Design and Implementation of an E-Learning Platform

CCAS.4.3 Software Engineering

Project

Ahmed Mahmoud - 202101585
Donia Elanany - 202101818
Nada Adel - 202102397

Contents

1 Requirements

1.1 Functional Requirements

ID	Requirement	Description	Category
REQ_001	Manage student profiles	Store and manage student information including name, ID, email, and enrolled courses	Must have
REQ_002	Manage instructor profiles	Store and manage instructor information including name, ID and email.	Must have
REQ_003	Provide a course catalog	Display a searchable catalog of available courses with details like name, description, and instructor	Must have
REQ_004	Allow course enrollment	Enable students to enroll in courses through the platform	Must have
REQ_005	Provide access to learning materials	Enable students to access course-related materials such as PDFs, PowerPoint presentations, and assignments	Must have
REQ_006	Manage payments and invoices	Process payments for paid courses and provide invoices to students	Must have
REQ_007	Store and manage assignments	Allow instructors to upload assignments, and students to submit them within the platform	Must have
REQ_008	User Authentication (Login)	Allow users to log in to the platform using their credentials	Must have
REQ_009	User Registration (Signup)	Allow new users to register on the platform, specifying their role (student or instructor)	Must have
REQ_010	Password Reset	Allow users to request a password reset and securely reset their passwords	Must have
REQ_011	Display course details	Display the details of a specific course, including learning materials and assignments	Must have
REQ_012	Submit Assignments	Allow students to submit assignments to a given course	Must have
REQ_013	Remove Assignment Submissions	Allow students to remove submitted assignments	Must have
REQ_014	Grade Submissions	Allow instructors to grade submitted assignments	Must have
REQ_015	View Student Submissions	Allow instructors to view all assignment submissions of a student in a particular course	Must have
REQ_016	Remove Learning Material	Allow instructors to remove uploaded learning materials	Must have
REQ_017	Remove Assignments	Allow instructors to remove uploaded assignments	Must have
REQ_018	Allow user or course deletion	Enable administrators to delete user accounts or courses, ensuring associated data is removed or updated accordingly	Should have
REQ_019	Track course progress	Allow students and instructors to monitor progress within a course, including completed lessons and grades	Could have

REQ_020	Support online assessments	Provide functionality for quizzes, tests, and other assessments within courses	Could have
REQ_021	Provide course feedback system	Allow students to submit feedback on courses, instructors, and learning materials	Could have
REQ_022	Allow course reviews and ratings	Enable students to provide feedback on courses they have completed	Could have
REQ_023	Enable communication between students and instructors	Provide messaging, forums, or Q&A features to facilitate communication	Could have
REQ_024	Provide attendance tracking for live sessions	Track and record attendance during live online sessions	Won't have
REQ_025	Support multiple languages	Allow students and instructors to use the platform in different languages	Won't have

ID	Requirement	Description	Category
REQ_026	Scalability and Performance	The platform should support high concurrent usage with fast response times and be scalable for future growth	Must have
REQ_027	Data Integrity	Ensure that all transactions (e.g., payments, grades, and enrollment) are consistent and error-free	Must have
REQ_028	Compatibility	The system should be compatible with major browsers (Chrome, Firefox, Safari) and operating systems	Must have
REQ_029	Backup and Recovery	Implement daily data backups with recovery capabilities to prevent data loss	Should have
REQ_030	Maintainability	The platform codebase should be modular and well-documented to support future updates and bug fixes	Should have
REQ_031	Analytics and Reporting	Provide analytics on user engagement, and platform usage statistics	Should have
REQ_032	Resource Efficiency	Optimize server and database usage to reduce operational costs while maintaining performance	Should have
REQ_033	Audit Logs	Maintain detailed logs of user and system activities for auditing and troubleshooting purposes	Should have
REQ_034	Accessibility	Ensure compliance with accessibility standards (e.g., WCAG 2.1) to support users with disabilities	Should have
REQ_035	Localization	Allow the platform to adapt to different regional requirements, such as time zones and currencies	Could have

1.2 Use Case Scenarios for Functional Requirements

1. **Use Case: User Registration (Signup)**
   - **Actors:** New User (Student or Instructor)
   - **Description:** New users can create an account on the platform, specifying their role (student or instructor).
   - **Preconditions:**
     - The user must have a valid email address.
   - **Scenario:**
     – The new user navigates to the signup page of the platform.
     – The user provides required details such as name, email, and password.
     – The user specifies their role (student or instructor).
     – The system creates an account for the user and sends a confirmation email.
   - **Postconditions:**
     - The new user's account is created, and they can log in to the platform.
2. **Use Case: User Authentication (Login)**
   - **Actors:** Student, Instructor
   - **Description:** Users can log into the platform using their credentials (email and password).
   - **Preconditions:**
     - The user must have an active account on the platform.
   - **Scenario:**
     – The user navigates to the login page of the platform.
     – The user enters their registered email and password.
     – The system verifies the credentials and grants access to the platform.
   - **Postconditions:**
     - The user is logged into the platform and granted access to the appropriate features based on their role.
3. **Use Case: Profile Management (Student & Instructor)**
   - **Actors:** Student, Instructor
   - **Description:** Both students and instructors can view and update their profiles, which include their name, ID, email and enrolled courses (for students).
   - **Preconditions:**
     - The user must have an active account on the platform.
   - **Scenario:**
     – The user logs into the platform using their credentials.
     – They can view their current profile information.
     – If necessary, they can update their contact details or courses.
     – The system saves any updates made to the profile.
   - **Postconditions:**
     - The user's profile is updated with the latest changes.
4. **Use Case: Course Enrollment**
   - **Actors:** Student
   - **Description:** A student can enroll in available courses offered by instructors. The student must pay for the course and receive a confirmation email upon successful enrollment.
   - **Preconditions:**

- The student must be logged into the platform.
- The student must have a valid account and course options available.

- **Scenario:**
  - The student logs into the platform and browses the available courses.
  - The student selects a course they wish to enroll in.
  - The student proceeds to the payment section and enters payment details.
  - The system processes the payment and confirms the transaction.
  - Once the payment is successful, the student is registered for the course.
  - The student receives a confirmation email with details of the course and the payment.
- **Postconditions:**
  - The student is enrolled in the selected course, and the system records the enrollment.
  - The payment is processed and recorded.
  - The student receives a confirmation email with enrollment and payment details.

5. **Use Case: Course Catalog Display**
   - **Actors:** Student
   - **Description:** Students can view a searchable catalog of available courses with details like name, description, and instructor.
   - **Preconditions:**
     - The student must be logged into the platform.
     - The platform must have available courses listed.
   - **Scenario:**
     - The student logs into the platform.
     - They access the course catalog.
     - The student can search for courses by name.
     - They can view details about each course, including the description, price, and instructor information.
   - **Postconditions:**
     - The student can view the courses and relevant details.
6. **Use Case: Assignment Submission**
   - **Actors:** Student
   - **Description:** A student can submit their assignments for a particular course through the platform.
   - **Preconditions:**
     - The student must be enrolled in the course.
     - The assignment must be available for submission.
   - **Scenario:**
     - The student logs into the platform and navigates to the course they are enrolled in.
     - They access the assignment section.
     - The student uploads the assignment file.
     - The student submits the assignment.
     - The system confirms the submission.
   - **Postconditions:**
     - The assignment is successfully submitted, and the system updates the submission record.
7. **Use Case: Instructor Grading**
   - **Actors:** Instructor
   - **Description:** An instructor can grade assignments submitted by students.

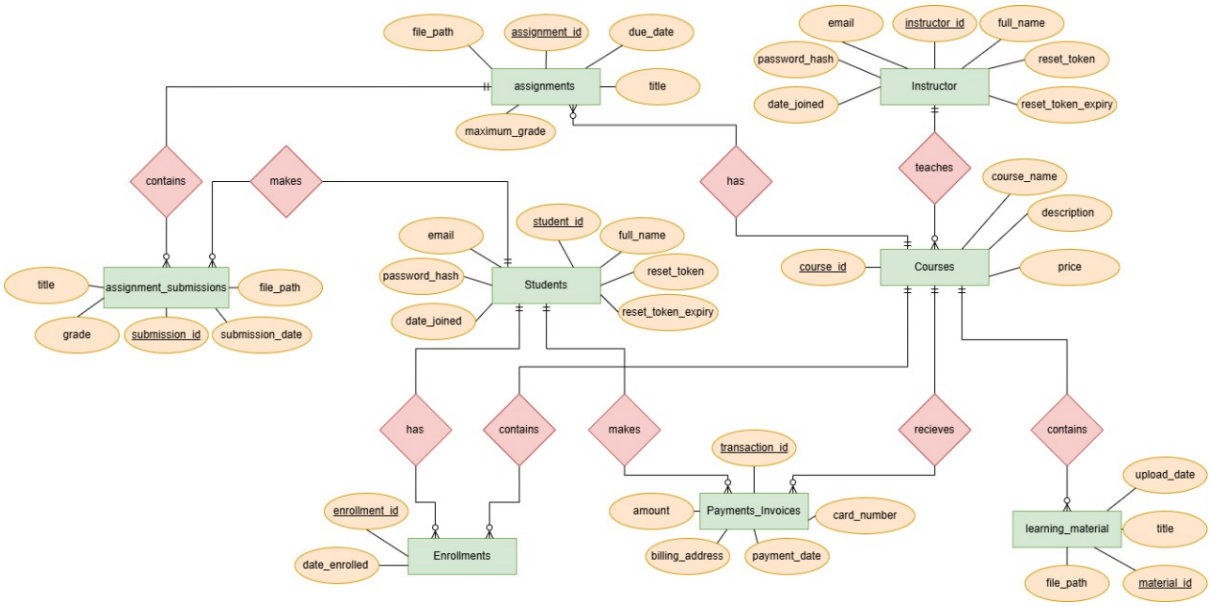
- **Preconditions:**
  - The instructor must be logged into the platform.
  - The instructor must have assignments to grade.
  - Students must have submitted assignments.
- **Scenario:**
  - The instructor logs into the platform and accesses the course they are teaching.
  - They view the list of submitted assignments.
  - The instructor grades the assignments based on the submission content.
  - The system saves the grade and updates the student's assignment record.
- **Postconditions:**
  - The student's assignment is graded, and the grade is recorded in the system.

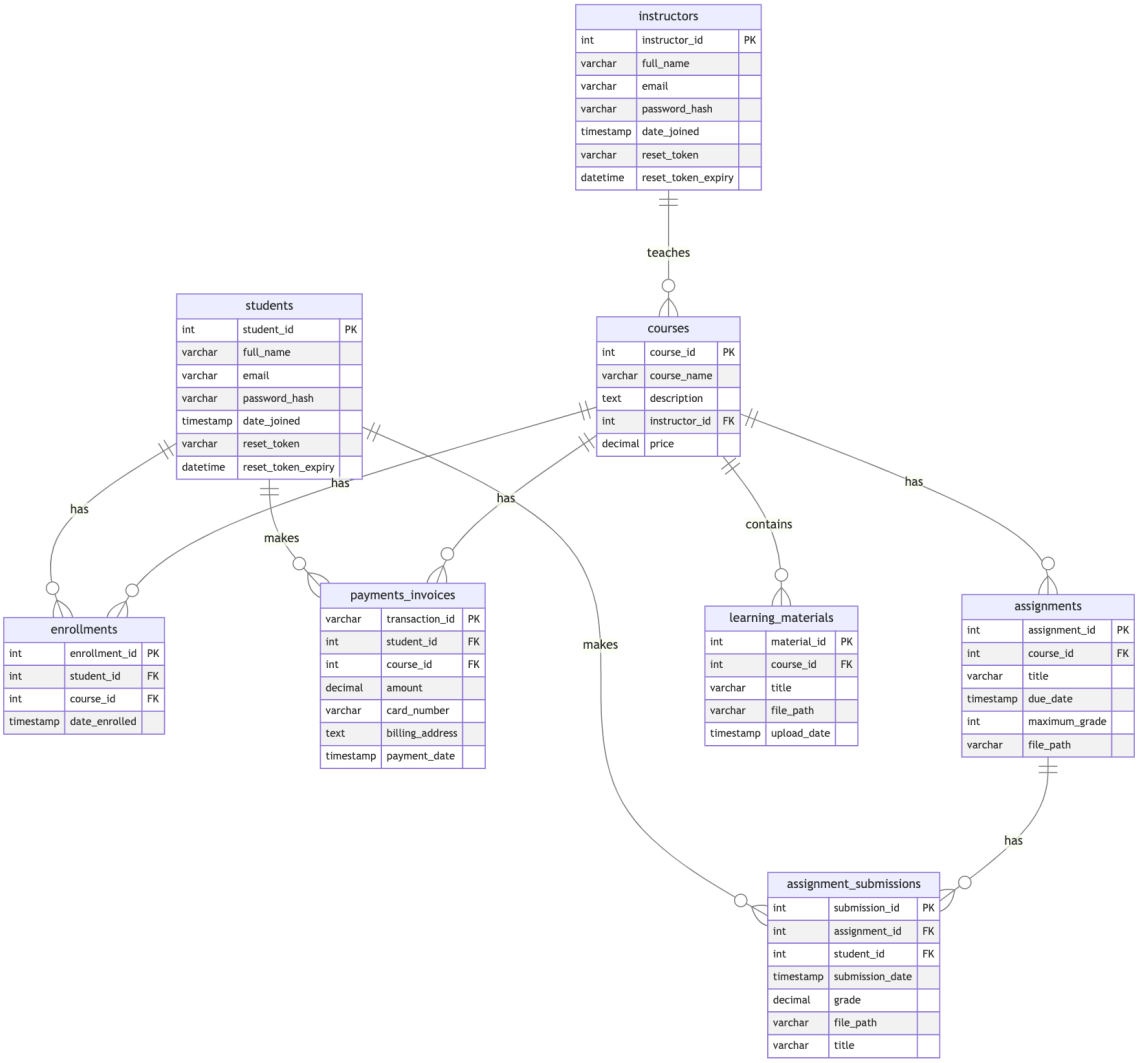
8. **Use Case: Learning Material Access**

- **Actors:** Student, Instructor
- **Description:** Students can access course-related learning materials, and instructors can upload new materials.
- **Preconditions:**
  - The student must be enrolled in the course.
  - The instructor must have uploaded learning materials.
- **Scenario:**
  - The student logs into the platform and navigates to the course they are enrolled in.
  - The student accesses the learning materials section.
  - The student can download or view materials such as PDFs, presentations, and other course content.
  - The instructor logs into the platform and uploads new learning materials as needed.
- **Postconditions:**
  - The student has access to the materials, and any newly uploaded materials are available.

2 DBMS Design and Implementation

2.1 Entity-Relationship Diagram (ERD)





2.2 Relational Database Schema

Students Table

Column Name	Data Type	Description
student_id	INT (PK)	Unique identifier for each student
full_name	VARCHAR(255)	Full name of the student
email	VARCHAR(255)	Email address of the student
password_hash	VARCHAR(255)	Encrypted password
date_joined	DATE	Date the student joined the platform
reset_token	VARCHAR(255)	Token for password reset
reset_token_expiry	DATETIME	Expiry date of reset token

Instructors Table

Column Name	Data Type	Description
instructor_id	INT (PK)	Unique identifier for each instructor
full_name	VARCHAR(255)	Full name of the instructor
email	VARCHAR(255)	Email address of the instructor
password_hash	VARCHAR(255)	Encrypted password
date_joined	DATE	Date the instructor joined the platform
reset_token	VARCHAR(255)	Token for password reset
reset_token_expiry	DATETIME	Expiry date of reset token

Courses Table

Column Name	Data Type	Description
course_id	INT (PK)	Unique identifier for each course
course_name	VARCHAR(255)	Name of the course
description	TEXT	Description of the course
instructor_id	INT (FK)	Foreign key referencing instructors table
price	DECIMAL(10,2)	Price of the course

Enrollments Table

Column Name	Data Type	Description
enrollment_id	INT (PK)	Unique identifier for each enrollment
student_id	INT (FK)	Foreign key referencing students table
course_id	INT (FK)	Foreign key referencing courses table
date_enrolled	DATE	Date the student enrolled in the course

Assignments Table

Column Name	Data Type	Description
assignment_id	INT (PK)	Unique identifier for each assignment
course_id	INT (FK)	Foreign key referencing courses table
title	VARCHAR(255)	Title of the assignment
due_date	DATETIME	Due date for the assignment
maximum_grade	INT	Maximum grade for assignment
file_path	VARCHAR(255)	Path to the assignment file

Assignment Submissions Table

Column Name	Data Type	Description
submission_id	INT (PK)	Unique identifier for each submission
assignment_id	INT (FK)	Foreign key referencing assignments table
student_id	INT (FK)	Foreign key referencing students table
submission_date	DATETIME	Date the assignment was submitted
grade	DECIMAL(5,2)	Grade for the assignment
file_path	VARCHAR(255)	Path to the submitted file
title	VARCHAR(255)	Title of the submission

Payments Invoices Table

Column Name	Data Type	Description
transaction_id	VARCHAR(255) (PK)	Unique identifier for each transaction
student_id	INT (FK)	Foreign key referencing students table
course_id	INT (FK)	Foreign key referencing courses table
amount	DECIMAL(10,2)	Payment amount
card_number	VARCHAR(255)	Card number (encrypted)
billing_address	TEXT	Billing address
payment_date	DATETIME	Date of payment

Learning Materials Table

Column Name	Data Type	Description
material_id	INT (PK)	Unique identifier for each material
course_id	INT (FK)	Foreign key referencing courses table
title	VARCHAR(255)	Title of the learning material
file_path	VARCHAR(255)	Path to the file for the material
upload_date	DATETIME	Date the material was uploaded

2.3 SQL Queries

Insert Queries

QUR_NUM	SQL Query
QUR_001	`INSERT INTO students (email, password_hash, full_name)VALUES (?, ?, ?)`
QUR_002	`INSERT INTO instructors (email, password_hash, full_name)VALUES (?, ?, ?)`
QUR_003	`INSERT INTO payments_invoices (student_id, course_id, amount, transaction_id , card_number, billing_address)VALUES (?, ?, ?, ?, ?, ?)`
QUR_004	`INSERT INTO enrollments (student_id, course_id)VALUES (?, ?)`
QUR_005	`INSERT INTO learning_materials (course_id, title, file_path)VALUES (?, ?, ?)`
QUR_006	`INSERT INTO assignments (course_id, title, file_path, due_date, maximum_grade)VALUES (?, ?, ?, ?, ?)`
QUR_007	`INSERT INTO assignment_submissions (assignment_id, student_id, file_path, title)VALUES (?, ?, ?, ?)`

Update Queries

QUR_NUM	SQL Query
QUR_008	`UPDATE students SET full_name = ?, email = ? WHERE student_id = ?`
QUR_009	`UPDATE instructors SET full_name = ?, email = ? WHERE instructor_id = ?`
QUR_010	`UPDATE students SET reset_token = ?, reset_token_expiry = ? WHERE email = ?`
QUR_011	`UPDATE instructors SET reset_token = ?, reset_token_expiry = ? WHERE email = ?`
QUR_012	`UPDATE students SET password_hash = ?, reset_token = NULL, reset_token_expiry = NULL WHERE student_id = ?`
QUR_013	`UPDATE instructors SET password_hash = ?, reset_token = NULL, reset_token_expiry = NULL WHERE instructor_id = ?`
QUR_014	`UPDATE assignment_submissions SET grade = ? WHERE submission_id = ?`

Delete Queries

QUR_NUM	SQL Query
QUR_015	`DELETE FROM enrollments WHERE student_id = ? AND course_id IN (?)`
QUR_016	`DELETE FROM assignment_submissions WHERE student_id = ? AND assignment_id = ?`
QUR_017	`DELETE FROM learning_materials WHERE material_id = ?`
QUR_018	`DELETE FROM assignment_submissions WHERE assignment_id = ?`
QUR_019	`DELETE FROM assignments WHERE assignment_id = ?`

Select Queries

QUR_NUM	SQL Query
QUR_020	`SELECT * FROM (SELECT student_id AS userId, full_name, email, password_hash , 'student' AS role FROM students UNION ALL SELECT instructor_id AS userId , full_name, email, password_hash, 'instructor' AS role FROM instructors ) users WHERE email = ? LIMIT 1`

QUR_NUM	SQL Query
QUR_021	SELECT * FROM students WHERE student_id = ?
QUR_022	SELECT * FROM instructors WHERE instructor_id = ?
QUR_023	SELECT course_id, course_name FROM courses
QUR_024	SELECT courses.course_id, courses.course_name, courses.description, courses .price, instructors.full_name AS instructor_name, instructors.email AS instructor_email, courses.instructor_id FROM courses LEFT JOIN instructors ON courses.instructor_id = instructors.instructor_id
QUR_025	SELECT c.course_name, c.course_id FROM courses c JOIN enrollments e ON c. course_id = e.course_id WHERE e.student_id = ?
QUR_026	SELECT course_id, course_name FROM courses WHERE instructor_id = ?
QUR_027	SELECT student_id AS userId, full_name, email, password_hash, ‘student‘ AS role FROM students WHERE email = ? UNION ALL SELECT instructor_id AS userId , full_name, email, password_hash, ‘instructor‘ AS role FROM instructors WHERE email = ?
QUR_028	SELECT courses.*, instructors.full_name AS instructor_name, instructors. email AS instructor_email FROM courses LEFT JOIN instructors ON courses. instructor_id = instructors.instructor_id WHERE courses.course_id = ?
QUR_029	SELECT material_id, title, file_path FROM learning_materials WHERE course_id = ?
QUR_030	SELECT assignment_id, title, file_path, due_date, maximum_grade FROM assignments WHERE course_id = ?
QUR_031	SELECT email FROM students WHERE student_id = ?
QUR_032	SELECT * FROM students WHERE email = ?
QUR_033	SELECT * FROM instructors WHERE email = ?
QUR_034	SELECT * FROM (SELECT student_id AS userId, ‘students‘ AS tableName, reset_token_expiry FROM students WHERE reset_token = ? UNION ALL SELECT instructor_id AS userId, ‘instructors‘ AS tableName, reset_token_expiry FROM instructors WHERE reset_token = ?)AS combined WHERE reset_token_expiry > ? LIMIT 1
QUR_035	SELECT * FROM enrollments WHERE student_id = ? AND course_id = ?
QUR_036	SELECT due_date FROM assignments WHERE assignment_id = ?
QUR_037	SELECT a.assignment_id, a.title, a.due_date, s.submission_id, s.file_path, s.submission_date, s.title AS submission_title, s.grade FROM assignments a LEFT JOIN assignment_submissions s ON a.assignment_id = s.assignment_id WHERE a.course_id = ? AND s.student_id = ?
QUR_038	SELECT course_id FROM courses WHERE instructor_id = ?

Requirements and Associated Queries

Requirement	Associated Queries
REQ-001 (Manage student profiles)	QUR_008, QUR_015, QUR_021, QUR_025
REQ-002 (Manage instructor profiles)	QUR_013, QUR_022, QUR_026, QUR_038
REQ-003 (Provide a course catalog)	QUR_023, QUR_024
REQ-004 (Allow course enrollment)	QUR_004, QUR_035
REQ-005 (Provide access to learning materials)	QUR_005, QUR_029
REQ-006 (Manage payments and invoices)	QUR_003, QUR_031
REQ-007 (Store and manage assignments)	QUR_005, QUR_006, QUR_029, QUR_030
REQ-008 (User Authentication (Login))	QUR_020
REQ-009 (User Registration (Signup))	QUR_001, QUR_002, QUR_009
REQ-010 (Password Reset)	QUR_010, QUR_011, QUR_012, QUR_013, QUR_032, QUR_033, QUR_034,
REQ-011 (Display course details)	QUR_028, QUR_029, QUR_030
REQ-012 (Submit Assignments)	QUR_007, QUR_036
REQ-013 (Remove Assignment Submissions)	QUR_016
REQ-014 (Grade Submissions)	QUR_014
REQ-015 (View Student Submissions)	QUR_037
REQ-016 (Remove Learning Material)	QUR_017
REQ-017 (Remove Assignments)	QUR_018, QUR_019

3 GUI Development Using Prompt Engineering

Stage 1: Initial, Broad Prompt

Prompt: *"Create a basic website for an online learning platform. The platform should include pages for home, courses, login, signup, profile, and payment."*

Stage 2: Adding Structure and Basic Functionality

Prompt: *"Create a website for an online learning platform. It should include the following pages: a home page with a hero section, a courses page listing available courses, a login page, a signup page with fields for name, email, password and role, a profile page showing the user's information and enrolled courses, and a payment page with payment fields. The site should use HTML, CSS, and JavaScript. The user must have a log out button on the profile page."*

Stage 3: Refining Layout and Styling

Prompt: *"Create a website for an online learning platform. It should include the following pages: a home page with a hero section and a list of features, a courses page displaying courses in a grid layout, a login page with email and password fields, a signup page with fields for name, email, password, confirm password, and role (dropdown), a profile page showing the user's full name, email, join date, role and enrolled courses, and a payment page with fields for payment details. The site should use HTML, CSS, and JavaScript. The home, courses, and profile pages should have a navigation bar. The profile page should have an update profile button that takes the user to an update profile page. The course cards should have a unique look and a button to enroll or go to a specific course. The forms should have proper labels and placeholders. The website should have a clean and modern design with a consistent color scheme using orange as a primary color, a white background, and black for the text. The user must have a log out button on the profile page. The user should be able to toggle the search bar on the home, course, and profile pages."*

Stage 4: Adding Interactivity and Dynamic Functionality

Prompt: *"Create a website for an online learning platform. It should include the following pages: a home page with a hero section and a list of features, a courses page displaying courses in a grid layout, a login page with email and password fields, a signup page with fields for name, email, password, confirm password, and role (dropdown with student or instructor options), a profile page showing the user's full name, email, join date, role and enrolled courses, a payment page with payment details, and a course detail page. The site should use HTML, CSS, and JavaScript. The home, courses, and profile pages should have a responsive navigation bar. The profile page should have an update profile button that takes the user to an update profile page where they can update the data and also remove courses. The course cards should have a button to enroll or go to a specific course, and it must have a unique card look and responsive style. The login and signup forms must use proper labels and placeholders. The website should have a clean and modern design with a consistent color scheme using orange as a primary color, a white background, and black for the text. The courses page should load the courses dynamically from a JSON file, and be searchable using the name. The search bar should be a togglable overlay on the navbar, and should dynamically update the results as the user types. The payment form should validate user inputs such as card number (16 digits), expiry date (MM/YY), CVV (3-4 digits), and billing address, and send a POST request to '/submit-payment'. The website should implement user sessions, and the user must have a log out button on the profile page which sends a post request to '/api/logout' using the fetch API, and redirects the user to the home page after successful logout. The profile page must also load the user information and the courses they are enrolled in via a GET request to the '/api/profile' endpoint. The update profile form must also send a POST request to '/submit-profile-update'. The website should allow the student to remove the courses they are enrolled in. The course details page must display the learning materials and assignments, and the instructor must have the option to upload new material and assignments, and if logged in, the student has the option to submit their answer."*

Stage 5: Adding More Detail & Refining Layout and Functionality

Prompt: *"Create a website for an online learning platform. It should include the following pages: a home page with a hero section and a list of features, a courses page displaying courses in a grid layout, a login page with email and password fields, a signup page with fields for name, email, password, confirm password, and role (dropdown with student or instructor options), a profile page showing the user's full name, email, join date, role and enrolled courses, a payment page with payment details, and a course detail page with the ability to upload material and assignments for the instructors, and the ability to submit answers for the students. The site should use HTML, CSS, and JavaScript. The home, courses, and profile pages should have a responsive navigation bar. The profile page should have an update profile button that takes the user to an update profile page where they can update the data and also remove courses, the update profile form must send the data using a post request to '/submit-profile-update'. The courses page should load the courses from the '/courses/list' endpoint, and the courses can be filtered by name. The search bar should be a togglable overlay on the navbar, and should dynamically update the results as the user types, and take the user to the course detail page. The courses should show the instructor name and email. The payment form should validate user inputs such as card number (16 digits), expiry date (MM/YY), CVV (3-4 digits), and billing address, and send a POST request to '/submit-payment'. The website should implement user sessions, and the user must have a log out button on the profile page which sends a post request to '/api/logout' using the fetch API, and redirects the user to the home page after successful logout. The profile page must also load the user information and the courses they are enrolled in via a GET request to the '/api/profile' endpoint. On the course details page, the instructor should be able to upload material and assignments, and submit them via a POST request to '/api/upload-material' and add a due date for assignments, while also being able to see the existing assignments. The student should be able to submit their answer for assignments using the '/api/submit-assignment' endpoint. If the user is an instructor, they have the option to grade assignments, and remove them and they are also able to remove learning material. The website should have a clean and modern design with a consistent color scheme using orange as a primary color, a white background, and black for the text. Use transition effects on hover for buttons and navigation links. The submitted assignments should be removable, and the user should receive a message after successful payment using the 'payment-success.html' page, and the system should send an email for the successful payment. The website should allow a user to request a password reset, using a '/forgot-password' form and use a form to reset the password at the /reset-password route."*

4 Testing and Validation

To ensure that the e-learning platform functions as intended and meets the project requirements, the following testing strategy was used:

4.1 Test Plan

A comprehensive test plan was developed that includes the following levels of testing:

- Integration Testing: Test cases were defined to validate the interaction between components such as user registration, course enrollment, and assignment submission.
- System Testing: End-to-end test cases were created to ensure that the platform functions correctly as a whole, such as a complete workflow from user login to assignment grading.
- Acceptance Testing: Test cases were designed to verify that the platform fulfills all specified functional requirements, including user authentication and course management.

4.2 Manual Test Cases

1. User Registration (Signup)

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_001	Register a new student successfully	Valid email and input data	Name, valid email, password	1. Navigate to the sign up page. 2. Enter valid details. 3. Click Sign Up.	Student account is created successfully, and user is redirected to the login page.	As expected	Pass
TC_002	Register a new instructor successfully	Valid email and input data	Name, valid email, password	1. Navigate to the sign up page. 2. Enter instructor details. 3. Click Sign Up.	Instructor account is created successfully, and user is redirected to the login page.	As expected	Pass
TC_003	Attempt to register with an already used email	Duplicate email	Name, duplicate email, password	1. Navigate to the sign up page. 2. Enter duplicate email. 3. Click Sign Up.	Registration fails, and error message ”Email is already registered as a student or instructor” is displayed.	As expected	Pass

2. User Login

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_004	Login with valid credentials (student)	Student account exists	Valid email, password	1. Navigate to the login page. 2. Enter valid credentials. 3. Click login.	User logs in successfully and is redirected to the profile page.	As expected	Pass
TC_005	Login with valid credentials (instructor)	Instructor account exists	Valid email, password	1. Navigate to the login page. 2. Enter instructor credentials. 3. Click login.	Instructor logs in successfully and is redirected to the profile page.	As expected	Pass
TC_006	Attempt login with invalid credentials	Account exists	Invalid email or password	1. Navigate to the login page. 2. Enter invalid credentials. 3. Click login.	Login fails, and error message "Invalid email or password" is displayed.	As expected	Pass

3. Course Enrollment

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_007	Successfully enroll in a course	Student logged in	Course ID, payment details	1. Navigate to the course catalog. 2. Select a course. 3. Proceed to payment. 4. Confirm payment.	Enrollment is successful, and confirmation email is sent.	As expected	Pass
TC_008	Attempt to enroll in a course with an invalid card number	Student logged in, course selected and on payment page	Course ID, payment details, card number with only 10 digits	1. Enter payment details, including a card number with only 10 digits. 2. Click submit payment.	Payment fails, and error message "Please match the requested format. Card number must be 16 digits" is displayed.	As expected	Pass

4. Assignment Submission

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_009	Submit an assignment successfully	Student enrolled in course	Assignment file	1. Navigate to the course page. 2. Select assignment. 3. Upload file. 4. Submit.	Assignment submission is successful, and confirmation message is displayed.	As expected	Pass
TC_010	Submit a revised assignment	Assignment already submitted	New assignment file	1. Navigate to the assignment page. 2. Remove existing submission. 3. Upload and submit revised file.	Revised assignment is submitted successfully.	As expected	Pass
TC_011	Attempt to submit an assignment after due date	Assignment overdue	Assignment file	1. Navigate to the course page. 2. Attempt to submit overdue assignment.	Submission fails, and error message "Assignment due date has passed!" is displayed.	As expected	Pass

5. Assignment Grading

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_012	Grade a submitted assignment successfully	Instructor logged in, assignment submitted by student	Assignment ID, student submission, grade	1. Navigate to the course page. 2. Select a student's submission to an assignment. 3. Enter a grade. 4. Submit the grade.	A confirmation message is displayed. Grade is saved successfully, and the student can view it.	As expected	Pass
TC_013	Update a grade for an assignment	Grade already assigned	Assignment ID, student submission, new grade	1. Navigate to the graded assignment page. 2. Edit the grade. 3. Save changes.	A confirmation message is displayed. Updated grade is saved successfully, and the student can view it.	As expected	Pass
TC_014	Attempt to enter a grade above the maximum grade	Instructor logged in, assignment submitted by student, maximum grade defined	Assignment ID, student submission, grade above maximum	1. Navigate to the course page. 2. Select a student's submission to an assignment. 3. Enter a grade above the maximum grade. 4. Submit the grade.	A validation message is displayed, indicating that the entered grade exceeds the maximum allowed. The grade is not saved.	As expected	Pass

6. Learning Material Management

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_015	Upload new learning material successfully	Instructor logged in	Material file	1. Navigate to the course page. 2. Select "Upload Material". 3. Upload file. 4. Submit.	A confirmation message is displayed. Material is uploaded successfully and visible to students.	As expected	Pass
TC_016	Replace existing learning material	Existing material uploaded	New material file	1. Navigate to the materials page. 2. Remove existing material file. 3. Upload new file.	A confirmation message is displayed. Updated material is saved successfully.	As expected	Pass

7. Password Reset

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_017	Request a password reset successfully	User registered, valid email	Valid email address	1. Navigate to the login page. 2. Select "Forgot Password". 3. Enter valid email. 4. Submit request.	Password reset email is sent successfully.	As expected	Pass
TC_018	Reset password successfully using the link	Password reset email received	New password	1. Open password reset email. 2. Click on reset link. 3. Enter new password. 4. Confirm password change.	Password reset successfully, and user can log in with new password.	As expected	Pass
TC_019	Attempt to reset password with invalid link	Expired or invalid reset link	Invalid link	1. Navigate to reset link. 2. Enter new password. 3. Submit request.	Error message "Invalid or expired link" is displayed.	As expected	Pass

8. Updating User Profile

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_020	Successfully update user email and name	User logged in	New email, new name	1. Navigate to the profile page. 2. Click "Update Profile". 3. Enter a new email and name. 4. Click "Update Profile".	Profile is updated successfully, and confirmation message "Profile updated" is displayed.	As expected	Pass
TC_021	Successfully drop a course	User enrolled in a course	Course name to drop	1. Navigate to the profile page. 2. Click "Update Profile". 3. Select course to drop. 4. Click "Update Profile".	Course is removed from the user's enrolled courses list.	As expected	Pass
TC_022	Attempt to update profile with blank name field	User logged in	Blank name	1. Navigate to the profile page. 2. Click "Update Profile". 3. Leave the name field blank. 4. Click "Update Profile".	Update fails, and error message indicating that name can't be blank is displayed. Course is removed from the user's enrolled courses list	As expected	Pass

9. Course Search

Test Case ID	Test Case Description	Pre-conditions	Test Data	Test Case Steps	Expected Results	Actual Results	Pass or Fail
TC_001	Search for an existing course	The system contains courses	Search term: "Data Science"	1. Click on "Search Courses". 2. Enter "Data Science" in the search bar.	A list of courses containing "Data Science" in their titles is displayed.	As expected	Pass
TC_002	Search for courses with partial match	The system contains courses	Search term: "Data"	1. Click on "Search Courses". 2. Enter "Data" in the search bar.	A list of courses containing "Data" in their titles is displayed.	As expected	Pass
TC_003	Search for a course that does not exist	The system contains courses	Search term: "Rocket Science"	1. Click on "Search Courses". 2. Enter "Rocket Science" in the search bar.	The system displays a message: "No matching courses found. Please try a different search term."	As expected	Pass

The execution of all test cases, as detailed in the preceding tables, resulted in a successful outcome. Both positive and negative test scenarios were thoroughly evaluated, and the actual results observed precisely matched the expected results for each case. This comprehensive testing process did not reveal any defects or functional issues within the tested areas of the application. Therefore, the functionalities under test have been confirmed to be operating as designed.

4.3 Automated Test Cases

5 Task Distribution

- **Project Initiation & Planning**
  - **Donia:**
    - Defining initial project scope and objectives
    - Leading all team meetings and discussions
  - **Ahmed:**
    - Identifying and outlining the functional requirements
    - Implementing Use Case Scenarios
  - **Nada:**
    - Identifying and outlining the non-functional requirements
    - Choosing the appropriate technologies (e.g., HTML, CSS, JavaScript, database)
- **Database Design & Implementation**
  - **Nada:**
    - Creating the Entity-Relationship Diagram (ERD)
    - Writing the SQL queries for data manipulation.
  - **Donia:**
    - Designing the relational database schema (tables, columns, data types)
    - Writing the SQL queries for data manipulation.
  - **Ahmed:**
    - Writing the SQL queries for data manipulation.
    - Populating database with dummy data (for testing purposes)
- **Frontend Development**
  - **Donia:**
    - Setting up the project structure for the frontend (file organization)
    - Designing the overall user interface (UI) and user experience (UX)
    - Implementing HTML structure for each page (home, login, signup, etc.)
    - Reviews for Designing the visual presentation for course cards and display
    - Integrating API calls to fetch data for course catalog, and user profile information
  - **Nada:**
    - Styling pages with CSS (layout, colors, responsiveness)
    - Implementing interactivity (buttons, forms)
    - Contributes to Implementing HTML structure for each page (home, login, signup, etc.)
- **Backend Development**
  - **Donia:**
    - Setting up the server-side environment
    - Implementing user authentication and login logic (login, signup, password reset)
    - Implementing payment processing functionality
    - Handling user profile management (updates, data handling)
    - Setting up APIs for data communication with the frontend
  - **Nada:**
    - Implementing user sessions to secure platform access
    - Handling the creation and management of courses, assignments, and learning materials

- **Testing and Validation**
  - **Nada:**
    - Developing test cases for various features
    - Documenting test procedures and results
  - **Donia and Ahmed:**
    - Identifying, documenting, and tracking bugs
    - Executing Test Cases
- **Documentation & Reporting**
  - **Ahmed:**
    - Editing and refining the report for clarity
  - **Nada:**
    - Writing the project report
  - **Donia:**
    - Creating the technical documentation for the project
- **Other**
  - **Donia, Nada, & Ahmed:**
    - General bug fixes and improvements across the project
    - Setting up the git repository
    - Prompt Engineering