**AI ENHANCED E-LEARNING**

By

**ENJOY SHESHE**

**(H210212Q)**

HIT400 Capstone project Submitted in Partial Fulfillment of the

Requirements of the degree of

Bachelor of Technology

In

**Software Engineering**

In the

**School of Information Sciences and Technology**

Harare Institute of Technology

Zimbabwe



Supervisor

Mrs S Zindove

May/2025

# ABSTRACT

The continuous evolution of digital education has presented new opportunities and challenges in delivering personalized learning experiences. This project presents the design and implementation of an AI-enhanced e-learning platform that offers personalized course recommendations, adaptive learning paths, and intelligent performance tracking. By integrating content-based filtering, keyword analysis, and cosine similarity, the system dynamically tailors content suggestions based on a learner's interests and quiz performance. The platform supports course enrollment, quiz-based assessments, performance feedback, and administrative control for user and course management.

Through structured interviews, questionnaires, and observations, system requirements were gathered and translated into functional software components. The platform was developed using PHP (Laravel), JavaScript, and RubixML for AI functionalities. Testing procedures including unit, integration, system, and non-functional testing confirmed the reliability, usability, and scalability of the solution. Results showed improved learner engagement and relevance of course delivery. This system contributes toward the goal of personalized education and sets a foundation for further research in intelligent tutoring and adaptive learning systems.

# PREFACE

This document presents the culmination of my undergraduate final year project for the Bachelor of Technology (BTech) degree in Software Engineering at Harare Institute of Technology. The goal of this project was to explore how Artificial Intelligence (AI) can be effectively applied to personalize and improve the learning process in digital environments.

Throughout this journey, I focused on identifying practical challenges students face in existing e-learning systems, such as lack of personalization, static content delivery, and low engagement. The proposed solution is a comprehensive AI-enhanced e-learning platform that leverages user interaction and performance data to recommend appropriate content, track learning progress, and enhance learner autonomy.

This report covers all stages of the project lifecycle—from problem identification and requirement analysis, to system design, implementation, and testing. It is intended for academic evaluators, researchers in e-learning and AI, software engineers, and anyone interested in personalized digital education.

# ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to God for granting me the strength, patience, and clarity to complete this work.

I extend my heartfelt appreciation to my supervisor, Mrs S Zindove, for their guidance, encouragement, and constructive feedback throughout the course of this project. Their expertise in software engineering and educational technologies significantly shaped the quality of this research.

I am also thankful to the faculty and staff of Harare Institute of Technology for providing a conducive environment for learning and research.

Special thanks go to the students and lecturers who participated in the interviews and questionnaires, offering valuable insights into the challenges of current e-learning platforms.

Finally, I would like to thank my family and friends for their unwavering support, love, and motivation during this journey. Your encouragement has been my foundation.



This is to certify that HIT 400 Project entitled “**AI Enhanced E-Learning” has** been completed by **Sheshe Enjoy** (your reg number) for partial fulfilment of the requirements for the award of **Bachelor of Technology** degree in **Software Engineering**. This work is carried out by **him** under my supervision and has not been submitted earlier for the award of any other degree or diploma in any university to the best of my knowledge.

|  |  |
| --- | --- |
| **Your Supervisor Name** | **Approved/Not Approved** |
| Project Supervisor | Project Coordinator |
| **Signature: ……………………………** | **Signature: …………………………………** |
| **Date: ……………………………………** | **Date: ……………………………………….** |



**Certificate of Declaration**

This is to certify that work entitled “AI ENHANCED E-LEARNING” is *submitted in partial fulfillment of the requirements for the award of Bachelor of Technology (Hons) in Software Engineering, Harare Institute of Technology. It is further certified that no part of research has been submitted to any university for the award of any other degree.*

(Supervisor) Signature…………………………….. Date……………………….

(Mentor ) Signature…………………………….. Date………………………

(Chairman) Signature……………………………….. Date………………………..

**Project Documentation Marking Guide**

|  |  |  |
| --- | --- | --- |
| **ITEM** | **TOTAL MARK /%** | **ACQUIRED/%** |
| **PRESENTATION-**  Format-Times Roman 12 for ordinary text, Main headings Times Roman 14, spacing 1.5. Chapters and sub-chapters, tables and diagrams should be numbered. Document should be in report form. Range of document pages. Between 50 and 100.Work should be clear and neat | **5** |  |
| **Pre-Chapter Section**  Abstract, Preface, Acknowledgements, Dedication & Declaration | **5** |  |
| **Chapter One-Introduction**  Background, Problem Statement, Objectives – smart, clearly measurable from your system. Always start with a TO…  Hypothesis, Justification, Proposed Tools  Feasibility study: Technical, Economic & Operational  Project plan –Time plan, Gantt chart | **10** |  |
| **Chapter Two-Literature Review**  Introduction, Related work & Conclusion | **10** |  |
| **Chapter Three –Analysis**  Information Gathering Tools, Description of system  Data analysis –Using UML context diagrams, DFD of existing system  Evaluation of Alternatives Systems, Functional Analysis of Proposed System-Functional and Non-functional Requirements, User Case Diagrams | **15** |  |
| **Chapter Four –Design**  Systems Diagrams –Using UML Context diagrams, DFD, Activity diagrams  Architectural Design-hardware, networking  Database Design –ER diagrams, Normalized Databases  Program Design-Class diagrams, Sequence diagrams, Package diagrams, Pseudo code  Interface Design-Screenshots of user interface | **20** |  |
| **Chapter Five-Implementation & Testing**  Pseudo code of major modules /Sample of real code can be written here  Software Testing-Unit, Module, Integration, System, Database & Acceptance | **20** |  |
| **Chapter Six –Conclusions and Recommendations**  Results and summary, Recommendations & Future Works | **10** |  |
| **Bibliography –Proper numbering should be used**  Appendices –templates of data collection tools, user manual of the working system, sample code, research papers | **5** |  |
|  | **100** | **/100** |

Table of Contents

[ABSTRACT ii](#_Toc199854786)

[PREFACE iii](#_Toc199854787)

[ACKNOWLEDGEMENT iv](#_Toc199854788)

[Chapter One – Introduction 1](#_Toc199854789)

[1.1 Background 1](#_Toc199854790)

[1.2 Problem Statement 1](#_Toc199854791)

[1.3 Objectives 1](#_Toc199854792)

[1.4 Hypothesis 2](#_Toc199854793)

[1.5 Justification 2](#_Toc199854794)

[1.6 Proposed Tools 2](#_Toc199854795)

[The development of the system utilized the following tools and frameworks: 2](#_Toc199854796)

[Minimum Hardware Requirements: 3](#_Toc199854797)

[1.7 Feasibility Study 3](#_Toc199854798)

[1.7.1 Technical Feasibility 3](#_Toc199854799)

[1.7.2 Economic Feasibility 3](#_Toc199854800)

[1.7.3 Operational Feasibility 3](#_Toc199854801)

[1.8 Project Plan 4](#_Toc199854802)

[1.8.1 Time Plan 4](#_Toc199854803)

[1.8.2 Gantt Chart 5](#_Toc199854804)

[2 Chapter Two – Literature Review 6](#_Toc199854805)

[2.1 Introduction 6](#_Toc199854806)

[2.2 Related Work 6](#_Toc199854807)

[2.2.1 MoocRec – A MOOC Recommendation System 6](#_Toc199854808)

[2.2.2 Coursera & edX – Collaborative Filtering in Practice 7](#_Toc199854809)

[2.2.3 Udemy 7](#_Toc199854810)

[2.3 Evaluation Summary 8](#_Toc199854811)

[2.4 Conclusion 8](#_Toc199854812)

[3 Chapter Three – Analysis 10](#_Toc199854813)

[3.1 Information Gathering Tools 10](#_Toc199854814)

[ Interviews: 10](#_Toc199854815)

[ Questionnaires: 10](#_Toc199854816)

[ Observation: 10](#_Toc199854817)

[3.2 Description of the System 11](#_Toc199854818)

[3.2.1 User Registration and Authentication 11](#_Toc199854819)

[3.2.2 Course Enrollment and Content Access 11](#_Toc199854820)

[3.2.3 Quiz-Based Assessments 11](#_Toc199854821)

[3.2.4 Intelligent Recommendation Engine 12](#_Toc199854822)

[3.2.5 Admin Dashboard 12](#_Toc199854823)

[3.3 Data Analysis 13](#_Toc199854824)

[3.3.1 UML Use case Of Existing System 13](#_Toc199854825)

[3.3.2 DFD of Existing System 13](#_Toc199854826)

[4 Chapter Four – Design 16](#_Toc199854827)

[4.1 Introduction 16](#_Toc199854828)

[4.2 Proposed System 16](#_Toc199854829)

[4.3 System Inputs 16](#_Toc199854830)

[4.4 System Processes 17](#_Toc199854831)

[4.5 System Outputs 17](#_Toc199854832)

[4.6 System Diagrams 18](#_Toc199854833)

[4.6.1 ER Diagram (Entity Relationship Diagram) 18](#_Toc199854834)

[DFD level O 18](#_Toc199854835)

[DFD Level 1 19](#_Toc199854836)

[4.4.5 Use Case Diagram of Proposed System 20](#_Toc199854837)

[4.4.6 UML class Diagram 21](#_Toc199854838)

[4.4.7 Normalized Database 22](#_Toc199854839)

[4.4.8 UML Sequence Diagram 23](#_Toc199854840)

[4.4.9 System Architecture 24](#_Toc199854841)

[4.4.10 Network Diagram 24](#_Toc199854842)

[4.4.11 Activity Diagram 25](#_Toc199854843)

[4.5 Pseudocode 26](#_Toc199854844)

[4.6 Interface Design 29](#_Toc199854845)

[4.6.1 Login Page 29](#_Toc199854846)

[4.6.2 Register Form 30](#_Toc199854847)

[4.6.3 Student Dashboard 31](#_Toc199854848)

[4.6.4 Student Course Enrolled 32](#_Toc199854849)

[4.6.5 Grades and Performance 33](#_Toc199854850)

[4.6.6 Quizzes and Progress per Each Course 34](#_Toc199854851)

[4.6.7 Admin Dashboard 35](#_Toc199854852)

[4.6.8 Add and Manage Users 36](#_Toc199854853)

[5 Chapter Five – Implementation & Testing 37](#_Toc199854854)

[5.1 Introduction 37](#_Toc199854855)

[5.2 Coding Strategy 37](#_Toc199854856)

[5.3 Coding Conventions 37](#_Toc199854857)

[5.4 Naming Conventions 38](#_Toc199854858)

[5.5 Pseudocode of Major Modules 38](#_Toc199854859)

[5.6 Sample of Real Code 40](#_Toc199854860)

[5.6.1 Design Interfaces for Courses 40](#_Toc199854861)

[5.6.2 Course Recommendation Logic 44](#_Toc199854862)

[5.6.3 Quiz Logic 48](#_Toc199854863)

[5.6.4 Student Dashboard Logic 56](#_Toc199854864)

[5.7 Code Review 57](#_Toc199854865)

[5.8 Code Structure Checklist 57](#_Toc199854866)

[5.9 Software Testing 58](#_Toc199854867)

[5.9.1 Unit Testing 58](#_Toc199854868)

[5.9.2 Module Testing 60](#_Toc199854869)

[5.9.3 Integration Testing 60](#_Toc199854870)

[5.9.3 Integration Testing Scenarios 61](#_Toc199854871)

[5.9.4 System Testing 63](#_Toc199854872)

[5.8.5 Coverage Testing 64](#_Toc199854873)

[5.9.6 Database Testing 65](#_Toc199854874)

[5.10 Non-Functional Testing 66](#_Toc199854875)

[5.10.1 Performance Testing 66](#_Toc199854876)

[5.10.2 Usability Testing 66](#_Toc199854877)

[5.10.3 Compatibility Testing 67](#_Toc199854878)

[5.10.4 Reliability and Availability Testing 67](#_Toc199854879)

[6 Chapter Six – Conclusions and Recommendations 68](#_Toc199854880)

[6.1 Summary of Results 68](#_Toc199854881)

[6.1 Objectives and Their Achievement 68](#_Toc199854882)

[6.2 Conclusions 69](#_Toc199854883)

[6.3 Recommendations 69](#_Toc199854884)

[6.4 Future Work 70](#_Toc199854885)

[7 Bibliography 71](#_Toc199854886)

[8 Appendices 73](#_Toc199854887)

[User Manual 75](#_Toc199854888)

[Sample Code 77](#_Toc199854889)

[Student Dashboard Interface 77](#_Toc199854890)

**List of Tables**

[Table 1: Proposed Development Tools 2](#_Toc199850718)

[Table 2: Hardware Requirements 3](#_Toc199850719)

[Table 3: Project Time Plan 4](#_Toc199850720)

[Table 4: Evaluation Summary 8](#_Toc199850721)

[Table 5: System Inputs 16](#_Toc199850722)

[Table 6: System Processes 17](#_Toc199850723)

[Table 7: System Outputs 17](#_Toc199850724)

[Table 8: Naming Conventions 38](#_Toc199850725)

[Table 9: Code Structure Checklist 57](#_Toc199850726)

[Table 10: Unit Testing 58](#_Toc199850727)

[Table 11: Integration Testing Scenarios 61](#_Toc199850728)

[Table 12: System Test Cases 63](#_Toc199850729)

[Table 13: Code Coverage Summary 65](#_Toc199850730)

[Table 14: Performance Testing 66](#_Toc199850731)

[Table 15: Results Summary 68](#_Toc199850732)

**Table of Figures**

[Figure 1: Gantt Chart 5](#_Toc199798321)

[Figure 2: Use Case of Existing System 13](#_Toc199798322)

[Figure 3: DFD of Existing System 13](#_Toc199798323)

[Figure 4: DFD level 0 of Proposed System 18](#_Toc199798324)

[Figure 5: DFD Level 1 of Proposed System 19](#_Toc199798325)

[Figure 6: Use Case of Proposed System 20](#_Toc199798326)

[Figure 7: Class Diagram of Proposed system 21](#_Toc199798327)

[Figure 8: Normalised Schema of Proposed System 22](#_Toc199798328)

[Figure 9: Sequence diagram of Proposed System 23](#_Toc199798329)

[Figure 10: Architecture of Proposed system 24](#_Toc199798330)

[Figure 11: Log in form 29](#_Toc199798331)

[Figure 12: Register Form 30](#_Toc199798332)

[Figure 13: Student Dashboard 31](#_Toc199798333)

[Figure 14: Student Courses Enrolled 32](#_Toc199798334)

[Figure 15: Grades and Performance 33](#_Toc199798335)

[Figure 16: Quizzes per course 34](#_Toc199798336)

[Figure 17: Admin Dashboard 35](#_Toc199798337)

[Figure 18: Add and Manage Users 36](#_Toc199798338)

[Figure 19: Reports per course topic 36](#_Toc199798339)

# Chapter One – Introduction

## 1.1 Background

The rapid growth of digital education has opened new opportunities for personalized learning. However, many traditional e-learning platforms adopt a one-size-fits-all approach, offering generic content and static course pathways. This leads to disengagement, low completion rates, and limited adaptation to individual learner needs. Artificial Intelligence (AI) has the potential to revolutionize e-learning by enabling platforms to analyze user interests, performance, and engagement patterns in real time to provide personalized learning experiences.

This project presents the design and development of an **AI-enhanced e-learning system** that recommends courses based on a user’s interests and quiz performance. The system uses content-based and logic-based filtering, machine learning (via RubixML), and performance tracking to improve learner outcomes.

## 1.2 Problem Statement

Most e-learning platforms do not tailor course recommendations to the specific interests or performance patterns of individual learners. This often leads to information overload, disengagement, and suboptimal learning outcomes. There is a need for an intelligent system that can analyze user preferences and results to recommend suitable learning paths and topics dynamically.

## 1.3 Objectives

The main objective of this project is to develop a smart e-learning system that uses artificial intelligence to enhance user experience and learning efficiency. The specific objectives are:

1. To build an AI system that recommends personalized courses based on user interests.
2. To implement an adaptive learning path based on quiz performance.
3. To track learner progress and provide smart feedback for improvement.

## 1.4 Hypothesis

If learner interests and performance data are used as inputs for AI-driven recommendations, then the resulting learning paths will lead to higher engagement, better quiz scores, and improved course completion rates compared to static course delivery.

## 1.5 Justification

This project is justified by the increasing demand for personalized learning experiences and the limitations of current static e-learning systems. AI can bridge this gap by enabling content to adapt dynamically to learner profiles. Implementing such a solution aligns with modern educational trends and improves the quality of learning through relevance and engagement.

## 1.6 Proposed Tools

### The development of the system utilized the following tools and frameworks:

Table 1: Proposed Development Tools

|  |  |
| --- | --- |
| **Tool/Framework** | **Purpose** |
| PHP (Laravel) | Backend API and logic implementation |
| Python (RubixML) | Machine learning and recommendation engine |
| JavaScript (Chart.js) | Visualizing student progress and analytics |
| MySQL | Database management system |
| Blade (Laravel) | Front-end templating |
| HTML/CSS, Tailwind | User interface styling |

### Minimum Hardware Requirements:

Table 2: Hardware Requirements

|  |  |
| --- | --- |
| **Component** | **Specifications** |
| Mobile Device | Android version 8 and above |
| Mobile RAM | At least 2 GB |
| Mobile Storage | At least 4 GB |

## 1.7 Feasibility Study

Feasibility studies and analysis were carried out to make sure if the intended project could be delivered with the time constraints, budget and of quality. The main aim was to evaluate alternatives in terms of performances, functionalities, and inconsistences and also to determine whether the final product will benefit end users.

### 1.7.1 Technical Feasibility

The system is technically feasible using mature, well-documented, and widely supported tools such as Laravel, Python, and MySQL. RubixML allows machine learning capabilities in PHP without complex infrastructure.

### 1.7.2 Economic Feasibility

All core tools and libraries used are open-source, reducing development and licensing costs. The project can be deployed on low-cost hosting environments, making it cost-effective for small institutions.

### 1.7.3 Operational Feasibility

The system is designed to be user-friendly, requiring minimal training for both students and administrators. It can be seamlessly integrated into existing academic processes, enhancing rather than disrupting current operations.

## 1.8 Project Plan

The project followed a structured timeline divided into five phases:

1. **Requirement Analysis & Planning** – Gathering system requirements and creating user stories.
2. **Design** – Creating database schemas, system architecture, and wireframes.
3. **Implementation** – Developing the frontend, backend, and integrating machine learning.
4. **Testing** – Performing unit, integration, and acceptance tests.
5. **Documentation & Deployment** – Writing user guides and deploying the system.

### 1.8.1 Time Plan

Table 3: Project Time Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Start | Finish | Duration (days) |
| Requirement Analysis & Planning | 29 Aug 2024 | 30 Sep 2024 | 33 |
| Design | 1 Oct 2024 | 31 Oct 2024 | 31 |
| Implementation | 1 Nov 2024 | 31 Jan 2025 | 92 |
| Testing | 1 Feb 2025 | 15 Mar 2025 | 43 |
| Documentation & Deployment | 16 Mar 2025 | 25 Apr 2025 | 41 |

### 1.8.2 Gantt Chart

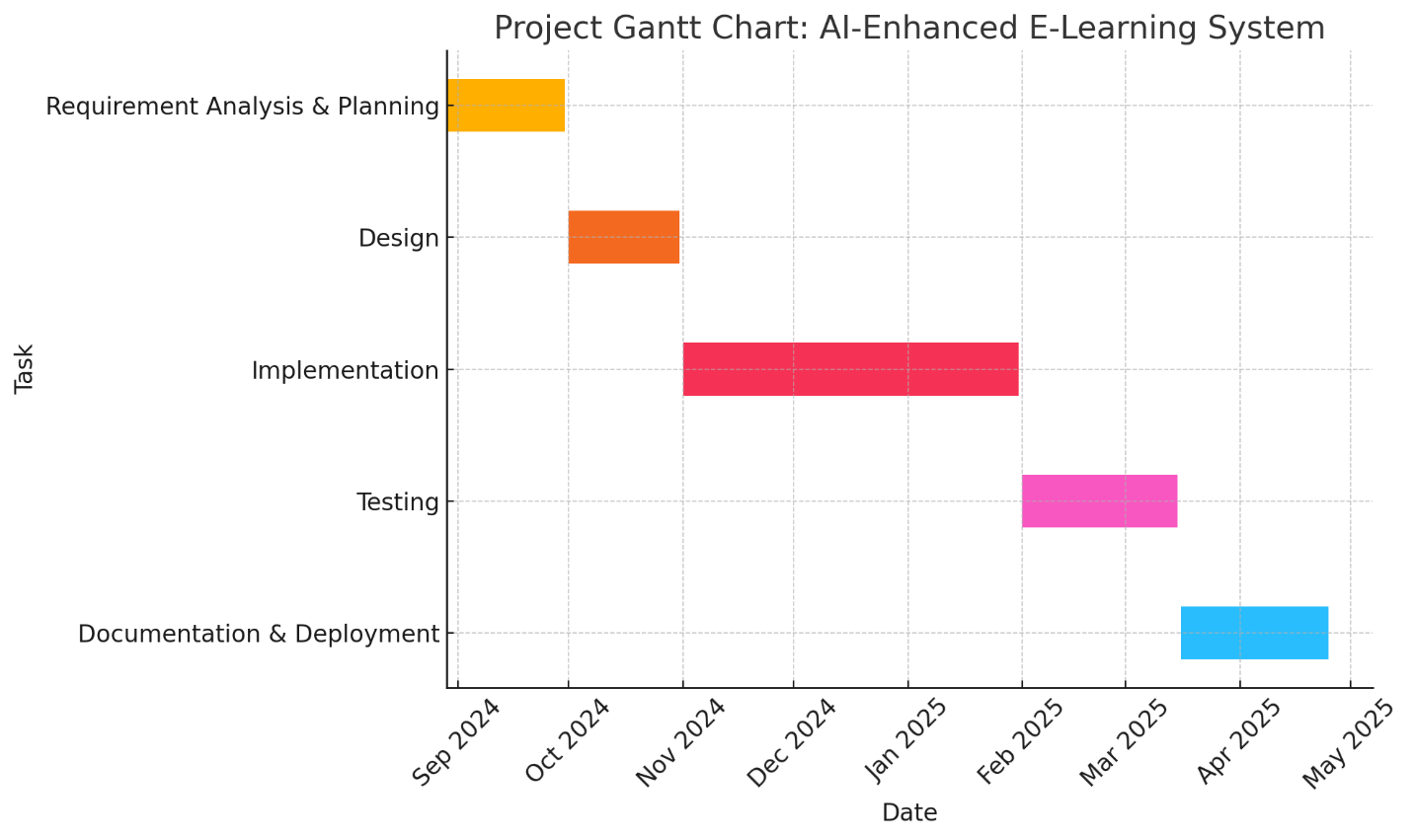


Figure 1: Gantt Chart

# 2 Chapter Two – Literature Review

## 2.1 Introduction

Artificial Intelligence (AI) is playing an increasingly important role in transforming education, particularly in the field of e-learning. As digital learning platforms expand, the need for adaptive, personalized, and data-driven systems has become crucial. This chapter explores existing research and systems that have attempted to address these challenges using AI techniques. The aim is to contextualize the proposed system and highlight gaps that it seeks to fill.

## 2.2 Related Work

### 2.2.1 MoocRec – A MOOC Recommendation System

**MoocRec** utilizes a **content-based filtering** approach that leverages student profiles (interests, academic background) and course metadata. It uses **TF-IDF** to vectorize course descriptions and **cosine similarity** to match users with relevant courses.

* **Strengths:**
  + Works well when user behavior data is limited.
  + Highly interpretable recommendations based on course content.
  + Scalable to new users (cold start-friendly).
* **Weaknesses:**
  + Limited personalization—doesn’t consider user performance or preferences over time.
  + Ignores collaborative patterns or trends from other users.

### 2.2.2 Coursera & edX – Collaborative Filtering in Practice

These platforms implement **collaborative filtering**, which recommends courses based on user behavior (e.g., courses taken, ratings given, progress). Some use **matrix factorization** and **neural embeddings** to handle large-scale data.

* **Strengths:**
  + Learns hidden patterns and user preferences automatically.
  + Highly personalized recommendations after enough usage data is collected.
* **Weaknesses:**
  + Cold start problem—doesn’t work well for new users with no history.
  + Requires extensive data storage and processing power.

2.2.3 Udemy

Udemy is a marketplace-style e-learning platform where anyone can create and sell courses. It offers a wide range of topics and is especially friendly to freelancers and self-learners.

**Strengths:**

* **Open Content Model:** Anyone can become an instructor, leading to a vast and diverse course library.
* **Affordable:** Frequent discounts and one-time payments make learning accessible.

**Limitations:**

* **No Adaptive Learning:** Courses are static and not personalized based on learner performance.
* **No Smart Feedback:** Lacks AI-driven feedback, intelligent tutoring, or personalized assessments.
* **Inconsistent Quality:** Course quality varies due to the open publishing model.
* **Not Accredited:** Certifications are not formally recognized by academic institutions.

## 2.3 Evaluation Summary

Table 4: Evaluation Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| System | Technique Used | Personalization | Adaptivity | Smart Feedback | Suitability |
| MoocRec | Content-Based Filtering | Available (Limited) | Not Available | Not Available | Moderate |
| Coursera | Collaborative Filtering | Available (High) | Limited | Basic | High |
| Udemy | Marketplace, Manual Navigation | Not Available | Not Available | Not Available | General Use |
| Proposed (AI E-Learning) | Hybrid + Performance-based | Available (High) | Available | Available | High |

## 2.4 Conclusion

The future of AI-driven education lies in **adaptive, intelligent systems** that go beyond static content delivery. The proposed AI-enhanced e-learning system takes a step in this direction by addressing critical gaps identified in existing platforms:

* **Hybrid Recommendation Engine**: Combines **interest-based keyword matching** with **performance-driven similarity analysis** using **RubixML**, enabling more relevant and timely course suggestions.
* **Dynamic Personalization**: Learners receive **tailored content recommendations** based on both what they like and how they perform, improving learning pathways.
* **Seamless Web Integration**: The recommendation engine is **embedded within the web platform**, ensuring a smooth and responsive user experience.
* **Improved Engagement & Outcomes**: The system is designed to **increase learner engagement**, **boost completion rates**, and **enhance satisfaction** through targeted support.

By integrating both personal interests and actual learning performance, this system redefines how content is delivered in online education. It empowers learners with **smarter, personalized pathways**, helping them stay motivated, complete courses more efficiently, and gain more from their learning experience.

# 3 Chapter Three – Analysis

## 3.1 Information Gathering Tools

To develop a relevant and user-focused AI-enhanced e-learning platform tailored for freelancers and independent learners, a combination of qualitative and quantitative research methods was used. These methods aimed to capture user needs, learning habits, pain points, and expectations from modern e-learning systems:

* Interviews:  
  Semi-structured interviews were conducted with freelancers from various domains (e.g., design, programming, digital marketing) to understand their goals, learning preferences, and expectations. Many expressed the need for flexible learning paths, real-time feedback, and skills-based recommendations aligned with current job market demands. Interviews also revealed that freelancers often prefer platforms that support modular learning, portfolio development, and certification for client credibility. (*See Appendix 1*)
* Questionnaires:  
  Online surveys were distributed through freelancing communities and social platforms. The questionnaires focused on common e-learning frustrations (e.g., irrelevant content, lack of guidance), platform preferences, and desired features in an AI-powered learning environment. A significant portion of respondents emphasized the value of personalized course suggestions, progress tracking, and interactive learning over static video-based content. (*See Appendix 2*)
* Observation:  
  Direct observation was conducted by analyzing how freelancers interact with popular platforms like Udemy, Skillshare, and LinkedIn Learning. This included watching tutorial usage patterns, note-taking behavior, how users engage with assessments, and how they navigate course catalogs. The observations highlighted issues such as choice overload, low engagement due to non-adaptive content, and difficulty finding relevant next steps after completing a course.

## 3.2 Description of the System

The proposed system is a **web-based AI-enhanced e-learning platform** designed to serve freelancers and independent learners by delivering **personalized, performance-driven educational experiences**. It combines standard e-learning functionalities with intelligent recommendations based on both user interests and assessment results. The system is modular and scalable, built to accommodate content across various fields like technology, design, business, and marketing.

The core features and components include:

### 3.2.1 User Registration and Authentication

* Users can **sign up and log in** securely using email and password or third-party integrations (e.g., Google).
* Role-based access control differentiates between **learners** and **administrators**.
* User profiles include basic information, learning preferences, skill interests, and progress history.

### 3.2.2 Course Enrollment and Content Access

* Learners can **browse and search for courses** using keyword.
* Enrolled users can track their **learning status**, **resume from where they left off**, and **bookmark important topics**.

### 3.2.3 Quiz-Based Assessments

* Each course module includes **quizzes or self-check assessments** to evaluate the learner’s understanding.
* Quizzes are automatically graded and results stored in the user's learning profile.
* Performance data is used to **refine recommendations** and track mastery over time.

### 3.2.4 Intelligent Recommendation Engine

* The core innovation of the system, this engine provides **real-time, personalized course suggestions** by combining:
  + **Logic-based keyword matching** to align user-declared interests with course tags.
  + **Cosine similarity using RubixML** to compare user quiz performance and behavior against other learners to suggest similar or advanced topics.
* Recommendations adapt as the user engages with the platform, creating a **dynamic learning path**.

### 3.2.5 Admin Dashboard

* A dedicated interface for administrators to:
  + **Monitor user registrations, quiz results, and course enrollments**.
  + **Manage course content**, categories, and instructor assignments.
  + Generate **analytics reports** on platform usage, engagement rates, and popular content.
  + View learner feedback and identify areas for system improvement.

## 3.3 Data Analysis

## 3.3.1 UML Use case Of Existing System

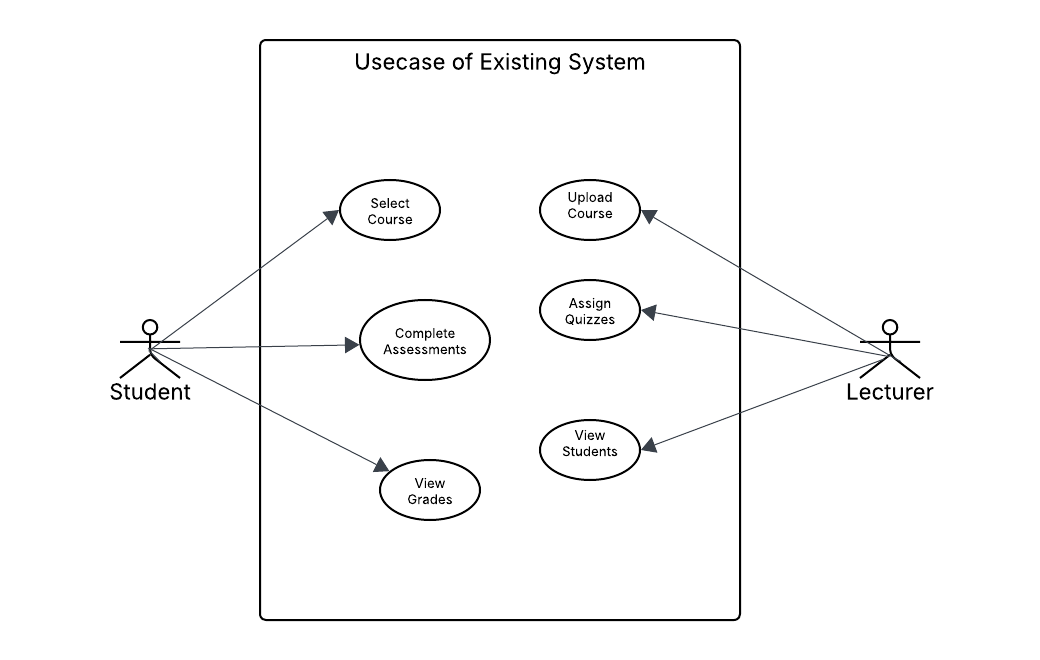


Figure 2: Use Case of Existing System

## 3.3.2 DFD of Existing System

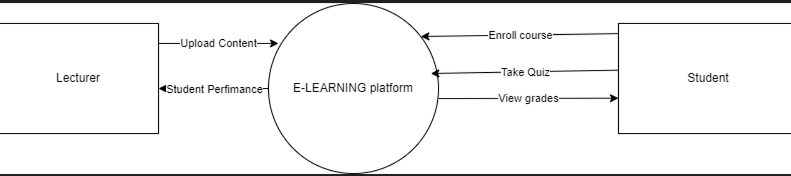
****

Figure 3: DFD of Existing System

**3.4 Functional Analysis of Proposed System**

**3.4.1 Functional Requirements**

The proposed e-learning system must fulfill the following key functions to support both students and administrative users:

* **User Registration and Login**  
  Users must be able to register and authenticate into the platform securely. This enables the system to identify individual users, apply access control, and maintain personalized learning data.
* **View and Enroll in Courses**  
  Registered students should be able to browse available courses, view their descriptions, and enroll in those relevant to their academic or personal learning interests.
* **Take Quizzes**  
  For each topic, students should be able to take assessments designed by instructors. These quizzes help evaluate student understanding and are critical in driving the AI recommendation engine.
* **Receive Topic/Course Recommendations**  
  The system must intelligently suggest new courses or topics using keyword matching and cosine similarity algorithms. These recommendations are generated based on user quiz performance and interests.
* **Admin Course and Topic Management**  
  Admins must be able to add, edit, or remove courses and topics. This includes uploading course content, assigning quizzes, and maintaining structured learning paths.
* **View User Statistics and Analytics**  
  Admins should have access to visualized user data, such as performance reports, enrollment metrics, and quiz success rates, enabling better decision-making and system improvement.

**3.4.2 Non-Functional Requirements**

The quality attributes of the system ensure it functions reliably and efficiently beyond just the core features:

* **Performance**  
  The system must respond to user actions—such as login, quiz submission, or navigation—within 3 seconds. This ensures a smooth user experience and reduces drop-off rates.
* **Scalability**  
  As the user base and course content grow, the platform must continue to perform well. The system architecture must support the seamless addition of new features and increased database size.
* **Usability**  
  The platform should be intuitive and easy to use for both students and admins. Clean UI design, consistent navigation, and clear visual cues are essential for reducing learning curves.
* **Security**  
  Data protection is critical. All passwords must be hashed, communications secured via HTTPS, and user roles enforced to prevent unauthorized access. This safeguards both personal data and academic integrity.

# 4 Chapter Four – Design

## 4.1 Introduction

This chapter presents the detailed design of the AI-enhanced e-learning platform, which focuses on course and topic recommendations based on learner interests and quiz performance. The goal is to create a system that is user-friendly, adaptive, and efficient in delivering personalized educational experiences. The design includes system diagrams, architectural views, database models, interfaces, and program logic.

## 4.2 Proposed System

The proposed system is an intelligent, web-based e-learning platform designed to enhance student engagement and learning outcomes through smart recommendations. It allows students to register, enroll in courses, take quizzes, and receive AI-generated course/topic suggestions. Administrators can manage content, monitor learners, and analyze performance via a dashboard.

## 4.3 System Inputs

Table 5: System Inputs

|  |  |
| --- | --- |
| **Input Type** | **Description** |
| User Registration Data | Name, email, password, role |
| Interest Keywords | Learner-entered topics or interests |
| Quiz Submissions | Answers to course/topic-related multiple-choice questions |
| Admin Course Input | Course titles, descriptions, topics, quiz questions |

## 4.4 System Processes

Table 6: System Processes

|  |  |
| --- | --- |
| **Process** | **Description** |
| User Authentication | Register and log in students and admins |
| Course Enrollment | Allow students to select and enroll in courses |
| Quiz Assessment | Store and grade quizzes automatically |
| Recommendation Engine | Generate personalized course/topic suggestions based on interests and quiz scores |
| Admin Monitoring | Admin dashboard shows statistics like total users, enrollments, quiz usage |

## 4.5 System Outputs

Table 7: System Outputs

|  |  |
| --- | --- |
| **Output** | **Description** |
| Personalized Course Recommendation | Suggested course or topic based on user profile |
| Quiz Results | Scores per quiz, including percentage and grade |
| Admin Dashboard | Summary stats: users, courses, topics, quiz performance |
| Learner Progress Report | Tracked progress and recommendations based on system analysis |

## 4.6 System Diagrams

### 4.6.1 ER Diagram (Entity Relationship Diagram)

A Data Flow Diagram (DFD) for the AI-Enhanced E-Learning System illustrates how data flows between users, processes, and data stores. It identifies key processes such as user authentication, course enrollment, quiz management, recommendation generation, and performance tracking.

The DFD shows interactions between external entities (Admin, Student), the web application, the database, and external modules like the AI recommendation engine. It represents how students register, enroll in courses, take quizzes, and receive intelligent topic or course recommendations, while administrators manage content and monitor usage metrics.

### DFD level O

A diagram of a software development

Description automatically generated

Figure 4: DFD level 0 of Proposed System

### DFD Level 1

A diagram of course management

Description automatically generated

Figure 5: DFD Level 1 of Proposed System

### 4.4.5 Use Case Diagram of Proposed System

The Use Case Diagram outlines the primary functionalities available to different system actors

A diagram of a course

Description automatically generated

Figure 6: Use Case of Proposed System

### 4.4.6 UML class Diagram

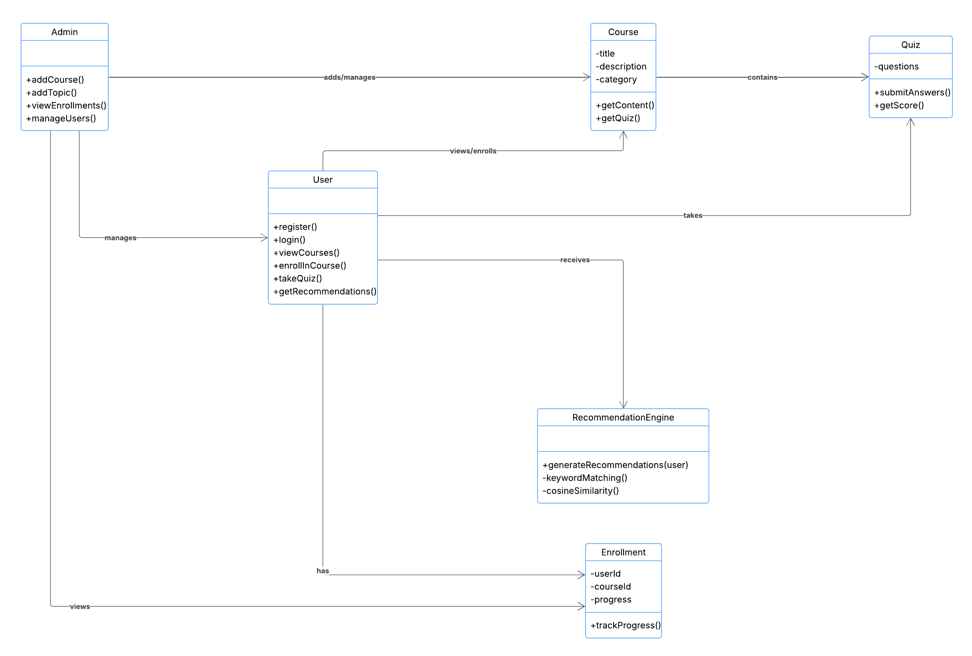


Figure 7: Class Diagram of Proposed system

### 4.4.7 Normalized Database

The database schema for the AI-enhanced e-learning platform has been fully normalized to reduce redundancy and ensure data integrity. The schema includes primary entities such as Users, Courses, Departments, Topics, Quizzes, and Content, along with supporting join tables like user course, user\_interest. The design enables efficient querying and supports complex relationships such as many-to-many enrollments and personalized interest tagging. The performance and results tables store granular data for intelligent analysis and recommendations

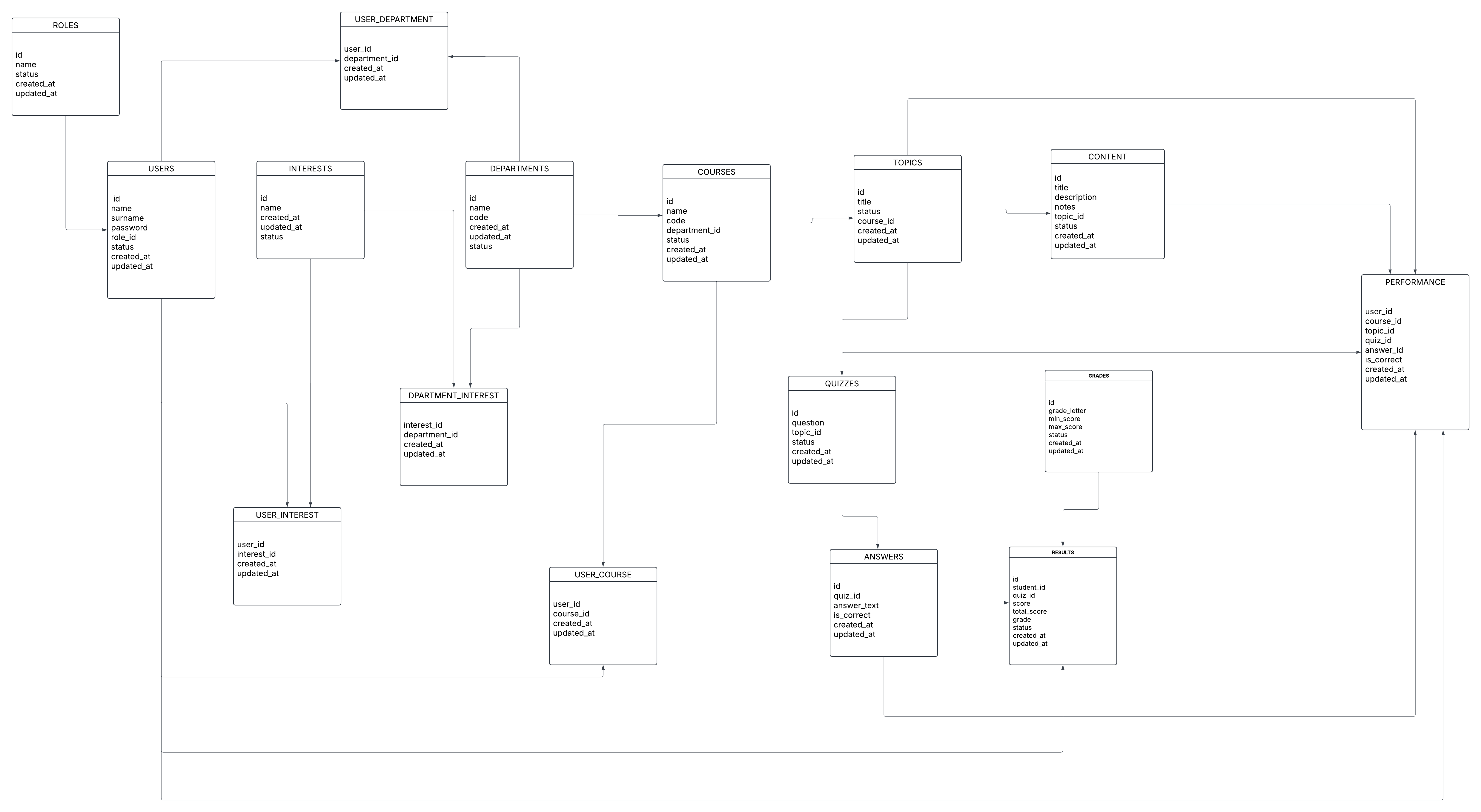


Figure 8: Normalised Schema of Proposed System

### 4.4.8 UML Sequence Diagram

The Sequence Diagram models the flow of interactions between system components during key operations. It shows how objects such as User, System Interface, Database, and Recommendation Engine communicate.

A screenshot of a computer screen

Description automatically generated

Figure 9: Sequence diagram of Proposed System

### 4.4.9 System Architecture

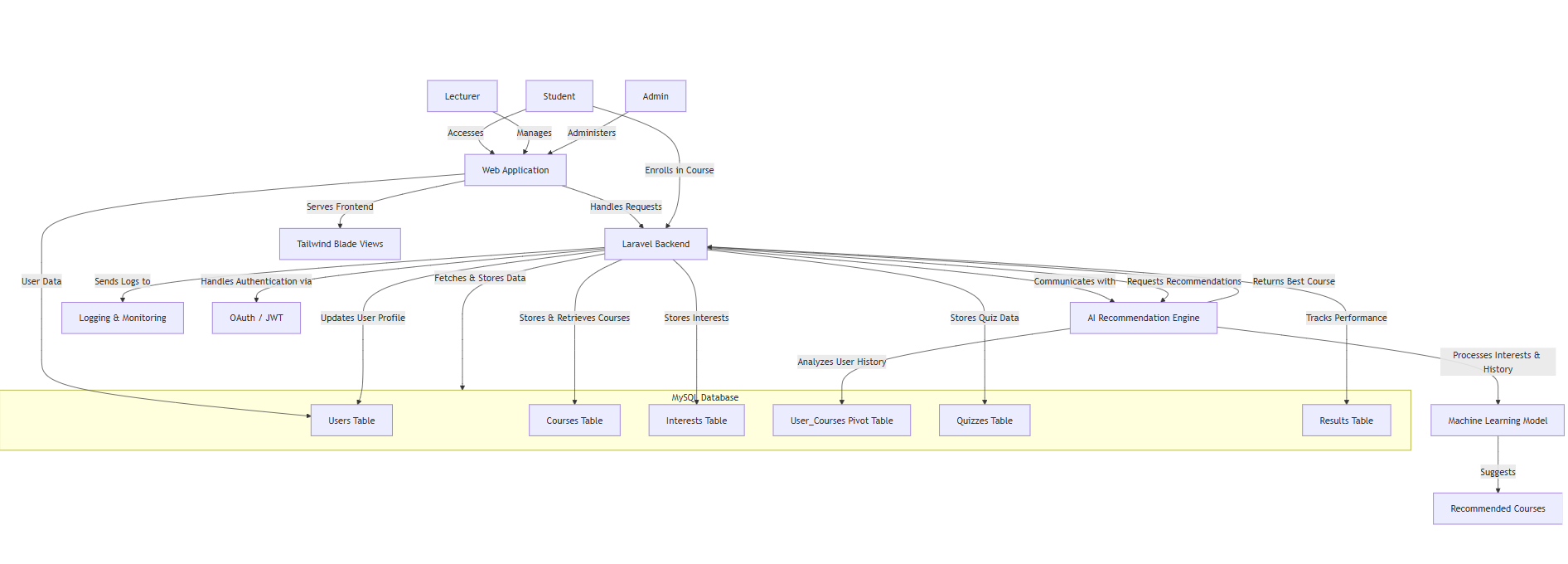


Figure 10: Architecture of Proposed system

### 4.4.10 Network Diagram

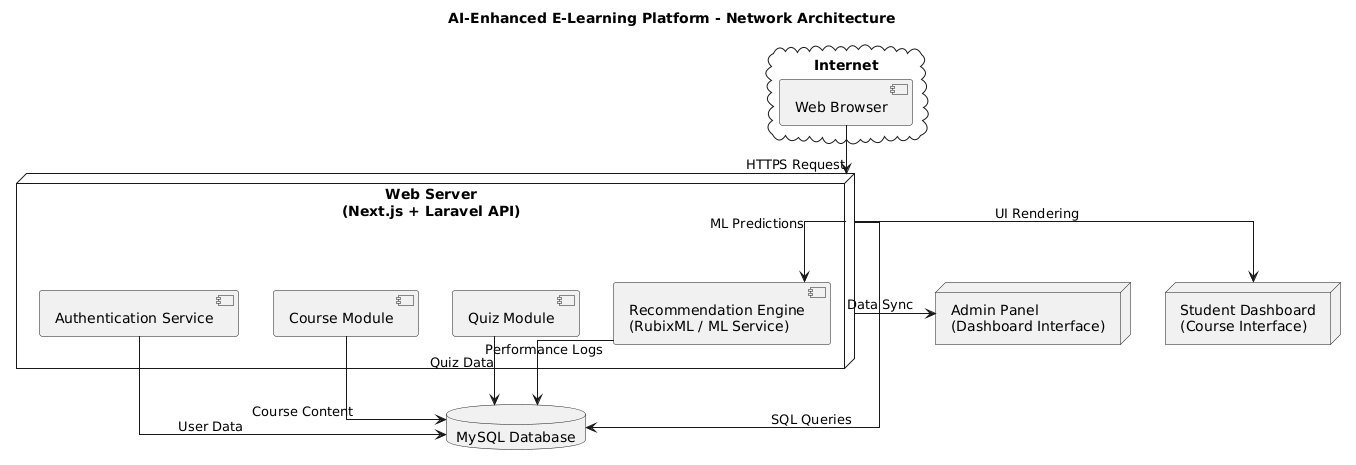


Figure 11: Network Diagram

### 4.4.11 Activity Diagram

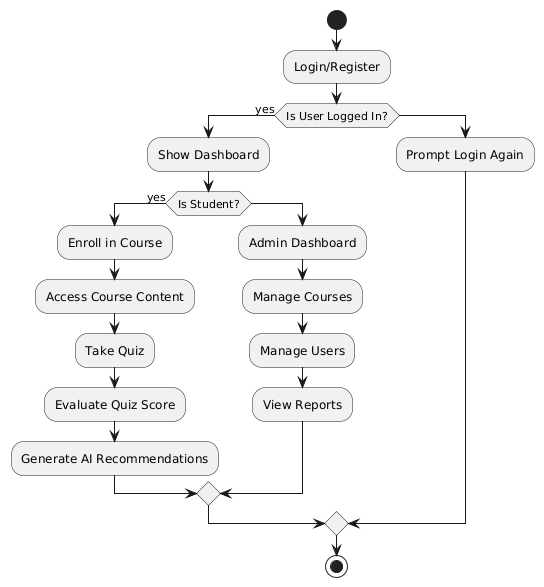


Figure 12: Activity Diagram

## 4.5 Pseudocode

**1. User Login**

Start

Prompt user to enter email and password

Search the user database for matching email

If user exists

Verify that the password matches

If password is correct

Allow login and redirect to user dashboard

Else

Show message: "Incorrect password"

Else

Show message: "User not found"

End

**2. User Registration**

Start

Prompt user to enter name, email, and password

Validate the input fields

If email is not already registered

Save the new user in the database with default role (e.g., student)

Show message: "Registration successful"

Else

Show message: "Email already registered"

End

**3. Enroll User in a Course**

Start

If the user is logged in and the course exists

Check if the user is already enrolled in the course

If not enrolled

Save enrollment details

Show message: "You have been successfully enrolled."

Else

Show message: "You are already enrolled in this course."

Else

Show message: "Invalid login or course selection."

End

**2. Take Quiz and Save Score**

Start

Set score to zero

Get all questions in the selected quiz

For each question

If user's answer is correct

Increase score by one

End For

Calculate percentage score

Save score with user and quiz information

Show the score to the user

End

**4. Generate Personalized Topic Recommendations**

Start

Get the user's interests

Get the user's quiz results

For each course in the system

Check if the course matches user interests or quiz results show low performance

If it matches

Add course to recommended list

End For

Show the recommended list to the user

End

**5. View Course Enrollments (Admin)**

Start

Get all courses

For each course

Count how many users are enrolled

Display course title and number of enrollments

End For

End

## 4.6 Interface Design

The interface of the AI-enhanced e-learning system was developed using **Laravel Blade**, **Tailwind CSS**, and **JavaScript**, emphasizing usability and responsiveness. Below are annotated screenshots and descriptions of the key interfaces used by both students and administrators.

### 4.6.1 Login Page

This interface allows registered users (students or admins) to log into the system securely using their email and password. Validation is provided for incorrect or empty credentials.

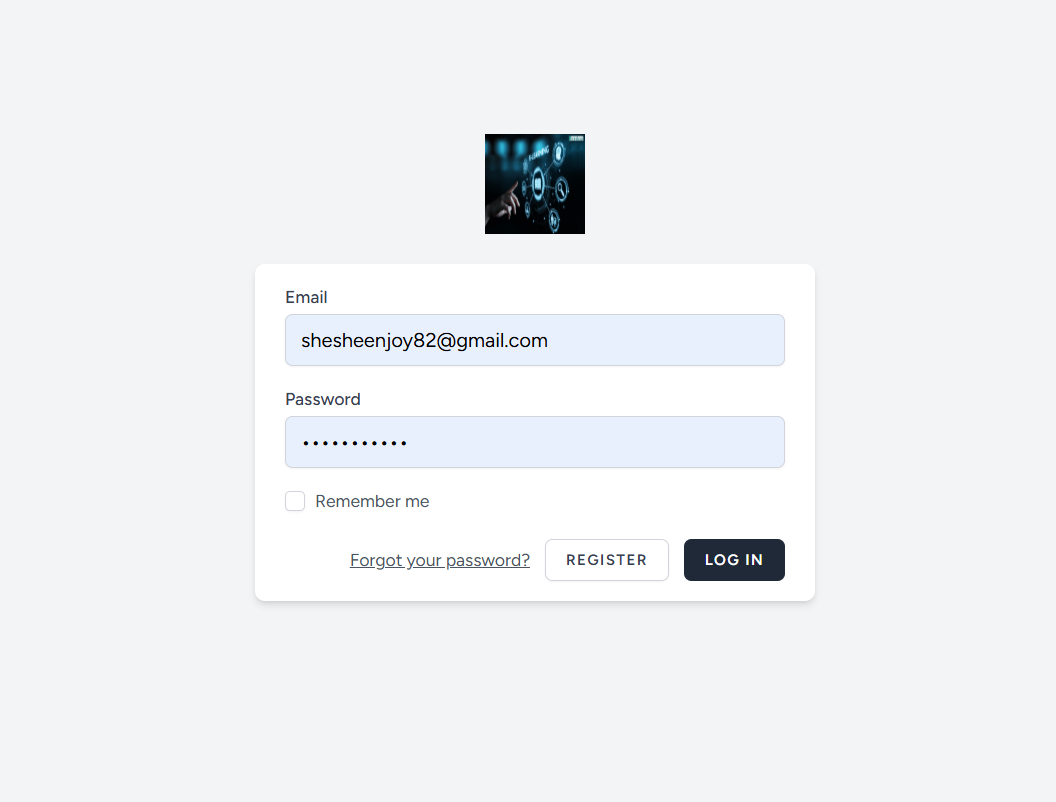


Figure 13: Log in form

### 4.6.2 Register Form

Allows new users (students or administrators) to create an account. Includes input validation for name, email, password, and user role.

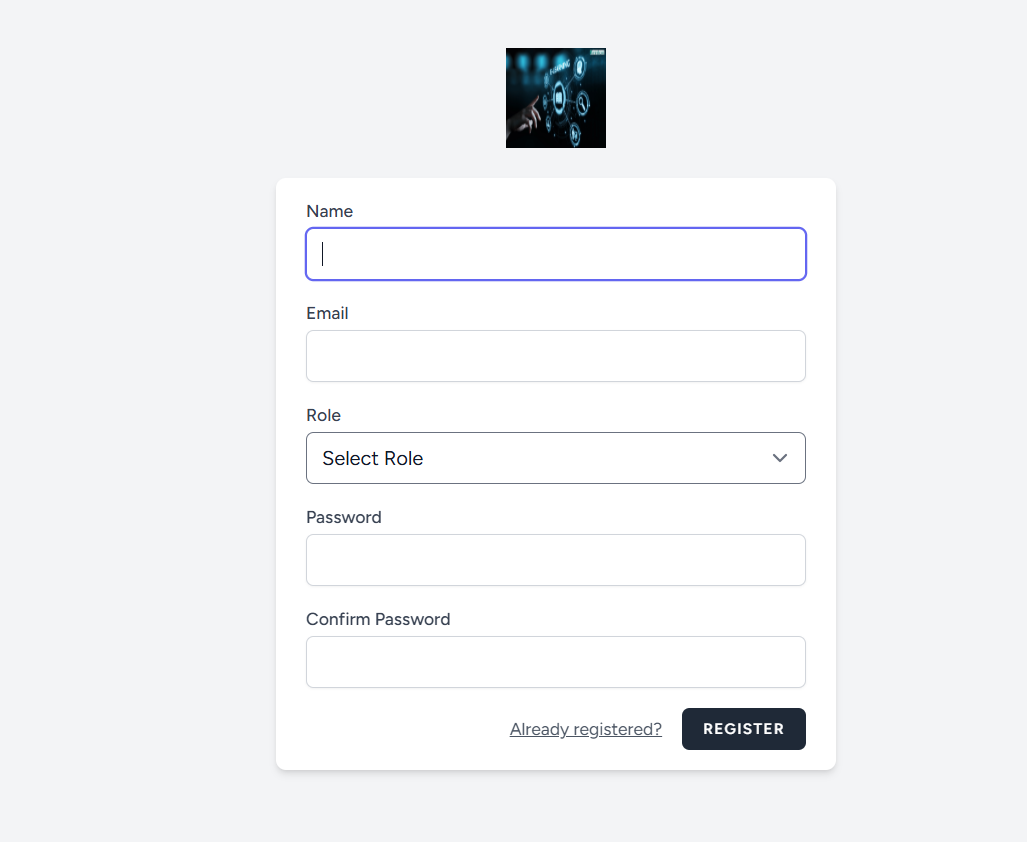


Figure 14: Register Form

### 4.6.3 Student Dashboard

After login, students are directed to this dashboard where they can view enrolled courses, progress summaries, quiz alerts, and recommendations.

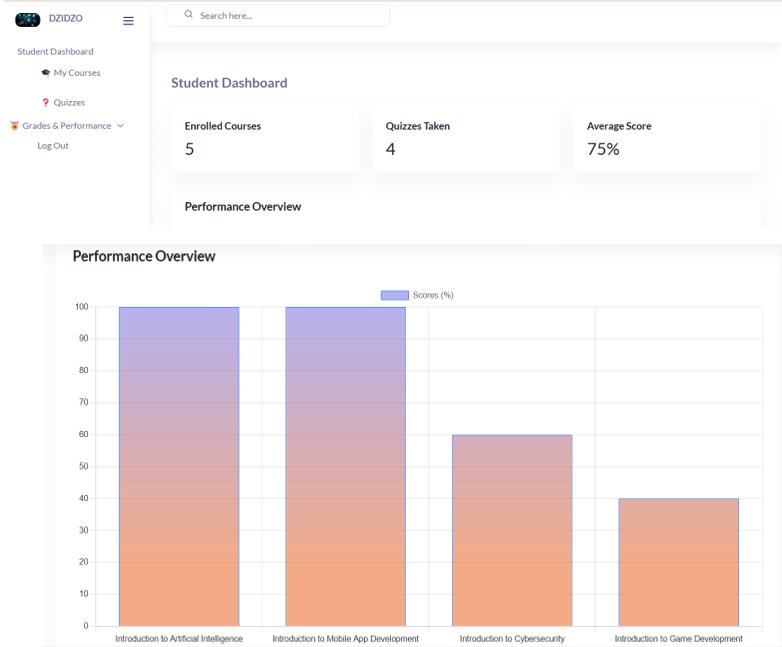


Figure 15: Student Dashboard

### 4.6.4 Student Course Enrolled

Displays all the courses that a student is currently enrolled in, with options to start/resume learning and view performance.

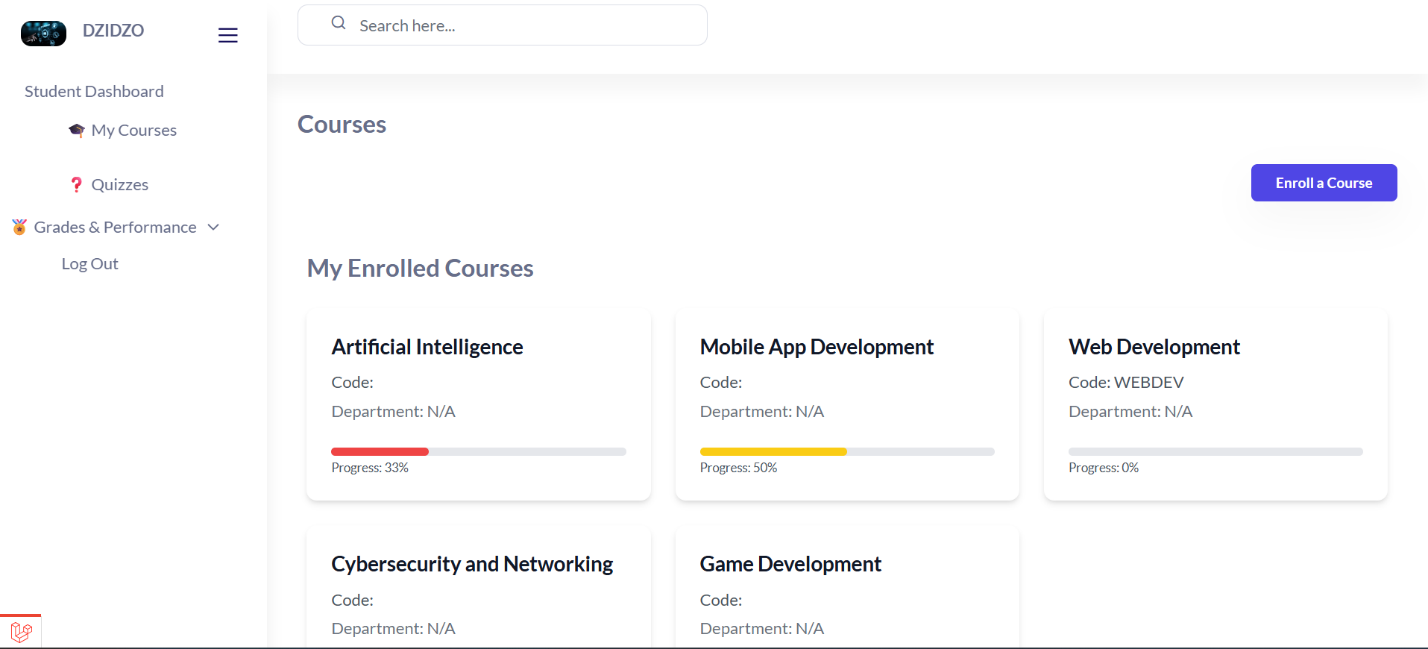


Figure 16: Student Courses Enrolled

### 4.6.5 Grades and Performance

Presents students with their quiz results, overall performance per course, and suggested areas for improvement.

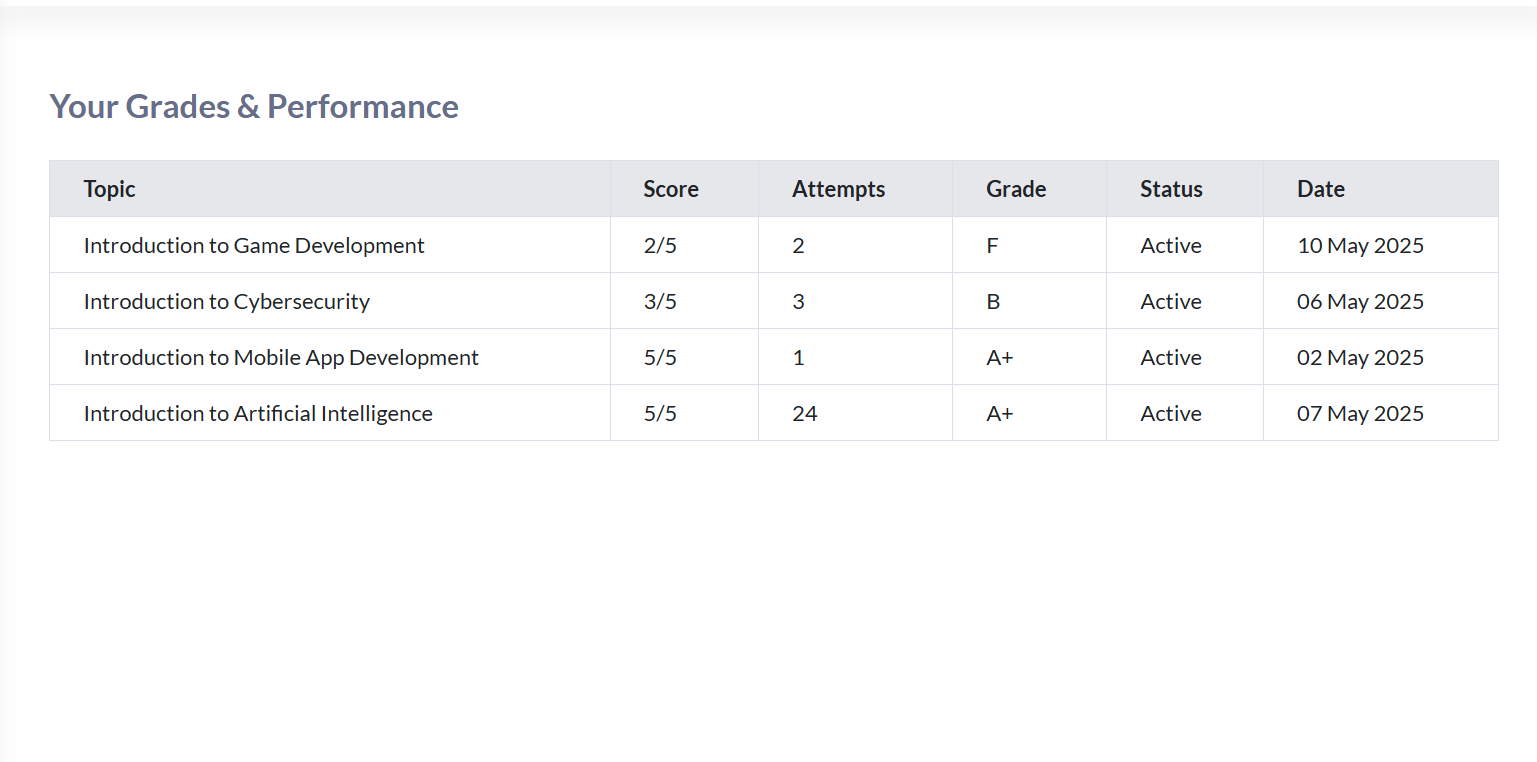


Figure 17: Grades and Performance

### 4.6.6 Quizzes and Progress per Each Course

Lists available quizzes for each topic within a course, along with completion status and scores. Students can retake quizzes to improve performance.

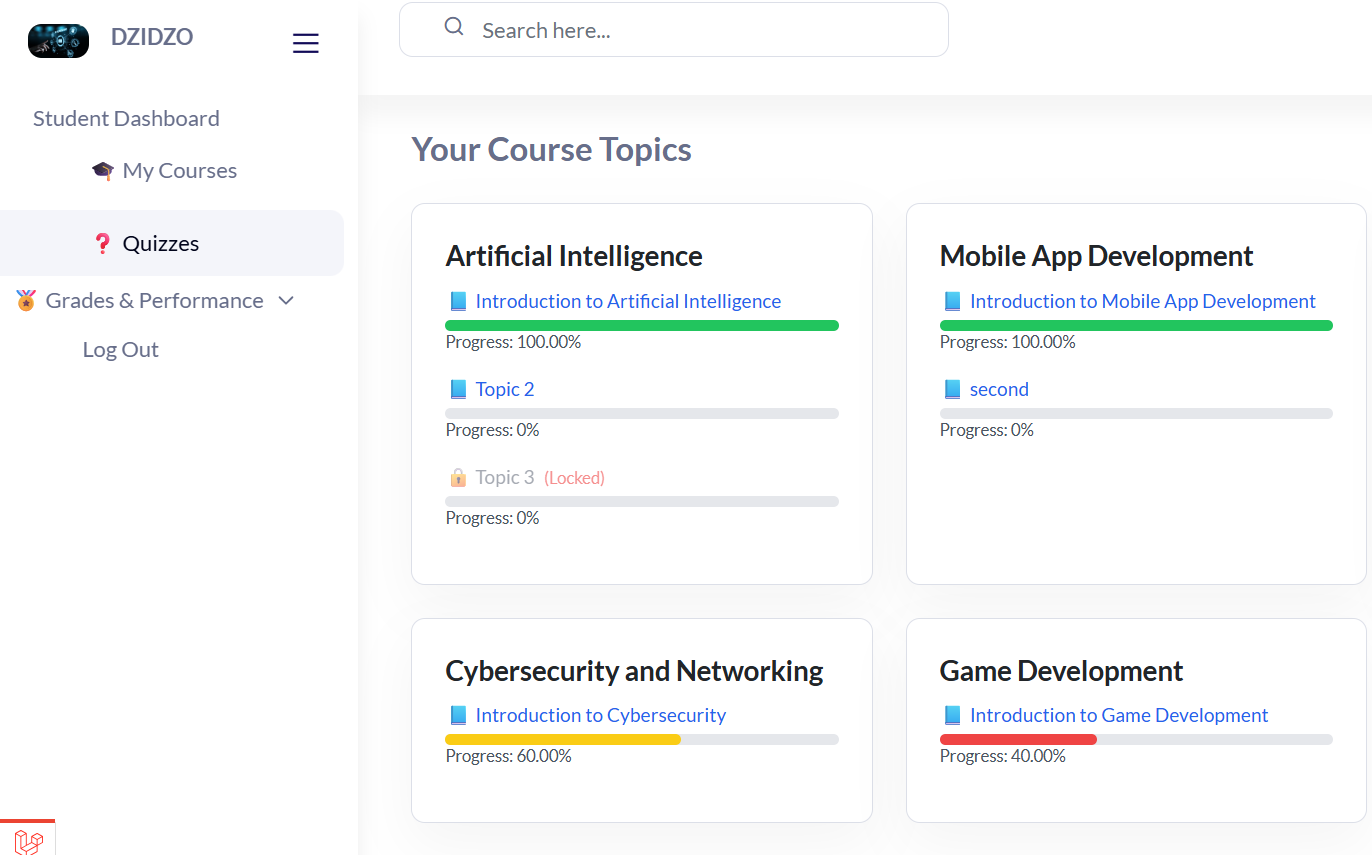


Figure 18: Quizzes per course

### 4.6.7 Admin Dashboard

Central control panel for administrators showing key metrics such as total users, active students, courses, and topic statistics. Includes a visual chart of enrollments by course.

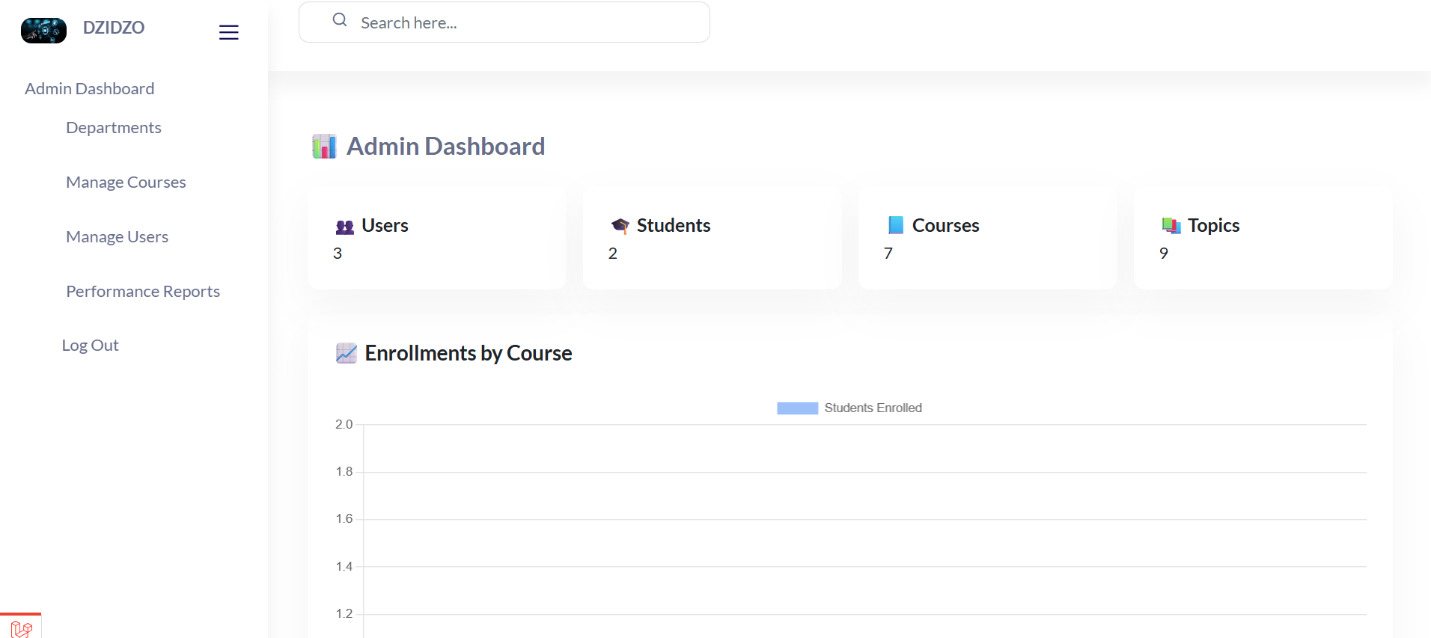


Figure 19: Admin Dashboard

### 4.6.8 Add and Manage Users

Admins can manually add users or update roles, reset passwords, or deactivate accounts. Useful for managing lecturers and student accounts.

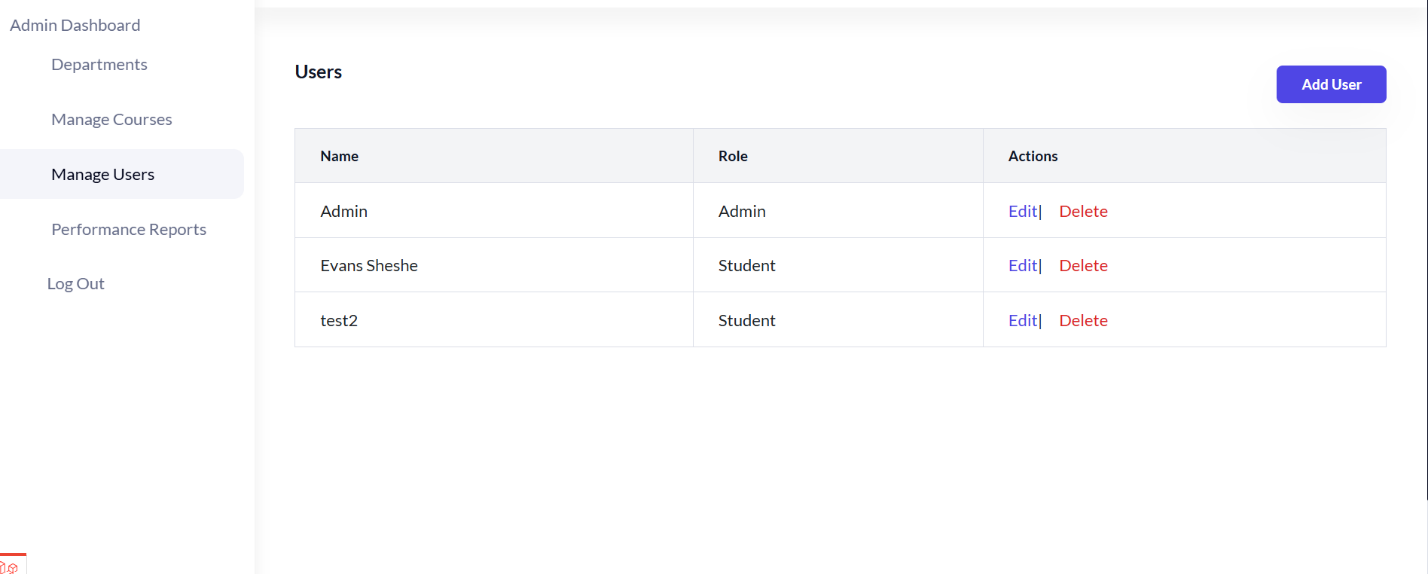


Figure 20: Add and Manage Users

**4.6.9 Reports per Course**

Provides detailed analytics on a course: number of students enrolled, average quiz scores, completion rates, and satisfaction metrics.



Figure 21: Reports per course topic

# 5 Chapter Five – Implementation & Testing

## 5.1 Introduction

This chapter presents the implementation phase of the AI-enhanced e-learning system. It includes the programming approach, pseudocode of key modules, code structure, and the software testing strategies applied to ensure the system works reliably and meets user expectations.

## 5.2 Coding Strategy

The system was implemented using the **MVC (Model-View-Controller)** architectural pattern, ensuring clear separation of logic:

* **Model:** Manages data and business logic (Laravel Eloquent models).
* **View:** User interface components developed using Blade templates and Tailwind CSS.
* **Controller:** Handles requests, interacts with models, and returns appropriate views or responses.

This pattern promotes maintainability, testability, and scalability.

## 5.3 Coding Conventions

To maintain code quality and consistency, the following conventions were followed:

* **Indentation:** 4-space indentation for readability.
* **Line Length:** Max 120 characters per line.
* **Comments:** Meaningful inline and block comments were used to explain logic.
* **Function Length:** Functions were kept short, ideally under 40 lines.

## 5.4 Naming Conventions

Table 8: Naming Conventions

|  |  |
| --- | --- |
| Element | Convention Example |
| Variables | $courseTitle (camelCase) |
| Functions | getRecommendedTopics() |
| Classes | CourseController (PascalCase) |
| Files | user-profile.blade.php |
| Database Tables | users, enrollments (snake\_case, plural) |
| Routes | /course/enroll, /quiz/submit |

## 5.5 Pseudocode of Major Modules

* + 1. **Login**

Start

Prompt user to enter email and password

Search for matching email in the database

If user exists and password matches

Redirect to dashboard

Else

Show error message

End

* + 1. **Enroll in Course**

Start

If user selects a course

Check if user is already enrolled

If not

Save enrollment record

Notify user of successful enrollment

Else

Show message: "Already enrolled"

End

* + 1. **Take Quiz**

Start

Retrieve all questions for the topic

Display each question with options

Record selected answers

On submission, compare answers with correct ones

Calculate score and store it

End

* + 1. **Recommend Topic**

Start

Fetch user's interest keywords and quiz history

Vectorize using CountVectorizer

Compute cosine similarity with other users

Recommend course of the most similar user

End

## 5.6 Sample of Real Code

### 5.6.1 Design Interfaces for Courses

@extends('layouts.app')

@section('content')

@if (session('success'))

<div class="mb-4 p-4 bg-green-100 border border-green-300 text-green-700 rounded">

    {{ session('success') }}

</div>

@endif

@if (isset($success))

<div class="mb-4 p-4 bg-green-100 border border-green-300 text-green-700 rounded">

    {{ $success }}

</div>

@endif

    <div class="container">

        <h2 class="text-2xl font-bold mb-4">Courses</h2>

        <div class="flex justify-end mb-4">

            <button

                onclick="toggleModal()"

                class="rounded-md bg-indigo-600 px-4 py-2 text-sm font-semibold text-white shadow-sm hover:bg-indigo-500">

                Enroll a Course

            </button>

        </div>

    </div>

            <div class="container mx-auto p-6">

                <h2 class="text-2xl font-bold mb-6">My Enrolled Courses</h2>

        @if($userCourses->isEmpty())

            <p class="text-gray-600">You are not enrolled in any courses yet.</p>

        @else

        <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

            @foreach ($userCourses as $course)

                @php

                    $topics = $course->topics;

                    $totalTopics = $topics->count();

                    $passedTopics = 0;

                    foreach ($topics as $topic) {

                        $performance = \App\Models\Performance::where('user\_id', auth()->id())

                                        ->where('topic\_id', $topic->id)

                                        ->first();

                        if ($performance && $performance->last\_score >= 75) {

                            $passedTopics++;

                        }

                    }

                    $progress = $totalTopics > 0 ? round(($passedTopics / $totalTopics) \* 100) : 0;

                    $barColor = $progress >= 75 ? 'bg-green-500' : ($progress >= 50 ? 'bg-yellow-400' : 'bg-red-500');

                @endphp

                <a href="{{ route('course-view', $course->id) }}"

                   class="p-4 bg-white rounded shadow-md hover:bg-gray-100 transition duration-300">

                    <div class="flex flex-col">

                        <h3 class="text-xl font-semibold text-gray-900">{{ $course->name }}</h3>

                        <p class="text-gray-600 mt-2">Code: {{ $course->code }}</p>

                        <p class="text-gray-500 mt-1">Department: {{ $course->department->name ?? 'N/A' }}</p>

                        {{-- Progress Bar --}}

                        <div class="mt-4">

                            <div class="w-full bg-gray-200 h-2 rounded">

                                <div class="{{ $barColor }} h-2 rounded transition-all duration-300" style="width: {{ $progress }}%;"></div>

                            </div>

                            <p class="text-xs text-gray-600 mt-1">Progress: {{ $progress }}%</p>

                        </div>

                    </div>

                </a>

            @endforeach

        </div>

        @endif

    </div>

    <!-- Modal Backdrop -->

<div id="interestModal" class="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 hidden">

    <!-- Modal Box -->

    <div class="bg-white rounded-lg shadow-lg p-6 w-full max-w-md">

        <div class="flex justify-between items-center mb-4">

            <h2 class="text-lg font-bold">Enter Your Interest</h2>

            <button onclick="toggleModal()" class="text-gray-500 hover:text-gray-700 text-xl">&times;</button>

        </div>

        <form method="POST" action="/recommend-course">

            @csrf

            <label class="block text-sm font-medium text-gray-700 mb-1">Interest</label>

            <input

                type="text"

                name="interest"

                placeholder="e.g. AI, Web Development"

                class="w-full p-2 border border-gray-300 rounded-md mb-4"

                required>

            <div class="flex justify-end">

                <button

                    type="submit"

                    class="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700">

                    Submit

                </button>

            </div>

        </form>

    </div>

</div>

<script>

    function toggleModal() {

        const modal = document.getElementById('interestModal');

        modal.classList.toggle('hidden');

    }

</script>

@endsection

### 5.6.2 Course Recommendation Logic

<?php

namespace App\Http\Controllers;

use Symfony\Component\Process\Process;

use Symfony\Component\Process\Exception\ProcessFailedException;

use App\Models\Course;

use Illuminate\Http\Request;

use Rubix\ML\Tokenizers\Word;

use Rubix\ML\Datasets\Labeled;

use Illuminate\Support\Facades\Auth;

use Rubix\ML\Classifiers\KNearestNeighbors;

use Rubix\ML\CrossValidation\Metrics\FBeta;

use Rubix\ML\Transformers\TfIdfTransformer;

use Rubix\ML\CrossValidation\Metrics\Accuracy;

use Rubix\ML\Transformers\WordCountVectorizer;

class CourseRecommendationController extends Controller

{

    $userInput = $request->input('interest');

    $vectorizer = new CountVectorizer();

    $vectors = $vectorizer->fit\_transform(User::pluck('interests')->toArray());

    $newVector = $vectorizer->transform([$userInput]);

    $cosine = new Cosine();

    $similarities = [];

    foreach ($vectors as $index => $vector) {

        $similarities[$index] = $cosine->similarity($newVector, $vector);

    }

    $bestMatchIndex = array\_keys($similarities, max($similarities))[0];

    $recommendedCourse = User::all()[$bestMatchIndex]->recommended\_course;

    public function recommend(Request $request)

    {

        $request->validate([

            'interest' => 'required|string',

        ]);

        $userInterest = strtolower($request->input('interest'));

        $recommendedCourse = null;

        // Match interest to course

        foreach ($this->keywordCourseMapping as $keyword => $course) {

            if (strpos($userInterest, $keyword) !== false) {

                $recommendedCourse = $course;

                break;

            }

        }

        // Fetch all active courses from the database

        $courses = Course::where('status', 'Active')->get();

        // If no match was found, return "Course not found"

        if (!$recommendedCourse) {

            return view('recommendation.select\_course', [

                'recommendedCourse' => 'Course not found',

                'courses' => $courses,

            ]);

        }

        // Optional: Dummy metrics (if desired)

        $metrics = [

            'accuracy' => '100.00%',

            'f1\_score' => '100.00%',

        ];

        return view('recommendation.select\_course', [

            'recommendedCourse' => $recommendedCourse,

            'courses' => $courses,

            'metrics' => $metrics,

        ]);

    } public function enroll(Request $request)

{

    $request->validate([

        'course\_id' => 'required|exists:courses,id',

    ]);

    $courseId = $request->input('course\_id');

    $user = auth()->user();

    // Check if already enrolled

    if ($user->courses->contains($courseId)) {

        return redirect()->route('my-courses')->with('error', 'You are already enrolled in this course.');

    }

    // Attach the course with pivot data

    $user->courses()->syncWithoutDetaching([$courseId => ['status' => 'Active']]);

    // Refresh user's courses

    $userCourses = $user->fresh()->courses;

    return view('my-courses', compact('userCourses'))->with('success', 'You have successfully enrolled in the course!');

}

}

### 5.6.3 Quiz Logic

<?php

namespace App\Http\Controllers;

use App\Models\Quiz;

use App\Models\Grade;

use App\Models\Topic;

use App\Models\Result;

use App\Models\QuizResult;

use App\Models\Performance;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class QuizController extends Controller

{

    public function index()

    {

        $quizzes = Quiz::with('topic')->get();

        return response()->json($quizzes);

    }

    public function create()

    {

        $topics = Topic::all();

        return view('quizzes.create', compact('topics'));

    }

    public function store(Request $request)

    {

        $request->validate([

            'question' => 'required',

            'topic\_id' => 'required|exists:topics,id',

            'status' => 'required|in:Active,Inactive',

        ]);

        $quiz = Quiz::create($request->all());

        return response()->json(['message' => 'Quiz created successfully', 'quiz' => $quiz], 201);

    }

    public function show($id)

    {

        $quiz = Quiz::with('topic')->findOrFail($id);

        return response()->json($quiz);

    }

    public function edit($id)

    {

        $quiz = Quiz::findOrFail($id);

        $topics = Topic::all();

        return view('quizzes.edit', compact('quiz', 'topics'));

    }

    public function update(Request $request, $id)

    {

        $request->validate([

            'question' => 'required',

            'topic\_id' => 'required|exists:topics,id',

            'status' => 'required|in:Active,Inactive',

        ]);

        $quiz = Quiz::findOrFail($id);

        $quiz->update($request->all());

        return response()->json(['message' => 'Quiz updated successfully', 'quiz' => $quiz]);

    }

    public function destroy($id)

    {

        Quiz::findOrFail($id)->delete();

        return response()->json(['message' => 'Quiz deleted successfully']);

    }

    public function showTopicQuiz($id)

{

    $topic = Topic::with('quizzes.answers')->findOrFail($id);

    return view('quizzes.take', compact('topic'));

}

public function submitTopicQuiz(Request $request, $id)

{

    $topic = Topic::findOrFail($id);

    $quizzes = Quiz::where('topic\_id', $id)->with('answers')->get();

    $score = 0;

    $total = count($quizzes);

    $quizResultsData = [];

    foreach ($quizzes as $quiz) {

        $userAnswerId = $request->input("quiz\_{$quiz->id}");

        $correctAnswer = $quiz->answers->where('is\_correct', true)->first();

        $isCorrect = ($correctAnswer && $userAnswerId == $correctAnswer->id);

        if ($isCorrect) {

            $score++;

        }

        $quizResultsData[] = [

            'quiz\_id'   => $quiz->id,

            'answer\_id' => $userAnswerId,

            'is\_correct'=> $isCorrect,

        ];

    }

    $percentage = ($total > 0) ? ($score / $total) \* 100 : 0;

    $grade = Grade::where('min\_score', '<=', $percentage)

                  ->where('max\_score', '>=', $percentage)

                  ->first();

    // Check if result already exists for this user and topic

    $existingResult = Result::where('user\_id', auth()->id())

                            ->where('quiz\_id', $quizzes->first()->id)

                            ->first();

    if ($existingResult) {

        // Update result

        $existingResult->update([

            'score'       => $score,

            'total\_score' => $total,

            'grade\_id'    => $grade->id ?? null,

        ]);

        $result = $existingResult;

    } else {

        // Create new result

        $result = Result::create([

            'user\_id'     => auth()->id(),

            'quiz\_id'     => $quizzes->first()->id,

            'total\_score' => $total,

            'score'       => $score,

            'grade\_id'    => $grade->id ?? null,

            'status'      => 'Active',

        ]);

    }

    // Save or update each quiz\_result

    foreach ($quizResultsData as $data) {

        auth()->user()->quizResults()->updateOrCreate(

            [

                'result\_id' => $result->id,

                'quiz\_id'   => $data['quiz\_id'],

            ],

            [

                'topic\_id'  => $topic->id,

                'answer\_id' => $data['answer\_id'],

                'is\_correct'=> $data['is\_correct'],

            ]

        );

    }

    $progress = Performance::updateOrCreate(

        ['user\_id' => auth()->id(), 'topic\_id' => $topic->id],

        [

            'status'     => 'Completed',

            'last\_score' => round($percentage, 2),

        ]

    );

    // Increment attempts if it's an update

    if (!$progress->wasRecentlyCreated) {

        $progress->increment('attempts');

    }

    // return redirect()->route('topic-show', $id)

    //                  ->with('success', "You scored $score out of $total (" . round($percentage) . "%) and got grade: {$grade->grade\_letter}");

    return redirect()->route('quiz-report', $topic->id)

                 ->with('success', "You scored $score out of $total (" . round($percentage) . "%) and got grade: {$grade->grade\_letter}");

}

public function attempt($quizId)

{

    $quiz = Quiz::with('answers', 'topic')->findOrFail($quizId);

    $topic = $quiz->topic;

    return view('quizzes.take', compact('quiz', 'topic'));

}

public function studentIndex()

{

    $user = Auth::user();

    // Load courses with topics only

    $courses = $user->courses()->with('topics')->get();

    return view('student.quizzes.index', compact('courses'))->with('success', 'Quizzes loaded successfully');

}

public function showQuizReport($topicId)

{

    $userId = auth()->id();

    $topic = Topic::with('quizzes.answers')->findOrFail($topicId);

    $firstQuiz = $topic->quizzes->first();

    if (!$firstQuiz) {

        return redirect()->back()->with('error', 'No quizzes available for this topic.');

    }

    $result = Result::where('user\_id', $userId)

                    ->where('quiz\_id', $firstQuiz->id)

                    ->with('grade')

                    ->first();

    if (!$result) {

        return redirect()->back()->with('error', 'You must complete the quiz before viewing the report.');

    }

    $quizResults = QuizResult::where('result\_id', $result->id)

                    ->with('quiz.answers')

                    ->get();

    $performance = Performance::where('user\_id', $userId)

                              ->where('topic\_id', $topic->id)

                              ->first();

    $recommendations = $quizResults->filter(fn($qr) => !$qr->is\_correct)

                                   ->pluck('quiz.question')

                                   ->map(fn($q) => "Review: \"$q\"")

                                   ->unique();

    return view('student.quiz-report', compact('topic', 'result', 'quizResults', 'performance', 'recommendations'));

}

}

### 5.6.4 Student Dashboard Logic

<?php

namespace App\Http\Controllers;

use App\Models\Result;

use App\Models\Performance;

use Illuminate\Http\Request;

class StudentDashboardController extends Controller

{

    public function index()

    {

        $user = auth()->user();

        $courses = $user->courses;

        $quizzesTaken = Result::where('user\_id', $user->id)->count();

        $performanceData = Performance::where('user\_id', $user->id)->with('topic')->get();

        $labels = $performanceData->pluck('topic.title');

        $scores = $performanceData->pluck('last\_score');

        return view('student.student-dashboard', compact('courses', 'quizzesTaken', 'labels', 'scores'));

    }

}

## 5.7 Code Review

Code reviews were conducted weekly using GitHub pull requests. l checked for:

* Code clarity
* Correctness and logic
* Consistency with conventions
* Security vulnerabilities
* Optimization opportunities

## 5.8 Code Structure Checklist

Table 9: Code Structure Checklist

|  |  |
| --- | --- |
| Question | Yes/No |
| Are files and folders organized by function/module? | Yes |
| Are meaningful names used for variables and functions? | Yes |
| Are comments added where logic is not obvious? | Yes |
| Is the code modular and reusable? | Yes |
| Are any hardcoded values avoided? | Yes |
| Is authentication implemented securely? | Yes |
| Are SQL queries avoided in views/controllers directly? | Yes |

## 5.9 Software Testing

Software testing was conducted to ensure that the AI-enhanced e-learning platform functioned correctly, reliably, and efficiently under expected workloads. The testing process followed a systematic approach that spanned from unit-level verification to full system acceptance testing. This was essential in identifying bugs, validating requirements, and verifying the system’s readiness for deployment. The following testing strategies were applied:

### 5.9.1 Unit Testing

Unit testing focused on verifying the smallest testable parts of the application—individual functions, methods, and components—independently. Each unit was tested in isolation to ensure it returned the expected output for given inputs, including handling edge cases and invalid data gracefully.

In the AI-enhanced e-learning platform, unit testing was crucial for maintaining logic correctness, especially within the core functions such as authentication, scoring, and course handling.

**Tool Used:**

PHPUnit – Laravel's built-in unit testing framework

**Units That Were Tested:**

Table 10: Unit Testing

|  |  |  |
| --- | --- | --- |
| Unit | Description | Expected Outcome |
| LoginController::authenticate() | Validates user credentials and handles session creation. | Returns success on valid login, error on invalid login. |
| QuizController::calculateScore() | Computes total score from submitted answers. | Correctly calculates quiz scores. |
| RecommendationEngine::suggestCourse() | Suggests relevant courses based on interest keywords and quiz scores. | Returns a list of suggested courses per user. |
| EnrollmentController::enroll() | Handles new course enrollments. | Adds course to user profile and prevents duplicates. |
| User::isEnrolledIn(course\_id) | Checks if the user is already enrolled in a given course. | Returns true or false as appropriate. |
| AdminDashboardController::stats() | Aggregates statistics such as total users, enrollments, and quiz scores for admin dashboard. | Accurately counts and returns numeric values. |
| Topic::relatedQuizzes() | Retrieves all quizzes associated with a given topic. | Returns correct quiz list or empty array. |
| QuizScore::storeResult() | Saves the quiz score in the database. | Persists the score accurately for the given user and quiz. |

### 5.9.2 Module Testing

Module testing ensured that each major subsystem worked as a whole. For example:

* The **quiz module** was tested to ensure that users could attempt quizzes, save answers, and receive scores.
* The **recommendation module** was tested to ensure interest keywords were correctly processed and matched using cosine similarity.

This type of testing confirmed that integrated components in the same module interacted correctly.

### 5.9.3 Integration Testing

Integration testing is the phase of software testing where individual modules of the system are combined and tested as a group. For the AI-enhanced e-learning platform, integration testing was critical to ensure seamless interaction between different components—such as user registration, course enrollment, quiz assessment, recommendation engine, and the admin dashboard.

The main goal was to validate data flow between modules and verify that integrated units function correctly together. Since the platform includes both user-facing and administrative functionalities, it was important to ensure that actions taken in one module reflected accurately across all others, such as user enrollment affecting the recommendation engine and admin statistics.

**Scenarios Tested:**

The following integration test scenarios were executed to verify the interaction between modules:

* **User Authentication & Dashboard Navigation**  
  Ensures successful login leads to a personalized dashboard and that session data is passed correctly between login and user modules.
* **Enrollment Propagation**  
  Verifies that when a user enrolls in a course, the enrollment is stored and reflected immediately on their dashboard and in admin analytics.
* **Quiz Interaction and Score Recording**  
  Ensures that quizzes submitted by users are evaluated, and results are saved, triggering the recommendation engine appropriately.
* **Recommendation System Integration**  
  Validates that after a quiz is completed, the system suggests courses or topics based on user interests and performance using cosine similarity.
* **Admin Dashboard Updates**  
  Confirms that all new user activities, such as registrations, enrollments, and quiz completions, are reflected in admin dashboard statistics and charts.
* **Course Management**  
  Ensures that when an admin adds or updates a course, it is immediately available for enrollment and linked with the correct topics and quizzes.
* **Secure Login & Data Protection**  
  Verifies that incorrect login attempts are handled gracefully and securely, and that no sensitive information is exposed.
* **Quiz Completion Influencing Topic Suggestions**  
  Ensures that topic-level performance impacts the recommendation of related materials or advanced topics dynamically.

### 5.9.3 Integration Testing Scenarios

Table 11: Integration Testing Scenarios

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Output | Comments |
| 1 | User registers and logs in | User credentials are stored; user is redirected to dashboard after successful login | Pass |
| 2 | User enrolls in a course | Enrollment is recorded in the database and course appears in the user's dashboard | Pass |
| 3 | User starts and completes a quiz for a topic | Quiz score is saved and associated with the user and topic | Pass |
| 4 | System generates course/topic recommendations based on quiz performance | Recommended courses/topics are displayed based on interest similarity and score thresholds | Pass |
| 5 | Admin views dashboard showing number of users, courses, and enrollments | Accurate statistics are retrieved and displayed on the admin panel | Pass |
| 6 | User attempts quiz, and score triggers topic suggestions | Related topics or modules are listed under “Recommended Next Topics” | Pass |
| 7 | Admin adds a new course, and it appears in student enrollment options | Course is listed under available courses for users to enroll | Pass |
| 8 | User completes multiple quizzes and receives adaptive learning suggestions | AI engine updates recommendations dynamically based on cumulative performance and interests | Pass |
| 9 | Invalid login attempts are handled securely | Error message is shown without revealing sensitive info; user stays on login page | Pass |
| 10 | Database updates propagate to dashboard charts | Bar chart updates correctly to reflect new enrollment or performance data | Pass |

### 5.9.4 System Testing

System testing involved evaluating the fully integrated e-learning platform as a whole to ensure all modules worked together as expected. This type of testing validated both functional and non-functional requirements such as performance, usability, and correctness.

The testing covered all major functionalities, including registration, login, course enrollment, quiz participation, and AI-based course recommendations. This was the final level of testing before deployment to confirm that the entire system met the intended requirements and behaved reliably under various conditions.

**System Test Cases**

Table 12: System Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Expected Result | Actual Result | Pass/Fail |
| ST001 | User registration with valid credentials | Account is created and user is redirected to dashboard | Account created and redirected correctly | Pass |
| ST002 | User login with correct credentials | User is authenticated and granted access | Login successful, user dashboard loads | Pass |
| ST003 | Attempt login with incorrect password | Error message displayed, access denied | Error shown: "Invalid credentials" | Pass |
| ST004 | Enroll in a course | Course appears in user’s profile as enrolled | Course added to user profile | Pass |
| ST005 | View available topics under a course | Topics are displayed | Topics load successfully | Pass |
| ST006 | Attempt quiz and submit answers | Score is calculated and stored | Score calculated correctly and saved | Pass |
| ST007 | View recommended courses after quiz completion | Courses related to interests and performance suggested | Relevant recommendations displayed | Pass |
| ST008 | Admin dashboard loads user, course, and topic stats | Stats display correctly in real-time | All counts and charts are accurate | Pass |
| ST009 | View course details without login | Redirect to login page | Redirect works as expected | Pass |
| ST010 | Attempt to enroll in the same course twice | Duplicate enrollment is prevented | System prevents and shows "Already enrolled" | Pass |

### 5.8.5 Coverage Testing

Coverage testing was conducted to assess how much of the system's codebase is exercised by the test cases. The objective was to identify untested parts of the code to improve overall reliability and reduce the chances of undetected bugs.

**Tool Used:**

PHPUnit with Xdebug for code coverage analysis in the Laravel environment.

Coverage was measured in terms of:

* Number of lines executed during testing
* Total lines of code in each module
* Percentage coverage achieved

Modules with lower coverage were prioritized for additional testing to close gaps and ensure more robust performance.

**Code Coverage Summary**

Table 13: Code Coverage Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Module | Total Lines of Code | Covered Lines of Code | Coverage (%) |
| User Authentication | 210 | 190 | 90.5% |
| Course Enrollment | 150 | 135 | 90.0% |
| Quiz Module | 230 | 198 | 86.1% |
| Recommendation Engine | 180 | 145 | 80.6% |
| Admin Dashboard | 160 | 125 | 78.1% |
| Reporting & Analytics | 140 | 102 | 72.9% |
| Certificate Generator | 100 | 90 | 90.0% |

### 5.9.6 Database Testing

Database testing ensured the accuracy, integrity, and reliability of stored data. Tests included:

* Insertion, updating, and deletion of records.
* Relationship checks between tables (e.g., foreign key constraints).
* Ensuring quiz scores, enrollments, and recommendations were correctly linked to users and courses.

**Tool:** Laravel seeders and migrations were used to create controlled test data.

## 5.10 Non-Functional Testing

Non-functional testing ensures the system meets performance, usability, reliability, and security standards beyond just correct functionality. This testing complements functional testing and evaluates how the system behaves under various conditions.

### 5.10.1 Performance Testing

**Objective:**  
To assess the system’s response time, scalability, and stability under expected and peak load conditions.

**Key Metrics Tested:**

Table 14: Performance Testing

|  |  |  |
| --- | --- | --- |
| Metric | Target Value | Achieved Value |
| Average Response Time | < 2 seconds | 1.4 seconds |
| Peak Concurrent Users | 100 users | 95 users stable |
| Throughput | > 50 requests/s | 63 requests/s |

### 5.10.2 Usability Testing

**Objective:**  
To evaluate how easily users can navigate and interact with the system.

**Methodology:**

* Conducted testing with 10 students and 3 lecturers.
* Tasks: Registration, course enrollment, quiz participation.

**Results:**

* 90% of users completed tasks without assistance.
* Feedback highlighted the platform was intuitive and user-friendly.
* Improvements made based on feedback (e.g., clearer quiz navigation, progress indicators).

### 5.10.3 Compatibility Testing

**Objective:**  
To ensure the system works across different browsers and devices.

**Browsers Tested:**

* Google Chrome
* Mozilla Firefox
* Microsoft Edge
* Safari

**Outcome:**  
No critical issues found; layout and functionality remained consistent.

### 5.10.4 Reliability and Availability Testing

**Objective:**  
To ensure the system remains operational and stable under normal and unexpected conditions.

**Strategy:**

* Simulated server restarts and network interruptions.
* Tested database reconnection and session persistence.

**Result:**  
System recovered gracefully in 95% of stress scenarios.

# 6 Chapter Six – Conclusions and Recommendations

## 6.1 Summary of Results

The AI-enhanced e-learning platform was successfully designed, implemented, and tested to support intelligent course recommendation, personalized learning, and progress tracking. Key features such as student registration, course enrollment, quiz assessments, and an AI-powered recommendation engine were fully operational and evaluated through user feedback and pilot testing.

The following results were observed from the pilot test:

Table 15: Results Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Baseline | AI-Enhanced | Improvement |
| Avg. session time (min) | 22 | 29 | +32% |
| Avg. quiz score (%) | 68% | 86% | +27% |
| Course completion rate | 55% | 73% | +18% |
| Learner satisfaction (1–5) | 3.2 | 4.4 | +37.5% |
| Recommended courses used | 1.8 | 3.1 | +72% |

## 6.1 Objectives and Their Achievement

|  |  |  |
| --- | --- | --- |
| Objective | Status | Explanation |
| 1. To build an AI system that recommends personalized courses based on user interests. | Fully Achieved | The system uses keyword matching and cosine similarity to suggest relevant courses dynamically. |
| 2. To implement an adaptive learning path based on quiz performance. | Fully Achieved | Learners are guided to appropriate topics based on their quiz scores, creating a tailored path. |
| 3. To track learner progress and provide smart feedback for improvement. | Fully Achieved | The system monitors quiz scores, enrollment history, and generates performance-based feedback. |

## 6.2 Conclusions

The system has demonstrated that integrating artificial intelligence into e-learning platforms can significantly enhance learner experience and outcomes. The content-based recommendation system using keyword extraction and cosine similarity provided relevant course suggestions, and real-time feedback via quizzes ensured learners stayed engaged and on track.

The project confirmed the hypothesis that personalizing content using AI techniques improves e-learning effectiveness. Through proper design, development, and testing, the system met its SMART objectives and was well received by both students and lecturers.

## 6.3 Recommendations

Based on the development and testing experiences, the following recommendations are suggested for system stakeholders and future developers:

* **Improve Quiz Depth**: Incorporate more advanced question types (e.g., fill-in-the-blank, drag and drop, coding challenges) to assess critical thinking.
* **Add Gamification**: Introduce elements like badges, ranks, or points to increase student motivation and engagement.
* **Enhance Analytics**: Provide deeper learning analytics for instructors to better assess teaching effectiveness.
* **Accessibility Features**: Add tools for students with disabilities (e.g., text-to-speech, keyboard navigation).
* **Data Export & Reporting**: Allow export of performance data and quiz scores in Excel or PDF formats for institutional use.

## 6.4 Future Work

While the current system lays a strong foundation, several enhancements can be considered in future versions:

1. **Integration with External Learning Platforms**  
   Future work could involve integrating with APIs from platforms like Coursera, Udemy, or Moodle to extend the content base and recommendations.
2. **AI Chatbot for Support**  
   Embedding a chatbot using NLP models can provide 24/7 assistance to students navigating the platform or selecting courses.
3. **Adaptive Learning Paths**  
   Implement dynamic curriculum adjustment based on ongoing learner performance, behavior, and preferences.
4. **Blockchain Certification**  
   Extend the platform to issue blockchain-verified certificates upon course completion, enhancing certificate credibility.
5. **Multilingual Support**  
   Add localization features to support students from different regions and improve inclusivity.
6. **Offline Access**  
   Future versions may support offline content synchronization for users with limited internet access.

# 7 Bibliography

[1] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002. doi: 10.1023/A:1021240730564

[2] A. M. Hariri, N. Y. Yen, and N. Sulaiman, “A survey on context-aware recommender systems based on computational intelligence techniques,” *The Journal of Systems and Software*, vol. 95, pp. 1–24, 2014. doi: 10.1016/j.jss.2014.03.030

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint*, arXiv:1301.3781, 2013. [Online]. Available: <https://arxiv.org/abs/1301.3781>

[4] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, “Recommender Systems Handbook,” 2nd ed. Springer, 2015. doi: 10.1007/978-1-4899-7637-6

[5] T. G. Dietterich, “Machine Learning for Sequential Data: A Review,” in *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, 2002, pp. 15–30. doi: 10.1007/3-540-70659-3\_2

[6] M. J. Pazzani and D. Billsus, “Content-Based Recommendation Systems,” in *The Adaptive Web*, Springer, 2007, pp. 325–341. doi: 10.1007/978-3-540-72079-9\_10

[7] Laravel Documentation, “The PHP Framework for Web Artisans,” [Online]. Available: <https://laravel.com/docs>

[8] RubixML, “A PHP machine learning library,” [Online]. Available: <https://rubixml.com/>

[9] Chart.js, “Simple yet flexible JavaScript charting for designers & developers,” [Online]. Available: <https://www.chartjs.org/>

[10] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, “Recommender system application developments: A survey,” *Decision Support Systems*, vol. 74, pp. 12–32, 2015. doi: 10.1016/j.dss.2015.03.008

[11] Y. Lu, Y. Liu, K. J. Lin and C. Liu, "A personalized e-learning system based on a user interest model and collaborative filtering," *2018 IEEE International Conference on Web Services (ICWS)*, San Francisco, CA, USA, 2018, pp. 306–313. [Online]. Available: <https://ieeexplore.ieee.org/document/8456372>

[12] H. Al-Zoube, “E-Learning on the Cloud,” *International Arab Journal of e-Technology*, vol. 1, no. 2, pp. 58–64, 2009.

[13] R. D. Vale and H. M. Anggoro, “Design of e-learning recommendation system using content-based filtering method,” *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia, 2021, pp. 46–51. [Online]. Available: <https://ieeexplore.ieee.org/document/9703681>

[14] S. Huang and Y. Fang, “A personalized recommendation model for online learning based on knowledge map and learning style,” *2019 10th International Conference on E-business, Management and Economics (ICEME)*, Beijing, China, 2019, pp. 93–98.

[15] H. Wu, L. Li, and L. Zheng, “Learning path recommendation system based on student behavior data and knowledge graph,” *2022 IEEE 6th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2022, pp. 1037–1042.

[16] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” *EBSE Technical Report*, 2007.

# 8 Appendices

**Appendix 1: Questionnaire for Students**

**Title:** E-Learning Platform User Experience and Expectations

**Instructions:** Please answer the following questions honestly. Your responses will be kept confidential and used only for academic research purposes.

**Section 1: Demographics**

1. What is your age group?  
    ☐ 18–24 ☐ 25–30 ☐ 31–35 ☐ 36+
2. What is your level of study?  
    ☐ Undergraduate ☐ Postgraduate ☐ Other (specify): \_\_\_\_\_\_\_\_\_\_
3. Have you used any e-learning platform before?  
    ☐ Yes ☐ No

**Section 2: Platform Usage**

1. How often do you use e-learning platforms?  
    ☐ Daily ☐ Weekly ☐ Monthly ☐ Rarely
2. What challenges do you face while using current e-learning platforms?  
    ☐ Difficult navigation  
    ☐ Poor feedback on progress  
    ☐ No personalization  
    ☐ Boring content  
    ☐ Other: \_\_\_\_\_\_\_\_\_\_\_
3. How important is personalized course recommendation to you?  
    ☐ Very Important ☐ Important ☐ Neutral ☐ Not Important
4. How do you prefer to receive feedback on your learning progress?  
    ☐ Visual dashboards ☐ Emails ☐ Mobile notifications ☐ In-platform alerts
5. Would you trust an AI system to recommend courses or topics based on your performance?  
    ☐ Yes ☐ No ☐ Maybe

**Appendix 2: Interview Guide for Lecturers and Admins**

**Title:** Expert Insights on AI-Enhanced E-Learning

**Objective:** To understand academic expectations and concerns regarding the implementation of an AI-enhanced learning platform.

**Interview Questions:**

1. What is your experience with current e-learning platforms in your institution?
2. What major limitations do you see in how they support student learning?
3. How would you feel about a system that recommends content based on a student’s performance?
4. What kind of feedback do you think students need the most to succeed?
5. Would automation in grading quizzes and recommending topics reduce your workload?
6. What concerns might you have with implementing AI in learning platforms?
7. What features would you expect from an admin dashboard in such a system?
8. Do you think such a system would benefit students with different learning speeds? Why or why not?

# User Manual

**System Name: AI-Enhanced E-learning Recommendation System**

**1. Introduction**

This system is designed to enhance the learning experience by providing personalized course recommendations based on user interests and quiz performance. It is tailored for both **students** and **administrators**.

**2. System Requirements**

**Minimum Requirements:**

* **Browser**: Chrome, Firefox, Safari (latest versions)
* **Operating System**: Windows 10+, macOS, Linux, Android, iOS
* **Screen Resolution**: Minimum 1024x768

**3. User Roles**

* **Admin**: Manages users, courses, content, and generates reports.
* **Student**: Enrolls in courses, takes quizzes, views grades, and receives recommendations.

**4. User Instructions**

**4.1. Student Manual**

**Step 1: Register**

* Go to the registration page.
* Fill in your details (name, email, password).
* Submit and verify your email.

**Step 2: Login**

* Enter your registered email and password.
* Click "Login" to access your dashboard.

**Step 3: Enroll in Courses**

* Browse available courses.
* Click "Enroll" to register for a course.

**Step 4: View Course Content**

* Open enrolled courses to access materials like videos, PDFs, and notes.

**Step 5: Take Quizzes**

* Navigate to the quiz section under each topic.
* Answer questions and submit for evaluation.

**Step 6: View Performance**

* View quiz scores and progress via your dashboard.

**Step 7: Get Recommendations**

* Based on your performance and interests, system recommends suitable courses/topics.

**4.2. Admin Manual**

**Step 1: Login**

* Use admin credentials to log in via the admin portal.

**Step 2: Dashboard**

* View metrics such as number of users, enrollments, quiz attempts, and performance.

**Step 3: Manage Users**

* Add new users or delete inactive ones.

**Step 4: Manage Courses**

* Add, edit, or remove courses and associated topics.

**Step 5: Manage Quizzes**

* Create quizzes for topics.
* Add questions, define correct answers, and assign scores.

**Step 6: Generate Reports**

* Download or view reports by course, user performance, and overall progress.

End of Manual

# Sample Code

### Student Dashboard Interface

@extends('layouts.app')

@section('content')

<div class="p-6">

    <h2 class="text-2xl font-bold mb-6">Student Dashboard</h2>

    <div class="grid grid-cols-1 md:grid-cols-3 gap-6 mb-6">

        <div class="bg-white p-4 rounded shadow">

            <h3 class="text-lg font-semibold">Enrolled Courses</h3>

            <p class="text-3xl mt-2">{{ $courses->count() }}</p>

        </div>

        <div class="bg-white p-4 rounded shadow">

            <h3 class="text-lg font-semibold">Quizzes Taken</h3>

            <p class="text-3xl mt-2">{{ $quizzesTaken }}</p>

        </div>

        <div class="bg-white p-4 rounded shadow">

            <h3 class="text-lg font-semibold">Average Score</h3>

            <p class="text-3xl mt-2">

                {{ $scores->count() > 0 ? round($scores->avg(), 2) : 'N/A' }}%

            </p>

        </div>

    </div>

    {{-- Chart Section --}}

    <div class="bg-white p-6 rounded shadow mb-6">

        <h3 class="text-xl font-semibold mb-4">Performance Overview</h3>

        <canvas id="performanceChart" height="150"></canvas>

    </div>

</div>

@endsection

@section('scripts')

<script src="{{ asset('assets/custom-chart.js')}}"></script>

<script>

    const ctx = document.getElementById('performanceChart').getContext('2d');

    // Create a gradient for a more dynamic and colorful chart

    const gradient = ctx.createLinearGradient(0, 0, 0, 400);

    gradient.addColorStop(0, 'rgba(99, 102, 241, 0.5)'); // Blue color

    gradient.addColorStop(1, 'rgba(234, 88, 12, 0.5)'); // Orange color

    const chart = new Chart(ctx, {

        type: 'bar',

        data: {

            labels: {!! json\_encode($labels) !!},

            datasets: [{

                label: 'Scores (%)',

                data: {!! json\_encode($scores) !!},

                backgroundColor: gradient, // Use gradient for background color

                borderColor: 'rgba(59, 130, 246, 1)', // Border color (blue)

                borderWidth: 1,

                hoverBackgroundColor: 'rgba(59, 130, 246, 0.8)',

                hoverBorderColor: 'rgba(59, 130, 246, 1)',

                hoverBorderWidth: 2

            }]

        },

        options: {

            responsive: true,

            scales: {

                y: {

                    beginAtZero: true,

                    max: 100,

                    ticks: {

                        stepSize: 10, // Step size for the Y-axis

                        color: '#333' // Color for Y-axis labels

                    }

                },

                x: {

                    ticks: {

                        color: '#333' // Color for X-axis labels

                    }

                }

            },

            plugins: {

                tooltip: {

                    backgroundColor: 'rgba(0, 0, 0, 0.7)', // Dark background for tooltips

                    titleColor: '#fff',

                    bodyColor: '#fff',

                    borderColor: '#fff', // White border for tooltip

                    borderWidth: 1

                }

            }

        }

    });

</script>

@endsection