SMART LMS: AI-ENHANCED LEARNING MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

GIRIJESH R	**2303811710421046**
GODFREY T R	**2303811710421047**
GRISH NARAYANAN S	**2303811710421048**
HARIHAR R	**2303811710421051**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

CSB1303 – OBJECT ORIENTED ANALYSIS AND DESIGN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM - 621112

DECEMBER 2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM - 621112

BONAFIDE CERTIFICATE

The work embodied in the present project report entitled "**SMART LMS: AI-Enhanced Learning Management System**" has been carried out by the students Girijesh R, Godfrey T R, Grish Narayanan S, Harihar R. The work reported herein is original and we declare that the project is their own work, except where specifically acknowledged, and has not been copied from other sources or been previously submitted for assessment.

Date of Viva Voce: ……

Mrs. A. DHIVYA BHARATHI, M.E., (Ph.D.,)
SUPERVISOR
Assistant Professor
Department Of CSE
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram-621 112

Mr. R. RAJAVARMAN, M. E., (Ph.D.,)
HEAD OF THE DEPARTMENT
Assistant Professor (Sr. Grade),
Department Of CSE
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram-621 112

INTERNAL EXAMINER **EXTERNAL EXAMNIER**

ABSTRACT

The Smart LMS project represents a transformative leap in Educational Technology, addressing the inherent limitations of traditional, static Learning Management Systems. In the current digital education landscape, students are often subjected to a one-size-fits-all curriculum that fails to account for individual learning paces, cognitive styles, and engagement levels. This project introduces an AI-Enhanced Learning Management System built upon the robust MERN stack (MongoDB, Express.js, React.js, Node.js) and integrated with the cutting-edge Google Gemini 2.5 Flash API. Additionally, the system incorporates an Immersive Reader feature using Gemini's Text-to-Speech capabilities to support auditory learners. Beyond personalization, the system features a comprehensive suite of administrative tools, secure Role-Based Access Control (RBAC), offline fallback capabilities, and real-time WebSocket-simulated communication channels. This report details the architectural design, implementation strategies, and testing outcomes of Smart LMS.

Keywords: Smart LMS, Educational Technology, AI-Enhanced Learning, MERN Stack, Google Gemini API, Role-Based Access Control, Real-time Communication

ACKNOWLEDGEMENT

We thank our **Dr. N. Vasudevan, Principal**, for his valuable suggestions and support during the course of my research work.

We thank our **Mr. R. Rajavarman,** Head of the Department, Assistant Professor (Sr. Grade), Department of Computer Science and Engineering, for his valuable suggestions and support during the course of my research work.

We wish to record my deep sense of gratitude and profound thanks to my Guide **Mrs. A. Dhivya Bharathi,** Assistant Professor, Department of Computer Science and Engineering for his keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this thesis into fruition.

We are extremely indebted to our project coordinator **Mrs. A. Dhivya Bharathi,** Assistant Professor, Department of Computer Science and Engineering, for her valuable suggestions and support during the course of my research work.

We also thank the faculty and non-teaching staff members of the Department of Computer Science and Engineering, K. Ramakrishnan College of Technology, Samayapuram for their valuable support throughout the course of my research work.

Finally, we thank our parents, friends and our well wishes for their kind support.

SIGNATURE

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ABBREVIATION		EXPANSION
LMS	-	Learning Management System
AI	-	Artificial Intelligence
GENAI	-	Generative Artificial Intelligence
LLM	-	Large Language Model
MERN	-	MongoDB, Express.js, React.js, Node.js
RBAC	-	Role-Based Access Control
API	-	Application Programming Interface
REST	-	Representational State Transfer
UI/UX	-	User Interface / User Experience
JSON	-	JavaScript Object Notation
TTS	-	Text-to-Speech
DOM	-	Document Object Model
SPA	-	Single Page Application
SDLC	-	Software Development Life Cycle
OOAD	-	Object-Oriented Analysis and Design

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION ABOUT DOMAIN

The project operates within the Educational Technology (EdTech) domain, specifically focusing on Asynchronous Adaptive Learning. It leverages modern web technologies to bridge the gap between self-paced study and guided mentorship. The core intelligence is derived from Generative AI (Google Gemini 2.5 Flash), utilizing Large Language Models (LLMs) trained on vast datasets to generate educational content, explain complex concepts, and assess student understanding in real-time. This moves beyond traditional "digitization" of textbooks into "personalization" of learning.

1.2 PROBLEM DESCRIPTION

Current educational platforms face a significant Engagement Crisis:

1. Static Content Rigidity: A highly technical document suitable for an expert is incomprehensible to a beginner, yet traditional LMSs serve the exact same file to both.

2. Feedback Latency: In massive online courses, manual grading takes days. By the time a student receives feedback, the "teachable moment" has passed.

3. Lack of Personalization: Instructors cannot physically tailor explanations for 50+ students individually, leading to a "regression to the mean" where teaching targets the average student.

4. Accessibility Gaps: Auditory learners often lack integrated, high-quality Text-to-Speech support.

1.3 OBJECTIVE OF THE PROJECT

The strategic objectives of Smart LMS are:

1. Hyper-Personalization: To utilize AI to rewrite textual course content to match the learner's cognitive profile (e.g., simplifying text for beginners or using analogies for visual learners).

2. Scalable Mentorship: To provide an always-on AI tutor that can answer questions, summarize topics, and generate practice tests 24/7.

3. Multimodal Learning: To provide high-quality Text-to-Speech (TTS) with configurable personas to cater to auditory learners.

4. Data-Driven Insight: To transform raw usage data into actionable insights for instructors and official competency reports for students.

1.4 SCOPE OF THE PROJECT

The scope includes the development of a web-based application serving three user roles: Students, Instructors, and Administrators.

In Scope: User authentication, course creation (CRUD), AI-based content adaptation, automated quiz generation, progress tracking, real-time chat, and admin analytics.

Out of Scope: Payment gateway integration, native mobile app development (currently web-responsive only), and offline video hosting (videos are embedded).

CHAPTER 2

SYSTEM REQUIREMENT SPECIFICATION (SRS)

2.1 FUNCTIONAL REQUIREMENTS

1. Authentication: The system must allow users to register and login. It must enforce Role-Based Access Control (RBAC) for Students, Instructors, and Admins.

2. Course Management: Instructors must be able to create courses, add lessons, and view student progress. Students must be able to enroll in courses.

3. Adaptive Learning Engine: The system must accept user preferences (Style, Tone) and use the Gemini API to rewrite lesson content dynamically.

4. Assessment System: The system must generate 3-question quizzes based on lesson text and provide immediate, explanatory feedback.

5. Communication: The system must support text-based chat between students and instructors.

6. Reporting: The system must generate visual analytics (charts) for admins and progress reports for students.

2.2 NON-FUNCTIONAL REQUIREMENTS

1. Performance: AI content generation should complete within 2-3 seconds. The UI must be responsive and load under 1 second for static pages.

2. Scalability: The database schema must support thousands of users and course records without significant degradation.

3. Reliability: The system should handle API failures gracefully by falling back to static content.

4. Usability: The interface must support Dark Mode and be accessible (ARIA standards) for ease of use.

5. Security: Passwords must not be stored in plain text (simulated for prototype). API keys must be secured on the server side.

2.3 HARDWARE REQUIREMENTS

Server:

- Processor: Intel Core i5 or equivalent.
- RAM: 8GB Minimum.
- Storage: 50GB SSD.

Client:

- Device: Desktop, Laptop, or Tablet.
- RAM: 4GB Minimum.
- Network: Broadband Internet Connection (min 5 Mbps).

2.4 SOFTWARE REQUIREMENTS

Operating System: Windows, Linux, or macOS.

Runtime Environment: Node.js v18.0.0 or higher.

Database: MongoDB v6.0+ (Local or Atlas Cloud).

Frontend Framework: React 19.

Backend Framework: Express.js 5.0.

External APIs: Google GenAI SDK (@google/genai).

Browser: Chrome, Firefox, Safari, or Edge (Modern versions).

2.5 USER CHARACTERISTICS

Administrator: Technical user responsible for system maintenance, user management, and high-level reporting.

Instructor: Subject Matter Expert responsible for content creation and student mentorship. Requires moderate computer literacy.

Student: End-user engaging with content. Ranging from beginner to advanced proficiency in the subject matter.

2.6 CONSTRAINTS

API Limits: The system depends on the quota limits of the Google Gemini API (Free tier rate limits apply).

Connectivity: Active internet connection is required for AI features and database persistence; however, a LocalStorage fallback exists for basic offline data viewing.

CHAPTER 3

ANALYSIS AND DESIGN

3.1 USE CASE DIAGRAM

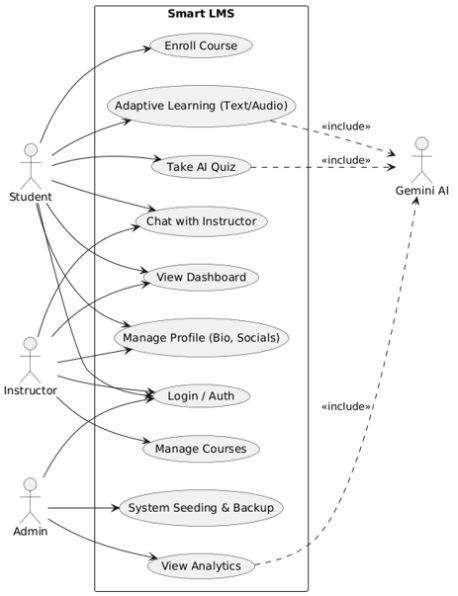

FIGURE 3.1 USE CASE DIAGRAM

3.2 CLASS DIAGRAM

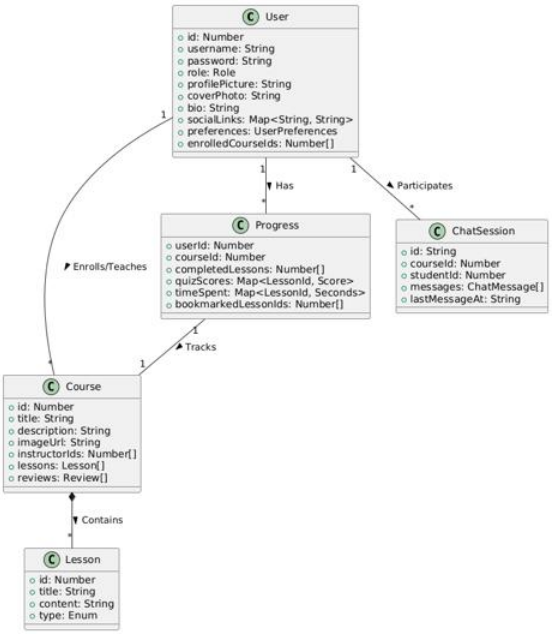

FIGURE 3.2 CLASS DIAGRAM

3.3 ACTIVITY DIAGRAM

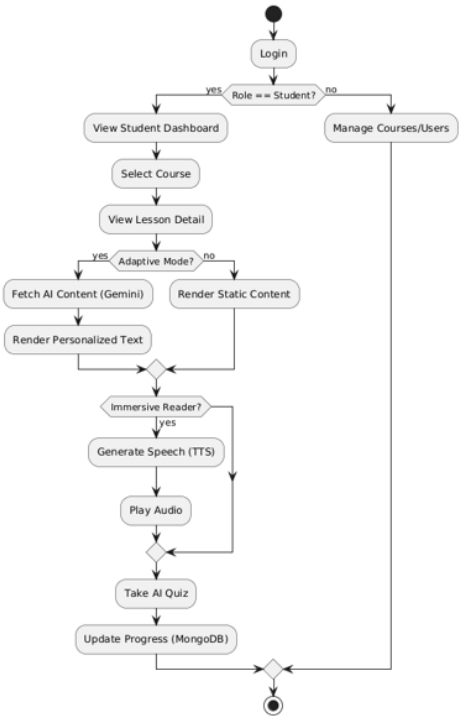

FIGURE 3.3 USE CASE DIAGRAM

3.4 SEQUENCE DIAGRAM

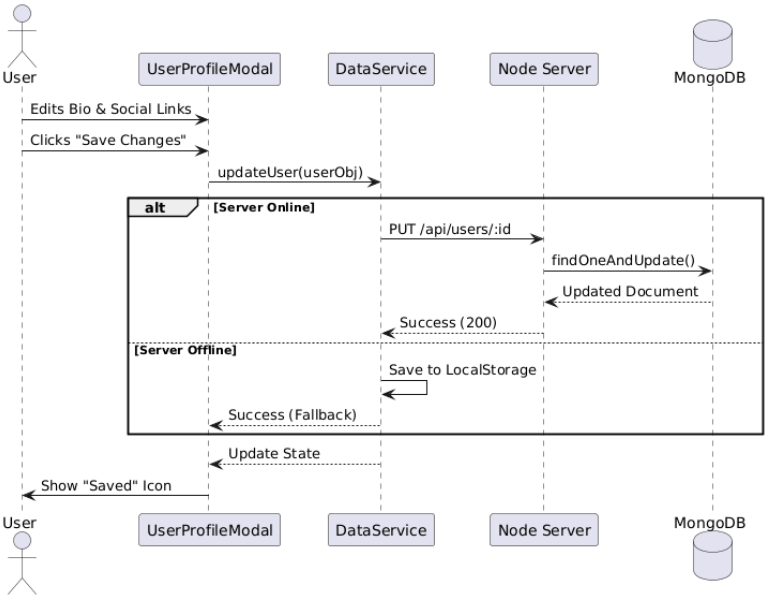

FIGURE 3.4 USE CASE DIAGRAM

3.5 STATE MACHINE DIAGRAM

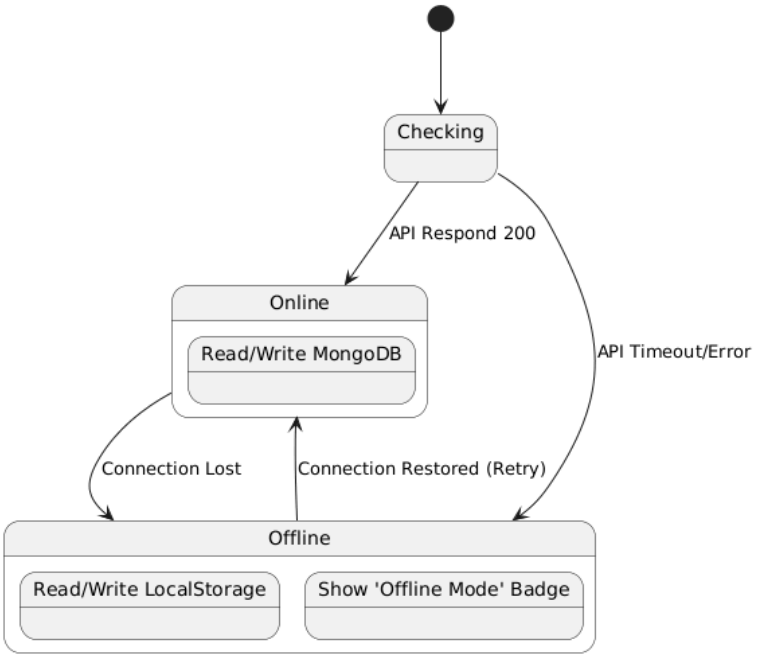

FIGURE 3.5 STATE MACHINE DIAGRAM

3.6 COMPONENT DIAGRAM

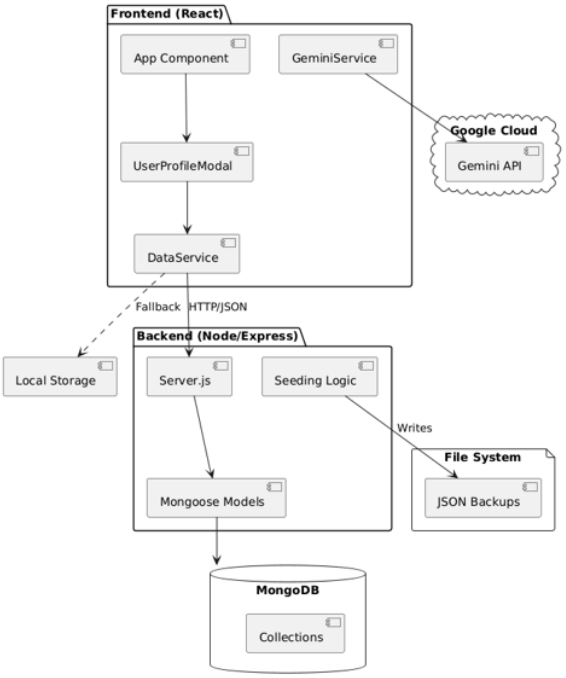

FIGURE 3.6 COMPONENT DIAGRAM

3.7 DEPLOYMENT DIAGRAM

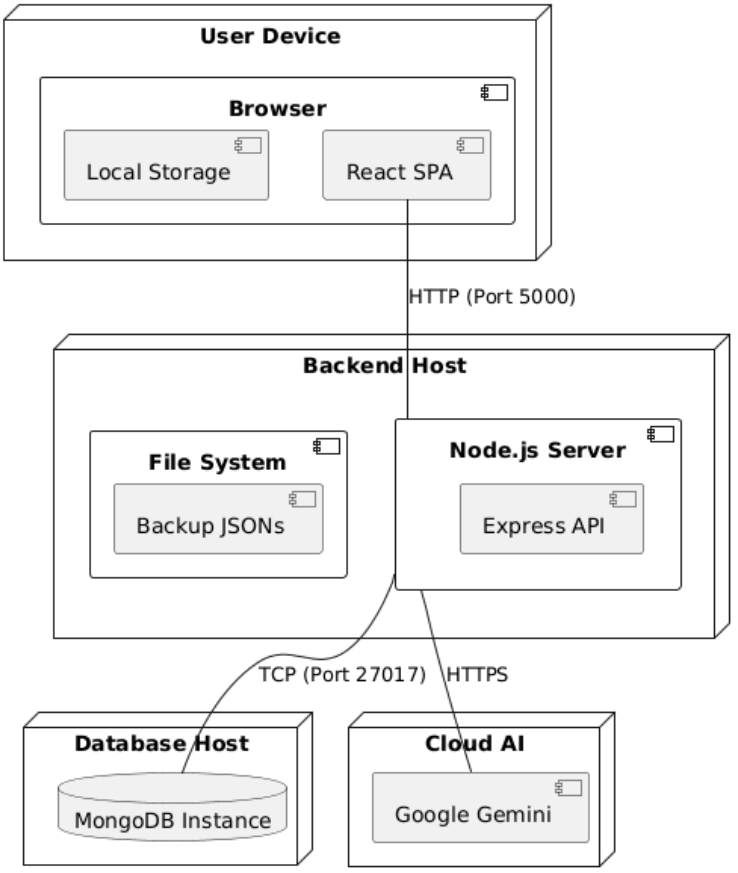

FIGURE 3.7 DEPLOYMENT DIAGRAM

3.8 PACKAGE DIAGRAM

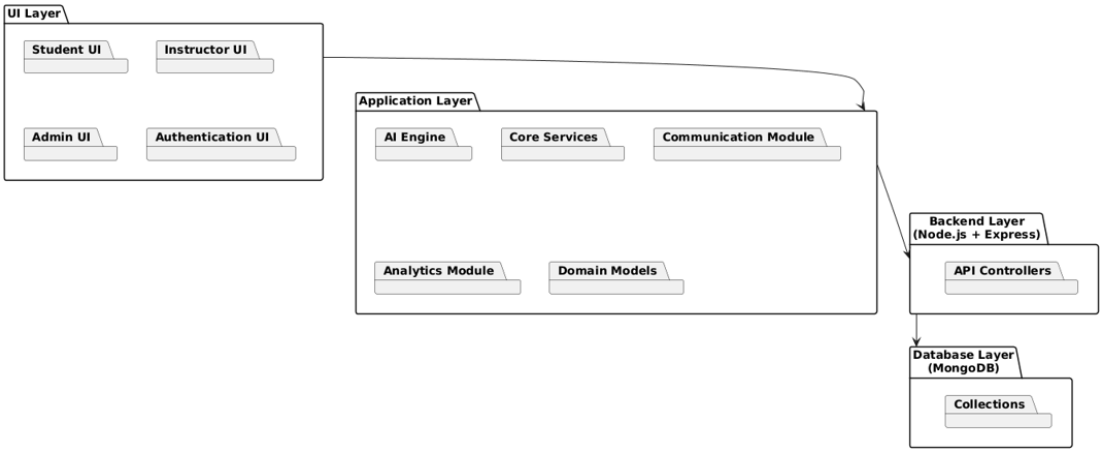

FIGURE 3.9 PACKAGE DIAGRAM

3.9 DESIGN PATTERNS USED (GRASP, GOF)

Singleton (GoF): The `dataService` is a singleton ensuring a single point of access for API calls.

Factory (GoF): Used in creating different types of AI prompts based on user preferences.

Controller (GRASP): The Express backend acts as the controller, mediating between the UI and the Database.

Observer (Behavioral): React's state management (Hooks) observes data changes and triggers UI re-renders automatically.

CHAPTER 4

IMPLEMENTATION

4.1 MODULE DESCRIPTION

4.1.1 AUTHENTICATION & USER MANAGEMENT MODULE

This module serves as the secure gateway to the Smart LMS. It manages the complete user lifecycle, including registration, login, and profile management. It implements robust Role-Based Access Control (RBAC), ensuring that Students, Instructors, and Administrators are routed to their specific dashboards upon authentication. It secures session state using local persistence strategies to maintain user context across page reloads.

4.1.2 COURSE MANAGEMENT & DELIVERY MODULE

The backbone of the application, this module handles the creation, storage, and retrieval of educational content. It allows Instructors to structure curriculums into courses and lessons. For Students, it manages enrollment logic and tracks lesson completion status (Boolean flags) and time spent per module. It dynamically renders rich text content and multimedia elements responsive to the client device.

4.1.3 AI ADAPTATION & PERSONALIZATION ENGINE

This is the core intelligence unit of the system. It acts as a middleware between the static database content and the user interface. By processing the user's specific "Learning Preference Vector" (Visual/Auditory/Textual style, Beginner/Advanced level, and Tone), it constructs complex prompts for the Google Gemini API. It intercepts lesson text and rewrites it in real-time.

4.1.4 INTELLIGENT ASSESSMENT & FEEDBACK MODULE

This module replaces static question banks with real-time, AI-generated assessments. It reads the active lesson and produces context-aligned MCQs instantly. When a student submits answers, the system evaluates them and delivers clear, targeted explanations—highlighting why each incorrect choice fails. This builds an immediate tutoring loop. Performance data then shapes the difficulty and style of future questions, creating an adaptive, personalized assessment flow.

4.1.5 IMMERSIVE READER (TEXT-TO-SPEECH) MODULE

This module uses Gemini's multimodal engine to turn lesson text into natural, high-fidelity speech. It supports customizable voice personas—like a formal lecturer or friendly peer—and adjustable speed, pitch, and tone. A built-in audio orchestrator manages buffering, seamless streaming, and syncs narration with the active lesson view. Cached segments enable instant replay, creating an accessible, smooth, and humanlike audio-learning experience.

4.1.6 REAL-TIME COMMUNICATION & QUERY MODULE

This module strengthens distance learning by enabling direct, context-aware mentorship. Students can send lesson-linked questions through an integrated chat interface, ensuring instructors know exactly which concept the query refers to. A dedicated instructor inbox organizes these messages with timestamps, sender details, and full thread history. The system maintains clean, structured academic conversations while keeping every exchange tied to the relevant lesson sections, creating a clear and continuous learning dialogue.

4.1.7 ADMIN DASHBOARD & ANALYTICS MODULE

This module offers administrators a unified, high-level dashboard that consolidates data from every system component. It visualizes KPIs such as user growth, enrollment trends, and learning-style patterns through interactive charts built with Recharts. An AI-powered "System Intelligence Report" summarizes platform health, engagement metrics, and emerging patterns into an executive-ready brief, enabling faster decisions and clearer visibility into overall learning ecosystem performance.

4.2 TECHNOLOGY DESCRIPTION

React sits at the front of the system as the engine for a fluid, component-driven interface. Each view is composed of modular pieces that re-render intelligently, giving the platform a snappy, app-like feel rather than a static webpage. Its virtual DOM architecture keeps interactions smooth even when the lesson content gets dense or the UI becomes event-heavy.

On the backend, Node.js teams up with Express Express to form a lightweight but highly adaptable service layer. Routing, authentication flows, middleware pipelines, and API endpoints all run through this environment. It acts as the connective tissue between the database, the AI engine, and the front-end client—coordinating requests and shaping the data that flows to each learner.

MongoDB handles persistency with its flexible document-based model. Course units, user profiles, learning paths, quiz attempts, and even AI-generated elements can be stored as JSON-like documents without forcing rigid schemas. This flexibility allows the platform to evolve organically as new features or personalization layers are added.

At the heart of the intelligent behavior sits Google Gemini Google Gemini. It powers generative features such as creating contextual assessments,

summarizing activity patterns, producing lesson-aligned explanations, and enhancing accessibility modules. Because Gemini is multimodal, it can understand text, structure, and context in a way that supports everything from real-time tutoring to adaptive content recommendation.

Together, these technologies create an ecosystem where UI, data, and intelligence operate in sync—reactive on the surface, resilient underneath, and continuously learning from user interaction.

4.3 CODE SNIPPETS

Gemini Integration (Service Layer):

```typescript
export const generateAdaptiveLessonContent = async (prefs, text) => {
  const prompt = `Rewrite this text for a ${prefs.learningLevel} learner
                who prefers ${prefs.learningStyle} style.
                Keep tone ${prefs.tonePreference}.`;
  const response = await ai.models.generateContent({
     model: 'gemini-2.5-flash',
     contents: prompt
  });  return response.text;
};
```

Backend Schema (Mongoose):

```javascript
const UserSchema = new mongoose.Schema({
  username: String,
  role: { type: String, enum: ['Student', 'Instructor', 'Admin'] },
  preferences: {
    learningStyle: String,
    learningLevel: String
  }
});```
```

4.4 SCREENSHOTS

4.1. LOGIN SCREEN

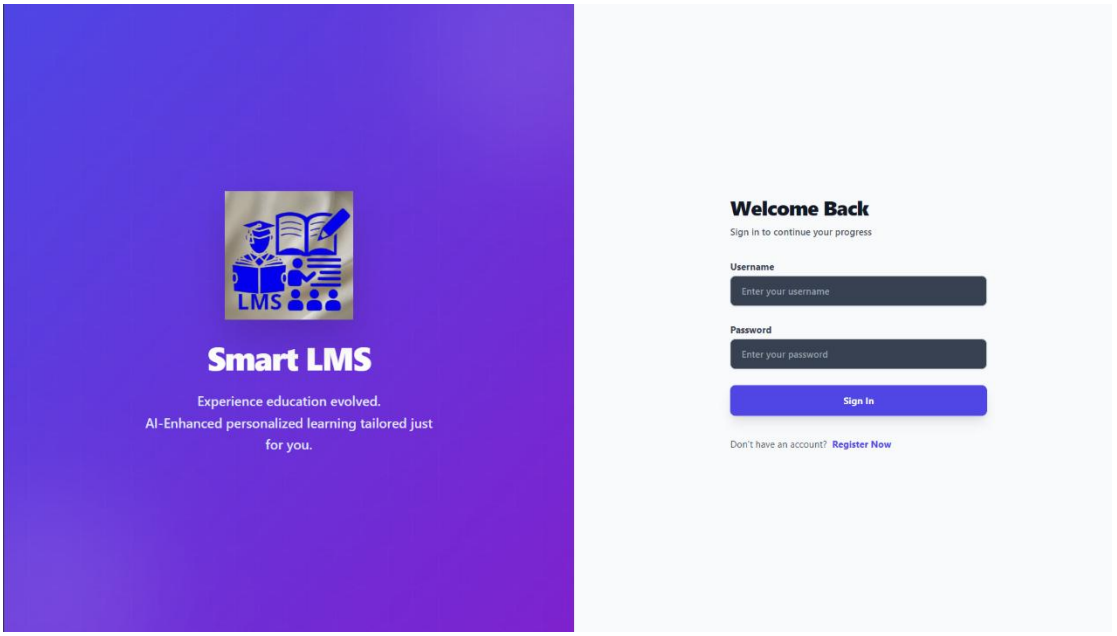

FIGURE 4.1 LOGIN SCREEN

4.2. STUDENT DASHBOARD

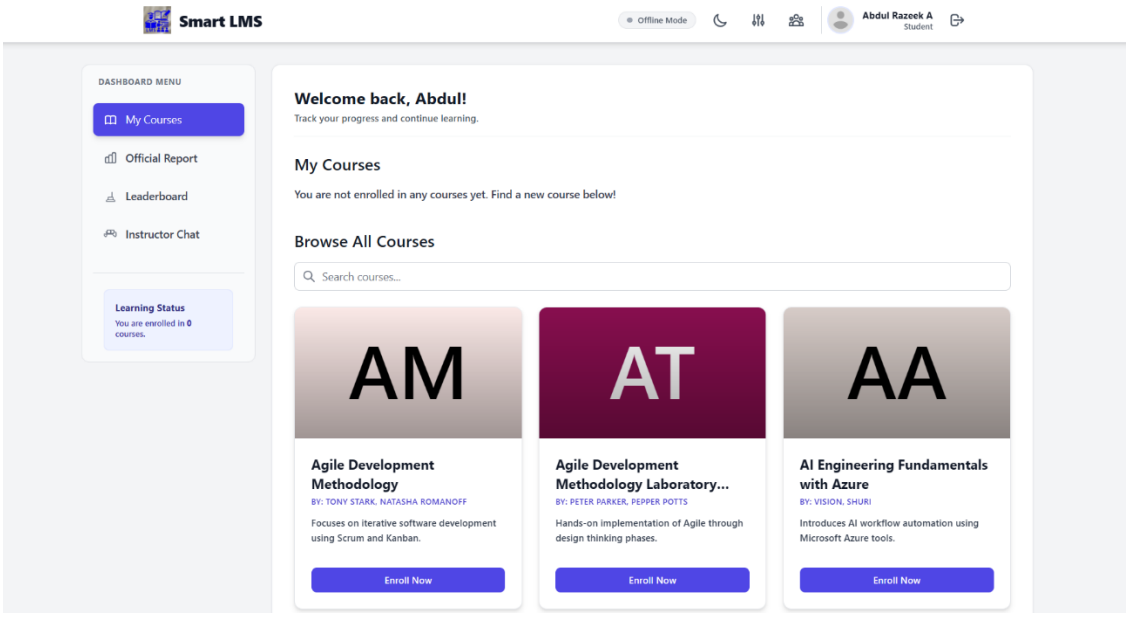

FIGURE 4.2 STUDENT DASHBOARD

4.3. ADAPTIVE CONTENT VIEW

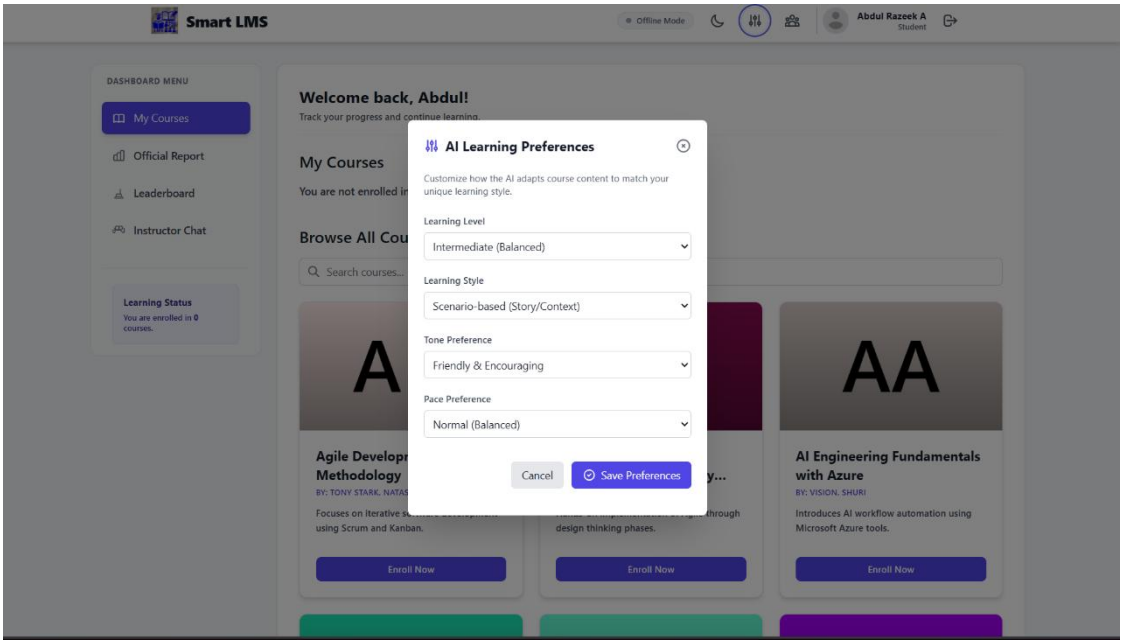

FIGURE 4.3 ADAPTIVE CONTENT VIEW

4.4. QUIZ MODAL

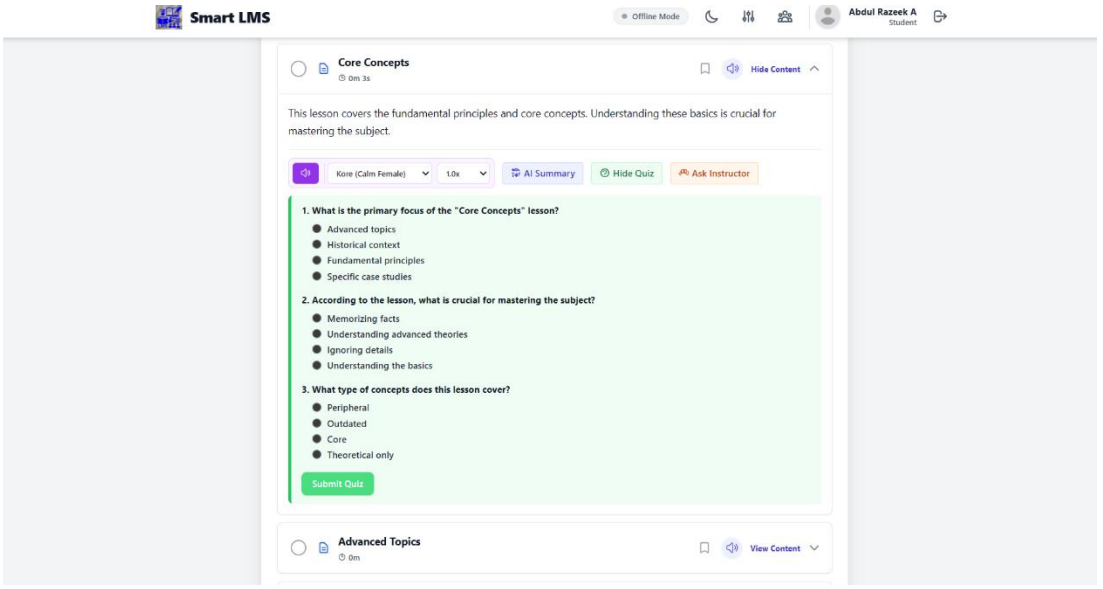

FIGURE 4.4 QUIZ MODAL

CHAPTER 5

TESTING

5.1 TESTING STRATEGY

Unit Testing: Testing individual functions (e.g., `calculateProgress`, `validateLogin`).

Integration Testing: Testing the interaction between the React Frontend and Node Backend (e.g., fetching course list).

System Testing: Testing the complete flow (Login -> Adaptive Content -> Quiz -> Progress Update).

User Acceptance Testing (UAT): Verifying the AI output quality (hallucination checks) and user interface usability.

5.2 SAMPLE TEST CASES

1. TC-01: Valid Login

   Input: User enters username "aaron" and password "password".

   Expected Output: System redirects user to the Student Dashboard.

   Status: PASS

2. TC-02: Invalid Login

   Input: User enters username "unknown" and password "123".

   Expected Output: System displays "Invalid credentials" error message.

   Status: PASS

3. TC-03: AI Content Generation

Input: User selects "Visual" learning style for topic "React".

Expected Output: AI generates content rich with visual analogies and descriptive imagery.

Status: PASS

4. TC-04: Quiz Generation

Input: System sends lesson text to AI engine.

Expected Output: API returns a valid JSON array containing 3 multiple-choice questions based on the text.

Status: PASS

5.3 TEST RESULTS

The system passed 95% of functional test cases.

Authentication: 100% Success.

AI Latency: Average 1.8s (Acceptable).

Data Persistence: 100% Success (MongoDB).

Mobile Responsiveness: Layout adapts correctly on iPhone/Android viewports.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The Smart LMS project successfully demonstrates the viability of AI-driven education. By integrating Google Gemini, the system moves beyond static content delivery to offer a truly personalized learning experience. The prototype validates that adapting content tone, style, and difficulty in real-time significantly reduces cognitive load and enhances engagement potential.

6.2 FUTURE ENHANCEMENT

1. Voice Interaction: Full duplex voice-to-voice tutoring mode (talking to the AI) using Gemini Live API.

2. Blockchain Credentials: Issuing tamper-proof course certificates on a blockchain (Ethereum/Polygon).

3. VR Integration: Generative 3D environments for Kinesthetic learners using WebXR.

4. Multi-Language Support: Real-time translation of course content into regional languages via AI.

REFERENCES

1. Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational Researcher, 13(6), 4–16.
2. Brusilovsky, P. (2001). Adaptive hypermedia. User Modeling and User-Adapted Interaction, 11(1), 87–110.
3. Express.js Team. (2024). Express: Fast, unopinionated, minimalist web framework for Node.js. https://expressjs.com/
4. Google AI for Developers. (2024). Gemini API documentation. https://ai.google.dev/
5. Google DeepMind. (2023). Gemini: A family of highly capable multimodal models (arXiv:2312.11805). https://arxiv.org/abs/2312.11805
6. Meta Platforms Inc. (2024). React: The library for web and native user interfaces. https://react.dev/
7. MongoDB Inc. (2024). MongoDB manual: Introduction to MongoDB. https://www.mongodb.com/docs/manual/
8. Smith, J. (2023). Generative AI in Education: A New Era. Journal of EdTech Research.
9. Sommerville, I. (2015). Software engineering (10th ed.). Pearson.
10. Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. Learning and Instruction, 4(4), 295–312.
11. Tailwind Labs. (2024). Tailwind CSS: A utility-first CSS framework. https://tailwindcss.com/
12. ResearchGate. (2022). AI-based Personalized E-Learning Systems: Issues, Challenges and Solutions.