Case Specification 'Delete User ADMIN' 43

Table 3.7-Use Case Specification 'Manage Access Status ADMIN' 44

Table 3.8-Use Case Specification 'View Test ADMIN' 44

Table 3.9-Use Case Specification 'View Group ADMIN' 45

Table 3.10-Use Case Specification 'View Teacher ADMIN' 45

Table 3.11-Use Case Specification 'View User ADMIN' 46

Table 3.12-Use Case Specification 'Manage Students Status TEACHER '.....46

Table 3.13-Use Case Specification 'Update Account TEACHER' 47

Table 3.14-Use Case Specification 'Delete Account TEACHER' 48

Table 3.15-Use Case Specification 'Create Test TEACHER'.....**49**

Table 3.16-Use Case Specification 'View Test TEACHER'.....**49**

Table 3.17-Use Case Specification 'Update Test TEACHER' **50**

Table 3.18-Use Case Specification 'Delete Test TEACHER'.....**50**

Table 3.19-Use Case Specification 'Choose Student TEACHER'.....**51**

Table 3.20-Use Case Specification 'View Student TEACHER'.....51

Table 3.21-Use Case Specification 'Create Question TEACHER'.....**52**

Table 3.22-Use Case Specification 'View Question TEACHER'.....**53**

Table 3.23-Use Case Specification 'Delete Question TEACHER'.....**53**

Table 3.24-Use Case Specification 'Create Group TEACHER'.....**54**

Table 3.25-Use Case Specification ‘View Group TEACHER'……**55**
Table 3.26-Use Case Specification ‘Update Group TEACHER’……**55**
Table 3.27-Use Case Specification ‘Delete Group TEACHER’……56
Table 3.28-Use Case Specification ‘Manage Status STUDENT’……**56**
Table 3.29-Use Case Specification ‘Update Account STUDENT’……**57**
Table 3.30-Use Case Specification ‘Delete Account STUDENT’……**58**
Table 3.31-Use Case Specification ‘View Test STUDENT’……**58**
Table 3.32-Use Case Specification ‘View Teacher STUDENT’……**59**
Table 3.33-Use Case Specification ‘View Group STUDENT’……**59**
Table 3.34-Use Case Specification ‘Enter Chat STUDENT’……**60**
Table 3.35-Use Case Specification ‘Send Message STUDENT’……**60**
Table 3.36-Use Case Specification ‘Delete Message STUDENT’……61
Table 3.37-Use Case Specification ‘View Student STUDENT’……**61**
Table 3.38-TEST CASES……86
Table 3.39-RTM……89
Table 4.1-RTM Update……104
Table 5.1-TEST CASES EXECUTION……110
Table 5.2-Update RTM……114

TABLE OF SHORTCUTS:

Shortcut	definition
ERD	Entity Relation Diagram
SRS	Software Requirements Specification
RTM	Requirements Traceability Matrix
MVT	Model-View-Template
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
Python	Programming Language
SOW	Statement Of Work
WBS	Work Breakdown Structure
PF	Productivity Factor

Chapter One: Introduction and Project Definition

1- Introduction:

This chapter introduces the project by outlining the main motivations and challenges that led to the development of the system. It also presents the objectives and goals that the system aims to achieve. In addition, the chapter provides an overview of the key concepts, technologies, and tools used in the design and implementation of the project.

2- Problem Definition:

Educational institutions increasingly rely on digital tools for conducting assessments; however, many still face challenges due to inefficient and outdated examination systems. Common issues include difficulties in user authentication, unstructured exam creation processes, limited behavioral monitoring during assessments, and complex result management procedures. These challenges place additional administrative burdens on institutions, increase the risk of academic dishonesty, and negatively affect the overall student experience. Without an integrated and role-based online examination system, the assessment lifecycle remains fragmented, insufficiently monitored, and time-consuming.

3- Project Objectives:

This project aims to design and implement an Online Examination System that supports the full lifecycle of electronic examinations in a structured, secure, and efficient manner.

The primary objectives of the system include providing centralized user account management with role-based access for administrators, teachers, and students, as well as enabling subject, question bank, and exam management through well-defined workflows. The system allows teachers to create exams, manage questions both manually and through AI-based question generation, and assign exams to specific student groups.

In addition, the system supports secure exam execution, including time control, attempt tracking, and automatic submission, ensuring fairness and consistency during assessments. Real-time exam monitoring and notification mechanisms are provided to assist teachers in supervising ongoing exams and tracking student activity.

Overall, the system aims to reduce manual administrative workload, improve exam organization and transparency, and ensure accurate data handling throughout the examination process.

4-Aim of the Project:

The primary aim of the project is to address key weaknesses in traditional exam management by implementing a robust, integrated examination system. Specifically, we aim to:

1-Provide role-based access for admins, teachers, and students.

2-Enable secure and structured exam creation and assignment.

3-Improve the transparency, speed, and integrity of the entire exam lifecycle.

4-Integrating AI tools to optimize productivity, speed and accuracy.

5-How Do We Solve The Problem:

To address the identified challenges, we propose implementing several key solutions:

Developing exam creation tools with time settings, question types, and assignment options for teachers.

Enabling secure real-time student participation and automatic result submissions directly linked to their profiles.

Implementing centralized profile and account management to ensure system-wide control and transparency.

6-Project Management:

Project Management for Examination System Website

6.1-Human Resources:

The project team consists of three students from the Faculty of Engineering at Syrian Private University, specializing in Software and Information Systems Engineering.

They are responsible for the design, development, and deployment of the Examination System web application.

6.2-Major Tasks:

The project will be build using the Incremental Methodology.

Major Tasks Are (Planning – Analysis –Design – Implementation -Testing)

6.3-Theoretical and Reference Phase:

Conduct a theoretical study of the educational sector and digital examination systems.

Review existing academic platforms and identify their weaknesses in exam delivery and monitoring.

Define a general framework and scope for the system, including user roles, exam types, and access control mechanisms.

6.4-Planning Phase of the Project:

Establish a development timeline with key milestones and deliverables.

Identify stakeholders including university administrators, instructors, and students.

Gathering system requirements.

Document functional requirements and prioritize features such as exam creation, secure login, and result management.

6.5-Analytical Study Phase:

Analyze the collected requirements to ensure clarity and completeness.

Create Use Case Diagrams to show interactions between different users (admin, teacher, student) and the system.

Define all system functionalities such as profile management, exam scheduling, and real-time result submission.

Build supporting diagrams including Activity Diagrams and Sequence Diagrams to support the design process.

6.6-System Design Phase:

Design the full system architecture using Django's MVT (Model-Template-View) structure.

Develop the database schema.

Create a Class Diagram showing the relationship between major components.

Ensure proper linking and separation of modules to allow secure and scalable development

6.7-Testing Phase:

Conduct unit and integration testing for each module (e.g., login, exam creation, participation).

Ensure the system meets all functional requirements and handles edge cases properly.

Perform performance and load testing under different user scenarios (e.g., multiple concurrent exams).

Document and fix any bugs or exceptions found during testing to ensure a stable and functional system.

Chapter Two:
Theoretical Study

1- Similar Applications:

4.1-Exam.net:

Exam.net is a secure, browser-based online examination platform built for academic institutions such as schools and universities. It provides a simple yet powerful way to conduct digital exams while maintaining integrity and control over the exam environment.

Key Features with Expanded Explanation:

- **Secure Browser Lockdown:**
  Prevents students from switching tabs, using copy/paste, or opening other applications during the exam. Ensures the exam environment is as controlled as a physical classroom.
- **No Student Account Required:**
  Teachers generate a secure code that students use to join the exam, reducing friction and avoiding registration steps.
- **Identity Verification:**
  Supports ID or photo uploads before the exam, helping verify the identity of the test taker.
- **Exam Customization Options:**
  Teachers can upload PDFs, use question editors, set time limits, and randomize questions to fit various assessment needs.

- **Offline Support:**
  Exams can temporarily continue offline if the internet connection drops, minimizing disruptions.
- **Cheating Detection Tools:**
  Detects unusual activity such as multiple window openings and sends reports to the teacher for review.

Exam.net is best suited for schools and universities that require a reliable, secure, and easy-to-use online examination environment without heavy setup requirements.

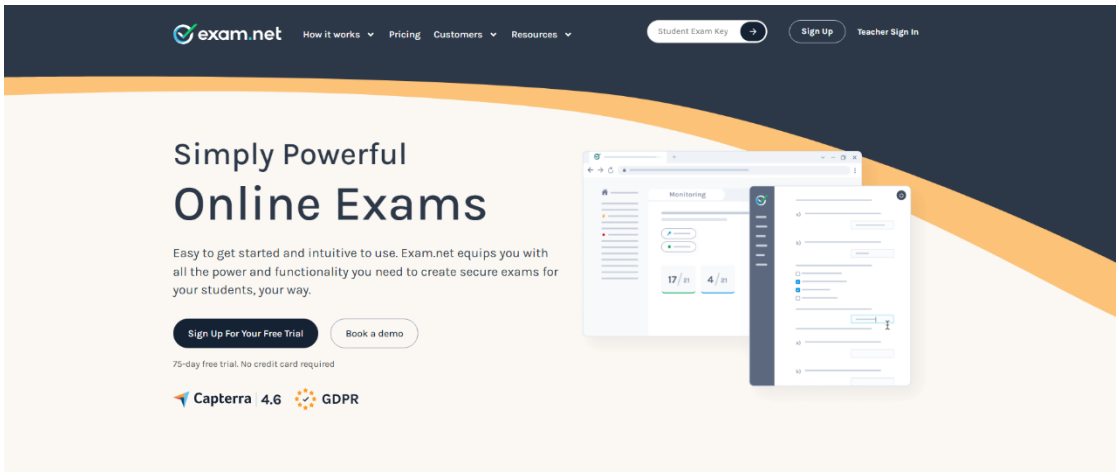


Figure 2.1-website Exam

4.2-TestinviteWebsite:

Testinvite is a professional-grade online assessment platform designed for educational institutions and corporations. It supports everything from individual quizzes to large-scale certification exams.

Key Features with Expanded Explanation:

- **Structured Exam Builder:**
  Offers detailed exam customization including sections, timers, adaptive scoring, and randomization from a question bank.
- **Granular User Roles:**
  Defines access levels for admins, test creators, proctors, and examinees, ensuring secure workflow separation.
- **Advanced Analytics and Reports:**
  Tracks detailed performance metrics, allowing organizations to assess difficulty, success rates, and question quality.
- **Custom Question Banks:**
  Enables building reusable collections of questions categorized by topic, difficulty, or exam type.
- **Secure Environment Options:**
  Includes navigation control, timed access, and anti-cheating settings to enhance test integrity.
- **Bulk Testing Support:**
  Suitable for handling thousands of users simultaneously, with real-time monitoring dashboards.

Testinvite is ideal for organizations seeking a highly configurable testing platform with powerful reporting and enterprise features.

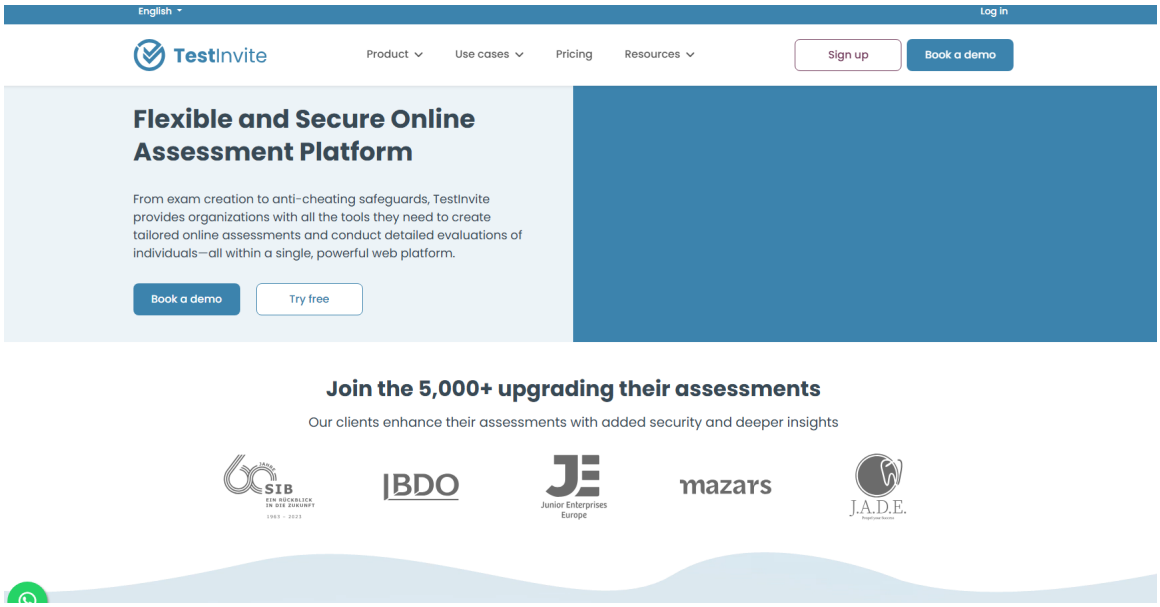


Figure 2.3-website Testinvite

4.3-FlexiQuizWebsite:

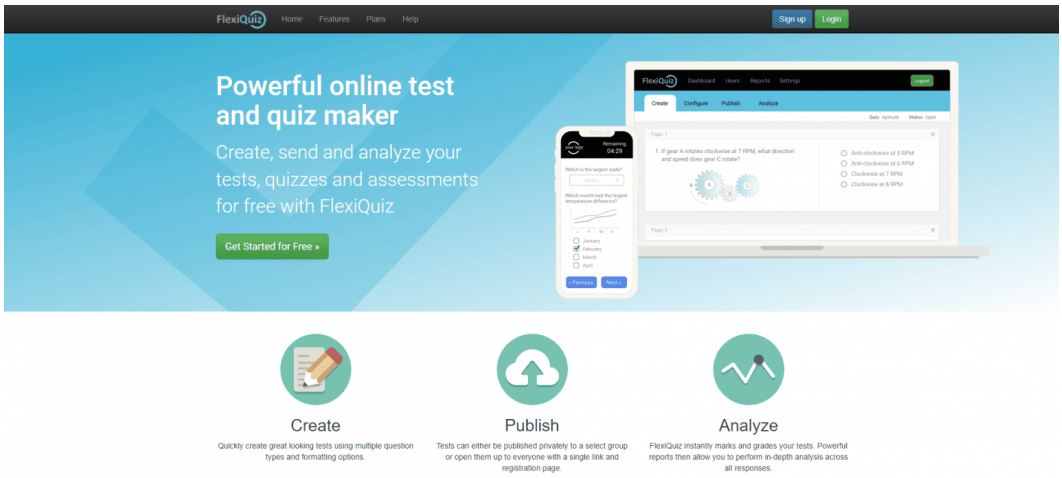
FlexiQuiz is an online examination platform designed for educators, trainers, and institutions to create and deliver secure, customizable quizzes. It offers an intuitive interface that allows teachers to build assessments quickly, with no technical expertise required, and provides tools to maintain exam integrity.

Key Features with Expanded Explanation:

- **Multiple Question Types:**
  Supports various question formats including multiple-choice, true/false, short answer, matching, and file uploads, making it suitable for diverse assessment needs.
- **No Student Registration Required:**
  Students can access exams via a unique link or access code, simplifying participation and eliminating the need to create accounts.
- **Time Limits and Navigation Control:**
  Teachers can set time limits for entire exams or individual questions, and restrict navigation to prevent students from skipping or revisiting questions.
- **Automated Grading and Feedback:**
  Objective questions are automatically scored, with the option to show correct answers and feedback immediately after submission.
- **Custom Branding and Certificates:**
  Allows educators to apply custom branding and generate completion certificates automatically for successful students.
- **Reporting and Analytics:**
  Provides detailed insights into student performance,

including scores, time spent, and pass/fail rates, helping instructors evaluate results effectively.

FlexiQuiz is ideal for teachers and trainers who want a flexible, user-friendly platform to manage assessments, with features that support both remote learning and in-person exams.



Feature	Our Project	Exam.net	Testinvite	FlexiQuiz
Secure Login (2FA)	✓	✗	✗	✗
Cheating Prevention	✓	✓	✓	✓
Question Management	✓	✓	✓	✓
AI Question Generator	✓	✗	✗	✗
Performance Analytics	✓	✗	✓	✓

Chat Groups	✓	✗	✗	✗
Smart Notifications	✓	✗	✗	✗
Custom Dashboards	✓	✗	✓	✓
No Student Account Needed	✗	✓	✓	✓
Auto-Grading	✓	✓	✓	✓

Figure 2.4-website FlexiQuiz

Chapter Three:

Analytical Study

1- Introduction:

In this chapter, we will introduce the actors, the functional requirements, when to use it,and the diagrams that describes the system components.

- **SOW:**

Examination System

- **PROJECT DESCRIPTION:**

Our project is based on making the process of accessing educational assessments easier and more transparent, while also offering opportunities for teachers to contribute, manage exams, and potentially expand their professional scope.

.

- **PROJECT OBJECTIVES:**

Develop a web-based examination system that connects teachers with students and facilitates seamless communication between them throughout the exam process.

- **PROJECT SCOPE:**

Deliver a web-based examination system that allows teachers to create and manage exams, enables students to access and submit responses securely, supports real-time communication between users, and ensures academic integrity and data protection for all participants.

- **PROJECT CONSTRAINTS:**

- Submit reports periodically.

- The project must be submitted before the deadline.

- **PROJECT TECHNOLOGY AND TOOLS:**

  -Django

  -HTML.

  -CSS.

  -JS.

  - Incremental Methodology.

➢ **WBS:**

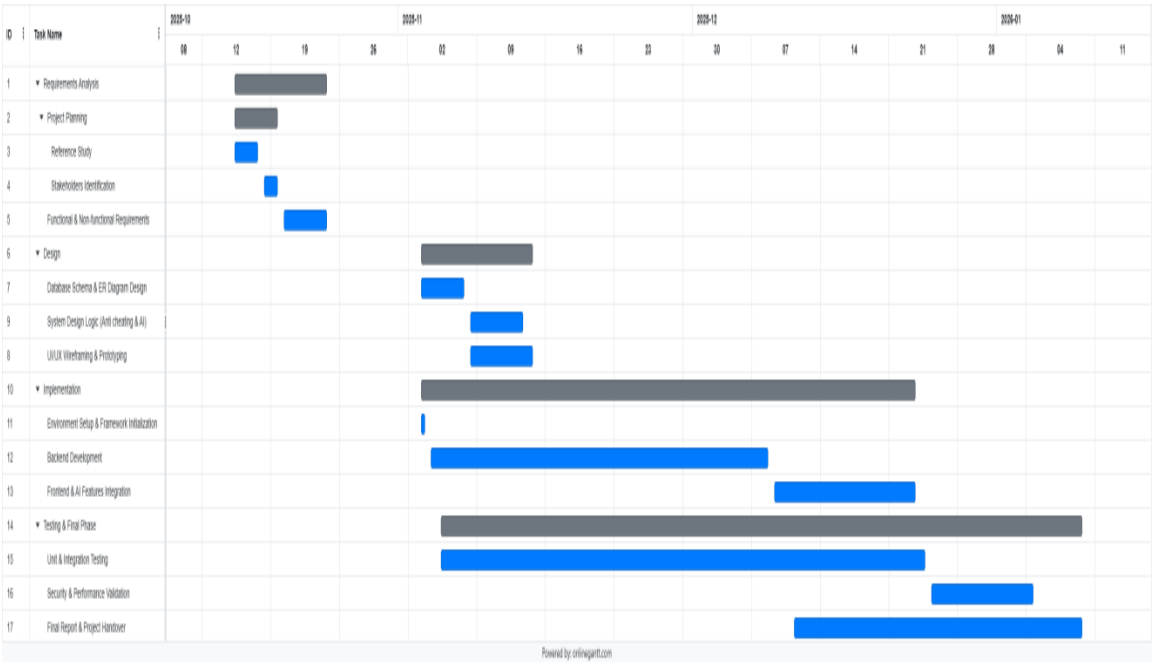


Figure 3.1-WBS

➢ ACTIVITY NETWORK:

TASK	DURATION	ANCESSTOR
T1 **[planning]**	5 days	
T2 **[planning documentation]**	5 days	T1(M1)
T3 **[analysis]**	6 days	T1(M1)
T4 **[analysis documentation]**	4 days	T3(M2)
T5 **[design]**	8 days	T3(M2)
T6 **[design documentation]**	1 days	T5(M3)
T7 **[implementation]**	36 days	T5(M3)
T8 **[implementation documentation]**	14 days	T7(M4)
T9 **[test]**	10 days	T7(M4)
T10 **[test documentation]**	35 day	T9(M5)

Table 3.1-activity network

2- Software Requirements Specification (SRS):

4.1- Project Title: “Online Examination System”.

4.2-Intruduction:

This project is based on the idea of facilitating access for educational users—students, teachers, and administrators—to manage, participate in, and monitor academic examinations securely and efficiently.

4.2.1-Problem Definition:

The education sector is rapidly embracing digital transformation, with institutions increasingly adopting online examination systems to support remote and blended learning. However, many still face challenges in delivering secure, organized, and scalable exam experiences for students and instructors.

This results in inefficiencies in managing assessments, inconsistencies in evaluating student performance, and difficulties maintaining academic integrity—ultimately leading to frustration for both educators and learners.

4.2.2-purpose:

-The purpose of this documentation is to describe the requirements in develop “Examination System”.

4.2.3-The System will accomplish the following tasks:

ID	TITLE	DESCRIPTION	TYPE	Priority

UC-ADM-01	View Users	The system shall allow an administrator to view users.	Functional	Medium
UC-ADM-02	Create User	The system shall allow an administrator to create a user account.	Functional	High
UC-ADM-03	Block User	The system shall allow an administrator to block a user account.	Functional	High
UC-ADM-04	Unblock User	The system shall allow an administrator to unblock a user account.	Functional	High
UC-ADM-05	Delete User	The system shall allow an administrator to delete a user account.	Functional	High
UC-ADM-06	View Teachers	The system shall allow an administrator to view teachers.	Functional	Medium
UC-ADM-07	Create Teacher	The system shall allow an administrator to create a teacher account.	Functional	High
UC-ADM-08	View Analytics Dashboard	The system shall allow an administrator to view the analytics dashboard.	Functional	Medium
UC-ADM-09	View Groups	The system shall allow an administrator to view groups.	Functional	Medium
UC-ADM-10	View Exams	The system shall allow an administrator to view exams.	Functional	Medium
UC-ADM-11	View Exam Details	The system shall allow an administrator to view exam details.	Functional	Medium
UC-ADM-12	View Exam Results	The system shall allow an administrator to view exam results.	Functional	High
UC-ADM-13	Monitor Exam	The system shall allow an administrator to monitor an exam.	Functional	High
UC-ADM-14	View Reports Inbox	The system shall allow an administrator to view the reports inbox.	Functional	Medium
UC-ADM-15	View Report	The system shall allow an administrator to view a report.	Functional	Medium
UC-ADM-16	Send Report	The system shall allow an administrator to send a report.	Functional	Medium
UC-ADM-17	Reply to Report	The system shall allow an administrator to reply to a report.	Functional	Medium
UC-ADM-18	View System Settings	The system shall allow an administrator to view system settings.	Functional	Medium
UC-ADM-19	Update Email Settings	The system shall allow an administrator to update email configuration settings.	Functional	Medium
UC-ADM-20	Update App Password	The system shall allow an administrator to update the application password.	Functional	Medium
UC-ADM-21	Update Platform Branding	The system shall allow an administrator to update platform branding settings.	Functional	Low

UC-COM-01	Log In	The system shall allow an authenticated user to log in.	Functional	High
UC-COM-02	Log Out	The system shall allow an authenticated user to log out.	Functional	Medium
UC-COM-03	Reset Password	The system shall allow an authenticated user to reset their password.	Functional	High
UC-COM-04	Update Profile	The system shall allow an authenticated user to update their profile.	Functional	Medium
UC-COM-05	Update Profile Information	The system shall allow an authenticated user to update their profile information.	Functional	Medium
UC-COM-06	Change Password	The system shall allow an authenticated user to change their password.	Functional	High
UC-PAIR-01	Verify Email Address	The system shall allow a teacher or student to verify their email address.	Functional	High
UC-PAIR-02	Resend Verification Code	The system shall allow a teacher or student to request a new verification code.	Functional	High
UC-PAIR-03	View Messages Inbox	The system shall allow a teacher or student to view their messages inbox.	Functional	Medium
UC-PAIR-04	Send Message	The system shall allow a teacher or student to send a message.	Functional	Medium
UC-PAIR-05	Reply to Message	The system shall allow a teacher or student to reply to a message.	Functional	Medium
UC-PAIR-06	View Group Messages	The system shall allow a teacher or student to view group messages.	Functional	Medium
UC-PAIR-07	Send Group Message	The system shall allow a teacher or student to send a group message.	Functional	Medium
UC-STD-01	View Teachers	The system shall allow a student to view teachers.	Functional	Medium
UC-STD-02	Send Join Request	The system shall allow a student to send a join request.	Functional	High
UC-STD-03	View Joined Teachers	The system shall allow a student to view joined teachers.	Functional	Medium
UC-STD-04	View Groups	The system shall allow a student to view groups.	Functional	Medium
UC-STD-05	Send Group Join Request	The system shall allow a student to send a group join request.	Functional	High
UC-STD-06	View Invitations	The system shall allow a student to view invitations.	Functional	Medium
UC-STD-07	Accept Invitation	The system shall allow a student to accept an invitation.	Functional	High

UC-STD-08	Reject Invitation	The system shall allow a student to reject an invitation.	Functional	High
UC-STD-09	View Available Exams	The system shall allow a student to view available exams.	Functional	Medium
UC-STD-10	View Exam Events	The system shall allow a student to view exam events.	Functional	Medium
UC-STD-11	Take Exam	The system shall allow a student to take an exam.	Functional	High
UC-STD-12	View Exam Result	The system shall allow a student to view an exam result.	Functional	High
UC-STD-13	View Exam Events	The system shall allow a student to view exam events.	Functional	Medium
UC-STD-14	View Exam Notifications	The system shall allow a student to view exam notifications.	Functional	Medium
UC-STD-15	View Group Details	The system shall allow a student to view group details.	Functional	Medium
UC-TCH-01	View Subjects	The system shall allow a teacher to view subjects.	Functional	Medium
UC-TCH-02	Create Subject	The system shall allow a teacher to create subject.	Functional	Medium
UC-TCH-03	Edit Subject	The system shall allow a teacher to edit subject.	Functional	Medium
UC-TCH-04	Delete Subject	The system shall allow a teacher to delete subject.	Functional	Medium
UC-TCH-05	View Subject Students	The system shall allow a teacher to view subject students.	Functional	Medium
UC-TCH-06	View Groups	The system shall allow a teacher to view groups.	Functional	Medium
UC-TCH-07	Create Group	The system shall allow a teacher to create group.	Functional	Medium
UC-TCH-08	View Group Details	The system shall allow a teacher to view group details.	Functional	Medium
UC-TCH-09	Edit Group	The system shall allow a teacher to edit group.	Functional	Medium
UC-TCH-10	Delete Group	The system shall allow a teacher to delete group.	Functional	Medium
UC-TCH-11	View Students	The system shall allow a teacher to view students.	Functional	Medium
UC-TCH-12	Invite Student	The system shall allow a teacher to invite a student.	Functional	High
UC-TCH-13	Accept Join Request	The system shall allow a teacher to accept join request.	Functional	High

UC-TCH-14	Reject Join Request	The system shall allow a teacher to reject join request.	Functional	High
UC-TCH-15	Remove Student	The system shall allow a teacher to remove a student.	Functional	High
UC-TCH-16	Accept Group Join Request	The system shall allow a teacher to accept group join request.	Functional	High
UC-TCH-17	Reject Group Join Request	The system shall allow a teacher to reject group join request.	Functional	High
UC-TCH-18	View Question Bank	The system shall allow a teacher to view the question bank.	Functional	Medium
UC-TCH-19	Create Question (Manually)	The system shall allow a teacher to create a question manually.	Functional	Medium
UC-TCH-20	Edit Question	The system shall allow a teacher to edit a question.	Functional	Medium
UC-TCH-21	Delete Question	The system shall allow a teacher to delete a question.	Functional	Medium
UC-TCH-22	Generate Questions Using AI	The system shall allow a teacher to generate questions using AI assistance.	Functional	Medium
UC-TCH-23	View Exams	The system shall allow a teacher to view exams.	Functional	Medium
UC-TCH-24	Create Exam	The system shall allow a teacher to create an exam.	Functional	High
UC-TCH-25	Edit Exam	The system shall allow a teacher to edit an exam.	Functional	High
UC-TCH-26	Delete Exam	The system shall allow a teacher to delete an exam.	Functional	High
UC-TCH-27	Manage Exam Questions	The system shall allow a teacher to manage exam questions.	Functional	High
UC-TCH-28	View Exam Results	The system shall allow a teacher to view exam results.	Functional	High
UC-TCH-29	Correct Student Attempt	The system shall allow a teacher to correct a student's attempt.	Functional	High
UC-TCH-30	Monitor Exam	The system shall allow a teacher to monitor an exam.	Functional	High
UC-TCH-31	View Live Monitor Data	The system shall allow a teacher to view live exam monitoring data.	Functional	Medium
UC-TCH-32	Monitor Student Audio Stream	The system shall allow a teacher to monitor a student's audio stream during an exam.	Functional	High
UC-TCH-33	View Student Camera Snapshots	The system shall allow a teacher to view a student's camera snapshots during an	Functional	High

		exam.		
UC-TCH-34	Perform Student Monitor Action	The system shall allow a teacher to perform a monitoring action on a student during an exam.	Functional	High
UC-TCH-35	Send Exam Notification	The system shall allow a teacher to send an exam notification to students.	Functional	Medium
UC-TCH-36	End Monitoring Session	The system shall allow a teacher to end an exam monitoring session.	Functional	High
UC-TCH-37	Talk to Student During Exam	The system shall allow a teacher to communicate with a student during an exam.	Functional	Medium

Table 3.2-TABLE REQUIREMENTS

4.2.4-System Description:

The system allows three user types—Admin, Teacher, and Student—to register, log in, and access their dedicated dashboards.

Teachers can create tests, add questions (via file upload and manual selection), assign tests to specific student groups, and manage those groups.

Admins, Teachers, and Students can browse existing users, groups, and tests based on their permissions.

Students can take assigned tests, communicate with peers via group chats, and manage their own account settings and messages

4.3-NonFunctional Requirements:

The Examination System ensures security through role-based access and Django's authentication system, protecting user data and exam integrity. It is designed for usability and maintainability, with modular code using Django's MVT architecture andclearrole separation.

Though currently using SQLite, the system is portable and scalable, allowing futureupgradestoproduction-readydatabases and performance enhancements.

4.4-Functional Requirements:

- **Shared:**
  - **I. Log In**
  - **II. Log Out**
  - **III. Reset Password**
  - **IV. Update Profile**
  - **V. Update Profile Information**
  - **VI. Change Password**

- **Pair:**
  - **I. Verify Email Address**
  - **II. Resend Verification Code**
  - **III. View Messages Inbox**
  - **IV. Send Message**
  - **V. Reply to Message**
  - **VI. View Group Messages**
  - **VII. Send Group Message**

- **Admin:**
  - **I. View Users**
  - **II. Create User**
  - **III. Block User**
  - **IV. Unblock User**
  - **V. Delete User**
  - **VI. View Teachers**
  - **VII. Create Teacher**
  - **VIII. View Analytics Dashboard**
  - **IX. View Groups**
  - **X. View Exams**
  - **XI. View Exam Details**
  - **XII. View Exam Results**
  - **XIII. Monitor Exam**
  - **XIV. View Reports Inbox**
  - **XV. View Report**
  - **XVI. Send Report**
  - **XVII. Reply to Report**
  - **XVIII. View System Settings**
  - **XIX. Update Email Settings**
  - **XX. Update App Password**
  - **XXI. Update Platform Branding**

- **Teacher:**
  - **I. View Subjects**
  - **II. Create Subject**
  - **III. Edit Subject**
  - **IV. Delete Subject**
  - **V. View Subject Students**
  - **VI. View Groups**
  - **VII. Create Group**
  - **VIII.View Group Details**
  - **IX. Edit Group**
  - **X. Delete Group**
  - **XI. View Students**
  - **XII. Invite Student**
  - **XIII. Accept Join Request**
  - **XIV. Reject Join Request**
  - **XV. Remove Student**
  - **XVI. Accept Group Join Request**
  - **XVII. Reject Group Join Request**
  - **XVIII. View Question Bank**
  - **XIX. Create Question (Manually)**
  - **XX. Edit Question**
  - **XXI. Delete Question**
  - **XXII. Generate Questions Using AI**
  - **XXIII. View Exams**
  - **XXIV. Create Exam**

XXV. Edit Exam

XXVI. Delete Exam

XXVII. Manage Exam Questions

XXVIII. View Exam Results

XXIX. Correct Student Attempt

XXX. Monitor Exam

XXXI. View Live Monitor Data

XXXII. Monitor Student Audio Stream

XXXIII. View Student Camera Snapshots

XXXIV. Perform Student Monitor Action

XXXV. Send Exam Notification

XXXVI. End Monitoring Session

XXXVII. Talk to Student During Exam

- **Student:**
  - **I. View Teachers**
  - **II. Send Join Request**
  - **III. View Joined Teachers**
  - **IV. View Groups**
  - **V. Send Group Join Request**
  - **VI. View Invitations**
  - **VII. Accept Invitation**
  - **VIII. Reject Invitation**
  - **IX. View Available Exams**
  - **X. View Exam Events**
  - **XI. Take Exam**
  - **XII. View Exam Result**
  - **XIII. View Exam Events**
  - **XIV. View Exam Notifications**
  - **XV. View Group Details**

5-Unified Modeling

5.1- Use Case Diagram:

5.1.5-Use case system Level-1:

5.1.

6-

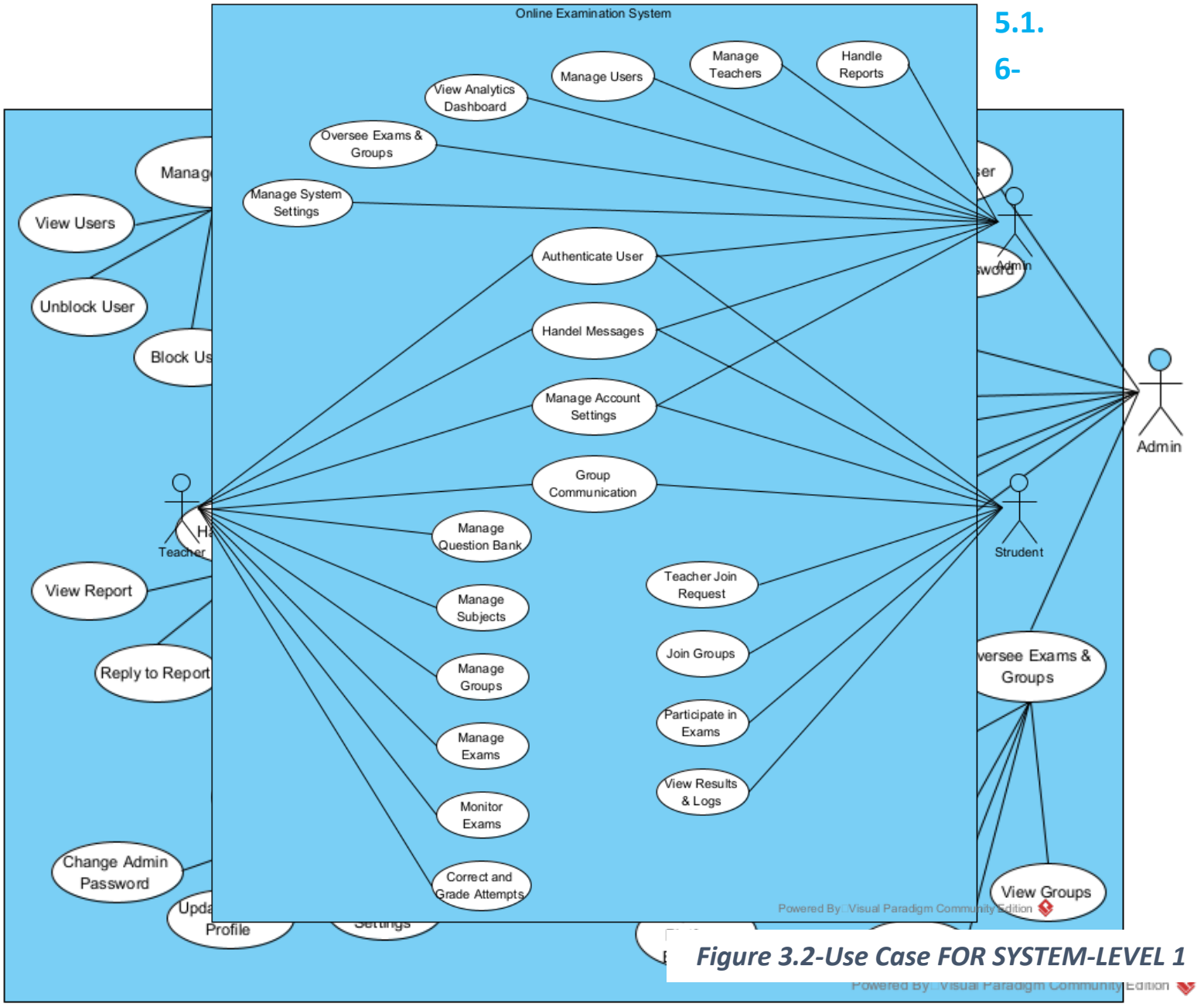
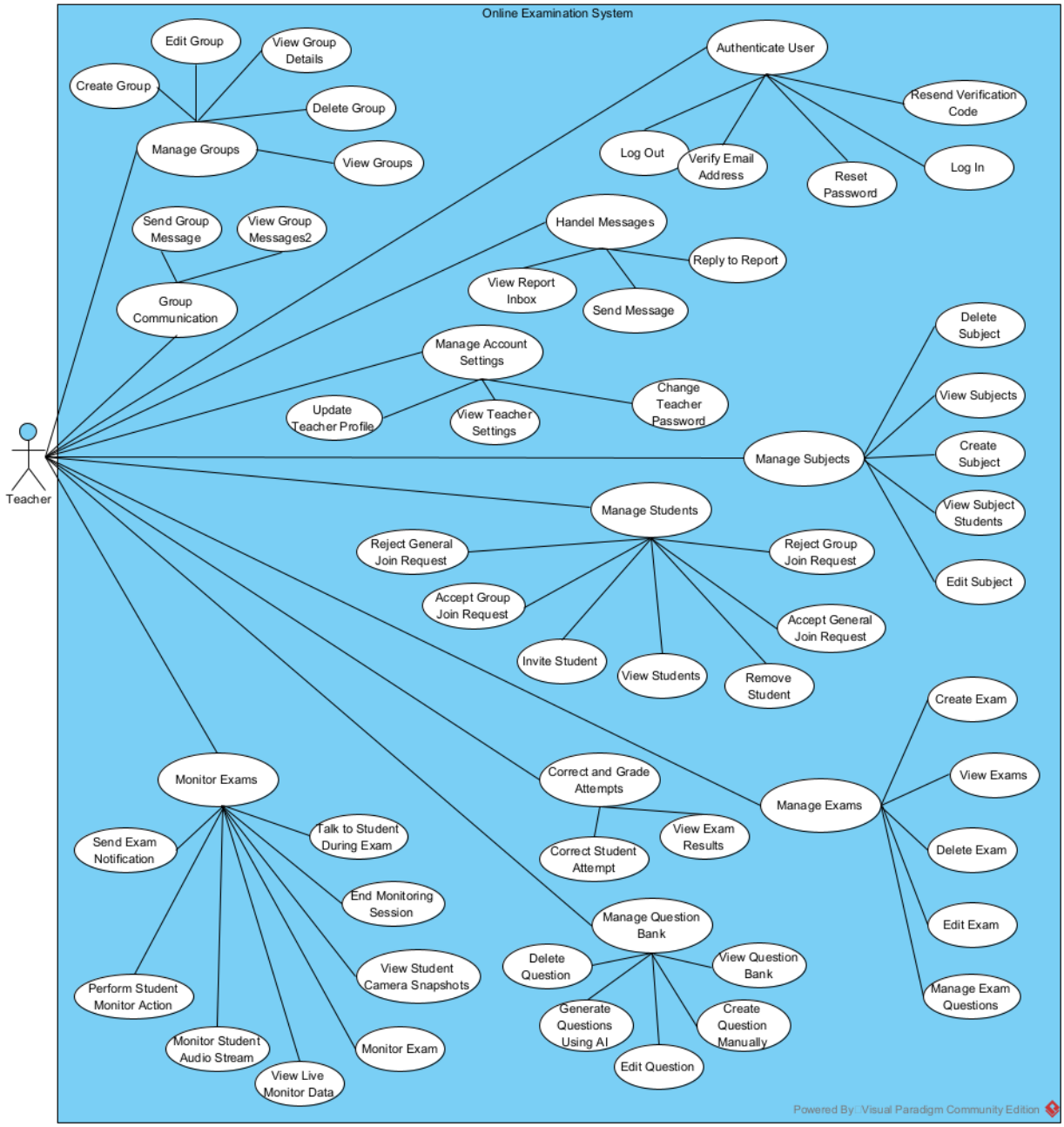


Figure 3.2-Use Case FOR SYSTEM-LEVEL 1

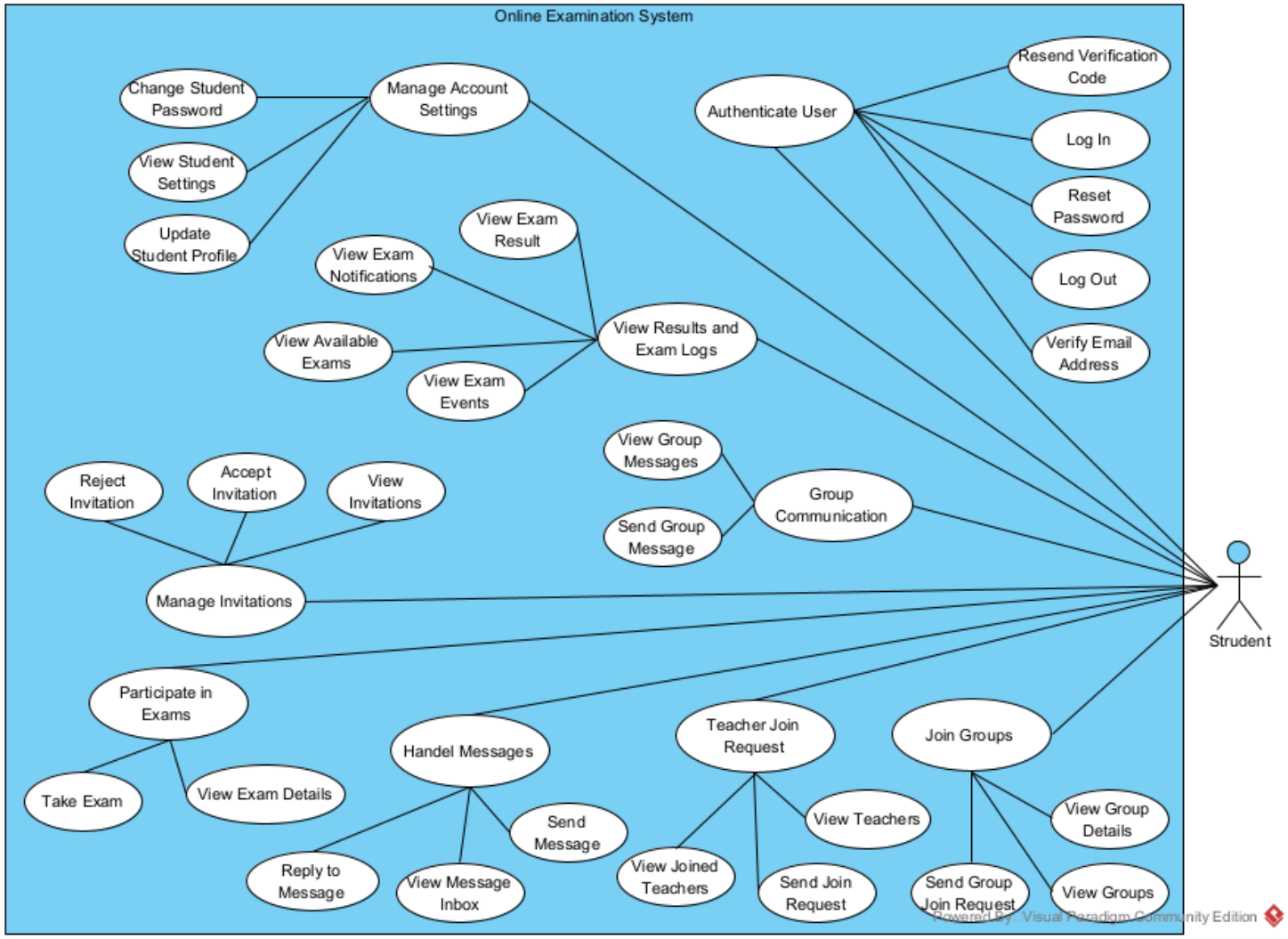
Use case system Low-Level(Admin):

Figure 3.3-Use Case FOR SYSTEM-LEVEL 2

5.1.6-Use case system Low-Level(Teacher):



5.1.6-Use case system Low-Level(Student):



5.2-Use Case Specification:

Use Case ID	**UC-COM-01**
Use Case Name	Log In
Primary Actor	Admin, Teacher, Student
Preconditions	User has a registered account.
Trigger	User selects “Log In” and submits credentials.
Basic Flow	1) System displays login form. 2) User enters username/email and password. 3) System validates credentials. 4) System starts a session and redirects user to the appropriate dashboard based on role.
Alternative/Exception Flows	A1) Invalid credentials → System shows error message and remains on login page. A2) Account is inactive/blocked → System denies access and shows an appropriate message. A3) Email not verified (Teacher/Student) → System redirects user to email verification flow / shows verification prompt.
Postconditions	Success: User session is created and user is authenticated. Failure: No session is created and user remains unauthenticated.
Business Rules	Blocked/inactive accounts cannot log in.

Table 3.3.1-Use Case Specification ‘Login’

Use Case ID	UC-COM-02
Use Case Name	Log Out
Primary Actor	Admin, Teacher, Student
Preconditions	User is logged in (active session exists).
Trigger	User selects “Log Out”.
Basic Flow	1) System invalidates the current session. 2) System redirects the user to the login page (or public landing page).
Alternative/Exception Flows	A1) Session already expired → System redirects to login page.
Postconditions	Success: User is logged out and session is terminated. Failure: User remains logged in (rare; only if logout fails).

Business Rules	N/A

Table 3.3.2-Use Case Specification 'Log Out'

Use Case ID	UC-COM-03
Use Case Name	Reset Password
Primary Actor	Admin, Teacher, Student
Preconditions	User has an account with a valid email address.
Trigger	User selects "Forgot/Reset Password".
Basic Flow	1) System displays password reset request form. 2) User enters email address. 3) System sends password reset link/instructions. 4) User opens the reset link. 5) System prompts for a new password. 6) User submits the new password. 7) System updates password and confirms success.
Alternative/Exception Flows	A1) Email not found → System shows a generic response (security) and does not reveal account existence. A2) Reset token invalid/expired → System shows error and allows requesting a new reset link. A3) New password fails validation → System shows validation errors and asks user to re-enter.
Postconditions	Success: Password is changed and user can log in with the new password. Failure: Password remains unchanged.
Business Rules	Password must meet the system's password validation policy.

Table 3.3.3-Use Case Specification 'Reset Password'

Use Case ID	UC-COM-04
Use Case Name	Update Profile
Primary Actor	Authenticated User (Admin / Teacher / Student)

Goal	Update personal profile information through the account settings pages.
Preconditions	1) User is authenticated. 2) User is editing their own profile.
Trigger	User submits profile changes from Settings.
Main Flow (Basic Success Scenario)	1) User opens Settings. 2) System loads current profile data. 3) User edits allowed fields (e.g., full name, email where allowed). 4) User submits the form. 5) System validates inputs. 6) System saves updates. 7) System shows a success message and displays the updated profile.
Alternative Flows	A1: Validation fails → System shows field errors and does not save changes.
Exception Flows	E1: Server/database error → System shows an error and keeps the previous values.
Postconditions	Profile updates are persisted and reflected in the UI.
Business Rules / Constraints	Only authenticated users may update their own profile. Role-specific settings pages must apply the same ownership rule.
UI / URL References (from code)	core/urls.py: - Admin: /admin-dashboard/settings/ (admin_settings) - Teacher: /teacher-dashboard/settings/ (teacher_settings) - Student: /student-dashboard/?page=settings (student_dashboard with page=settings)
Code References (views / templates)	core/views.py: - admin_settings - teacher_settings - student_dashboard (handles page=settings) Templates: - core/templates/core/admin_settings.html - core/templates/core/teacher_settings.html - core/templates/core/student_dashboard.html (settings section)

Table 3.3.4-Use Case Specification 'Update Profile

Use Case ID	UC-COM-05
Use Case Name	Update Profile Information
Primary Actor	Admin, Teacher, Student
Preconditions	User is logged in and account settings page is accessible.
Trigger	User submits updated profile information.
Basic Flow	1) User edits profile fields (e.g., name, email). 2) User submits the update form. 3) System validates input data. 4) System updates user profile information. 5) System displays success confirmation.
Alternative/Exception Flows	A1) Invalid input data → System displays validation errors and does not save changes. A2) Email already in use → System rejects update and shows error message.
Postconditions	Success: User profile information is updated. Failure: Profile information remains unchanged.
Business Rules	Profile data must comply with system validation rules.

Table 3.3.5-Use Case Specification 'Update Profile Information'

Use Case ID	UC-COM-06
Use Case Name	Change Password
Primary Actor	Admin, Teacher, Student
Preconditions	User is logged in.
Trigger	User submits password change request from account settings page.
Basic Flow	1) User enters current password. 2) User enters new password and confirms it. 3) System verifies current password. 4) System validates new password against password policy. 5) System updates the password.

	6) System confirms successful password change.
Alternative/Exception Flows	A1) Current password incorrect → System rejects request and shows error. A2) New password does not meet policy → System displays validation errors.
Postconditions	Success: Password is updated and remains valid for future logins. Failure: Password remains unchanged.
Business Rules	New password must meet the system's password policy.

Table 3.3.6-Use Case Specification 'Change Password'

PAIR

Use Case ID	UC-PAIR-01
Use Case Name	Verify Email Address
Primary Actor	Teacher, Student
Preconditions	User has registered and email is not verified.
Trigger	User submits verification code or opens verification page.
Basic Flow	1) System displays email verification page. 2) User enters verification code. 3) System validates the code. 4) System marks the email as verified. 5) System confirms verification success.
Alternative/Exception Flows	A1) Code is invalid/expired → System shows error and asks user to re-enter or request a new code.
Postconditions	Success: User email is verified. Failure: Email remains unverified.
Business Rules	Only unverified accounts can perform email verification.

Table 3.3.7-Use Case Specification 'Verify Email Address'

Use Case ID	UC-PAIR-02
Use Case Name	Resend Verification Code

Primary Actor	Teacher, Student
Preconditions	User email is not verified.
Trigger	User selects “Resend Verification Code”.
Basic Flow	1) User requests a new verification code. 2) System generates a new code. 3) System sends the code to the user’s email address. 4) System confirms code was sent.
Alternative/Exception Flows	A1) Resend limit exceeded → System blocks resend temporarily and shows warning.
Postconditions	Success: A new verification code is sent. Failure: No code is sent.
Business Rules	Resend requests may be rate-limited.

Table 3.3.8-Use Case Specification ‘Resend Verification Code***’***

Use Case ID	UC-PAIR-03
Use Case Name	View Messages Inbox
Primary Actor	Teacher, Student
Preconditions	User is logged in.
Trigger	User navigates to the messages/reports page.
Basic Flow	1) System retrieves all messages related to the user. 2) System displays messages ordered by date (newest first). 3) System highlights unread messages.
Alternative/Exception Flows	A1) No messages found → System displays empty inbox message.
Postconditions	Success: Messages inbox is displayed. Failure: Messages are not displayed.
Business Rules	Users can only view messages they are sender or recipient of.

Table 3.3.9-Use Case Specification ‘View Messages Inbox***’***

Use Case ID	UC-PAIR-04
Use Case Name	Send Message
Primary Actor	Teacher, Student
Preconditions	User is logged in.
Trigger	User selects “Send Message” and submits message content.

Basic Flow	1) System displays message composition form. 2) User selects recipient (Teacher or Student). 3) User enters message content. 4) User submits the message. 5) System saves the message and notifies the recipient.
Alternative/Exception Flows	A1) Invalid or empty message content → System shows validation error. A2) Recipient not found or not allowed → System rejects message.
Postconditions	Success: Message is sent and stored. Failure: Message is not sent.
Business Rules	Users may only send messages to users they are allowed to communicate with (e.g., same group or related context).

Table 3.3.10-Use Case Specification 'Send Message'

Use Case ID	UC-PAIR-05
Use Case Name	Reply to Message
Primary Actor	Teacher, Student
Preconditions	User is logged in and has at least one received message.
Trigger	User selects "Reply" on an existing message.
Basic Flow	1) System displays the selected message. 2) System displays reply input field. 3) User enters reply content. 4) User submits the reply. 5) System saves the reply and notifies the original sender.
Alternative/Exception Flows	A1) Empty reply content → System shows validation error.
Postconditions	Success: Reply message is sent and linked to the original message. Failure: Reply is not sent.
Business Rules	Replies must be associated with an existing message thread.

Table 3.3.11-Use Case Specification 'Reply to Message'

Use Case ID	UC-PAIR-06
Use Case Name	View Group Messages
Primary Actor	Teacher, Student

Preconditions	User is logged in and is a member of the group.
Trigger	User opens a group details page.
Basic Flow	1) System retrieves messages related to the selected group. 2) System displays group messages in chronological order.
Alternative/Exception Flows	A1) Group has no messages → System displays empty group message list. A2) User is not a group member → System denies access.
Postconditions	Success: Group messages are displayed. Failure: Messages are not displayed.
Business Rules	Only group members can view group messages.

Table 3.3.12-Use Case Specification 'View Group Messages***'***

Use Case ID	UC-PAIR-07
Use Case Name	Send Group Message
Primary Actor	Teacher, Student
Preconditions	User is logged in and is a member of the group.
Trigger	User submits a new message within a group.
Basic Flow	1) User enters message content in the group message form. 2) User submits the message. 3) System validates message content. 4) System saves the message and updates the group message list.
Alternative/Exception Flows	A1) Empty or invalid message content → System shows validation error.
Postconditions	Success: Message is posted to the group and visible to all group members. Failure: Message is not posted.
Business Rules	Only group members can send messages to the group.

Table 3.3.13-Use Case Specification 'Send Group Message***'***

Admin

Use Case ID	UC-ADM-01
Use Case Name	View Users
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens the users management page.
Basic Flow	1) System retrieves all users. 2) System displays users list with basic information.
Alternative/Exception Flows	A1) No users found → System displays empty list message.
Postconditions	Success: Users list is displayed. Failure: Users are not displayed.
Business Rules	Only admin can access the users list.

Table 3.3.14-Use Case Specification 'View Users***'***

Use Case ID	UC-ADM-02
Use Case Name	Create User
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin submits a new user creation form.
Basic Flow	1) System displays create user form. 2) Admin enters user information. 3) Admin submits the form. 4) System validates input. 5) System creates the user account.
Alternative/Exception Flows	A1) Invalid input → System shows validation errors. A2) Email already exists → System rejects creation.
Postconditions	Success: New user account is created. Failure: User is not created.
Business Rules	User email must be unique.

Table 3.3.15-Use Case Specification 'Create User'

Use Case ID	UC-ADM-03
Use Case Name	Block User
Primary Actor	Admin
Preconditions	Admin is logged in and user exists.
Trigger	Admin selects "Block User".
Basic Flow	1) Admin selects a user from the users list. 2) Admin triggers block action. 3) System marks the user account as blocked.
Alternative/Exception Flows	A1) Target user is admin/superuser → System prevents blocking.
Postconditions	Success: User account is blocked. Failure: User account remains active.
Business Rules	Admin cannot block themselves or superusers.

Table 3.3.16-Use Case Specification 'Block User'

Use Case ID	UC-ADM-04
Use Case Name	Unblock User
Primary Actor	Admin
Preconditions	Admin is logged in and user is blocked.
Trigger	Admin selects "Unblock User".
Basic Flow	1) Admin selects a blocked user. 2) Admin triggers unblock action. 3) System reactivates the user account.
Alternative/Exception Flows	A1) User is not blocked → System ignores the action.
Postconditions	Success: User account is active. Failure: User remains blocked.
Business Rules	Only blocked users can be unblocked.

Table 3.3.17-Use Case Specification 'Unblock User'

Use Case ID	UC-ADM-05
Use Case Name	Delete User
Primary Actor	Admin
Preconditions	Admin is logged in and user exists.
Trigger	Admin selects “Delete User”.
Basic Flow	1) Admin selects a user. 2) Admin confirms deletion. 3) System deletes the user account.
Alternative/Exception Flows	A1) Target user is admin/superuser → System prevents deletion.
Postconditions	Success: User account is deleted. Failure: User account remains unchanged.
Business Rules	Admin cannot delete themselves or superusers.

Table 3.3.18-Use Case Specification ‘Delete User***’***

Use Case ID	UC-ADM-06
Use Case Name	View Teachers
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens teachers management page.
Basic Flow	1) System retrieves all teacher accounts. 2) System displays teachers list.
Alternative/Exception Flows	A1) No teachers found → System displays empty list message.
Postconditions	Success: Teachers list is displayed.
Business Rules	Only admin can view teachers list.

Table 3.3.19-Use Case Specification ‘View Teachers***’***

Field	Content
Use Case ID	UC-ADM-07
Use Case Name	Create Teacher
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin submits a new teacher creation form.
Basic Flow	1) System displays create teacher form. 2) Admin enters teacher information. 3) Admin submits the form. 4) System validates input. 5) System creates teacher account.
Alternative/Exception Flows	A1) Invalid input → System shows validation errors.
Postconditions	Success: Teacher account is created.
Business Rules	Teacher accounts must have teacher role assigned.

Table 3.3.20-Use Case Specification 'Create Teacher***'***

Use Case ID	UC-ADM-08
Use Case Name	View Analytics Dashboard
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens the analytics dashboard page.
Basic Flow	1) System retrieves aggregated system statistics. 2) System calculates analytics metrics (users, teachers, students, exams, groups, reports). 3) System displays analytics dashboard with charts and counts.
Alternative/Exception Flows	A1) Analytics data unavailable → System displays partial data or error message.
Postconditions	Success: Analytics dashboard is displayed. Failure: Dashboard is not displayed.

Business Rules	Analytics data is read-only and visible only to admin.

Table 3.3.21-Use Case Specification 'View Analytics Dashboard'

Use Case ID	UC-ADM-09
Use Case Name	View Groups
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens groups page.
Basic Flow	1) System retrieves all groups. 2) System displays groups list.
Alternative/Exception Flows	A1) No groups found → System shows empty list.
Postconditions	Success: Groups list is displayed.
Business Rules	Groups are view-only for admin.

Use Case ID	UC-ADM-10
Use Case Name	View Exams
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens exams page.
Basic Flow	1) System retrieves all exams. 2) System displays exams list.
Alternative/Exception Flows	A1) No exams found → System shows empty list.
Postconditions	Success: Exams list is displayed.
Business Rules	Exams are view-only for admin.

Table 3.3.22-Use Case Specification 'View Exams'

Use Case ID	UC-ADM-11
Use Case Name	View Exam Details
Primary Actor	Admin
Preconditions	Admin is logged in and exam exists.
Trigger	Admin selects an exam from the exams list.
Basic Flow	1) System retrieves exam details. 2) System displays exam information.
Alternative/Exception Flows	A1) Exam not found → System shows error.
Postconditions	Success: Exam details are displayed.
Business Rules	Exam details are read-only.

Table 3.3.22-Use Case Specification 'View Exam Details'

Use Case ID	UC-ADM-12
Use Case Name	View Exam Results
Primary Actor	Admin
Preconditions	Admin is logged in and exam exists.
Trigger	Admin selects "View Results" for an exam.
Basic Flow	1) System retrieves exam results. 2) System displays student results.
Alternative/Exception Flows	A1) No results available → System displays message.
Postconditions	Success: Exam results are displayed.
Business Rules	Results are read-only for admin.

Table 3.3.23-Use Case Specification 'View Exam Results'

Use Case ID	UC-ADM-13
Use Case Name	Monitor Exam
Primary Actor	Admin
Preconditions	Admin is logged in and exam is active.
Trigger	Admin selects “Monitor Exam”.
Basic Flow	1) System displays live exam monitoring data. 2) Admin observes exam status and events.
Alternative/Exception Flows	A1) Exam not active → System denies monitoring.
Postconditions	Success: Exam monitoring view is displayed.
Business Rules	Monitoring is view-only for admin.

Table 3.3.24-Use Case Specification ‘Monitor Exam***’***

Use Case ID	UC-ADM-14
Use Case Name	View Reports Inbox
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens the reports page.
Basic Flow	1) System retrieves all reports related to admin. 2) System displays reports list ordered by date. 3) System highlights unread reports.
Alternative/Exception Flows	A1) No reports available → System displays empty inbox message.
Postconditions	Success: Reports inbox is displayed.
Business Rules	Only admin can access reports inbox.

Table 3.3.25-Use Case Specification ‘View Reports Inbox***’***

Use Case ID	UC-ADM-15
Use Case Name	View Report
Primary Actor	Admin
Preconditions	Admin is logged in and report exists.
Trigger	Admin selects a report from the inbox.
Basic Flow	1) System retrieves report details. 2) System displays report content and metadata. 3) System marks report as read.
Alternative/Exception Flows	A1) Report not found → System displays error message.
Postconditions	Success: Report details are displayed.
Business Rules	Reports are read-only unless replied to.

Table 3.3.26-Use Case Specification 'View Report'

Use Case ID	UC-ADM-16
Use Case Name	Send Report
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin selects "Send Report".
Basic Flow	1) System displays report composition form. 2) Admin enters report subject and content. 3) Admin submits the report. 4) System saves the report and notifies the recipient.
Alternative/Exception Flows	A1) Empty or invalid content → System shows validation error.
Postconditions	Success: Report is sent and stored.
Business Rules	Reports must have valid subject and content.

Table 3.3.27-Use Case Specification 'Send Report'

Use Case ID	UC-ADM-17
Use Case Name	Reply to Report
Primary Actor	Admin
Preconditions	Admin is logged in and viewing an existing report.
Trigger	Admin selects “Reply” on a report.
Basic Flow	1) System displays reply input field. 2) Admin enters reply content. 3) Admin submits reply. 4) System saves reply and notifies the original sender.
Alternative/Exception Flows	A1) Empty reply → System shows validation error.
Postconditions	Success: Reply is sent and linked to the report.
Business Rules	Replies must be associated with an existing report.

Table 3.3.28-Use Case Specification ‘Reply to Report***’***

Use Case ID	UC-ADM-18
Use Case Name	View System Settings
Primary Actor	Admin
Preconditions	Admin is logged in.
Trigger	Admin opens system settings page.
Basic Flow	1) System retrieves current system configuration. 2) System displays system settings.
Alternative/Exception Flows	A1) Settings unavailable → System displays error message.
Postconditions	Success: System settings are displayed.
Business Rules	Only admin can view system settings.

Table 3.3.29-Use Case Specification 'View System Settings'

Use Case ID	UC-ADM-19
Use Case Name	Update Email Settings
Primary Actor	Admin
Preconditions	Admin is logged in and system settings page is open.
Trigger	Admin submits updated email configuration.
Basic Flow	1) Admin updates email settings. 2) Admin submits changes. 3) System validates configuration. 4) System saves updated email settings.
Alternative/Exception Flows	A1) Invalid email configuration → System rejects changes.
Postconditions	Success: Email settings are updated.
Business Rules	Email configuration must be valid and reachable.

Table 3.3.30-Use Case Specification 'Update Email Settings'

Use Case ID	UC-ADM-20
Use Case Name	Update App Password
Primary Actor	Admin
Preconditions	Admin is logged in and system settings page is open.
Trigger	Admin submits updated app password.
Basic Flow	1) Admin enters new app password. 2) Admin submits the update. 3) System validates and stores the new password.
Alternative/Exception Flows	A1) Invalid password format → System rejects update.

Postconditions	Success: App password is updated.
Business Rules	App password must follow security requirements.

Table 3.3.31-Use Case Specification 'Update App Password'

Use Case ID	UC-ADM-21
Use Case Name	Update Platform Branding
Primary Actor	Admin
Preconditions	Admin is logged in and system settings page is open.
Trigger	Admin submits branding changes.
Basic Flow	1) Admin updates branding information (e.g., logo, name). 2) Admin submits changes. 3) System saves branding updates.
Alternative/Exception Flows	A1) Invalid branding data → System rejects changes.
Postconditions	Success: Platform branding is updated.
Business Rules	Branding changes apply system-wide.

Table 3.3.32-Use Case Specification 'Update Platform Branding'

Teacher

Use Case ID	UC-TCH-01
Use Case Name	View Subjects
Primary Actor	Teacher

Preconditions	Teacher is logged in.
Trigger	Teacher opens subjects page.
Basic Flow	1) System retrieves subjects owned by the teacher. 2) System displays subjects list.
Alternative/Exception Flows	A1) No subjects found → System displays empty list message.
Postconditions	Success: Subjects list is displayed.
Business Rules	Teacher can view only their own subjects.

Table 3.3.33-Use Case Specification 'View Subjects'

Use Case ID	UC-TCH-02
Use Case Name	Create Subject
Primary Actor	Teacher
Preconditions	Teacher is logged in.
Trigger	Teacher submits create subject form.
Basic Flow	1) System displays create subject form. 2) Teacher enters subject information. 3) Teacher submits the form. 4) System validates input. 5) System creates the subject.
Alternative/Exception Flows	A1) Invalid input → System shows validation errors.
Postconditions	Success: Subject is created.
Business Rules	Subject name must be unique per teacher.

Table 3.3.34-Use Case Specification 'Create Subject'

Use Case ID	UC-TCH-03
Use Case Name	Edit Subject

Primary Actor	Teacher
Preconditions	Teacher is logged in and subject exists.
Trigger	Teacher selects “Edit Subject”.
Basic Flow	1) System displays subject details. 2) Teacher updates subject information. 3) Teacher submits changes. 4) System validates and saves updates.
Alternative/Exception Flows	A1) Invalid input → System rejects changes.
Postconditions	Success: Subject information is updated.
Business Rules	Teacher can edit only their own subjects.

Table 3.3.35-Use Case Specification ‘Edit Subject***’***

Use Case ID	UC-TCH-04
Use Case Name	Delete Subject
Primary Actor	Teacher
Preconditions	Teacher is logged in and subject exists.
Trigger	Teacher selects “Delete Subject”.
Basic Flow	1) Teacher confirms deletion. 2) System deletes the subject.
Alternative/Exception Flows	A1) Subject has active groups/exams → System prevents deletion.
Postconditions	Success: Subject is deleted.
Business Rules	Subject cannot be deleted if linked to active groups or exams.

Table 3.3.36-Use Case Specification ‘Delete Subject***’***

Use Case ID	UC-TCH-05
Use Case Name	View Subject Students
Primary Actor	Teacher
Preconditions	Teacher is logged in and subject exists.

Trigger	Teacher opens subject students page.
Basic Flow	1) System retrieves students enrolled in the subject. 2) System displays students list.
Alternative/Exception Flows	A1) No students enrolled → System displays empty list.
Postconditions	Success: Subject students list is displayed.
Business Rules	Teacher can view students only for their own subjects.

Table 3.3.37-Use Case Specification 'View Subject Students'

Use Case ID	UC-TCH-06
Use Case Name	View Groups
Primary Actor	Teacher
Preconditions	Teacher is logged in.
Trigger	Teacher opens groups page.
Basic Flow	1) System retrieves groups created by the teacher. 2) System displays groups list.
Alternative/Exception Flows	A1) No groups found → System displays empty list.
Postconditions	Success: Groups list is displayed.
Business Rules	Teacher can view only their own groups.

Table 3.3.38-Use Case Specification 'View Groups'

Use Case ID	UC-TCH-07
Use Case Name	Create Group
Primary Actor	Teacher

Preconditions	Teacher is logged in and subject exists.
Trigger	Teacher submits create group form.
Basic Flow	1) System displays create group form. 2) Teacher selects subject and enters group information. 3) Teacher submits the form. 4) System validates input. 5) System creates the group.
Alternative/Exception Flows	A1) Invalid input → System shows validation errors.
Postconditions	Success: Group is created.
Business Rules	Group must be associated with an existing subject.

Table 3.3.39-Use Case Specification 'Create Group'

Use Case ID	UC-TCH-08
Use Case Name	View Group Details
Primary Actor	Teacher
Preconditions	Teacher is logged in and group exists.
Trigger	Teacher selects a group from the groups list.
Basic Flow	1) System retrieves group details. 2) System displays group information and members.
Alternative/Exception Flows	A1) Group not found → System shows error.
Postconditions	Success: Group details are displayed.
Business Rules	Teacher can view only their own groups.

Table 3.3.40-Use Case Specification 'View Group Details'

Use Case ID	UC-TCH-09
Use Case Name	Edit Group
Primary Actor	Teacher
Preconditions	Teacher is logged in and group exists.

Trigger	Teacher selects “Edit Group”.
Basic Flow	1) System displays group edit form. 2) Teacher updates group information. 3) Teacher submits changes. 4) System validates and saves updates.
Alternative/Exception Flows	A1) Invalid input → System rejects changes.
Postconditions	Success: Group information is updated.
Business Rules	Teacher can edit only their own groups.

Table 3.3.41-Use Case Specification ‘Edit Group***’***

Use Case ID	UC-TCH-10
Use Case Name	Delete Group
Primary Actor	Teacher
Preconditions	Teacher is logged in and group exists.
Trigger	Teacher selects “Delete Group”.
Basic Flow	1) Teacher confirms deletion. 2) System deletes the group.
Alternative/Exception Flows	A1) Group has enrolled students or active exams → System prevents deletion.
Postconditions	Success: Group is deleted.
Business Rules	Group cannot be deleted if it contains students or active exams.

Table 3.3.42-Use Case Specification ‘Delete Group***’***

Use Case ID	UC-TCH-11
Use Case Name	View Students
Primary Actor	Teacher
Preconditions	Teacher is logged in.
Trigger	Teacher opens the students management page.
Basic Flow	1) System retrieves students associated with the teacher (groups/requests). 2) System displays students list.
Alternative/Exception Flows	A1) No students found → System displays empty list message.
Postconditions	Success: Students list is displayed.
Business Rules	Teacher can view only students related to their groups/subjects.
Use Case ID	UC-TCH-12
Use Case Name	Invite Student
Primary Actor	Teacher
Preconditions	Teacher is logged in; target student exists.

Trigger	Teacher submits an invitation request for a student.
Basic Flow	1) Teacher selects a student to invite. 2) Teacher selects target group (if applicable). 3) Teacher submits invitation. 4) System records invitation and notifies the student.
Alternative/Exception Flows	A1) Student already invited/already in group → System rejects invitation. A2) Invalid student/group → System shows error.
Postconditions	Success: Invitation is created and visible to student. Failure: No invitation is created.
Business Rules	A student cannot be invited twice to the same group.

Table 3.3.43-Use Case Specification 'View Students'

Use Case ID	UC-TCH-13
Use Case Name	Accept Join Request
Primary Actor	Teacher
Preconditions	Teacher is logged in; a pending join request exists.
Trigger	Teacher selects "Accept" for a join request.
Basic Flow	1) System displays pending join requests. 2) Teacher selects a request and accepts it. 3) System adds the student to the relevant group (or teacher context). 4) System marks request as accepted and notifies the student.
Alternative/Exception Flows	A1) Request already handled → System prevents duplicate handling. A2) Group is not available → System rejects acceptance.
Postconditions	Success: Student becomes a member and request is accepted. Failure: Student membership is unchanged.
Business Rules	Only pending requests can be accepted.

Table 3.3.44-Use Case Specification 'Accept Join Request'

Use Case ID	UC-TCH-14
Use Case Name	Reject Join Request
Primary Actor	Teacher
Preconditions	Teacher is logged in; a pending join request exists.
Trigger	Teacher selects “Reject” for a join request.
Basic Flow	1) System displays pending join requests. 2) Teacher selects a request and rejects it. 3) System marks request as rejected and notifies the student.
Alternative/Exception Flows	A1) Request already handled → System prevents duplicate handling.
Postconditions	Success: Request is rejected and student is not added. Failure: Request remains pending.
Business Rules	Only pending requests can be rejected.

Table 3.3.45-Use Case Specification ‘Reject Join Request***’***

Use Case ID	UC-TCH-15
Use Case Name	Remove Student
Primary Actor	Teacher
Preconditions	Teacher is logged in; student is associated with the teacher.
Trigger	Teacher selects “Remove Student”.
Basic Flow	1) Teacher selects a student from the students list. 2) Teacher confirms removal. 3) System removes the association between the teacher and the student. 4) System updates related memberships and access.
Alternative/Exception Flows	A1) Student not associated with teacher → System ignores the action or shows error.
Postconditions	Success: Student is no longer associated with the teacher. Failure: Association remains unchanged.

Business Rules	Teacher can remove only students associated with them.

Table 3.3.46-Use Case Specification 'Remove Student'

Use Case ID	UC-TCH-16
Use Case Name	Accept Group Join Request
Primary Actor	Teacher
Preconditions	Teacher is logged in; a pending group join request exists.
Trigger	Teacher selects "Accept" on a group join request.
Basic Flow	1) System displays pending group join requests. 2) Teacher selects a request. 3) Teacher accepts the request. 4) System adds the student to the target group. 5) System marks the request as accepted and notifies the student.
Alternative/Exception Flows	A1) Request already handled → System prevents duplicate action.

	A2) Group is no longer available → System rejects acceptance.
Postconditions	Success: Student becomes a member of the group. Failure: Student is not added to the group.
Business Rules	Only pending requests can be accepted.

Table 3.3.47-Use Case Specification 'Accept Group Join Request***'***

Use Case ID	UC-TCH-17
Use Case Name	Reject Group Join Request
Primary Actor	Teacher
Preconditions	Teacher is logged in; a pending group join request exists.
Trigger	Teacher selects "Reject" on a group join request.
Basic Flow	1) System displays pending group join requests. 2) Teacher selects a request. 3) Teacher rejects the request. 4) System marks the request as rejected and notifies the student.
Alternative/Exception Flows	A1) Request already handled → System prevents duplicate action.

Postconditions	Success: Join request is rejected. Failure: Request remains pending.
Business Rules	Only pending requests can be rejected.

Table 3.3.48-Use Case Specification 'Reject Group Join Request***'***

Use Case ID	UC-TCH-18
Use Case Name	View Question Bank
Primary Actor	Teacher
Preconditions	Teacher is logged in; subject exists and belongs to teacher.
Trigger	Teacher opens the question bank page for a subject.
Basic Flow	1) System retrieves all questions for the selected subject. 2) System displays questions list with key details (type, difficulty, etc.).
Alternative/Exception Flows	A1) No questions found → System displays empty question bank message. A2) Subject not found / not owned → System denies access.

Postconditions	Success: Question bank is displayed. Failure: Question bank is not displayed.
Business Rules	Teacher can view question bank only for their own subjects.

Table 3.3.48-Use Case Specification 'View Question Bank'

Use Case ID	UC-TCH-19
Use Case Name	Create Question (Manually)
Primary Actor	Teacher
Preconditions	Teacher is logged in; subject exists and belongs to teacher.
Trigger	Teacher submits a new question form in the question bank.
Basic Flow	1) System displays question creation form. 2) Teacher selects question type and enters question data (text, options, correct answer, marks). 3) Teacher submits the form. 4) System validates the question data. 5) System saves the question in the question bank.

Alternative/Exception Flows	A1) Invalid input → System shows validation errors and does not save.
Postconditions	Success: New question is created and appears in the question bank. Failure: Question is not created.
Business Rules	Question must include required fields based on its type (e.g., options for MCQ).

Table 3.3.49-Use Case Specification 'Create Question (Manually)***'***

Use Case ID	UC-TCH-20
Use Case Name	Edit Question
Primary Actor	Teacher
Preconditions	Teacher is logged in; question exists under teacher's subject.
Trigger	Teacher selects "Edit" for a question and submits changes.
Basic Flow	1) System displays existing question data in edit form. 2) Teacher updates question fields. 3) Teacher submits changes. 4) System validates updated data. 5) System saves updates to the question.

Alternative/Exception Flows	A1) Invalid input → System shows validation errors and does not save. A2) Question not found / not owned → System denies access.
Postconditions	Success: Question is updated. Failure: Question remains unchanged.
Business Rules	Teacher can edit only questions belonging to their subjects.

Table 3.3.50-Use Case Specification 'Edit Question***'***

Use Case ID	UC-TCH-21
Use Case Name	Delete Question
Primary Actor	Teacher
Preconditions	Teacher is logged in; question exists under teacher's subject.
Trigger	Teacher selects "Delete" for a question and confirms.
Basic Flow	1) Teacher selects a question to delete. 2) System asks for confirmation. 3) Teacher confirms deletion. 4) System deletes the question from the question bank.

Alternative/Exception Flows	A1) Question not found / not owned → System denies action.
Postconditions	Success: Question is deleted. Failure: Question remains available.
Business Rules	Teacher can delete only questions belonging to their subjects.

Table 3.3.51-Use Case Specification 'Delete Question*'*

Use Case ID	UC-TCH-22
Use Case Name	Generate Questions Using AI
Primary Actor	Teacher
Preconditions	Teacher is logged in; subject exists and belongs to teacher.
Trigger	Teacher submits AI generation request (text prompt and/or file).
Basic Flow	1) System displays AI generation form. 2) Teacher provides source content (text or file) and generation settings. 3) Teacher submits the request.

	4) System generates draft questions. 5) System displays generated questions for review. 6) Teacher confirms saving selected questions. 7) System stores saved questions in the question bank.
Alternative/Exception Flows	A1) Invalid/empty input → System shows validation error. A2) Generation fails → System shows failure message and allows retry.
Postconditions	Success: Generated questions are saved to the question bank (fully or partially). Failure: No questions are saved.
Business Rules	Teacher must review before saving AI-generated questions.

Table 3.3.52-Use Case Specification 'Generate Questions Using AI***'***

Use Case ID	UC-TCH-23
Use Case Name	View Exams
Primary Actor	Teacher
Preconditions	Teacher is logged in.
Trigger	Teacher opens the exams page.
Basic Flow	1) System retrieves exams created by the teacher. 2) System displays exams list with key details (status, schedule, subject/group).

Alternative/Exception Flows	A1) No exams found → System displays empty list message.
Postconditions	Success: Exams list is displayed.
Business Rules	Teacher can view only exams they created.

Table 3.3.53-Use Case Specification 'View Exams'

Use Case ID	UC-TCH-24
Use Case Name	Create Exam
Primary Actor	Teacher
Preconditions	Teacher is logged in; at least one subject exists for the teacher.
Trigger	Teacher submits create exam form.
Basic Flow	1) System displays create exam form. 2) Teacher selects subject/group and configures exam settings (title, duration, schedule, rules). 3) Teacher submits the form. 4) System validates exam data. 5) System creates the exam.
Alternative/Exception Flows	A1) Invalid input → System shows validation errors and does not create the exam.
Postconditions	Success: Exam is created and appears in exams list. Failure: Exam is not created.
Business Rules	Exam must be linked to an existing subject (and group if required).

Table 3.3.54-Use Case Specification 'Create Exam'

Use Case ID	UC-TCH-25
Use Case Name	Edit Exam
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam exists and belongs to teacher.
Trigger	Teacher selects "Edit Exam" and submits changes.

Basic Flow	1) System displays exam data in an edit form. 2) Teacher modifies exam settings. 3) Teacher submits changes. 4) System validates updated data. 5) System saves exam updates.
Alternative/Exception Flows	A1) Invalid input → System shows validation errors and does not save. A2) Exam not found / not owned → System denies access.
Postconditions	Success: Exam settings are updated. Failure: Exam remains unchanged.
Business Rules	Teacher can edit only exams they created.

Table 3.3.55-Use Case Specification 'Edit Exam***'***

Use Case ID	UC-TCH-26
Use Case Name	Delete Exam
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam exists and belongs to teacher.
Trigger	Teacher selects "Delete Exam" and confirms deletion.
Basic Flow	1) Teacher selects an exam to delete. 2) System asks for confirmation.

	3) Teacher confirms deletion. 4) System deletes the exam.
Alternative/Exception Flows	A1) Exam is active/has attempts → System prevents deletion. A2) Exam not found / not owned → System denies action.
Postconditions	Success: Exam is deleted. Failure: Exam remains unchanged.
Business Rules	Active exams or exams with attempts cannot be deleted.

Table 3.3.56-Use Case Specification 'Delete Exam***'***

Use Case ID	UC-TCH-27
Use Case Name	Manage Exam Questions
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam exists and belongs to teacher; question bank contains questions (optional).
Trigger	Teacher opens exam questions management page.
Basic Flow	1) System displays current exam questions (if any). 2) Teacher selects questions to add from question bank

	(or available pool). 3) Teacher removes questions (if needed). 4) Teacher saves changes. 5) System updates exam question set.
Alternative/Exception Flows	A1) No available questions → System shows message. A2) Invalid question selection → System rejects update.
Postconditions	Success: Exam question set is updated. Failure: Exam questions remain unchanged.
Business Rules	Only teacher-owned questions/subjects can be used for the exam.

Table 3.3.57-Use Case Specification 'Manage Exam Questions***'***

Use Case ID	UC-TCH-28
Use Case Name	View Exam Results
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam exists and belongs to teacher; exam has at least one submitted attempt.
Trigger	Teacher selects “View Results” for an exam.

Basic Flow	1) System retrieves all student attempts for the selected exam. 2) System calculates and aggregates results (scores, status). 3) System displays exam results list.
Alternative/Exception Flows	A1) No attempts submitted → System displays empty results message. A2) Exam not found / not owned → System denies access.
Postconditions	Success: Exam results are displayed. Failure: Results are not displayed.
Business Rules	Teacher can view results only for exams they created.

Table 3.3.58-Use Case Specification 'View Exam Results***'***

Use Case ID	UC-TCH-29
Use Case Name	Correct Student Attempt
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam exists and belongs to teacher; student attempt exists and requires correction.

Trigger	Teacher selects a student attempt for correction.
Basic Flow	1) System displays the selected student attempt. 2) Teacher reviews student answers. 3) Teacher assigns marks for subjective questions (if any). 4) Teacher submits corrections. 5) System saves grades and updates attempt status.
Alternative/Exception Flows	A1) Attempt already graded → System prevents re-correction. A2) Invalid grading input → System shows validation errors.
Postconditions	Success: Student attempt is graded and results are updated. Failure: Attempt grading remains unchanged.
Business Rules	Only ungraded or partially graded attempts can be corrected.

Table 3.3.59-Use Case Specification 'Correct Student Attempt***'***

Use Case ID	UC-TCH-30
Use Case Name	Monitor Exam
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam exists and is currently active.

Trigger	Teacher opens the exam monitoring page.
Basic Flow	1) System displays live monitoring dashboard for the exam. 2) System shows connected students and exam status.
Alternative/Exception Flows	A1) Exam is not active → System denies monitoring access.
Postconditions	Success: Exam monitoring dashboard is displayed.
Business Rules	Only active exams can be monitored.

Table 3.3.60-Use Case Specification 'Monitor Exam'

Use Case ID	UC-TCH-31
Use Case Name	View Live Monitor Data
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam monitoring session is active.
Trigger	Teacher views live monitoring data panel.
Basic Flow	1) System retrieves live monitoring data for students. 2) System displays real-time status indicators (events, alerts).
Alternative/Exception Flows	A1) Monitoring data unavailable → System displays partial or empty data.
Postconditions	Success: Live monitoring data is displayed.
Business Rules	Live data is read-only for the teacher.

Table 3.3.61-Use Case Specification 'View Live Monitor Data'

Field	**Content**
Use Case ID	UC-TCH-32
Use Case Name	Monitor Student Audio Stream

Primary Actor	Teacher
Preconditions	Teacher is logged in; exam monitoring session is active; student audio stream is available.
Trigger	Teacher selects a student audio stream.
Basic Flow	1) System connects to the selected student's audio stream. 2) System plays the live audio to the teacher.
Alternative/Exception Flows	A1) Audio stream unavailable → System shows error message.
Postconditions	Success: Student audio stream is monitored.
Business Rules	Audio monitoring is permitted only during active exams.

Table 3.3.62-Use Case Specification 'Monitor Student Audio Stream'

Field	**Content**
Use Case ID	UC-TCH-33
Use Case Name	View Student Camera Snapshots
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam monitoring session is active; camera data exists.
Trigger	Teacher opens camera snapshots for a student.
Basic Flow	1) System retrieves camera snapshots for the selected student. 2) System displays the snapshots to the teacher.
Alternative/Exception Flows	A1) No snapshots available → System displays empty view.
Postconditions	Success: Student camera snapshots are displayed.
Business Rules	Camera snapshots are view-only.

Table 3.3.63-Use Case Specification 'View Student Camera Snapshots'

Use Case ID	UC-TCH-34
Use Case Name	Perform Student Monitor Action
Primary Actor	Teacher

Preconditions	Teacher is logged in; exam monitoring session is active; student exists.
Trigger	Teacher selects a monitoring action for a student.
Basic Flow	1) Teacher selects a student. 2) Teacher selects a monitoring action (e.g., warn, flag). 3) System applies the action and logs the event.
Alternative/Exception Flows	A1) Invalid action → System rejects request.
Postconditions	Success: Monitoring action is applied and recorded.
Business Rules	All monitoring actions must be logged.

Table 3.3.64-Use Case Specification 'Perform Student Monitor Action'

Use Case ID	UC-TCH-35
Use Case Name	Send Exam Notification
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam monitoring session is active.
Trigger	Teacher submits an exam notification.
Basic Flow	1) Teacher enters notification content. 2) Teacher submits the notification. 3) System sends the notification to exam participants.
Alternative/Exception Flows	A1) Empty content → System shows validation error.
Postconditions	Success: Notification is sent to students.
Business Rules	Notifications are sent only to students in the active exam.

Table 3.3.65-Use Case Specification 'Send Exam Notification'

Use Case ID	UC-TCH-36
Use Case Name	End Monitoring Session
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam monitoring session is active.
Trigger	Teacher selects “End Monitoring Session”.
Basic Flow	1) Teacher confirms ending the monitoring session. 2) System ends the monitoring session and disconnects streams.
Alternative/Exception Flows	A1) Session already ended → System ignores request.
Postconditions	Success: Monitoring session is ended.
Business Rules	Ending monitoring does not terminate the exam itself.

Table 3.3.66-Use Case Specification ‘End Monitoring Session***’***

Use Case ID	UC-TCH-37
Use Case Name	Talk to Student During Exam
Primary Actor	Teacher
Preconditions	Teacher is logged in; exam monitoring session is active; student audio channel is available.
Trigger	Teacher activates voice communication with a student.
Basic Flow	1) Teacher selects a student during monitoring. 2) Teacher activates voice communication. 3) System streams teacher’s audio to the selected student. 4) Teacher speaks to the student.
Alternative/Exception Flows	A1) Student audio channel unavailable → System shows error message.
Postconditions	Success: Student hears teacher’s voice during the exam. Failure: No audio is transmitted.
Business Rules	Voice communication is allowed only during active exam monitoring sessions.

Table 3.3.67-Use Case Specification ‘Talk to Student During Exam***’***

Students

Use Case ID	UC-STD-01
Use Case Name	View Teachers
Primary Actor	Student
Preconditions	Student is logged in.
Trigger	Student opens teachers list page.
Basic Flow	1) System retrieves available teachers. 2) System displays teachers list with basic information.
Alternative/Exception Flows	A1) No teachers available → System displays empty list message.
Postconditions	Success: Teachers list is displayed.
Business Rules	Teachers list is read-only for students.

Table 3.3.68-Use Case Specification 'View Teachers'

Use Case ID	UC-STD-02
Use Case Name	Send Join Request
Primary Actor	Student
Preconditions	1) Student is authenticated (logged in). 2) Target teacher exists.
Trigger	Student selects a teacher and clicks "Send Join Request".
Basic Flow	1) System displays available teachers. 2) Student selects a teacher. 3) Student submits a join request. 4) System records the request as pending. 5) System notifies the selected teacher.
Alternative/Exception Flows	A1) Join request already exists → System prevents duplicate request and shows a message. A2) Teacher not available → System displays an error message.
Postconditions	Success: Join request is created with pending status. Failure: No join request is created.
Business Rules	A student can have only one pending join request per teacher.

Table 3.3.69-Use Case Specification 'Send Join Request'

Use Case ID	UC-STD-03
Use Case Name	View Joined Teachers
Primary Actor	Student
Preconditions	Student is logged in.
Trigger	Student opens joined teachers page.
Basic Flow	1) System retrieves teachers associated with the student. 2) System displays joined teachers list.
Alternative/Exception Flows	A1) No joined teachers → System displays empty list message.
Postconditions	Success: Joined teachers list is displayed.
Business Rules	Student can view only teachers they are associated with.

Table 3.3.70-Use Case Specification 'View Joined Teachers'

Use Case ID	UC-STD-04
Use Case Name	View Groups
Primary Actor	Student
Preconditions	Student is logged in.
Trigger	Student opens groups page.
Basic Flow	1) System retrieves groups available to the student. 2) System displays groups list.
Alternative/Exception Flows	A1) No groups available → System displays empty list message.
Postconditions	Success: Groups list is displayed.
Business Rules	Groups visibility depends on student's teacher associations.

Table 3.3.71-Use Case Specification 'View Groups'

Use Case ID	UC-STD-05
Use Case Name	Send Group Join Request
Primary Actor	Student
Preconditions	Student is logged in; group exists.
Trigger	Student selects “Join Group”.
Basic Flow	1) Student selects a group. 2) Student submits join request. 3) System records the request and notifies the teacher.
Alternative/Exception Flows	A1) Already a group member → System prevents request.
Postconditions	Success: Group join request is pending. Failure: Request is not created.
Business Rules	Student can request to join a group only once.

Table 3.3.72-Use Case Specification ‘Send Group Join Request***’***

Use Case ID	UC-STD-06
Use Case Name	View Invitations
Primary Actor	Student
Preconditions	1) Student is logged in.
Trigger	Student opens invitations page/section.
Basic Flow	1) System retrieves pending invitations for the student. 2) System displays invitations list.
Alternative/Exception Flows	A1) No invitations available → System displays an empty list message.
Postconditions	Success: Invitations list is displayed.
Business Rules	Student can view only their own invitations.

Table 3.3.73-Use Case Specification ‘View Invitations***’***

Use Case ID	UC-STD-07
Use Case Name	Accept Invitation
Primary Actor	Student
Preconditions	1) Student is authenticated (logged in). 2) A pending invitation exists.
Trigger	Student selects “Accept” on an invitation.
Basic Flow	1) System validates the invitation status. 2) System marks the invitation as accepted. 3) System grants the related access (teacher or group).
Alternative/Exception Flows	A1) Invitation already handled → System rejects the action.
Postconditions	Success: Student becomes associated according to the invitation type.
Business Rules	Only pending invitations can be accepted.

Table 3.3.74-Use Case Specification ‘Accept Invitation***’***

Use Case ID	UC-STD-08
Use Case Name	Reject Invitation
Primary Actor	Student
Preconditions	1) Student is logged in. 2) A pending invitation exists.
Trigger	Student selects “Reject” on an invitation.
Basic Flow	1) System validates the invitation status. 2) System marks the invitation as rejected.
Alternative/Exception Flows	A1) Invitation already handled → System rejects the action.
Postconditions	Success: Invitation is rejected and no access is granted.
Business Rules	Only pending invitations can be rejected.

Table 3.3.75-Use Case Specification ‘Reject Invitation***’***

Use Case ID	UC-STD-09
Use Case Name	View Available Exams
Primary Actor	Student
Preconditions	1) Student is logged in.
Trigger	Student opens the exams page.
Basic Flow	1) System retrieves exams available to the student. 2) System displays the exams list with schedule and status.
Alternative/Exception Flows	A1) No exams available → System displays an empty list message.
Postconditions	Success: Available exams are displayed.
Business Rules	Student can view only exams assigned to their teachers or groups.

Table 3.3.76-Use Case Specification 'View Available Exams***'***

Use Case ID	UC-STD-10
Use Case Name	View Exam Events
Primary Actor	Student
Preconditions	1)Student is logged in. 2)exam attempt exists.
Trigger	Student opens exam events/logs page.
Basic Flow	1) System retrieves exam details. 2) System displays exam information (rules, duration, schedule).
Alternative/Exception Flows	A1) No events logged → System displays empty timeline.
Postconditions	Success: Exam events are displayed.
Business Rules	Exam events are read-only.

Table 3.3.77-Use Case Specification 'View Exam Events***'***

Use Case ID	UC-STD-11
Use Case Name	Take Exam
Primary Actor	Student
Preconditions	1) Student is logged in. 2) Exam is active and available.
Trigger	Student clicks “Start Exam”.
Basic Flow	1) System starts the exam session. 2) System displays exam questions. 3) Student answers questions. 4) Student submits the exam. 5) System records the attempt.
Alternative/Exception Flows	A1) Exam time expires → System auto-submits the attempt.
Postconditions	Success: Exam attempt is submitted.
Business Rules	Exam rules (time, attempts, monitoring) must be enforced.

Table 3.3.78-Use Case Specification ‘Take Exam***’***

Use Case ID	UC-STD-12
Use Case Name	View Exam Result
Primary Actor	Student
Preconditions	1) Student is authenticated logged in. 2) Exam result is published.
Trigger	Student selects “View Exam Result”.
Basic Flow	1) System retrieves the student’s exam result. 2) System displays score and status.
Alternative/Exception Flows	A1) Result not published → System displays a message.
Postconditions	Success: Exam result is displayed.
Business Rules	Student can view only their own exam results.

Table 3.3.79-Use Case Specification ‘View Exam Result***’***

Use Case ID	UC-STD-13
Use Case Name	View Exam Events
Primary Actor	Student
Preconditions	1) Student islogged in. 2) Exam attempt exists.
Trigger	Student opens exam events/logs page.
Basic Flow	1) System retrieves logged exam events. 2) System displays events timeline.
Alternative/Exception Flows	A1) No events available → System displays an empty timeline.
Postconditions	Success: Exam events are displayed.
Business Rules	Exam events are read-only.

Table 3.3.80-Use Case Specification 'View Exam Events'

Use Case ID	UC-STD-14
Use Case Name	View Exam Notifications
Primary Actor	Student
Preconditions	1) Student is logged in.
Trigger	Student opens exam notifications panel.
Basic Flow	1) System retrieves exam-related notifications. 2) System displays notifications list.
Alternative/Exception Flows	A1) No notifications available → System displays an empty list.
Postconditions	Success: Exam notifications are displayed.
Business Rules	Notifications are read-only for students.

Table 3.3.81-Use Case Specification 'View Exam Notifications'

Use Case ID	UC-STD-15
Use Case Name	View Group Details
Primary Actor	Student
Preconditions	1) Student is logged in. 2) Student is a member of the group or has access to it.
Trigger	Student selects a group from the groups list.
Basic Flow	1) System retrieves selected group details. 2) System displays group information (name, teacher, members, related exams).
Alternative/Exception Flows	A1) Group not found or not accessible → System displays an error message.
Postconditions	Success: Group details are displayed.
Business Rules	Student can view only groups they belong to or are allowed to access.

Table 3.3.82-Use Case Specification 'View Group Details'

5.3-Activity Diagram:

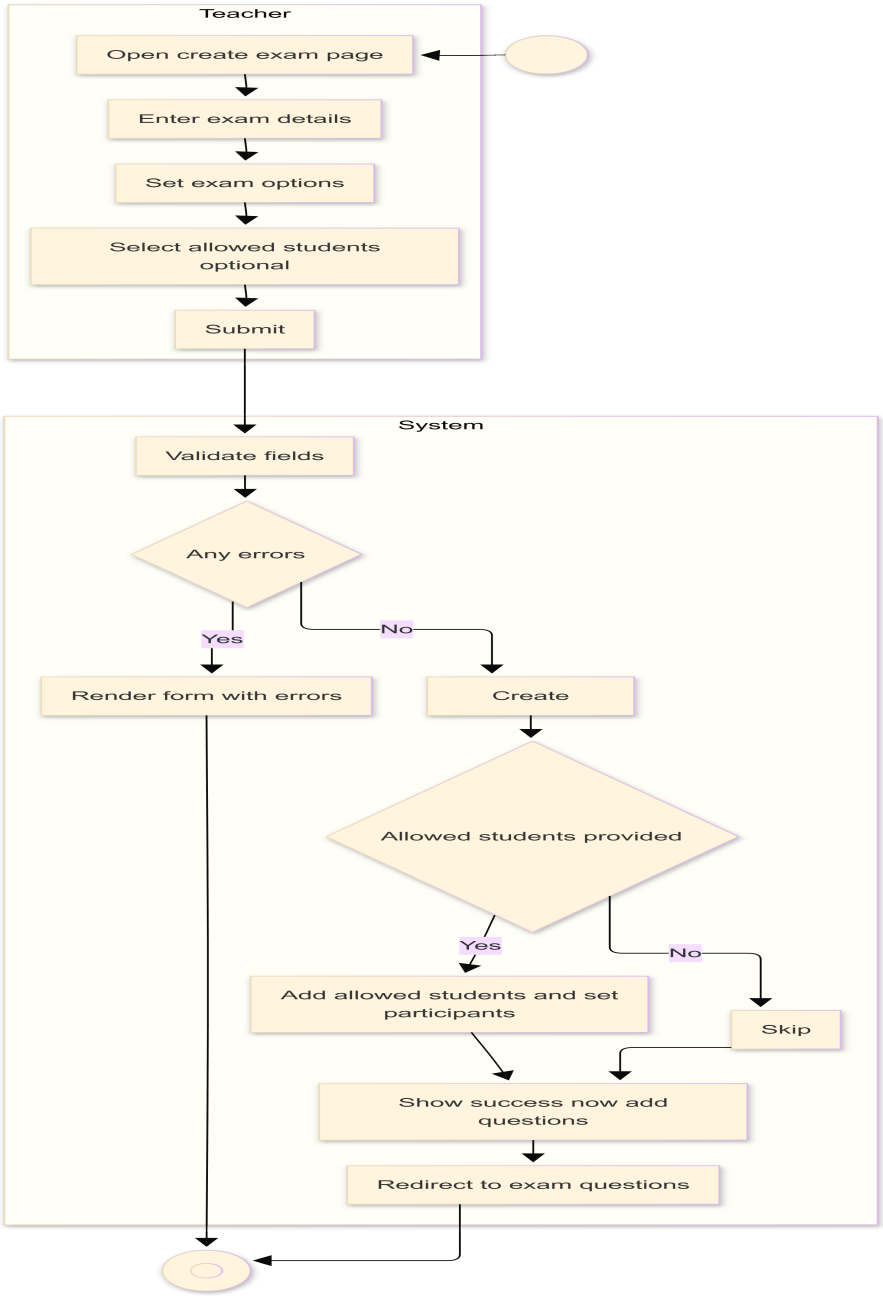


Figure 3.4.1-Activity Diagram for 'Create Exam'

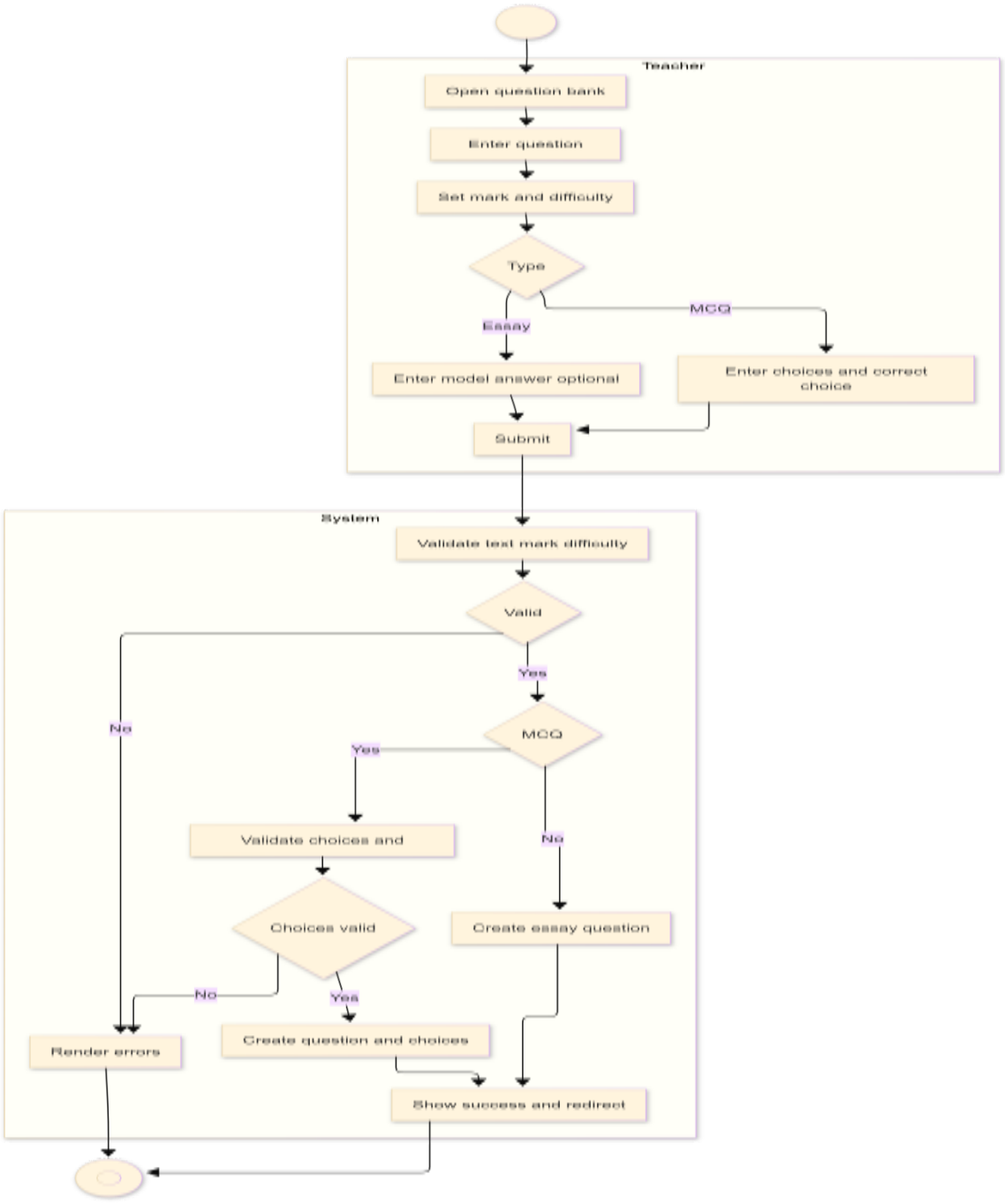


Figure 3.4.2-Activity Diagram for 'Create Questions (Manually)'

Figure 3.4.3-Activity Diagram for 'Create Subject'

Figure 3.4.4-Activity Diagram for 'Genarate Questions Using AI'

Figure 3.4.5-Activity Diagram for 'Monitor Exam'

Figure 3.4.5-Activity Diagram for 'Reset Password'

Figure 3.4.6-Activity Diagram for 'Take Exam'

Figure 3.4.7-Activity Diagram for 'Verify Email'

Figure 3.4.8-Activity Diagram for 'View Exam Notifications'

Figure 3.4.9-Activity Diagram for 'Monitor Student Audio Stream'

Figure 3.4.13-Activity Diagram for 'View Student Camera Snapshots'

5.4-Entity Relation Diagram:

5.4.1-Conceptual ERD:

Figure 3.5-Conceptual ERD

5.5-Sequence Diagram:

Figure 3.6.1-Sequence Diagram 'Create Exam'

Figure 3.6.2-Sequence Diagram 'Create Questions (Manually)'

Figure 3.6.3-Sequence Diagram 'Create Subject'

Figure 3.6.4-Sequence Diagram 'Generate Questions Using AI'

Figure 3.6.5-Sequence Diagram 'Reset Password'

Figure 3.6.6-Sequence Diagram 'Take Exam'

Figure 3.6.7-Sequence Diagram 'Verfiy Email Address'

Figure 3.6.8-Sequence Diagram 'View Exam Notifications'

Table name	Attribute	Description	Type
EXAM	auto_fail_on_cheating	Whether the system auto-fails attempts when cheating is detected.	boolean
EXAM	auto_proctoring	Whether proctoring is enabled automatically.	boolean
EXAM	created_at	Creation timestamp.	datetime
EXAM	duration_minutes	Duration in minutes.	int
EXAM	end_time	Scheduled end time (or computed end time).	datetime
EXAM	id	Primary key identifier for an exam.	bigint (PK)
EXAM	late_join_minutes	Grace period for late joining after start.	int
EXAM	marking_type	Marking mode (e.g., auto/manual/mixed).	string
EXAM	pass_mark	Minimum marks required to pass.	int
EXAM	shuffle_questions	Whether questions are shuffled per student.	boolean
EXAM	start_time	Scheduled start time.	datetime
EXAM	status	Exam status (e.g., draft/published/active/ended).	string
EXAM	subject_id	Subject the exam belongs to.	bigint (FK -> SUBJECT.id)
EXAM	submitted_count	Number of submitted attempts (cached counter).	int
EXAM	teacher_id	Teacher who created/owns the exam.	bigint (FK -> USER.id)
EXAM	title	Exam title.	string
EXAM	total_mark	Total marks available in the exam.	int
EXAM	total_participants	Number of students assigned/participating (cached counter).	int

EXAM	updated_at	Last update timestamp.	datetime
EXAM_ALLOWED_STUDENTS	exam_id	Exam the student is allowed to take.	bigint (FK -> EXAM.id)
EXAM_ALLOWED_STUDENTS	id	Primary key identifier for exam allowlist record.	bigint (PK)
EXAM_ALLOWED_STUDENTS	user_id	Student user allowed for the exam.	bigint (FK -> USER.id)
EXAM_EVENT	created_at	Creation timestamp.	datetime
EXAM_EVENT	event_type	Event type label (e.g., warning, cheating, status change).	string
EXAM_EVENT	exam_id	Exam associated with the event.	bigint (FK -> EXAM.id)
EXAM_EVENT	id	Primary key identifier for a recorded exam event.	bigint (PK)
EXAM_EVENT	message	Event details or message.	text
EXAM_EVENT	student_id	Student associated with the event.	bigint (FK -> USER.id)
EXAM_NOTIFICATION	created_at	Creation timestamp.	datetime
EXAM_NOTIFICATION	exam_id	Exam associated with the notification.	bigint (FK -> EXAM.id)
EXAM_NOTIFICATION	id	Primary key identifier for an exam notification.	bigint (PK)
EXAM_NOTIFICATION	message	Notification message content.	text
EXAM_NOTIFICATION	recipient_id	User who receives the notification.	bigint (FK -> USER.id)
EXAM_NOTIFICATION	sender_id	User who sent the notification.	bigint (FK -> USER.id)
EXAM_QUESTION	exam_id	Exam that includes the question.	bigint (FK -> EXAM.id)
EXAM_QUESTION	id	Primary key identifier for linking a question to an exam.	bigint (PK)
EXAM_QUESTION	mark	Mark assigned to this question in this exam (may override default).	decimal

EXAM_QUESTION	order	Display/order index of the question within the exam.	int
EXAM_QUESTION	question_id	Question included in the exam.	bigint (FK -> QUESTION.id)
GROUP	code	Join/access code for the group.	string
GROUP	created_at	Creation timestamp.	datetime
GROUP	id	Primary key identifier for a group (class/group within a subject).	bigint (PK)
GROUP	name	Group name.	string
GROUP	subject_id	Subject this group belongs to.	bigint (FK -> SUBJECT.id)
GROUP	teacher_id	Teacher who manages the group.	bigint (FK -> USER.id)
GROUP_MESSAGE	content	Message content/body.	text
GROUP_MESSAGE	created_at	Creation timestamp.	datetime
GROUP_MESSAGE	group_id	Group where the message was posted.	bigint (FK -> GROUP.id)
GROUP_MESSAGE	id	Primary key identifier for a group chat message.	bigint (PK)
GROUP_MESSAGE	sender_id	User who sent the group message.	bigint (FK -> USER.id)
GROUP_STUDENTS	group_id	Group the student belongs to.	bigint (FK -> GROUP.id)
GROUP_STUDENTS	id	Primary key identifier for group membership record.	bigint (PK)
GROUP_STUDENTS	user_id	Student user who is a member of the group.	bigint (FK -> USER.id)
MESSAGE	body	Message content/body.	text
MESSAGE	category	Message category label (e.g., report, notification).	string
MESSAGE	created_at	Creation timestamp.	datetime
MESSAGE	direction	Message direction/type used by the system (e.g., report, reply).	string

MESSAGE	id	Primary key identifier for a direct message/report.	bigint (PK)
MESSAGE	in_reply_to_id	Optional reference to another message this one replies to.	bigint (FK -> MESSAGE.id)
MESSAGE	is_read	Whether the recipient has read the message.	boolean
MESSAGE	recipient_id	User who receives the message.	bigint (FK -> USER.id)
MESSAGE	sender_id	User who sent the message.	bigint (FK -> USER.id)
MESSAGE	title	Message subject/title.	string
PROCTOR_AUDIO_STREAM	answer_sdp	WebRTC answer SDP returned by peer.	text
PROCTOR_AUDIO_STREAM	bytes_received	Total bytes received on the audio stream.	bigint
PROCTOR_AUDIO_STREAM	ended_at	Stream end timestamp.	datetime
PROCTOR_AUDIO_STREAM	ice_candidates	ICE candidates exchanged for WebRTC.	json
PROCTOR_AUDIO_STREAM	id	Primary key identifier for an audio stream record.	bigint (PK)
PROCTOR_AUDIO_STREAM	last_activity_at	Last activity timestamp for the stream.	datetime
PROCTOR_AUDIO_STREAM	offer_sdp	WebRTC offer SDP used to establish the connection.	text
PROCTOR_AUDIO_STREAM	packets_lost	Packet loss counter for stream quality monitoring.	int
PROCTOR_AUDIO_STREAM	session_id	Proctor session this audio stream belongs to.	bigint (FK -> PROCTOR_SESSION.id)
PROCTOR_AUDIO_STREAM	started_at	Stream start timestamp.	datetime
PROCTOR_AUDIO_STREAM	status	Stream status (e.g., connecting/active/ended).	string

PROCTOR_SESSION	camera_enabled	Whether camera monitoring is enabled.	boolean
PROCTOR_SESSION	created_at	Creation timestamp.	datetime
PROCTOR_SESSION	ended_at	Session end timestamp.	datetime
PROCTOR_SESSION	exam_id	Exam being proctored.	bigint (FK -> EXAM.id)
PROCTOR_SESSION	id	Primary key identifier for a proctoring session.	bigint (PK)
PROCTOR_SESSION	is_active	Whether the proctor session is currently active.	boolean
PROCTOR_SESSION	last_snapshot	Path/filename of the last captured snapshot.	string
PROCTOR_SESSION	last_snapshot_at	Timestamp of the last captured snapshot.	datetime
PROCTOR_SESSION	microphone_enabled	Whether microphone monitoring is enabled.	boolean
PROCTOR_SESSION	peer_connection_data	WebRTC peer connection state/data for proctoring streams.	json
PROCTOR_SESSION	snapshots_count	Count of captured snapshots in this session.	int
PROCTOR_SESSION	student_exam_id	Attempt being monitored.	bigint (FK -> STUDENT_EXAM.id)
PROCTOR_SESSION	student_id	Student being monitored.	bigint (FK -> USER.id)
PROCTOR_SESSION	updated_at	Last update timestamp.	datetime
PROCTOR_SESSION	warnings_count	Count of warnings raised during monitoring.	int
QUESTION	created_at	Creation timestamp.	datetime
QUESTION	difficulty	Difficulty level label.	string
QUESTION	id	Primary key identifier for a question.	bigint (PK)
QUESTION	mark	Default mark for the question.	decimal

QUESTION	model_answer	Reference/model answer (especially for essay questions).	text
QUESTION	question_type	Question type (e.g., MCQ/Essay).	string
QUESTION	subject_id	Subject the question belongs to.	bigint (FK -> SUBJECT.id)
QUESTION	teacher_id	Teacher who created/owns the question.	bigint (FK -> USER.id)
QUESTION	text	Question text/body.	text
QUESTION	updated_at	Last update timestamp.	datetime
QUESTION_CHOICE	id	Primary key identifier for a multiple-choice option.	bigint (PK)
QUESTION_CHOICE	is_correct	Whether this choice is the correct answer.	boolean
QUESTION_CHOICE	order	Order index of the choice.	int
QUESTION_CHOICE	question_id	Question this choice belongs to.	bigint (FK -> QUESTION.id)
QUESTION_CHOICE	text	Choice text.	string
STUDENT_ANSWER	answered_at	Timestamp when the answer was recorded.	datetime
STUDENT_ANSWER	attempt_id	Student attempt this answer belongs to.	bigint (FK -> STUDENT_EXAM.id)
STUDENT_ANSWER	essay_text	Written answer text (for essay).	text
STUDENT_ANSWER	exam_question_id	Exam question being answered.	bigint (FK -> EXAM_QUESTION.id)
STUDENT_ANSWER	id	Primary key identifier for an answer submitted by a student.	bigint (PK)
STUDENT_ANSWER	is_correct	Whether the answer is correct (auto-graded where applicable).	boolean
STUDENT_ANSWER	mark_obtained	Marks obtained for this answer.	decimal

STUDENT_ANSWER	selected_choice_id	Selected choice (for MCQ).	bigint (FK -> QUESTION_CHOICE.id)
STUDENT_EXAM	created_at	Creation timestamp.	datetime
STUDENT_EXAM	exam_id	Exam being attempted.	bigint (FK -> EXAM.id)
STUDENT_EXAM	finished_at	Attempt finish/submit timestamp.	datetime
STUDENT_EXAM	id	Primary key identifier for a student's exam attempt.	bigint (PK)
STUDENT_EXAM	last_activity_at	Last activity timestamp (used for monitoring/inactivity).	datetime
STUDENT_EXAM	score	Score achieved by the student.	decimal
STUDENT_EXAM	started_at	Attempt start timestamp.	datetime
STUDENT_EXAM	status	Attempt status (e.g., in_progress/submitted/graded).	string
STUDENT_EXAM	student_id	Student who is attempting the exam.	bigint (FK -> USER.id)
STUDENT_JOIN_REQUEST	created_at	Creation timestamp.	datetime
STUDENT_JOIN_REQUEST	group_id	Optional group requested to join (if applicable).	bigint (FK -> GROUP.id)
STUDENT_JOIN_REQUEST	id	Primary key identifier for a join request.	bigint (PK)
STUDENT_JOIN_REQUEST	source	Request source/type (e.g., general join vs group join).	string
STUDENT_JOIN_REQUEST	status	Request status (e.g., pending/accepted/rejected).	string
STUDENT_JOIN_REQUEST	student_id	Student who requests to join.	bigint (FK -> USER.id)
STUDENT_JOIN_REQUEST	teacher_id	Teacher being requested to join under.	bigint (FK -> USER.id)
SUBJECT	created_at	Creation timestamp.	datetime
SUBJECT	description	Subject/course description.	string

SUBJECT	id	Primary key identifier for a subject/course.	bigint (PK)
SUBJECT	image	Subject image/logo path.	string
SUBJECT	name	Subject/course name.	string
SUBJECT	teacher_id	Teacher who owns/creates the subject.	bigint (FK -> USER.id)
SUBJECT	updated_at	Last update timestamp.	datetime
SUBJECT_STUDENTS	id	Primary key identifier for subject enrollment record.	bigint (PK)
SUBJECT_STUDENTS	subject_id	Subject being enrolled into.	bigint (FK -> SUBJECT.id)
SUBJECT_STUDENTS	user_id	Student user enrolled in the subject.	bigint (FK -> USER.id)
SYSTEM_SETTINGS	id	Primary key identifier for system settings record.	bigint (PK)
SYSTEM_SETTINGS	platform_name	Displayed name of the platform (branding).	string
SYSTEM_SETTINGS	system_icon	Path/filename for the platform icon/logo.	string
SYSTEM_SETTINGS	two_factor_app_password	App password used by the platform to authenticate with the email service for 2FA.	string
SYSTEM_SETTINGS	two_factor_email	Email address used for sending 2FA OTP emails.	string
USER	email	User email address (also used for verification/2FA).	string
USER	full_name	User's full display name.	string
USER	id	Primary key identifier for a user account.	bigint (PK)
USER	is_active	Whether the user account is active (access enabled).	boolean
USER	role	User role (e.g., admin/teacher/student) controlling access	string

		permissions.	
USER	username	Unique username used for login.	string

Table 3.7-DATA DICTIONARY

7-TEST CASES:

ID	USECASE ID	DESCRIPTION	INPUT	EXPECTED RESULT
TC-01	UC-COM-01	Admin login succeeds and redirects to admin dashboard.	**POST** `/accounts/login/` with `{username=admin1, password=Pass1234!}`	User authenticated; redirect to `/admin-dashboard/`.
TC-02	UC-COM-01	Login fails with wrong password.	**POST** `/accounts/login/` with `{username=admin1, password=WrongPass}`	Login page re-renders with validation errors; no session created.
TC-03	UC-COM-02	Logout clears session.	**POST** `/accounts/logout/` while logged in	User is logged out; redirect to `/accounts/login/`.
TC-04	UC-PAIR-01	Email verification activates account and logs user in.	Precondition: signup created user with `is_active=False` + session contains `verification_code`. **POST** `/accounts/verify-email/` with `otp=<correct_code>`	User becomes `is_active=True`; auto-login; redirect by role (admin→admin_dashboard, teacher→teacher_dashboard, student→student_dashboard).
TC-05	UC-PAIR-02	Resend verification code updates session OTP.	Precondition: session has `verification_user_id`. **GET** `/accounts/resend-code/`	New OTP stored in session; success message; redirect to `/accounts/verify-email/`.
TC-06	UC-ADM-02	Admin creates a new user (student).	**POST** `/admin-dashboard/users/create/` with `{username=stud2, full_name=Student Two, email=stud2@example.com, role=student, password1=Pass1234!, password2=Pass1234!}`	User created with `is_active=True`; success message; redirect to `/admin-dashboard/users/`.
TC-07	UC-ADM-03	Admin blocks a user (toggle	**POST** `/admin-dashboard/users/` with `{user_id=<stud2_id>, action=toggle_block}`	Target user `is_active=False`; success message; blocked user cannot log in.

		is_active).		
TC-08	UC-ADM-05	Admin deletes a user.	**POST** `/admin-dashboard/users/` with `{user_id=<stud2_id>, action=delete}`	User removed from DB; success message; user no longer appears in list.
TC-09	UC-ADM-20	Admin updates **App Password** for 2FA email sending.	**POST** `/admin-dashboard/settings/` with `system_submit=1` and `{two_factor_email=platform2fa@example.com, two_factor_app_password=app-pass-123}`	`SystemSettings.two_factor_app_password` updated; "System settings updated" success message.
TC-10	UC-ADM-21	Admin updates platform branding name/icon.	**POST** `/admin-dashboard/settings/` with `system_submit=1` and `{platform_name=Smart Exams, system_icon=<png file>}`	`SystemSettings.platform_name` + icon updated; branding reflected in UI templates that read settings.
TC-11	UC-STD-02	Student sends join request to a teacher (no duplicates).	**GET** `/student-dashboard/join-request/<teacher_id>/` as `stud1`	`StudentJoinRequest` created with `status=pending`, `source=from_student`, `group=NULL`; success message; redirect to student teachers page.
TC-12	UC-TCH-13	Teacher accepts a **general join request**.	**POST** `/teacher-dashboard/students/` with `{action=accept_general, request_id=<join_request_id>}`	Join request status becomes `accepted`; success message; student appears in teacher's students lists.
TC-13	UC-TCH-24	Teacher creates an exam with allowed students and proctoring options.	**POST** `/teacher-dashboard/exams/create/` with `{title=Midterm, subject=<subject_id>, total_mark=100, duration_minutes=60, late_join_minutes=10, start_time=2026-02-01T10:00, auto_proctoring=on, auto_fail_on_cheating=on, selected_students=<stud1_id>}`	Exam created with `status=scheduled`; allowed_students includes stud1; redirect to `/teacher-dashboard/exams/<exam_id>/questions/`.
TC-14	UC-STD-11	Student starts exam (attempt created + JOIN event).	**GET** `/student-dashboard/exams/<exam_id>/take/` as `stud1` within allowed time	`StudentExam` created (`status=in_progress`); `ExamEvent` created with type `join`; first question displayed with timer.
TC-15	UC-STD-11	Student submits	**POST** `/student-dashboard/exams/<exam_id>/take/?q=1`	`StudentAnswer` saved; `StudentExam` becomes

		exam (answers saved + score calculated).	with `{choice=<correct_choice_id>, action=submit, current_index=1}`	`finished`; score computed (sum marks_obtained); `ExamEvent submit` recorded; redirect to `/student-dashboard/exams/` with success message.
TC-16	UC-STD-12	Student views exam result after finishing.	**GET** `/student-dashboard/exams/<exam_id>/result/` as `stud1`	Result page shows score/percent, pass/fail (>=50%), correct/wrong counts, time spent.
TC-17	UC-TCH-31	Teacher fetches live monitor data (JSON).	**GET** `/teacher-dashboard/exams/<exam_id>/monitor/data/` as `teach1`	JSON `{ok:true, stats:{total_students, active_students, submitted, absent}, students:[...], remaining_seconds, activity_log:[...]}`.
TC-18	UC-TCH-35	Teacher sends exam notification to all allowed students.	**POST** `/teacher-dashboard/exams/<exam_id>/monitor/notify/` with `{message=Keep camera on, recipient_id=''}`	`ExamNotification` created for each allowed student; success message; redirect to monitor page.
TC-19	UC-TCH-34	Teacher forces submission of a student attempt.	**POST** `/teacher-dashboard/exams/<exam_id>/monitor/students/<stud1_id>/action/` with `{action=force_submit}`	If attempt was in_progress → becomes finished, score computed, notification created for student; success message.
TC-20	UC-TCH-30	Student initializes proctoring session (API).	**POST** `/proctor/<exam_id>/init/` as `stud1` (attempt exists)	JSON `{ok:true, session_id, audio_stream_id}`; `ProctorSession.is_active=True` linked to exam/student/attempt.

***Table 3.8*-TEST CASES**

8-Requirements Traceability Matrix(RTM):

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-01	View Users				TC-07
UC-ADM-02	Create User				TC-08
UC-ADM-03	Block User				
UC-ADM-04	Unblock User				TC-09
UC-ADM-05	Delete User				
UC-ADM-06	View Teachers				
UC-ADM-07	Create Teacher				
UC-ADM-08	View Analytics Dashboard				
UC-ADM-09	View Groups				

Table 3.20-RTM

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-10	View Exams				
UC-ADM-11	View Exam Details				
UC-ADM-12	View Exam Results				
UC-ADM-13	Monitor Exam				TC-22
UC-ADM-14	View Reports Inbox				TC-22
UC-ADM-15	View Report				TC-22
UC-ADM-16	Send Report				
UC-ADM-17	Reply to Report				
UC-ADM-18	View System Settings				
UC-ADM-19	Update Email Settings				TC-06
UC-ADM-20	Update App Password				TC-10

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-21	Update Platform Branding				TC-01, TC-02
UC-COM-01	Log In				TC-03
UC-COM-02	Log Out				
UC-COM-03	Reset Password				
UC-COM-04	Update Profile				
UC-COM-05	Update Profile Information				TC-06
UC-COM-06	Change Password				TC-04
UC-PAIR-01	Verify Email Address				TC-05
UC-PAIR-02	Resend Verification Code				
UC-PAIR-03	View Messages Inbox				TC-22, TC-23
UC-PAIR-04	Send Message				

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-PAIR-05	Reply to Message				
UC-PAIR-06	View Group Messages				
UC-PAIR-07	Send Group Message				
UC-STD-01	View Teachers				
UC-STD-02	Send Join Request				
UC-STD-03	View Joined Teachers				
UC-STD-04	View Groups				
UC-STD-05	Send Group Join Request				
UC-STD-06	View Invitations				
UC-STD-07	Accept Invitation				
UC-STD-08	Reject Invitation				

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-STD-09	View Available Exams				
UC-STD-10	View Exam Events				TC-19, TC-20
UC-STD-11	Take Exam				TC-21
UC-STD-12	View Exam Result				TC-28
UC-STD-13	View Exam Events				
UC-STD-14	View Exam Notifications				
UC-STD-15	View Group Details				
UC-TCH-01	View Subjects				TC-11
UC-TCH-02	Create Subject				
UC-TCH-03	Edit Subject				
UC-TCH-04	Delete Subject				

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-TCH-05	View Subject Students				
UC-TCH-06	View Groups				TC-12
UC-TCH-07	Create Group				
UC-TCH-08	View Group Details				
UC-TCH-09	Edit Group				
UC-TCH-10	Delete Group				
UC-TCH-11	View Students				TC-13
UC-TCH-12	Invite Student				TC-14
UC-TCH-13	Accept Join Request				
UC-TCH-14	Reject Join Request				
UC-TCH-15	Remove Student				

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-TCH-16	Accept Group Join Request				
UC-TCH-17	Reject Group Join Request				
UC-TCH-18	View Question Bank				TC-16
UC-TCH-19	Create Question (Manually)				
UC-TCH-20	Edit Question				
UC-TCH-21	Delete Question				
UC-TCH-22	Generate Questions Using AI				
UC-TCH-23	View Exams				TC-15
UC-TCH-24	Create Exam				
UC-TCH-25	Edit Exam				
UC-TCH-26	Delete Exam				TC-17

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-TCH-27	Manage Exam Questions				
UC-TCH-28	View Exam Results				
UC-TCH-29	Correct Student Attempt				TC-24
UC-TCH-30	Monitor Exam				TC-18
UC-TCH-31	View Live Monitor Data				
UC-TCH-32	Monitor Student Audio Stream				TC-25
UC-TCH-33	View Student Camera Snapshots				TC-27
UC-TCH-34	Perform Student Monitor Action				TC-26
UC-TCH-35	Send Exam Notification				
UC-TCH-36	End Monitoring Session				
UC-TCH-37	Talk to Student During Exam				TC-07

REQ-ID	ANALYSIS (Function)	DESIGN	APP INTERFACE	CODE	TEST CASE ID
REQ-ID	ANALYSIS				TC-08

Table 3.9-RTM

9-SYSTEM ARCITECTURE:

System architecture serves as the backbone of any complex system, providing a conceptual model that defines the structure, behavior, and relationships between its various components.

It acts as a blueprint that guides the development, deployment, and maintenance of the system, ensuring that it meets its functional and non-functional requirements.

A well-defined architecture promotes clarity and communication among stakeholders, facilitates informed design decisions, and enables effective risk management.

Additionally, it fosters reusability and modularity, making it easier to maintain and evolve the system over time. By providing a comprehensive roadmap for development, system architecture plays a crucial role in building successful and sustainable software and hardware solutions.

9.1-MVT ARCHITECTURE:

Figure 3.10-MVC

9.2-INTRODUCTIONTOMVTANDITSIMPORTANCE:

The **Model-View-Template (MVT)** architecture is a core design pattern used in Django applications. It separates the application into three main components: **Model**, **View**, and **Template**,

allowing better organization and easier maintenance of complex systems like our **Examination System**.

Model Section:

The **Model** is responsible for handling the data and interacting directly with the database. In our Examination System, we created several models such as:

- Admin: handles admin login and permissions,
- Student: stores student details and exam records,
- Teacher: manages teacher accounts and subjects,
- Exam, Question, and Result: represent the core examination process.

Each model defines the structure of the data and includes logic for creating, retrieving, updating, and deleting records. For example, when an exam is created by a teacher, it is saved through the Exam model to the database.

View Section:

The **View** in Django acts as the main controller that processes user requests and determines the appropriate response. It handles the logic of the system.

In our Examination System, when an admin tries to log in, the view receives the login request, checks the credentials against the Admin model, and then decides what to show — whether to redirect to the dashboard or return an error message. Similarly, when a teacher wants to create a new exam, the view processes the form data, interacts with the models, and prepares data for the front-end.

Template Section:

The **Template** represents the front-end (UI) and is responsible for displaying the data to the user. It receives data from the view and renders it using HTML and Django Template Language (DTL).

For example, if a student wants to view available exams, the template will show a list of exams retrieved by the view. If a teacher checks exam results, the template will display performance reports passed from the view.

Summary:

In our system:

- The **Model** connects to the database and handles exam-related data,
- The **View** contains the logic for authentication, exam creation, performance analysis, and AI features,

- The **Template** displays forms, results, dashboards, and exam interfaces to admins, teachers, and students.

By using the MVT architecture, our Examination System achieves:

- A clean separation of concerns,
- Better scalability and maintenance,
- And easier integration of features like AI-powered question generation, performance analysis, and smart dashboards.

Chapter Four:

Design Study

1- Introduction:

In this chapter, we will introduce the design models, the database models, site map.

2- Class Diagram:

Figure 4.1-Class Diagram

3- Table Diagram:

Figure 4.2-Table Diagram

4-APP INTERFACES:

- **Login Page:**

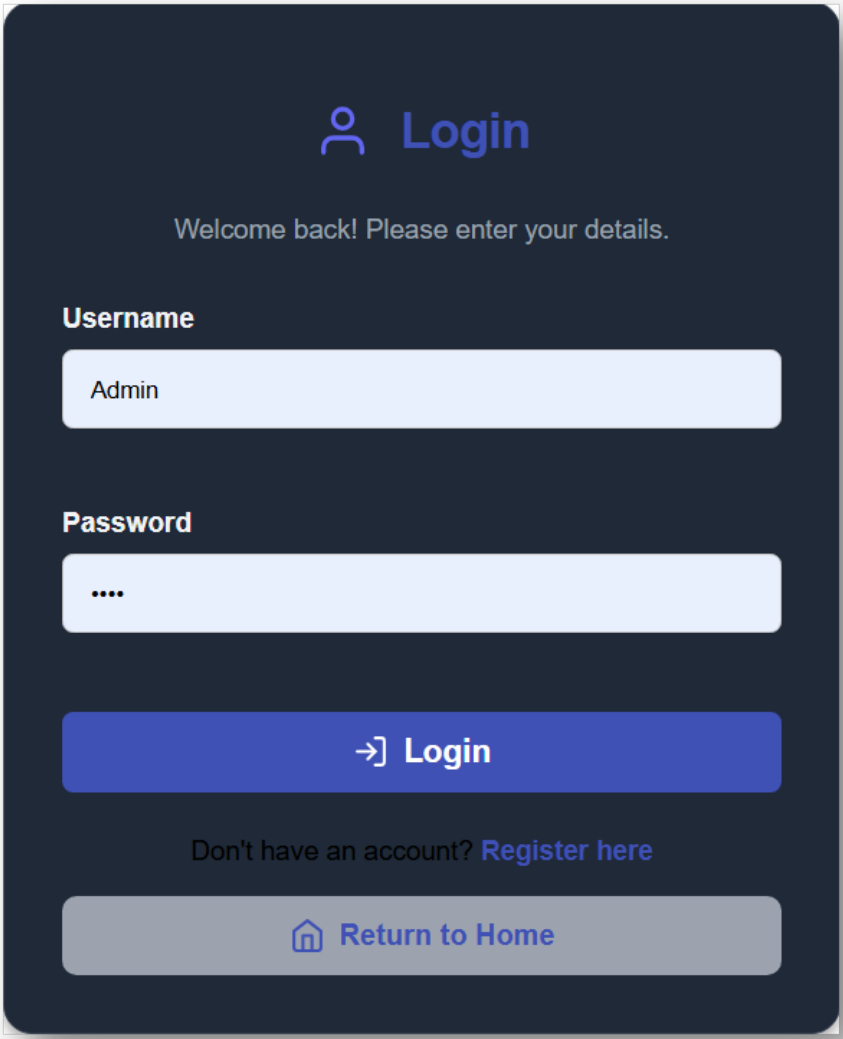


Figure 4.3-Login Page

- **Dashboard:**

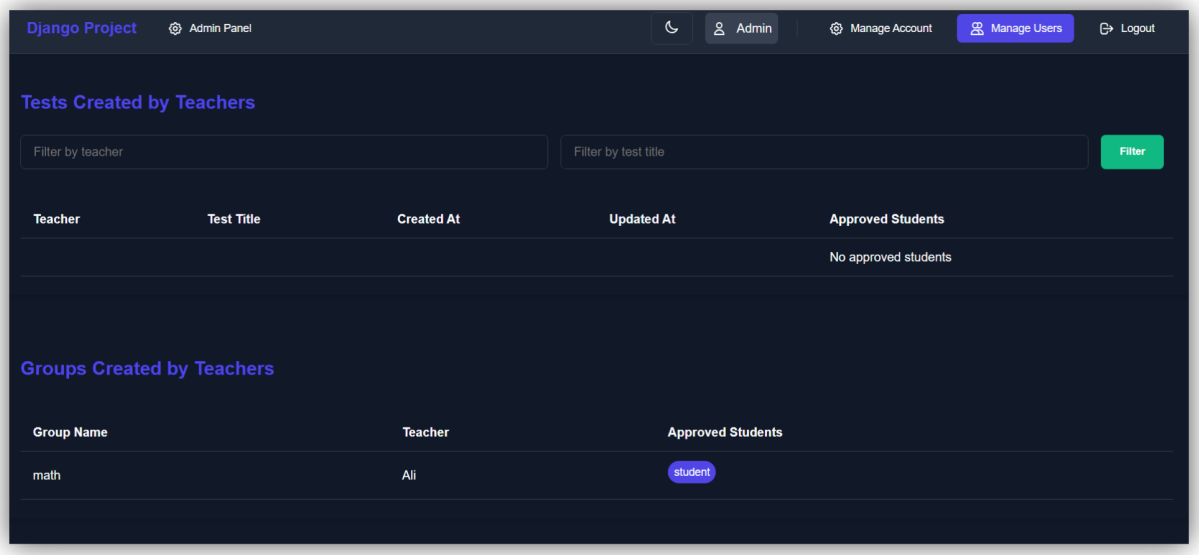


Figure 4.4-Dashboard Page ADMIN

- **Manage users:**

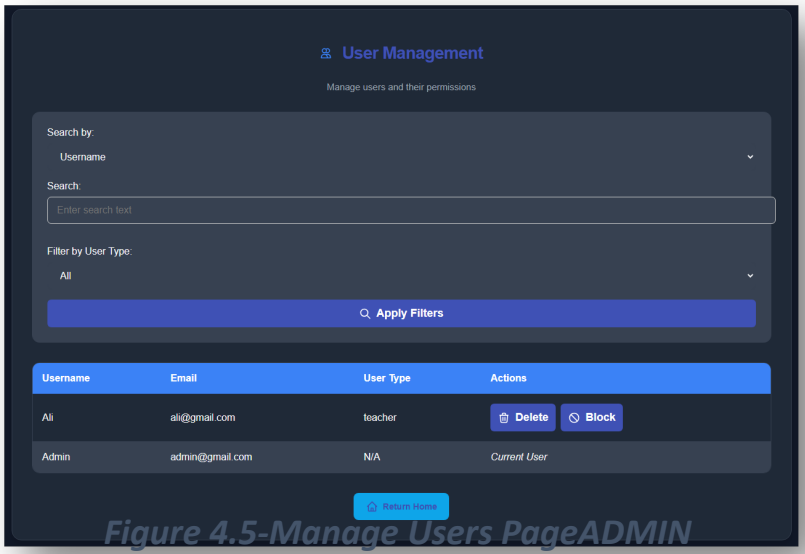


Figure 4.5-Manage Users PageADMIN

- **Dashboard:**

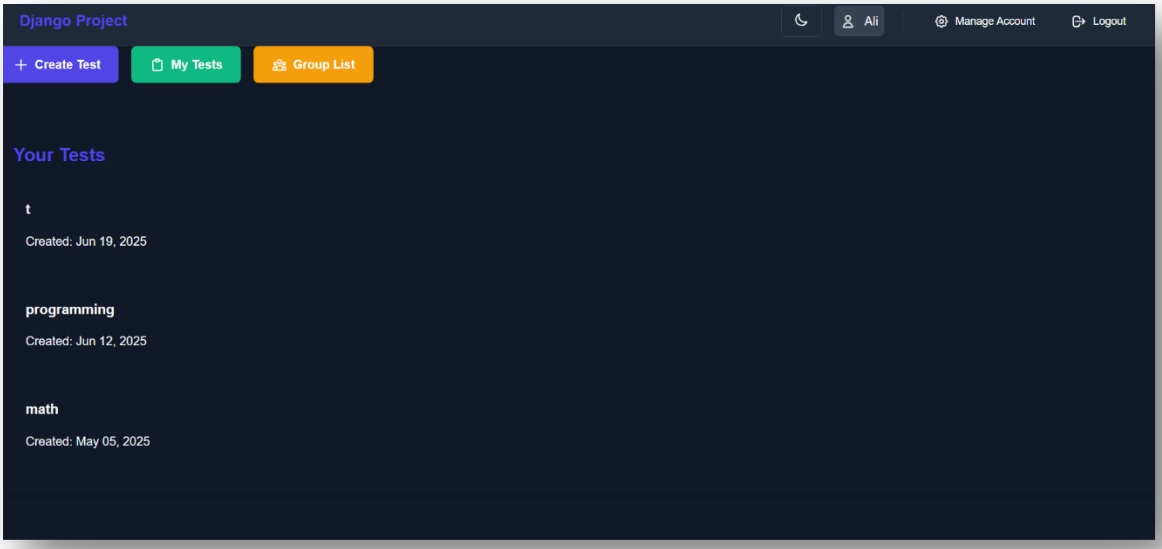


Figure 4.6-Dashboard PageTEACHER

- **Test:**



- **Chat:**

Figure 4.7-Test Page Teacher

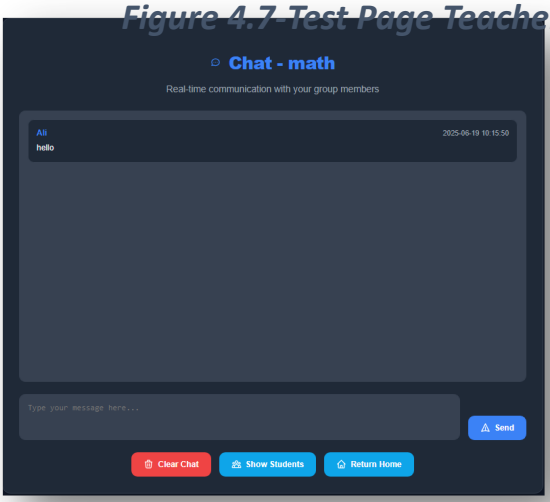
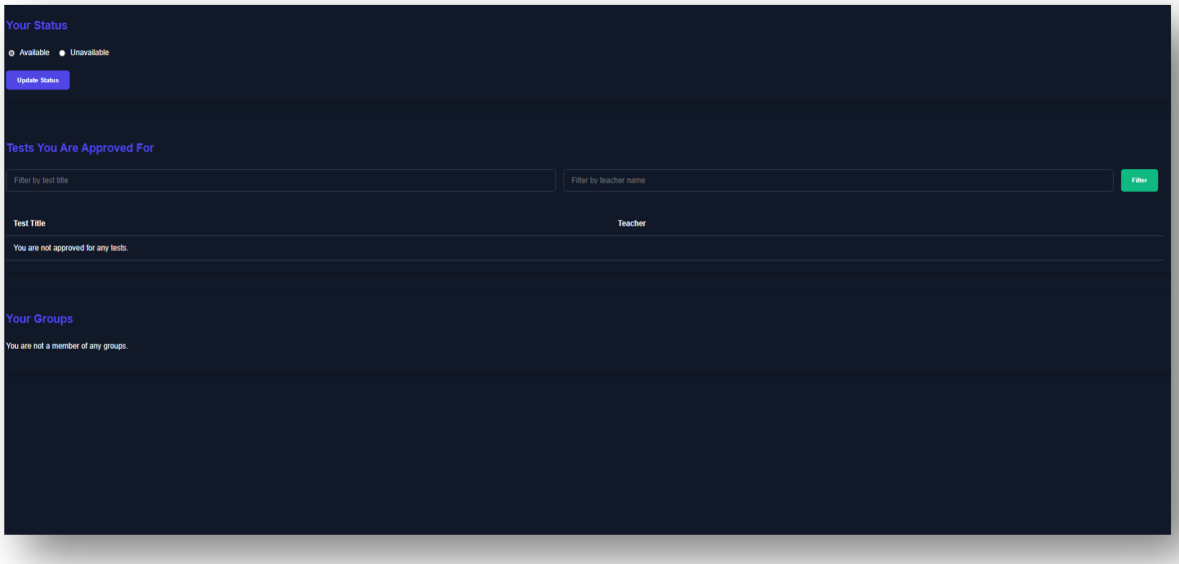
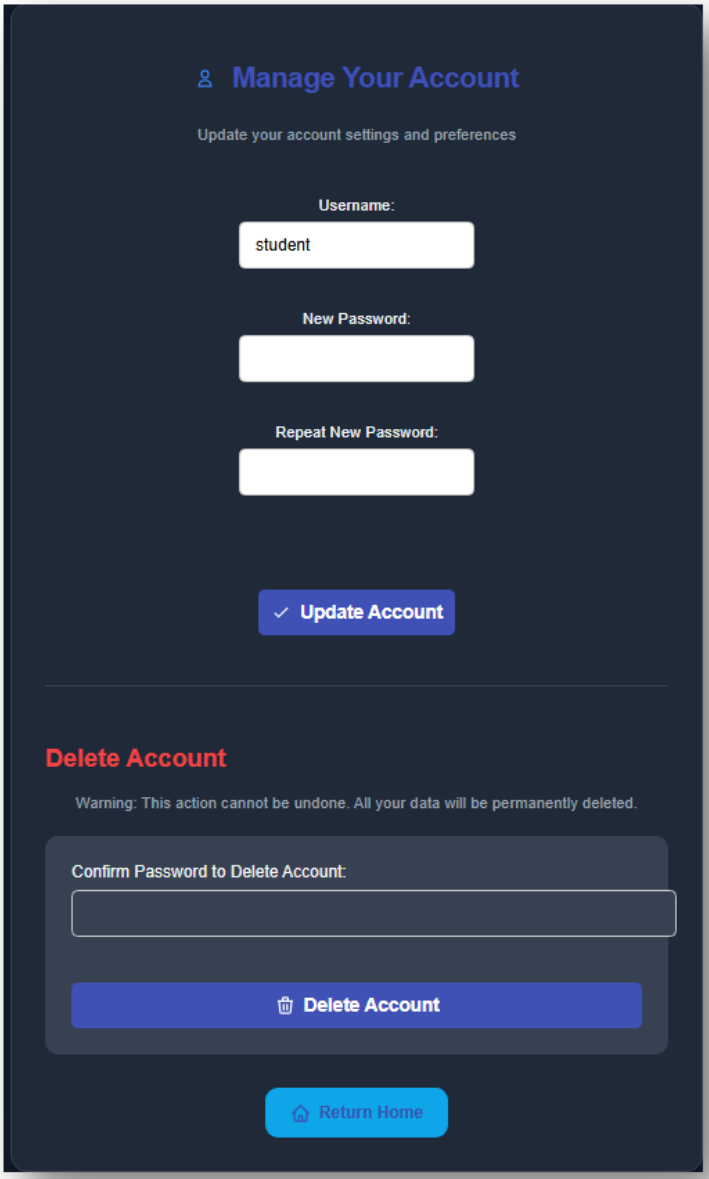


Figure 4.8-Chat Page Teacher

- **Dashboard:**

Figure 4.9-Dashboard Student





- **Update Account:**

Figure 4.10-Update AccountSTUDENT

4- Requirements Traceability Matrix (RTM):

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-01	View Users	—	—	—	
UC-ADM-02	Create User	—	—	—	TC-07
UC-ADM-03	Block User	—	—	—	TC-08
UC-ADM-04	Unblock User	—	—	—	
UC-ADM-05	Delete User	—	—	—	TC-09
UC-ADM-06	View Teachers	—	—	—	
UC-ADM-07	Create Teacher	—	—	—	
UC-ADM-08	View Analytics Dashboard	—	—	—	
UC-ADM-09	View Groups	—	—	—	

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-10	View Exams	—	—	—	
UC-ADM-11	View Exam Details	—	—	—	
UC-ADM-12	View Exam Results	—	—	—	
UC-ADM-13	Monitor Exam	—	—	—	
UC-ADM-14	View Reports Inbox	—	—	—	TC-22
UC-ADM-15	View Report	—	—	—	TC-22
UC-ADM-16	Send Report	—	—	—	TC-22
UC-ADM-17	Reply to Report	—	—	—	
UC-ADM-18	View System Settings	—	—	—	
UC-ADM-19	Update Email Settings	—	—	—	
UC-ADM-20	Update App Password	—	—	—	TC-06

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-21	Update Platform Branding	—	—	—	TC-10
UC-COM-01	Log In	—	—	—	TC-01, TC-02
UC-COM-02	Log Out	—	—	—	TC-03
UC-COM-03	Reset Password	—	—	—	
UC-COM-04	Update Profile	—	—	—	
UC-COM-05	Update Profile Information	—	—	—	
UC-COM-06	Change Password	—	—	—	TC-06
UC-PAIR-01	Verify Email Address	—	—	—	TC-04
UC-PAIR-02	Resend Verification Code	—	—	—	TC-05
UC-PAIR-03	View Messages Inbox	—	—	—	
UC-PAIR-04	Send Message	—	—	—	TC-22, TC-23

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-PAIR-05	Reply to Message	—	—	—	
UC-PAIR-06	View Group Messages	—	—	—	
UC-PAIR-07	Send Group Message	—	—	—	
UC-STD-01	View Teachers	—	—	—	
UC-STD-02	Send Join Request	—	—	—	
UC-STD-03	View Joined Teachers	—	—	—	
UC-STD-04	View Groups	—	—	—	
UC-STD-05	Send Group Join Request	—	—	—	
UC-STD-06	View Invitations	—	—	—	
UC-STD-07	Accept Invitation	—	—	—	
UC-STD-08	Reject Invitation	—	—	—	

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-STD-09	View Available Exams	—	—	—	
UC-STD-10	View Exam Events	—	—	—	
UC-STD-11	Take Exam	—	—	—	TC-19, TC-20
UC-STD-12	View Exam Result	—	—	—	TC-21
UC-STD-13	View Exam Events	—	—	—	TC-28
UC-STD-14	View Exam Notifications	—	—	—	
UC-STD-15	View Group Details	—	—	—	
UC-TCH-01	View Subjects	—	—	—	
UC-TCH-02	Create Subject	—	—	—	TC-11
UC-TCH-03	Edit Subject	—	—	—	
UC-TCH-04	Delete Subject	—	—	—	

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-TCH-05	View Subject Students	—	—	—	
UC-TCH-06	View Groups	—	—	—	
UC-TCH-07	Create Group	—	—	—	TC-12
UC-TCH-08	View Group Details	—	—	—	
UC-TCH-09	Edit Group	—	—	—	
UC-TCH-10	Delete Group	—	—	—	
UC-TCH-11	View Students	—	—	—	
UC-TCH-12	Invite Student	—	—	—	TC-13
UC-TCH-13	Accept Join Request	—	—	—	TC-14
UC-TCH-14	Reject Join Request	—	—	—	
UC-TCH-15	Remove Student	—	—	—	

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-TCH-16	Accept Group Join Request	—	—	—	
UC-TCH-17	Reject Group Join Request	—	—	—	
UC-TCH-18	View Question Bank	—	—	—	
UC-TCH-19	Create Question (Manually)	—	—	—	TC-16
UC-TCH-20	Edit Question	—	—	—	
UC-TCH-21	Delete Question	—	—	—	
UC-TCH-22	Generate Questions Using AI	—	—	—	
UC-TCH-23	View Exams	—	—	—	
UC-TCH-24	Create Exam	—	—	—	TC-15
UC-TCH-25	Edit Exam	—	—	—	
UC-TCH-26	Delete Exam	—	—	—	

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
UC-TCH-27	Manage Exam Questions	—	—	—	TC-17
UC-TCH-28	View Exam Results	—	—	—	
UC-TCH-29	Correct Student Attempt	—	—	—	
UC-TCH-30	Monitor Exam	—	—	—	TC-24
UC-TCH-31	View Live Monitor Data	—	—	—	TC-18
UC-TCH-32	Monitor Student Audio Stream	—	—	—	
UC-TCH-33	View Student Camera Snapshots				TC-25
UC-TCH-34	Perform Student Monitor Action				TC-27
UC-TCH-35	Send Exam Notification				TC-26
UC-TCH-36	End Monitoring Session				
UC-TCH-37	Talk to Student During Exam				

REQ-ID	ANALYSIS	DESIGN	APP INTERFACE	CODE	TEST CASE ID
REQ-ID	ANALYSIS				TC-07
UC-ADM-01	View Users				TC-08

Table 4.11-RTM Update

Chapter Five: Implementation & Testing

1- INRTODUCTION:

In this chapter we will introduce the used tools, some interfaces from the application, and the implementation of the test cases.

2- TOOLS:

- HTML:

Defines the structure and content

of all web pages in the system.

- CSS:

Styles the user interface to make it clean, responsive, and user-friendly.

- JAVA SCRIPT:

  Adds interactivity and
  dynamic behavior to
  forms, timers, and dashboards.

- PYTHON:

  The core backend language
  powering the application logic
  and data handling.

- DJANGO:

  A high-level Python framework
  that provides secure, scalable,
  and structured web development
  using the MVT pattern.

➢ SQLITE:

A lightweight, file-based database used for development and testing of application data.

3- TEST CASES EXCUTION:

Use Case ID	Test Case ID	Test Case Description	Input	Expected Results	Actual Results	Pass/Fail
UC-COM-01	TC-01	Login with valid Admin credentials and land on Admin Dashboard.	POST /accounts/login/ with username=admin1, password=Pass1234!	User authenticated; redirect to /admin-dashboard/; session created.	User authenticated; redirect to /admin-dashboard/; session created.	Pass
UC-COM-01	TC-02	Reject login with invalid password.	POST /accounts/login/ with username=admin1, password=WrongPass	Login page shows error; user not authenticated; no redirect.	Login page shows error; user not authenticated; no redirect.	Pass
UC-COM-02	TC-03	Logout ends the session and returns user to Login page.	POST /accounts/logout/ while logged in	Session cleared; redirect to /accounts/login/.	Session cleared; redirect to /accounts/login/.	Pass

UC-PAIR-01	TC-04	Verify email using correct OTP and activate account.	POST /accounts/verify-email/ with otp=<correct_code> (verification_user_id in session)	User.is_active=True; auto-login; verification session keys removed; redirect to role dashboard.	User.is_active=True; auto-login; verification session keys removed; redirect to role dashboard.	Pass
UC-PAIR-02	TC-05	Resend verification code regenerates OTP and sends email.	GET /accounts/resend-code/ (verification_user_id in session)	New OTP saved in session; success message; redirect to /accounts/verify-email/.	New OTP saved in session; success message; redirect to /accounts/verify-email/.	Pass
UC-COM-06	TC-06	Change password with correct current password updates password and keeps session.	POST /admin-dashboard/settings/ with password_submit=1, current_password=Pass1234!,new_password=NewPass1234!	Password updated; session remains valid; success message.	Password updated; session remains valid; success message.	Pass
UC-ADM-02	TC-07	Admin creates a new Student user account.	POST /admin-dashboard/users/create/ with username=stud2, full_name='Student Two', email=stud2@example.com, role=student, password1=Pass1234!, password2=Pass1234!	User created; success message; redirect to /admin-dashboard/users/.	User created; success message; redirect to /admin-dashboard/users/.	Pass
UC-ADM-03	TC-08	Admin blocks (deactivates) a user account.	POST /admin-dashboard/users/ with	User.is_active=False; success message;	User.is_active=False; success message;	Pass

			user_id=<stud2_id>, action=toggle_block	blocked user cannot login.	blocked user cannot login.	
UC-ADM-05	TC-09	Admin deletes a user account.	POST /admin-dashboard/users/ with user_id=<stud2_id>, action=delete	User deleted from DB; success message; user disappears from list.	User deleted from DB; success message; user disappears from list.	Pass
UC-ADM-21	TC-10	Admin updates platform branding name and icon in System Settings.	POST /admin-dashboard/settings/ with system_submit=1, platform_name='Smart Exams', system_icon=<png>	SystemSettings updated; success message; branding reflected in UI.	SystemSettings updated; success message; branding reflected in UI.	Pass
UC-TCH-02	TC-11	Teacher creates a Subject and assigns selected students (optional).	POST /teacher-dashboard/subjects/create/ with name='Math 101', description='Basics', selected_students='12,15'	Subject created with teacher=current; students added; redirect to Subjects Page.	Subject created with teacher=current; students added; redirect to Subjects Page.	Pass
UC-TCH-07	TC-12	Teacher creates a Group under a Subject with auto-generated unique code.	POST /teacher-dashboard/groups/create/ with name='Group A', subject=<subject_id>, selected_students='12,15'	Group created with unique code; students added; redirect to Groups Page.	Group created with unique code; students added; redirect to Groups Page.	Pass
UC-TCH-12	TC-13	Teacher invites a student by email (general invite).	POST /teacher-dashboard/students/ with	StudentJoinRequest created (pending);	StudentJoinRequest created (pending);	Pass

			action=invite_student, invite_email=stud1@example.com	success message.	success message.	
UC-TCH-13	TC-14	Teacher accepts a student's general join request.	POST /teacher-dashboard/students/ with action=accept_general, request_id=<jr_id>	Request status=accepted; student appears in teacher student list; success message.	Request status=accepted; student appears in teacher student list; success message.	Pass
UC-TCH-24	TC-15	Teacher creates an Exam with proctoring options and allowed students list.	POST /teacher-dashboard/exams/create/ with title='Midterm', subject=<subject_id>, total_mark=100, duration_minutes=60, late_join_minutes=10, start_time='2026-02-01T10:00', auto_proctoring=on, auto_fail_on_cheating=on, selected_students='12,15'	Exam created; allowed students saved; redirect to Exam Questions Page.	Exam created; allowed students saved; redirect to Exam Questions Page.	Pass
UC-TCH-19	TC-16	Teacher creates an MCQ question with choices and correct answer in Question Bank.	POST Question Bank with text='2+2?', question_type=mcq, mark=2, choices=['3','4'], correct_choice='4'	Question and choices created; correct choice flagged; success message.	Question and choices created; correct choice flagged; success message.	Pass
UC-TCH-27	TC-17	Teacher assigns selected	POST /teacher-dashboard/exa	ExamQuestion links created;	ExamQuestion links created;	Pass

		questions to an Exam and distributes marks equally when configured.	ms/<exam_id>/ questions/ with selected_questions=[q1,q2,q3, q4]	order set; marks distributed equally; success message.	order set; marks distributed equally; success message.	
UC-TCH-31	TC-18	Teacher reads live monitoring data for an Exam (JSON API).	GET /teacher-dashboard/exams/<exam_id>/ monitor/data/	JSON ok:true with stats, students list, remaining_seconds, activity_log.	JSON ok:true with stats, students list, remaining_seconds, activity_log.	Pass
UC-STD-11	TC-19	Student starts an allowed Exam within start window and gets an in-progress attempt created.	GET /student-dashboard/exams/<exam_id>/ take/ as allowed student	StudentExam created (in_progress); JOIN event logged; question page renders with timer.	StudentExam created (in_progress); JOIN event logged; question page renders with timer.	Pass
UC-STD-11	TC-20	Student answers MCQ and submits the Exam; score is computed and attempt is finished.	POST /student-dashboard/exams/<exam_id>/ take/?q=1 with choice=<correct_choice_id>, action=submit	StudentAnswer saved; StudentExam finished; score computed; SUBMIT event logged; redirect to exams list.	StudentAnswer saved; StudentExam finished; score computed; SUBMIT event logged; redirect to exams list.	Pass
UC-STD-12	TC-21	Student views Exam Result after finishing.	GET /student-dashboard/exams/<exam_id>/ result/	Result page shows score/percent, pass/fail, correct/wrong counts, time spent.	Result page shows score/percent, pass/fail, correct/wrong counts, time spent.	Pass
UC-PAIR-04	TC-22	Student sends a message to an associated Teacher.	POST Student Dashboard report form with teacher_id=<teacher_id>, title='Help', body='I have an	Message created; success message; appears in teacher/admin inbox where applicable.	Message created; success message; appears in teacher/admin inbox where applicable.	Pass

			issue'			
UC-PAIR-04	TC-23	Teacher sends a message to a linked Student.	POST /teacher-dashboard/reports/ with student_id=<student_id>, title='Notice', body='Exam starts soon'	Message created; success message; appears in student reports tab.	Message created; success message; appears in student reports tab.	Pass
UC-TCH-30	TC-24	Student initializes Proctoring session when auto_proctoring is enabled.	POST /proctor/<exam_id>/init/ as allowed student	JSON ok:true with session_id/audio_stream_id; ProctorSession active.	JSON ok:true with session_id/audio_stream_id; ProctorSession active.	Pass
UC-TCH-33	TC-25	Student uploads a camera snapshot to the active proctor session.	POST /proctor/<exam_id>/snapshot/ with {'image':'<...>'}	Snapshot stored/updated; snapshots_count increments; JSON ok:true.	Snapshot stored/updated; snapshots_count increments; JSON ok:true.	Pass
UC-TCH-35	TC-26	Teacher sends an exam notification to all allowed students.	POST /teacher-dashboard/exams/<exam_id>/monitor/notify/ with message='Keep camera on'	ExamNotification created for each allowed student; success message.	ExamNotification created for each allowed student; success message.	Pass
UC-TCH-34	TC-27	Teacher forces submission of a student's in-progress attempt (monitor action).	POST /teacher-dashboard/exams/<exam_id>/monitor/students/<student_id>/action/ with action=force_submit	Attempt finished; score computed; notification created; success message.	Attempt finished; score computed; notification created; success message.	Pass
UC-STD-13	TC-28	Student logs an exam event (camera denied) to be visible in monitoring	POST /student-dashboard/exams/<exam_id>/events/ with event_type='ca	ExamEvent created; appears in monitor activity_log.	ExamEvent created; appears in monitor activity_log.	Pass

		activity log.	mera_denied', message='User denied camera permission'			

Table 5.1-TEST CASES EXECUTION

4- UPDATE RTM:

REQ-ID	ANALYSIS	DESIGN (Actor Class)	APP INTERFACE	CODE	TEST CASE ID
UC-ADM-01	View Users	User	إدارة المستخدمين - منصة الاختبارات الذكية	User → admin_users()	
UC-ADM-02	Create User	User	إضافة مستخدم جديد - منصة الاختبارات الذكية	User → admin_user_create()	TC-07
UC-ADM-03	Block User	User	إدارة المستخدمين - منصة الاختبارات الذكية	User → admin_users()	TC-08
UC-ADM-04	Unblock User	User	إدارة المستخدمين - منصة الاختبارات الذكية	User → admin_users()	
UC-ADM-05	Delete User	User	إدارة المستخدمين - منصة الاختبارات الذكية	User → admin_users()	TC-09
UC-ADM-06	View Teachers	User	المدرسين - منصة الاختبارات الذكية	User → admin_teachers()	
UC-ADM-07	Create Teacher	User	إضافة مدرس - منصة الاختبارات الذكية	User → admin_teacher_create()	
UC-ADM-08	View Analytics Dashboard	Message	التحليلات والتقارير - منصة الاختبارات الذكية	Message → admin_analytics()	
UC-ADM-09	View Groups	Group	المجموعات - منصة الاختبارات الذكية	Group → admin_groups()	
UC-ADM-10	View Exams	Exam	الاختبارات - منصة الاختبارات الذكية	Exam → admin_exams()	
UC-ADM-11	View Exam	Exam	اختيار أسئلة الاختبار - منصة الاختبارات الذكية	Exam → admin_exam_details()	

	Details				
UC-ADM-12	View Exam Results	StudentExam	تفاصيل نتائج الاختبار - منصة الاختبارات الذكية	StudentExam → admin_exam_results()	
UC-ADM-13	Monitor Exam	StudentExam	مراقبة الاختبار - منصة الاختبارات الذكية	StudentExam → admin_exam_monitor()	
UC-ADM-14	View Reports Inbox	Message	التحليلات والتقارير - منصة الاختبارات الذكية	Message → admin_analytics()	TC-22
UC-ADM-15	View Report	Message	التحليلات والتقارير - منصة الاختبارات الذكية	Message → admin_analytics()	TC-22
UC-ADM-16	Send Report	Message	التحليلات والتقارير - منصة الاختبارات الذكية	Message → admin_analytics()	TC-22
UC-ADM-17	Reply to Report	Message	التحليلات والتقارير - منصة الاختبارات الذكية	Message → admin_analytics()	
UC-ADM-18	View System Settings	SystemSettings	الإعدادات العامة - منصة الاختبارات الذكية	SystemSettings → admin_settings()	
UC-ADM-19	Update Email Settings	SystemSettings	الإعدادات العامة - منصة الاختبارات الذكية	SystemSettings → admin_settings()	
UC-ADM-20	Update App Password	SystemSettings	الإعدادات العامة - منصة الاختبارات الذكية	SystemSettings → admin_settings()	TC-06
UC-ADM-21	Update Platform Branding	SystemSettings	الإعدادات العامة - منصة الاختبارات الذكية	SystemSettings → admin_settings()	TC-10
UC-COM-01	Log In	User	تسجيل الدخول - منصة الاختبارات الذكية	User → login_view()	TC-01, TC-02
UC-COM-02	Log Out	User	Logout	User → logout_view()	TC-03
UC-COM-03	Reset Password	User	استعادة كلمة المرور - منصة الاختبارات الذكية / تم إرسال رابط الاستعادة - منصة الاختبارات الذكية / تعيين كلمة مرور جديدة - منصة الاختبارات الذكية / تم تغيير كلمة المرور - منصة الاختبارات الذكية	User → PasswordResetView / PasswordResetConfirmView()	

UC-COM-04	Update Profile	User	الإعدادات العامة - منصة الاختبارات الذكية / الإعدادات العامة - لوحة المدرس / لوحة الطالب - منصة الاختبارات — الإعدادات العامة	User → admin_settings / teacher_settings / student_dashboard()	
UC-COM-05	Update Profile Information	User	الإعدادات العامة - منصة الاختبارات الذكية / الإعدادات العامة - لوحة المدرس / لوحة الطالب - منصة الاختبارات — الإعدادات العامة	User → admin_settings / teacher_settings / student_dashboard()	
UC-COM-06	Change Password	User	الإعدادات العامة - منصة الاختبارات الذكية / الإعدادات العامة - لوحة المدرس / لوحة الطالب - منصة الاختبارات — الإعدادات العامة	User → admin_settings / teacher_settings / student_dashboard()	TC-06
UC-PAIR-01	Verify Email Address	User	المصادقة الثنائية - منصة الاختبارات الذكية	User → verify_email()	TC-04
UC-PAIR-02	Resend Verification Code	User		User → resend_code()	TC-05
UC-PAIR-03	View Messages Inbox	Message	التحليلات والتقارير - منصة الاختبارات الذكية / التقارير - منصة الاختبارات الذكية / لوحة الطالب - منصة الاختبارات — التقارير	Message → admin_analytics / teacher_reports / student_dashboard()	
UC-PAIR-04	Send Message	Message	التحليلات والتقارير - منصة الاختبارات الذكية / التقارير - منصة الاختبارات الذكية / لوحة الطالب - منصة الاختبارات — التقارير	Message → admin_analytics / teacher_reports / student_dashboard()	TC-22, TC-23
UC-PAIR-05	Reply to Message	Message	التحليلات والتقارير - منصة الاختبارات الذكية / التقارير - منصة الاختبارات الذكية / لوحة الطالب - منصة الاختبارات — التقارير	Message → admin_analytics / teacher_reports / student_dashboard()	

UC-PAIR-06	View Group Messages	GroupMessage	GroupMessage → teacher_group_detail / student_group_detail()		
UC-PAIR-07	Send Group Message	GroupMessage	GroupMessage → teacher_group_detail / student_group_detail()		
UC-STD-01	View Teachers	User	لوحة الطالب - منصة الاختبارات — المدرسون	User → student_teachers()	
UC-STD-02	Send Join Request	StudentJoinRequest	لوحة الطالب - منصة الاختبارات — المدرسون	StudentJoinRequest → student_send_join_request()	
UC-STD-03	View Joined Teachers	User	لوحة الطالب - منصة الاختبارات — المدرسون	User → student_teachers()	
UC-STD-04	View Groups	Group	لوحة الطالب - منصة الاختبارات — مجموعاتي	Group → student_groups()	
UC-STD-05	Send Group Join Request	User		User → nan()	
UC-STD-06	View Invitations	StudentJoinRequest	لوحة الطالب - منصة الاختبارات — المدرسون	StudentJoinRequest → student_teachers()	
UC-STD-07	Accept Invitation	StudentJoinRequest	لوحة الطالب - منصة الاختبارات — المدرسون	StudentJoinRequest → student_invitation_action()	
UC-STD-08	Reject Invitation	StudentJoinRequest	لوحة الطالب - منصة الاختبارات — المدرسون	StudentJoinRequest → student_invitation_action()	
UC-STD-09	View Available Exams	Exam	لوحة الطالب - منصة الاختبارات — اختباراتي	Exam → student_exams()	
UC-STD-10	View Exam Events	User		User → nan()	
UC-STD-11	Take Exam	StudentExam	واجهة الاختبار - منصة الاختبارات الذكية	StudentExam → student_exam_take()	TC-19,

					TC-20
UC-STD-12	View Exam Result	StudentExam	نتائج الاختبار - منصة الاختبارات الذكية	StudentExam → student_exam_result()	TC-21
UC-STD-13	View Exam Events	ExamEvent	واجهة الاختبار - منصة الاختبارات الذكية	ExamEvent → student_exam_event()	TC-28
UC-STD-14	View Exam Notifications	ExamNotification	واجهة الاختبار - منصة الاختبارات الذكية	ExamNotification → student_exam_notifications()	
UC-STD-15	View Group Details	GroupMessage	دردشة المجموعة - منصة الاختبارات الذكية	GroupMessage → student_group_detail()	
UC-TCH-01	View Subjects	Subject	المواد الدراسية - منصة الاختبارات الذكية	Subject → teacher_subjects()	
UC-TCH-02	Create Subject	Subject	إضافة مادة جديدة - منصة الاختبارات الذكية	Subject → teacher_subject_create()	TC-11
UC-TCH-03	Edit Subject	Subject	إضافة مادة جديدة - منصة الاختبارات الذكية	Subject → teacher_subject_edit()	
UC-TCH-04	Delete Subject	Subject	المواد الدراسية - منصة الاختبارات الذكية	Subject → teacher_subject_delete()	
UC-TCH-05	View Subject Students	Subject	طلاب المادة - منصة الاختبارات الذكية	Subject → teacher_subject_students()	
UC-TCH-06	View Groups	Group	مجموعاتي - منصة الاختبارات الذكية	Group → teacher_groups()	
UC-TCH-07	Create Group	Group	إنشاء مجموعة جديدة - منصة الاختبارات الذكية	Group → teacher_group_create()	TC-12
UC-TCH-08	View Group Details	GroupMessage	محادثة المجموعة - منصة الاختبارات الذكية	GroupMessage → teacher_group_detail()	
UC-TCH-09	Edit Group	Group	إنشاء مجموعة جديدة - منصة الاختبارات الذكية	Group → teacher_group_edit()	
UC-TCH-10	Delete Group	Group	مجموعاتي - منصة الاختبارات الذكية	Group → teacher_group_delete()	
UC-TCH-11	View Students	StudentJoinRequest	إدارة الطلاب - منصة الاختبارات الذكية	StudentJoinRequest → teacher_students()	
UC-TCH-12	Invite Student	StudentJoinRequest	إدارة الطلاب - منصة الاختبارات الذكية	StudentJoinRequest → teacher_students()	TC-13
UC-TCH-13	Accept Join Request	StudentJoinRequest	إدارة الطلاب - منصة الاختبارات الذكية	StudentJoinRequest → teacher_students()	TC-14

UC-TCH-14	Reject Join Request	StudentJoinRequest	إدارة الطلاب - منصة الاختبارات الذكية	StudentJoinRequest → teacher_students()	
UC-TCH-15	Remove Student	User	إدارة الطلاب - منصة الاختبارات الذكية	User → teacher_students()	
UC-TCH-16	Accept Group Join Request	StudentJoinRequest	إدارة الطلاب - منصة الاختبارات الذكية	StudentJoinRequest → teacher_students()	
UC-TCH-17	Reject Group Join Request	StudentJoinRequest	إدارة الطلاب - منصة الاختبارات الذكية	StudentJoinRequest → teacher_students()	
UC-TCH-18	View Question Bank	Question	بنك الأسئلة - منصة الاختبارات الذكية	Question → teacher_subject_question_bank()	
UC-TCH-19	Create Question (Manually)	Question	بنك الأسئلة - منصة الاختبارات الذكية	Question → teacher_subject_question_bank()	TC-16
UC-TCH-20	Edit Question	Question	تعديل سؤال - منصة الاختبارات الذكية	Question → teacher_question_edit()	
UC-TCH-21	Delete Question	Question	بنك الأسئلة - منصة الاختبارات الذكية	Question → teacher_subject_question_bank()	
UC-TCH-22	Generate Questions Using AI	Question	بنك الأسئلة - منصة الاختبارات الذكية	Question → teacher_subject_question_bank()	
UC-TCH-23	View Exams	Exam	اختباراتي - منصة الاختبارات الذكية	Exam → teacher_exams()	
UC-TCH-24	Create Exam	Exam	إعداد اختبار جديد - منصة الاختبارات الذكية	Exam → teacher_exam_create()	TC-15
UC-TCH-25	Edit Exam	Exam	إعداد اختبار جديد - منصة الاختبارات الذكية	Exam → teacher_exam_edit()	
UC-TCH-26	Delete Exam	Exam	اختباراتي - منصة الاختبارات الذكية	Exam → teacher_exam_delete()	
UC-TCH-27	Manage Exam Questions	ExamQuestion	اختيار أسئلة الاختبار - منصة الاختبارات الذكية	ExamQuestion → teacher_exam_questions()	TC-17
UC-TCH-28	View Exam Results	StudentExam	نتائج الاختبار - منصة الاختبارات الذكية	StudentExam → teacher_exam_results()	

UC-TCH-29	Correct Student Attempt	StudentAnswer	تصحيح الاختبار - منصة الاختبارات الذكية	StudentAnswer → teacher_exam_student_correction()	
UC-TCH-30	Monitor Exam	StudentExam	مراقبة الاختبار - منصة الاختبارات الذكية	StudentExam → teacher_exam_monitor()	TC-24
UC-TCH-31	View Live Monitor Data	StudentExam	مراقبة الاختبار - منصة الاختبارات الذكية	StudentExam → teacher_exam_monitor_data()	TC-18
UC-TCH-32	Monitor Student Audio Stream	ProctorAudioStream	مراقبة الاختبار - منصة الاختبارات الذكية	ProctorAudioStream → proctor_teacher_signal()	
UC-TCH-33	View Student Camera Snapshots	ProctorSnapshot	مراقبة الاختبار - منصة الاختبارات الذكية	ProctorSnapshot → proctor_get_snapshots()	TC-25
UC-TCH-34	Perform Student Monitor Action	StudentExam	مراقبة الاختبار - منصة الاختبارات الذكية	StudentExam → teacher_exam_student_action()	TC-27
UC-TCH-35	Send Exam Notification	ExamNotification	مراقبة الاختبار - منصة الاختبارات الذكية	ExamNotification → teacher_exam_notify()	TC-26
UC-TCH-36	End Monitoring Session	ProctorSession	مراقبة الاختبار - منصة الاختبارات الذكية	ProctorSession → teacher_exam_end()	
UC-TCH-37	Talk to Student During Exam	ProctorAudioStream	مراقبة الاختبار - منصة الاختبارات الذكية	ProctorAudioStream → proctor_teacher_signal()	

Table 5.2-Update RTM

REFERENCES:

1- “Exam” website. https://exam.net/

2- “FlexiQuiz” website.https://www.flexiquiz.com/

3- “Mostakl” website.https://www.testinvite.com/

4- "Python"website.https://www.python.org/

5- "Visual-paradigm.https://www.visual-paradigm.com/

6- Ian Sommerville – Software Engineering

7- Martin Fowler – UML Distilled