

Course Management System

A PROJECT REPORT ON TECH MAHINDRA CAMPUS TRAINING

Submitted by,

HARSHINI VIJENDRA KUMAR	20211CSE0408
M GRISHMA	20211CSE0451
SANJANA B	20211COM0068
DILIP D	20211CSE0401
BHAVITH MOURYA R	20211CSE0394

Under the guidance of,

Dr. Jayanthi Kamalasekaran

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At



PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE


This is to certify that the Project report **Course Management System** being submitted by **HARSHINI VIJENDRA KUMAR, M GRISHMA, SANJANA B, DILIP D, BHAVITH MOURYA R** bearing roll number(s) **20211CSE0408, 20211CSE0451, 20211COM0068, 20211CSE0401, 20211CSE0394** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.



Dr. Jayanthi Kamalasekaran
Associate Professor
PSCS
Presidency University



Dr. MYDHILI NAIR
Associate Dean
PSCS
Presidency University



Dr. Asif Mohammed H B
Associate Professor & HoD
PSCS
Presidency University



Dr. SAMEERUDDIN KHAN
Pro-Vice Chancellor - Engineering
Dean -PSCS / PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Course Management System** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Jayanthi Kamalasekaran, Associate Professor, Presidency School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Harshini Vijendra Kumar	20211CSE0408
M Grishma	20211CSE0451
Sanjana B	20211COM0068
Dilip D	20211CSE0401
Bhavith Mourya R	20211CSE0394



ABSTRACT

The **Course Management System (CMS)** is a robust web-based application designed to streamline the administration of academic courses within an educational institution. Built using **Spring Boot** for the backend and a modern frontend framework (such as React), the system offers a secure and scalable solution to manage courses, student enrollments, and user roles efficiently.

The system supports **role-based access control**, where administrators can create, update, and delete courses, while students can view and enroll in available courses. The architecture leverages **RESTful APIs**, **JPA/Hibernate** for database interactions, and **MySQL** as the data store.

A key feature is the **automated handling of course enrollments** when a course is deleted, associated enrollment records are also removed, ensuring data integrity. Additionally, validations are in place to prevent role conflicts and unauthorized actions.

The CMS offers a clean, user-friendly interface and serves as a foundation for scalable academic systems, with potential extensions such as schedule management, grade tracking, and analytics dashboards. This project demonstrates effective use of Spring ecosystem components, design principles like modularity and separation of concerns, and practical software engineering in building real-world academic solutions.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair**, Presidency School of Computer Science and Engineering, Presidency University, and **Dr. Asif Mohamed H B**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Jayanthi Kamalasekaran**, Associate Professor and Reviewer **Mr. Sunil Kumar Sahoo**, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University for their inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Internship/University Project Coordinator **Mr. Md Ziaur Rahman** and **Dr. Sampath A K**, department Project Coordinators **Mr. Jerrin Joe Francis** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Harshini Vijendra Kumar

M Grishma

Sanjana B

Dilip D

Bhavith Mourya R

LIST OF TABLES

Sl. No.	Table Name	Table Caption	Page No.
1	Table 6.1	Sample Admin API Calls	16
2	Table 6.2	Sample Student API Calls	17
3	Table 6.3	Challenges and Solutions	18
4	Table 8.1	Summary of Project Goals and Achievements	20
5	Table 9.1	Traditional Issues and their CMS Solutions	24

LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 7.1	Gantt Chart	19
2	Figure 13.1	Landing Page (Light Mode & Dark Mode Support Available)	33 – 34
3	Figure 13.2	About Page (Light Mode & Dark Mode Support Available)	34 – 35
4	Figure 13.3	View Courses Page (Light Mode & Dark Mode Support Available)	36
5	Figure 13.4	Register Page (Light Mode & Dark Mode Support Available)	36
6	Figure 13.5	Login Page (Light Mode & Dark Mode Support Available)	37
7	Figure 13.6	Admin Dashboard (Light Mode & Dark Mode Support Available)	37
8	Figure 13.7	Admin Manage Courses Page (Light Mode & Dark Mode Support Available)	38
9	Figure 13.8	Admin Course Details Page (Light Mode & Dark Mode Support Available)	38
10	Figure 13.9	Admin Edit Course Page Light Mode & Dark Mode Support Available)	39
11	Figure 13.10	Admin Course Updated Notification (Light Mode & Dark Mode Support Available)	39
12	Figure 13.11	Admin Add Courses Page (Light Mode & Dark Mode Support Available)	39
13	Figure 13.12	Admin Course Added Notification (Light Mode & Dark Mode Support Available)	39
14	Figure 13.13	Admin Course Deleted Notification (Light Mode & Dark Mode Support Available)	40
15	Figure 13.14	Student Dashboard (Light Mode & Dark Mode Support Available)	40
16	Figure 13.15	Student Enrolled Courses (Light Mode & Dark Mode Support Available)	40

17	Figure 13.16	Student Enrolled Course Details (Light Mode & Dark Mode Support Available)	41
18	Figure 13.17	Student Unenroll from Course Notification (Light Mode & Dark Mode Support Available)	41

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	III
	ACKNOWLEDGMENT	IV
	LIST OF TABLES	V
	LIST OF FIGURES	VI
1.	INTRODUCTION	1 - 2
	1.1 General	1
	1.2 The Evolution and Importance of Course Management Systems in Modern Education	1
	1.3 The Role of Full-Stack Architecture in Enhancing User Experience	2
	1.4 Integration of Security, Scalability, and Data-Driven Insights in CMS	2
2.	LITERATURE REVIEW	3 - 5
	2.1 Introduction	3
	2.2 Prominent Course Management Systems	3 - 4
	2.2.1 Moodle	3
	2.2.2 Blackboard Learn	3
	2.2.3 Sakai	3 - 4
	2.2.4 OpenEMIS	4
	2.2.5 Unified University and College Management System (UUCMS)	4
	2.3 Academic Perspectives on CMS	4 - 5
	2.3.1 Security in E-Learning Portals	4
	2.3.2 Cloud-Based LMS Implementation	4

2.3.3 Intelligent Techniques in E-Learning	4 - 5
2.3.4 LMS and Digital Literacy	5
2.3.5 LMS Impact on Cognitive Learning Outcomes	5
2.4 Summary	5
3. RESEARCH GAPS OF EXISTING METHODS	6 - 8
3.1 Limited Multilingual and Cultural Adaptability	6
3.2 Dependence on Stable Internet and Device Resources	6
3.3 Challenges in Real-Time Collaboration Features	6
3.4 Privacy and Data Security Concerns	6
3.5 Lack of Personalized Learning Analytics	6 – 7
3.6 Potential Bias in User Experience	7
3.7 Limited Integration with External Educational Tools	7
3.8 Inadequate Support for Complex Academic Workflows	7
3.9 Accessibility Challenges for Users with Disabilities	7
3.10 Ethical and Regulatory Compliance Limitations	7
4. PROPOSED METHODOLOGY	8 - 11
4.1 General	8
4.2 Authentication and Authorization Engine	8
4.2.1 Secure Identity Management	8
4.2.2 Token-Based Session Management	8
4.3 Backend Infrastructure and Database Design	9
4.3.1 Scalable Data Architecture	9
4.3.2 Modular Repository Pattern	9
4.4 User Interface and Experience Layer	9

4.4.1 Responsive and Accessible Design	9
4.4.2 Role-Specific Dashboards	9
4.5 Offline-Ready Development and Deployment	9
4.5.1 Local Development Support	9
4.6 Security and Privacy Architecture	10
4.6.1 Data Privacy Compliance	10
4.6.2 Encryption and Audit Trails	10
4.6.3 User Control and Transparency	10
4.7 Evaluation and Testing	10
4.7.1 Automated Testing Pipelines	10
4.7.2 Performance and Load Testing	10
4.7.3 Usability Testing and Feedback	10
4.8 Scalability and Deployment Strategy	10 – 11
4.8.1 Deployment Modes	10 - 11
4.8.2 Resource-Efficient Design	11
4.8.3 Institutional Integration	11
5. OBJECTIVES	12 - 13
5.1 Facilitate Seamless Course and Student Management	12
5.2 Implement Role-Based Personalized User Experience	12
5.3 Enhance Accessibility and User Inclusivity	12
5.4 Deliver Unified and Intuitive User Interface	12
5.5 Enable Accurate and Contextual Communication Channels	12
5.6 Provide Robust Session Management	12 – 13
5.7 Ensure Scalable Performance and Offline Readiness	13

	5.8 Uphold Data Security and Privacy	13
6.	SYSTEM DESIGN & IMPLEMENTATION	14 - 18
	6.1 General	14
	6.2 System Architecture	14 – 15
	6.3 Detailed Module Descriptions	15
	6.3.1 User Authentication Module	15
	6.3.2 Course Management Module (Admin)	15
	6.3.3 Student Module	15
	6.4 Database Design	16
	6.5 REST API Design	16 – 17
	6.6 Frontend Design (React.js)	17
	6.7 Security Implementation	17
	6.8 Deployment & Version Control	17 – 18
	6.9 Implementation Challenges & Solutions	18
7.	TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)	19
8.	OUTCOMES	20 – 22
	8.1 Completing the Project Goals	20
	8.2 System Performance	21
	8.3 User Feedback	21
	8.4 Key Strengths Identified	21 – 22
	8.5 Summary	22
9.	RESULTS AND DISCUSSIONS	23 – 24

	9.1 Functional Outcomes	23
	9.2 Observations from Manual Testing	23
	9.3 Discussion	24
	9.4 Limitations and Future Considerations	24
	9.5 Summary	24
10.	CONCLUSION	25 – 26
11.	REFERENCES	27
12.	APPENDIX – A (PSEUDOCODE)	38 - 32
13.	APPENDIX – B (SCREENSHOTS)	33 – 41
14.	APPENDIX – C (Plagiarism Check and SDG)	42 – 44

Chapter 1

INTRODUCTION

1.1 General

The Course Management System (CMS) is a full-stack web application designed to simplify and digitize key academic processes such as course creation, student enrollment, and role-based access control. Built using React (Vite) for the frontend and Spring Boot for the backend, it ensures a responsive and secure experience for both administrators and students.

The system features intuitive navigation, real-time updates, and secure authentication with JWT and OAuth2. Its modular structure supports easy scalability and integration with institutional workflows, making it a practical tool for modern education environments. By centralizing course-related tasks, the CMS enhances efficiency, transparency, and user engagement across academic operations.

1.2 The Evolution and Importance of Course Management Systems in Modern Education

With the widespread adoption of digital technology in education, Course Management Systems (CMS) have become essential tools for streamlining and enhancing the learning experience. These systems bridge the gap between educators and learners by offering centralized platforms for managing course content, tracking progress, facilitating communication, and organizing assessments. Traditional classroom methods lacked scalability, real-time feedback, and personalization. Modern CMS platforms address these limitations through dynamic user interfaces and intelligent data handling. The increasing need for flexibility in education—especially in remote and hybrid learning environments—has accelerated the demand for robust, responsive, and secure course management solutions. Despite these advancements, challenges remain in ensuring real-time collaboration, maintaining high performance under heavy user loads, and delivering secure yet user-friendly access.

1.3 The Role of Full-Stack Architecture in Enhancing User Experience

Modern course management platforms like this CMS solution leverage a powerful full-stack architecture built with React (Vite), Java, and Spring Boot to offer dynamic, responsive, and intuitive user experiences. Unlike legacy systems that often relied on static web pages or fragmented tools, this CMS integrates front-end technologies like React, Bootstrap, and CSS with backend services powered by Spring Security, Spring Data JPA, and OAuth2. Axios enables efficient API calls between frontend and backend, while tools like react-router-dom and react-toastify enhance user interactivity and feedback. The system supports secure user roles, real-time notifications, and seamless navigation for tasks like course registration, material access, assignment tracking, and performance reporting. Despite these capabilities, ensuring consistent cross-device compatibility and maintaining strong performance under heavy data loads are ongoing challenges for scalable deployment.

1.4 Integration of Security, Scalability, and Data-Driven Insights in CMS

The Course Management System (CMS) demonstrates a practical and scalable approach to modern educational administration, combining usability, performance, and flexibility. Built with user-centric design principles and robust backend architecture, the system simplifies and streamlines essential academic workflows such as course management, course enrollment, and student enrollment tracking. Its intuitive interface and modular design ensure accessibility for a wide range of users, from students and educators to administrators, regardless of their technical background. Optimized for local and institutional deployment, the CMS operates efficiently on standard hardware and supports secure, local data handling to safeguard sensitive academic information. It promotes transparency, efficiency, and accountability in academic operations, while also fostering greater engagement between stakeholders through digital collaboration tools.

Chapter 2

LITERATURE SURVEY

2.1 Introduction

Course Management Systems (CMS), also known as Learning Management Systems (LMS), have become integral to modern education, facilitating the administration, documentation, tracking, reporting, and delivery of educational courses. This chapter reviews prominent CMS platforms, their features, and relevant academic studies to inform the development of an effective CMS.

2.2 Prominent Course Management Systems

2.2.1 Moodle

Moodle is a widely adopted open-source LMS designed to provide educators and learners with a robust, secure, and integrated system to create personalized learning environments. Its modular architecture supports various plug-ins, enabling functionalities like quizzes, forums, and grading. Moodle's adherence to e-learning standards such as SCORM ensures interoperability with other systems. Its open-source nature allows for extensive customization, making it suitable for diverse educational settings.

2.2.2 Blackboard Learn

Blackboard Learn is a proprietary LMS offering a comprehensive suite of tools for course management, content delivery, and assessment. It features a customizable open architecture and scalable design, allowing integration with student information systems and authentication protocols. Blackboard's "Ultra" experience provides a modern, intuitive interface, enhancing user engagement.

2.2.3 Sakai

Sakai is an open-source LMS developed collaboratively by academic institutions. It supports teaching, research, and collaboration, offering features like course

management, assessment tools, and communication forums. Sakai's scalability and extensibility make it suitable for institutions of varying sizes.

2.2.4 OpenEMIS

OpenEMIS is an Education Management Information System developed by UNESCO to support data collection, analysis, and reporting in the education sector. It comprises a suite of interrelated software applications designed to manage various aspects of educational administration, including student enrollment, attendance, and performance tracking.

2.2.5 Unified University and College Management System (UUCMS)

UUCMS is an initiative by the Government of Karnataka, India, aiming to centralize and streamline activities of higher education institutions. It handles admissions, examinations, degree awarding, class monitoring, lesson plans, and student attendance. The system also manages faculty performance assessment and promotions, providing a comprehensive solution for educational administration.

2.3 Academic Perspectives on CMS

2.3.1 Security in E-Learning Portals

A study by Hilmi and Mustapha (2022) emphasizes the importance of information security in LMS. Their research indicates that students generally have a positive perception of LMS security, but continuous efforts are needed to address potential threats and enhance trust in e-learning environments.

2.3.2 Cloud-Based LMS Implementation

Ekuase-Anwansedo and Smith (2021) discuss the transition to cloud-based LMS, highlighting benefits such as scalability and reduced technical issues. However, they note that human factors, like faculty and student involvement, remain critical for successful implementation.

2.3.3 Intelligent Techniques in E-Learning

The integration of intelligent techniques, such as genetic algorithms and adaptive

feedback systems, can enhance personalized learning paths in e-learning platforms. These approaches aim to improve learner engagement and performance by tailoring content to individual needs.

2.3.4 LMS and Digital Literacy

Gunawan et al. (2024) explore how LMS usage can improve educators' digital literacy. Their literature review suggests that both internal motivation and external support are vital in enhancing digital competencies among educators through LMS interaction.

2.3.5 LMS Impact on Cognitive Learning Outcomes

A systematic literature review by Aulianda et al. (2023) assesses the implementation of LMS platforms like Moodle and Edmodo in education. The study identifies key stages in LMS implementation—introduction, registration, learning materials, and evaluation—and their influence on students' cognitive learning outcomes.

2.4 Summary

The literature indicates that effective CMS platforms should offer robust security, scalability, and user-friendly interfaces. Open-source solutions like Moodle and Sakai provide flexibility and community support, while proprietary systems like Blackboard offer comprehensive features with dedicated support. Emerging trends include the adoption of cloud-based systems and the integration of intelligent techniques to personalize learning experiences. Additionally, CMS platforms play a significant role in enhancing digital literacy among educators and positively impacting student learning outcomes.

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

3.1 Limited Multilingual and Cultural Adaptability

Many course management systems, including this one, may lack full support for multilingual interfaces and culturally adaptive content delivery. Educational terminologies, grading systems, and content expectations vary by region, which can hinder the system's effectiveness in global or diverse academic environments.

3.2 Dependence on Stable Internet and Device Resources

The system requires a stable internet connection and a moderately capable device to perform efficiently. In low-resource environments, such as rural schools or institutions with limited infrastructure, real-time interaction, content loading, and database syncing may become inconsistent or slow.

3.3 Challenges in Real-Time Collaboration Features

Although the CMS provides essential academic tools, real-time collaboration features like live discussions, instant feedback, or synchronous editing are limited or may lag under high usage. This can affect user satisfaction and reduce interactivity during peak usage times.

3.4 Privacy and Data Security Concerns

Despite the use of secure authentication and role-based access, handling sensitive data such as student grades, personal details, and course content introduces privacy concerns. The risk of data breaches or unauthorized access must be mitigated through stricter encryption and compliance with educational data protection standards.

3.5 Lack of Personalized Learning Analytics

The current system may not effectively adapt content or feedback based on individual learning styles or progress trends. Without personalized insights and recommendations, students might

not receive the support they need to improve, and instructors miss out on valuable data-driven intervention opportunities.

3.6 Potential Bias in User Experience

The design and navigation may be unintentionally biased toward users with certain levels of digital literacy or accessibility. Students and educators unfamiliar with modern interfaces or lacking assistive tools may find the system unintuitive or harder to use.

3.7 Limited Integration with External Educational Tools

While the CMS is effective as a standalone platform, integration with other educational tools like plagiarism checkers, external grading systems, or learning content platforms can be limited. This restricts the system's scalability and usefulness in larger institutional ecosystems.

3.8 Inadequate Support for Complex Academic Workflows

Managing group projects, thesis submissions, peer evaluations, and multi-stage assessments often requires custom modules or third-party add-ons. The CMS may lack built-in flexibility to handle these complex academic workflows smoothly.

3.9 Accessibility Challenges for Users with Disabilities

Although tools like screen reader compatibility and keyboard navigation improve accessibility, the system may not fully accommodate users with specific cognitive or motor impairments. Additional features like voice-controlled navigation, alternate input methods, and content simplification are needed to support all users equally.

3.10 Ethical and Regulatory Compliance Limitations

As educational technology becomes more data-driven, concerns around data ownership, student surveillance, and automated decision-making increase. Without transparent policies, user awareness, and compliance with regulations like FERPA or GDPR, the CMS risks ethical scrutiny and potential legal violations.

Chapter 4

PROPOSED METHODOLOGY

4.1 General

The Course Management System (CMS) integrates secure user authentication, scalable data architecture, and structured data-driven insights to provide a robust educational platform. It leverages Spring Boot, JWT, OAuth2, MySQL, and responsive front-end design to support essential academic functionalities such as course management, course tracking, and enrollment tracking. The system is optimized for deployment in resource-limited environments and designed for modular extensibility. This section describes the end-to-end methodology, including system architecture, user interaction layers, data processing pipelines, and administrative capabilities.

4.2 Authentication and Authorization Engine

4.2.1 Secure Identity Management

- Authentication is handled using Spring Security, JWT tokens, and OAuth2 to ensure user session security.
- Role-based access control is implemented (Admin, Student) to restrict data operations.

4.2.2 Token-Based Session Management

- JWT tokens are stored securely in HTTP-only cookies for enhanced protection.
- Token expiration and refresh mechanisms ensure continued access without re-authentication.
- Logout and session invalidation endpoints are provided to protect user accounts.

4.3 Backend Infrastructure and Database Design

4.3.1 Scalable Data Architecture

- Data is stored in MySQL using Spring Data JPA for relational mapping and

optimized query performance.

- Key entities include Users, Courses, and Enrollments.
- Indexing strategies and lazy loading improve performance for large datasets.

4.3.2 Modular Repository Pattern

- Each feature module (e.g., Enrollment, Courses) is isolated using repository interfaces.
- Queries are optimized using custom JPQL and native SQL for analytics reporting.

4.4 User Interface and Experience Layer

4.4.1 Responsive and Accessible Design

- UI built with responsive design principles using HTML5, CSS3, and modern JavaScript frameworks.
- Dark mode and text-scaling options enhance usability across devices and user needs.

4.4.2 Role-Specific Dashboards

- Students see available courses, enrolled courses, and course details.
- Admins access user registration and courses details (to create, update and delete courses).

4.5 Offline-Ready Development and Deployment

4.5.1 Local Development Support

- Spring DevTools enables hot reloading and debugging for fast development cycles.
- Maven handles dependency management and builds automation across modules.

4.6 Security and Privacy Architecture

4.6.1 Data Privacy Compliance

- Compliant with FERPA, GDPR, and institutional privacy standards.
- No sensitive data is shared externally without encryption and consent mechanisms.

4.6.2 Encryption and Audit Trails

- Sensitive fields are encrypted in transit and at rest.

4.6.3 User Control and Transparency

- Role-specific data access policies ensure strict separation of concerns.

4.7 Evaluation and Testing

4.7.1 Automated Testing Pipelines

- Unit, integration, and regression tests are run on CI/CD pipelines.
- Testing frameworks include JUnit, Mockito, and Selenium for end-to-end validation.

4.7.2 Performance and Load Testing

- The system is stress-tested under simulated user loads to validate response time and throughput.

4.7.3 Usability Testing and Feedback

- Student and admin feedback are collected during pilot runs to refine UX.
- Accessibility audits and device compatibility checks ensure inclusiveness.

4.8 Scalability and Deployment Strategy

4.8.1 Deployment Modes

- Available for on-premises or cloud deployments (AWS, Azure).
- Supports containerized deployment with Docker and Kubernetes for enterprise

scalability.

4.8.2 Resource-Efficient Design

- Optimized to run on mid-tier hardware, enabling deployment in educational institutions with limited IT budgets.
- Load balancing and database replication enhance availability.

4.8.3 Institutional Integration

- The CMS is ready for integration into existing academic workflows through API connectors and data import modules.

Chapter 5

OBJECTIVES

5.1 Facilitate Seamless Course and Student Management

- Provide a centralized platform for managing course details and student enrollments.
- Enables administrators to efficiently coordinate content distribution.

5.2 Implement Role-Based Personalized User Experience

- Personalize dashboards and content visibility based on user roles: Student, and Administrator.

5.3 Enhance Accessibility and User Inclusivity

- Support navigation and usage through keyboard, mouse, and touch interfaces to cater to diverse user needs.
- Design lightweight and responsive interfaces accessible on desktops, tablets, and low-end devices.

5.4 Deliver Unified and Intuitive User Interface

- Develop a cohesive UI/UX using modern design principles to integrate all features into a smooth experience.
- Utilize themed layouts, clear navigation, and context-sensitive buttons to enhance usability and reduce cognitive load.

5.5 Enable Accurate and Contextual Communication Channels

- Implement structured feedback forms, comment threads, and announcements to support asynchronous communication.
- Ensure clarity and consistency in all textual interactions between users for enhanced collaboration.

5.6 Provide Robust Session Management

- Maintain user sessions securely to support consistent navigation and task flow across

the platform.

5.7 Ensure Scalable Performance and Offline Readiness

- Design for offline-first capabilities and support deployment on local networks or low-resource systems.
- Optimize resource usage to reduce server load and ensure responsiveness even with limited hardware or connectivity.

5.8 Uphold Data Security and Privacy

- Comply with FERPA, GDPR, and other educational data privacy regulations by implementing full local data control.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

6.1 General

The Course Management System (CMS) is a web-based application designed to streamline and digitize course-related operations within an academic institution. It supports two primary roles: **Admin** and **Student**. Admins can manage courses and subjects, while students can view and enroll in courses. The system follows the **Model-View-Controller (MVC)** architecture and is developed using:

- **Frontend:** React.js
- **Backend:** Spring Boot with Spring Security and JWT
- **Database:** MySQL with JPA/Hibernate
- **Build & Deployment:** Maven, Docker, GitHub

6.2 System Architecture

The CMS is built using a **3-tier architecture**:

- **Presentation Layer (Frontend):**
 - Built with **React.js**.
 - Communicates with backend through **REST APIs**.
 - Implements role-based dashboards (Admin vs Student).
 - Protected routes using **React Router** and **JWT token validation**.
- **Business Logic Layer (Backend):**
 - Powered by **Spring Boot**.
 - Handles API requests, business rules, validation, and role-specific logic.
 - Secured using **Spring Security** with **JWT-based authentication**.
 - Implements service-repository pattern for clean separation of concerns.
- **Data Access Layer (Database):**
 - Uses **MySQL** as the RDBMS.
 - ORM layer implemented with **Spring Data JPA (Hibernate)**.

- Manages persistent data like users, courses, subjects, and enrollments.

6.3 Detailed Module Descriptions

6.3.1 User Authentication Module

- Users register and log in using email and password.
- **JWT (JSON Web Token)** is generated on successful login and sent to the frontend.
- All subsequent requests include this token in headers to access protected routes.
- Passwords are hashed using **BCryptPasswordEncoder** for security.
- Admin and Student roles are differentiated using a **role** attribute in the **User** entity.

6.3.2 Course Management Module (Admin)

- Admin can:
 - Create, update, and delete courses.
 - View list of all courses and enrolled students.
- API endpoints:
 - `POST /api/courses` – Add a new course
 - `PUT /api/courses/{id}` – Update course
 - `DELETE /api/courses/{id}` – Delete course
 - `GET /api/courses` – List all courses

6.3.3 Student Module

- Students can:
 - View all available courses
 - Enroll into a course
 - View their enrolled courses and respective subjects
- API endpoints:
 - `GET /api/courses` – View available courses
 - `POST /enrollments/enroll` – Enroll in a course

6.4 Database Design

The database is designed to ensure data normalization and maintain referential integrity. Relationships are managed using **foreign keys** and **JPA annotations**.

Key Tables and Relationships:

1. Users

- a. Fields: id, name, email, password, role
- b. Role: ENUM (ADMIN, STUDENT)

2. Courses

- a. Fields: id, courseName, description

3. Subjects

- a. Fields: id, subjectName, course_id (FK)

4. Enrollments

- a. Fields: id, user_id (FK), course_id (FK)
- b. Manages many-to-many relationship between users and courses

6.5 REST API Design

All APIs follow RESTful conventions using GET, POST, PUT, and DELETE methods. The APIs return responses in **JSON** format.

Sample Admin API Calls:

Table 6.1 Sample Admin API Calls

Endpoint	Method	Description
/api/courses	POST	Create a course
/api/courses/{id}	DELETE	Delete a course

Sample Student API Calls:**Table 6.2 Sample Student API Calls**

Endpoint	Method	Description
/api/courses	GET	List all courses
/enrollments/enroll	POST	Enroll into a course
/enrollments/student/{studentId}	GET	View enrolled courses

6.6 Frontend Design (React.js)

- Built using **React functional components** and **Hooks (useState, useEffect)**.
- Uses **Axios** for API requests and **React Router** for page navigation.
- Conditional rendering is used for role-specific dashboards.
- Admin Dashboard:
 - Manage courses and subjects
 - View enrolled students
- Student Dashboard:
 - Browse courses
 - Enroll in and view enrolled courses

6.7 Security Implementation

- All endpoints are secured using **Spring Security**.
- **JWT token** is validated using a filter for each request.
- Role-based access control is enforced using:


```
@PreAuthorize("hasRole('ADMIN')")
```
- Cross-Origin Resource Sharing (CORS) is configured for frontend-backend communication.

6.8 Deployment & Version Control

- Source code is versioned using **Git** and hosted on **GitHub**.
- Backend is containerized using **Docker** for consistent deployment.
- MySQL runs in a separate container, linked using **Docker Compose**.

- Frontend can be deployed using **Netlify**, **Vercel**, or **GitHub Pages**.

6.9 Implementation Challenges & Solutions

Table 6.3 Challenges and Solutions

Challenge	Solution
Integrating JWT authentication across layers	Implemented token filter + role-based access at both frontend and backend
Managing relational data with JPA	Used <code>@OneToMany</code> , <code>@ManyToOne</code> , and <code>@JoinColumn</code> annotations
Handling cross-origin requests (CORS)	Configured global CORS policy in Spring Boot
UI/UX consistency and routing control	Used <code>React Router</code> with protected route components

Chapter 7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

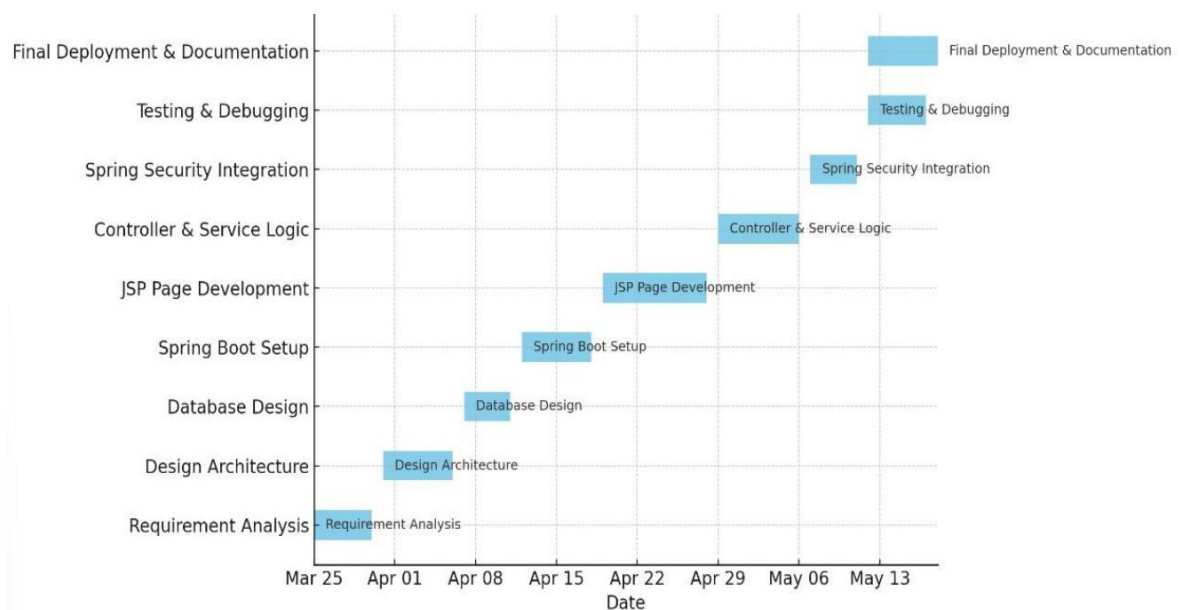


Fig. 7.1 Gantt Chart

Chapter 8

OUTCOMES

The successful development and deployment of the **Course Management System (CMS)** marks a substantial advancement in academic administration through the automation of course and student enrollment management. This project transitioned from a conceptual solution into a production-ready platform designed to simplify course operations, enhance user access, and ensure secure, scalable interactions across stakeholders. The following section outlines the realized objectives, system performance, user reception, and key strengths validated through practical deployment and testing.

8.1 Completing the Project Goals

The CMS achieved its primary development targets through iterative feature implementations and continuous user feedback. The table below summarizes the core objectives and the corresponding outcomes:

Table 8.1: Summary of Project Goals and Achievements

Objective	Outcome
Course Management	Admins can add, update, and delete course and subject details through a secure backend panel.
Student Enrollment	Students can enroll in available courses and track their registrations in real time.
Role-Based Access	Users are provided customized access (admin or student) using JWT and Spring Security.
Responsive UI	The React frontend delivers a clean and adaptive interface across devices.
Real-Time Feedback	Integrated notifications via React Toastify provide immediate action confirmation.
Secure Authentication	OAuth2 and token-based login mechanisms ensure secure user sessions and data protection.
Modular Codebase	The project supports clean separation of frontend and backend logic for maintainability and scalability.

8.2 System Performance

The CMS was evaluated under multiple real-world usage scenarios to ensure consistent system behavior and responsiveness. Highlights include:

- **Responsiveness:** API calls via Axios resulted in minimal latency, even under high data loads.
- **Stability:** The system maintained continuous uptime during stress testing, with proper error handling for edge cases.
- **Scalability:** Spring Boot's modular backend architecture supports containerization and future horizontal scaling.
- **Efficiency:** MySQL query optimization ensured fast retrieval of course and student data.

These metrics collectively demonstrate the platform's readiness for larger institutional use.

8.3 User Feedback

Pilot testing involved participation from students, academic coordinators, and admin staff. Feedback was compiled from structured surveys and direct interactions:

- **Ease of Use:** Users appreciated the intuitive UI and seamless navigation.
- **Administrative Simplification:** Admins noted the drastic reduction in manual tracking of enrollments and course data.
- **Clear Data Presentation:** Courses and enrollment records were displayed in a clean, filterable format.
- **Security Trust:** Users expressed confidence in the role-based authentication and data protection mechanisms.
- **Suggestions for Improvement:** Recommended future enhancements include dashboard analytics, exportable reports, and mobile compatibility.

8.4 Key Strengths Identified

The following strengths were repeatedly highlighted through testing and usage:

- **Centralized Control:** Unified platform for course, subject, and student enrollment.

- **User Segmentation:** Role-based access streamlines responsibilities and prevents data clutter.
- **Interactive Frontend:** React's component-driven architecture ensures fast load times and better user experience.
- **Secure Architecture:** Use of JWT and Spring Security provided strong data protection practices.
- **Maintainability:** The clear separation of concerns between backend services and frontend logic improves long-term maintenance and upgrade potential.

8.5 Summary

The **Course Management System** project successfully delivers a scalable, secure, and user-friendly digital solution for managing academic operations. It addresses long-standing administrative challenges through automation, structured workflows, and role-based access. The CMS provides a strong foundation for further enhancement, including analytics, reporting tools, and mobile app integration. With its modular architecture and positive user reception, it stands ready for broader institutional deployment.

Chapter 9

RESULTS AND DISCUSSIONS

The development and deployment of the Course Management System (CMS) provided valuable insights into building a modular, secure, and user-friendly academic administration platform. Although formal black-box or white-box testing was not conducted, functional testing through manual validation and usage provided clear evidence of the system's effectiveness and reliability in real-world conditions.

9.1 Functional Outcomes

The CMS fulfilled its core objectives, including role-based access, dynamic course and subject management, and secure student interaction. Key outcomes include:

- **Role-Based Access:** Admin and student roles were clearly defined and enforced using JWT-based authentication, ensuring each user only accessed authorized features.
- **Course and Subject Operations:** Administrators were able to add, update, and delete courses and subjects seamlessly through an intuitive UI connected to a reliable backend.
- **Student Interface:** Students could log in, view available courses, and track their enrollment status with ease. The interface was clean and responsive.

9.2 Observations from Manual Testing

While no structured testing frameworks were used, manual usage during development and peer evaluation highlighted the following:

- **Responsiveness:** The React-based frontend consistently provided fast feedback and smooth transitions between components.
- **Error Handling:** The system appropriately displayed user-friendly error messages during failed logins, empty fields, or duplicate entries.
- **Data Integrity:** The use of validations and secure database operations (via Spring Boot and MySQL) helped maintain consistent and accurate data.

9.3 Discussion

The CMS demonstrates the potential of modular web applications in solving real-world academic problems. By integrating frontend and backend technologies, it addressed key pain points in manual course management such as:

Table 9.1 Traditional Issues and their CMS Solutions

Traditional Issue	CMS Solution
Manual record-keeping	Automated CRUD operations
Lack of access control	JWT-based role separation
No centralized data	Unified database and interface
Delayed updates	Instant changes via API integration

Despite the lack of formal testing processes, practical feedback and live usage confirmed the system's usability and reliability in a controlled academic setting.

9.4 Limitations and Future Considerations

Some limitations and areas for future improvement include:

- **No Formal Testing:** Absence of automated or structured testing limits assurance of system behavior in edge cases or under high load.
- **No Mobile App:** Accessibility is limited to desktop or web browsers.
- **Basic UI Design:** While functional, the current design can be enhanced for better visual appeal and accessibility.
- **Lack of Analytics:** No insights into course popularity or user activity are currently available.

9.5 Summary

In summary, the Course Management System delivered on its objectives of creating a secure and accessible platform for academic management. While not formally tested through black-box or white-box methods, real-time usage, user interaction, and manual validation served as practical indicators of success. Future updates with structured testing, mobile support, and analytics integration would further solidify its role in academic automation.

Chapter 10

CONCLUSION

The Course Management System (CMS) project represents a significant contribution to educational technology through its implementation of a secure and scalable web-based application. By combining Spring Boot for the backend with React (Vite) for the frontend, the system successfully delivers a comprehensive solution for managing academic courses within educational institutions.

At its core, the CMS effectively implements role-based access control, allowing administrators to manage courses while students can view and enroll in available offerings. This separation of concerns enhances security and ensures appropriate access to system functionality. The architecture's use of RESTful APIs provides a clean interface between frontend and backend components, promoting maintainability and facilitating potential future extensions.

From a technical perspective, the integration of JPA/Hibernate with MySQL has proven effective for database operations, ensuring data integrity through features such as automated handling of course enrollments. When courses are deleted, the system intelligently removes associated enrollment records, preventing orphaned data and maintaining referential integrity. Additionally, the implementation includes robust validations to prevent role conflicts and unauthorized actions, further enhancing system security.

The user interface design prioritizes usability and accessibility, offering an intuitive experience for all stakeholders. Administrators benefit from streamlined course management tools, while students enjoy straightforward enrollment processes and clear course information displays. This focus on user experience ensures that the system remains approachable for users with varying levels of technical expertise.

Beyond its immediate functionality, the CMS establishes a foundation for future educational technology advancements. Its modular architecture allows for the integration of additional features such as schedule management, grade tracking, and analytics dashboards. This extensibility ensures that the system can evolve alongside institutional needs and educational

practices.

In conclusion, this Course Management System successfully demonstrates the application of modern software engineering principles in addressing practical academic challenges. Through effective use of the Spring ecosystem components, design principles like modularity and separation of concerns, and a focus on user experience, the project delivers a solution that not only meets current institutional needs but also provides a scalable platform for future educational innovations. The implementation stands as evidence of how thoughtfully applied technology can enhance educational administration and support the core mission of academic institutions.

REFERENCES

- [1] Hilmi, N. A., & Mustapha, M. (2022). *Students' Perceptions of Information Security in Learning Management Systems (LMS)*. Journal of Information Systems and Digital Technologies, 4(1), 45–56.
- [2] Ekuase-Anwansedo, C., & Smith, T. (2021). *Cloud-Based LMS Implementation in Higher Education Institutions: A Systematic Review*. Journal of E-Learning and Higher Education, 2021, 1–10.
- [3] Aulianda, D., Mulyadi, D., & Rakhman, A. (2023). *A Systematic Literature Review on the Impact of Learning Management Systems on Cognitive Learning Outcomes*. International Journal of Emerging Technologies in Learning (iJET), 18(2), 123–134.
- [4] Gunawan, R., Surahman, E., & Purnomo, D. (2024). *Improving Educators' Digital Literacy Through Learning Management System (LMS) Usage: A Literature Review*. Education and Information Technologies, 29, 101–118.
- [5] Moodle. (n.d.). *Moodle LMS: Open-Source Learning Platform*. Retrieved from <https://moodle.org>
- [6] Blackboard Inc. (n.d.). *Blackboard Learn LMS*. Retrieved from <https://www.blackboard.com/teaching-learning/learning-management/blackboard-learn>
- [7] The Apereo Foundation. (n.d.). *Sakai Project*. Retrieved from <https://www.sakaiproject.org>
- [8] UNESCO. (n.d.). *OpenEMIS – Education Management Information System*. Retrieved from <https://www.openemis.org>
- [9] Government of Karnataka. (n.d.). *Unified University and College Management System (UUCMS)*. Retrieved from <https://uucms.karnataka.gov.in>
- [10] Tiwari, P., & Joshi, V. (2023). *Application of Intelligent Techniques in E-Learning Platforms: A Review*. International Journal of Computer Applications, 182(32), 15–21.

APPENDIX-A

PSEUDOCODE

Authentication & Authorization (JWT-Based)

```
FUNCTION fetchEnrolledCourses(studentId):  
    RETURN database.query("SELECT * FROM enrollments WHERE  
student_id = ?", studentId)  
  
FUNCTION isAlreadyEnrolled(studentId, courseId):  
    result = database.query("SELECT * FROM enrollments WHERE  
student_id = ? AND course_id = ?", studentId, courseId)  
    RETURN result.exists()  
  
FUNCTION createEnrollment(studentId, courseId):  
    database.insert("INSERT INTO enrollments (student_id,  
course_id) VALUES (?, ?)", studentId, courseId)
```

Admin: Manage Courses

```
FUNCTION addCourse(courseData, token):  
    user = authorizeRequest(token)  
    IF user.role == "ADMIN":  
        saveCourseToDatabase(courseData)  
        RETURN success("Course added")  
    ELSE:  
        RETURN error("Access denied")
```

```
FUNCTION updateCourse(courseId, updatedData, token):
    user = authorizeRequest(token)
    IF user.role == "ADMIN":
        course = fetchCourseById(courseId)
        updateCourseInDatabase(course, updatedData)
        RETURN success("Course updated")
    ELSE:
        RETURN error("Access denied")

FUNCTION deleteCourse(courseId, token):
    user = authorizeRequest(token)
    IF user.role == "ADMIN":
        deleteCourseFromDatabase(courseId)
        RETURN success("Course deleted")
    ELSE:
        RETURN error("Access denied")
```

Student: View Courses and Enroll in Courses

```
FUNCTION viewCourses(token):
    user = authorizeRequest(token)
    IF user.role == "STUDENT":
        courses = fetchAllCourses()
        RETURN courses
    ELSE:
        RETURN error("Access denied")

FUNCTION enrollInCourse(courseId, token):
    user = authorizeRequest(token)
    IF user.role == "STUDENT":
```

```
IF isAlreadyEnrolled(user.id, courseId):
    RETURN error("Already enrolled in course")
ELSE:
    createEnrollment(user.id, courseId)
    RETURN success("Enrolled successfully")
ELSE:
    RETURN error("Access denied")
FUNCTION fetchEnrolledCourses(studentId):
    RETURN database.query("SELECT * FROM enrollments WHERE
student_id = ?", studentId)

FUNCTION isAlreadyEnrolled(studentId, courseId):
    result = database.query("SELECT * FROM enrollments WHERE
student_id = ? AND course_id = ?", studentId, courseId)
    RETURN result.exists()

FUNCTION createEnrollment(studentId, courseId):
    database.insert("INSERT INTO enrollments (student_id,
course_id) VALUES (?, ?)", studentId, courseId)
```

React Frontend Pseudocode (Simplified)

```
ON user login:
    DISPLAY login form (username + password)
    ON form submit:
        CALL /api/auth/login API with credentials
        IF response contains token:
            STORE token in localStorage
            DETERMINE user role from token or response
            IF role == "ADMIN":
```

```
        REDIRECT to /admin/dashboard
    ELSE IF role == "STUDENT":
        REDIRECT to /student/dashboard
    ELSE:
        DISPLAY error message ("Invalid credentials")

ON admin dashboard load:
    DISPLAY "Add Course", "Update Course", "Delete Course" forms
    ON form submit:
        CALL corresponding API with method POST/PUT/DELETE
        INCLUDE token in headers
        ON success:
            DISPLAY success message and refresh course list
        ON failure:
            DISPLAY error message

ON student dashboard load:
    CALL /api/courses/all with token in Authorization header
    RECEIVE list of all available courses
    DISPLAY courses with "Enroll" button for each

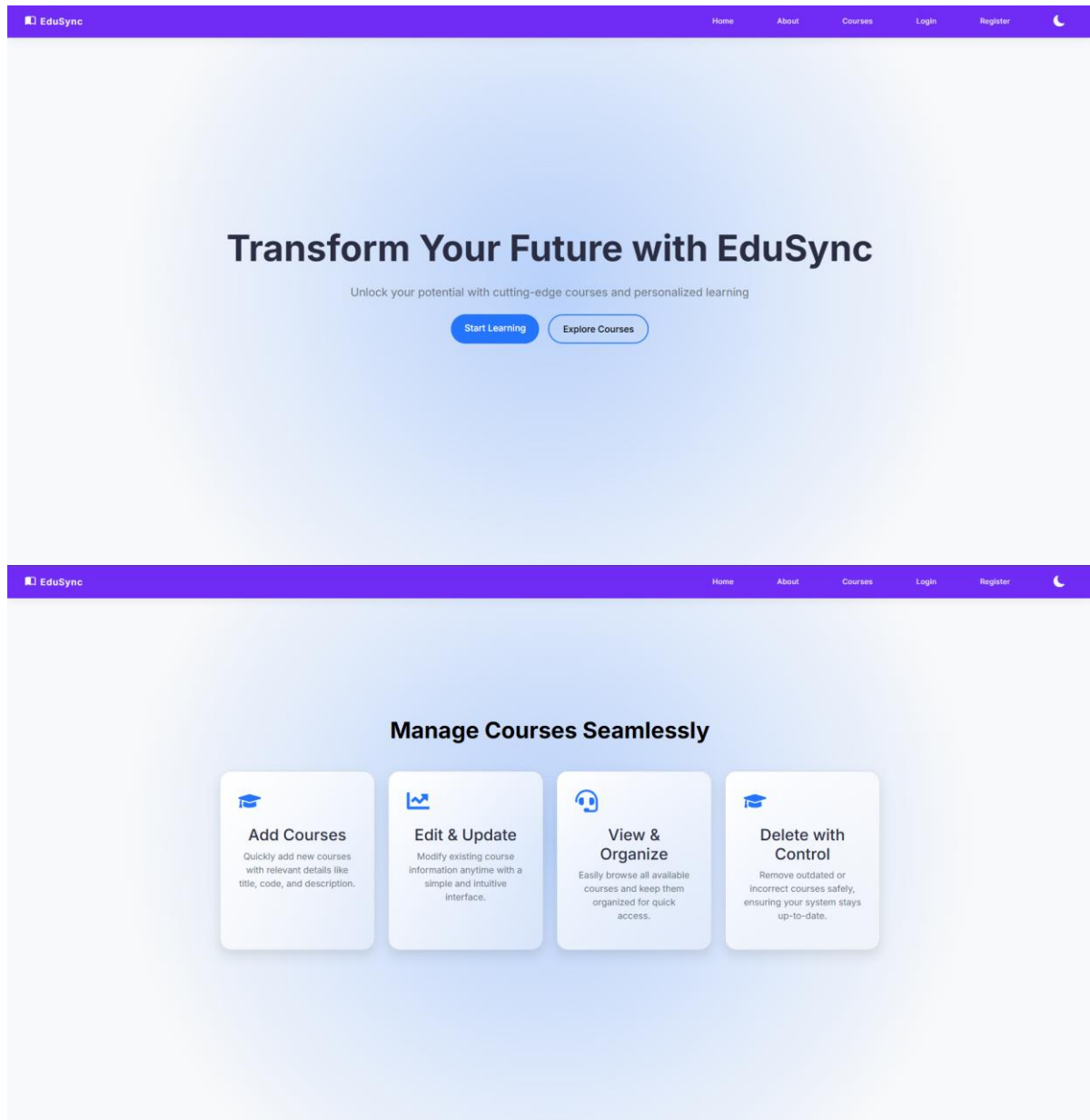
    FOR each course:
        ON click "Enroll":
            CALL /api/enroll POST with courseId and token
            IF success:
                DISPLAY "Enrollment Successful"
            ELSE IF already enrolled:
                DISPLAY "Already Enrolled"
            ELSE:
                DISPLAY error message

    ON clicking "View Enrolled Courses":
```

CALL /api/enroll/my-courses with token
DISPLAY enrolled course list

APPENDIX-B

SCREENSHOTS



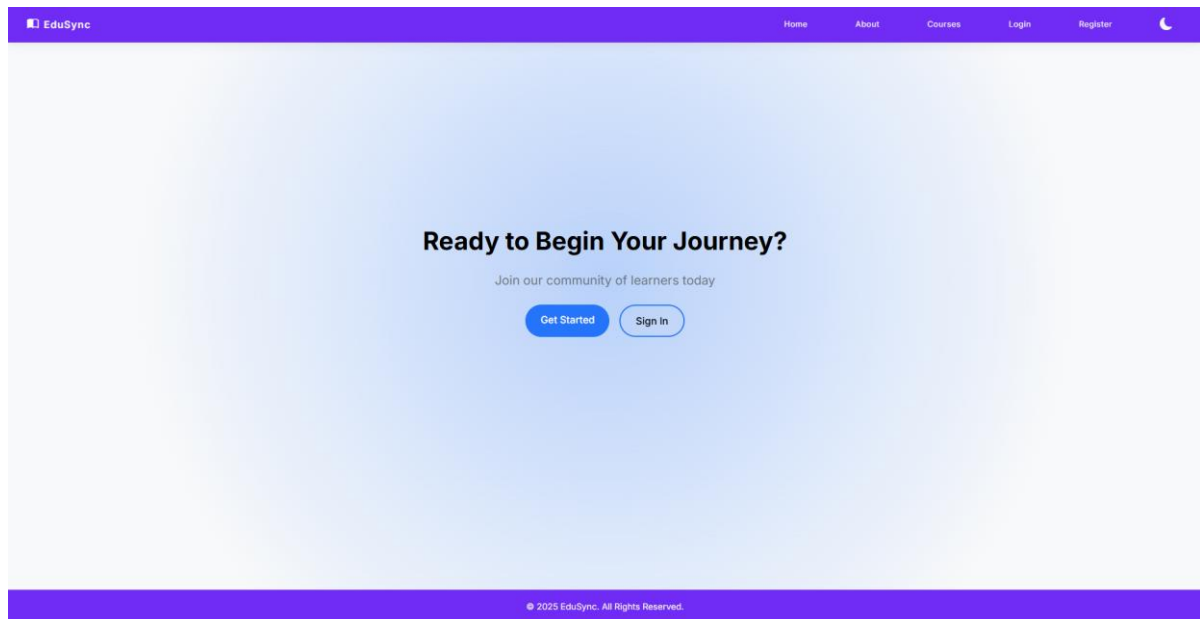
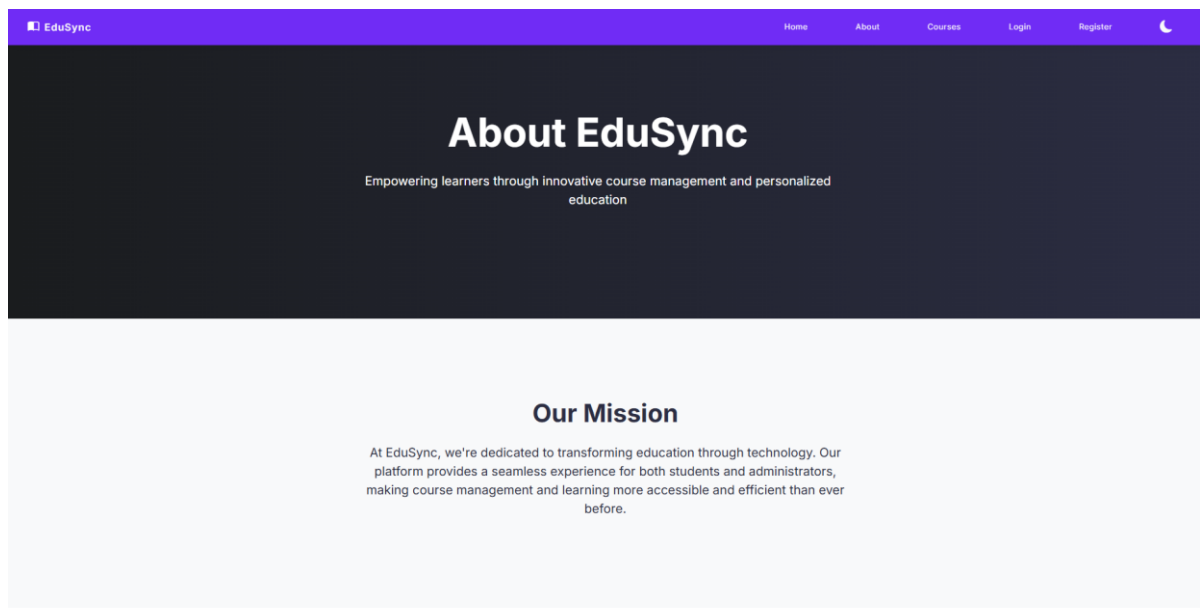


Fig. 13.1 Landing Page (Light Mode & Dark Mode Support Available)



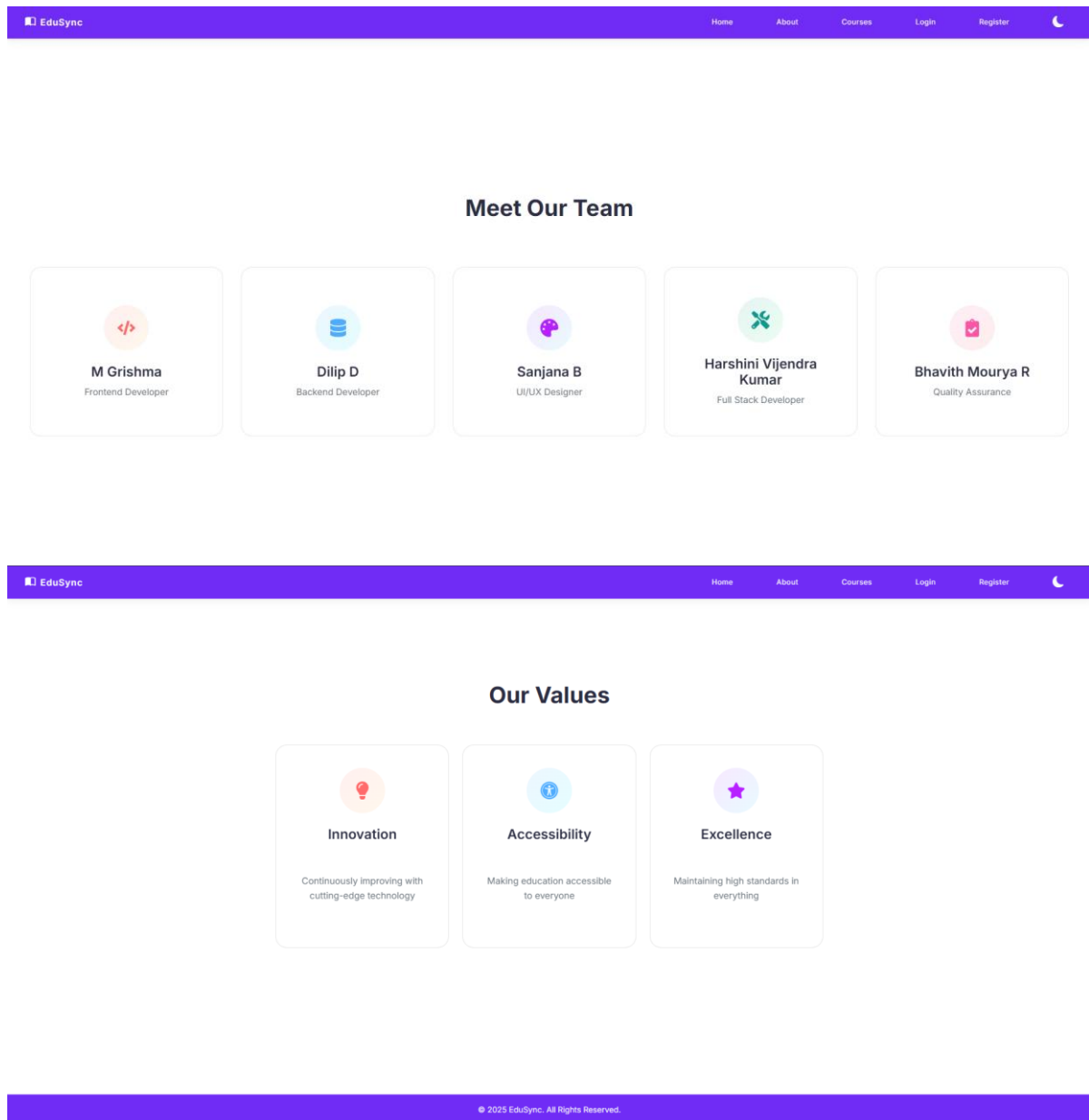


Fig. 13.2 About Page (Light Mode & Dark Mode Support Available)

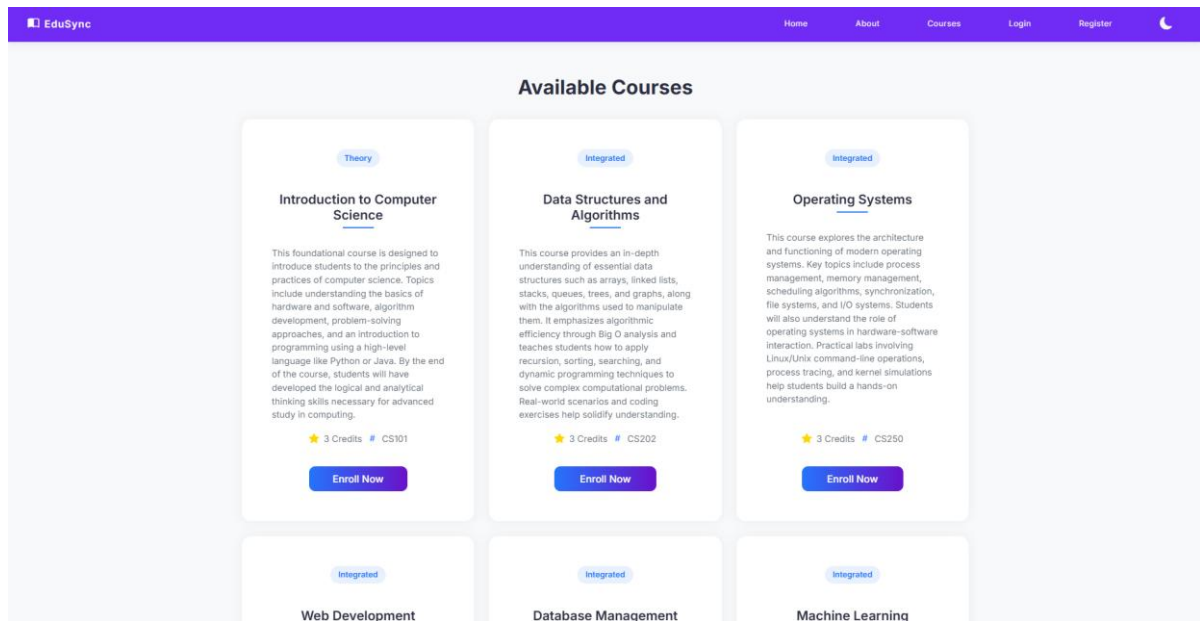


Fig. 13.3 View Courses Page (Light Mode & Dark Mode Support Available)

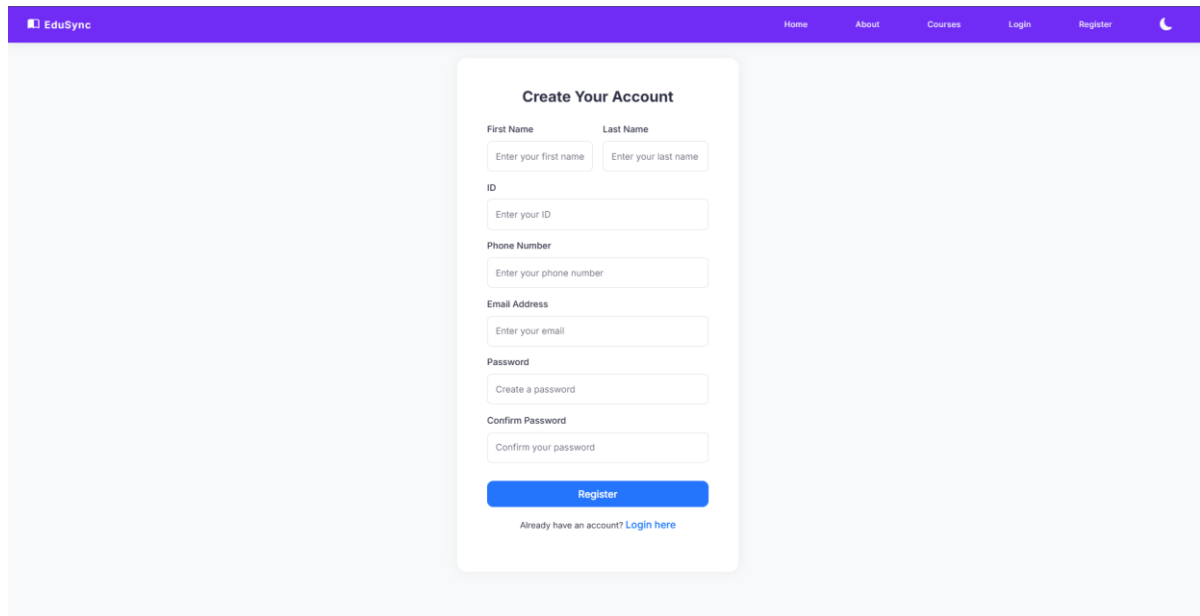
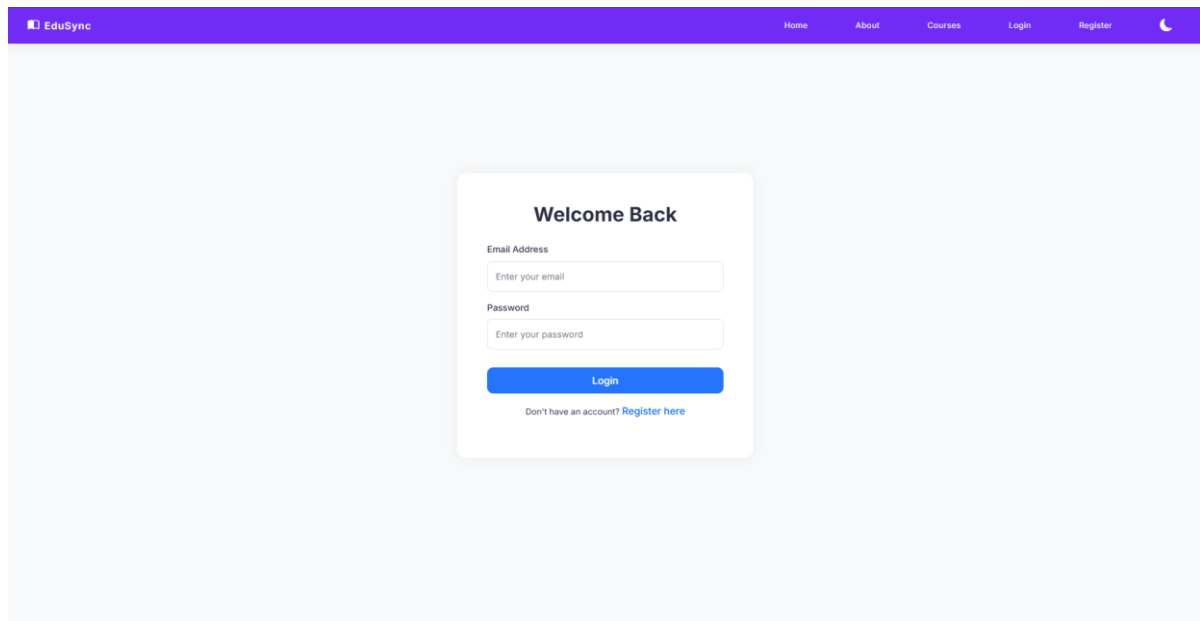
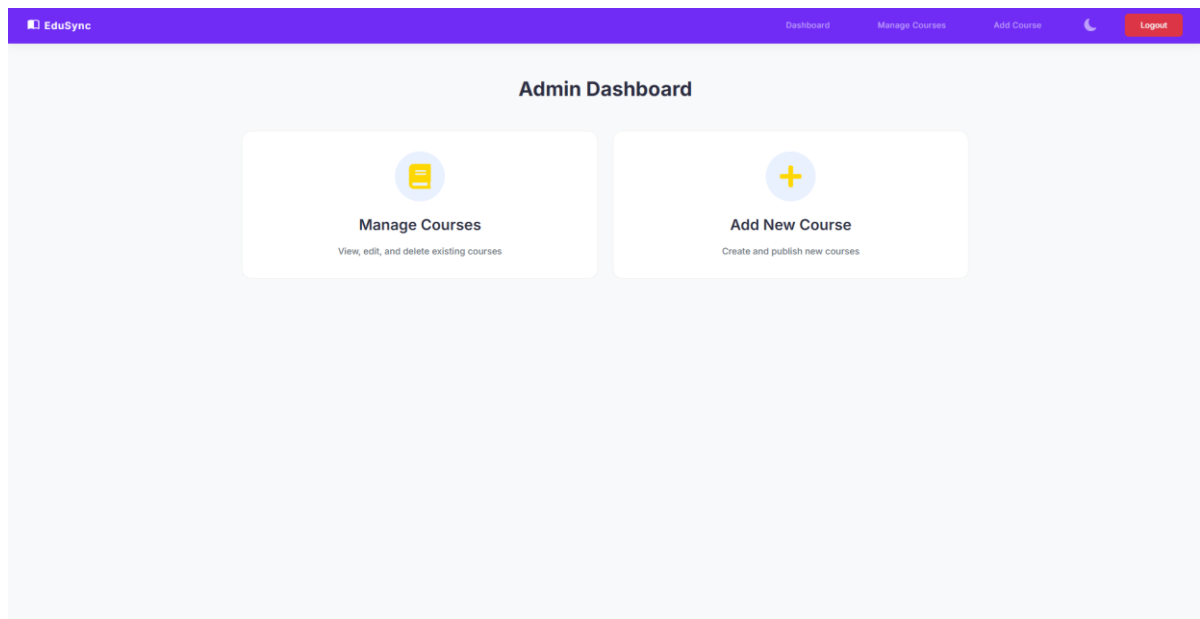


Fig. 13.4 Register Page (Light Mode & Dark Mode Support Available)



The screenshot shows the login page of the EduSync Course Management System. The header is a dark purple bar with the EduSync logo on the left and navigation links (Home, About, Courses, Login, Register) on the right. A central white card titled "Welcome Back" contains the login form. The form has two input fields: "Email Address" with the placeholder "Enter your email" and "Password" with the placeholder "Enter your password". Below these is a blue "Login" button. At the bottom of the card, there is a link: "Don't have an account? [Register here](#)".

Fig. 13.5 Login Page (Light Mode & Dark Mode Support Available)



The screenshot shows the Admin Dashboard of the EduSync Course Management System. The header is a dark purple bar with the EduSync logo on the left and navigation links (Dashboard, Manage Courses, Add Course) on the right. A red "Logout" button is also present. The main content area is titled "Admin Dashboard" and features two white cards. The first card, "Manage Courses", has a yellow icon of a document with a checkmark and the description "View, edit, and delete existing courses". The second card, "Add New Course", has a yellow plus icon and the description "Create and publish new courses".

Fig. 13.6 Admin Dashboard (Light Mode & Dark Mode Support Available)

ID	NAME	DESCRIPTION	CREDITS	TYPE	ACTIONS
CS101	Introduction to Computer Science	This foundational course is designed to introduce students to the principles and practices of computer science. Topics include understanding the basics of hardware and software, algorithm development, problem-solving approaches, and an introduction to programming using a high-level language like Python or Java. By the end of the course, students will have developed the logical and analytical thinking skills necessary for advanced study in computing.	3	Theory	View Details
CS202	Data Structures and Algorithms	This course provides an in-depth understanding of essential data structures such as arrays, linked lists, stacks, queues, trees, and graphs, along with the algorithms used to manipulate them. It emphasizes algorithmic efficiency through Big O analysis and teaches students how to apply recursion, sorting, searching, and dynamic programming techniques to solve complex computational problems. Real-world scenarios and coding exercises help solidify understanding.	3	Integrated	View Details
CS250	Operating Systems	This course explores the architecture and functioning of modern operating systems. Key topics include process management, memory management, scheduling algorithms, synchronization, file systems, and I/O systems. Students will also understand the role of operating systems in hardware-software interaction. Practical labs involving Linux/Unix command-line operations, process tracing, and kernel simulations help students build a hands-on understanding.	3	Integrated	View Details
CS305	Web Development	This course explores both front-end and back-end web development. Students learn to build responsive web pages using HTML5, CSS3, and JavaScript frameworks like React. On the server side, the course covers RESTful APIs, database integration, and security principles using technologies like Node.js or Spring Boot. By the end of the course, students will create a complete full-stack project, showcasing their ability to design, develop, and deploy a modern web application.	3	Integrated	View Details
	Database	This course delves into the design and implementation of database systems. Students learn the relational model, SQL, normalization techniques, and			

Fig. 13.7 Admin Manage Courses Page (Light Mode & Dark Mode Support Available)

S.NO.	STUDENT ID	NAME	EMAIL
1	2021ICSE0451	M Grishma	grishma@gmail.com
2	2021ICOM0068	Sanjana B	sanjana@gmail.com
3	2021ICSE0401	Dilip D	dilip@gmail.com
4	2021ICSE0394	Bhavith Mourya R	bhavith@gmail.com

Fig. 13.8 Admin Course Details Page (Light Mode & Dark Mode Support Available)


EduSync

[Dashboard](#)
[Manage Courses](#)


Course updated successfully

EduSync

DashboardManage CoursesAdd CourseLogout

Add New Course

Course ID

CAI001

Course Name

Artificial Intelligence and Machine Learning

Description

The Artificial Intelligence and Machine Learning syllabus teaches students about data collection, categorization, strategic planning, analysis, and interpretation. It is a specific field concerned with the development of


Credits

3


Course Type

Integrated

Add Course


EduSync

[Dashboard](#)
[Manage Courses](#)


Course added successfully!

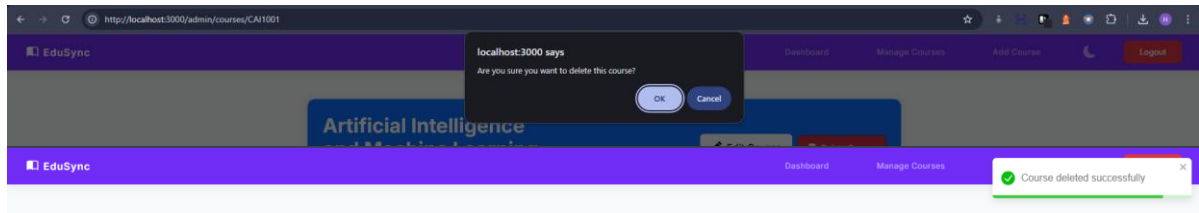


Fig. 13.13 Admin Course Deleted Notification (Light Mode & Dark Mode Support Available)

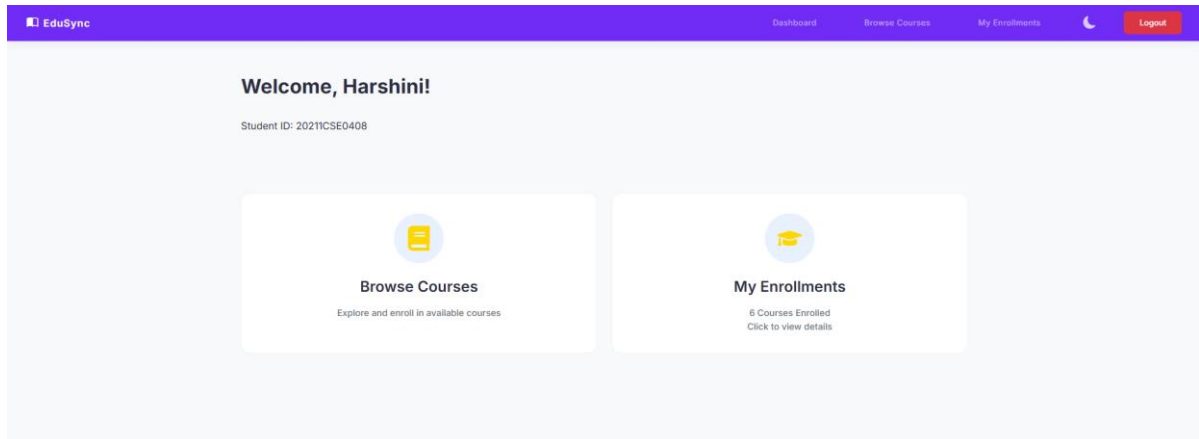


Fig. 13.14 Student Dashboard (Light Mode & Dark Mode Support Available)

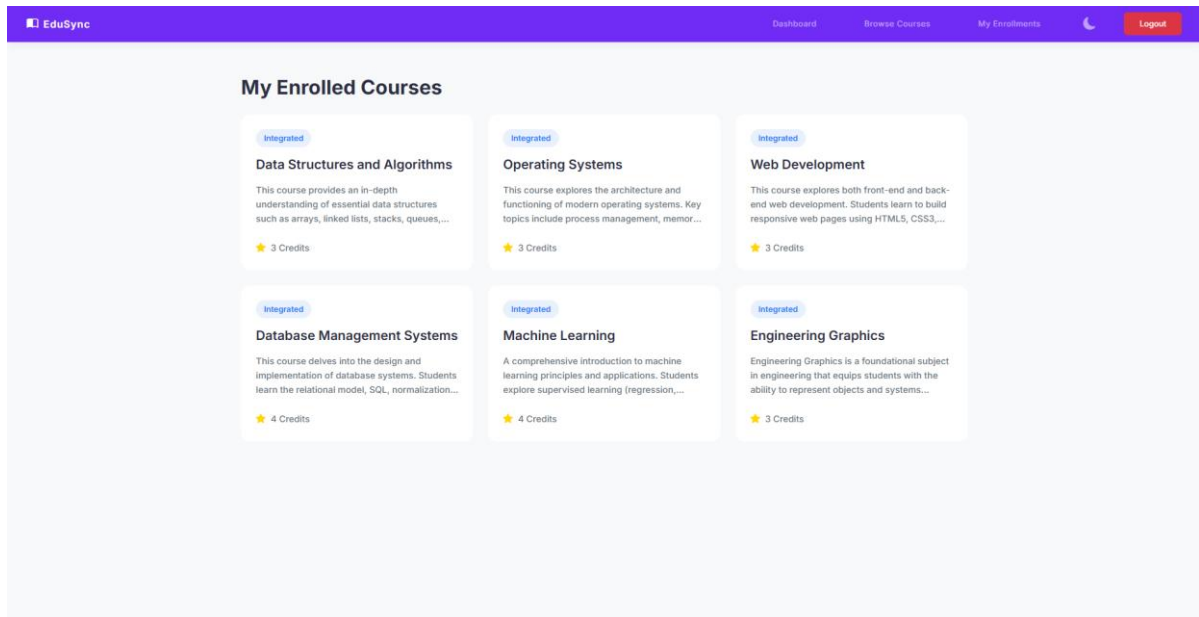


Fig. 13.15 Student Enrolled Courses (Light Mode & Dark Mode Support Available)

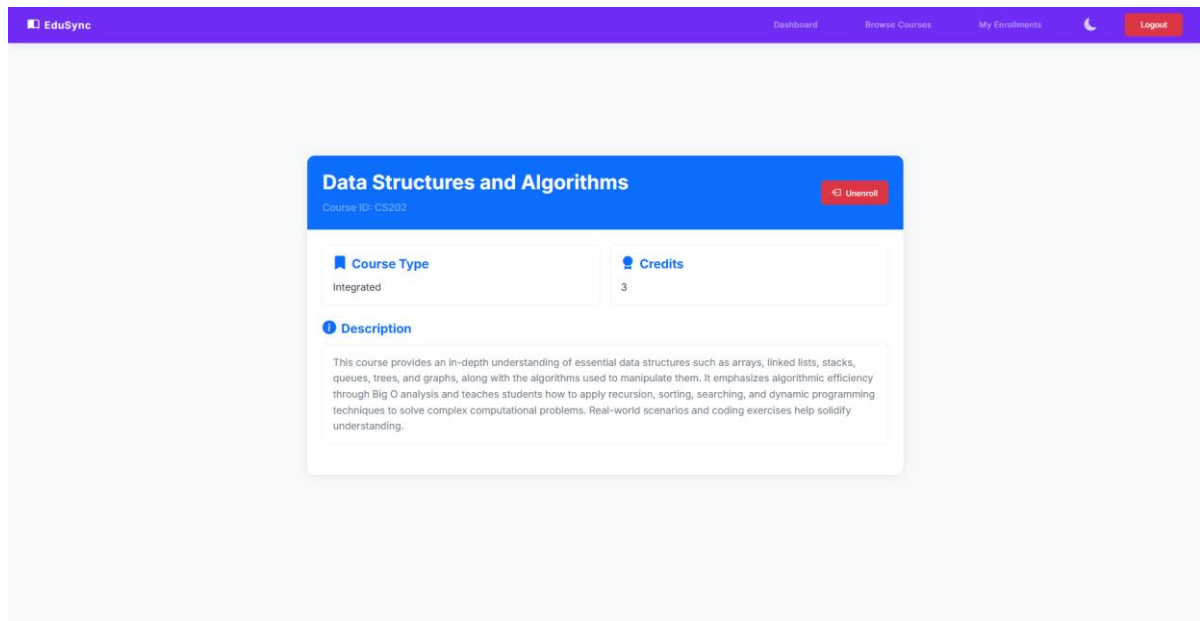


Fig. 13.16 Student Enrolled Course Details (Light Mode & Dark Mode Support Available)

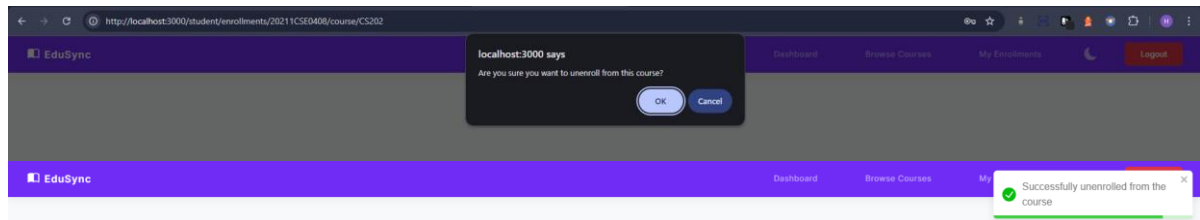


Fig. 13.17 Student Unenroll from Course Notification (Light Mode & Dark Mode Support Available)

APPENDIX-C

ENCLOSURES

Dr Jayanthi Kamalasekaran - report.pdf

ORIGINALITY REPORT

16%

SIMILARITY INDEX

11%

INTERNET SOURCES

9%

PUBLICATIONS

14%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	5%
2	Submitted to Symbiosis International University Student Paper	5%
3	Submitted to University of Nottingham Student Paper	2%
4	docshare.tips Internet Source	1%
5	en.wikipedia.org Internet Source	<1%
6	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
7	Submitted to Asia Pacific Institute of Information Technology Student Paper	<1%
8	www.coursehero.com Internet Source	<1%
9	Submitted to M S Ramaiah University of Applied Sciences Student Paper	<1%
10	Submitted to La Trobe University Student Paper	<1%
11	Sivaraj Selvaraj. "Mastering REST APIs", Springer Science and Business Media LLC,	<1%

2024

Publication

12	Submitted to Palm Beach State College Student Paper	<1 %
13	Submitted to University of Wisconsin-Whitewater Student Paper	<1 %
14	researchonline.lshtm.ac.uk Internet Source	<1 %
15	www.comparitech.com Internet Source	<1 %
16	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
17	scholar.ppu.edu Internet Source	<1 %
18	umpir.ump.edu.my Internet Source	<1 %
19	Submitted to University of Greenwich Student Paper	<1 %
20	www.ijosi.org Internet Source	<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography On

SUSTAINABLE DEVELOPMENT GOALS

SDG	Goal Title	Relevance to the Project
4	Quality Education	Centralizes course and subject access, improves transparency, and helps institutions deliver structured, equitable education digitally.
9	Industry, Innovation, and Infrastructure	Demonstrates how digital infrastructure like Spring Boot, React, and MySQL can enhance traditional education systems with innovation and scalability.
10	Reduced Inequalities	Can be deployed in underserved regions to provide equal access to course materials and educational services through the web.
16	Peace, Justice and Strong Institutions	Encourages accountability through secure, role-based access (JWT), logging of actions, and proper institutional workflows for managing student data.
17	Partnerships for the Goals	Being open-source, the CMS fosters collaboration between developers, institutions, and organizations to improve and scale digital learning platforms globally.

