

A CSCI313 Project

**By:**

**AbdelRahman Rahal**

**Mahmoud Abdgelil**

**Zaynab Abdullah**

**Alhussein Nagdy**

# Table of Contents

[Table of Contents 2](#_bookmark0)

1. [Introduction 4](#_bookmark1)
   1. [Purpose 4](#_bookmark2)
   2. [Project Scope 4](#_bookmark3)
   3. [Used Technologies 4](#_bookmark4)
   4. [Intended Audience 4](#_bookmark5)
   5. [Reference 5](#_bookmark6)
2. [Overall Description 6](#_bookmark7)
   1. [Product Perspective 6](#_bookmark8)
      1. [Product Function 6](#_bookmark9)
   2. [User Characteristics 7](#_bookmark10)
   3. [Constraints 7](#_bookmark11)
   4. [Assumptions and Dependencies 7](#_bookmark12)
3. [Interface 8](#_bookmark13)
   1. [System Interface 8](#_bookmark14)
   2. [Software Interface 12](#_bookmark15)
   3. [Hardware Interface 12](#_bookmark16)
4. [Functional Requirements 13](#_bookmark17)
   1. [User Class 1: The User 13](#_bookmark18)
   2. [User Class 2: The Shelter 14](#_bookmark19)
5. [Non-Functional Requirements 16](#_bookmark20)
   1. [Reliability 16](#_bookmark21)
   2. [Recoverability 16](#_bookmark22)
   3. [Performance 16](#_bookmark23)
   4. [Availability 16](#_bookmark24)
   5. [Usability 16](#_bookmark25)
   6. [Maintainability 16](#_bookmark26)
6. [Diagrams 17](#_bookmark27)
   1. [Use case Diagram 17](#_bookmark28)
      1. [Use case scenarios 18](#_bookmark29)
   2. [Sequence Diagram 28](#_bookmark30)
      1. [Report a Stray 28](#_bookmark31)
      2. [Create Pet Profile 29](#_bookmark32)
      3. [Offer Animal for Adoption 30](#_bookmark33)
      4. [Donate to Shelter 31](#_bookmark34)
   3. [Class Diagram 32](#_bookmark35)
7. [Test Cases 33](#_bookmark36)
   1. [Test Case 1 33](#_bookmark37)
   2. [Test Case 2 34](#_bookmark38)
   3. [Test Case 3 35](#_bookmark39)
   4. [Test Case 4 36](#_bookmark40)
8. [Links 36](#_bookmark41)
   1. [Scrum Meeting Link 37](#_bookmark42)
   2. [Implementation Apk Link 37](#_bookmark43)

# Introduction

## Purpose

The **Grade A** e-learning platform is a web-based system designed to enhance online education by delivering interactive, accessible, and engaging educational content. The system is built to address the needs of students, instructors, and administrators, ensuring a seamless learning experience with integrated tools for communication, progress tracking, and course management.

## Project Scope

## The "Grade A" e-learning platform is a web-based system designed to enhance online education by addressing challenges such as accessibility, interactivity, and engagement. It enables instructors to manage courses, students to track progress and access materials, and administrators to oversee operations seamlessly. Built with React for the frontend, Supabase serves as the backend, providing rapid prototyping and real-time functionality with PostgreSQL for scalable data storage. The platform prioritizes usability, performance, and security to deliver an inclusive, interactive, and efficient learning experience.

## Used Technologies

* **React**: Utilized for developing a responsive and dynamic frontend interface.
* **Supabase**: Serves as the backend server, providing authentication and real-time capabilities.
* **PostgreSQL**: Employed as the database management system via Supabase for secure and scalable data storage.
* **Git/GitHub**: Used for version control, facilitating smooth collaboration and code management.
* **Python with Pytest**: Integrated for testing to ensure code reliability and quality assurance

## 1.4 Intended Audience

• **Client**: To communicate the overall project goals, progress, and necessary adjustments at each development stage, ensuring the platform aligns with educational needs and expectations.

• **Developer**: To serve as a guide for implementing the required features, functionalities, and tasks, including course management, authentication, and progress tracking.

• **Software Tester**: To outline all the test cases that are necessary for validating the system's performance, functionality, and security to ensure that it meets the client’s and users' requirements.

• **Project Manager/SCRUM Master**: To define and organize the project’s timeline, goals, and tasks, ensuring the team works harmoniously while addressing any potential issues. This role will also ensure that all members stay focused on the overall project objectives and user personas.

• **End Users (Students, Instructors, Admins)**:

* + **Students**: To ensure a seamless, user-friendly platform for enrollment, course engagement, and progress tracking.
  + **Instructors**: To provide a functional and efficient interface for creating, managing, and interacting with courses and students.
  + **Admins**: To guarantee smooth operations of the platform, including user and content management, security, and performance monitoring.

.

## .Reference

IEEE. IEEE Std 830-1999 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1999.

# Overall Description

**2.1. Product Perspective**• "Grade A" is an e-learning platform that delivers engaging, interactive, and accessible educational content for students, instructors, and administrators.  
• The system offers distinct functionalities for different types of users: students, instructors, and admins. Each user group has access to specific features tailored to their needs.

**2.1.1. Product Function  
Student User:**

• After accessing the platform, the student can sign up by providing their name, email, password, and other required details.

• Once signed in, students can browse available courses, enroll in them, and access course materials. • Students can track their progress, view grades, and receive feedback on assignments.

• Students can engage in discussions, submit assignments, and participate in quizzes.

**Instructor User:**

* After signing in, instructors can create new courses by filling in necessary course details (e.g., title, description, materials).
* Instructors can upload a variety of content including PDFs, videos, and quizzes.
* They can monitor student progress through analytics and track completion rates for assignments.
* Instructors can send notifications to all enrolled students.
* The observer design pattern will be used to ensure that when an instructor sends a notification, all students enrolled in the course will be immediately notified of the update in real-time.

**Admin user:**

• Admins can manage user accounts (students and instructors), approve or deactivate courses, and monitor platform activity.

• Admins ensure the smooth operation of the system, overseeing data security and user support.

**2.2. User Characteristics**• The user should have basic knowledge of how to use web-based platforms. • The user should have reliable internet access to interact with the platform.

**2.3. Constraints  
•** Users must have internet connectivity to use the platform's features.

• A sign-in and password authentication process is required to identify users (students, instructors).

• The system should be accessible on desktops, tablets, and smartphones with a responsive design.

**2.4. Assumptions and Dependencies**• The platform is assumed to be available via any modern web browser (e.g., Chrome, Firefox, Edge).  
• The platform’s backend will be powered by Supabase, acting as the backend server, to handle functionalities such as authentication, data storage, and real-time updates. React will handle the frontend functionalities, providing an interactive and responsive user experience.  
• The platform will require users to have a functional email address for account creation.

# Interface

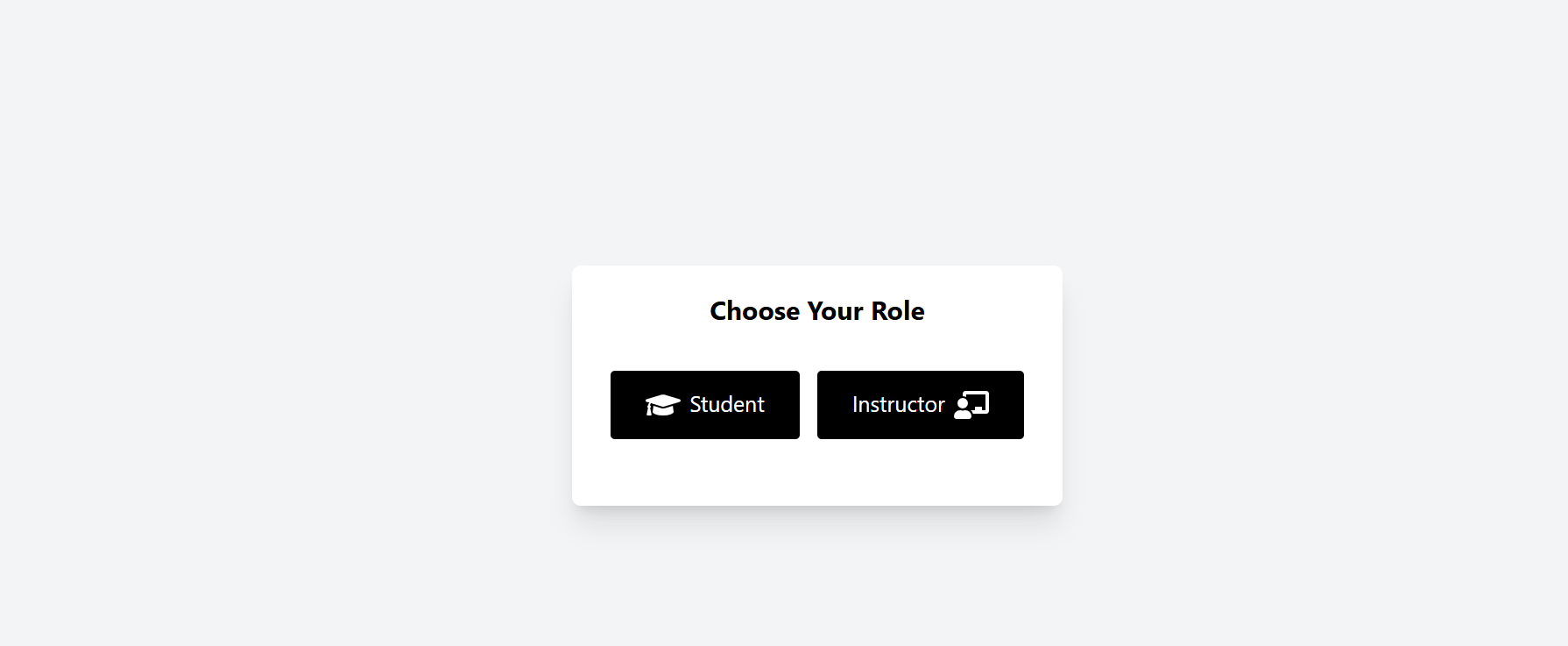
## System Interface

When the user opens the platform for the first time, they will be directed to a **sign-up page** where they will have two options:

1. Sign up as a student
2. Sign up as an instructor  
   In addition to these, a **sign-in option** is available for users who already have registered accounts.

**Sign-up as a Student or Instructor:**

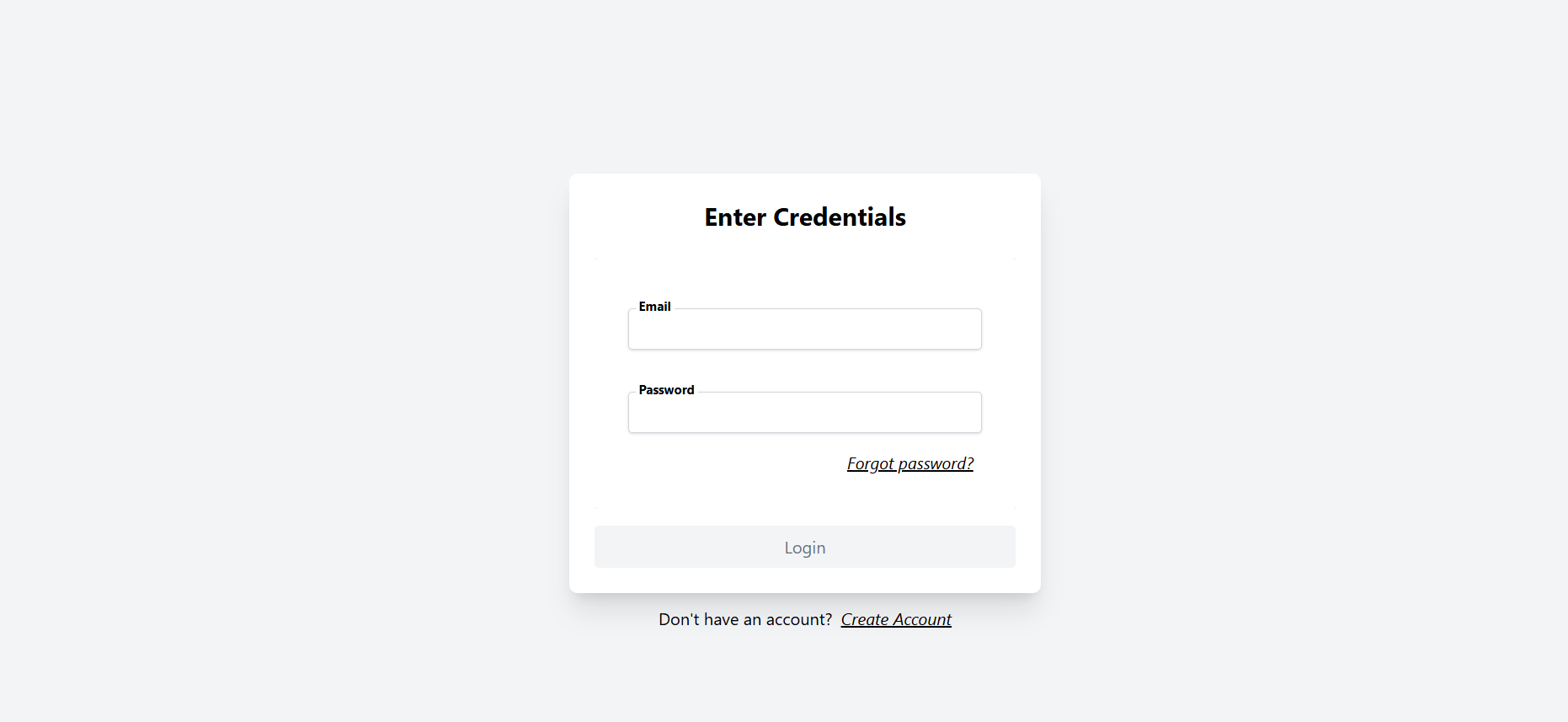
* When the user selects to sign up as a student or instructor, they will be prompted to provide their name, email address, password, and other necessary details such as location or role-specific information.
* After completing the registration form, users can submit the information and proceed to their designated dashboard upon successful registration.

A screenshot of a computer screen

Description automatically generated

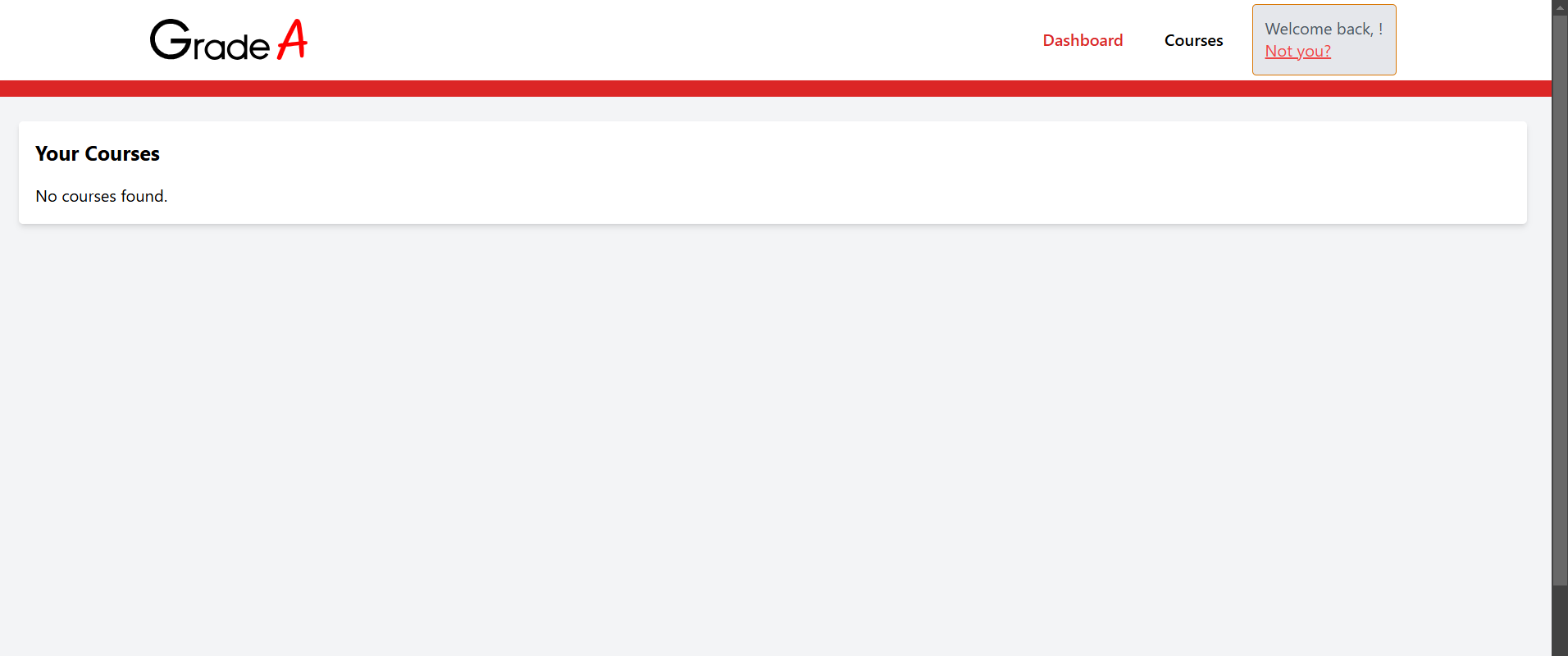
**Login Page:**

* For existing users, a login page will appear where they can enter their credentials (email and password).
* Upon successful login, users will be redirected to their **dashboard**, which will provide personalized access to courses, progress tracking, and communication tools.



**Dashboard:**

* The dashboard will be tailored to the user role:
  + **For Students:** Students will be able to view available courses, track progress, participate in discussions, and submit assignments.
  + **For Instructors:** Instructors can manage courses, upload materials, send notifications to students, and track the performance of enrolled students.



## Software Interface

## The "Grade A" e-learning platform is a web application built with React for the frontend and Supabase as the backend. React manages the user interface and core functionalities, such as course browsing, enrollment, assignment submission, and progress tracking. Supabase handles backend tasks like data storage, real-time updates, authentication, and API generation, using its PostgreSQL database for secure and scalable management.

## The development environment uses Visual Studio Code, integrating necessary libraries for a smooth user experience. Data flows between the frontend and backend in two directions: user actions (e.g., enrollment, assignments) are sent to Supabase for processing, while data such as course content and grades is retrieved from Supabase and displayed to users.

## Hardware Interface

* The "Grade A" e-learning platform requires an internet connection to function. Users will access the platform via standard devices, including desktop computers, laptops, tablets, and smartphones. The platform is designed to be fully responsive, ensuring a seamless experience across different screen sizes and devices.
* To fully utilize the platform’s functionalities, devices must have internet access for real-time updates, content retrieval, and submission of assignments. Additionally, users should have the capability to interact with multimedia content such as PDFs, videos, and quizzes. Devices do not require any special hardware beyond standard internet-enabled capabilities to access and use the platform effectively.

# Functional Requirements

## User Class 1: The User

* + 1. **Functional Requirements Title:** User registration

**Description:** System will allow the user to create an account so that they can login later.

### Required Information:

* + - * First Name
      * Last Name
      * Password (at least 8 characters)
      * Confirm password (at least 8 characters)
      * Email (unregistered email)

### Functional Requirements Title: Login

**Description:** System will allow the user to login if they entered a valid username and password.

### Functional Requirements

**Title:** Enroll in courses

**Description:**

**Required Information:**

### Student’s Details: Name, email, student ID, etc.

### Course Information: Course name, description, prerequisites, schedule, capacity.

### Student’s Enrollment Status: Current enrolled courses, progress, available credits.

### Functional Requirements

**Title:** Access course materials

**Description:**

The system will allow the user to access course materials, download content, and view assignments by providing all of the required information.

**Required Information:**

* Course Name  
  The name of the course the user is enrolled in or is trying to access.
* Course ID  
  The unique identifier for the course, which helps in finding the specific materials related to that course.
* Module/Section Name  
  The name of the module or section the user is accessing.
* Content Type  
  Specifies the type of material (e.g., video, article, document, assignment).
* File Format  
  The format in which the material is available (e.g., PDF, MP4, DOCX).
* Download Link  
  A link to download or access the material. This is used for users who wish to download the content for offline access.
* Download Permission  
  Ensures that the user has the appropriate permissions to download the content.

Extensions:

* View Assignments (extend relationship)  
  The user can click on the provided assignment link to view details, instructions, and submission requirements for the assignment.
* Download Content (include relationship)  
  The user can click on the download link to download the content (if the user has permission to do so).
  + 1. **Functional Requirements Title:** Track progress

**Description:** The system will allow the user to track their progress in a course by viewing their grades and comparing their performance with the class average.

**Required Information:**

* Course Name  
  The name of the course the user is enrolled in.
* Course ID  
  A unique identifier for the course, which helps in retrieving the correct grading information.
* Student ID  
  The unique identifier for the student, allowing the system to retrieve their individual grades.
* Assignment Grades  
  The grades for assignments, quizzes, or exams that the student has completed.
* Class Average for Assignment/Exam  
  The average grade for the class on a particular assignment, quiz, or exam, which allows for comparison with the student’s performance.
* Grade Thresholds  
  Minimum or maximum grade levels required for specific achievements, such as passing the course or qualifying for honors.

**Extensions:**

* View Grades (include relationship)  
  The user can view their individual grades for assignments, quizzes, and exams.
* Compare with Class Average (extend relationship)  
  The user can compare their grades with the class average for each assignment or exam to see how they performed relative to their peers. The comparison can be shown as a graphical representation or a numerical value, highlighting whether the student is above or below the average.

## User Class 2: instructor

* + 1. **Functional Requirements Title: register**

### Description:

### The system will allow an instructor to register for the platform by providing the necessary information.

### This registration enables the instructor to create and manage courses, access student progress, and contribute to the academic content of the platform

### Required Information:

### Instructor Name Full name of the instructor, used to identify them within the platform.

### Email Address The instructor's email address, which is required for communication, account verification, and notifications.

### Department The academic department to which the instructor belongs, helping categorize courses and assignments.

### Password A password chosen by the instructor to secure their account and ensure proper authentication.

### Confirm Password A field to confirm that the password entered matches the intended password

**Extensions:**

* **Validate Email (**include relationship)  
  The instructor will receive a confirmation email to verify their email address during the registration process. Clicking the verification link activates their account.
* **Forget Password** (extend relationship)  
  If the instructor forgets their password, they can use the “Forgot Password” option. The system will prompt them to enter their registered email address. A password reset link will be sent to the email address. Upon clicking the link, the instructor will be able to create a new password and regain access to their account.
  + 1. **Functional Requirements Title:** Upload Materials

### ****Description:****

### The system will allow instructors to upload educational materials (such as lecture notes, slides, videos, and other resources) for their courses.

### These materials will be made available to students to enhance learning. In addition, the system will allow instructors to organize and manage the uploaded materials efficiently.

### Required Information:

### Course Name The name of the course for which the material is being uploaded. This will help categorize and associate the material with the correct course.

### Material Title A descriptive title for the uploaded material to help students understand the content.

### Material Type The type of material being uploaded (e.g., PDF, Word document, PowerPoint slides, video, etc.). This helps the system organize different formats appropriately.

### Upload File The actual file to be uploaded (e.g., a document, video file, etc.). This will be stored in the system for access by students.

### Description A short description of the material content, to provide students with an overview of what the material covers.

### Date of Upload The date the material is uploaded. This will help in sorting and managing materials based on the time they were made available.

### Extensions:

### Organize Material (include relationship) The instructor can organize the uploaded materials by categories, units, or topics. They can also create folders to group related materials together, making it easier for students to find resources based on the syllabus or learning modules.

### Add Interactive Content (extend relationship) In addition to static materials, the instructor can also add interactive content like quizzes, polls, or links to external resources (e.g., online tutorials or websites). This feature allows instructors to engage students more actively and enhance their learning experience.

### Functional Requirements

**Title:** provide feedback

**Description:**

* The "Provide Feedback" feature is designed to enable instructors to give personalized feedback and notifications to students regarding their performance, such as grades, comments, or any updates on assignments. The feature is an essential part of the educational process, as it helps students understand how they are performing and what areas they need to improve.
* To make this system effective and scalable, we will implement the Observer Design Pattern. In this pattern, the instructor is the subject (or publisher), who triggers feedback notifications, while the students are the observers (or subscribers) who receive the notifications. When the instructor updates a student's grade or provides feedback on an assignment, all subscribed students will automatically receive a notification about the change.
* The Observer Design Pattern ensures that when a change occurs, such as when the instructor assigns grades or provides feedback, all subscribed students will be notified in real-time without needing to directly communicate with each one individually. This approach allows for scalability and flexibility in handling notifications and ensures that students stay informed of their academic progress.

**Required Information:**

**Instructor Role:**

* Provide Feedback on Assignments: Instructors should be able to provide feedback to students on their assignments, including text comments, grades, or suggestions for improvement.
* Grade Assignments: Instructors should be able to assign grades to students for each assignment. The grade could be a numerical value or a letter grade (e.g. 10/10 or A).
* Send Feedback Notifications: When feedback or grades are provided on an assignment or task, the instructor must have the ability to notify the students about the new feedback and grade through the system.
* Edit Feedback: The instructor should be able to update or modify the feedback at any time. If feedback or grades are updated, the system should send another notification to the students.

Student Role:

* View Feedback Notifications: Students should be able to view notifications for any new feedback or grades provided by the instructor on their assignments.
* View Feedback Details: Students should be able to access detailed feedback, including the grade received, comments from the instructor, and suggestions for improvement.

**Notification System:**

* Real-Time Notifications: The system should send real-time notifications to students when the instructor provides feedback or grades on an assignment or course task.
* Track Feedback History: All feedback and grade notifications and updates should be logged, and students should have access to their historical feedback and grades.
* Type of Feedback Notifications: The system should differentiate between different types of notifications, such as new feedback, new grades, updated feedback, or updated grades.

**Feedback Visibility:**

* Access Control: Feedback notifications should only be accessible to the relevant student and instructor. Privacy should be ensured, so no other student can access someone else’s feedback or grades.

### ****Extensions:****

• **Grade Assignments (include relationship)**  
The instructor can grade assignments, providing students with a clear evaluation of their work. The grading can include numerical values, letter grades, or a combination of both. The system ensures that grades are visible only to the respective student and instructor, maintaining privacy and confidentiality.

• **Send Notifications (extend relationship)**  
Once feedback and grades are provided for an assignment, the instructor can send real-time notifications to students. These notifications inform the student of new feedback, grades, or updates on their submitted assignments. This feature ensures that students are promptly notified and can access their feedback and grades without delay, helping them stay informed about their progress.

### 3.2.4 Functional Requirements

### Title: Manage Courses

**Description:**

* The Manage Courses feature allows instructors to oversee the creation, editing, and deletion of courses within the system.
* It provides an interface for instructors to efficiently handle all aspects of course management, ensuring that students have access to updated and relevant course content.
* Through this feature, instructors can create new courses, adjust existing course details, or remove courses that are no longer needed. This functionality is crucial for maintaining the structure and integrity of the course offerings in the system. The system ensures that courses are properly categorized and that students can easily access the courses they are enrolled in.

### Required Information:

### Create Courses Instructors can create new courses by entering relevant information such as the course name, description, syllabus, schedule, and instructor details. This feature allows for easy course setup and ensures that students can find and enroll in courses that align with their academic goals. It may include the option to set prerequisites, course objectives, and other key details.

### Edit Course (extend relationship) Instructors have the ability to edit courses they have created. They can modify details such as course content, schedule, prerequisites, or instructor assignments as needed. The edit functionality allows the course structure to remain flexible and adapt to changes, like updates in the syllabus or changes in course material.

### Delete Course (extend relationship) Instructors can also delete courses from the system. This is particularly useful for removing outdated courses or those that are no longer in use. When a course is deleted, all associated materials, assignments, and student enrollments are also removed, ensuring that the system remains clean and up-to-date.

### User class 3: Admin

### Title: System Management

### Description:

### The System Management feature from the admin's view provides the necessary tools for the administrative team to manage and monitor the entire learning platform. It includes capabilities to manage both users and courses, ensuring smooth operation and maintenance of the system. Admins have the authority to oversee the creation, modification, and removal of users and courses, providing centralized control over the platform.

### Admins can manage the platform's user base, including students, instructors, and other administrative roles. They can add new users, modify existing user information, or delete users who are no longer part of the system. Similarly, admins can manage the course catalog by creating, editing, or deleting courses, ensuring that the course offers align with the institution's academic objectives.

### Requirement information:

### Manage Users

### The "Manage Users" functionality allows admins to handle all user accounts in the system. This includes:

### Create Users Admins can create new user accounts for instructors, students, or administrative roles. Each account is associated with a specific role, and the admin can assign various permissions based on the user's responsibilities in the system. This feature ensures that only authorized users have access to appropriate resources.

### Edit User Details (Association) Admins can modify the details of any user, such as updating personal information, changing roles, or altering permissions. This ensures the platform remains flexible and can adapt to organizational changes, such as a user changing their role or responsibilities within the system.

### Delete Users (Association) Admins can remove users from the system if they are no longer needed or if they have left the institution. Deleting a user removes all associated data, ensuring the system remains clean and up-to-date.

# Non-Functional Requirements

**1. Performance**

* Concurrent Users:  
  The platform must be capable of supporting up to 100 concurrent users without significant performance degradation. This ensures that the system can handle peak usage times, such as during class registration or course material uploads, without slowing down.
* Response Time:  
  The average response time of the platform should be under 2 seconds during normal load conditions. This ensures a fast and smooth user experience, where interactions (such as submitting forms or loading pages) are quick and efficient, contributing to overall user satisfaction.

**2. Security**

* Authentication and Encryption:  
  Authentication protocols and data storage processes will adhere to industry-standard encryption methods, ensuring that sensitive user data (such as passwords, grades, and personal information) is securely stored and protected. This helps prevent unauthorized access and data breaches.
* HTTPS Enforcement:  
  HTTPS will be enforced for all communications between the client (user) and server. This ensures that all data transmitted across the network is encrypted, safeguarding it from potential interception or tampering during transmission.

**3. Usability**

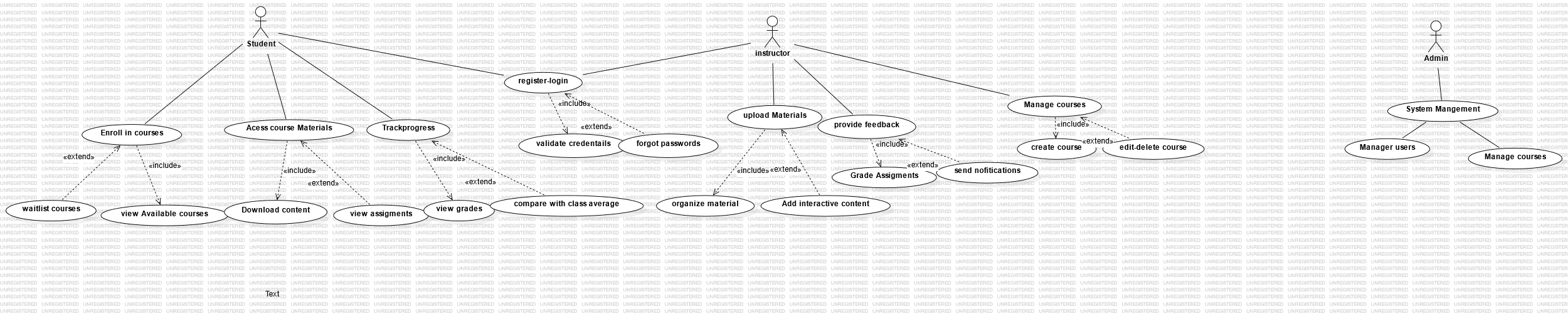
* Responsive Design:  
  The platform will feature a responsive and user-friendly interface, designed to work seamlessly across a variety of devices, including desktops, tablets, and smartphones. This ensures that users can access the platform from any device, whether they are at home or on the go, with a consistent and optimized experience.
* Clear Navigation:  
  Navigation within the platform will be intuitive, with clearly labeled menus, dashboards, and tooltips that guide users through the system. This allows users to easily find and use the platform’s features without confusion, reducing the learning curve for new users.

**4. Maintainability**

* Modular Code Design:  
  The system will adopt a modular code design, which allows for isolated and efficient updates, bug fixes, and additions. This approach ensures that the system remains flexible, easily extendable, and easier to maintain over time.
* Comprehensive Documentation:  
  Comprehensive documentation will be provided for all code components and system features. This documentation will serve as a reference for developers, ensuring a smooth onboarding process for new developers and easing future updates or maintenance.
* Extensibility:  
  The platform’s design will be flexible enough to allow for the addition of new functionalities and features in the future. As user needs evolve or new technologies emerge, the system will be able to incorporate these changes without requiring a complete overhaul, providing long-term value and scalability.

# Diagrams

## Use case Diagram

****

## Use case scenarios

Use Case 1

|  |  |
| --- | --- |
| register | |
| **Use case Name** | Register |
| **Actors** | Student/Instructor (Non-registered) |
| **Main success scenario** | 1- User **Accesses Registration Page:**   * The user (either a student or an instructor) navigates to the registration page of the application.   2-**Select Role (Student/Instructor):**   * On the registration page, the user is prompted to select their role (either "Student" or "Instructor"). * The user chooses one of the options based on their role in the system.   3-**Display Role-Specific Fields:**   * Based on the selected role, the system displays the relevant form fields for registration.   + **Student** might be asked for:     - Full Name     - Email Address     - Password     - Confirm Password     - Department   + **Instructor** might be asked for:     - Full Name     - Email Address     - Password     - Confirm Password     - Department   3-**User Fills Registration Form:**   * The user fills in the required details in the registration form based on the role they selected. * The system validates the inputs (e.g., checking for empty fields, correct email format, matching passwords, etc.).   4- **Submit Registration Form:**   * The user submits the registration form after filling in the required details. * The system verifies the data, including checking if the email already exists in the system.   5- **Display Registration Confirmation:**   * Upon successful registration, the system displays a confirmation message and may also send a verification email to the user with a link to verify their email address.   6- **Redirect to Login Page:**   * After registration, the user is either automatically logged in or redirected to the login page to log in with their new credentials |
| **Exceptions** | * Users can access the registration page. * Users can select their role (Student or Instructor) during registration. * System validates the user input during registration. * User receives a confirmation message after successful registration. * User profile is created in the system after successful registration. * Email verification (if required) is sent to the user. * User is redirected to the login page after registration (if applicable). * Role-specific permissions are assigned (Student or Instructor). * The system displays appropriate error messages for invalid inputs or registration failures. |
| **Actions** | * 1. System display to the user an alert “Please fill in all the required fields”.   2. User fills in the missing information.   3. System display to the user “Email already registered”.   4. User enters another email.   5. System display to the user “Username already exist”.   6. User enters another username. |
| **Pre-Condition** | * User must not be logged in. * User must have access to the registration page. * Role selection must be available. * Internet connection and system accessibility must be present. |
| **Post Condition** | * + User registration is successful or fails with proper error handling.   + User profile is created with appropriate details.   + Email verification and role-based permissions are applied.   + User is redirected to login or is automatically logged in.. |

ff

Use Case 2

|  |  |
| --- | --- |
| Log-In | |
| **Use case Name** | Log-In |
| **Actors** | Student/ instructor (registered) |
| **Main success scenario** | 1. Users navigate to the login page. 2. Users enter their registered email and password. 3. Users click the login button. 4. System validates the provided credentials. 5. System grants access if credentials are valid. 6. Users are redirected to the main dashboard. 7. Users can see role-specific features and content. 8. Login is successful, and user is logged into the application. |
| **Exceptions** | * Users can access the login page. * Users can enter a valid email and password. * System validates the user credentials correctly. * Users are granted access to the system if credentials are correct. * User is directed to the correct page based on their role (Student, Instructor). * The system provides error messages if login fails (e.g., incorrect email or password). * User can request password reset if they forget their password. * Session is initiated after successful login for the user. |
| **Actions** | 1. User accesses the login page from the login URL. 2. Users enter their email and password in the respective fields. 3. User clicks on the 'Login' button to submit the form. 4. System checks the email and password against the database for validation. 5. If the credentials match, the system authenticates the user. 6. Users are logged into the application and redirected to their role-specific page (dashboard). 7. If login fails, an error message is displayed (e.g., 'Invalid email or password'). 8. Users can click the 'Forgot Password' link to initiate a password reset process. 9. System provides the option to either reset the password or contact support in case of a failed login attempt. |
| **Pre-Condition** | * Users must be registered with a valid email and password. * Users have an active internet connection. * users have the correct login credentials. * The user is attempting to log in through a supported browser. * Users has access to the login page. |
| **Post Condition** | * User successfully logs in and gains access to the system. * User is redirected to the appropriate page based on their role (Student or Instructor). * Incorrect login attempts trigger an error message and a retry option. * Password reset process is initiated if the user clicks the 'Forgot Password' link. * The system securely stores the user's session information in subapase in Authentication part for further interaction. |

Use Case 3

|  |  |
| --- | --- |
| Enroll | |
| **Use case Name** | Enroll in courses |
| **Actors** | Student |
| **Main success scenario** | * Students access the course catalog or course list. * Students search or browse for a course they want to enroll in. * Students click on the "Enroll" button for the selected course. * The system checks for course prerequisites, availability, and user eligibility. * The system enrolls the student in the course if conditions are met. * Students receive confirmation of successful enrollment. * Students can now access course materials and track progress. |
| **Exceptions** | * Students can easily navigate to the course catalog or list. * Students can search or filter courses based on criteria like subject or availability. * System ensures the course is open for enrollment and has available seats. * A student is properly enrolled after clicking the "Enroll" button, given eligibility criteria are met. * A student receives confirmation upon successful enrollment. * System provides clear error messages if prerequisites or availability conditions are not met (e.g., course is full or prerequisites are missing). * Enrollment success triggers access to course materials and assignments. |
| **Actions** | * Student navigates to the course catalog or course list section. * Student filters or searches for a specific course. * Students clicks on the "Enroll" button next to the desired course. * System verifies if the student meets prerequisites or prerequisites are waived. * System checks if there are available spots in the course. * System registers the student in the course if all conditions are met. * Student receives a notification confirming enrollment |
| **Pre-Condition** | * Students must have a valid user account and be logged in. * The student must be eligible to enroll in the course if the admin approves. * Course must be open for enrollment. * The student has an active internet connection and access to the course platform. |
| **Post Condition** | * The student is successfully enrolled in the course. * Students receive enrollment confirmation. * Students can view the course materials, attend sessions, and participate in assignments. * Student enrollment status is updated in the system and visible in their profile. * Course materials become accessible for the enrolled student (e.g., syllabus, lectures, resources). |
|  |  |

Use Case 4

|  |  |
| --- | --- |
| Access  Course material | |
| **Use case Name** | Access course material |
| **Actors** | User |
| **Main success scenario** | * The student logs in with valid credentials. * The student selects the course they are enrolled in. * The student accesses the course material (e.g., lectures, notes, videos). * The material loads correctly and is available for the student to view or download. |
| **Exceptions** | The student should be able to find and open the course materials without errors. |
| **Actions** | * Log in to the Grade A system with valid credentials. * Navigate to the "Courses" section. * Select the appropriate course from the list of enrolled courses. * Click on the material links (e.g., PDF files, videos, documents) to view or download the content. |
| **Pre-Condition** | * The student must have a valid account and be enrolled in at least one course. * The student must be logged into the GradeA system. * Course materials must be uploaded and available in the system. |
| **Post Condition** | * The student has accessed the course material, and it is available for further interaction (view, download, or view additional details). * If any issues arise (e.g., a material is unavailable), the system should notify the student appropriately. |

Use Case 5

|  |  |
| --- | --- |
| Track progress | |
| **Use case Name** | Track progress |
| **Actors** | Student |
| **Main success scenario** | * The student logs into the Grade A system with valid credentials. * The student navigates to the "Progress" or "Dashboard" section. * The system displays the student's progress, including completed modules, assignments, tests, grades, and overall performance. * The student can view detailed information about their progress, such as percentages, achievements, and any upcoming tasks. |
| **Exceptions** | * The student should see clear and visual indicators (e.g., progress bars, completion percentages) representing their current standing in each course. * All progress data should be updated in real-time or with minimal delay. * The system should provide actionable insights, such as next steps or areas needing improvement. |
| **Actions** | * Log in to the GradeA system using valid credentials. * Navigate to the "Progress" or "Dashboard" section. * Select the course or module they want to track progress * Review the progress details, including completion rates, grades, and milestones. |
| **Pre-Condition** | * The student must have a valid account and be enrolled in at least one course. * The student must have completed some course activities (e.g., assignments, quizzes, or lessons) to track progress. * The system must be collecting and recording progress data (e.g., grades, completion statuses, milestones). |
| **Post Condition** | * The system displays up-to-date progress information for the student, including course completion rates, grades, and upcoming tasks. * The student has a clear understanding of their progress and areas needing improvement. |

Use Case 6

|  |  |
| --- | --- |
| upload material | |
| **Use case Name** | Upload Material |
| **Actors** | Instructor |
| **Main success scenario** | * The instructor logs into the Grade A system with valid credentials. * The instructor navigates to the "Course Materials" or "Content Upload" section. * The instructor selects the course to which the material will be uploaded. * The instructor uploads the course materials (e.g., PDF files, documents, videos) via the system interface. * The materials are successfully uploaded and are now available for students to access. |
| **Exceptions** | * The instructor should be able to upload files without errors, and the process should be intuitive. * The system should show a success message upon successful upload and an error message if the upload fails. * The uploaded materials should be immediately available for students to view. |
| **Actions** | * Log into the Grade system using valid instructor credentials. * Navigate to the course management section of the GradeA system. * Select the course to which materials will be uploaded. * Choose the "Upload Materials" option, select files (documents, videos, etc.), and upload them to the system. * Confirm the upload and ensure the materials are listed in the course content section. |
| **Pre-Condition** | * The instructor must have a valid account and be assigned to the course. * The course must exist in the system, and the instructor must have permission to upload content to that course. * The system must support file uploads, with a configured limit for file size and allowed formats (e.g., PDF, DOCX). |
| **Post Condition** | * The uploaded materials are now available in the course content section for students to access. * The system stores metadata related to the uploaded files (e.g., file name, upload time, and file size). * The system logs the instructor's upload action for auditing purposes. |

Use Case 7

|  |  |
| --- | --- |
| Provide  feebback | |
| **Usercase Name** | Provide feedback |
| **Actors** | Instructor |
| **Main success scenario** | * The instructor logs into the GradeA system with valid credentials. * The instructor navigates to the "Assignments" or "Grades" section. * The instructor selects a specific student’s submission for feedback. * The instructor enters comments or feedback in the designated field. * The feedback is saved, and the student is notified that feedback is available. |
| **Pre-Condition** | * The instructor must have a valid account and be assigned to a course. * The student must have submitted the assignment, quiz, or exam. * The system should have a mechanism for instructors to view submitted work. |
| **Post Condition** | * The feedback is saved and associated with the student’s submission. * The student can view the feedback in their gradebook or assignment portal. * The system logs the feedback activity for auditing purposes. |

Use Case 8

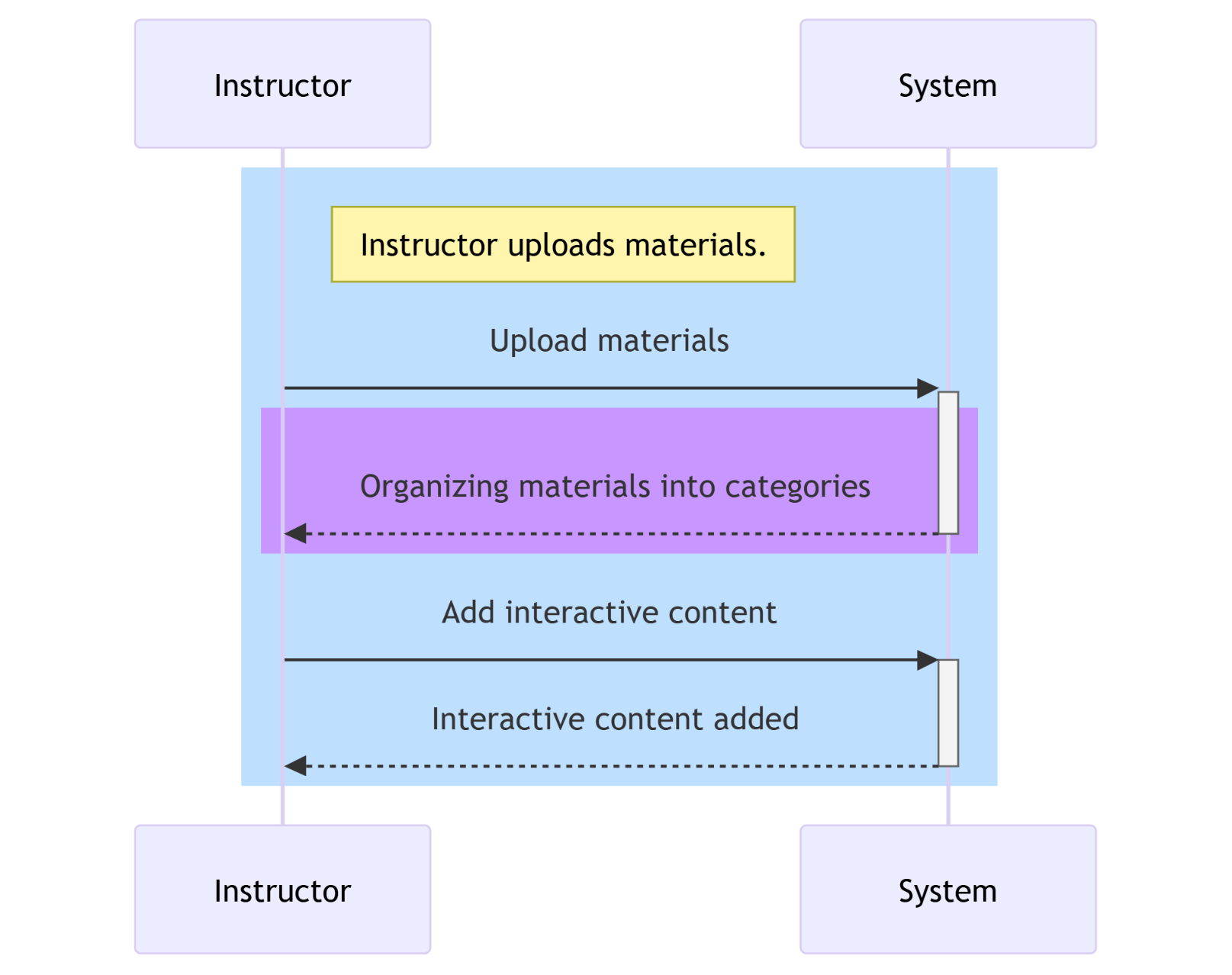
|  |  |
| --- | --- |
| Manage  courses | |
| **Use case Name** | Manage courses |
| **Actors** | Instructor |
| **Main success scenario** | * The instructor logs into the GradeA system with valid credentials. * The instructor navigates to the "Course Management" section. * The instructor selects the option to create, update, or delete a course. * If creating, the instructor enters course details (e.g., course name, description, materials). * The system updates or deletes the course accordingly, with feedback confirming the action. |
|  |  |
| **Exceptions** | * The instructor should be able to create new courses, modify existing ones, or delete courses without errors. * The system should provide feedback upon successful creation, update, or deletion. * The system should prevent instructors from deleting courses that have enrolled students or other dependencies. |
| **Actions** | * Log into the Grade system with valid instructor credentials. * Navigate to the "Course Management" section of the system. * Select the option to create, update, or delete a course. * Provide the required information for course creation or make necessary updates. * Confirm the action, and the system reflects the changes immediately. |
| **Pre-Condition** | * The instructor must have valid credentials and be authorized to manage courses. * The instructor must be in charge of one or more courses. * The system must be configured to allow course creation, updates, or deletions. |
| **Post Condition** | * The course is successfully created, updated, or deleted as per the instructor’s action. * The system reflects the changes in the course catalog or list of available courses. |

Use Case 9

|  |  |
| --- | --- |
| System  Mangement | |
| **Use case Name** | System Management |
| **Actors** | Admin |
| **Main success scenario** | * The admin logs into the GradeA system with valid administrator credentials. * The admin navigates to the "System Management" section. * The admin performs actions such as adding or removing users, modifying system settings, or monitoring system performance. * The system successfully processes the changes, and the admin receives confirmation of actions performed. |
| **Exceptions** | * The admin should be able to add, remove, or modify user accounts (instructors, students, etc.). * The system should provide clear, actionable feedback on each task (e.g., account creation, updates). * The system should maintain a record of all administrative changes for auditing purposes. |
| **Actions** | * Log into the Grade A system using valid admin credentials. * Navigate to the "System Management" or "Admin Panel" section. * Add, remove, or modify user accounts (students, instructors, etc.). * Modify system settings such as permissions, access levels, or configuration parameters. |
| **Pre-Condition** | * The admin must have a valid account with appropriate privileges to perform system management tasks. * The system must be operational and accessible for administration. * The admin must have access to the required management tools, dashboards, and logs. |
| **Post Condition** | The system reflects any changes made by the admin, such as new or updated user accounts and system settings. |

## Sequence Diagram

## Upload material



## Provide feedback

## 

## Access course materials

## 

## System Management

## 

## Class Diagram

## A screenshot of a computer screen

# Test Cases

## Test Cases for Login

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Outcome | Status |
| TC\_Login\_01 | Verify that the login form displays correctly. | The login form should contain fields for email, password, and a login button. |  |
| TC\_Login\_02 | Test login with valid credentials. | User should be redirected to the dashboard page. |  |
| TC\_Login\_03 | Test login with invalid credentials. | An error message should be displayed. |  |
| TC\_Login\_04 | Test login with empty email field. | An error message should appear indicating that the email is required. |  |
| TC\_Login\_05 | Test login with empty password field. | An error message should appear indicating that the password is required. |  |
| TC\_Login\_06 | Test login with incorrect email format. | The email field should display an error message for incorrect format. |  |
| TC\_Login\_07 | Test forgot password functionality. | When a valid email is entered, a password reset email should be sent. |  |
| TC\_Login\_08 | Test forgot password with empty email field. | An alert should prompt the user to enter an email for password reset. |  |

## Test cases for register

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Outcome | Status |
| TC\_Register\_01 | Verify that the registration form displays correctly. | The form should display fields for name, email, password, and confirm password. |  |
| TC\_Register\_02 | Test registration with valid credentials. | User should be successfully registered and redirected to the login page. |  |
| TC\_Register\_03 | Test registration with existing email. | An error message should appear indicating that the email is already in use. |  |
| TC\_Register\_04 | Test registration with mismatched password and confirm password. | An error message should be displayed indicating that passwords do not match. |  |
| TC\_Register\_05 | Test registration with empty email field. | An error message should appear indicating that the email is required. |  |
| TC\_Register\_06 | Test registration with empty password field. | An error message should appear indicating that the password is required. |  |
| TC\_Register\_07 | Test registration with invalid email format. | The email field should display an error message for incorrect format. |  |
| TC\_Register\_08 | Test successful registration with valid data and check for login redirection. | The user should be redirected to the login page after successful registration. |  |

## 3.3 Test Cases for Dashboard

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Outcome | Status |
| TC\_Dashboard\_01 | Verify that the correct role (Student/Professor) is displayed. | Displays 'Welcome, Student!' or 'Welcome, Professor!' based on the user's role. |  |
| TC\_Dashboard\_02 | Verify that the user's courses are displayed correctly. | Lists the courses associated with the user, displaying their names and descriptions. |  |
| TC\_Dashboard\_03 | Test the 'Manage Students' button for professors. | Professors should see a 'Manage Students' button and clicking it should show an alert. |  |
| TC\_Dashboard\_04 | Verify behavior when no courses are available for the user. | Displays a message 'No courses found.' if the user has no courses. |  |
| TC\_Dashboard\_05 | Test the loading state while data is being fetched. | A loading message 'Loading dashboard...' should be displayed while the data is loading. |  |
| TC\_Dashboard\_06 | Test error handling during data fetching. | Any errors during data fetching should be handled gracefully (e.g., error message displayed in console or UI). |  |
| TC\_Dashboard\_07 | Test the redirect functionality for users not logged in. | If the user is not logged in, they should be redirected to the login page. |  |

# Links

## Scrum Meeting Link

## Repository link of GitHub

## <https://github.com/AbdelRahmanRahal/GradeA.git>