CONVERT ENTERPRISE PDFS INTO SEARCHABLE KNOWLEDGE

<u>Internship Report</u>

Submitted by

Name of Student(s)	**University Roll No.**
Ismail Sk	**11601022070**
Supratim Nag	**11601022064**
Sanchari Biswas	**11601022049**

Conducted at

Intel Unnati Industrial Training 2025

Department of Computer Science & Engineering,
MCKV Institute of Engineering

Acknowledgement

We would like to express our sincere gratitude to the Intel® Unnati Industrial Training Program for providing us with a valuable opportunity to work on an industry-oriented project and gain practical exposure to real-world applications of artificial intelligence. This program enabled us to bridge the gap between theoretical knowledge and practical implementation by allowing us to explore modern AI tools, frameworks, and development practices in a structured and guided environment.

The training sessions, learning resources, and hands-on project experience offered through the Intel® Unnati program played a crucial role in strengthening our technical foundation and enhancing our problem-solving abilities. The exposure to industry-level workflows, best practices, and emerging technologies significantly contributed to our overall learning and professional growth.

We would also like to extend our heartfelt appreciation to the industry mentor(s) associated with the Intel® Unnati Industrial Training Program for their continuous guidance, constructive feedback, and technical support throughout the duration of the project. Their valuable insights, encouragement, and real-world perspectives helped us overcome challenges, refine our approach, and successfully complete this project.

Overall, the Intel® Unnati Industrial Training Program has been an enriching and rewarding experience, and we are truly thankful for the opportunity to learn, innovate, and grow under this initiative.

TABLE OF CONTENTS

LIST OF TABLES & FIGURES

Abstract

Enterprise organizations store vast volumes of information in PDF formats such as manuals, policies, reports, technical documentation, and scanned records. However, retrieving relevant information from these unstructured documents is time-consuming and inefficient due to the lack of semantic search capabilities. This project aims to develop a system that converts unstructured and scanned PDF documents into structured and searchable knowledge.

The proposed solution integrates Optical Character Recognition (OCR), intelligent document segmentation, table extraction, and vector-based semantic search. Text is divided into meaningful sections without disrupting chapter boundaries, tables are extracted and stored in a NoSQL database for precise queries, and the processed text is embedded into a vector database to enable context-aware information retrieval. Additionally, images and charts are translated into textual descriptions to make multimedia content searchable.

The project is being developed as part of the Intel® Unnati Industrial Training Program, and is currently in progress. Upon completion, the system is expected to deliver an end-to-end pipeline that transforms enterprise PDFs into a searchable knowledge base along with a search interface for efficient retrieval. This approach aims to reduce manual document exploration time, improve organizational knowledge accessibility, and provide a scalable foundation for enterprise-grade information management.

1. Introduction

In modern enterprises, critical information is distributed across thousands of PDF documents, including technical manuals, policy files, research papers, compliance documents, audit reports, and scanned letters. Although these documents hold valuable institutional knowledge, most of them remain unstructured, inconsistently formatted, and difficult to search. Traditional keyword-based search systems often fail to retrieve accurate information because they cannot understand document context, layout structures, tables, figures, or scanned text. As a result, employees spend considerable time manually searching through large PDF repositories, leading to reduced productivity and inefficient knowledge management.

With the rapid advancement of Generative AI and Retrieval-Augmented Generation (RAG) systems, organizations now have the opportunity to transition from basic keyword search to semantic, meaning-based document retrieval. This project addresses the need for an intelligent system that can transform unstructured and scanned PDFs into structured, searchable knowledge. The objective is to design a robust pipeline capable of accurately extracting text, tables, images, and charts from PDF documents and organizing them in a format optimized for efficient retrieval.

By integrating OCR for scanned content, meaningful text chunking, table extraction into a NoSQL database, and vector-based semantic search for contextual accuracy, the proposed system aims to significantly reduce information lookup time and enhance enterprise-wide knowledge accessibility. Ultimately, the project seeks to enable instant information discovery, remove bottlenecks associated with manual document search, and establish a strong foundation for future AI-powered enterprise knowledge assistants.

Table 1.1 Software and Tools used

Package / Tool	**Purpose**	**License**	**Link**
langchain	RAG pipeline	MIT	langchain.ai
langchain-google-genai	Gemini LLM + embeddings	MIT	google-genai
pinecone-client	Vector DB	Apache-2.0	pinecone
sentence-transformers	Embeddings	Apache-2.0	sbert
pymupdf	PDF extraction	AGPL-3.0	pymupdf
pytesseract	OCR engine	Apache-2.0	tesseract
fastapi	Backend API	MIT	fastapi
uvicorn	API server	BSD	uvicorn

2. Training works undertaken

The industrial training consisted of hands-on modules focused on modern AI development, software engineering workflows, and enterprise-grade application building. The work undertaken during the training can be summarized as follows:

A. Development Environment Setup and AI Text Generation

The first phase focused on setting up an industry-standard development environment for Linux-based AI development within Windows using Windows Subsystem for Linux (WSL). Visual Studio Code and Git were configured to enable seamless coding and version control. The UV package manager was utilized to create isolated Python environments and manage dependencies efficiently.

A text-generation demo was implemented using the Groq API, involving virtual environment creation, dependency installation, API key setup, and execution of inference scripts to generate natural-language outputs.

B. Retrieval-Augmented Generation (RAG) Workflow

The second phase concentrated on building a Retrieval-Augmented Generation workflow for document-aware AI systems. This included generating embeddings using Hugging Face models, storing them locally, and using them for semantic retrieval during question-answering tasks. A benchmarking pipeline was developed using the Llama 3.2 1B Instruct model to perform contextual generation based on retrieved document chunks.

C. Model Access, Tokenization, and Large-Model Handling

Training also covered mechanisms for accessing gated and non-gated models on the Hugging Face Hub. API tokens were configured for authenticated model downloads. Emphasis was placed on understanding model caching, dependency isolation for each RAG component, and performance considerations when working with large models.

D. Front-End AI-Powered Application Development (Vibe Coding)

The training introduced Vercel’s v0.dev platform for AI-assisted full-stack web application generation. A multi-page professional React/Next.js and Tailwind CSS application was generated using natural-language prompts. The project was downloaded and configured within the WSL environment, dependencies were installed using Node.js/npm, and the application was executed locally using the Vercel development server.

E. Software Development Practices

Throughout the training, strong emphasis was placed on:

- Version control using Git and GitHub
- Virtualized dependency management for isolated project setups
- Modular experimentation using separate virtual environments
- Debugging dependency conflicts using npm and Python tooling

3. Workflow Diagram

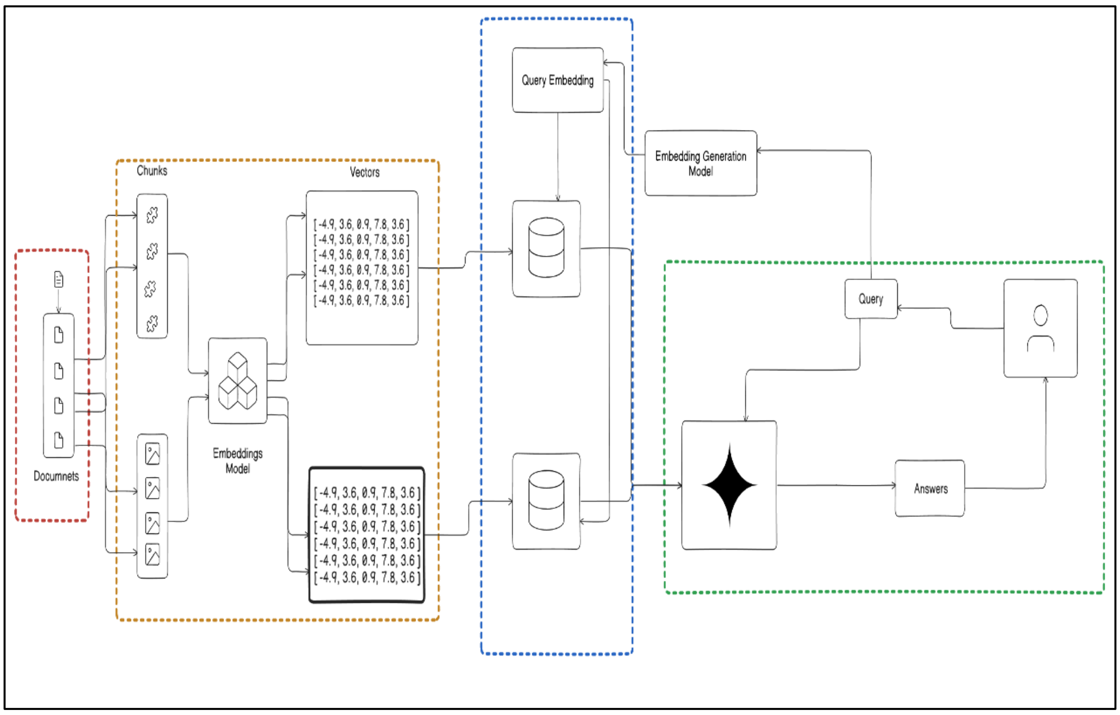


Figure 3.1 RAG Workflow

4. Procedure

The system is designed to convert unstructured enterprise PDF documents into a structured and semantically searchable knowledge base. It follows a modular Retrieval-Augmented Generation (RAG) architecture that includes authentication, document processing, embedding and storage, semantic retrieval, and conversational response generation. The methodology adopted in the development of the system is described below.

A. System Initialization and Environment Configuration

The application begins by loading configuration variables—such as API keys and database credentials—from secure environment files. During initialization, all core components, including the vector database, embedding model, and large language model (LLM) interface, are connected and validated. This ensures that every foundational service is properly configured and operational before the system starts processing user queries.

B. User Authentication and Access Control

A secure authentication layer manages user registration and login. Passwords are hashed prior to storage, and authentication tokens are issued upon successful login. These tokens accompany subsequent user requests, allowing the system to verify user identity, enforce access control, and maintain detailed interaction logs for auditability and traceability.

C. Document Ingestion and Vector Creation

Uploaded enterprise PDFs undergo a multi-stage processing pipeline. Text and images are extracted using a PDF parser, with OCR applied for scanned or image-based content when required. The extracted content is divided into meaningful, context-preserving chunks based on document hierarchy such as chapters, headings, and logical sections. Each chunk is transformed into a dense embedding using a language embedding model. These embeddings, along with metadata (e.g., section titles, document references), are stored in a high-performance vector database to support semantic retrieval.

D. Semantic Retrieval and RAG Pipeline

When a user submits a query, it is first converted into an embedding using the same embedding model. A similarity search is performed within the vector database to retrieve the most relevant content segments. The retrieved context is then integrated into a structured prompt template that is executed by the large language model. This ensures that the generated answer is grounded in the organization's uploaded documents rather than relying solely on generic model knowledge.

E. Response Delivery to the User

The system returns a final response that includes both the generated answer and supporting contextual details such as document references, section identifiers, or source metadata. These references allow users to trace the information back to its original location, enhancing trust, reliability, and transparency. The complete response is then presented through the user interface.

F. Project Progress Status

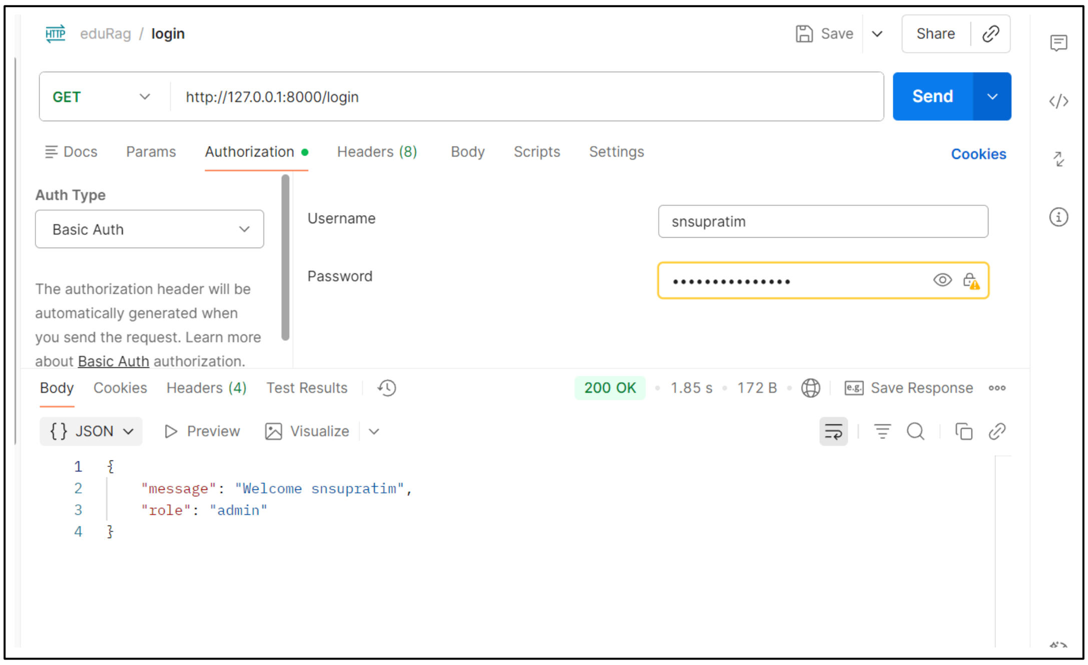
The complete RAG pipeline—including user authentication, document ingestion, embedding generation, vector storage, and semantic query handling—has been successfully implemented and finalized. As part of the document processing workflow, **Tesseract OCR** was integrated to extract textual content from scanned PDFs, images, and tabular data. The inclusion of Tesseract significantly improved text extraction accuracy, enabling the system to handle non-text PDFs and image-based documents more effectively. This enhancement resulted in better document indexing, improved semantic understanding, and more accurate query responses.

5. Results and discussion

The proposed system has been partially implemented and evaluated in stages to verify the correctness of the core Retrieval-Augmented Generation (RAG) pipeline. Since the image and table extraction modules are still under development, the current evaluation focuses exclusively on text-based extraction, embedding generation, vector storage, and semantic retrieval.

A. Backend Functionality and RAG Workflow

The RAG pipeline has been successfully deployed through the backend API. Text extraction from PDF documents, chunk segmentation, embedding generation, and semantic retrieval have all been validated. Responses produced by the large language model consistently utilized retrieved contextual segments rather than relying on generic model knowledge, confirming proper grounding. Testing performed through Postman demonstrated correct handling of authentication workflows, efficient retrieval of relevant document portions, and stable API response times under repeated and continuous queries.



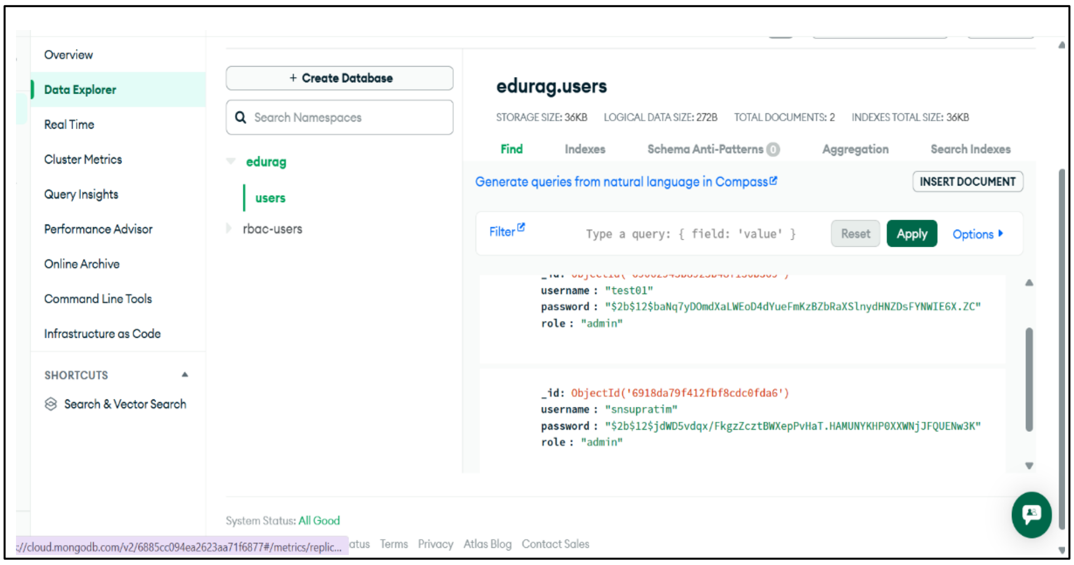
```
{
    "message": "Welcome snsupratim",
    "role": "admin"
}
```

Figure 5.1 Postman

B. Retrieval Accuracy and Response Quality

Preliminary evaluations show that the system can accurately retrieve semantically relevant document segments even when user queries do not contain exact keywords from the text. Responses generated by the LLM reflect a clear understanding of contextual information sourced from the ingested PDFs. Although full quantitative evaluation will be conducted after all system modules are completed, qualitative observations confirm:

- High relevance between retrieved segments and user intent.
- Low hallucination due to strong context grounding.
- Reliable performance across a variety of document types.



edurag.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 272B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

```
Type a query: { field: 'value' }
```

```
username : "test01"
password : "$2b$12$baNq7yDOmdXaLWEoD4dYueFmKzBZbRaXSlnydHNZDsFYNWIE6X.ZC"
role : "admin"

_id: ObjectId('6918da79f412fbf8cdc0fda6')
username : "snsupratim"
password : "$2b$12$jdWD5vdqx/FkgzZcztBWXepPvHaT.HAMUNYKHP0XXWNjJFQUENw3K"
role : "admin"
```

://cloud.mongodb.com/v2/6885cc094ea2623aa71f6877#/metrics/replic...

Figure 5.2 MongoDB

C. User Interface and Prototype Interaction

A **Vercel v0–based interface** has been developed for demonstration and testing purposes. The interface allows users to upload PDF documents and submit natural language queries through a clean and intuitive graphical user interface. The frontend communicates seamlessly with the backend using RESTful APIs and successfully displays the generated responses along with the relevant supporting contextual excerpts retrieved from the documents. Although image-, chart-, and table-level retrieval functionalities are not yet implemented, the prototype effectively demonstrates the end-to-end feasibility of semantic document search using the implemented RAG pipeline.

D. Performance metrics

--- Evaluation ---

PDF time per page (s): 0.0002

PDF time per document (s): 63.82

OCR accuracy (%): 96.43

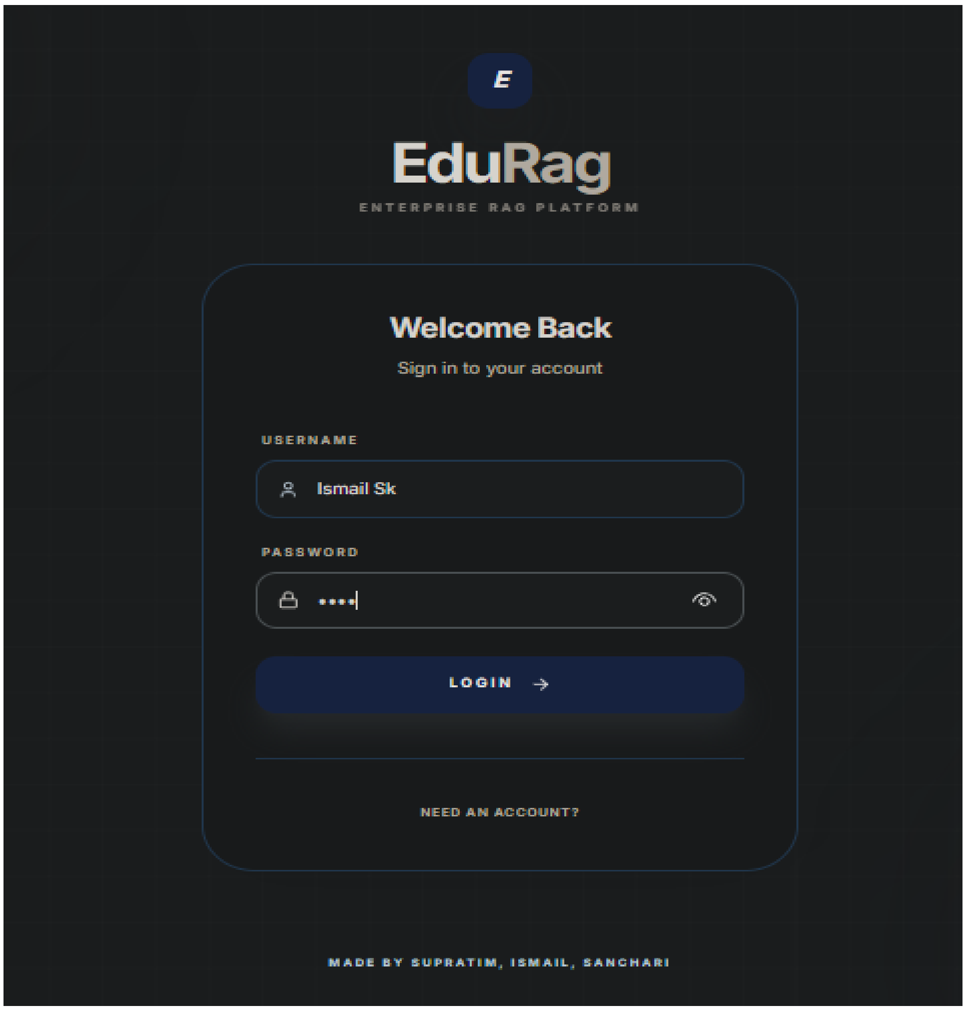
Chunking accuracy (%): 66.67

Search latency (s): 1.2781

Precision / Recall / MRR: 0.2 1.0 1.0

Table extraction accuracy (%): 95.00

Pinecone vector count: 382



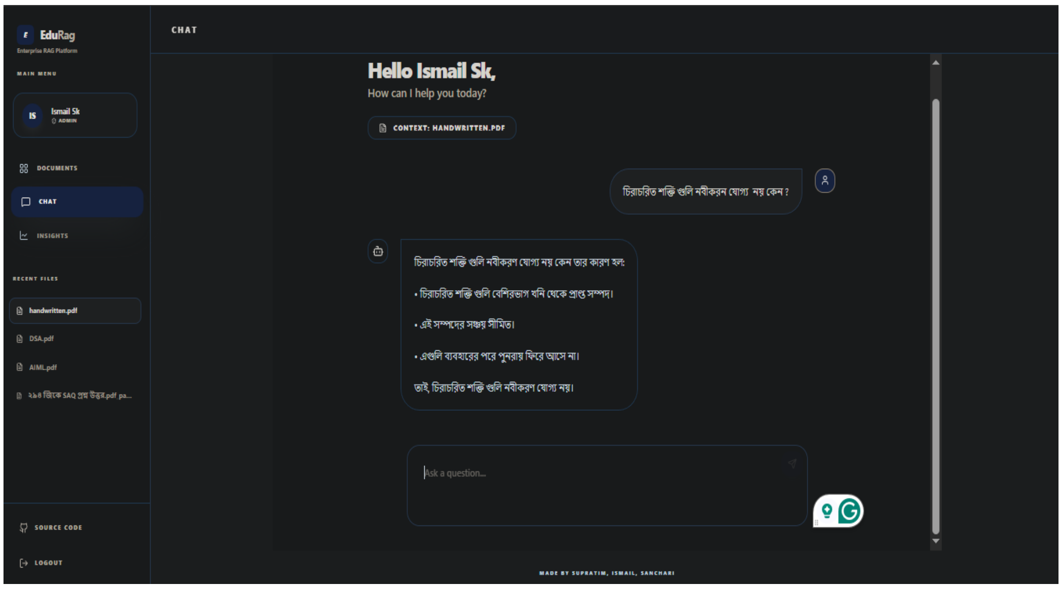
Welcome Back

Sign in to your account

USERNAME: Ismail Sk

PASSWORD: ••••

Figure 5.3 Website Login



Hello Ismail Sk,

How can I help you today?

চিরাচরিত শক্তি গুলি নবীকরন যোগ্য নয় কেন ?

চিরাচরিত শক্তি গুলি নবীকরণ যোগ্য নয় কেন তার কারণ হল:

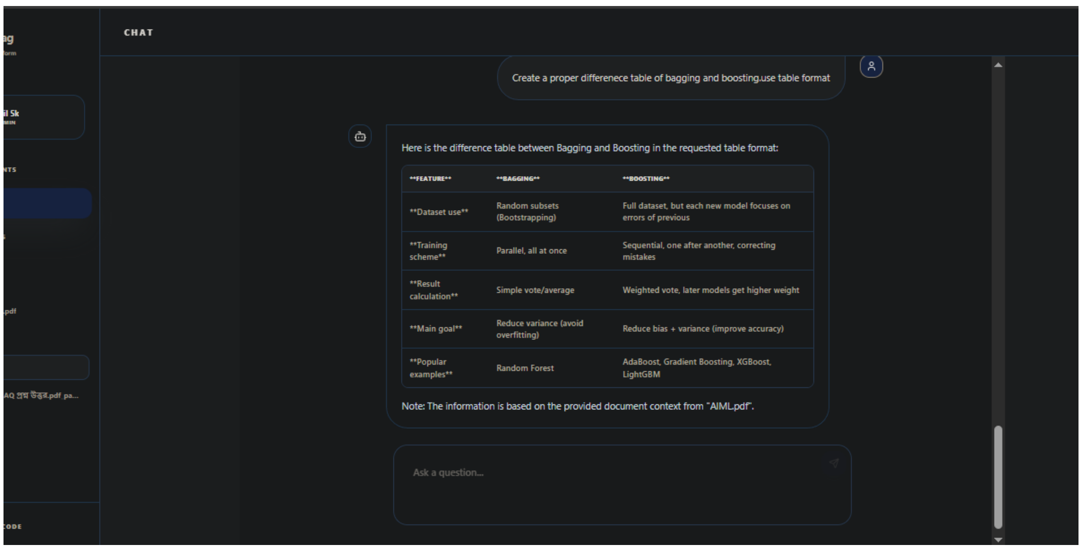
• চিরাচরিত শক্তি গুলি বেশিরভাগ খনি থেকে প্রাপ্ত সম্পদ।

• এই সম্পদের সঞ্চয় সীমিত।

• এগুলি ব্যবহারের পরে পুনরায় ফিরে আসে না।

তাই, চিরাচরিত শক্তি গুলি নবীকরণ যোগ্য নয়।

Figure 5.4 Multilingual Chatbot’s Reply



Create a proper differenece table of bagging and boosting.use table format

Here is the difference table between Bagging and Boosting in the requested table format:

FEATURE	**BAGGING**	**BOOSTING**
Dataset use	Random subsets (Bootstrapping)	Full dataset, but each new model focuses on errors of previous
Training scheme	Parallel, all at once	Sequential, one after another, correcting mistakes
Result calculation	Simple vote/average	Weighted vote, later models get higher weight
Main goal	Reduce variance (avoid overfitting)	Reduce bias + variance (improve accuracy)
Popular examples	Random Forest	AdaBoost, Gradient Boosting, XGBoost, LightGBM

Note: The information is based on the provided document context from "AIML.pdf".

Figure 5.5 Structured Tabular Output

6. Future scope of the work

The system is designed to support further enhancements during the continuation of the industrial training. Key future developments include:

- Development of performance benchmarking modules to evaluate relevance accuracy, retrieval latency, scalability, and memory efficiency.
- Implementation of advanced security features such as fine-grained access control, audit logging, API rate limiting, and improved credential protection.
- Containerization and automated deployment using Docker or similar tools to streamline deployment across cloud and on-premise environments.
- Introduction of collaboration and analytics capabilities, including usage statistics dashboards, frequently accessed topics, and cross-document insights to support enterprise knowledge discovery.
- As a future enhancement, the project aims to support robust processing of handwritten documents and low-quality scanned PDFs by integrating advanced image enhancement techniques and handwriting-aware OCR models

7. Conclusion

This project demonstrates the feasibility of converting enterprise PDF documents into a structured and semantically searchable knowledge base using a Retrieval-Augmented Generation (RAG) pipeline. The implemented system successfully performs text extraction, segmentation, embedding generation, vector database storage, and LLM-based contextual response generation, enabling semantic search rather than traditional keyword matching. Backend evaluations using Postman confirm that the system can retrieve relevant content and produce grounded responses tied to the uploaded PDFs. Although the current implementation focuses on text-based retrieval only, the results show significant potential for improving document accessibility and reducing manual information lookup. With future enhancements such as multimodal extraction (images, tables, charts), multilingual support, and a fully developed user interface, the system can be scaled into an enterprise-ready solution for intelligent knowledge management.

References

[1] LangChain Documentation. Accessed: Jan. 2025. [Online]. Available: https://python.langchain.com

[2] Pinecone Documentation. Accessed: Jan. 2025. [Online]. Available: https://www.pinecone.io

[3] FastAPI Documentation. Accessed: Jan. 2025. [Online]. Available: https://fastapi.tiangolo.com

[4] Streamlit Documentation. Accessed: Jan. 2025. [Online]. Available: https://streamlit.io

[5] Google Generative AI (Gemini) API Documentation. Accessed: Jan. 2025. [Online]. Available: https://ai.google.dev

[6] Groq API Documentation. Accessed: Jan. 2025. [Online]. Available: https://console.groq.com/docs

[7] PyMuPDF Documentation. Accessed: Jan. 2025. [Online]. Available: https://pymupdf.readthedocs.io

[8] Tesseract OCR Documentation. Accessed: Jan. 2025. [Online]. Available: https://github.com/tesseract-ocr