

Online Learning Management System

Team ID: 8

- Jaydeep Mojidra: 1002186827
- Pranjal Kamboj: 1002157885
- Samyak Jain: 1002166068
- Dhruv Shah: 1002170901

Date: 11/10/2024

Table of Contents

- 1) Abstract
- 2) Project Initiation
- 3) Project Scope
- 4) Requirements Analysis
- 5) Project Management
- 6) Project Planning
- 7) Design and Architecture
- 8) Implementation
- 9) Testing
- 10) Maintenance
- 11) Conclusion
- 12) References

Part 1: Abstract

Brief Summary of the Project

- The project's main goal was to create a Learning Management System (LMS) specifically designed for managing courses at the university level. The LMS will give teachers the ability to design and oversee classes, homework, and tests while offering students a smooth enrollment, assignment, and grade viewing experience.
- Effectively oversee classes, homework, and tests; monitor student development; and give comments. For Students: Sign up for classes, turn in homework, take tests, and see grades and comments.
- Important functionality including course development, student enrollment, assignment administration, and grading will be implemented by the project. With an emphasis on performance and usability, the system will be scalable and guarantee a seamless experience for both teachers and students.

Key Outcomes

- **Course Management:** Teachers can easily create and administer courses, assignments, and tests.
- **Student Enrollment:** Students can seamlessly sign up for classes, ensuring a smooth registration process.
- **Assignment and Grading:** Both assignment submission and grading features are integrated, making it easier for instructors to grade and provide feedback.
- **Performance and Usability:** The system is optimized for high performance and scalability, ensuring a reliable experience as the user base grows.

Contributions

The project involved significant contributions to both front-end and back-end development, including designing a user-friendly interface, implementing JWT-based authentication, and integrating MySQL for database management. With these features, the LMS aims to provide a comprehensive platform for managing courses and supporting the educational process.

Each member's task

Student A: Jaydeep Mojidra (Frontend Development - HTML, CSS, JS)

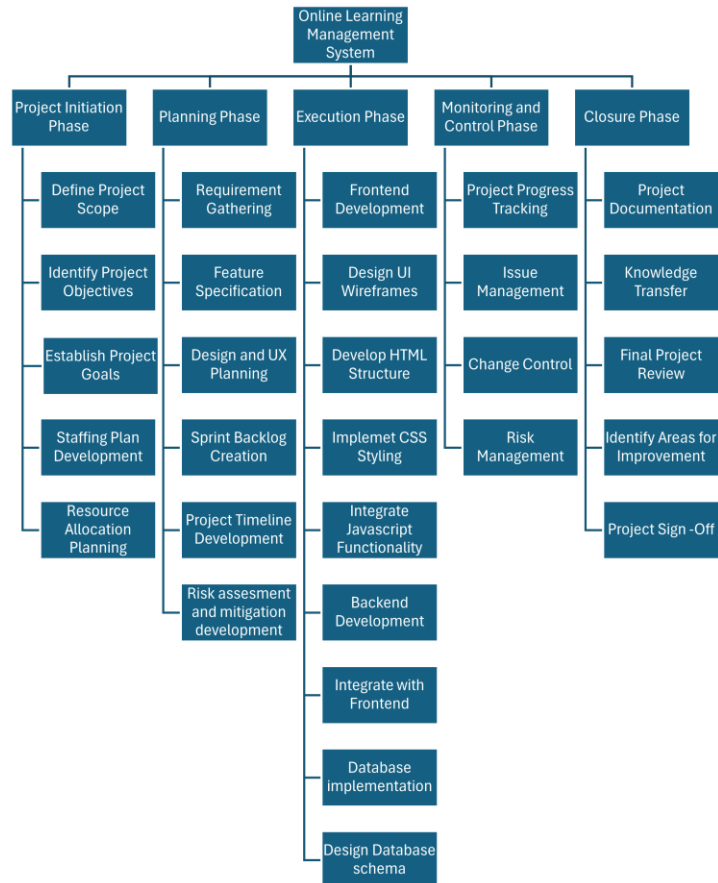
Student B: Pranjal Kamboj (Backend Development - JS, Node.js, React.js)

Student C: Samyak Jain (Backend Development - Database MySQL, Node.js)

Student D: Dhruv Shah (Frontend Development - HTML, CSS , JS)

Part 2: Project Initiation

Work Breakdown Structure:



Cost Estimation:

COCOMO Model (Constructive Cost Model)

Our choice stems from various factors, such as our team's small size and shared comprehension of the project's goals. Furthermore, our team members bring relevant experience from tackling similar challenges in the past. Thus, the Organic model suits our project dynamics, aiding in precise estimation and efficient management of resources and timelines.

$$E = a(KLOC)^b$$

YOUR BASIC COCOMO RESULTS!!								
MODE	"A" variable	"B" variable	"C" variable	"D" variable	KLOC	EFFORT, (in person/months)	DURATION, (in months)	STAFFING, (recommended)
organic	2.4	1.05	2.5	0.38	5.5	14.374477442610152	6.883748480599694	2.0881758656815257


Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort = $a * KLOC^b$, in person/months, with KLOC = lines of code, (in the thousands), and:

duration = $c * effort^d$, finally:

staffing = $effort / duration$

For further reading, see Boehm, "Software Engineering Economics", (81)



The model uses a formula to calculate the development effort based on the estimated size.

$$Effort \text{ (in person-months)} = a * (Size \text{ in KLOC})^b$$

Where 'a' and 'b' are coefficients specific to the type of software project being estimated. These coefficients can be derived from the table.

Function Point (FP) Estimation

- External Inputs (EI): Forms, buttons, or any input from users (e.g., login forms)
- External Outputs (EO): Reports, notifications, data shown to users.
- User Inquiries (EQ): Search functionality, data retrieval.
- Internal Logical Files (ILF): Internal databases or tables (courses, students).
- External Interface Files (EIF): External systems like external grading systems

For our project:

- EI: 5 (login, course creation, assignment submission, etc.)
- EO: 5 (grades, announcements, etc.)
- EQ: 3 (search courses, view grades)
- ILF: 3 (student, course, and assignment data)
- EIF: 5 (external APIs or external reports)

So, we get total 90 weighted measures and 0.65 factor so we get 59 Function points

Function Point Calculator
The Madison Utilities, Department of Computer Science, James Madison University

Direct Measure	Count	Weighted Measure		
Simple	Average	Complex		
External Inputs (EIs)	5	0	0	15
External Outputs (EOs)	5	0	0	20
External Inquiries (EQs)	3	0	0	9
Internal Logical Files (ILFs)	3	0	0	21
External Interface Files (EIFs)	5	0	0	25

Clear

Value Adjustment Factor	0	1	2	3	4	5
The system requires reliable backup and recovery.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Specialized data communications are required.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
There are distributed processing functions.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Performance is critical.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system runs in an existing, heavily utilized operational environment.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system requires on-line data entry.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The on-line data entry requires transactions over multiple screens/operations.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
ILFs are updated on-line.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The inputs, outputs, files or inquiries are complex.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The internal processing is complex.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The code is designed to be reusable.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Conversions /installation are included in the design.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system is designed for multiple installations in different organizations.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system is designed to facilitate change and ease of use.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Summary:
Total: 90
Factor: 0.65
FP: 59

Risk Identification

- There is a chance that additional feature requests or stakeholder changes could cause the project scope to grow, which could result in delivery delays or higher expenses.
- When frontend and backend components are integrated, problems might occur, particularly when integrating APIs or managing databases with Node.js, MySQL, and React. The development timetable can be delayed as a result.
- Unexpected events may render developers or important team members inaccessible, delaying project stages like development, testing, or deployment.
- If appropriate security mechanisms are not put in place, there is a chance of security breaches because the system manages sensitive data, such as user passwords and course data.
- The developed system may not meet the expectations of end-users in terms of usability, which could lead to low adoption rates and require additional modifications.

Risk Assessment

Result

	Risk	Likelihood (1-5)	Impact (1-5)	Risk Level
3	Security Vulnerabilities	4	5	20
0	Scope Creep	4	4	16
1	Technical Challenges	3	5	15
2	Resource Availability	3	3	9
4	Poor User Acceptance	2	3	6

Risk Mitigation Plan

1. Security Vulnerability

- Implement encryption, perform security audits, and apply patches regularly.

2. Scope Creep

- Set clear requirements, use change control processes, and prioritize new requests.

3. Project Delays

- Develop a detailed schedule, monitor progress, and allocate resources as needed.

4. Integration Challenges

- Plan integration early, conduct regular testing, and maintain clear documentation.

5. Data Loss

- Implement automated backups, validate data, and have a recovery plan in place.

Quality Assurance:

- **Functionality:** Ensure all features (e.g., user authentication, course management) work correctly.
- **Reliability:** System must handle heavy usage without failure.
- **Usability:** Easy and intuitive interface for users across devices.
- **Performance:** Fast response times, even with large user bases.
- **Security:** Protect user data and system access (e.g., encryption, secure authentication).
- **Scalability:** Ability to grow without performance issues.
- **Maintainability:** Modular, well-documented code for easy updates.
- **Compatibility:** Works on various devices and browsers.

Part 3: Project Scope

Developing a strong system for handling student assignment submissions and grading with an emphasis on dependable backend capabilities and user-friendly interfaces is the goal of this project.

Functionalities and Features:

- **User Roles:** Teachers and students have different roles with distinct permissions.
- **Assignments and Grading:** Teachers have the ability to examine and mark assignments that students turn in.
- **Grade Tracking:** To make it simpler to monitor academic progress, students can view graded contributions in this section.
- **Announcements and Notifications:** A static section for announcements of deadlines and general updates.
- **Management of Courses and Assignments:** Students have access to enrolled courses and the assignments that go with them.

Constraints:

- **Technology:** The project is built using a certain tech stack (React, Node.js, MySQL), which restricts the ability to switch between languages or frameworks.
- **Data Management:** The system needs to be carefully structured and maintained because it depends on predefined models and routes for data management, particularly for submissions and grading.

Target Audience:

- Originally created for educational institutions, this system is intended to help students, instructors, and administrators with academic monitoring, grading, and assignment submission.

Limitations:

- **Scalability:** The system may need additional optimization for high concurrent usage since it is currently designed for a moderate scale.
- **Real-time Notifications:** Since the announcement area is static, there are no dynamic, real-time notifications available.

Part 4: Requirement Analysis

Process for Collecting Requirements:

During the requirement-gathering process for the LMS project, we focused on key questions to align design and planning with user needs and technical requirements. These questions included:

1. User Roles and Access Levels:

- What roles will the system support (e.g., teacher, student, admin)?
- What permissions will each role need, such as course creation, enrollment, grading, and content access?

2. Course and Content Management:

- How should teachers create, organize, and manage courses, assignments, and tests?
- What features are necessary for students to view course content, submit assignments, and receive grades?

3. Assignment and Grading:

- What format(s) of assignments will the system support (e.g., file uploads, online submissions)?
- How should teachers review, grade, and provide feedback on assignments, and how will students view this feedback?

4. Communication and Notifications:

- What types of notifications are necessary (e.g., assignment deadlines, announcements, grades posted)?
- How should notifications be displayed or delivered to both teachers and students?

5. Enrollment and User Management:

- How will students enroll in courses, and what approval or registration steps are needed?
- What information will be required for user registration, and how will authentication and role assignment be handled?

6. Security and Data Privacy:

- What authentication methods are needed to ensure secure access for users?
- How will the system handle sensitive data like student grades and personal information?

7. Technical and Integration Requirements:

- What third-party libraries or tools will be required for file uploads, database management, and API handling?
- How will the frontend and backend components communicate, and what database structure is necessary for scalability?

8. User Experience and Interface Design:

- What elements are essential for a user-friendly interface for both teachers and students?
- How should the system handle dynamic content updates, such as grade postings and new announcements?

Analysis of related systems, and a careful examination of the documentation were all used to collect the requirements for this project. Below is an explanation of each technique:

- **Documentation Review:** Standard features and workflows that our system should incorporate were identified by looking at existing tools, such as LMS platforms (e.g., Canvas, Blackboard). We learned about common pain areas and efficient user processes as a result.

Functional Requirements:

- **User Authentication:** Roles (student, teacher, admin) should allow users to log in.
- **Role-Specific Dashboards:** Teachers and students ought to have their own dashboards with features tailored to their roles.
- **Submission Portal:** Assignments for every course in which a student is enrolled should be available for submission. These contributions ought to be accessible for teachers to see, download, and grade.
- **Grading System:** Each submission might have a grade assigned by the teacher. Each assignment's grade is visible to students.
- **Course and Assignment Management:** Instructors must be capable of developing and overseeing tasks for their classes. Students should check the due dates for impending assignments. Important updates can be provided in the announcements area, which is static.

Non-functional requirements include:

- **Usability:** Users should be able to easily access their assignments, submissions, and grades thanks to an intuitive interface.
- **Performance:** Even when people are using the system simultaneously, it should react to their activities quickly and without apparent lag.
- **Scalability:** Without requiring a major redesign, the system should be able to accommodate the addition of new features and a larger user base.
- **Security:** Only authorized roles should be able to access user data, particularly grades and submissions, which should be safely preserved.
- **Compatibility:** The system ought to function on mobile devices and in the majority of web browsers.

Use Cases

- **User Login:** Users log in to access LMS features based on their role.
- **Course Creation:** Teachers create and manage courses, adding relevant details.
- **Course Enrollment:** Students enroll in courses to access materials and assignments.
- **Assignment Submission:** Students submit assignments for teacher review and grading.
- **Assignment Grading:** Teachers grade assignments and provide feedback.
- **Viewing Grades:** Students view grades and feedback on their submissions.
- **Announcements & Notifications:** Teachers post announcements; students receive notifications for updates and deadlines.

Part 5: Project Management

Sprint Backlog (Jira)

Jira was used as the management tool for our project, Following are related screenshots:

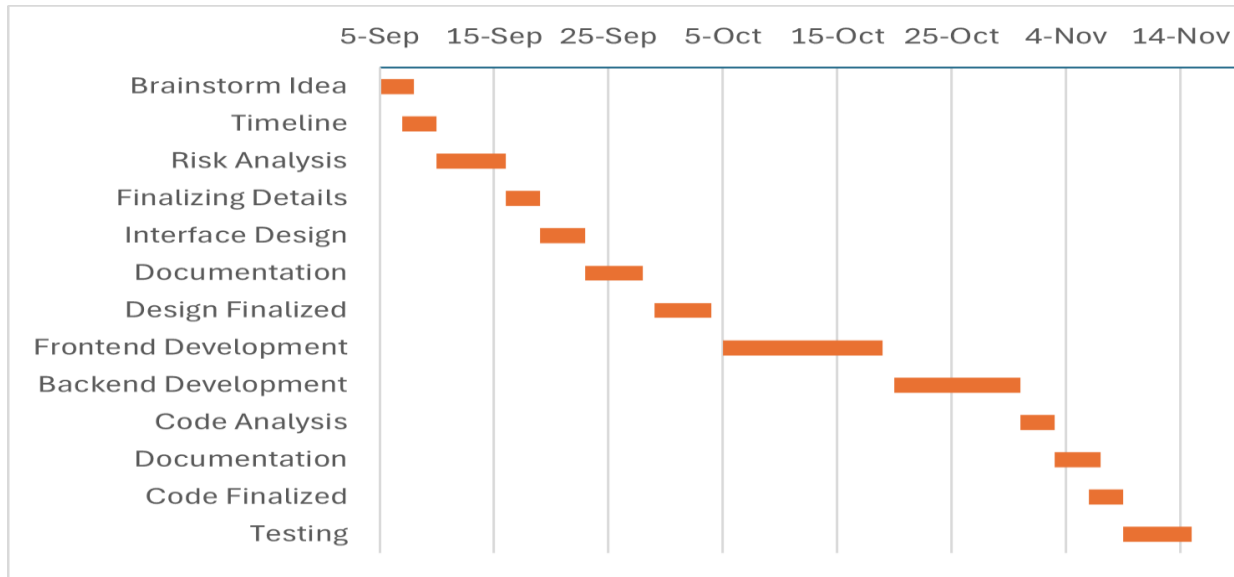
The screenshot displays the Jira Sprint Backlog interface. At the top, there is a search bar, user avatars, and a dropdown menu labeled 'Epic'. On the right, there are links for 'Insights' and 'View settings'. The backlog is organized into four sprints, each with a title, dates, and a count of issues.

- OLMSS Sprint 1** (5 Sep – 19 Sep, 5 issues):
 - Core Setup and User Authentication
 - OLMSS-1: Initialize the React.js project with basic Material-UI components. (Status: TO DO)
 - OLMSS-2: Initialize Node.js and Express.js, set up basic server configuration. (Status: TO DO)
 - OLMSS-3: Design the database schema for users (teachers, students), courses, assignments. (Status: TO DO)
 - OLMSS-4: Implement login and registration features for teachers and students using JWT. (Status: TO DO)
 - OLMSS-5: Create test cases to verify user login, registration, and role-based access. (Status: TO DO)
- OLMSS Sprint 2** (20 Sep – 4 Oct, 6 issues):
 - Course and Enrollment Management
 - OLMSS-6: Implement API for teachers to create and edit courses. (Status: TO DO)
 - OLMSS-7: Develop UI for displaying course list to teachers and students. (Status: TO DO)
 - OLMSS-8: Create an API for students to enroll in courses. (Status: TO DO)
 - OLMSS-9: Build the frontend UI for students to view available courses and enroll. (Status: TO DO)
 - OLMSS-10: Connect course creation and enrollment with MySQL using Sequelize. (Status: TO DO)
 - OLMSS-11: Test course creation and enrollment APIs with Postman. (Status: TO DO)
- OLMSS Sprint 3** (4 Oct – 1 Nov, 6 issues):
 - Assignment and Submission Management
 - OLMSS-12: Implement API for teachers to create and manage assignments for each course. (Status: TO DO)
 - OLMSS-13: Build UI for teachers to create assignments and students to view and submit them. (Status: TO DO)
 - OLMSS-14: Enable file upload functionality for students to submit assignments. (Status: TO DO)
 - OLMSS-15: Implement functionality for teachers to grade submitted assignments. (Status: TO DO)
 - OLMSS-16: Store submission data (files, grades) in the MySQL database. (Status: TO DO)
 - OLMSS-17: Test assignment creation, submission, and grading using Postman. (Status: TO DO)
- OLMSS Sprint 4** (4 Nov – 11 Nov, 4 issues):
 - Grading, Comments, and Final Touches
 - OLMSS-18: Develop UI for students to view their grades and comments on assignments. (Status: TO DO)
 - OLMSS-19: Allow teachers to add comments to assignments after grading. (Status: TO DO)
 - OLMSS-20: Optimize the API and frontend for faster load times, especially on course/assignment pages. (Status: TO DO)
 - OLMSS-21: Perform user testing sessions with a small group of teachers and students. (Status: TO DO)

At the bottom of each sprint section, there is a '+ Create issue' button.

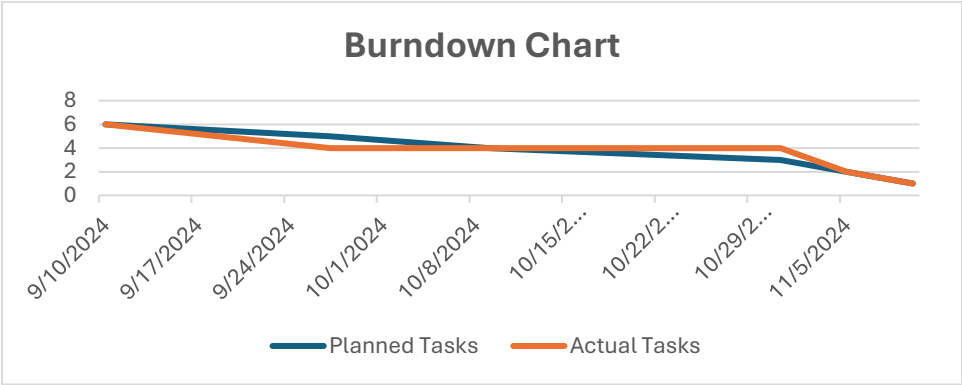
Part 6: Project Planning

Gantt chart:



Task	Start Date	End Date	Duration
Brainstorm Idea	5-Sep	7-Sep	3
Timeline	7-Sep	9-Sep	3
Risk Analysis	10-Sep	15-Sep	6
Finalizing Details	16-Sep	18-Sep	3
Interface Design	19-Sep	22-Sep	4
Documentation	23-Sep	27-Sep	5
Design Finalized	29-Sep	2-Oct	5
Frontend Development	5-Oct	18-Oct	14
Backend Development	20-Oct	30-Oct	11
Code Analysis	31-Oct	2-Nov	3
Documentation	3-Nov	6-Nov	4
Code Finalized	6-Nov	8-Nov	3
Testing	9-Nov	14-Nov	6

Burndown Chart



Days	Dates	Planned Tasks	Actual Tasks
1	9/10/2024	6	6
2	9/27/2024	5	4
3	10/9/2024	4	4
4	10/31/2024	3	4
5	11/5/2024	2	2
6	11/10/2024	1	1

Reasons for deviation from the planned curve:

- Task Complexity: Some tasks, like backend integration of course management or grading, may require more time than initially anticipated due to unforeseen technical challenges, such as handling dynamic UI updates or ensuring smooth data flow between components.

Table for Each Person's Tasks and Responsibilities

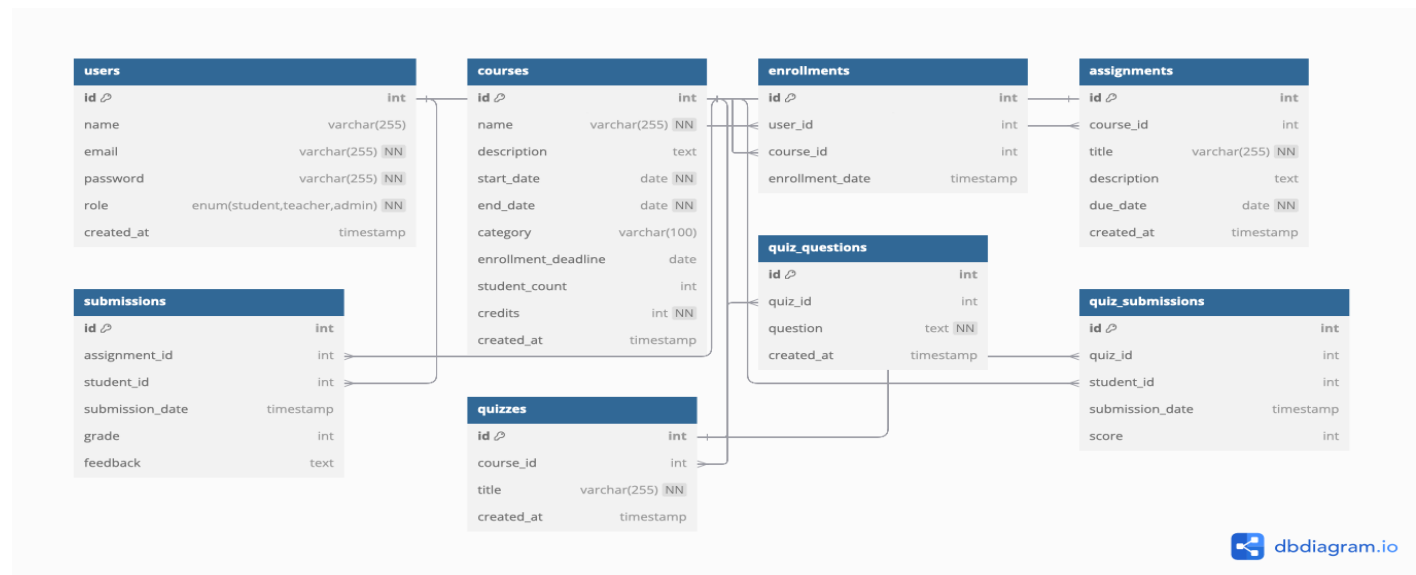
Tasks	Assigned Task
Gathering Requirements	Jaydeep, Pranjal, Samyak, Dhruv
User Interface Design	Samyak, Dhruv
Implementation	Jaydeep, Pranjal
Testing	Jaydeep, Pranjal, Samyak, Dhruv
Debugging	Jaydeep, Pranjal, Samyak, Dhruv
Documentation	Jaydeep, Pranjal, Samyak, Dhruv

Part 7: Design and Architecture

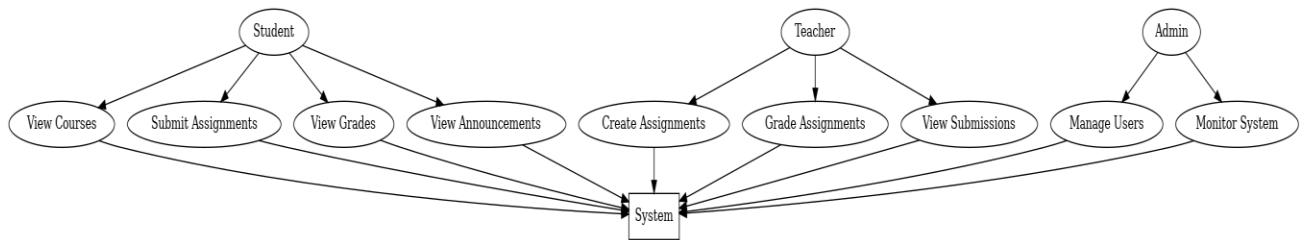
Our system architecture follows a typical three-tier architecture:

- **Presentation Layer (Frontend):** This is built using React, handling the user interface and interactions. It communicates with the backend through RESTful API requests to retrieve and display data, and to send user inputs (e.g., submissions, grades) to be processed by the backend.
- **Application Layer (Backend):** Implemented with Node.js and Express, this layer contains the business logic, handles API endpoints, and manages communication between the frontend and the database.
- **Data Layer (Database):** A MySQL database is used to store persistent data, such as user profiles, courses, assignments, submissions, grades, and announcements. The backend interacts with the database via Sequelize ORM, which handles data manipulation and retrieval.

Entity Relationship Diagram:



UML Diagram:



Part 8: Implementation

Development Environment and tools:

To facilitate both frontend and backend development, our project was created utilizing a contemporary web stack. The main surroundings and resources consist of:

- Frontend: React was used to create a dynamic and responsive user interface.
- Backend: Developed using Express and Node.js to manage server-side routing and logic.
- Database: To provide organized storage and effective querying, MySQL was selected to handle data storage.

Other essential tools include:

- Visual Studio Code is an IDE for effective code development and debugging, among other crucial features.
- Postman: Used for API endpoint testing and verification.
- MySQL Workbench: For managing and querying databases.

Major components:

- User Management: Manages role assignment (teacher, student), user registration, and authentication. The ability to create, enroll, and examine course details is provided via course management.
- Assignment Submission and Grading: This feature lets teachers grade assignments that students turn in.
- Announcement and Notifications: Show both course-specific notifications and static announcements.

Challenges:

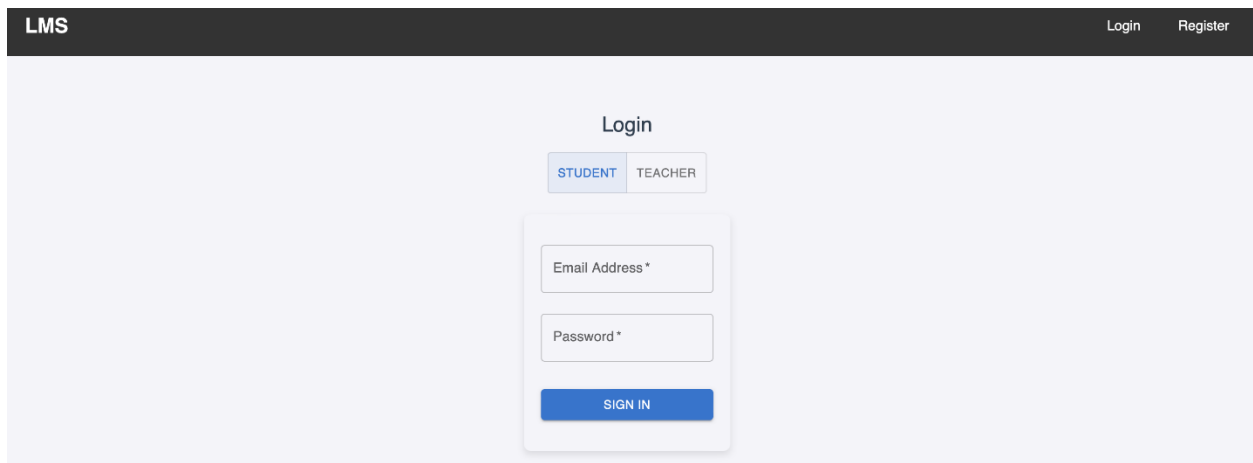
- Backend Integration: Managing intricate pathways and effectively integrating them, particularly when incorporating functions like announcements and grading.
- Dynamic UI Updates: Making sure that dynamic data, such as grades and courses, was regularly and appropriately shown in the frontend.
- Error Handling: Configuring reliable error handling across components to give users insightful messages.

Third party libraries:

- Multer: For handling file uploads on the server, used in the assignment submission module.
- Axios: For making HTTP requests from the frontend to the backend API.
- Material-UI: For creating responsive, aesthetically pleasing UI components.
- Sequelize: Used for database ORM, facilitating interaction with the MySQL database.

Modules:

Login:



The screenshot shows a web application interface for an LMS. At the top, there is a dark header bar with the text "LMS" on the left and "Login" and "Register" links on the right. Below the header, the main content area is light gray. In the center, there is a white box with a light gray border and a subtle shadow. Inside this box, the word "Login" is centered at the top. Below it, there are two tabs: "STUDENT" (which is highlighted with a blue background) and "TEACHER". Under the tabs, there are two input fields: "Email Address *" and "Password *". At the bottom of the box, there is a blue button with the text "SIGN IN" in white capital letters.

Assignment Grades (Viewed by student):

LMS

Student Home

Enrollment

Submission

Logout

Welcome Back, Student!

Current Courses

Upcoming Assignments

Select Course

No assignments for this course.

Recent Grades

Machine Learning: Assignment 1

Grade: A

Web Development: Project 1

Grade: B+

Machine Learning: Quiz 1

Grade: A-

Course Creation (For Teachers):

LMS

Teacher Home

Course Creation

Create an Assignment

Grading

Logout

Create a New Course

Course Name *

Course Description *

Start Date *

10/23/2024

End Date *

10/23/2024

Category *

Enrollment Deadline

10/23/2024

Credits *

3

Student Count *

0

CREATE COURSE

Assignment Creation:

LMS

Teacher Home

Course Creation

Create an Assignment

Grading

Logout

Create a New Assignment

Assignment Title *

Assignment Description *

Due Date *
10/23/2024

Select Course * ▾

CREATE ASSIGNMENT

Part 9: Testing

In our Learning Management System (LMS) project, we utilize the Jest automation testing framework to ensure the reliability and performance of both the front-end and backend components. Jest allows us to create and run tests for:

- **User Interface Components:** We validate React components, such as user login, course enrollment, assignment submission, and grade viewing, to ensure they function correctly, and handle user interactions as expected.
- **Backend API Integration:** Jest enables us to mock API responses, allowing for isolated testing of frontend components. This approach ensures reliable data handling between the frontend and backend, such as verifying role-based access and assignment grading processes.
- **Dynamic Updates and Real-Time Data Handling:** Using Jest's asynchronous testing, we test dynamic data updates for grades and announcements, providing users with a seamless experience as information refreshes in real-time.
- **Error Handling and Edge Cases:** We simulate common error scenarios, like failed enrollments or file upload issues, to confirm that meaningful error messages are displayed, ensuring robust error handling across the system.

This testing framework strengthens our LMS by automating checks across all major functionalities, enhancing system reliability and user satisfaction.

Unit Testing

Test Case ID	Test Description	Input	Expected Result	Status
UT001	Test user login functionality	Valid email and password	Login successful, redirect to dashboard	Pass
UT002	Test invalid login credentials	Invalid email or password	Error message "Invalid credentials"	Pass
UT003	Test course creation functionality	Valid course data	Course created, displayed on teacher's page	Pass
UT004	Test assignment submission by student	Valid submission	Assignment submitted successfully	Pass
UT005	Test course enrollment by student	Valid course selection	Enrollment successful, student listed	Pass

Integration Testing

Test Description	Input	Expected Result	Status
Test user registration and login flow	Valid registration and login data	User registers, logs in, and is redirected	Pass
Test course creation and display on teacher's page	Valid course data	Course appears on the teacher's dashboard	Pass
Test assignment submission and grading integration	Submit assignment, grade it	Assignment is graded, marks are saved	Pass
Test student enrollment and course access	Enroll in course, view course	Student is able to view course content	Pass

System Testing

Test Case ID	Test Description	Input	Expected Result	Status
ST001	Test full LMS workflow for teacher	Create course, create assignment	Teacher can manage course, assignments	Pass
ST002	Test full LMS workflow for student	Enroll, submit assignment	Student can enroll, submit assignments	Pass
ST003	Test role-based access control	Access teacher-only features	Only teacher can create courses, assignments	Pass
ST004	Test JWT token expiration and refresh	Access with expired token	User is logged out when token expires	Pass

User Acceptance Testing

Test Case ID	Test Description	Input	Expected Result	Status
UAT001	Verify ease of course creation for teachers	Teacher inputs course data	Teacher can easily create and view courses	Pass
UAT002	Verify smooth assignment submission by students	Submit assignment	Student is able to upload and submit files	Pass
UAT003	Verify course enrollment process for students	Select course	Enrollment successful, course displayed	Pass
UAT004	Verify intuitive navigation for teachers and students	Navigate through LMS	Users can easily navigate through LMS	Pass

Automation Testing Results:

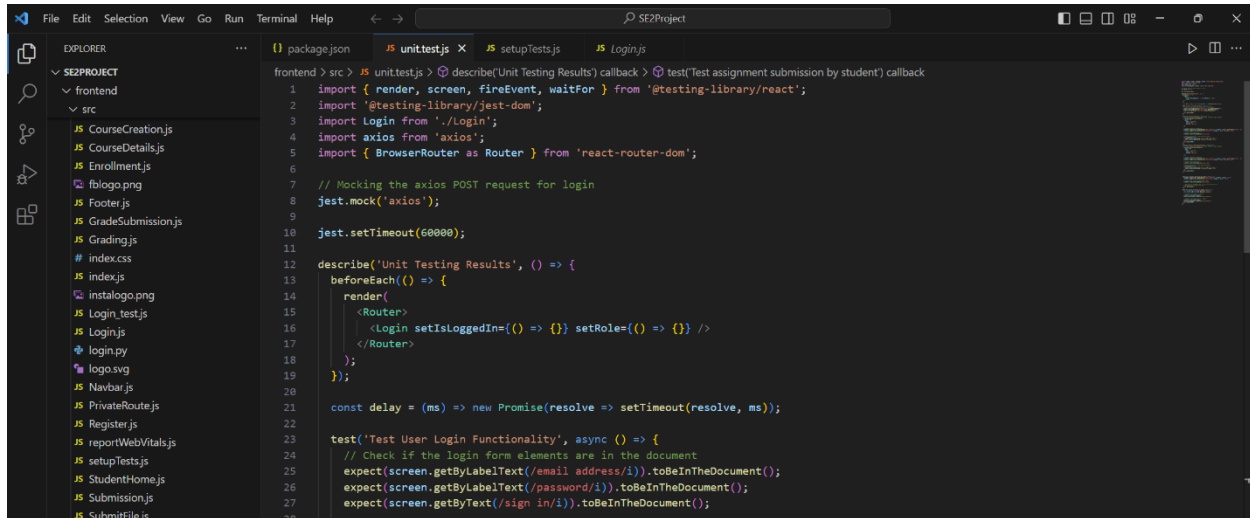
```
C:\WINDOWS\system32\cmd. x + v
    at actImplementation (node_modules/react/cjs/react.develo
    at node_modules/@testing-library/react/dist/act-compat.js
    at Object.eventWrapper (node_modules/@testing-library/rea
    at fireEvent (node_modules/@testing-library/dom/dist/even
    at Function.fireEvent.<computed> [as click] (node_modules
    at Function.click (node_modules/@testing-library/react/di
    at Object.<anonymous> (src/unit.test.js:91:15)

PASS src/unit.test.js (178.251 s)
Unit Testing Results
  ✓ Test User Login Functionality (20140 ms)
  ✓ Test Invalid Credentials (20134 ms)
  ✓ Test Course Creation Functionality (40066 ms)
  ✓ Test assignment submission by student (50074 ms)
  ✓ Test course enrollment by student (40046 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        180.999 s
Ran all test suites.

Watch Usage
  > Press f to run only failed tests.
  > Press o to only run tests related to changed files.
  > Press q to quit watch mode.
  > Press p to filter by a filename regex pattern.
  > Press t to filter by a test name regex pattern.
  > Press Enter to trigger a test run.
```

Testing code snippets:



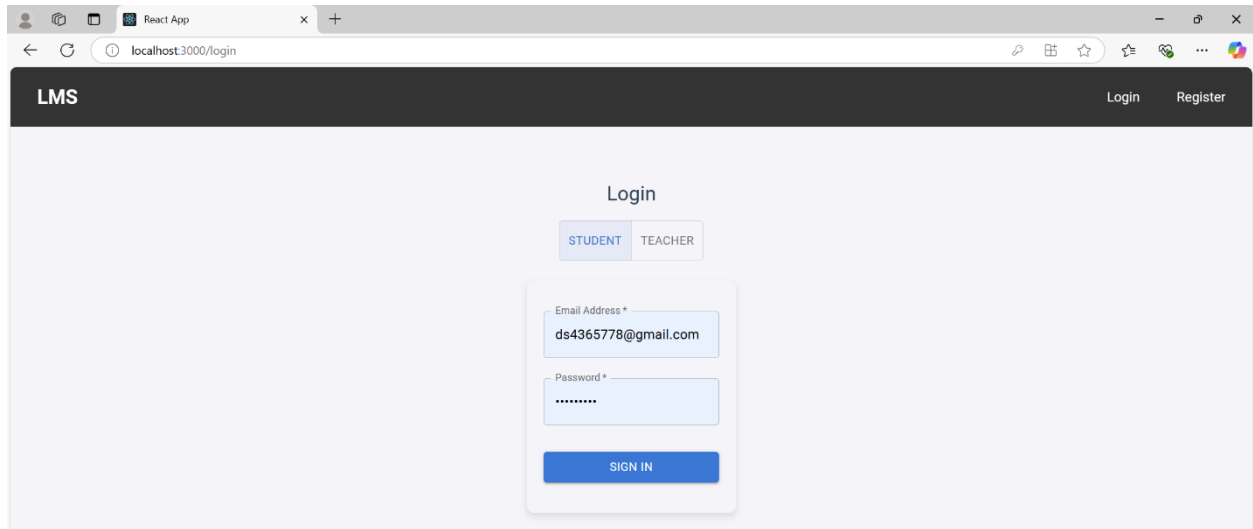
The screenshot shows a VS Code editor with a project named 'SE2Project'. The Explorer sidebar on the left shows a file tree with 'frontend' and 'src' folders. The 'src' folder contains files like 'CourseCreation.js', 'CourseDetails.js', 'Enrollment.js', 'fblogo.png', 'Footer.js', 'GradeSubmission.js', 'Grading.js', 'index.css', 'index.js', 'installlogo.png', 'Login_test.js', 'Login.js', 'login.py', 'logo.svg', 'Navbar.js', 'PrivateRoute.js', 'Register.js', 'reportWebVitals.js', 'setupTests.js', 'StudentHome.js', 'Submission.js', and 'SubmitFile.js'. The main editor area shows the 'unitTests.js' file with the following code:

```
1  import { render, screen, fireEvent, waitFor } from '@testing-library/react';
2  import '@testing-library/jest-dom';
3  import Login from './Login';
4  import axios from 'axios';
5  import { BrowserRouter as Router } from 'react-router-dom';
6
7  // Mocking the axios POST request for login
8  jest.mock('axios');
9
10 jest.setTimeout(60000);
11
12 describe('Unit Testing Results', () => {
13   beforeEach(() => {
14     render(
15       <Router>
16         <Login setIsLoggedIn={() => {}} setRole={() => {}} />
17       </Router>
18     );
19   });
20
21   const delay = (ms) => new Promise(resolve => setTimeout(resolve, ms));
22
23   test('Test User Login Functionality', async () => {
24     // Check if the login form elements are in the document
25     expect(screen.getByLabelText(/email address/i)).toBeInTheDocument();
26     expect(screen.getByLabelText(/password/i)).toBeInTheDocument();
27     expect(screen.getByText(/sign in/i)).toBeInTheDocument();
28   });
29 });
```

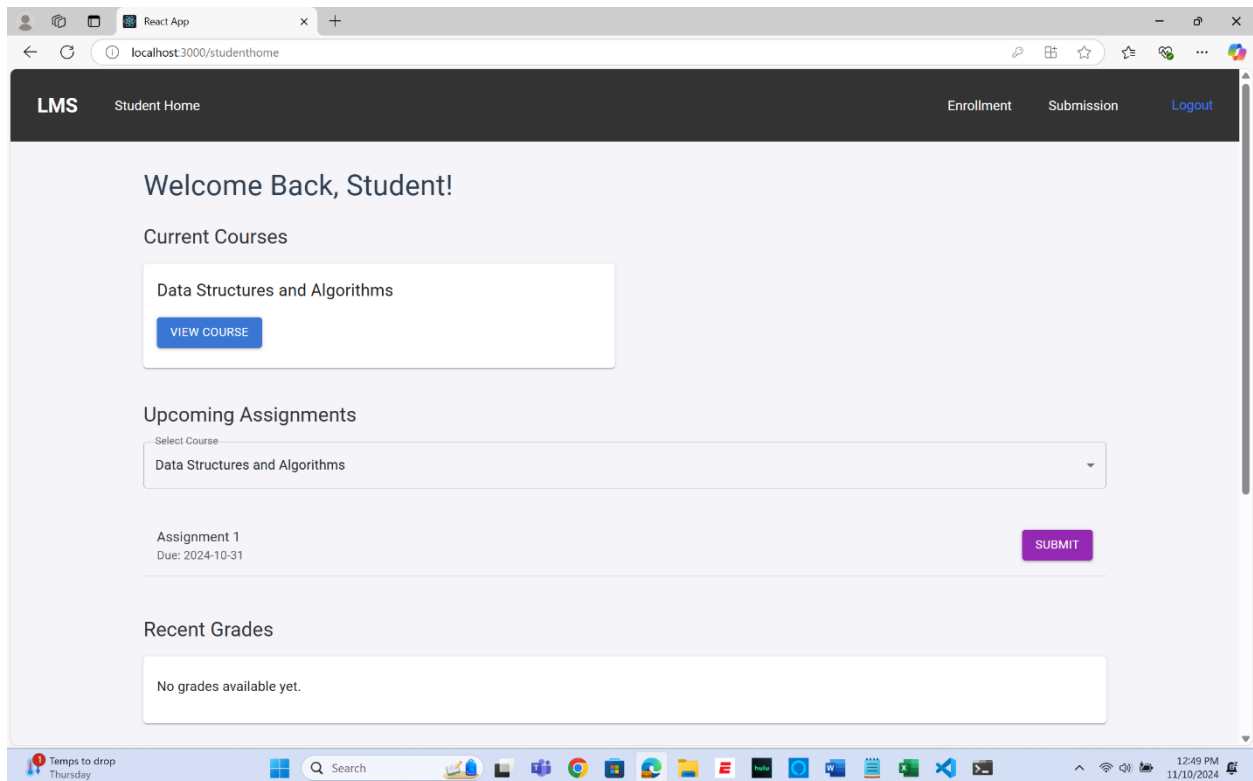
```
C:\Users\ShahDhruvVikram\Desktop\SE2\SE2Project\backend>node index.js
Connected to MySQL database.
Database synced successfully.
Database synced successfully.
Server is running on port 5001
Received course data: {
  name: 'Software Engineering 2',
  description: 'SE2',
  start_date: '2024-08-10',
  end_date: '2024-11-25',
  category: 'SE2',
  enrollment_deadline: '2024-11-28',
  status: 'upcoming',
  credits: 3,
  student_count: 0
}
```

Testing Results:

Test Case 1: Verify Login with Valid Credentials

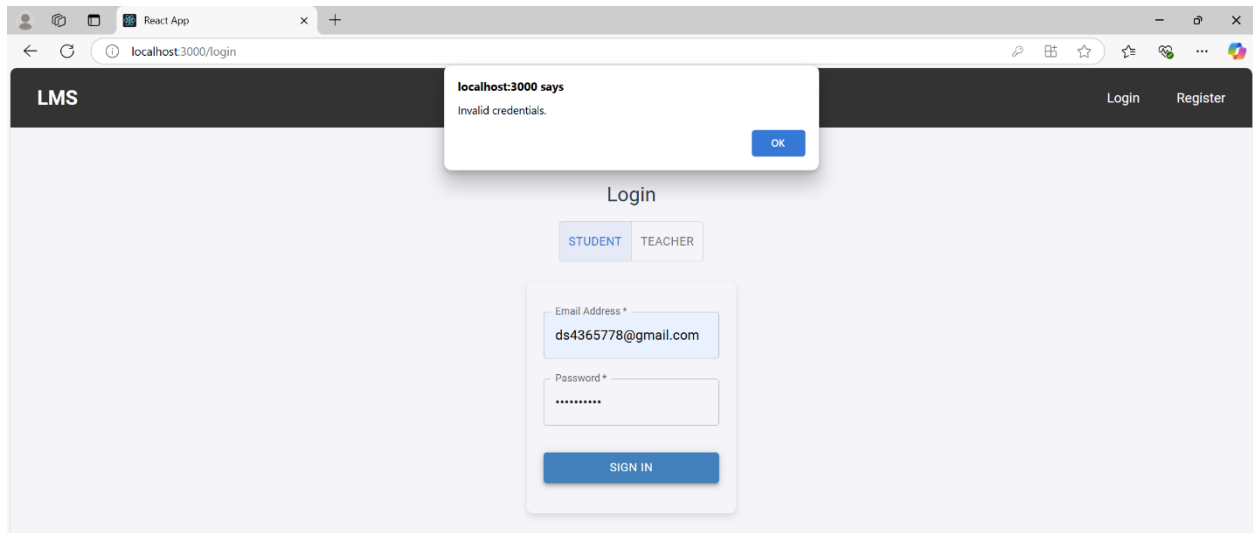


A screenshot of a web browser showing the LMS Login page. The browser's address bar displays 'localhost:3000/login'. The page has a dark header with 'LMS' on the left and 'Login' and 'Register' links on the right. The main content area is light gray and contains a 'Login' section. At the top of this section are two tabs: 'STUDENT' (selected) and 'TEACHER'. Below the tabs is a form with two input fields: 'Email Address *' containing 'ds4365778@gmail.com' and 'Password *' with masked characters. A blue 'SIGN IN' button is positioned below the password field.

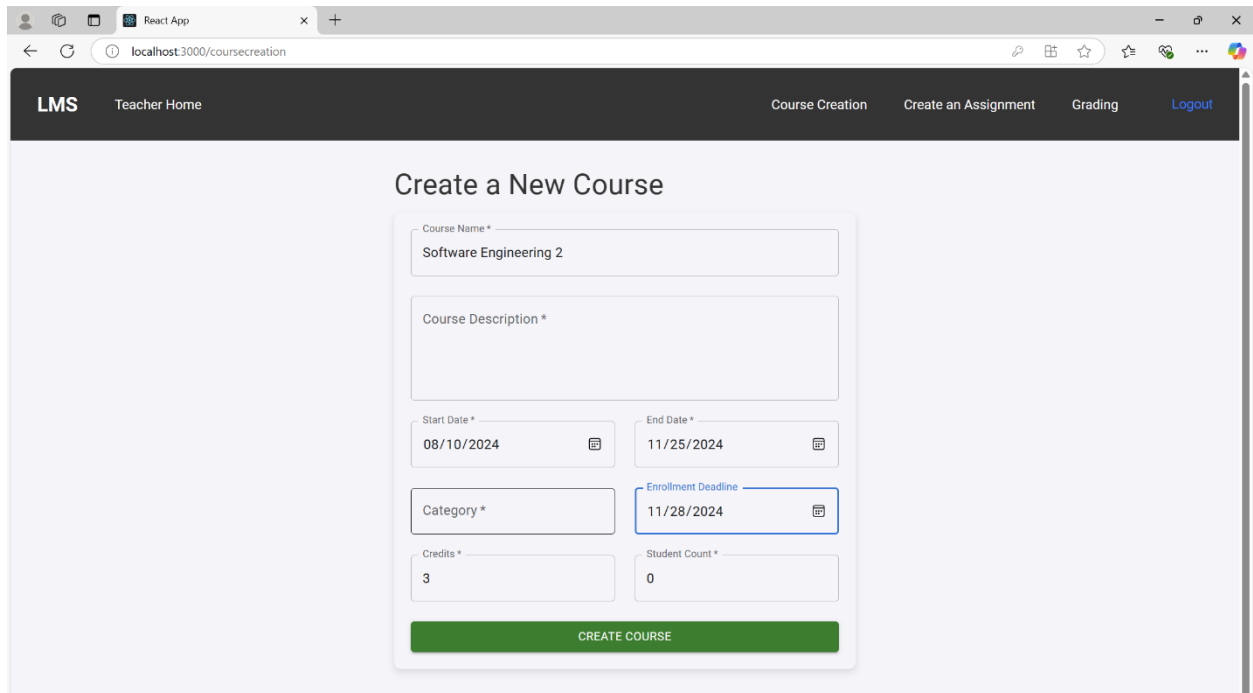


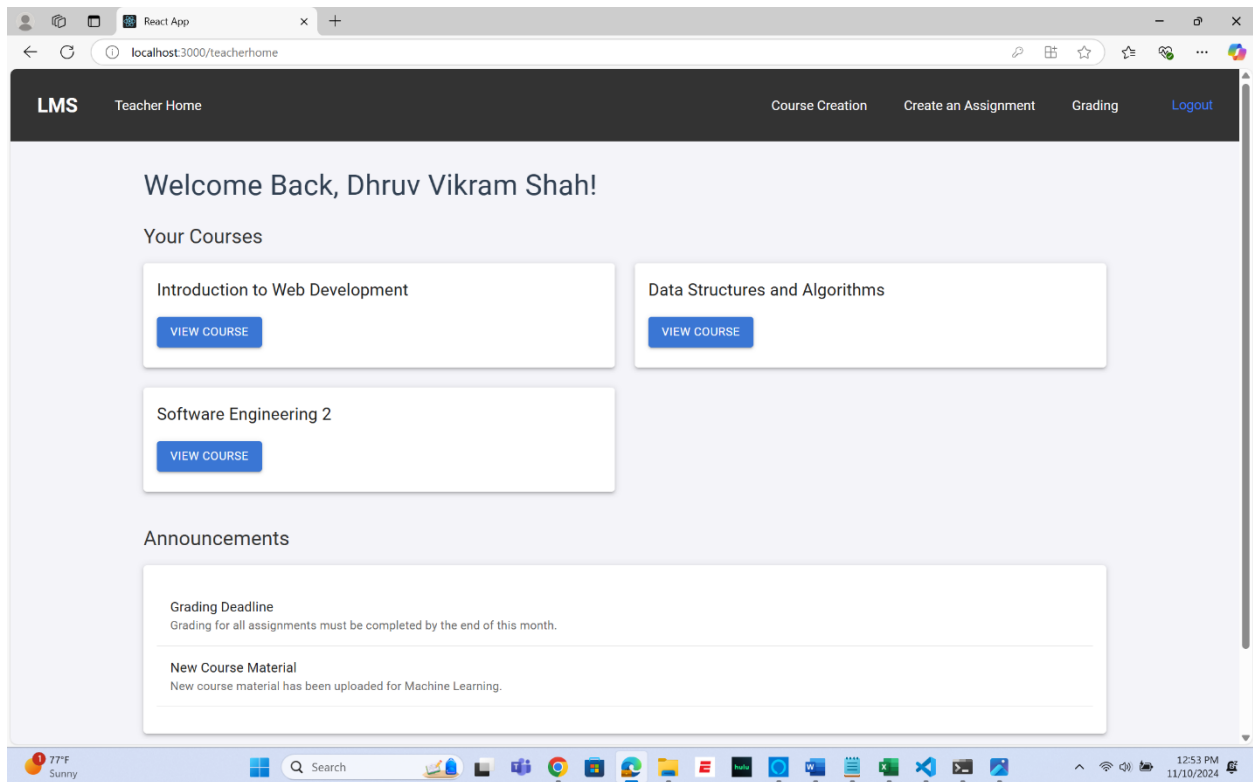
A screenshot of the LMS Student Home page. The browser's address bar shows 'localhost:3000/studenthome'. The dark header includes 'LMS' and 'Student Home' on the left, and 'Enrollment', 'Submission', and 'Logout' links on the right. The main content area is light gray and features several sections: 'Welcome Back, Student!', 'Current Courses' with a card for 'Data Structures and Algorithms' and a 'VIEW COURSE' button, 'Upcoming Assignments' with a dropdown menu set to 'Data Structures and Algorithms', 'Assignment 1' with a due date of '2024-10-31' and a 'SUBMIT' button, and 'Recent Grades' with a message 'No grades available yet.' The Windows taskbar is visible at the bottom, showing the time as 12:49 PM on 11/10/2024.

Test Case 2: Verify with Invalid credentials

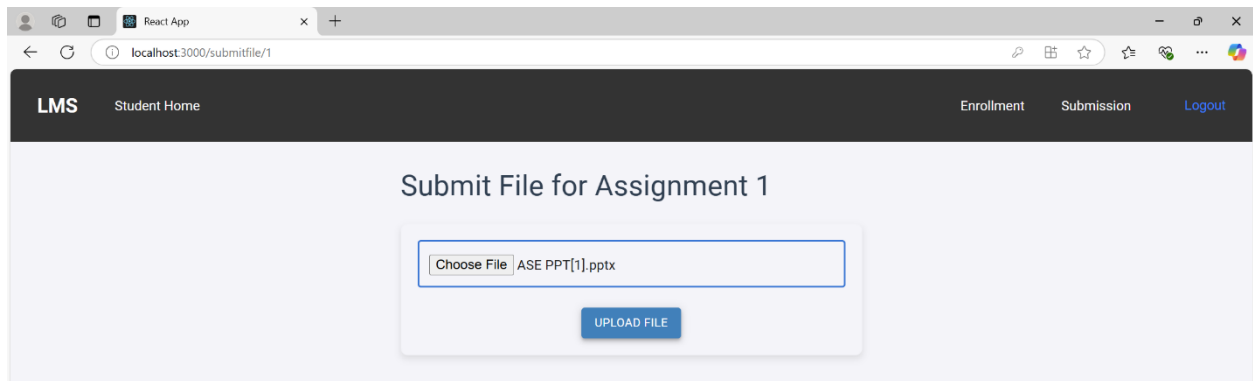


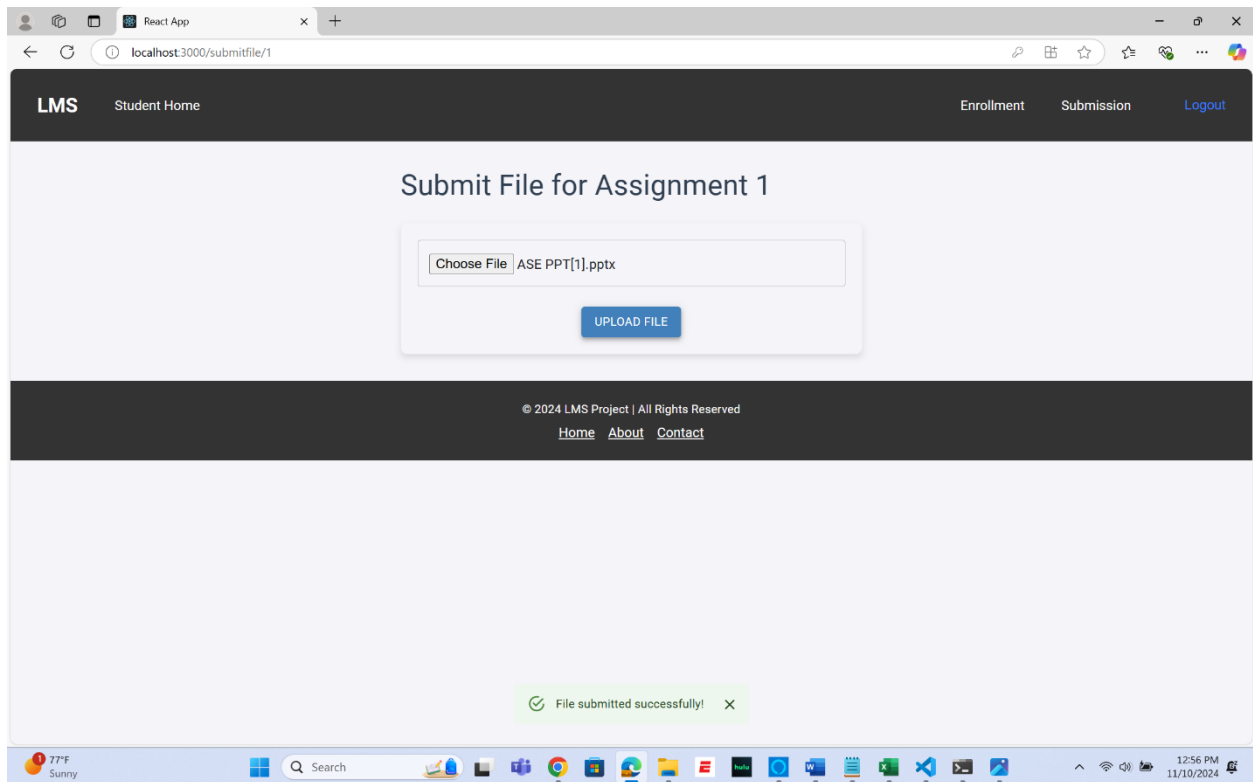
Test Case 3: Test Course Creation Functionality



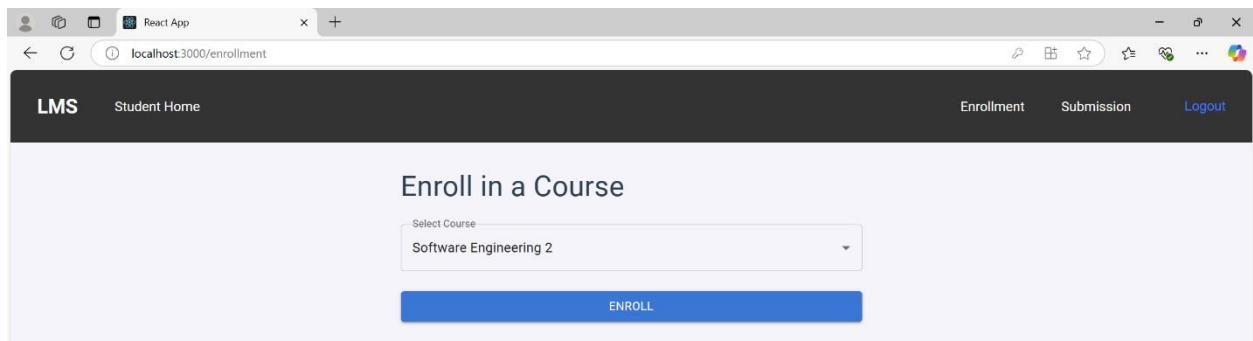


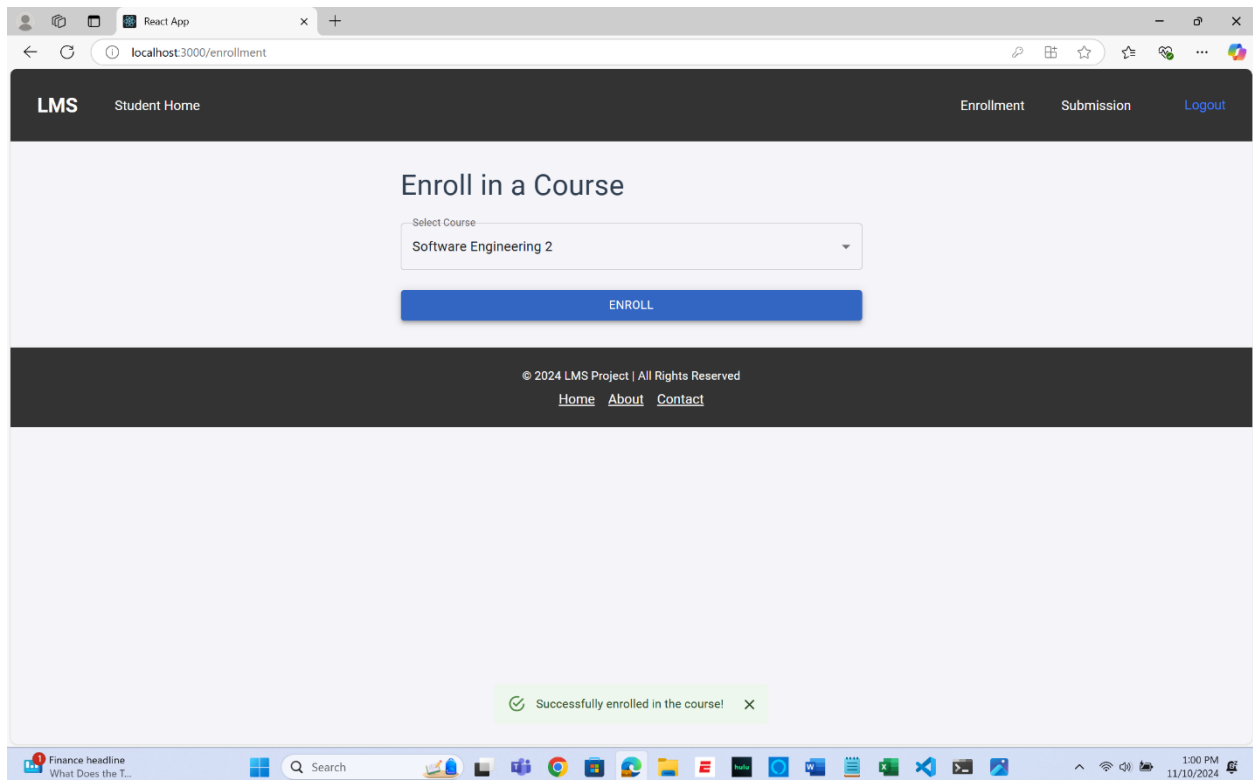
Test Case 4: Test Assignment Submission by student





Test Case 5: Test Course Enrollment by student





Bugs/Issues

While running the test cases, a common issue encountered was timeouts, which were primarily caused by slow page rendering.

To solve this, we added a delay time so that it can be properly loaded and also gave an estimated time also for that

```
jest.setTimeout(60000);
```

```
});

const delay = (ms) => new Promise(resolve => setTimeout(resolve, ms));

test('Test User Login Functionality', async () => {
  // Check if the login form elements are in the document
  expect(screen.getByLabelText(/email address/i)).toBeInTheDocument();
  expect(screen.getByLabelText(/password/i)).toBeInTheDocument();
  expect(screen.getByText(/sign in/i)).toBeInTheDocument();

  await delay(20000);
});
```


Part 10: Maintenance

Updates, Patches, and Bug Fix Management:

We've created a maintenance plan that emphasizes frequent updates, bug fixes, and scalability to guarantee the project stays operational and scalable.

- **Version Control:** Version control will still be handled via GitHub. Branches and pull requests will be used to handle new features, bug fixes, and changes, enabling code review and testing prior to integration.
- **Frequent Code Reviews:** To find possible refactoring opportunities and out-of-date dependencies, the team will examine the codebase on a regular basis. This keeps the code neat and effective.
- **Planned upgrades:** To maintain dependencies (such third-party libraries) up to date, we will plan recurring upgrades. This will guarantee compatibility with more recent technologies and reduce security flaws.

Scalability and Future-Proofing Considerations:

- **Modular Architecture:** By structuring the codebase in a modular way, it is simpler to grow the program and add new features without requiring significant rewrites.
- **Database Optimization:** To guarantee effective data access and manage increases in the volume of users and contributions, indexing, optimized queries, and frequent database health checks are planned.
- **Containerization:** Docker may be used in the future to containerize the application, which would simplify deployment and provide consistent performance in various environments.

Type of Maintenance	Description
Corrective Maintenance	Fixing issues reported by users, such as problems with course creation, enrollment, assignment submission, or grading. Resolving sign-in issues and ensuring smooth access for students and teachers. Addressing problems with dynamic data updates like grades or announcements.
Adaptive Maintenance	Modifying the system to accommodate changes in technology or educational requirements. Updating the system for improved mobile and tablet compatibility. Adding new course categories, features, or modules to meet evolving academic needs.
Preventive Maintenance	Regularly checking for issues that may affect data integrity, such as broken links or inaccessible resources. Updating security protocols and authentication methods. Periodically backing up course content, grades, and user data to secure information.
Perfective Maintenance	Enhancing the user interface for better usability, such as improving the enrollment process or assignment submission experience. Adding features like progress tracking or enabling communication within courses. Optimizing performance for pages that handle large data sets, such as grading and course management sections.

Part 11: Conclusion

Project Outcome and Achievements:

- **Implementation of Features:** The system has essential features like grading, assignment submission, user management, course enrollment, and a simplified peer review procedure. The main goals of the project are successfully met by this strong feature set, which benefits both teachers and students.
- **Scalable Architecture:** The system is ready for future improvements and scalability thanks to a React-based frontend and a modular, efficient backend.
- **User-Friendly Interface:** The design prioritizes usability, offering end users an enhanced experience through accessible features, simple forms, and straightforward navigation.

Lessons Learned:

During the project, a number of significant revelations surfaced that can enhance subsequent development procedures:

- **The Value of Requirement Refinement:** It is essential to start with precise, comprehensive needs. The necessity of careful planning and validation to prevent mid-development pivots was brought to light by early requirements modifications.
- **Effective Error Handling:** Complex faults were avoided later by addressing frontend and backend communication problems (such missing data fields or incorrect API configurations) as soon as possible. Improved error handling and reporting led to better troubleshooting.
- **Dependency Management:** Stability required regular upgrades and reliable version control of dependencies. It was really helpful to learn how to control and debug third-party dependencies.

Areas for improvement:

Although the project achieved its main objectives, there are still several things we could do better:

- **Improved Testing:** Some of the runtime issues that are seen might be avoided with more thorough testing, which includes load, integration, and unit tests.
- **Better Documentation:** Although we kept up-to-date documentation, new developers might find it easier to onboard if there was more information on the architecture, configuration, and functions of each component.
- **Advanced Features:** The system's usefulness might be further increased by incorporating advanced features like analytics, automatic grading, and AI-driven recommendations for course enhancements.

Future Work:

- **Integration with External Tools and Platforms**
Integrating with popular learning tools such as Google Classroom, Zoom, or Microsoft Teams to enhance the collaborative and interactive experience.
- **AI-Powered Recommendations**
Implementing AI features to recommend courses, assignments, or resources based on student preferences, performance, and learning patterns.
- **Mobile Application Development**
Developing native mobile apps for iOS and Android to provide a better mobile experience for students and teachers, including offline access to course materials.
- **Advanced Analytics and Reporting**
Incorporating more advanced analytics features to help teachers track student progress, identify struggling students, and generate detailed reports for course management.
- **Gamification**
Adding gamification elements like badges, leaderboards, and achievements to engage students and encourage participation.
- **Enhanced Communication Features**
Introducing real-time chat or forums within the system to allow students and teachers to collaborate more efficiently.
- **Multilingual Support**
Extending the platform's reach by supporting multiple languages, making it accessible to a more diverse user base.
- **Cloud-based Infrastructure**
Migrating to a cloud-based infrastructure to improve scalability, data security, and ease of maintenance.

- **Voice and Speech Recognition**

Exploring the integration of voice recognition for hands-free navigation or as an accessibility feature for students with disabilities.

- **Blockchain for Certification**

Implementing blockchain technology for storing and verifying course certifications, ensuring they are tamper-proof and easily shareable with employers or other institutions.

Part 12: References

Backend

- Express.js: Used as the main framework for the backend server, facilitating the creation of RESTful APIs.
- MySQL: Used as the database to store user data, course details, assignments, submissions, and grades.
- Sequelize: An ORM (Object-Relational Mapping) tool that enabled us to manage MySQL databases through JavaScript models, simplifying database queries and data management.
- Multer: Used for handling file uploads, specifically for assignment submissions.
- Nodemailer: Implemented for sending email notifications, especially for registration confirmations or grade updates.
- dotenv: Managed environment variables securely, including database credentials and API keys.

Frontend

- React: Used as the main library for building the user interface, enabling efficient state management and dynamic page rendering.
- Material-UI: Provided pre-styled components for creating a responsive, user-friendly interface with minimal custom CSS.
- Axios: A promise-based HTTP client used to handle API requests and responses between the frontend and backend.

Deployment and Development Tools

- Postman: Used for testing and debugging API endpoints during the development process.
- Git: For version control and collaboration, allowing us to manage code changes and track progress throughout development.

Sources and Documentation

- MDN Web Docs: For general JavaScript and web development resources.
- Sequelize Documentation: Referenced for ORM setup and advanced query handling.
- React Documentation: Used to understand component lifecycle methods, hooks, and best practices in React.
- Material-UI Documentation: For component usage and styling guides, helping us apply a consistent UI theme.
- Nodemailer Documentation: To set up and troubleshoot email functionality.

<https://developer.mozilla.org/en-US/>

<https://sequelize.org/docs/v6/getting-started/>

<https://legacy.reactjs.org/docs/getting-started.html>

<https://github.com/mui/material-ui>

<https://www.nodemailer.com/>

<https://github.com/react-rev/RLMS>