

Internship Report

“E-Guru AI-based Learning Management System”



by

Mr. Sanket G Shanbhag

4SU21AD041

8th Semester

B.E in Artificial Intelligence and Data Science

Under the guidance of

Dr. Antheesh R

Associate Professor of ADE Dept.

Department of Artificial Intelligence and Data Science

SDM Institute of Technology, Ujire – 574 240

2024-25

Acknowledgement

It is my pleasure to express my deepest gratitude to my guide **Mr. Shashanka R H**, and **Dr. Bharati V**, of HAL Management Academy for their supervision and guidance which enabled me to understand and develop this project.

I am indebted to **Mrs. Veena Bhat**, Head of the Department and **Dr. Ashok Kumar T**, Principal, for their advice and suggestions at various stages of the work. I also extend our heartfelt gratitude to **Dr. Antheesh R**, Associate Professor and Project Coordinator for his assistance and the management of SDM Institute of Technology, Ujire, for providing us with a good learning environment, library and laboratory facilities.

Lastly, we take this opportunity to offer our regards to all of those who have supported us directly or indirectly in the successful completion of this project work.

Sanket G Shanbhag

Abstract

The E-Guru platform is a locally deployed, AI-integrated e-learning management system (LMS) designed to enhance digital training through automated content delivery, intelligent administration, and interactive learning experiences. Built using Python and Django for the backend, HTML, CSS, and JavaScript for the frontend, the system utilizes MongoDB as its primary database for flexible and scalable data storage. MongoDB's document-oriented architecture enables efficient handling of diverse training materials, user profiles, and analytics, supporting dynamic querying and seamless scalability as the platform grows.

The platform features role-based dashboards for administrators and users, facilitating course management, user enrollment tracking, real-time analytics, and document resource organization. A core innovation of E-Guru is its LLM-powered chatbot, which employs natural language processing (NLP) and retrieval-augmented generation (RAG) to extract and contextualize information from multi-format training materials, including PDFs, Word documents, PowerPoint presentations, and scanned images. The chatbot integrates optical character recognition (OCR) via Tesseract or PyMuPDF to process text embedded in images, along with structured data extraction for tables, figures, and metadata, ensuring accurate, reference-linked responses.

All AI functionalities, including embedding generation (using Sentence Transformers) and vector storage, operate entirely on local infrastructure, ensuring data privacy, security, and operational autonomy. The platform's modular architecture, combined with its intuitive user interface and intelligent content retrieval mechanisms, positions E-Guru as a scalable, secure, and efficient solution for institutions and organizations seeking personalized, privacy-compliant learning experiences.

Table of Contents

	PageNo.
Abstract	ii
Table of Contents	iii
List of Figures	v
List of Tables	vi
Chapter 1 Introduction	1
Chapter 2 Problem Statement and Objectives	2
2.1 Problem Statement	2
2.2 Objectives	2
Chapter 3 System Requirements	4
3.1 Hardware Requirements	4
3.2 Software Requirements	4
3.3 Methodology Used	4
Chapter 4 System Design and Architecture	6
Chapter 5 System Implementation	12
5.1 Implementation Methodology	12
5.2 System Features	12
5.2.1 Role-Based Dashboards	12
5.2.2 Document Upload and Indexing	12
5.2.3 AI-Powered eGuru Chatbot	13
5.2.4 Interactive Monitoring and Analytics Dashboard	13
5.3 Algorithms and Techniques	13
5.3.1 AI Embedding and Search Algorithms	13
5.3.2 Data Optimization Techniques	13
5.3.3 Model Invocation and API Handling	14
5.3.4 Serialization and Caching	14
Chapter 6 Testing and Validation	15

6.1	Testing Strategies	15
6.2	Test Cases and Results	15
6.2.1	Test Case 1: PDF and Document Upload with Content Extraction	15
6.2.2	Test Case 2: AI Chatbot Response from Indexed Content	16
6.2.3	Test Case 3: Training Enrollment and Dashboard Navigation	17
6.2.4	Test Case 4: Quiz Generation from Uploaded PDF Content	18
6.2.5	Test Case 5: AI-Powered Feedback Summary Generation from Training Feedback	19
6.3	Validation and Verification	20
Chapter 7	Result and Discussion	21
7.1	Output of the Project	21
7.2	Performance Analysis	23
7.3	Comparisons with Existing Solutions	24
Chapter 8	Conclusion and Future Enhancements	25
8.1	Summary of Achievements	25
8.2	Limitations of the Project	25
8.3	Future Enhancements	26
	References	27

List of Figures

Figure No.	Description	Page No.
Figure 4.1	System Architecture	7
Figure 4.2	User / Student Journey	8
Figure 4.3	Admin Journey	9
Figure 4.4	Quiz Generation Flowchart	10
Figure 4.5	LLM Flowchart	11
Figure 6.1	Uploaded PDF's and PPT's	16
Figure 6.3	Training Enrolment waiting for the Admin approval	17
Figure 6.4	Review Generated Quiz Questions	18
Figure 6.5	AI-Powered Summary based of Feedbacks	19
Figure 7.1	Sign Up Page	21
Figure 7.2	Sign In Page	22
Figure 7.3	Creating new users	22
Figure 7.4	eGuru AI assistant	23

List of Tables

Table No.	Description	Page No.
Table 3.1	Hardware Requirements	4
Table 3.2	Software Requirements	4

Chapter 1: Introduction

In recent years, the growing demand for flexible, scalable, and intelligent e-learning solutions has accelerated the development of digital learning management systems (LMS). Traditional LMS platforms, while offering basic functionalities such as course enrollment and content delivery, often fall short in providing personalized learning experiences, intelligent content interaction, and comprehensive data privacy. To address these limitations, the eGuru platform has been developed as an AI-enhanced, locally operated e-learning management system, aimed at bridging the gap between user-centric learning and secure digital education.

The eGuru platform is architected with modern web technologies including Python, Django, HTML, CSS, and JavaScript, offering a clean and intuitive user interface tailored for two distinct roles: administrators and users. Administrators can efficiently manage trainings, monitor user engagement, perform data analytics, and oversee document repositories. Users, on the other hand, can seamlessly browse available trainings, enroll in courses, track their progress, and interact with a built-in intelligent chatbot for contextual learning support.

A standout feature of eGuru is the integration of a Large Language Model (LLM)-powered chatbot that can extract, process, and understand content from uploaded documents in various formats such as PDFs, Word files, and PowerPoint presentations. The system extends its capabilities to handle complex document structures including images, tables, and text extracted via optical character recognition (OCR). Responses generated by the chatbot are enriched with references to the original source file and page number, promoting transparency and trust in the information provided.

Furthermore, eGuru is designed to operate entirely within a local environment, eliminating dependence on external APIs or cloud services. This ensures complete data security, privacy, and regulatory compliance, making the platform suitable for institutions and organizations with stringent data protection requirements.

Through its modular architecture, role-based management, intelligent document interaction, and robust local deployment, the eGuru platform represents a significant advancement toward the future of AI-driven e-learning ecosystems. This paper/report details the system's architecture, core functionalities, technological framework, and potential future enhancements.

Chapter 2: Problem Statement and Objectives

2.1 Problem Statement

The rapid evolution of digital learning has led to an increased demand for intelligent, flexible, and secure e-learning platforms. However, traditional Learning Management Systems (LMS) often suffer from several critical limitations:

1. **Lack of Personalization:** Most LMS platforms offer a one-size-fits-all approach, failing to adapt to individual learning styles or provide tailored content recommendations.
2. **Limited Interactivity:** Conventional systems primarily focus on content delivery without enabling dynamic interactions, leaving learners with passive reading or video-based experiences.
3. **Dependence on External APIs:** Many AI-enhanced learning tools rely on cloud-based services, raising concerns over data privacy, security, and regulatory compliance, especially for institutions handling sensitive information.
4. **Inefficient Content Retrieval:** Users struggle to extract precise information from lengthy documents, as traditional keyword-based search mechanisms often yield irrelevant or incomplete results.
5. **Absence of AI-Driven Assistance:** While some platforms integrate chatbots, they typically lack deep contextual understanding of uploaded course materials, limiting their usefulness in academic or professional training scenarios.

To address these challenges, the eGuru Platform was conceived as an AI-enhanced, locally operated e-learning system that combines document intelligence with an interactive chatbot, enabling secure, personalized, and context-aware learning experiences.

2.2 Objectives

The primary objectives of the eGuru Platform are:

1. **Develop a Localized AI-Powered LMS:**
 - Create a self-contained system that operates entirely within a local environment, eliminating dependence on external cloud APIs while ensuring data privacy and compliance.

- Integrate a Large Language Model (LLM) to enable intelligent document processing and contextual query resolution.

2. Enable Dynamic Document Interaction:

- Support uploads of diverse file formats (PDFs, Word, PowerPoint) with robust text extraction, including OCR for images and tables.
- Implement semantic search and embeddings to allow users to query documents conversationally.

3. Enhance Learning Through AI Assistance:

- Deploy a chatbot that provides accurate, source-referenced responses (e.g., citing document names and page numbers) to foster transparency and trust.
- Automate quiz generation from uploaded materials to reinforce learning outcomes.

4. Ensure Role-Based Accessibility and Usability:

- Design distinct interfaces for administrators (e.g., training management, analytics) and learners (e.g., course enrollment, progress tracking).
- Optimize the platform for responsiveness across devices and browsers.

5. Validate Performance and Scalability:

- Test the system under real-world conditions to ensure low-latency responses, even with large documents or concurrent users.
- Benchmark against existing LMS solutions to demonstrate superior functionality in document comprehension and user engagement.

By achieving these objectives, eGuru aims to bridge the gap between static e-learning content and interactive, AI-driven education, setting a new standard for secure and adaptive digital learning environments.

Chapter 3: System Requirements

3.1 Hardware Requirements

Table 3.1: Hardware Requirements

Component	Minimum Specification	Recommended Specification
Processor (CPU)	Dual-core 2.0 GHz	Quad-core 3.0 GHz or higher
Memory (RAM)	4 GB	8 GB or higher
Storage	10 GB free disk space	20 GB SSD storage
Display	1366 × 768 resolution	1920 × 1080 (Full HD)
Network	Local network connectivity	High-speed local network

3.2 Software Requirements

Table 3.2: Software Requirements

Component	Specification
Operating System	Windows 10/11, Linux (Ubuntu 20.04+), or macOS
Python	Version 3.9 or higher
Django Framework	Version 4.0 or higher
Web Browser	Chrome, Firefox, or Edge
MongoDB	Version 6.0 or higher

3.3 Methodology

The methodology adopted for developing the eGuru Platform is based on a modular, role-driven approach to deliver a functional, interactive, and intelligent e-learning web application. The platform was designed to support two primary user roles i.e., Admin (Teachers) and User (Students), each with access to different features. The development began with a thorough requirement analysis to identify essential functionalities such as user authentication, training management, file upload capability, and an intelligent chatbot assistant powered by a local language model. Django was selected as the backend framework for its robustness in handling

authentication, routing, and database operations, while HTML, CSS, and JavaScript were used for the frontend to create an interactive and responsive user interface.

User authentication and role-based access control were implemented using Django's built-in system, allowing users to log in securely and access features based on their role. Admin users were given the ability to manage training sessions by creating, editing, or deleting training content, while regular users could browse available trainings, view detailed information, and enroll in selected courses. A key feature of the platform is the document upload module, where users can upload educational content in PDF, PPT, or Word formats. These documents are parsed to extract raw textual content and metadata, which is then segmented into logical chunks for further processing.

Once the content is extracted, it is transformed into embeddings using a locally running instance of the LLaMA 3.2 language model. These embeddings are stored in a MongoDB collection named `eguru-llm-index`, which facilitates fast and efficient semantic search during chatbot interactions. When a user submits a query through the chatbot interface, the system performs a similarity-based search on the indexed content to retrieve the most relevant chunks. These retrieved segments serve as contextual references for generating an accurate, document-aware response from the LLM. The chatbot also provides reference information such as the file name and page number to help users trace the source of the response.

The entire interaction—from login and document upload to chatbot query resolution—was tested using both unit and integration testing to ensure each module functioned correctly and harmoniously. The performance of the platform, particularly the responsiveness of the chatbot and the accuracy of content retrieval, was evaluated under local deployment conditions. The system runs entirely on local infrastructure, with Django handling server-side operations and MongoDB managing data persistence. All language model operations are performed offline, ensuring complete user data privacy and security. This modular and secure architecture allows eGuru to serve as a robust foundation for document-based learning and AI-assisted educational support.

Chapter 4: System Design and Architecture

The architecture of the eGuru Platform follows a layered design approach, structured to ensure modularity, maintainability, and scalability. It is composed of three primary layers: the Presentation Layer, the Application Layer, and the Data Layer, each responsible for distinct functionalities within the system. These layers work together seamlessly to offer complete e-learning experience with role-based access, training content management, document processing, and a locally integrated intelligent chatbot assistant.

The Presentation Layer serves as the front-facing component of the platform, offering interactive and role-specific interfaces for Admin and User accounts. Admin users have access to a dashboard that allows them to manage training programs, view analytics, and oversee user interactions. Users, on the other hand, can browse available training modules, enroll in courses, and interact with the chatbot assistant. The interface also includes sections for uploading documents, viewing uploaded files, and accessing the chatbot in a dedicated or floating chat window. This layer ensures a smooth and user-friendly experience, keeping the interface clean and intuitive.

The Application Layer forms the logical core of the system. It handles user authentication and enforces role-based access controls to secure different areas of the platform. This layer also manages training workflows such as creation, editing, enrollment, and deletion of training sessions. The document processing engine resides here, responsible for parsing uploaded PDF, PPT, and Word documents to extract readable content. Once extracted, the text is broken down into meaningful chunks that are later used for semantic search. This layer also manages query handling and communication with the embedded language model, ensuring the chatbot responds with accurate, context-aware information.

The Data Layer underpins the platform's storage and retrieval capabilities. All user data, training metadata, uploaded document content, and generated embeddings are stored here. After document content is extracted and chunked, it is transformed into vector embeddings using a local embedding model and stored in an indexed format. These indexed embeddings are used for semantic similarity

search whenever the user interacts with the chatbot. This design ensures that responses are relevant and grounded in the actual content provided by users, maintaining transparency and traceability in the interaction. The structure of the database supports quick lookup operations and allows scalable storage of multiple files and interactions per user.

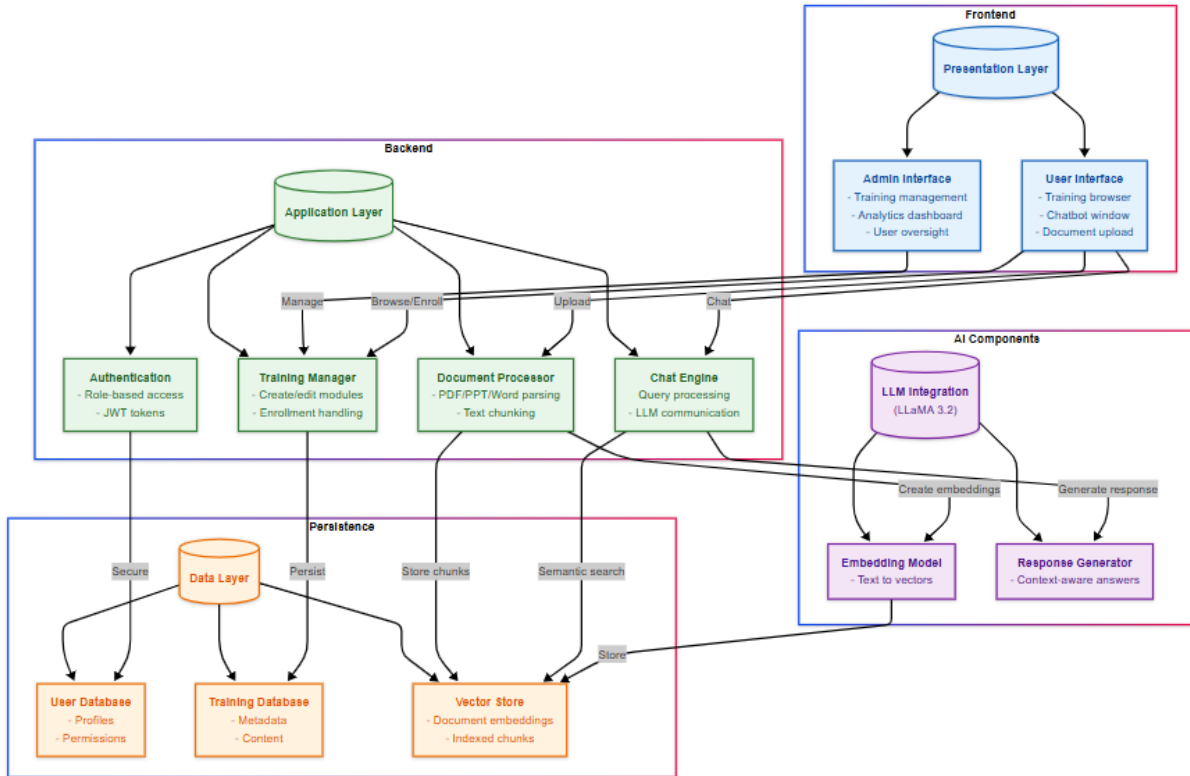


Figure 4.1: System Architecture

An important aspect of architecture is the LLM Integration, where a large language model (LLaMA 3.2) is used both for generating embeddings of the document content and for producing responses to user queries. The system performs a similar search between the user's question and the indexed content, selecting the top-matching chunks to construct a response. This architecture enables a private and fully offline chatbot experience, ensuring data privacy while still delivering powerful AI-driven assistance.

Overall, the system's architecture supports a robust, extensible platform that accommodates multiple document types, manages role-based user access, and integrates a local AI model for

knowledge-aware conversations. Each layer communicates efficiently with the others, enabling a cohesive and secure flow of data and control throughout the platform.

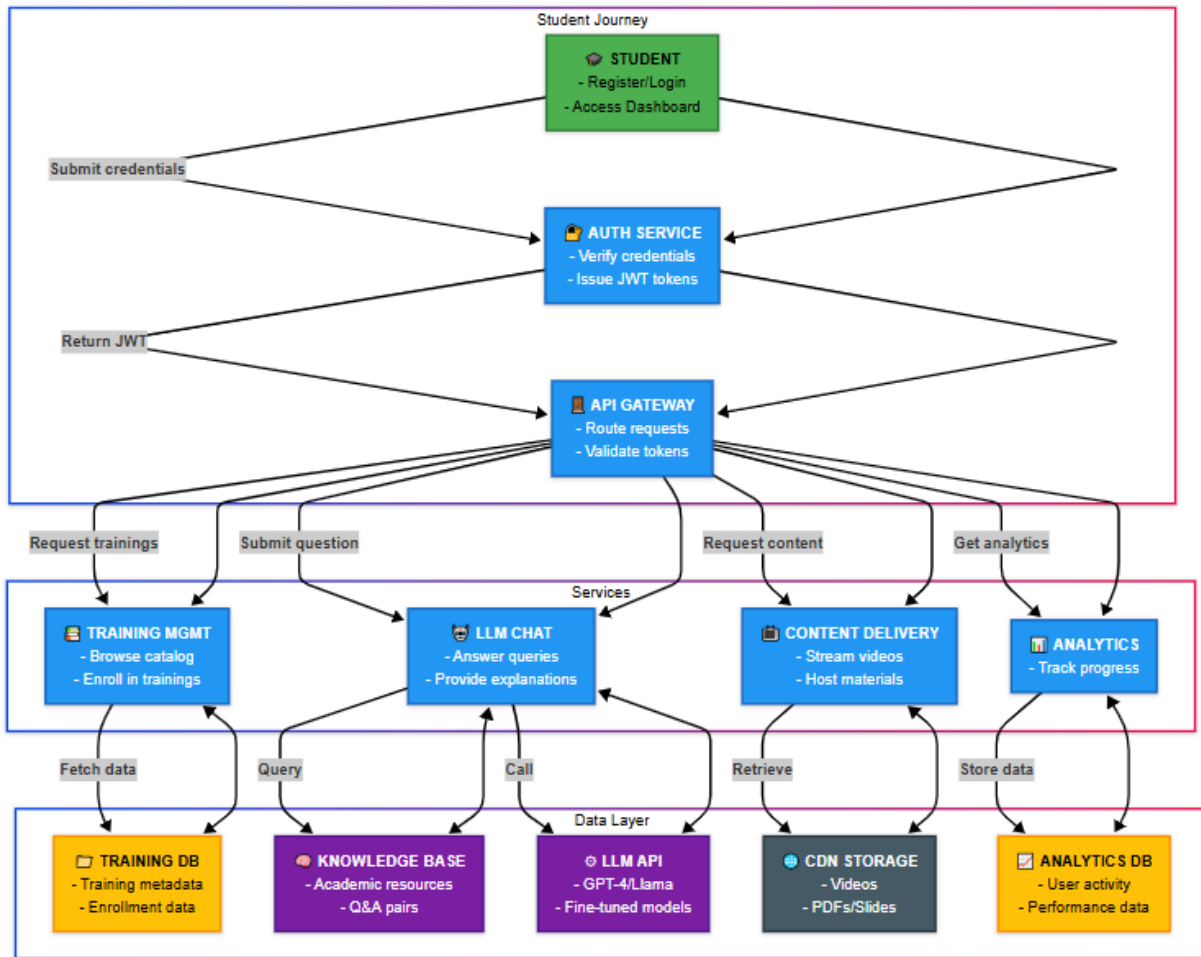


Figure 4.2: User / Student Journey

The student journey begins with authentication, where users register/login through the Auth Service to receive a JWT token. The API Gateway then routes all requests, allowing students to browse training, ask academic queries to the LLM Chat, and access learning materials via Content Delivery. Each interaction connects to dedicated backend services - Course Service manages catalog data, LLM Chat processes questions using the Knowledge Base, while Analytics tracks progress. Database systems securely store course information, LLM training data, and user activity logs. This end-to-end design ensures seamless navigation through all platform functionalities while maintaining security and performance.

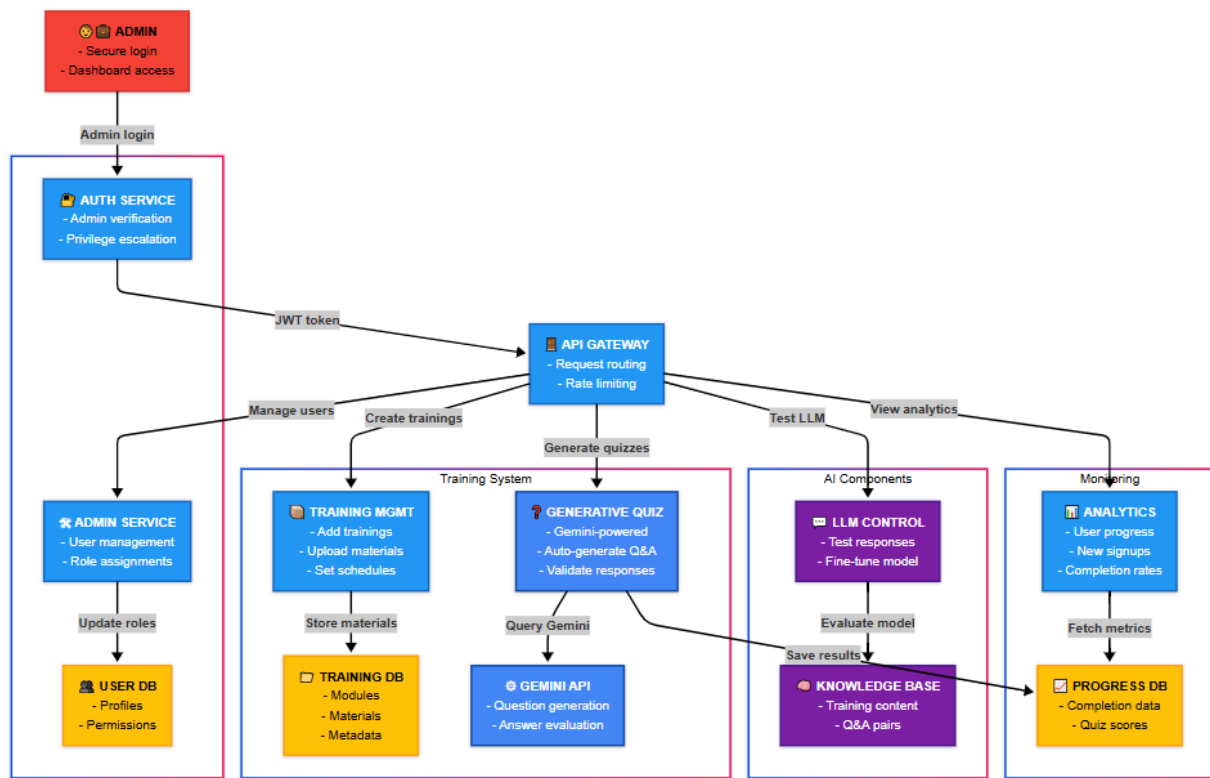


Figure 4.3: Admin Journey

The admin flowchart outlines a privileged control system where administrators first authenticate via a secure Auth Service to receive JWT tokens. The API Gateway routes their requests to core services: User Management, Training Management (add materials/schedules), and Generative Quiz Engine (Gemini-powered assessments). Admins can test/fine-tune the LLM through a dedicated control panel and monitor user progress (completion rates, new signups) via Analytics. All training content is stored in a Training DB, while quiz results sync with a Progress DB. The Gemini API auto-generates questions, and the knowledge base stores LLM training data. Each action flows through centralized services with role-based access control, ensuring secure administration of the learning platform.

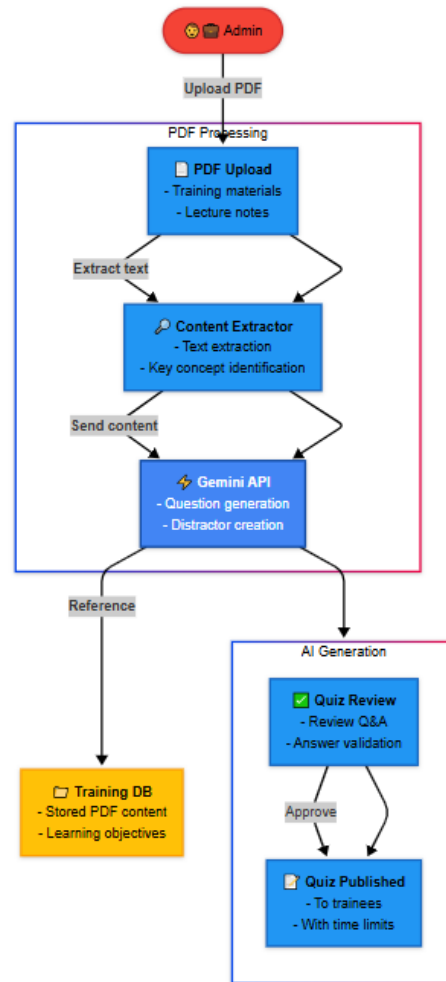


Figure 4.4: Quiz Generation Flowchart

The eGuru LLM system enables trainees to submit academic queries, which are processed through a context-aware AI model. The system retrieves information from a knowledge base containing training materials. Responses undergo validation for accuracy and source citation before delivering step-by-step answers. This closed-loop workflow ensures reliable, reference-backed explanations while maintaining safety filters. The LLM specifically references uploaded training content (PDFs, lectures) to provide tailored responses.

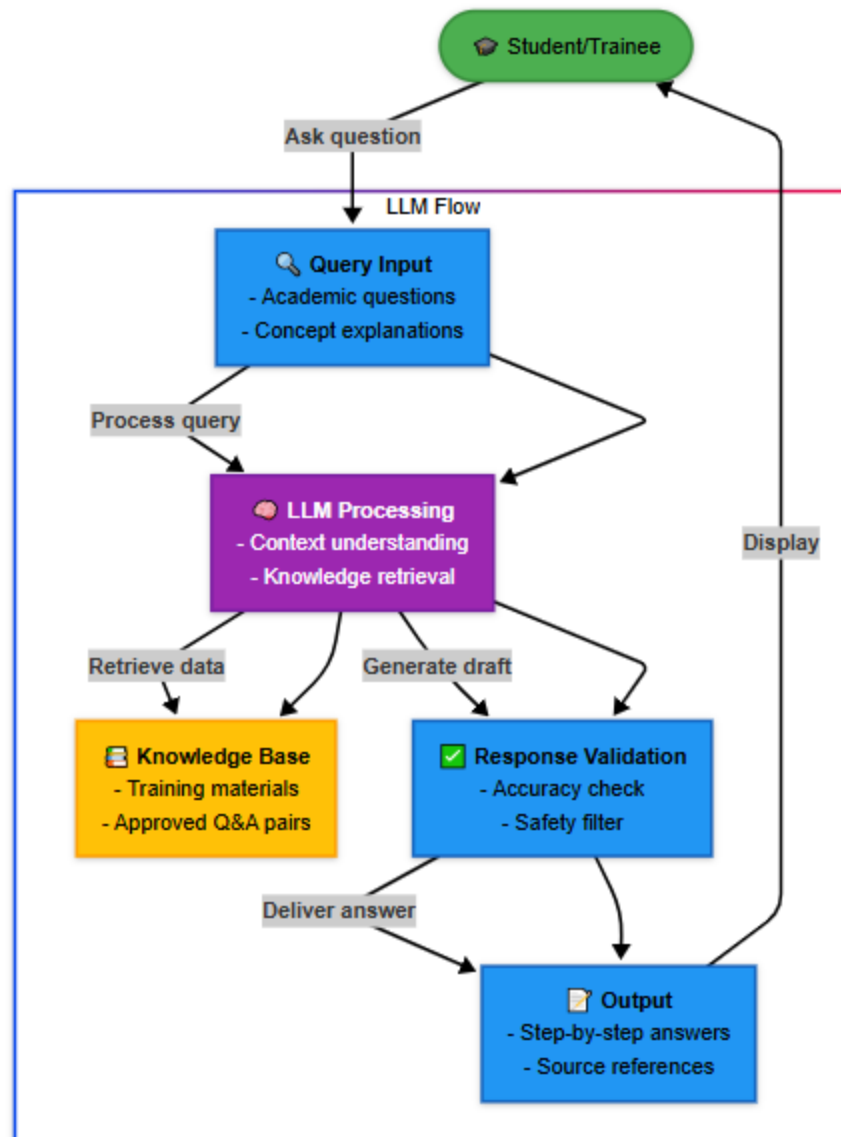


Figure 4.5: LLM Flowchart

Admins upload training PDFs, from which text and key concepts are extracted automatically. The Gemini API processes this content to generate quiz questions and distractors, cross-referencing the training database for alignment with learning objectives. Quizzes undergo manual review for difficulty adjustment and answer validation before being published to trainees. This AI-assisted workflow transforms static materials into interactive assessments efficiently.

Chapter 5: System Implementation

5.1 Implementation Methodology

The implementation of the eGuru Learning Platform followed a modular and agile development methodology. The platform was divided into several core components to streamline the development and testing process: user authentication, role-based dashboards, course management, AI-powered chatbot integration, document handling, and analytics visualization. Django was chosen as the backend framework due to its robust support for rapid development and scalability. The frontend was developed using HTML, CSS, and JavaScript to provide an intuitive and responsive user experience. MongoDB was used as the primary database to store user data, training materials, and AI-indexed content. APIs were developed for seamless communication between frontend components, the database, and external AI services. Agile sprints and regular code reviews were used to ensure each module was fully functional and testable before integration.

5.2 System Features

5.2.1 Role-Based Dashboards

The eGuru platform supports two primary user roles—Admin and User. Each role is provided with a customized dashboard to access only the functionalities relevant to them. Admins have access to the overall system management, including managing training sessions, user enrollment, analytics, and chatbot document indexing. Users, on the other hand, are provided with a simplified dashboard that allows them to browse available courses, enroll in sessions, and interact with the AI-powered eGuru chatbot for study assistance. Role-specific sidebars and interfaces ensure a clean and focused user experience.

5.2.2 Document Upload and Indexing

A core feature of the platform is the ability to upload study materials, including PDFs, PowerPoint presentations, and Word documents. Once uploaded, these documents are processed to extract text, tables, images, and even embedded text using OCR (Optical Character Recognition) techniques. The extracted data is then stored in a structured format within MongoDB. This indexed data forms the knowledge base for the AI chatbot, enabling accurate and context-aware responses to user queries.

5.2.3 AI-Powered eGuru Chatbot

The eGuru chatbot is an intelligent learning assistant integrated into the platform. It is designed to assist users by answering questions based on the uploaded study materials. The chatbot leverages advanced large language models (LLMs) accessed through secure API keys from global AI providers such as OpenAI or Anthropic. When a user asks a question, the chatbot fetches relevant content from the indexed data, generates a meaningful response, and includes references to the source document and page number. This ensures transparency, accuracy, and contextual relevance in the responses.

5.2.4 Interactive Monitoring and Analytics Dashboard

The admin dashboard includes interactive visualizations and analytics tools that provide insights into user activity, course enrollments, document uploads, and chatbot engagement statistics. These analytics help in identifying popular courses, improving learning materials, and monitoring platform usage patterns for better decision-making.

5.3 Algorithms and Techniques

5.3.1 AI Embedding and Search Algorithms

The system uses embedding techniques to convert textual content into vector representations for efficient similarity search. Upon document upload, the extracted content is passed through an embedding model provided by Ollama, and the resulting vectors are stored in MongoDB. During chatbot interactions, user queries are converted into embeddings and matched with the stored document vectors to retrieve the most relevant content.

5.3.2 Data Optimization Techniques

To ensure fast response times and efficient storage, data optimization techniques such as content chunking, stop-word removal, and noise reduction are applied. OCR results undergo pre-processing to filter out low-confidence extractions, and tables are stored in structured formats to maintain their integrity during indexing and retrieval.

5.3.3 Model Invocation and API Handling

The system interacts with global AI models via API keys. Each user query is routed through an asynchronous API handler that ensures secure, authenticated, and rate-limit-respecting calls to the model provider. The response is parsed, post-processed, and displayed to the user with contextual references.

5.3.4 Serialization and Caching

To minimize repeated API calls and reduce latency, responses to frequently asked questions are serialized and cached. This also enables partial offline capabilities for recently accessed data and enhances the scalability of the platform by reducing the load on external APIs.

Chapter 6: Testing and Validation

6.1 Testing Strategies

The eGuru Learning Platform underwent a structured and methodical testing process to ensure reliability, performance, and usability across all components. A multi-tiered testing approach was adopted, including unit testing, integration testing, and system testing. In the unit testing phase, individual functions and modules such as document uploads, AI indexing, chatbot processing, and user authentication were tested in isolation to verify expected behavior. This allowed for early detection of bugs and ensured that each feature functioned correctly before integration.

Integration testing focused on validating the interaction between modules. For example, the successful extraction of content from documents was checked alongside the embedding generation and AI model communication. Similarly, integration between user interfaces and the backend API was verified to ensure smooth transitions from uploading documents to asking questions through the chatbot.

System testing encompassed end-to-end evaluations of the platform. This included user flows such as logging in, uploading learning materials, enrolling in courses, and querying the chatbot. Tests were also conducted across multiple browsers and devices to ensure responsiveness and accessibility. Throughout all phases, test environments were designed to simulate real-world usage scenarios, ensuring that the platform could handle diverse input formats and concurrent user interactions without performance degradation.

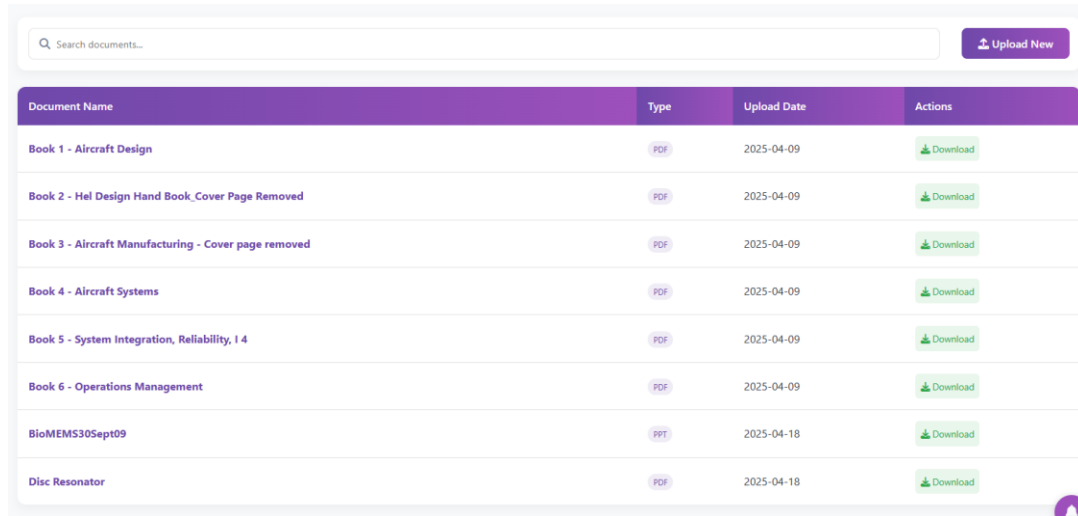
6.2 Test Cases and Results

6.2.1 Test Case 1: PDF and Document Upload with Content Extraction

Objective: To verify whether the system correctly handles uploads of documents in various formats and extracts relevant content for AI indexing.

Steps:

- Upload multiple documents including PDF, PPT, and DOCX files.
- Check if the system successfully parses text, tables, and images.
- Validate extracted content stored in MongoDB.



Document Name	Type	Upload Date	Actions
Book 1 - Aircraft Design	PDF	2025-04-09	Download
Book 2 - Hel Design Hand Book_Cover Page Removed	PDF	2025-04-09	Download
Book 3 - Aircraft Manufacturing - Cover page removed	PDF	2025-04-09	Download
Book 4 - Aircraft Systems	PDF	2025-04-09	Download
Book 5 - System Integration, Reliability, I 4	PDF	2025-04-09	Download
Book 6 - Operations Management	PDF	2025-04-09	Download
BioMEMS30Sept09	PPT	2025-04-18	Download
Disc Resonator	PDF	2025-04-18	Download

Figure 6.1: Uploaded PDF's and PPT's

The first test case evaluated the system's ability to handle document uploads and extract meaningful content. Various formats such as PDF, Word documents, and PowerPoint presentations were uploaded. The system successfully parsed text, tables, and images from these files. It also performed OCR to extract text from scanned documents and embedded images. The extracted data was then stored in the backend database and prepared for AI-based indexing, demonstrating the robustness of the content ingestion pipeline.

6.2.2 Test Case 2: AI Chatbot Response from Indexed Content

Objective: To test if the chatbot returns contextually accurate responses using AI embeddings derived from uploaded content.

Steps:

- Ask questions related to uploaded documents.
- Check if responses are relevant, complete, and cite the document and page number.
- Assess if AI model returns fallback response for unanswerable queries.

The second test case examined the performance of the AI chatbot when responding to user queries based on uploaded content. After uploading training materials, users interacted with the chatbot by asking questions directly related to the content. The chatbot consistently returned relevant and well-structured answers, referencing the correct documents and sections. In cases where the

information was unavailable or unclear, the system provided appropriate fallback responses, maintaining a coherent interaction flow.

6.2.3 Test Case 3: Training Enrollment and Dashboard Navigation

Objective: To validate the functionality of user-side actions such as enrolling in a training, viewing details, and navigating the dashboard.

Steps:

- Log in as a user and explore the “All Trainings” section.
- Click on “Enroll” for a course and navigate to “Enrolled Trainings.”
- View training details and access any AI-powered support via chatbot.

The third test case assessed the course enrollment process and dashboard navigation from a learner’s perspective. Users were able to browse available training modules, enroll in courses, and view detailed information through a clean and intuitive interface. Additionally, the chatbot remained accessible throughout the platform, allowing learners to ask content-related questions at any point. Navigation between sections was smooth, and all actions performed by the user were reflected accurately in the system.

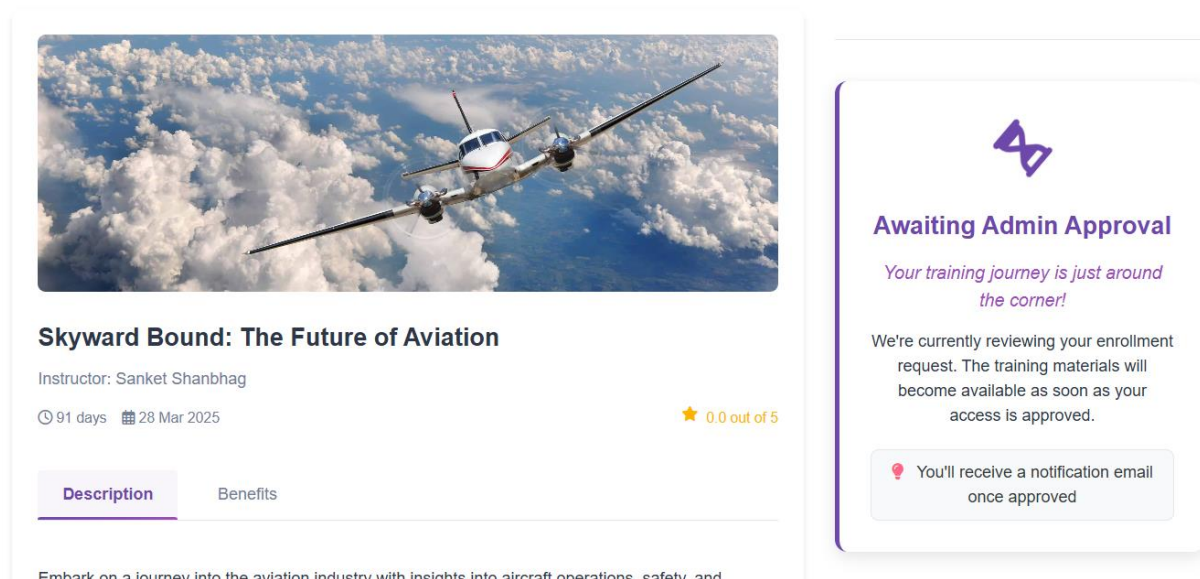


Figure 6.3: Training Enrollment waiting for the Admin approval

6.2.4 Test Case 4: Quiz Generation from Uploaded PDF Content

Objective: To verify the system's ability to generate relevant and accurate quiz questions from uploaded PDF training materials.

Steps:

- Upload PDF files containing training or instructional content.
- Initiate the quiz generation process from the UI or backend.
- Evaluate the generated multiple-choice questions (MCQs) for relevance, clarity, and content coverage.

Review Generated Questions

1. What percentage of Earth's atmosphere is composed of nitrogen?

☐ 21%

☒ 78%

☐ 50%

☐ 1%

[Edit](#) [Delete](#)

2. What is the approximate standard lapse rate in degrees Celsius per 1000 meters?

☐ 1.5°C

☐ 3.6°C

☒ 6.5°C

☐ 10°C

[Edit](#) [Delete](#)

Figure 6.4: Review Generated Quiz Questions

This test case validated the system's quiz generation feature. After uploading training documents in PDF format, the system successfully generated contextually appropriate MCQs. The questions covered key concepts, included correct answer choices, and maintained logical formatting. The output was consistent with the themes and depth of the original content, demonstrating the effective use of AI for content-based assessment creation.

6.2.5 Test Case 5: AI-Powered Feedback Summary Generation from Training Feedback

Objective: To test the system's capability to generate concise and meaningful AI summaries from multiple user-submitted feedback.

Steps:

- Submit multiple user feedback for a training session.
- Trigger the AI feedback summarization process.
- Evaluate the generated summary for tone, coherence, and representation of key feedback points.

This test case examined the system's feedback generation module, which processes textual feedback submitted by users for specific training sessions. Upon receiving multiple feedbacks, the AI-generated summary reflected the overall sentiment, highlighted frequently mentioned comments, and presented them in a readable format. This helped training organizers get a quick overview of collective feedback without needing to read each feedback individually.

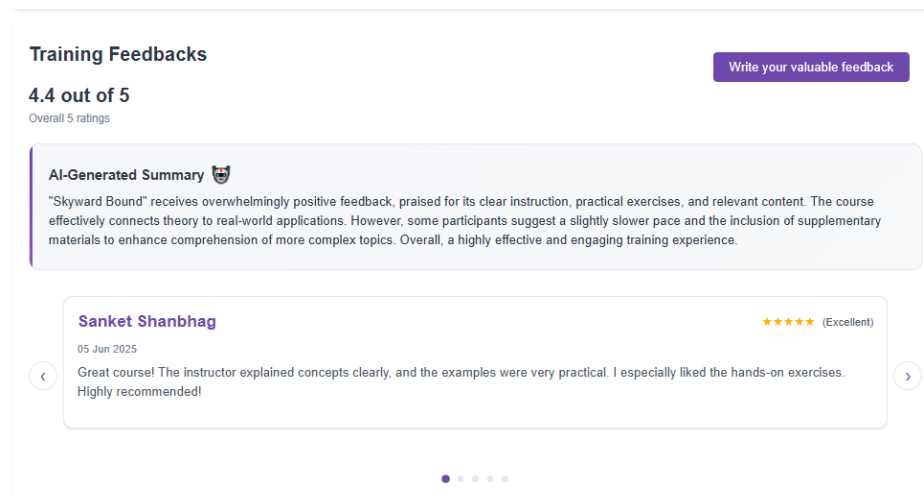


Figure 6.5: AI-Powered Summary based of Feedbacks

These test cases collectively validated the core functionalities of the eGuru platform, confirming that the system operates reliably under various conditions and provides a user-friendly experience across its major components.

6.3 Validation and Verification

Validation and verification were carried out to ensure that the eGuru platform met both its technical and functional requirements. Validation focused on aligning the system's behavior with real-world use cases by involving multiple testers, including end users and stakeholders, who confirmed that the platform's outputs matched expected outcomes. For example, chatbot responses were reviewed for clarity, relevance, and correctness in the context of the learning materials provided.

Verification was achieved through comprehensive code reviews, functionality checks, and system performance evaluations. Automated tests were used to confirm consistent behavior under repeated scenarios, while manual testing helped uncover edge cases related to user interaction. Role-based access control was verified to ensure that users with different permissions experienced the platform appropriately. Security mechanisms such as API key management, encrypted communication, and user session handling were also reviewed to maintain data integrity and privacy.

This rigorous validation and verification process affirmed that the platform is reliable, secure, and scalable for deployment in educational institutions and training environments.

Chapter 7: Result and Discussion

7.1 Output of the Project

The eGuru Learning Platform successfully achieved its goal of creating an interactive, AI-assisted environment for enhanced learning and document-based knowledge retrieval. Upon deployment, the system allowed users to upload various educational documents, including PDFs, Word files, and PowerPoint presentations. The uploaded content was accurately extracted and semantically indexed using advanced AI embedding models connected through secure APIs. This enabled the chatbot to access, interpret, and respond to learner queries based on the exact content of those documents.

One of the major outputs of the platform was the ability to engage users through a responsive chatbot interface embedded in the dashboard. Learners could ask questions at any time and receive informative, context-aware answers sourced directly from the uploaded learning materials. This transformed the static nature of documents into an interactive learning experience. Additionally, the system highlighted references such as the file name and section or page number from which the answer was derived, adding a layer of transparency and credibility to the chatbot responses.

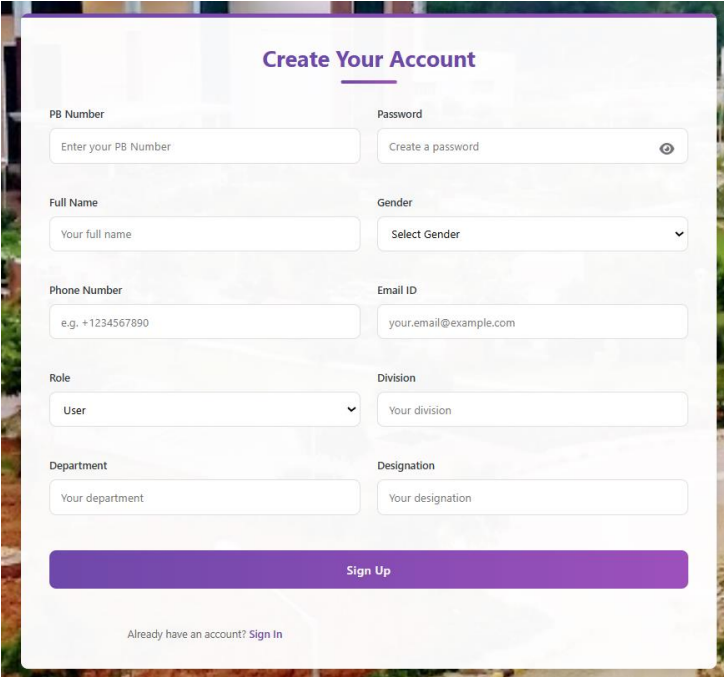


Figure 7.1: Sign Up Page

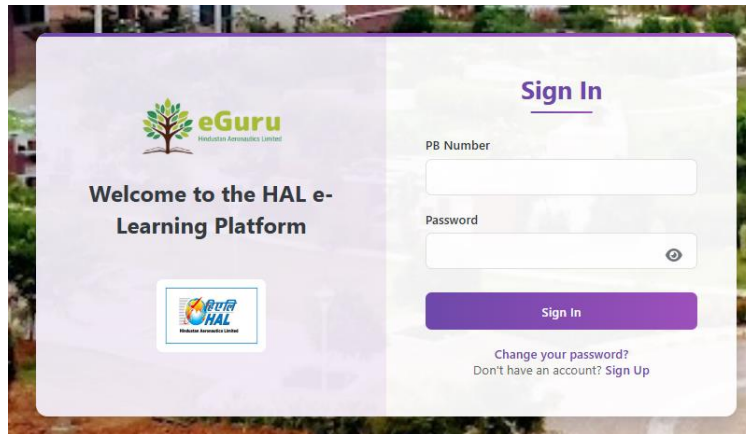


Figure 7.2: Sign In Page

Figure 7.3: Creating new users

The admin and user dashboards were also implemented with intuitive navigation and clearly segmented features. Admins were able to manage users, trainings, and documents, while users could easily browse available trainings, enroll in courses, and interact with the AI assistant. The dynamic updates within the dashboard ensured that user activities such as course enrollments and document uploads were reflected in real time, making the experience both efficient and seamless.

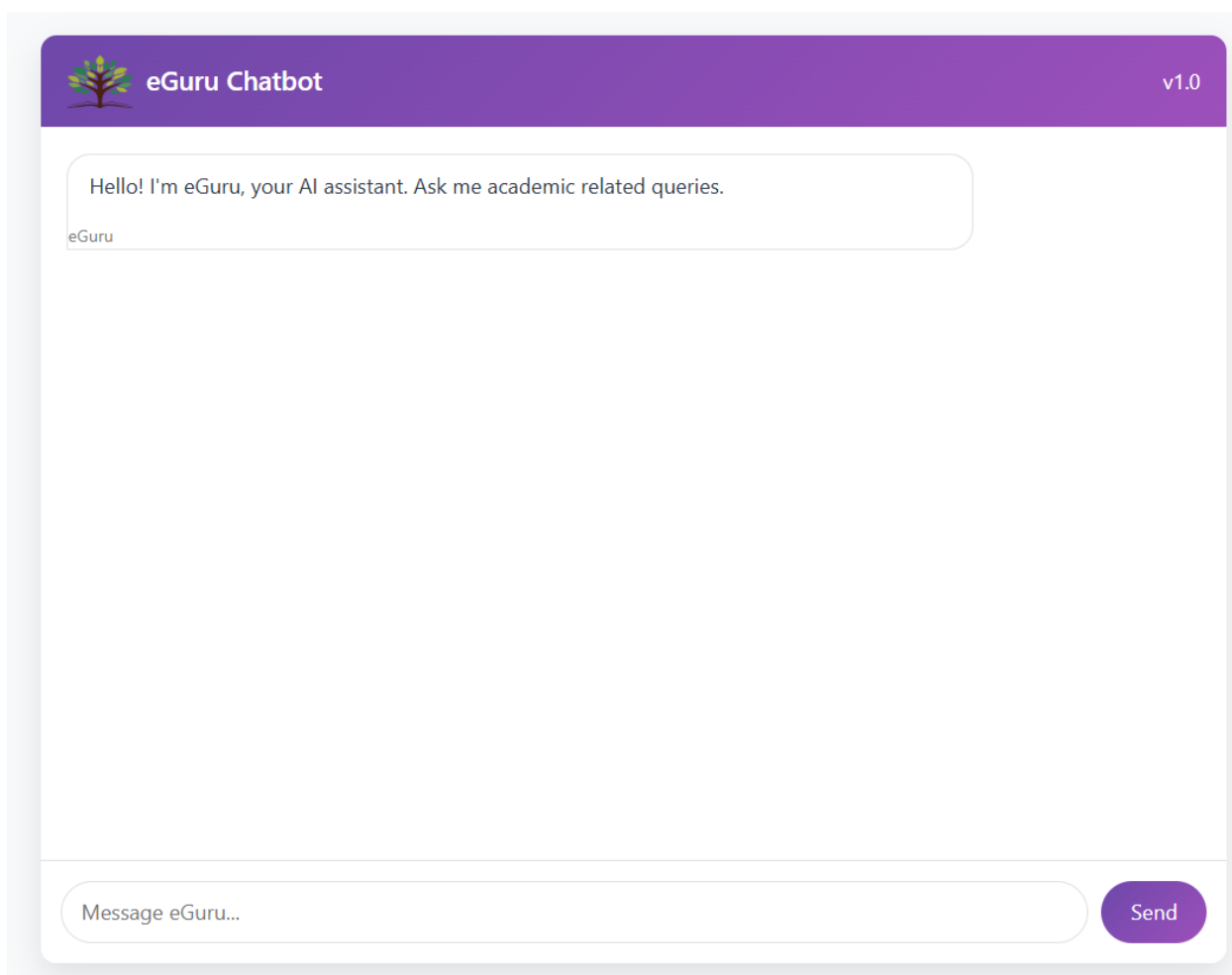


Figure 7.4: eGuru AI assistant

7.2 Performance Analysis

The overall performance of the eGuru platform was found to be stable, responsive, and scalable during the testing phase. The AI chatbot consistently responded within acceptable latency, and its ability to handle large documents and complex queries without crashing or misinterpreting input demonstrated its reliability. Document parsing and content indexing occurred promptly after file upload, with the system maintaining smooth processing even during batch uploads or simultaneous user interactions.

Load testing revealed that the system could comfortably support multiple concurrent users, with no degradation in response times or user experience. The performance of the content extraction module was particularly noteworthy, handling diverse document layouts and formats effectively.

The backend infrastructure efficiently managed the extraction, embedding, and storage workflows without memory leaks or server overloads. This indicated that the platform could support future scaling needs, such as larger datasets or increased user traffic.

The user interface was optimized for responsiveness, adapting well across desktop and mobile devices. All major features, from chatbot access to training enrollment, were functional across different screen sizes and operating systems. This cross-platform compatibility enhanced the usability of the platform in both academic and professional training environments.

7.3 Comparisons with Existing Solutions

When compared with traditional learning management systems and document readers, eGuru stands out through its integration of AI-based document comprehension and interactive support. While most existing platforms offer basic resource access, course tracking, and video lectures, eGuru enhances the learning experience by turning passive reading material into an active knowledge source. The chatbot, powered by advanced language models, acts as an on-demand tutor, guiding learners through complex topics in real time.

Moreover, unlike many static document repositories, eGuru ensures that all learning content is searchable, conversational, and responsive to learner needs. This reduces the time spent manually searching for answers and increases engagement with course material. The platform's ability to understand natural language queries and link them back to document sources gives it a significant advantage over conventional keyword-based search systems.

Additionally, eGuru's user interface is designed to be intuitive and accessible, which is not always the case with existing systems that often require technical onboarding. By minimizing the learning curve and maximizing interactivity, eGuru not only matches but in many cases exceeds the capabilities of comparable educational platforms in terms of innovation, functionality, and user-centric design.

Chapter 8: Conclusion and Future Enhancements

8.1 Summary of Achievements

The development of the eGuru Learning Platform marks a significant step forward in merging traditional educational resources with modern AI-powered interactivity. The system successfully enables the uploading, parsing, and semantic indexing of diverse educational materials such as PDFs, Word documents, and PowerPoint presentations. These documents are not only stored but are also made accessible through a smart AI chatbot, transforming static content into dynamic, conversational knowledge.

Key features such as real-time chatbot interaction, OCR-based text extraction, centralized dashboards for both admin and user roles, and seamless course enrollment functionalities were effectively implemented. These capabilities come together to create an immersive learning environment that supports curiosity, facilitates self-paced education, and ensures that users can derive value from their content immediately. The use of external AI models through secured API access further broadens the system's linguistic and contextual understanding, making it suitable for a wide variety of academic and training contexts.

Through rigorous testing and user-centric design, eGuru ensures both technical robustness and high usability. The platform performs well under real-world conditions, demonstrating stability, fast response times, and accurate content referencing—all crucial traits for educational platforms aimed at scale.

8.2 Limitations of the Project

Despite the platform's success, there are certain limitations to acknowledge. The current system depends on the availability and stability of external AI APIs for its chatbot functionality. Any downtime or restrictions in those services could temporarily limit chatbot performance. Furthermore, while document extraction handles text, tables, and embedded images effectively, complex formatting or handwritten notes can sometimes challenge the accuracy of the OCR module.

The platform's current focus is also primarily on document-based learning. While it serves this domain well, it does not yet include support for video lectures, quizzes, or real-time collaborative tools. Additionally, while role-based dashboards are implemented, there is room to expand administrative controls, analytics, and user performance tracking features.

Scalability for very large datasets or integration with institutional learning management systems (LMS) like Moodle or Canvas remains another area that requires further development and optimization to meet enterprise-level requirements.

8.3 Future Enhancements

The eGuru platform continues to evolve beyond its core document-based learning capabilities. One significant enhancement that has already been implemented is the automatic quiz generation module. This feature extracts content from uploaded materials and intelligently creates multiple-choice questions, enabling learners to assess their understanding in a self-paced and contextual manner. This advancement transforms the learning experience by not only offering instant answers through a chatbot but also reinforcing knowledge through assessments.

Further developments will include more advanced analytics to track user engagement and learning outcomes. These analytics will support educators and administrators in tailoring content and improving overall learning effectiveness. Mobile optimization and the launch of a dedicated mobile application will enhance accessibility and allow learners to interact with the platform seamlessly from any device.

To promote collaborative learning, features such as discussion forums, real-time chat among enrolled students, and shared annotations on documents are also under consideration. These future directions will ensure that eGuru not only remains current with educational trends but also leads in offering a holistic, AI-driven learning ecosystem.

References

1. Brown, T. et al. “Language Models are Few-Shot Learners.” *Advances in Neural Information Processing Systems*, 2020.
2. Vaswani, A. et al. “Attention Is All You Need.” *NeurIPS*, 2017.
3. Devlin, J. et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *NAACL*, 2019.
4. Rajpurkar, P. et al. “SQuAD: 100,000+ Questions for Machine Comprehension of Text.” *EMNLP*, 2016.
5. OpenAI API Documentation: <https://platform.openai.com/docs>
6. Google Cloud Vision API Documentation: <https://cloud.google.com/vision/docs>
7. MongoDB Documentation: <https://www.mongodb.com/docs/>
8. PyMuPDF (fitz) for PDF processing: <https://pymupdf.readthedocs.io/>
9. Tesseract OCR: <https://github.com/tesseract-ocr/tesseract>
10. LangChain Framework: <https://docs.langchain.com/>
11. Scikit-learn Documentation: <https://scikit-learn.org/stable/documentation.html>
12. Hugging Face Transformers: <https://huggingface.co/docs/transformers>
13. Tailwind CSS Documentation: <https://tailwindcss.com/docs>