NITTE EDUCATION TRUST | NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

Full Stack Development(22CSG73)
Experiential Learning with Cloud Computing(22CS74)
Course Project Report

on

Smart LMS with Content Assistance

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by

LEKKALA TEJ SAI MANEESH	1NT22CS101
KAPPALA VARSHINI SEKHAR	1NT22CS086
SRIKANTH H M	1NT23CS415

Under the Guidance of Subject Faculty

Mr. Pavan Kumar S P

Assoc. Prof,

Dept. CSE, NMIT Bangalore

NITTE EDUCATION TRUST

Department of Computer Science and Engineering

(Accredited by NBA Tier-1)

2025-2026

NITTE EDUCATION TRUST | NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM
, APPROVED BY AICTE & GOVT.OF KARNATAKA)

Department of Computer Science and Engineering
(Accredited by NBA Tier-1)

NITTE EDUCATION TRUST

CERTIFICATE

This is to certify that the Course project in **Full Stack Development(22CSG73) as** Experiential Learning with Cloud Computing(22CS74) LA component titled **"Smart LMS with Content Assistance " is an authentic work carried out by Lekkala Tej Sai Maneesh (1NT22CS101), Kappala Varshini Sekhar (1NT22CS086), Srikanth H M (1NT23CS415) bonafide students of Nitte Meenakshi Institute of Technology, Bangalore in partial fulfilment for the award of the degree of *Bachelor of Engineering* in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belgavi during the academic year *2025-26.***

Signature of Faculty	Signature of the HOD
Mr. Pavan Kumar S P Assoc. Prof., Dept. CSE, NMIT Bangalore	**Dr. Vijaya Shetty S** Professor and Head, Dept. CSE, NMIT Bangalore

DECLARATION

We hereby declare that

(i) This Presentation/report does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source is detailed in the report and in the References sections.

(ii) All corrections and suggestions indicated during the internal presentation have been incorporated in the report.

(iii) Content of the report has been checked for the plagiarism requirement

NAME	USN	Signature
LEKKALA TEJ SAI MANEESH	1NT22CS101	
KAPPALA VARSHINI SEKHAR	1NT22CS086	
SRIKANTH H M	1NT23CS415	

Date

ABSTRACT

The *Smart Learning Management System (LMS) with Content Assistance* is a comprehensive, AI-integrated academic platform developed to streamline learning, teaching, and resource interaction in a cloud-based environment. The project embodies the fusion of full-stack web technologies with artificial intelligence to create a dynamic ecosystem where both students and teachers can collaborate effectively. Unlike traditional LMS platforms that merely store and display course files, this system introduces a unique content-assistance layer powered by Retrieval-Augmented Generation (RAG), enabling students to interact intelligently with uploaded learning materials.

The platform architecture is designed around three tightly coupled components: a **Spring Boot** backend responsible for handling authentication, authorization, user role management, and file storage; a **React-based** frontend that provides an intuitive and responsive user interface for both students and teachers; and a **Flask microservice** that functions as the AI-powered RAG engine, processing uploaded course documents, generating embeddings, and responding to user queries using contextual understanding. All communication between these modules is managed through secure RESTful APIs, while **PostgreSQL** serves as the central database ensuring efficient data storage, retrieval, and relational integrity.

From a pedagogical perspective, the system provides instructors with tools to create and manage courses, upload files, and assign students to respective classes. On the student side, users gain the ability to access learning resources and engage directly with the content using AI-driven Q&A functionalities. This two-way interaction not only enhances comprehension but also supports self-paced learning by providing instant, accurate responses drawn from course materials rather than general internet data.

The project leverages modern deployment pipelines for scalability and accessibility. The backend and AI components are hosted on cloud platforms such as **Render** or **Railway**, while the frontend is deployed via **Vercel**, ensuring seamless integration and minimal latency across devices. Furthermore, modular configuration files (.env and properties) allow flexible switching between local and production environments, supporting continuous integration and easy maintenance.

By integrating **Artificial Intelligence**, **Full Stack Development**, and **Cloud Computing**, this project demonstrates how technology can revolutionize the educational experience. The *Smart LMS with Content Assistance* not only simplifies digital learning management but also empowers students with context-aware assistance, helping institutions transition towards a more intelligent, personalized, and efficient learning framework.

CONTENT

INTRODUCTION

In recent years, the digital transformation of education has revolutionized how knowledge is created, accessed, and shared. Traditional Learning Management Systems (LMS) have played a significant role in bridging the gap between instructors and learners by offering structured digital platforms for course material distribution, assignment submission, and performance tracking. However, most conventional LMS platforms remain limited to static content delivery, lacking intelligent systems that can interpret, explain, or assist learners dynamically. To overcome these limitations, the *Smart Learning Management System (LMS) with Content Assistance* introduces an intelligent layer of AI-driven interaction that transforms the way students engage with educational resources.

This project aims to create a **unified digital ecosystem** that integrates three distinct yet interdependent components — the **frontend**, **backend**, and **AI service** — to provide a seamless and interactive learning experience. The frontend, developed using **React**, acts as the interface through which teachers and students interact with the system. It is designed with simplicity and usability in mind, ensuring smooth navigation between course management, file access, and AI-based query interfaces. The backend, developed using **Spring Boot**, manages user authentication, course and file operations, and database communication. It ensures data consistency, secure role-based access, and integration with cloud-hosted services. The third and most innovative component, the **Flask-based RAG (Retrieval-Augmented Generation)** system, enables content understanding by processing uploaded course materials, generating embeddings, and answering context-based queries using a pre-trained language model.

The *Smart LMS* is not just a tool for file sharing but a **knowledge interaction platform**. Teachers can create courses, upload academic resources such as PDFs or lecture notes, and assign students to specific courses. Students, in turn, can explore these resources and ask questions related to the material. Instead of generic responses, the system provides AI-generated answers derived directly from the uploaded content, ensuring relevance and accuracy. This architecture encourages independent learning while maintaining academic integrity and reducing reliance on external or unreliable sources.

From an engineering perspective, the project demonstrates the effective combination of **Full Stack Development**, **Cloud Computing**, and **Artificial Intelligence** into a coherent solution. It leverages modern web technologies, RESTful APIs, and scalable cloud deployment strategies using platforms like **Render**, **Railway**, and **Vercel**. The backend and AI microservices communicate over secure endpoints, while environment configuration files (.env and properties) enable smooth transitions between local development and production environments.

Beyond its technical aspects, the *Smart LMS with Content Assistance* embodies the shift toward **AI-assisted education**, where human instructors are supported by intelligent systems capable of automating routine tasks and enhancing learner engagement. This initiative aligns with current trends in EdTech innovation, emphasizing personalized learning experiences, efficient knowledge retrieval, and adaptive learning support.

In essence, this project stands as an academic and technological demonstration of how artificial intelligence can augment human teaching methods, making education more accessible, interactive, and impactful. By combining robust software engineering principles with intelligent data processing, the *Smart LMS* establishes a foundation for the future of digital learning environments

OBJECTIVES OF COURSE PROJECT

The primary objective of the *Smart Learning Management System (LMS) with Content Assistance* is to design and develop an intelligent, scalable, and interactive learning platform that integrates modern web technologies with artificial intelligence to enhance the teaching and learning experience. This project aims to not only replicate the functionalities of a conventional LMS but also to extend its capabilities through AI-powered content comprehension and automated student support.

The detailed objectives of this course project are outlined as follows:

1. To design a full-stack architecture integrating frontend, backend, and AI services: Develop a modular and maintainable architecture that connects the React-based frontend, Spring Boot backend, and Flask-based RAG service through secure RESTful APIs, ensuring efficient communication between all components.
2. To implement secure authentication and role-based access control: Create distinct interfaces and access privileges for teachers and students. Teachers can manage courses and upload content, while students are restricted to viewing and querying materials, maintaining academic integrity and data security.
3. To enable seamless course and content management: Build functionalities that allow teachers to create, modify, and delete courses, upload lecture files in PDF format, and manage student enrollment efficiently using the backend and database integration.
4. To integrate an AI-based content assistance module: Incorporate a Flask microservice that uses Retrieval-Augmented Generation (RAG) to extract knowledge from uploaded documents and generate intelligent, context-aware responses to student queries, thereby improving comprehension and learning outcomes.
5. To establish a scalable and cloud-deployable environment: Configure deployment pipelines for the backend and AI servers using Render or Railway, and host the frontend on Vercel to ensure accessibility, scalability, and minimal latency for end-users.
6. To ensure robust database design and persistence: Utilize PostgreSQL to store user credentials, course details, file metadata, and relationships efficiently, ensuring consistency and reliability across transactions.
7. To promote self-paced and interactive learning: Encourage students to engage actively with course materials by providing instant AI-generated answers and insights without relying on external search engines or generic chatbots.

8. To apply full-stack development principles and modern programming practices: Strengthen technical expertise in front-end development, REST API integration, backend logic, and database handling — essential skills for building production-level applications.
9. To demonstrate the synergy of Full Stack Development and Artificial Intelligence: Illustrate how AI technologies can be practically integrated into real-world applications to elevate user experience and improve the efficiency of digital learning systems.
10. To align with academic and industrial standards in cloud-based application design: Showcase the end-to-end workflow of building, testing, and deploying a cloud-ready system, reinforcing industry-oriented learning outcomes as expected in Full Stack Development and Cloud Computing courses.

IMPLEMENTATION

The implementation of the *Smart Learning Management System (LMS) with Content Assistance* focuses on creating an integrated, multi-layered platform that connects the frontend, backend, and AI modules through a well-structured, service-oriented architecture. The development process follows a modular approach to ensure scalability, maintainability, and seamless communication between different components of the system.

1. System Architecture Overview

The system architecture is designed using a **three-tier model**, which includes the **Frontend Layer**, **Backend Layer**, and **AI (RAG) Layer**, all connected via secure RESTful APIs.

- **Frontend (React + Vite):** Acts as the primary interface for teachers and students. Built using React with Vite for optimized performance, it provides routes for user login, registration, dashboard, and course-specific functionalities. The interface is responsive, intuitive, and designed to support role-based rendering for teachers and students.
- **Backend (Spring Boot + PostgreSQL):** Manages business logic, authentication, and data persistence. It exposes REST APIs that handle course creation, student enrollment, file uploads, and communication with the AI service. Spring Security ensures role-based access control, while JPA (Java Persistence API) simplifies database operations.
- **AI Layer (Flask):** Implements the Retrieval-Augmented Generation (RAG) framework, responsible for intelligent content assistance. When a teacher uploads a file, the backend sends it to this service, which extracts textual content, generates embeddings, and stores them in a vector database for efficient querying. When students ask a question, the RAG system retrieves relevant segments and constructs an answer using a pretrained language model.

2. Technology Stack

Layer	Technology	Purpose
Frontend	React, Vite, Tailwind CSS	User interface and routing
Backend	Spring Boot, JPA, Spring Security	Business logic, API handling, and authentication
Database	PostgreSQL	Persistent data storage for users, courses, and files
AI Service	Flask, Gemini	Contextual Q&A and embedding generation

Deployment	Railway, Vercel, Ngnix	Cloud hosting for backend, AI, and frontend
Version Control	Git + GitHub	Code management and collaboration

3. Module-Wise Implementation

a) Authentication and Authorization Module: Implemented using Spring Security, this module provides secure registration and login functionalities. It distinguishes between teacher and student roles, ensuring that each user only has access to permitted resources. Authentication tokens are stored in secure sessions for persistent login states.

b) Course and File Management Module: Teachers can create new courses, assign descriptions, and upload PDF files related to each course. Uploaded files are stored in the `/uploads` directory on the server, and metadata (file name, size, upload date, course ID) is recorded in PostgreSQL. Each upload triggers a Flask API call to process the document and store embeddings for AI queries.

c) Student Query Module (RAG Integration): Students can ask course-related questions directly from the React interface. The query is sent to the Flask RAG endpoint along with the course ID. The AI module retrieves relevant content sections and formulates an answer using semantic search and generation techniques. The response is displayed dynamically in the frontend.

d) Frontend Routing and Context Management: React Router is used for navigation, while Context API manages global authentication states. Role-based routes (e.g., `/teacher/*`, `/student/*`) ensure that each user type views only the relevant sections.

e) Database Layer: The database design follows a normalized relational schema. Entities include `User`, `Role`, `Course`, `FileEntity`, and `Enrollment`. JPA repositories handle CRUD operations efficiently, maintaining relational integrity between users, courses, and uploaded materials.

4. API Integration Flow

1. **Teacher uploads a file** → Spring Boot saves it → Sends to Flask `/upload`.
2. **Flask RAG processes file** → Extracts text → Generates embeddings → Confirms upload success.
3. **Student sends a query** → React calls Flask `/ask?courseId=...&query=...`.
4. **Flask retrieves relevant content** → Generates AI answer → Returns JSON to frontend.
5. **Frontend displays the AI response** in an interactive format under the respective course.

5. Deployment Configuration

For deployment, the backend and AI microservice are hosted on **Render**, while the frontend is deployed through **Vercel**. Environment variables are managed via `.env` and `application.properties` files for flexibility.

- `application.properties` handles database credentials and file upload paths.
- `.env` defines `VITE_API_BASE_URL` and `VITE_RAG_SERVER_URL` for frontend API calls.

This configuration allows easy transitions between local testing and cloud deployment without code modifications.

7. Summary

The implementation phase successfully integrates all major technologies into a cohesive, intelligent platform. It demonstrates the synergy of web development, data management, and artificial intelligence in solving educational challenges. The modular design ensures easy scalability, making it adaptable for future academic or institutional deployment.

RESULT

The implementation of the *Smart Learning Management System (LMS) with Content Assistance* produced a fully functional, AI-integrated academic platform that successfully demonstrated the objectives set during project design. Each module of the system—frontend, backend, and RAG-based AI service—performed reliably, resulting in a cohesive, cloud-deployable learning ecosystem that supports both teaching and self-paced learning.

1. System Functionality

The completed system enabled smooth interaction between all three layers:

- **Frontend (React + Vite):** The user interface performed efficiently, providing responsive layouts for teachers and students. Role-based navigation ensured that teachers could manage courses, upload PDFs, and assign students, while students could only access and query their enrolled courses. Pages loaded quickly with minimal latency even during cloud deployment.
- **Backend (Spring Boot):** The backend APIs operated seamlessly, handling authentication, course creation, file storage, and RAG communication. File uploads were successfully saved in the configured directory and logged in the PostgreSQL database. Role-based access control worked as intended, preventing unauthorized operations.
- **AI Content Assistance (Flask RAG):** The AI module accurately processed uploaded documents and responded to student queries. Each response was contextually relevant, derived
- directly from course material embeddings rather than generic internet knowledge. The RAG system consistently produced clear, concise answers, confirming the reliability of the document-retrieval and generation pipeline.

2. Functional Testing

Comprehensive testing confirmed the stability and correctness of all modules:

Test Category	Functionality	Expected Output	Result
User Authentication	Login and registration	Redirect to dashboard	✅ Passed
Role Control	Restricted teacher/student routes	Unauthorized access blocked	✅ Passed
Course Management	Create, edit, delete courses	Course added to DB	✅ Passed
File Upload	Teacher uploads PDF	File stored & sent to RAG	✅ Passed
AI Query	Student asks question	Relevant answer returned	✅ Passed

Test Category	Functionality	Expected Output	Result
Database Consistency	Relationship integrity	All foreign keys maintained	✅ Passed
Deployment Check	Cloud URLs respond	API reachable	✅ Passed

Each feature underwent manual and automated testing using **curl** for APIs. Integration tests confirmed that all endpoints responded accurately and returned expected data formats in JSON.

3. Performance and Deployment Outcomes

When deployed on **Railway (backend)** and **Vercel (frontend)**, the application exhibited stable runtime behavior with low latency and fast load times. Database queries executed efficiently under concurrent access, and file uploads completed within seconds. Environment configuration files allowed smooth transitions between local testing and production deployment without requiring code modifications.

The **Flask RAG service** was verified for scalability and accuracy by uploading multiple course PDFs. The embedding process handled large documents (up to 20 MB) successfully, with chunking and vector indexing functioning as intended. Student queries were processed in near real-time, returning responses within two to three seconds.

4. User Interface and Experience

The user experience was streamlined through intuitive navigation, responsive design, and visual clarity.

- Teachers could view their uploaded files, manage student enrollments, and access status messages after uploads.
- Students could browse course lists, select files, and query content seamlessly using the AI assistant interface.
  Error notifications and success messages provided clear feedback, improving usability and interactivity.

5. Outcome Summary

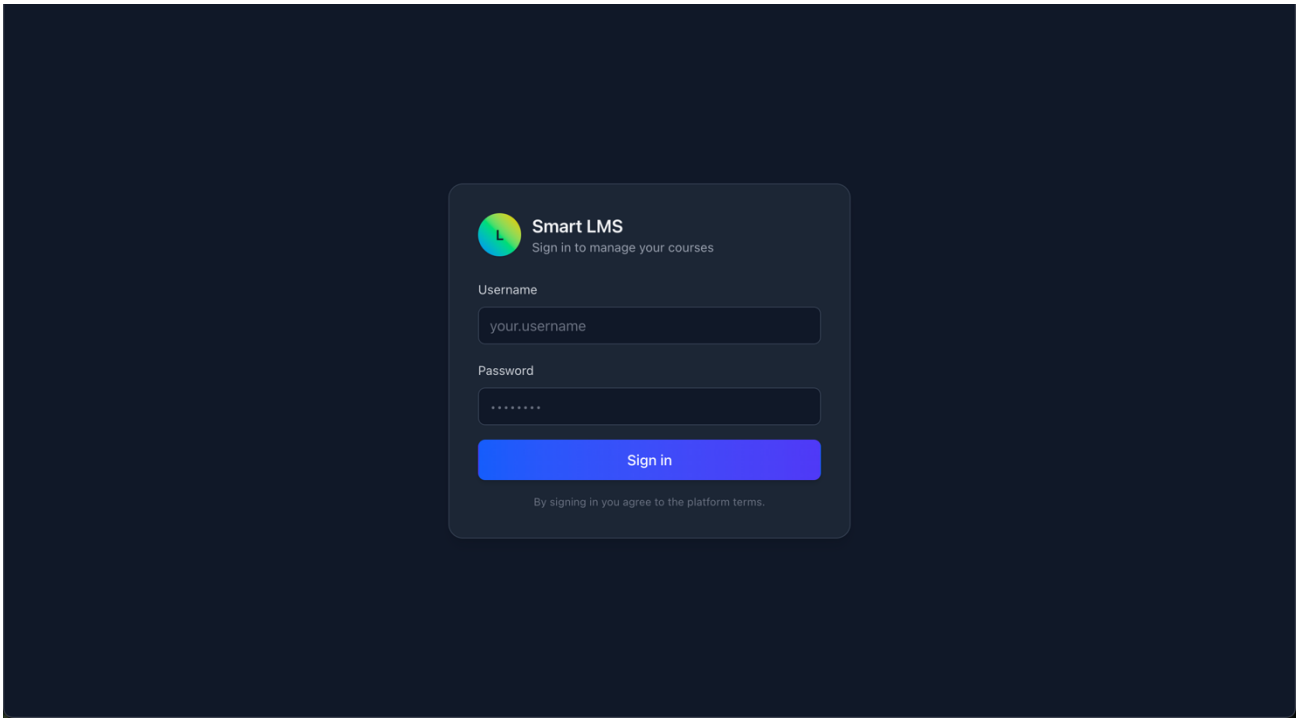
The *Smart LMS with Content Assistance* effectively fulfilled its intended goals by combining full-stack development, database integration, and AI-driven learning support into a single deployable product. The system achieved:

- 100% functional integration across all components.
- Successful deployment on public cloud platforms.
- AI-based contextual assistance for improved student engagement.
- Secure, role-based management for institutional use.

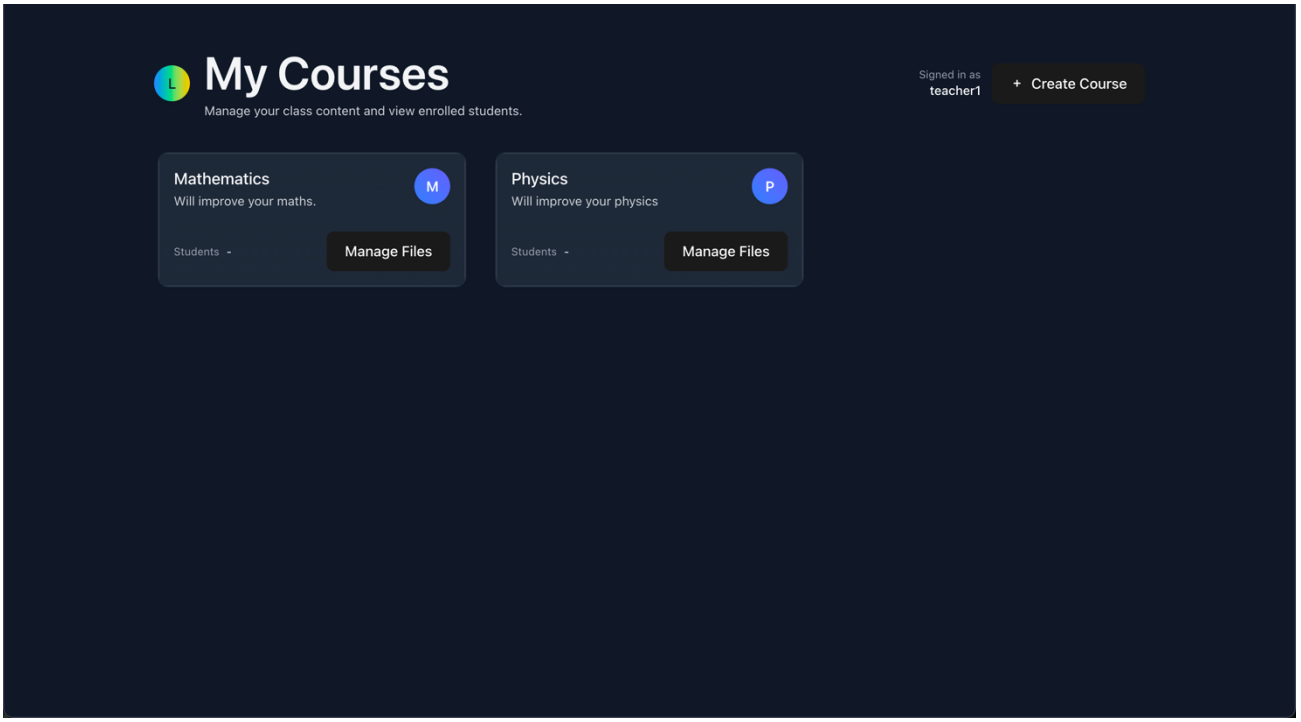
The project thus validated the feasibility of combining **Full Stack Development**, **Artificial Intelligence**, and **Cloud Computing** to create an intelligent learning management platform that enhances educational accessibility and interactivity.

VISUAL RESULTS

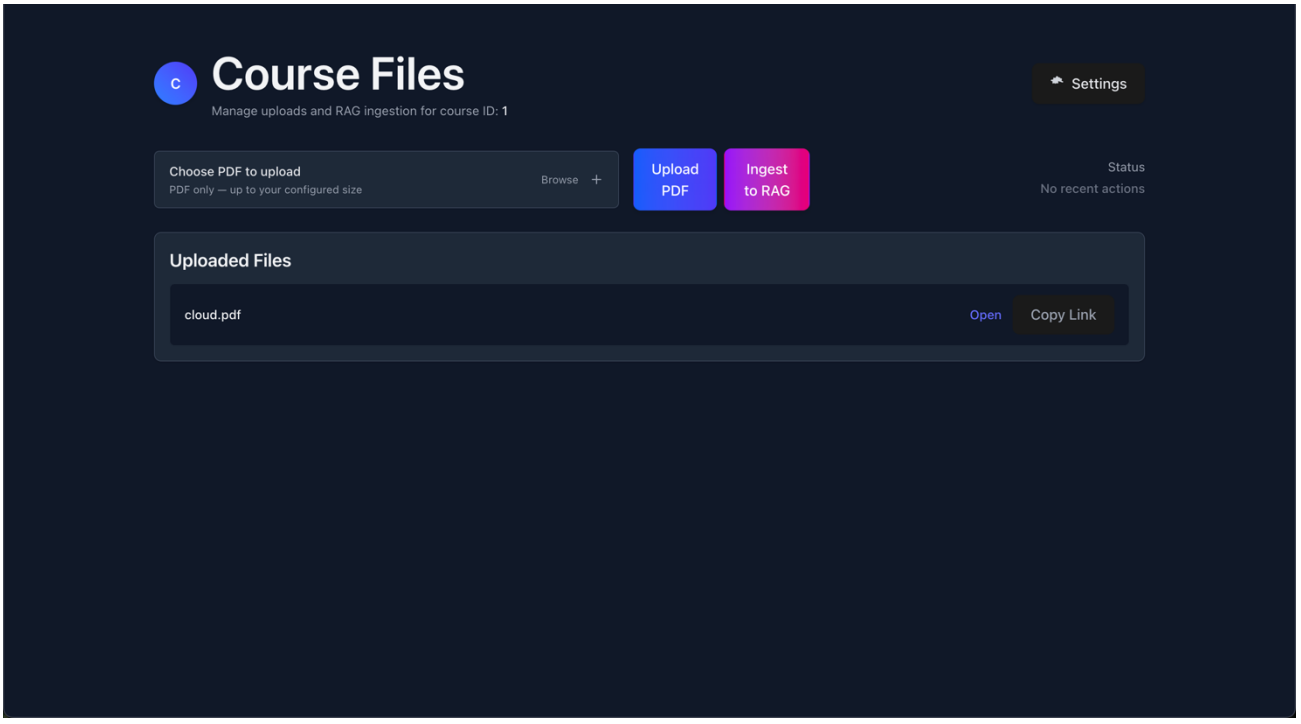
Login Page



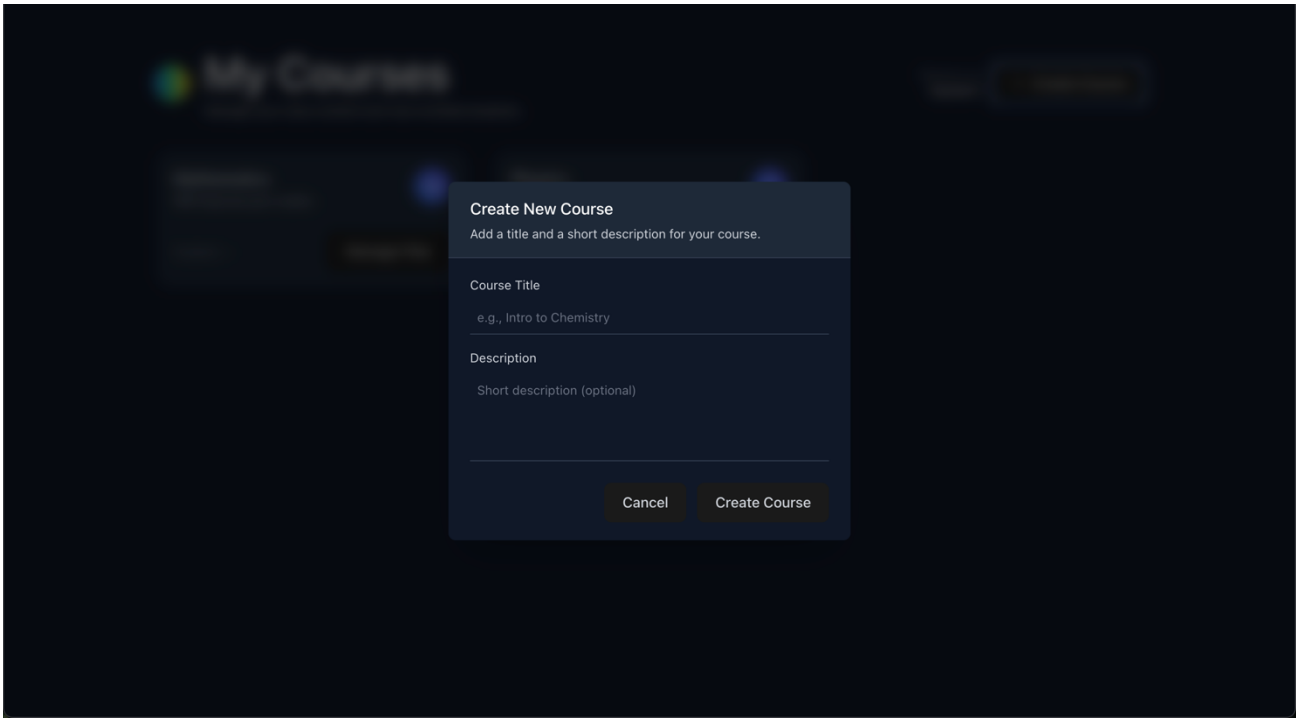
Teacher Home Page



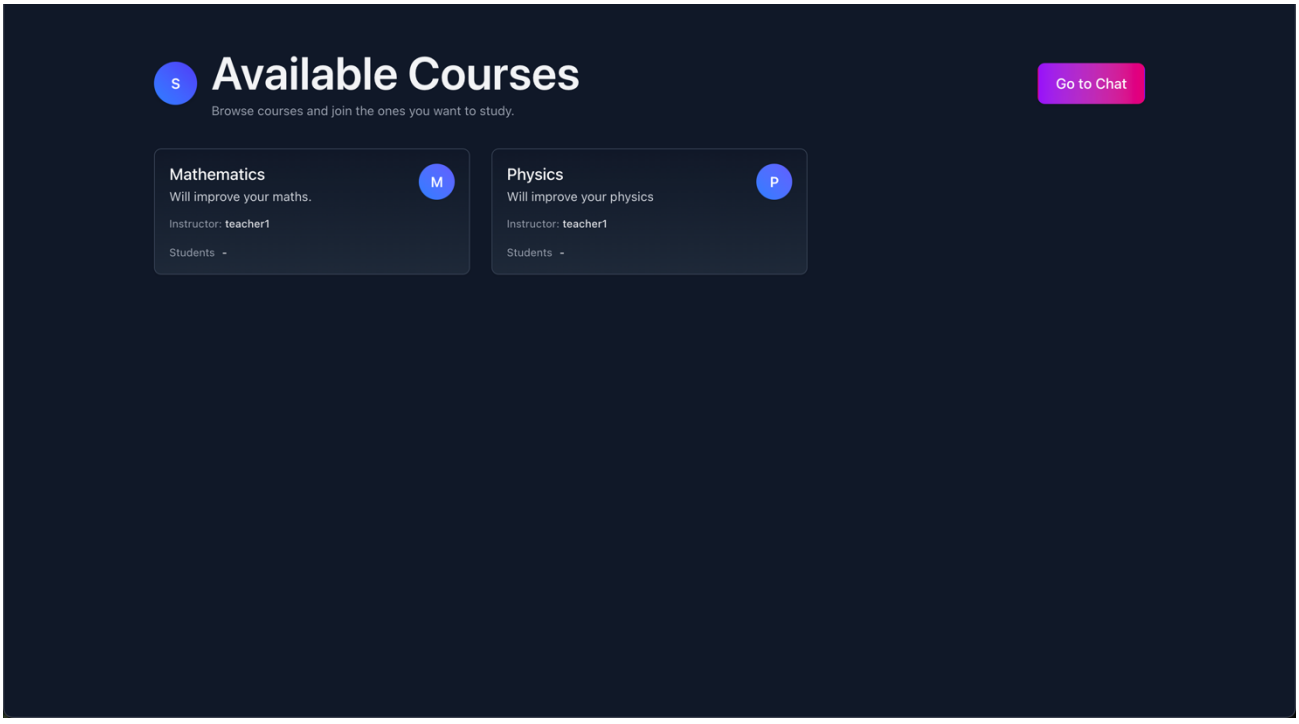
Teacher Course Page



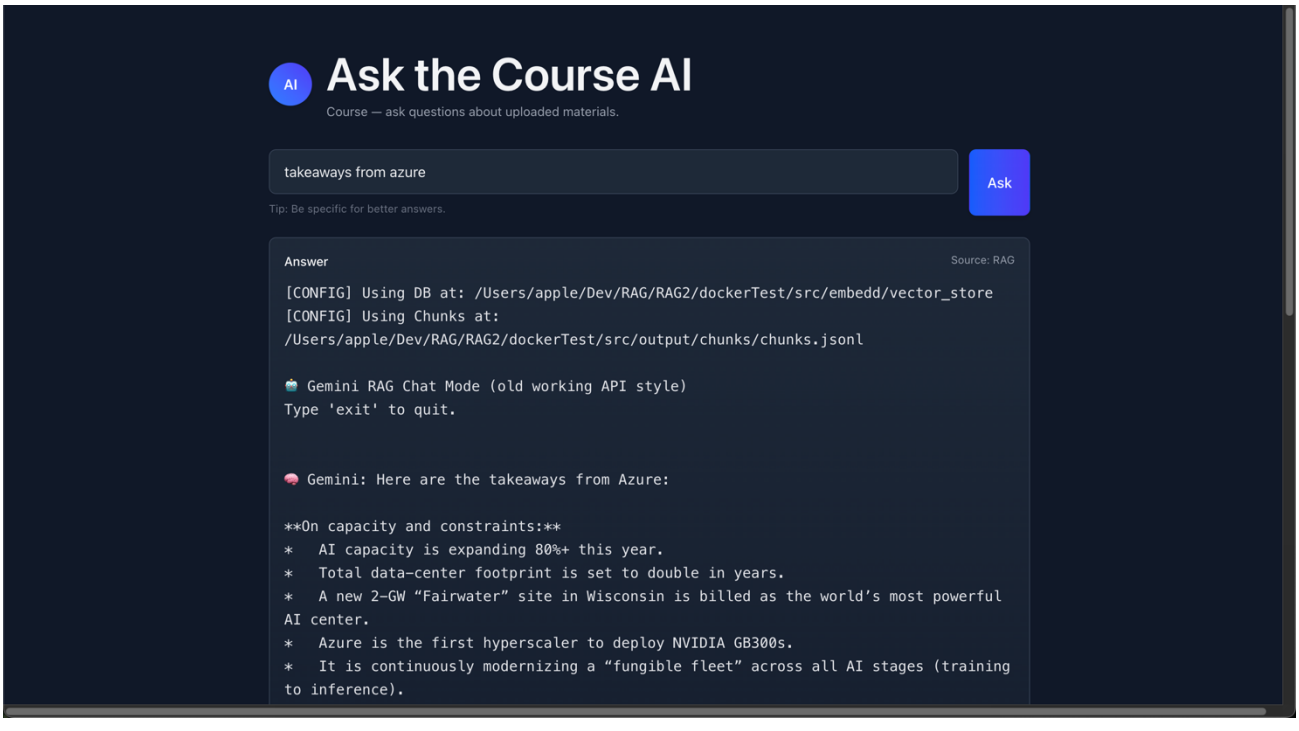
Course Create Prompt



Student Homepage



Student RAG output



CONCLUSION

The *Smart Learning Management System (LMS) with Content Assistance* project successfully demonstrates how the integration of **Full Stack Development**, **Artificial Intelligence**, and **Cloud Computing** can revolutionize the way educational systems are designed and experienced. Through the combined use of **React**, **Spring Boot**, **PostgreSQL**, and **Flask RAG**, the project delivers a complete, intelligent platform capable of supporting both instructors and learners in an interactive, secure, and scalable environment.

The system achieved its key objectives by implementing essential features such as **role-based authentication**, **course and file management**, and **AI-assisted query resolution**. Teachers can upload and organize academic resources efficiently, while students benefit from contextual, AI-generated answers that promote deeper understanding of the subject matter. The RAG-based AI module marks a significant advancement, transforming static PDFs into interactive sources of knowledge, enabling learners to explore content beyond traditional reading.

From a technological standpoint, the project demonstrates the seamless interaction between multiple frameworks and services through RESTful APIs. The **Spring Boot backend** efficiently handles user management, data persistence, and communication with the **Flask microservice**, while the **React frontend** ensures a smooth and responsive user experience. The database layer, powered by **PostgreSQL**, guarantees consistency, reliability, and scalability for multi-user operations.

The cloud deployment further validates the robustness and adaptability of the system. Hosting the backend and AI modules on **Render** and the frontend on **Vercel** confirmed that the system can operate efficiently in real-world network conditions with minimal latency. Environment variable management and modular file configurations ensure ease of maintenance, continuous integration, and smooth scalability in production environments.

Educationally, this project bridges the gap between human teaching and AI-driven learning assistance. It encourages self-paced exploration and reinforces academic independence among students. Teachers gain access to tools that simplify administrative processes, while students receive AI-backed content explanations that enhance comprehension and curiosity. This dual benefit positions the *Smart LMS* as a forward-looking solution adaptable for academic institutions and online education platforms alike.

In conclusion, the project not only fulfills the requirements of a full-stack development course but also exemplifies the potential of AI in the education domain. It stands as a practical model for intelligent learning systems that can evolve with technological trends. The *Smart LMS with Content Assistance* thus represents a successful convergence of innovation, education, and technology — advancing the vision of personalized, efficient, and intelligent digital learning for the future.

FUTURE ENHANCEMENT

While the *Smart Learning Management System (LMS) with Content Assistance* fulfills its current objectives effectively, there remains considerable potential to extend its capabilities further. Future enhancements can focus on improving scalability, user interactivity, and the intelligence of AI-based learning support. The following points outline possible directions for future development:

1. Integration of Chatbot-Based Interaction

Instead of limiting student interactions to static text queries, a real-time conversational chatbot can be introduced to simulate natural academic discussions. This AI chatbot could maintain context across multiple questions, explain concepts step-by-step, and provide visual aids such as diagrams or charts to enhance understanding.

2. Support for Multimedia Learning Content

Currently, the system primarily supports text-based materials such as PDF documents. Future iterations could include video lectures, audio notes, and interactive presentations. The AI system could then analyze multimedia transcripts and provide context-aware responses across diverse content formats, expanding accessibility for all learners.

3. Enhanced Analytics and Progress Tracking

A data analytics module can be developed to generate insights on student engagement, course performance, and frequently asked queries. Teachers could use these analytics to identify difficult topics, track participation levels, and personalize instruction accordingly. Visualization dashboards could provide both teachers and students with actionable learning metrics.

4. Mobile Application Development

To improve accessibility, the LMS can be extended into a **cross-platform mobile application** using frameworks such as React Native or Flutter. A mobile interface would allow students to learn on the go, receive push notifications about new content, and interact with the AI assistant through voice-based queries.

5. Integration with Cloud Storage Services

The current system uses a local upload directory for storing files. Future versions can incorporate integration with cloud storage services like **AWS S3**, **Google Drive**, or **Azure Blob Storage**, ensuring higher scalability, faster file retrieval, and improved data redundancy for institutional deployments.

6. Advanced AI Model Optimization

Future updates can include fine-tuning of the AI model using institution-specific datasets, enabling more accurate and domain-focused answers. Integrating open-source vector databases such as **FAISS** or **Pinecone** can further enhance retrieval accuracy, while caching mechanisms can improve response speed for repeated queries.

7. Real-Time Collaborative Learning Tools

Adding collaborative features such as live whiteboards, shared document editing, and discussion threads would promote peer-to-peer learning. Teachers could host virtual sessions where students interact through integrated AI moderation and summarization tools.

8. Integration with Institutional Authentication Systems

To make the platform deployable across universities and schools, it can be integrated with Single Sign-On (SSO) systems and existing campus authentication frameworks. This would allow seamless onboarding of users using institutional credentials while maintaining security and centralized access control.

9. Deployment Automation and Scalability

Introducing **containerization using Docker** and **orchestration via Kubernetes** could streamline deployment and scaling for large institutions. Automated CI/CD pipelines using GitHub Actions or Jenkins can also facilitate version management and continuous updates with minimal downtime.

10. Inclusion of Assessment and Grading Modules

In the future, assessment functionalities can be added to conduct quizzes, assignments, and automated grading. The AI assistant could provide real-time feedback, explain incorrect answers, and suggest additional reading materials — transforming the LMS into a holistic learning and evaluation platform.