Project Report

on

AI-Driven Knowledge Synthesis Platform

In partial fulfilment of requirements for the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

UMANG GOSWAMI [22100BTCSAII11062]

SHREYANSH GUPTA [22100BTCSAII11052]

NAMAN MATHUR [22100BTCSAII11027]

Under the guidance of

Prof. Neeraj Mehta

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JAN - JUN 2025

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE
SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We here declare that work which is being presented in the project entitled **"AI-Driven Knowledge Synthesis Platform"** in partial fulfilment of degree of **Bachelor of Technology in Computer Science & Engineering** is an authentic record of our work carried out under the supervision and guidance of **Prof. Neeraj Mehta Asst. Professor** of Computer Science & Engineering. The matter embodied in this project has not been submitted for the award of any other degree.

UMANG GOSWAMI [22100BTCSAII11062]

SHREYANSH GUPTA [22100BTCSAII11052]

NAMAN MATHUR [22100BTCSAII11027]

Date:

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDOR

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROJECT APPROVAL SHEET

The following team has done the appropriate work related to the **"AI-Driven Knowledge Synthesis Platform"** in partial fulfilment for the award of **Bachelor of Technology in Computer Science & Engineering** of "SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY" and is being submitted to SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE.

Team:

1. Umang Goswami [22100BTCSAII11062]
2. Shreyansh Gupta [22100BTCSAII11052]
3. Naman Mathur [22100BTCSAII11027]

Internal Examiner **External Examiner**

Date:

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE
SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Mr. Umang Goswami**, **Mr. Shreyansh Gupta** and **Mr. Naman Mathur** working in a team have satisfactorily completed the project entitled **"AI-Driven Knowledge Synthesis Platform"** under the guidance of Prof. Neeraj Mehta in the partial fulfilment of the degree of **Bachelor of Technology in Computer Science & Engineering** awarded by SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY affiliated to SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE during the session Jan - Jun 2025.

Prof. Neeraj Mehta	Dr. Abhishek Singh Rathore
Project Guide	Head, Department of Computer Science & Engineering

Dr. Anand Rajavat
Director & Head,
Department of Computer Science & Engineering

ACKNOWLEDGEMENT

We are grateful to a number of persons for their advice and support during the time of complete our project work. First and foremost our thanks goes to Dr. Abhishek Singh Rathore Head of the Department of Computer Science & Engineering and Mr. Neeraj Mehta the mentor of our project for providing us valuable support and necessary help whenever required; and, also helping us explore new technologies by the help of their technical expertise. His direction, supervision and constructive criticism were indeed the source of inspiration for us.

We would also like to express our sincere gratitude towards our Director Dr. Anand Rajavat for providing us valuable support.

We forward our sincere thanks to all teaching and non-teaching staff of Computer Science & Engineering department, SVVV Indore for providing necessary information and kind co-operation.

We would like to thanks our parents and family members, our classmates and our friends for their motivation and their valuable suggestion during the project. Last, but not the least, we thank all those people, who have helped us directly or indirectly in accomplishing this work. It has been a privilege to study at SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE.

ABSTRACT

The AI-Driven Knowledge Synthesis Platform automates the transformation of unstructured data—such as PDFs, e-books, spreadsheets, and web articles—into structured, multimodal outputs including summaries, quizzes, flashcards, and podcasts. Leveraging Retrieval-Augmented Generation (RAG) pipelines, LangChain orchestration, and transformer-based NLP models, the platform enables dynamic cross-document knowledge synthesis tailored to user roles like students, educators, and professionals. It utilizes semantic embeddings stored in a FAISS vector database for accurate and efficient retrieval. The system features a Gradio- based web interface, Python backend, and cloud deployment via Microsoft Azure. Designed for scalability, personalization, and automation, it significantly reduces manual content curation time, enhances learning and training outcomes, and supports educational institutions and corporate environments with customizable AI-generated content.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

CHAPTER 1 – INTRODUCTION

1. Introduction

In today's data-driven world, organizations are inundated with vast amounts of unstructured digital documents — including PDFs, spreadsheets, presentations, and reports — stored across different platforms and in multiple formats. Manually searching, extracting, and synthesizing meaningful information from this data is time-consuming and inefficient. The evolution of Natural Language Processing (NLP), semantic search, and vector databases has made it possible to build intelligent systems that understand and retrieve relevant information with human-like efficiency.

This project presents the development of a Document Understanding and Retrieval System using state-of-the-art AI frameworks like LangChain and vector databases such as FAISS, deployed and scaled on Azure cloud infrastructure. The system ingests multiple file formats, parses them into semantic chunks, embeds them into vector space, and performs retrieval-augmented generation (RAG) to answer queries intelligently.

The need for such a system is evident in enterprise settings, academic environments, legal research, and even personal productivity tools. This project aims to provide a modular, cloud-scalable solution that automates the ingestion, chunking, embedding, and retrieval of documents, producing contextual responses in real time.

This report details the entire system design, from requirement analysis and architecture to UML diagrams, deployment models, and future scope. It also evaluates the system's performance on parameters like speed, scalability, and usability.

2. Problem Statement

Organizations and users regularly deal with unstructured documents in formats like PDFs, Word files, presentations, and spreadsheets. Extracting relevant information or answering specific questions from such documents remains a manual and inefficient task. Traditional keyword-based search engines often fail to grasp the semantic context, leading to poor results.

There is a pressing need for an AI-powered system that can intelligently understand document content, chunk it into meaningful segments, and store it in a format that supports fast, context-aware retrieval. The system should not only fetch relevant segments but also generate human-like responses based on the retrieved context.

This project addresses the problem by building a scalable Document Understanding and Retrieval System that leverages semantic embeddings, retrieval-augmented generation (RAG), and multimodal output synthesis — all deployed on cloud infrastructure to ensure performance, scalability, and availability.

3. Objective

- Automate content extraction from unstructured documents.
- Generate multimodal outputs like quizzes, summaries, flashcards, and podcasts.
- Enable cross-document synthesis and personalization.
- Support scalable deployment for institutions and enterprises.

4. Modules of the System

The system is structured into several core modules, each responsible for a distinct stage in the document understanding and response generation pipeline. These modules work in coordination to deliver accurate, personalized, and multimodal outputs.

4.1. Document Ingestion

This module handles the initial intake of documents in various formats, such as PDFs, Word documents, spreadsheets, and presentation slides. It includes functionalities for:

i. Uploading documents via user interfaces or APIs.

ii. Converting them into machine-readable text using OCR (for scanned files) or parsers.

iii. Organizing documents into a consistent internal structure for downstream processing.

4.2. Knowledge Extraction

Once documents are ingested, this module processes the raw content to extract meaningful semantic units. It includes:

i. Text segmentation (or chunking) using natural language processing techniques.

ii. Metadata extraction such as headings, authorship, and timestamps.

iii. Entity recognition and relation extraction to build a structured knowledge representation.

4.3. Cross-Document Retrieval

This module enables semantic search across multiple documents by leveraging vector-based similarity measures. It includes:

i. Generating embeddings for each document chunk using transformer-based models.

ii. Storing these embeddings in a vector database (e.g., FAISS) for fast nearest-neighbour retrieval.

iii. Accepting user queries and retrieving the most contextually relevant chunks across the document corpus using a Retrieval-Augmented Generation (RAG) approach.

4.4. Multimodal Response Generation

After retrieving the relevant content, this module uses generative AI models (like GPT) to synthesize responses in various formats:

i. Textual summaries and explanations.

ii. Interactive quizzes based on extracted key points.

iii. Flashcards for revision and learning.

iv. Audio outputs or podcast scripts for auditory consumption.

5. Scope

The scope of this project encompasses the design, development, and deployment of an AI-powered document understanding system using a Retrieval-Augmented Generation (RAG) pipeline. The system is built to handle diverse document formats including PDFs, spreadsheets, and presentation slides, and enables end- to-end semantic search and synthesis capabilities.

Key components include:

i. **Document Ingestion:** Upload and preprocess multiple document types from users.

ii. **Semantic Chunking:** Parse documents into meaningful text segments using natural language processing (NLP).

iii. **Vector Embedding and Storage:** Generate and store semantic embeddings in a FAISS vector database for efficient retrieval.

iv. **Query Processing & Retrieval:** Accept user queries and retrieve contextually relevant document segments using a RAG architecture.

v. **Response Generation:** Use generative AI models to synthesize coherent, informative, and multimodal responses.

vi. **Cloud Deployment:** Implement the entire system on Microsoft Azure with features such as auto-scaling, caching, load-balancing, and high availability.

The system is designed to support scalability, fast response time (<3 seconds), and robust performance for enterprise-grade applications like knowledge management, research automation, legal document analysis, and customer support.

CHAPTER – 2

LITERATURE SURVEY

CHAPTER 2 – LITERATURE SURVEY

6. Existing System

In recent years, the integration of artificial intelligence in education and productivity platforms has seen a rise. However, most of the existing systems offer limited and fragmented functionalities. For instance:

- Quizlet: a widely used platform, enables users to create flashcards manually. While it supports learning through spaced repetition and collaborative usage, it is heavily dependent on user-generated content and lacks any advanced automation or AI-based synthesis capabilities.
- Grammarly: another prevalent tool, focuses primarily on grammar checking and language refinement. Although powered by NLP techniques, its scope is narrowly limited to writing enhancement and does not cater to knowledge extraction or multimodal content creation.

Furthermore, traditional document summarization tools are generally single-format focused and lack the capability to process and integrate information across multiple types of documents such as PDFs, spreadsheets, and web pages.

6.1. Limitations in Existing Systems:

- No support for cross-format knowledge synthesis.
- Lack of personalized or dynamic output generation.
- Minimal or no use of advanced AI-based document retrieval and synthesis pipelines.
- Outputs are either static or user-dependent with little automation.

7. Proposed System

The proposed system, for AI-Driven Knowledge Synthesis Platform, is an intelligent, cloud-based solution designed to automate the process of document understanding, knowledge extraction, and multimodal output generation. Unlike

traditional tools that operate in silos —such as basic summarizers, grammar checkers, or flashcard creators—this platform offers a unified workflow powered by advanced Natural Language Processing (NLP) and Retrieval-Augmented Generation (RAG) techniques.

At its core, the system allows users to upload various types of documents including PDFs, spreadsheets, and presentations. These documents are parsed and semantically broken down into meaningful chunks using models orchestrated by LangChain. Each chunk is transformed into a high-dimensional embedding vector and stored in a FAISS vector database, enabling context- aware semantic search. When a user submits a query, the system retrieves the most relevant document segments using both keyword and vector similarity, and passes them through generative language model to create human-like responses.

What sets this system apart is its ability to generate multimodal outputs—including textual summaries, quizzes, flashcards, and podcasts—tailored to different user roles such as students, educators, or corporate professionals. The entire application is containerized using Docker and deployed on Microsoft Azure, ensuring scalability, high availability, and ease of integration into institutional or enterprise environments. This proposed system thus addresses a critical gap in intelligent document processing and stands as a comprehensive, user-friendly, and scalable solution for modern knowledge management needs.

7.1. Core Highlights:

- **Document Parsing:** Ingests and converts diverse formats like PDFs, spreadsheets, e-books, and web articles into structured data.
- **Retrieval-Augmented Generation (RAG):** Uses semantic and keyword-based search to fetch relevant chunks from a document base, enabling high-quality content generation.
- **Multimodal Output Generation:** Automatically creates summaries, quizzes, flashcards, and even audio-based outputs (like podcasts) based on user needs.

The platform offers personalization through role-based output generation and ensures scalability via cloud deployment. Unlike existing tools, it leverages vector

databases (FAISS), LangChain orchestration, and transformer-based NLP models to provide a robust end-to-end solution.

7.2. Feasibility Study

Feasibility analysis determines the viability of implementing the proposed system from multiple dimensions:

7.2.1. Technical Feasibility

The system leverages proven, open-source and cloud-compatible technologies:

- Transformers from Hugging Face for embedding and generation tasks.
- LangChain, which allows seamless orchestration of RAG pipelines.
- Vector databases (FAISS) for fast, similarity-based content retrieval.
- The backend is developed in Python, ensuring compatibility with popular AI frameworks.

Frontend is built using Gradio for fast and interactive user interfaces. These tools are mature, well-supported, and capable of handling real-time data processing and transformation.

7.2.2. Economic Feasibility

- Deployment via Microsoft Azure offers a flexible pricing model with pay-as- you-go plans, making it cost-efficient for academic institutions and startups.
- Open-source libraries and frameworks minimize initial development and licensing costs.
- Containerization using Docker enables portability and reduces infrastructure related expenditures.
- Thus, from a cost-benefit perspective, the system is economically viable for small teams and scalable enough for enterprise environments.

7.2.3. Operational Feasibility

- The interface is intuitive and accessible via a standard web browser, requiring no installation.
- Built using Gradio, it supports interactive document uploads, role-based personalization, and easy navigation.
- Since outputs are generated automatically using AI pipelines, users (educators, students, professionals) can benefit without requiring any technical background.

This makes the system practical and easy to integrate into existing academic and corporate workflows, ensuring high user acceptance and minimal learning curve.

CHAPTER – 3

REQUIREMENTS ANALYSIS

CHAPTER 3 – REQUIREMENTS ANALYSIS

8. Requirement Analysis Method

The requirement analysis for this project was carried out using a combination of interviews, observation, and document analysis to thoroughly understand user needs and system expectations. The approach included the following steps:

8.1. Stakeholder Identification:

- Identified key stakeholders such as end-users (students, researchers), system administrators, and technical teams involved in implementation.

8.2. Requirement Gathering Techniques:

- **Interviews & Questionnaires:** Conducted structured discussions with users to gather expectations about functionalities like document upload, search accuracy, response time, and system accessibility.
- **Document Study:** Reviewed technical papers and similar existing platforms to define benchmark features and non-functional needs.
- **Observation:** Simulated scenarios of document processing and query interactions to understand pain points and usability gaps.

8.3. Requirement Categorization:

- Clearly distinguished between Functional Requirements (FR) and Non-Functional Requirements (NFR).
- Each requirement was mapped to its corresponding system module to ensure traceability.

8.4. Validation and Verification:

- Requirements were verified against stakeholder feedback and validated for technical feasibility with the proposed technology stack (LangChain, FAISS, Azure).

9. Data Requirements

- **Input:** PDFs, Word documents, spreadsheets, e-books
- **Output:** Summaries, Flashcards, Quizzes, Podcasts.

10. Functional Requirements

Functional requirements define the core behavior and capabilities that the system must offer to its users. These features are integral to the operation of the AI-Driven Knowledge Synthesis Platform and are aligned with the system's objectives of enabling automated, multimodal knowledge synthesis. Below is a detailed list of the functional requirements:

10.1. Document Upload and Ingestion

The system shall allow users to upload sources in various formats such as:

- PDF
- DOCX
- Excel sheets (XLSX/CSV)
- PowerPoint presentations (PPTX)
- E-books (EPUB)
- Web articles (via URL input)
- Uploaded documents shall be stored temporarily for processing and retrieval.

10.2. Document Parsing and Preprocessing

- The platform shall extract raw text from uploaded documents using format- specific parsers.
- Content shall be segmented into semantically meaningful chunks using LangChain's document chunking pipeline.
- Metadata such as title, author, and date shall be extracted where available.

10.3. Semantic Embedding and Storage

- Each document chunk shall be converted into vector embeddings using pre-trained transformer models (e.g., BERT or Sentence-BERT).
- These embeddings shall be stored in a FAISS-based vector database to support similarity-based searches.
- The system shall ensure traceability by tagging each embedding with source document references.

10.4. Query Input and Cross-Document Retrieval

- The system shall provide a query interface for users to input questions or topics of interest.
- Upon query submission, the system shall:
  - Perform keyword matching and vector similarity search.
  - Retrieve the most relevant document chunks from across all uploaded documents.
- The system shall present a ranked list of retrieved sources to ensure transparency.

10.5. Multimodal Content Generation

The platform shall dynamically generate output in multiple formats:

- **Summaries:** Both extractive (keyword-based) and abstractive (language generation).
- **Quizzes:** Auto-generated MCQs based on key concepts extracted from documents.
- **Flashcards:** Generated in Q&A format for spaced-repetition learning.
- **Podcasts:** Summarized content converted to speech using text-to-speech APIs.

10.6. Role-Based Personalization

- The system shall allow users to select roles (e.g., Educator, Student, Professional).

- Based on the selected role, output formatting, complexity, and tone shall be adapted. For example, educators may receive presentation slides, while students receive summaries and flashcards.

10.7. User Authentication and Access Control

The system shall implement user authentication to support session-based access. Different access privileges shall be defined based on roles:

- **Admin:** Full access including user management and analytics.
- **General User:** Upload, query, and view outputs.

10.8. Output Download and Sharing

Users shall be able to download generated outputs in popular formats:

- **Summaries and flashcards:** PDF or DOCX
- **Quizzes:** PDF or HTML
- **Podcasts:** MP3 or WAV
- Outputs shall be optionally shareable via unique public URLs.

10.9. Feedback Collection

- The system shall offer feedback mechanisms to rate the accuracy and relevance of outputs.
- This feedback shall be stored to improve future results via retraining or fine-tuning.

10.10. System Logs and Monitoring

- All user activities (upload, query, generation) shall be logged for monitoring and analytics.
- Admin users shall be able to view system usage reports and error logs via a dashboard.

11. Non-Functional Requirements

Non-functional requirements define the performance characteristics, quality attributes, and constraints under which the system must operate. These requirements

ensure that the system is usable, efficient, reliable, scalable, and secure, which are essential for both academic and professional adoption.

The non-functional requirements for the AI-Driven Knowledge Synthesis Platform are categorized as follows:

11.1. Performance Requirements

- The system shall process user queries and generate responses within a maximum latency of 3 seconds under normal load conditions.
- Embedding generation and document parsing for a standard document (less than 10MB) shall complete within 5–8 seconds.
- Simultaneous processing of at least 20 concurrent users shall be supported without degradation in performance on standard Azure VM configuration.

11.2. Scalability

- The platform shall be horizontally scalable to accommodate increasing user loads, documents, and queries.
- It shall support auto-scaling of compute resources in the cloud (Azure) during peak usage.
- Modular architecture using microservices and containerization (via Docker) shall enable independent scaling of the document processing, embedding, retrieval, and generation modules.

11.3. Reliability and Availability

- The system shall maintain 99.5% uptime, ensuring high availability for users across different time zones.
- All critical services shall be monitored using health checks, and failure recovery mechanisms (such as automatic restarts) shall be in place.
- Azure’s distributed infrastructure shall be leveraged to ensure redundancy and failover support.

11.4. Usability

- The user interface shall be intuitive, clean, and responsive, making it accessible to users with minimal technical expertise.
- Instructions and tooltips shall be available for every major feature (e.g., file upload, output options).
- The platform shall support keyboard navigation, form validation, and mobile responsiveness for accessibility on tablets and smartphones.

11.5. Maintainability

- The system architecture shall follow a modular structure, ensuring ease of updates and bug fixes.
- Code shall adhere to standard coding conventions (e.g., PEP 8 for Python).
- Version control shall be implemented via Git, with proper documentation for all modules to support collaborative development and future enhancement.

11.6. Portability

- The application shall run in containerized environments and be deployable on any cloud platform (e.g., Azure, AWS, GCP) or on-premise system with Docker support.
- It shall be accessible across all major web browsers (Chrome, Firefox, Edge, Safari).
- No platform-specific features shall be hardcoded to ensure cross-platform compatibility.

11.7. Security Requirements

- All data transmissions shall be encrypted using HTTPS with TLS 1.3.
- Uploaded documents shall be sandboxed and removed from storage after processing to ensure data privacy.

- Role-based access control (RBAC) shall be enforced to prevent unauthorized access to administrative functions.
- Authentication tokens (JWT or OAuth2) shall be used to secure user sessions.
- The system shall be regularly tested for vulnerabilities (e.g., XSS, CSRF, injection attacks).

11.8. Compliance and Data Privacy

The system shall comply with applicable data protection regulations (e.g., GDPR, India's DPDP Act) by ensuring:

- No long-term storage of personal documents or user data unless explicitly consented.
- Users can request deletion of their uploaded data at any time.

11.9. Localization and Language Support

- The system shall support generation of outputs (summaries, quizzes, flashcards) in multiple languages (e.g., English, Hindi, Spanish) using multilingual models.
- The UI text shall be translatable based on user preference, supporting basic localization features.

11.10. Auditability and Traceability

Every generated output (summary, quiz, podcast) shall be traceable to its source document chunks through unique identifiers. All user activities (uploads, queries, downloads) shall be logged with timestamps to support auditing.

12. System Specifications

12.1. Hardware

Component	Specification
Processor	Intel Core i3 (10th Gen) or equivalent

Ram	4 GB
Hard Disk	256 GB SSD
Network Adapter	802.11 B/G/N Wireless or Ethernet
Display	720p resolution monitor
Input Devices	Standard Keyboard and Mouse

12.2. Software

Python, Flask, LangChain, HuggingFace Transformers, FAISS, Docker, Microsoft Azure

CHAPTER – 4

DESIGN

CHAPTER 4 – DESIGN

13. Software Requirements Specification (SRS) Summary

The platform is designed to handle a variety of document formats, including PDFs, e-books, spreadsheets, presentations, and web articles. It employs advanced natural language processing models and retrieval-augmented generation (RAG) pipelines to extract, synthesize, and repurpose content into outputs such as summaries, quizzes, flashcards, and podcasts.

These outputs are generated dynamically based on the user's input and role, ensuring a personalized and context-aware experience. The system leverages technologies like LangChain for orchestration, Hugging Face Transformers for semantic understanding, and FAISS for efficient vector-based similarity searches. This ensures that content is not only retrieved accurately but also generated in a way that aligns with the user's intent and learning objectives.

Expected outcomes of the system include a significant reduction in time and effort for manual content curation, improved accuracy in cross-document knowledge synthesis, and an engaging user experience through multimodal output delivery.

This makes the platform especially useful for academic institutions aiming to automate course material creation, professionals compiling reports and training modules, and organizations managing large volumes of informational content. Overall, the SRS presents the AI-Driven Knowledge Synthesis Platform as a forward-thinking, AI-powered solution built to bridge the gap between unstructured knowledge sources and personalized, actionable insights.

This section reiterates the key functional and non-functional requirements identified in Chapter 3, and maps them to the high-level components of the system:

Requirement ID	Description	Mapped Component
FR-1	Upload documents in PDF, spreadsheet, presentation, etc.	Document Ingestion Service

FR-2	Parse and chunk documents into semantic units	Document Parser (LangChain)
FR-3	Generate and store vector embeddings	Embedding Service & FAISS DB
FR-4	Retrieve relevant chunks via multimodal outputs	Retrieval Engine (RAG)
NFR-1	Response time under 3 seconds	Caching & Load-Balancing Layer
NFR-2	Auto-scaling on Azure	Azure Deployment & Scaling Groups
NFR-3	High availability	Azure Availability Sets

13.1. Glossary

- **Chunk:** A semantically coherent segment of a document.
- **Embedding:** A dense numerical vector representing semantic content.
- **RAG (Retrieval-Augmented Generation):** A framework combining retrieval of relevant text snippets with generative models.
- **Multimodal Output:** Content generated in multiple formats (text, audio, flashcards).

13.2. Supplementary Specifications

- **Security:** All document uploads and outputs are encrypted in transit via HTTPS; stored embeddings are encrypted at rest with Azure Key Vault.
- **Accessibility:** The UI complies with WCAG 2.1 AA guidelines.
- **Internationalization:** Supports English and Hindi localization.

14. Use Case Mode

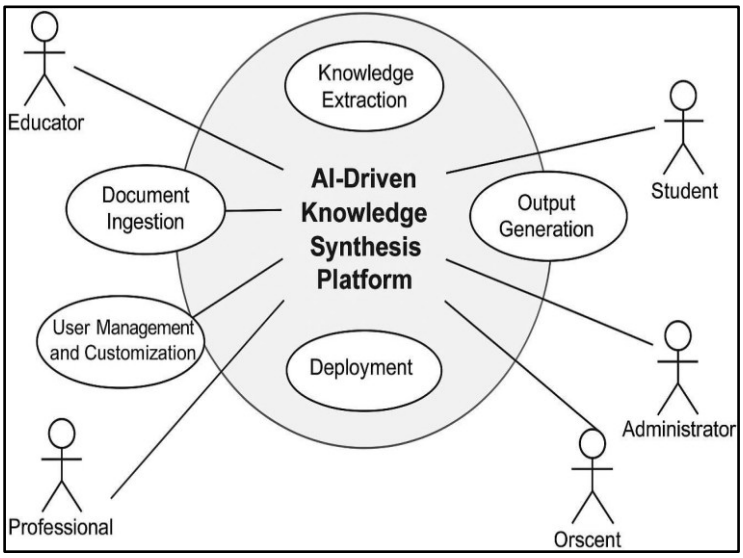

Figure 1– Use Case Diagram

15. Conceptual Class Diagram

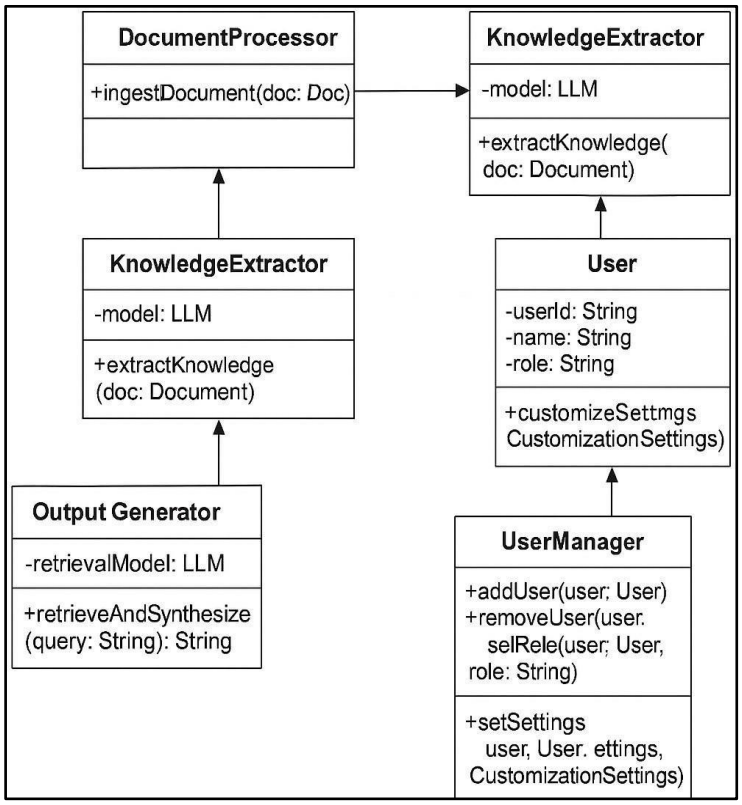

Figure 2 – Conceptual Class Diagram

16. Activity Diagram

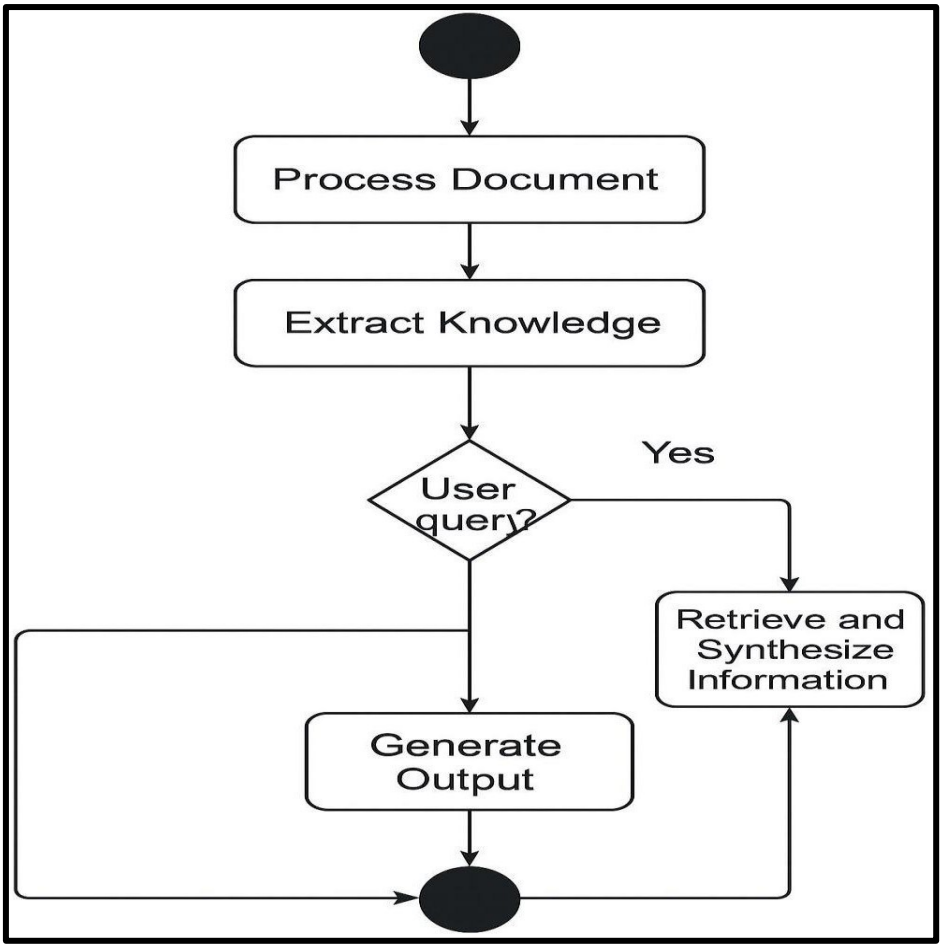

Figure 3 – Activity Diagram

17. Data Flow Diagrams

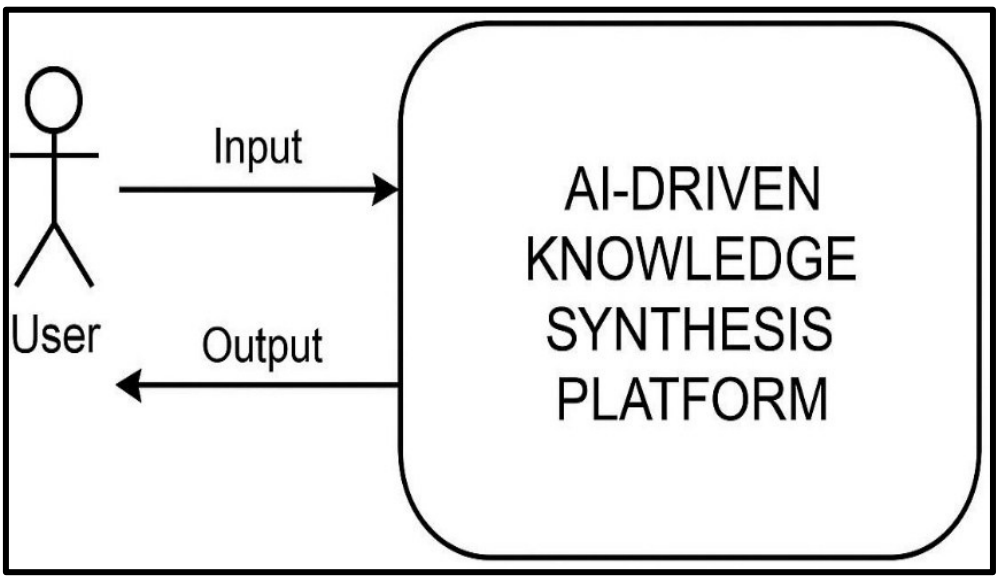

Figure 4 – DFD Level 0

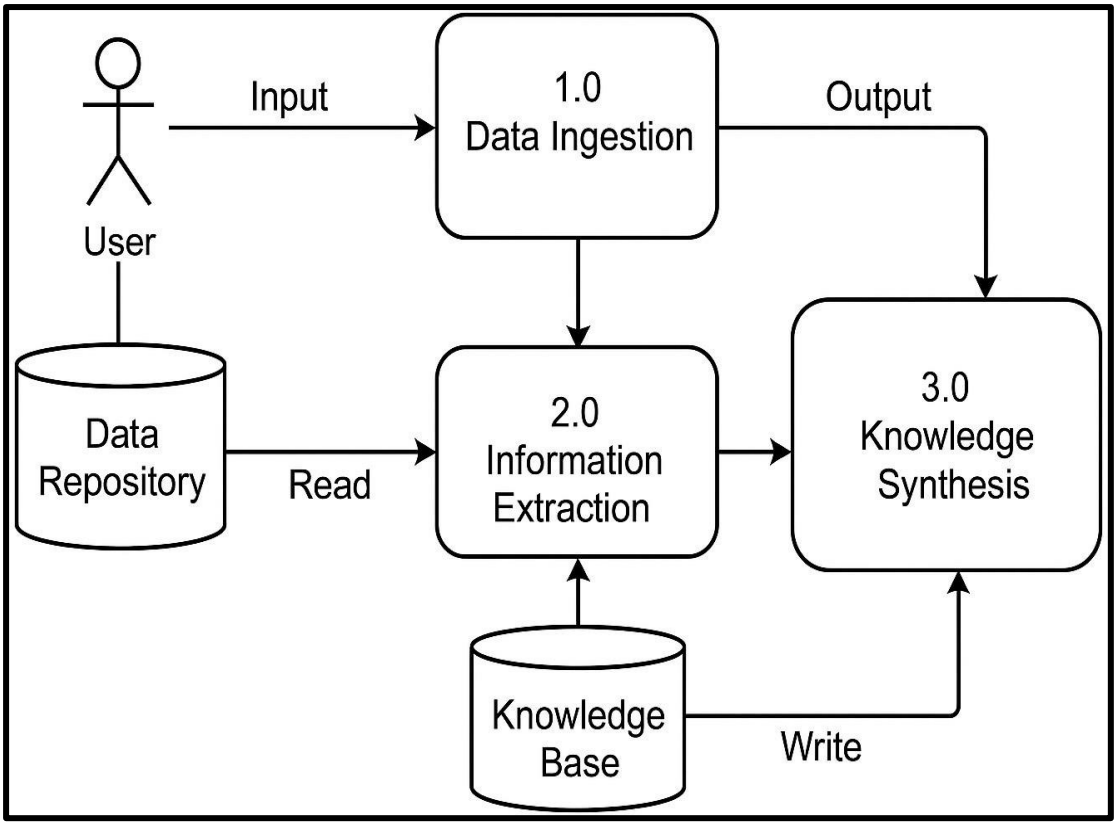

Figure 5– DFD Level 1

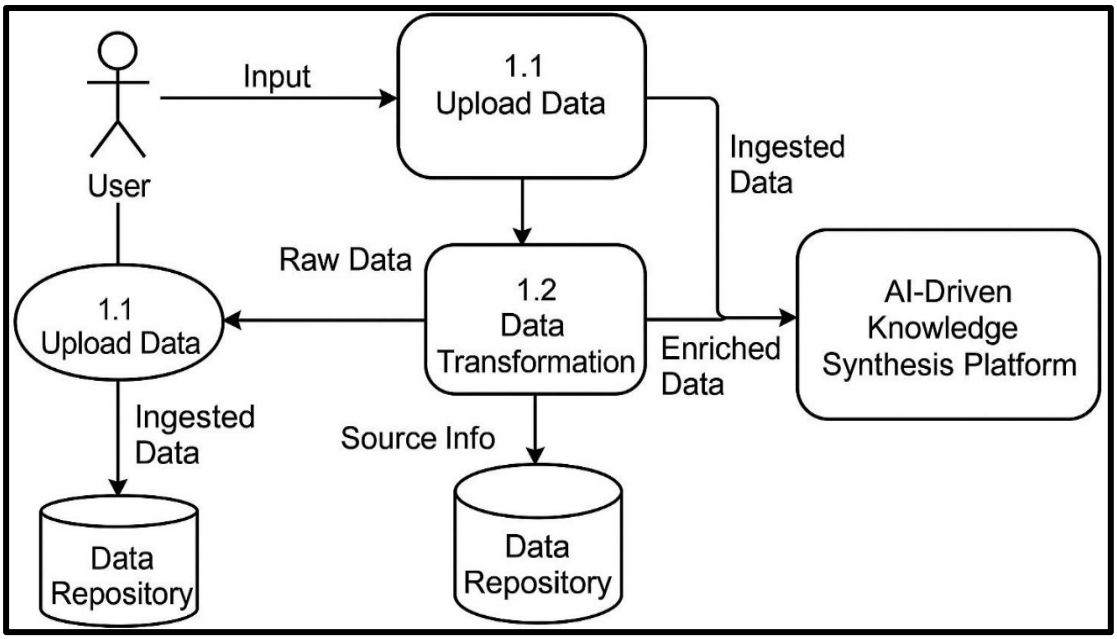

Figure 6 – DFD Level 2

18. Database Design (ER Diagram)

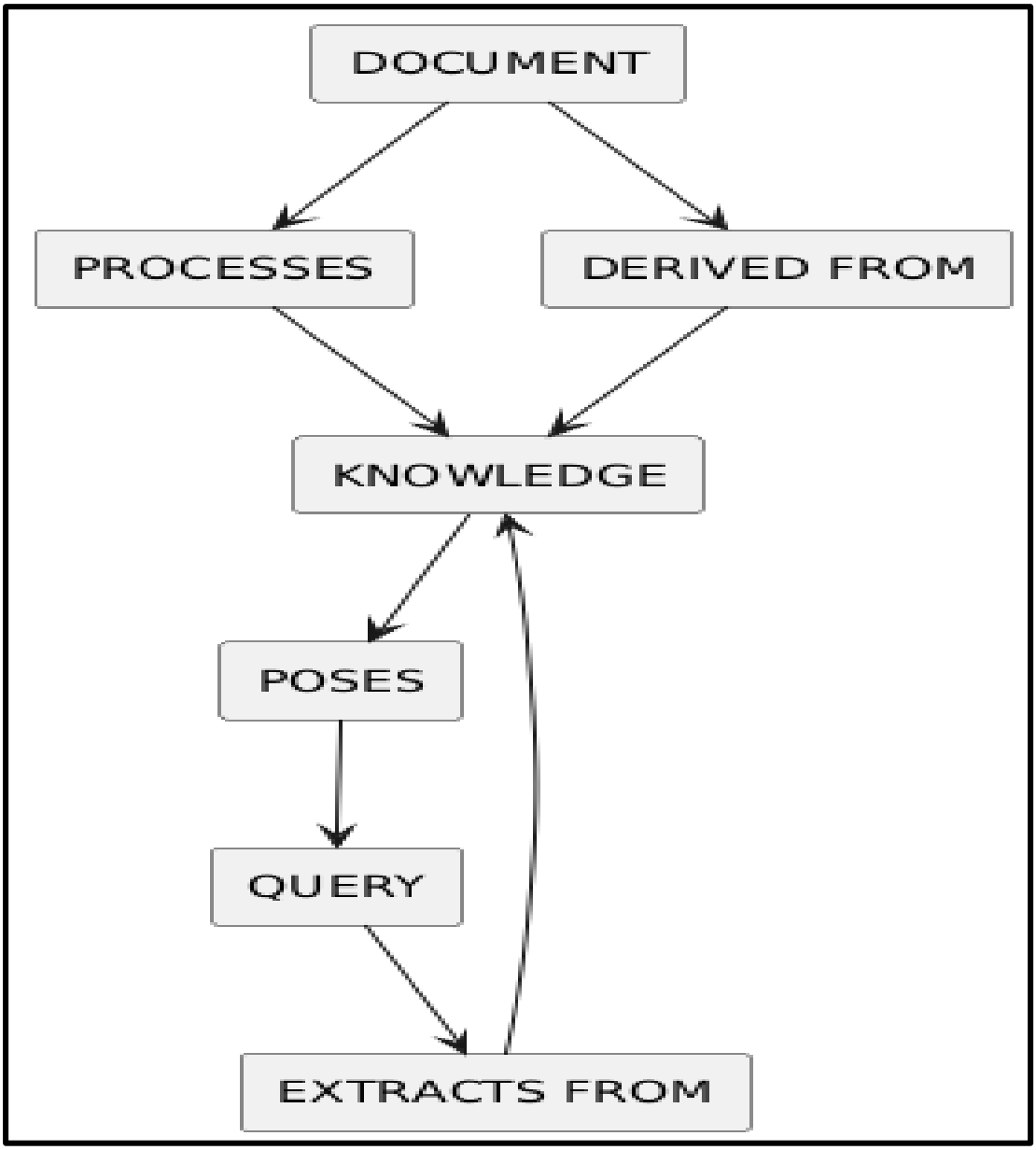

Figure 7 – ER Diagram

CHAPTER – 5

SYSTEM MODELING

CHAPTER 5 – SYSTEM MODELING

19. Detailed Class Diagram

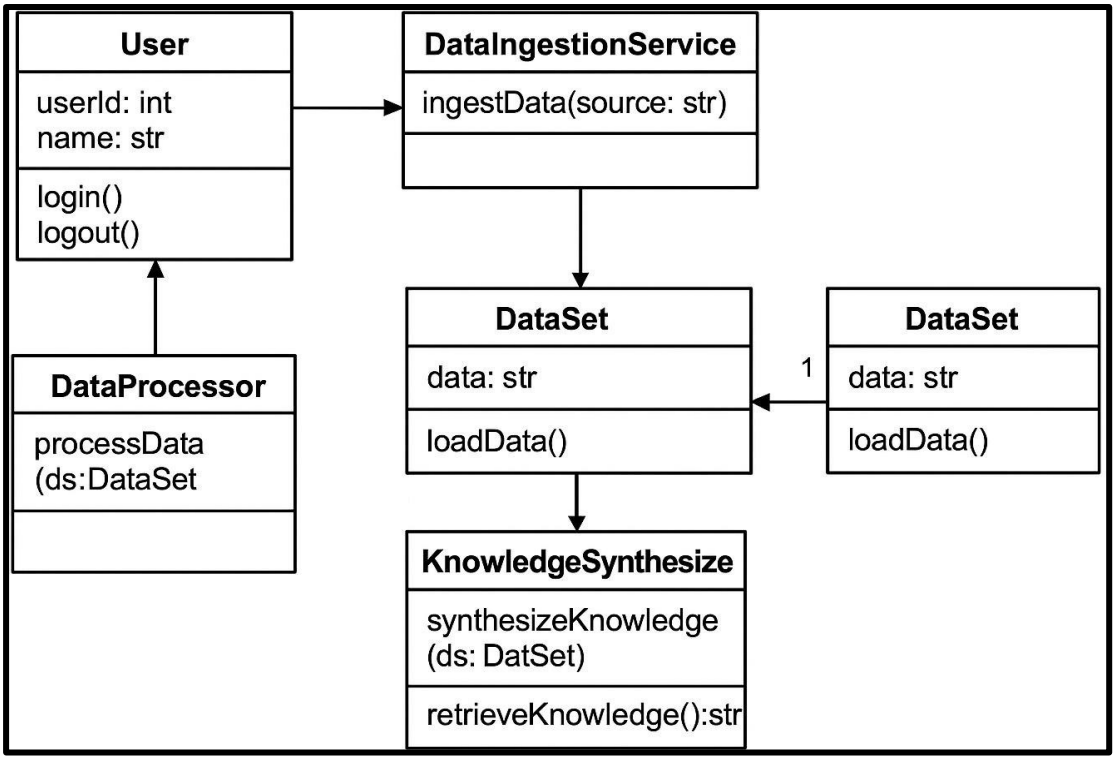

Figure 8 – Detailed Class Diagram

20. Interaction Diagrams

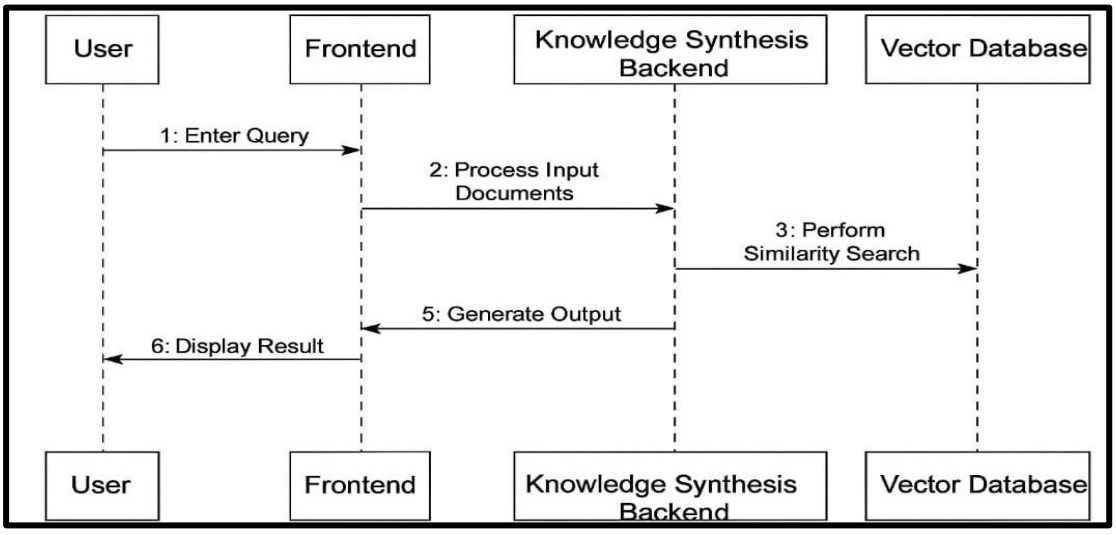

Figure 9 – Sequence Diagram

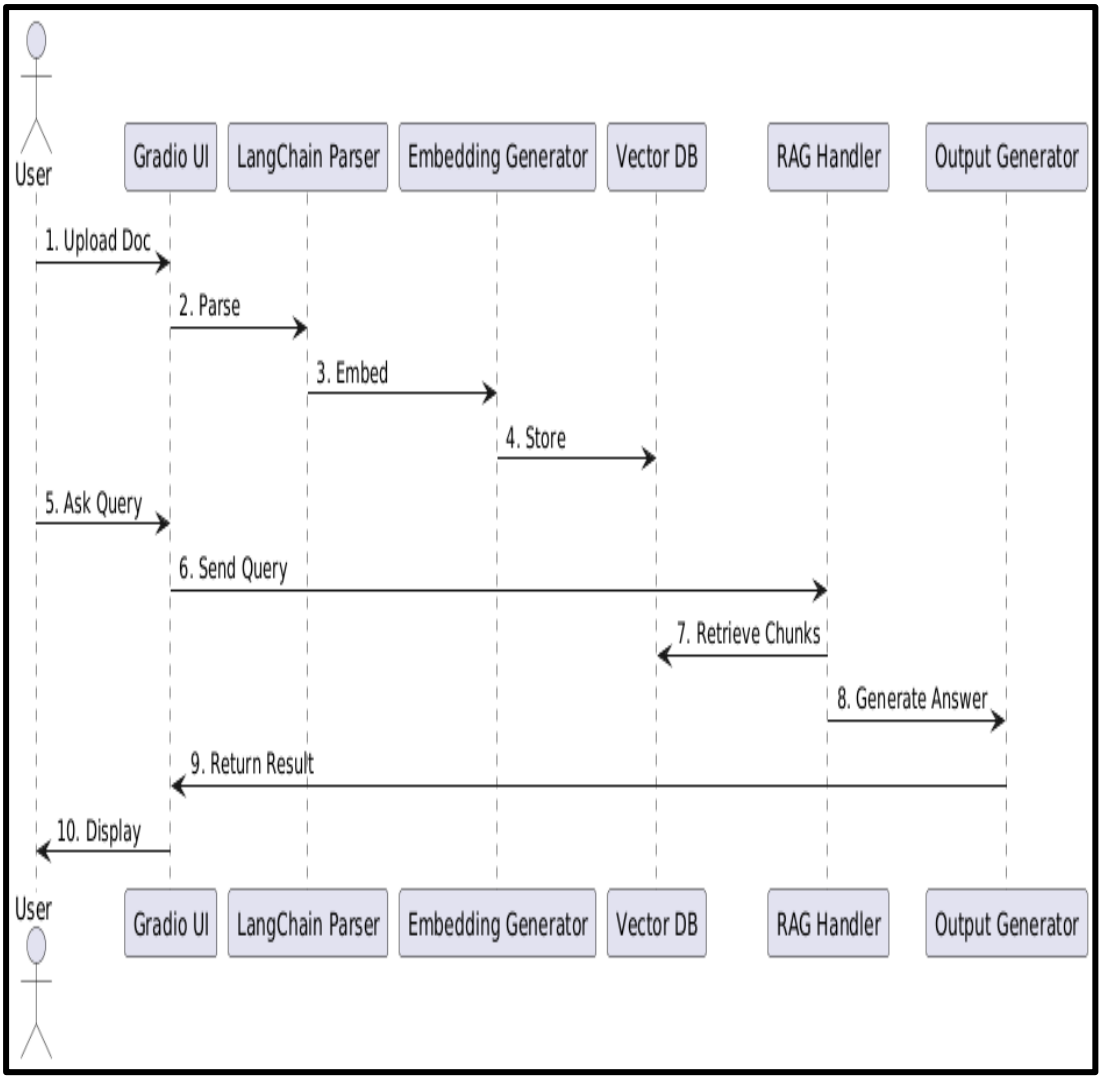

Figure 10 – Collaboration Diagram

21. State Diagram

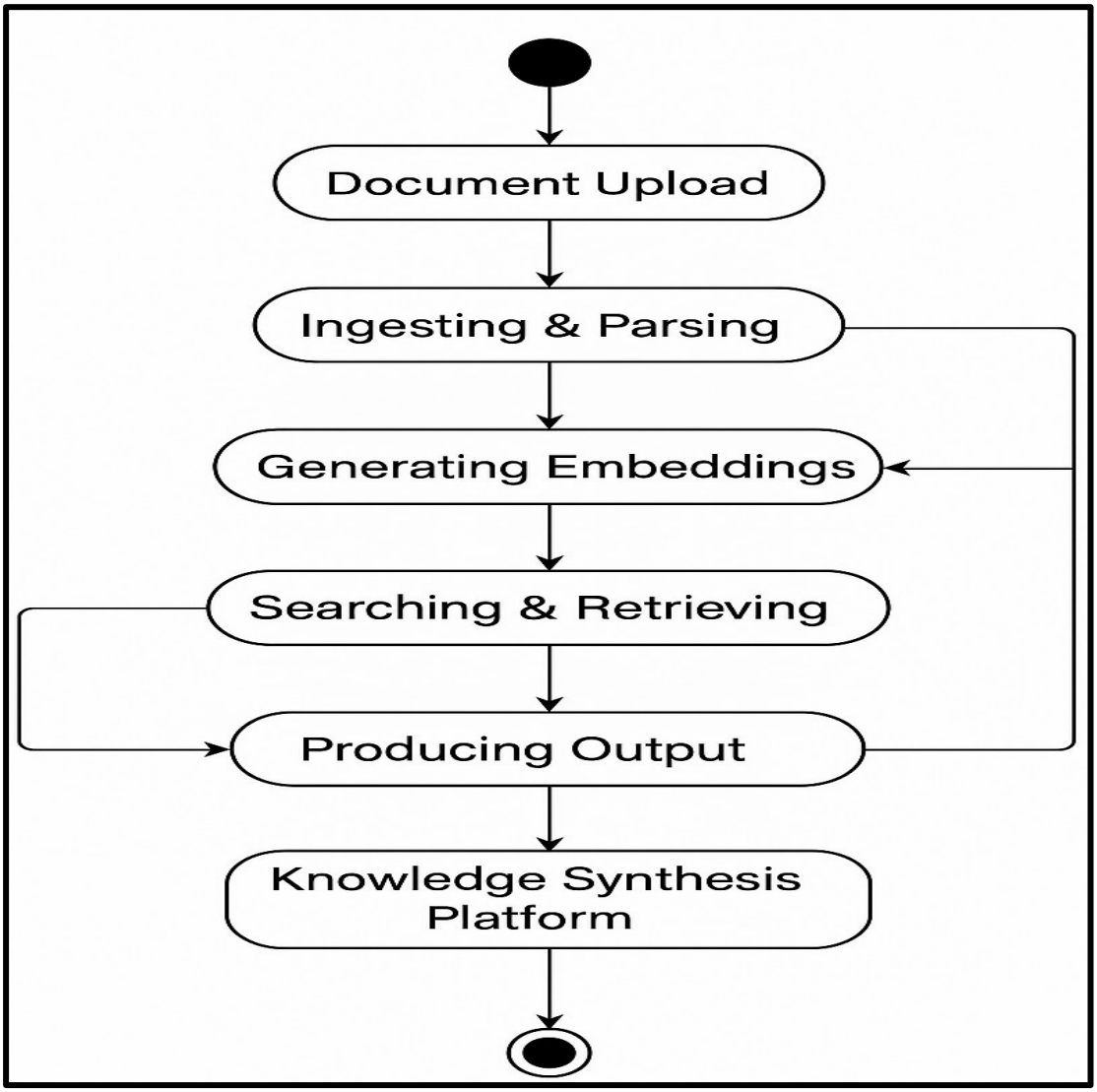

Figure 11 – State Diagram

22. Activity Diagram

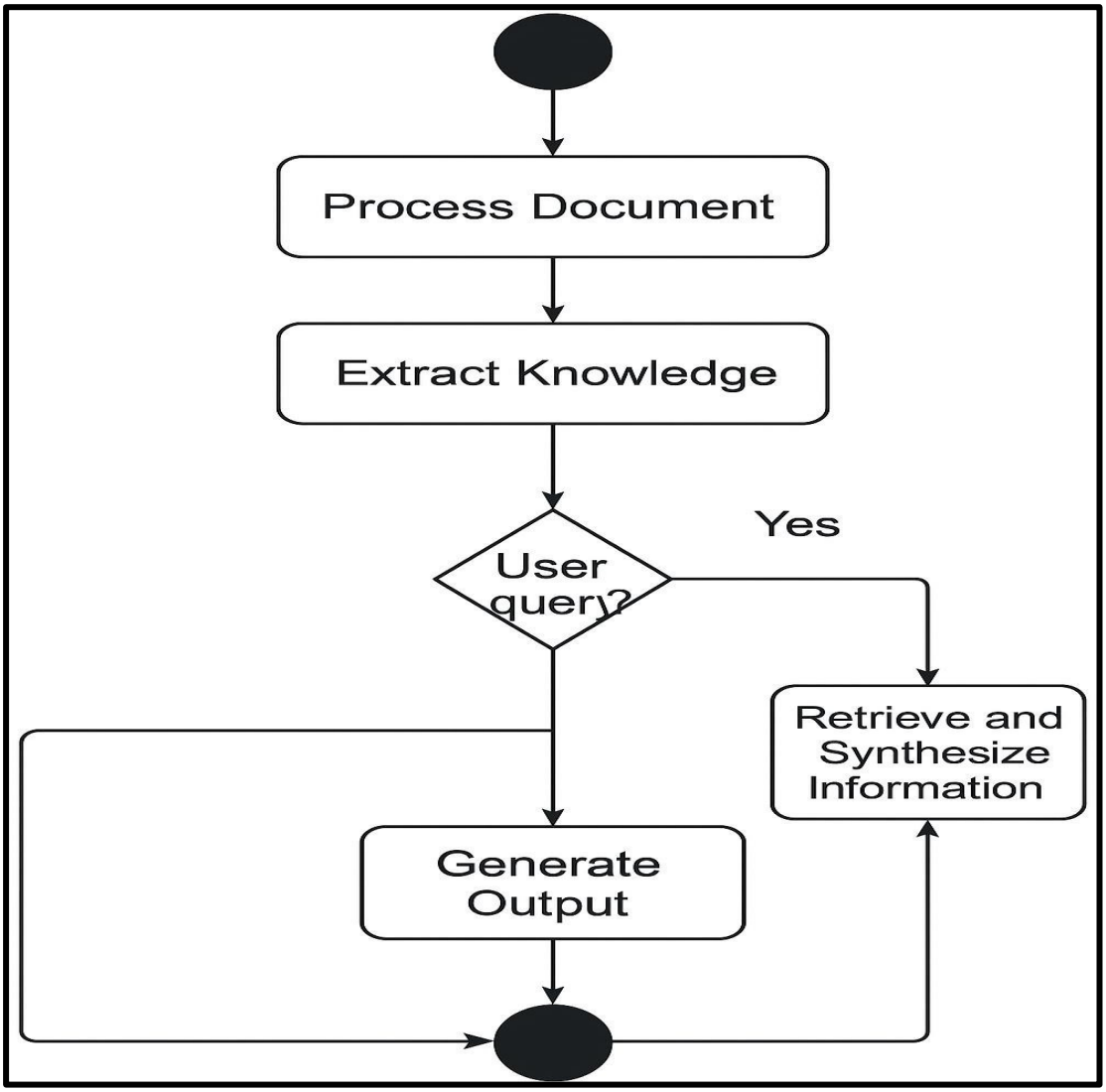

Figure 12 – Activity Diagram

23. Object Diagram

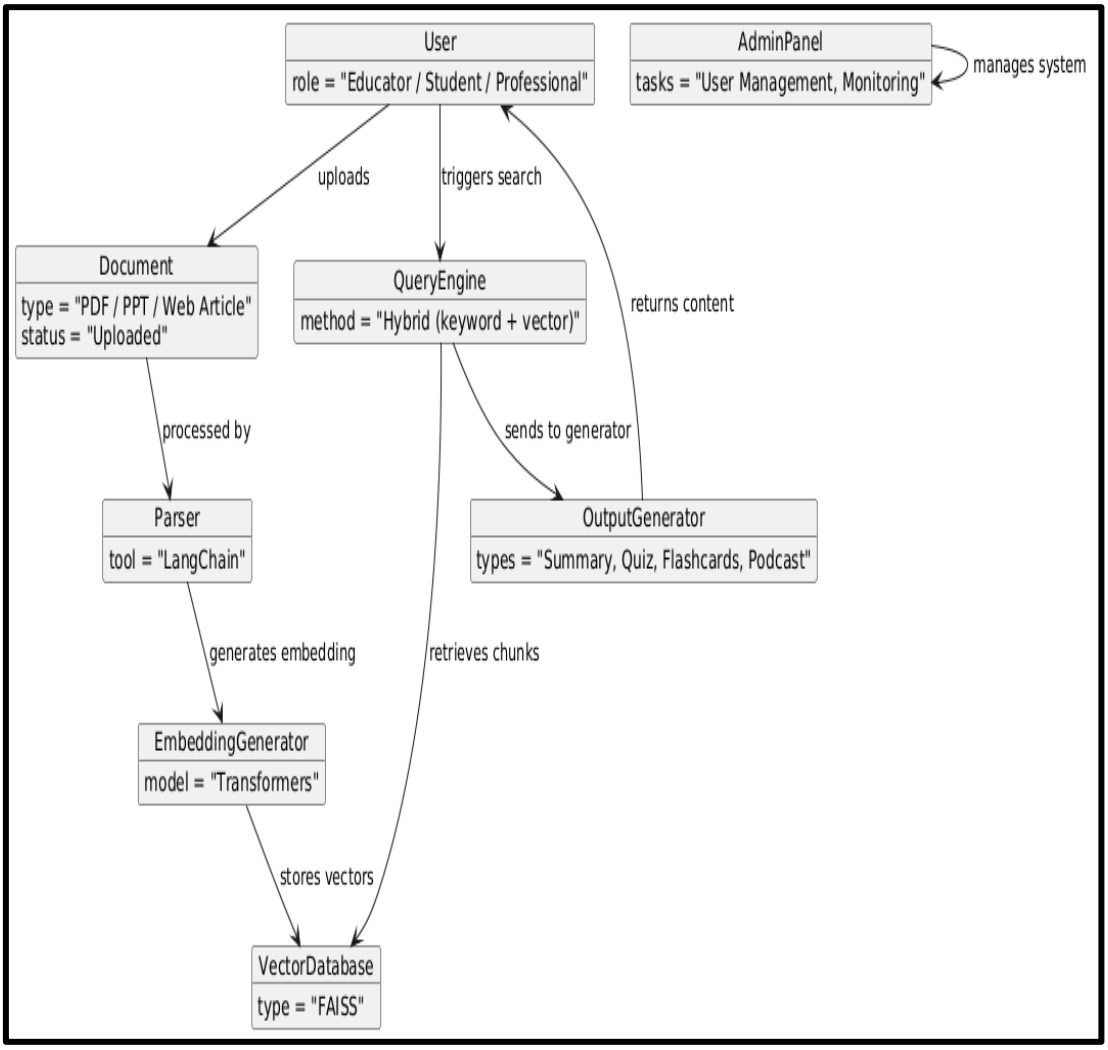

Figure 13 – Object Diagram

24. Component Diagram

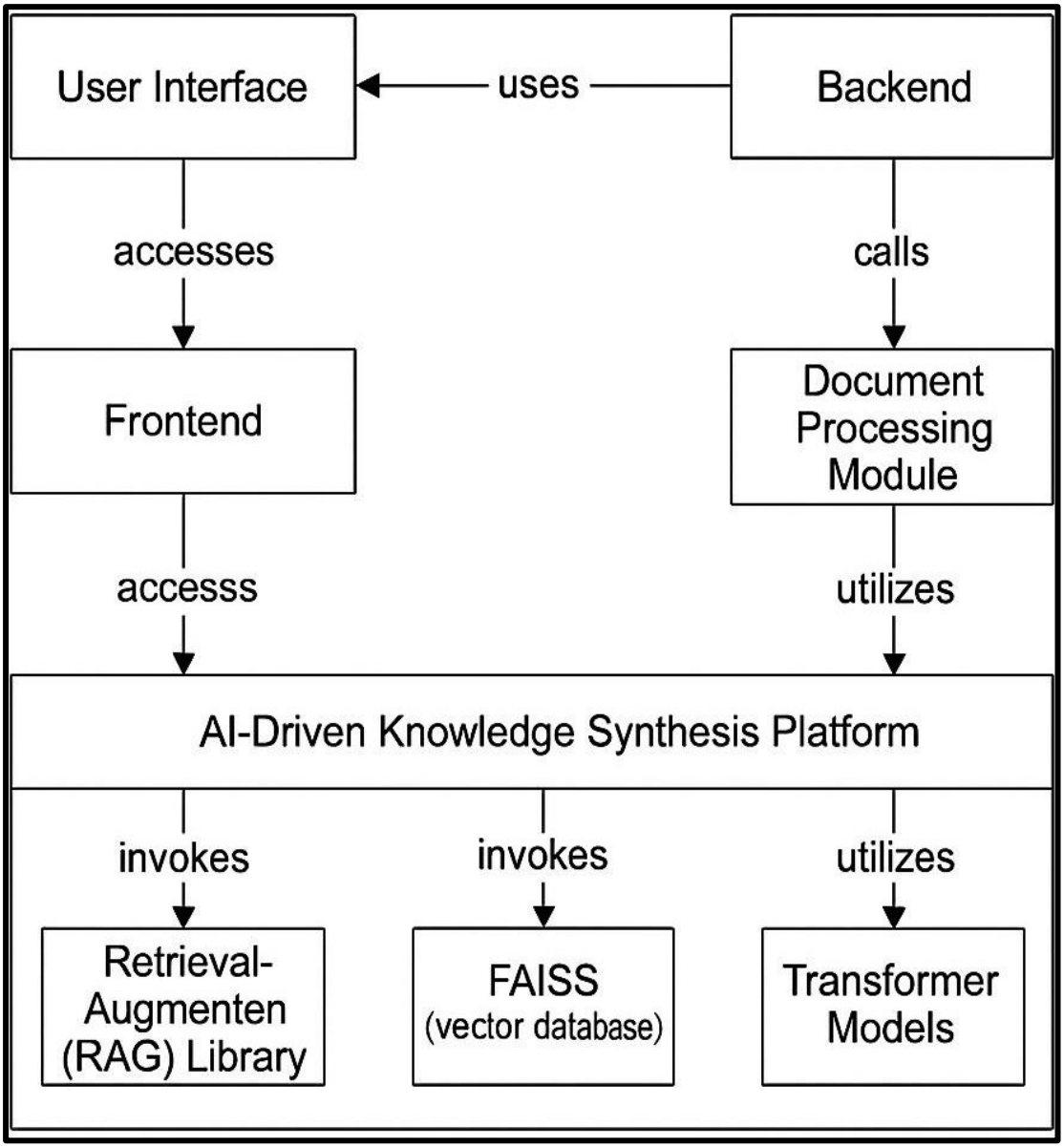

Figure 14 – Component Diagram

25. Deployment Diagram

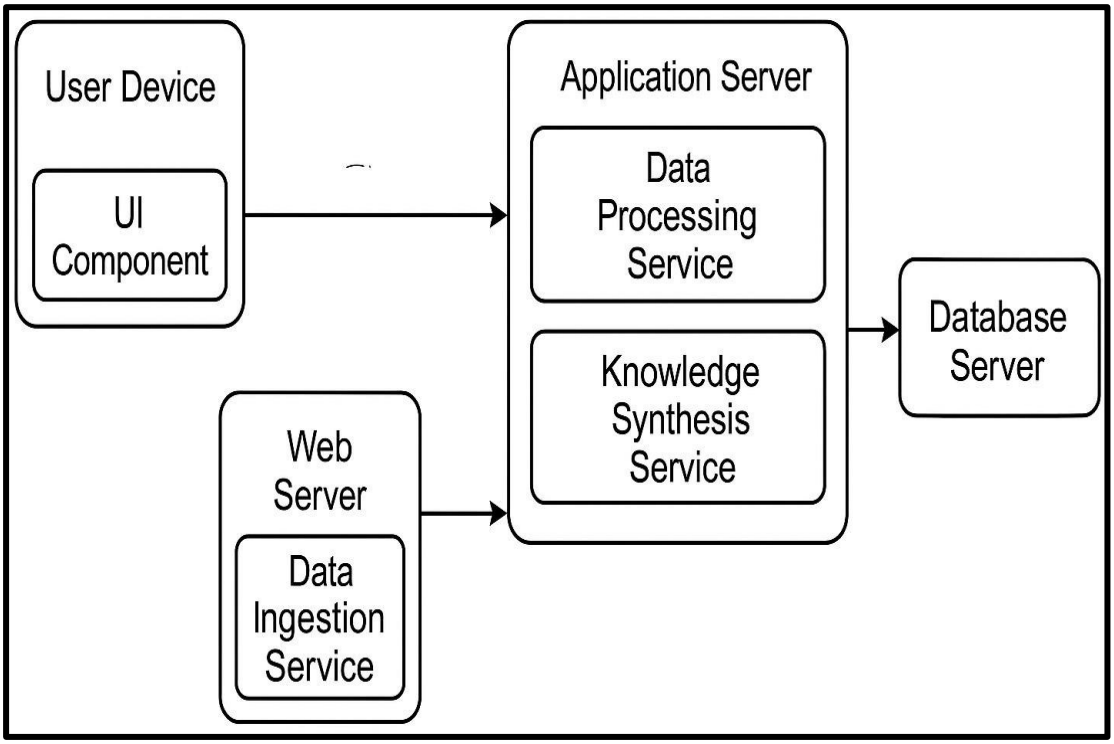

Figure 15 – Deployment Diagram

26. Testing

26.1. Unit Testing

- **DocumentParserTests:** Verify correct chunking of PDF and DOCX inputs.
- **EmbeddingServiceTests:** Validate embedding vector dimensions and reproducibility.

26.2. Functional Testing

- **End-to-End Tests:** Upload a sample research paper → request summary → verify summary contains key sections.
- **Performance Tests:** Ensure average query response < 3 seconds under 100 concurrent users.

CHAPTER – 6

CONCLUSION& FUTURE WORK

CHAPTER 6 – CONCLUSION & FUTURE WORK

27. Limitations of the Project

27.1. Dependency on Third-Party NLP Libraries

The platform relies heavily on open-source libraries such as Hugging Face Transformers, LangChain, and FAISS. While these tools offer powerful capabilities, they are subject to external updates and breaking changes. Any major version change in these dependencies may require substantial code refactoring or re-integration.

27.2. Scalability Bottlenecks in Vector Database

As the number of processed documents and corresponding vector embeddings increases, the performance of the FAISS vector database may degrade without the implementation of techniques like sharding or hierarchical indexing. In its current form, the system is optimized for moderate-scale use but may require architectural upgrades to handle enterprise-level data volumes.

28. Future Enhancements

To ensure the continued evolution, adaptability, and scalability of the platform, the following future enhancements are proposed:

28.1. Domain-Specific Model Fine-Tuning

Incorporating fine-tuned versions of large language models trained on specific domains (e.g., legal, medical, or academic texts) can significantly improve the accuracy and contextual relevance of the synthesized outputs. This will also allow the system to adapt better to industry-specific terminology.

28.2. Sharding and Replication for FAISS

To handle exponentially growing vector data, the system architecture should integrate sharding techniques across multiple FAISS instances and use replication strategies to ensure high availability and faster retrieval.

28.3. Support for Additional Output Modalities

The current system supports summaries, flashcards, quizzes, and podcasts. In future iterations, it can be extended to generate interactive learning content such as mind-maps, infographic slides, and AI-generated instructional videos, making it even more engaging for learners.

28.4. Administrative Dashboard with Analytics

A feature-rich analytics dashboard will be integrated to help administrators track system usage patterns, user engagement metrics, and content popularity. This will aid decision-making, resource allocation, and continuous improvement of the system.

CHAPTER – 7

BIBLIOGRAPHY & REFERENCES

CHAPTER 7 – BIBLIOGRAPHY & REFERENCES

29. BIBLIOGRAPHY & REFERENCES

[1] "Azure AI Search Documentation," 25 February 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/search/. [Accessed 10 March 2025].

[2] "FAISS GitHub Wiki," Meta AI Research, 24 February 2025. [Online]. Available: https://github.com/facebookresearch/faiss/wiki. [Accessed 10 March 2025].

[3] "Flask Documentation (3.1.x)," Pallets, 5 January 2025. [Online]. Available: https://flask.palletsprojects.com. [Accessed 10 March 2025].

[4] "LangChain Documentation," LangChain, Inc., 30 January 2025. [Online]. Available: https://python.langchain.com/docs/introduction/. [Accessed 10 March 2025].

[5] "LangChain HuggingFace Integrations Documentations," LangChain, Inc., 16 October 2024. [Online]. Available: https://python.langchain.com/docs/integrations/providers/huggingface/. [Accessed 10 March 2025].

[6] "LangChain Python API Reference," LangChain Inc., 2 March 2025. [Online]. Available: https://python.langchain.com/api_reference/. [Accessed 10 March 2025].

[7] "Gradio Documentation," Gradio, 9 March 2025. [Online]. Available: https://www.gradio.app/docs. [Accessed 10 March 2025].

[8] "Python 3.13.2 documentation," 10 March 2025. [Online]. Available: https://docs.python.org/3/. [Accessed 10 March 2025].

[9] "GitHub Docs," GitHub, Inc., [Online]. Available: https://docs.github.com/en. [Accessed 10 March 2025].

APPENDIX – 1

SNAPSHOTS

APPENDIX 1 – SNAPSHOTS

1. Home Page

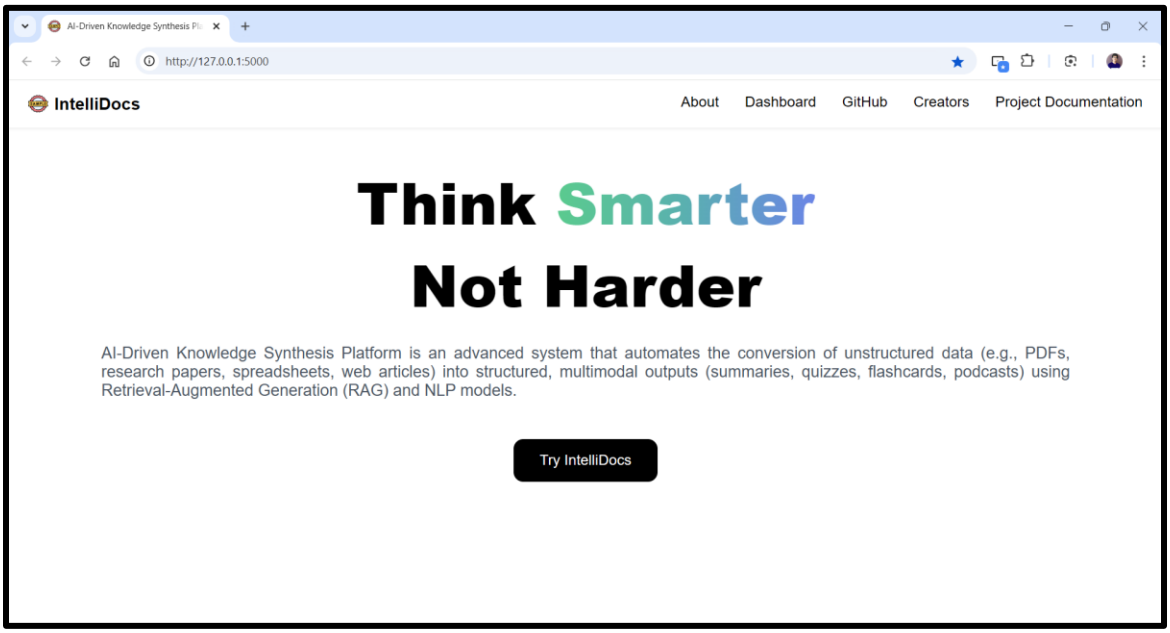

Figure 16 – Home Page

2. Dashboard (Project Working)

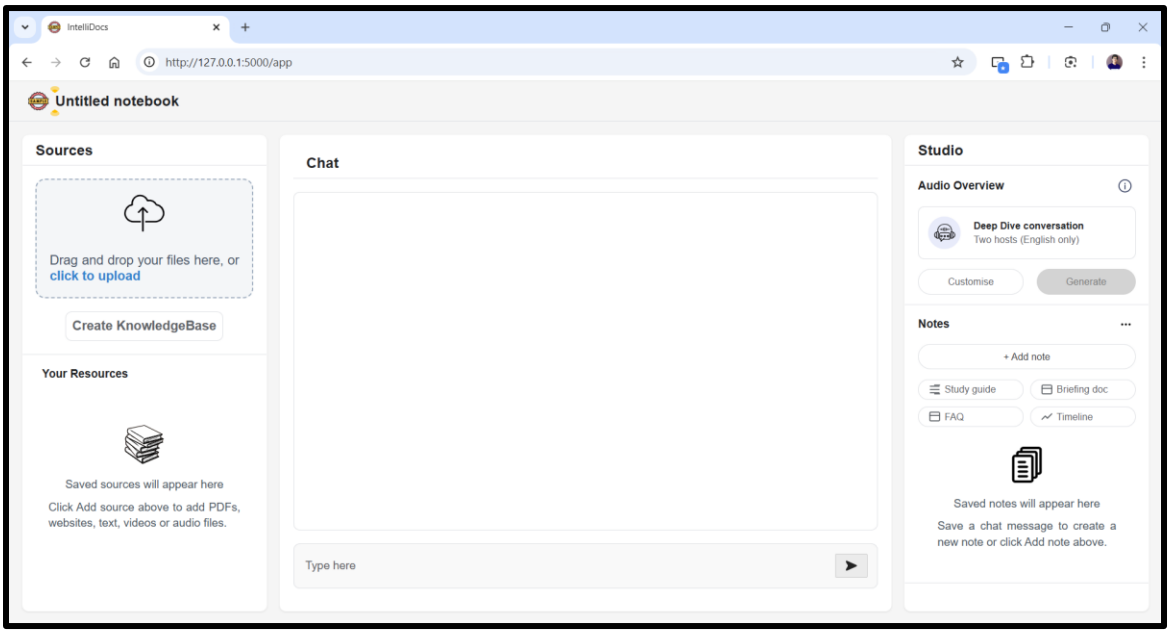

Figure 17 – Just after opening the Dashboard or the app workspace

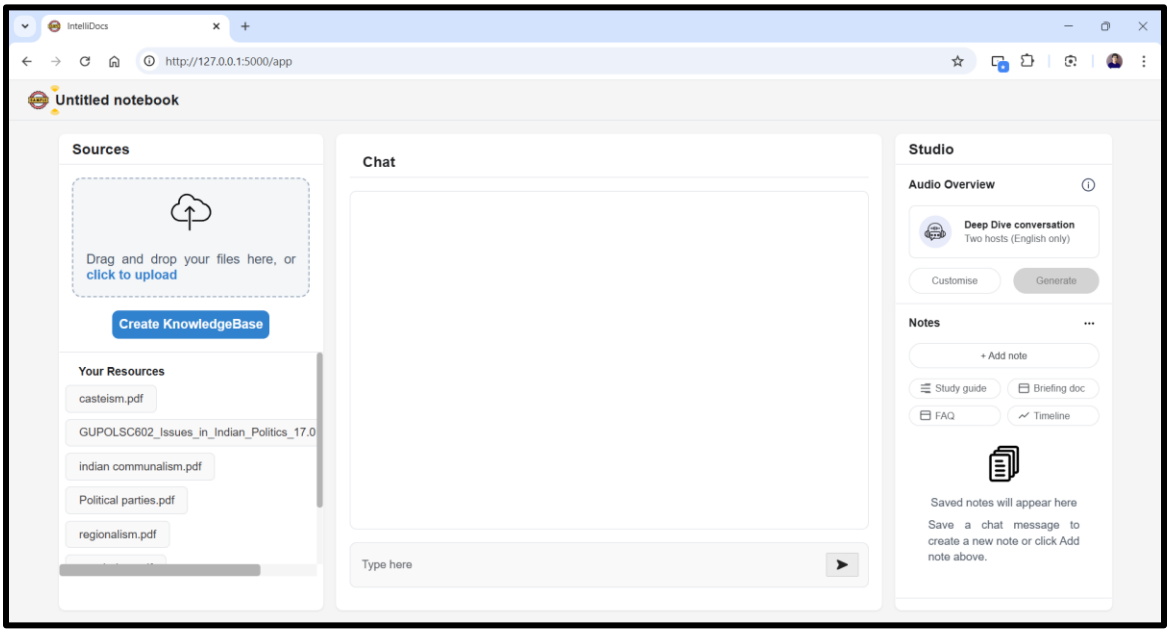

Figure 18 – After uploading the resources

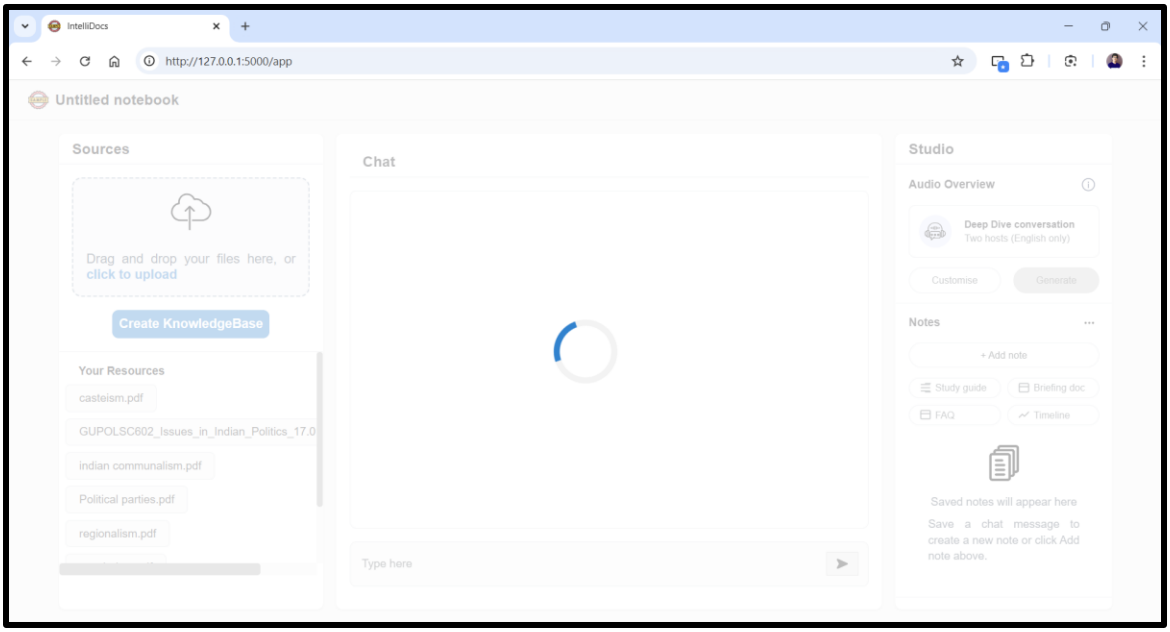

Figure 19 – KnowledgeBase is being created

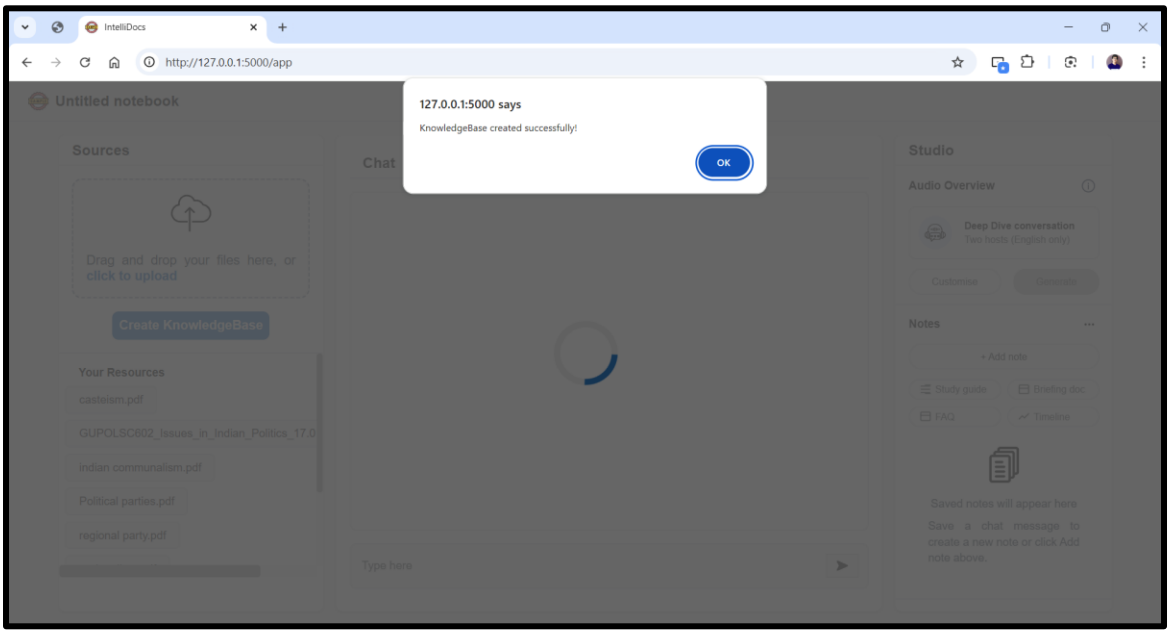

Figure 20 – An notification alert that the KnowledgeBase is successfully created

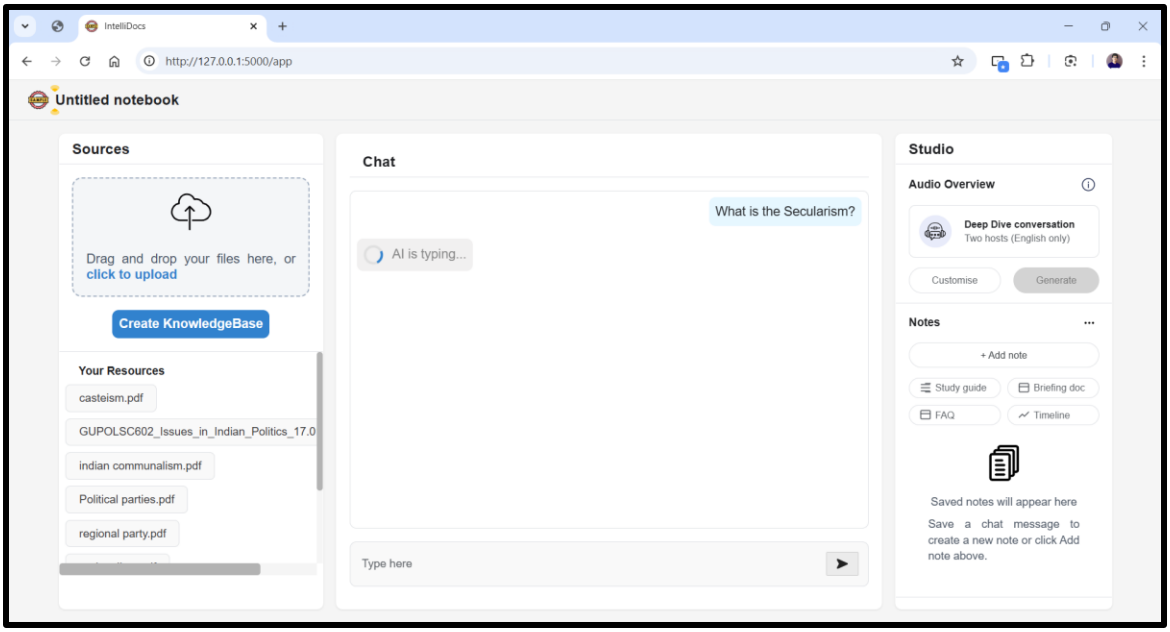

Figure 21 – Conversing with the AI

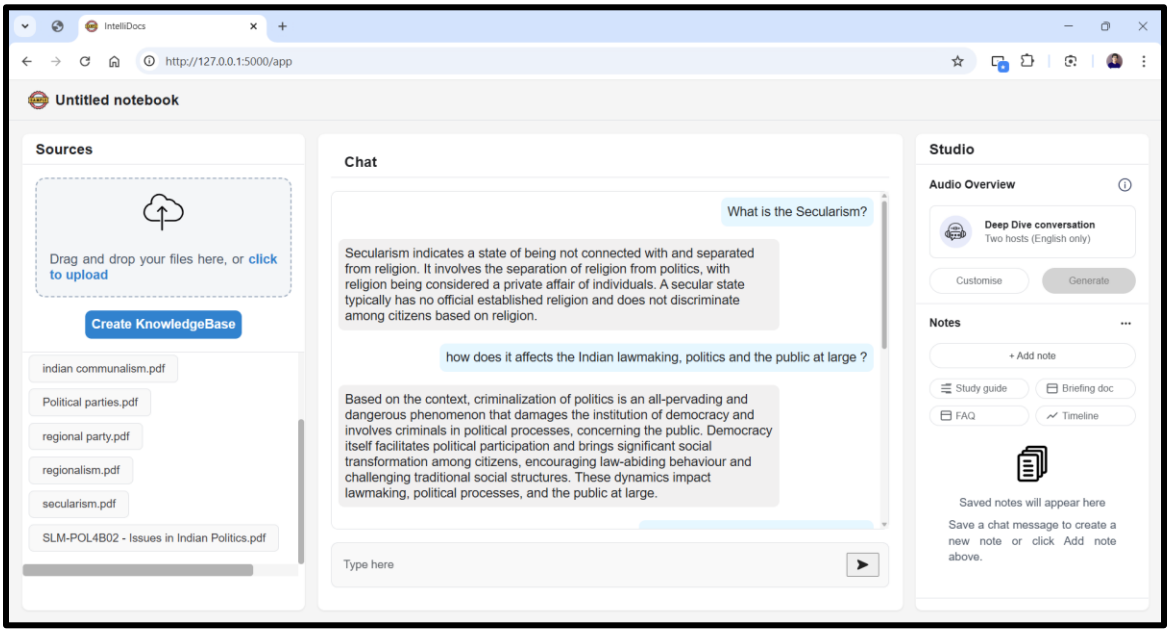

Figure 22 - Conversing further with the AI

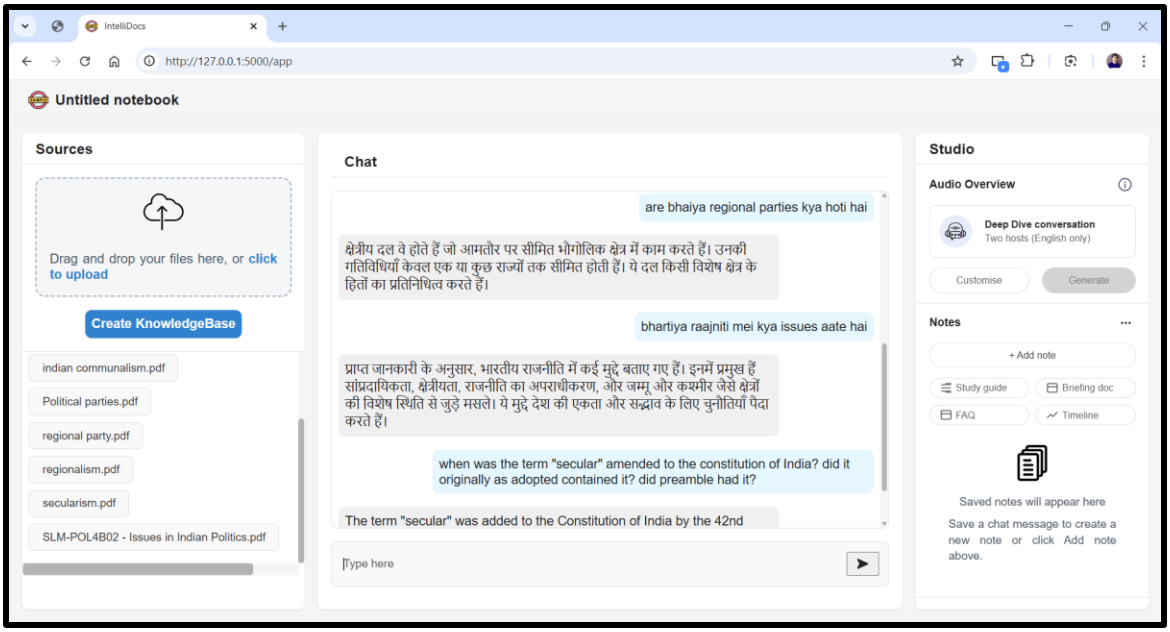

Figure 23 – Conversing further with AI and getting response in Hindi language and Devanagari script

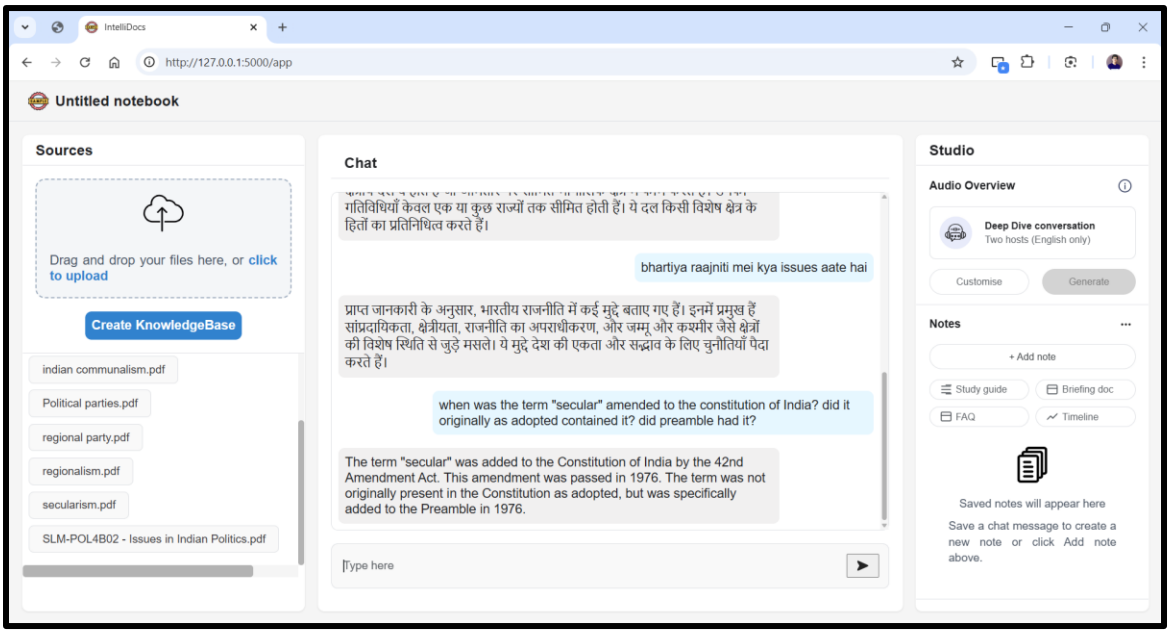

Figure 24 – Conversing further with AI

3. GitHub Repository

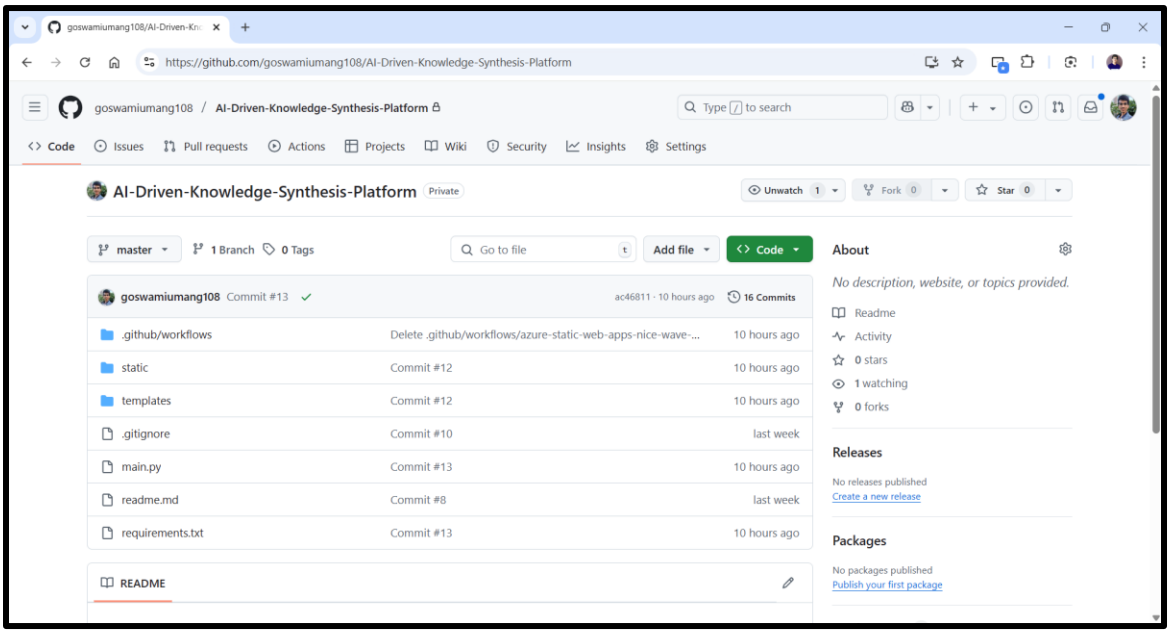

Figure 25 – The GitHub Repository of our project

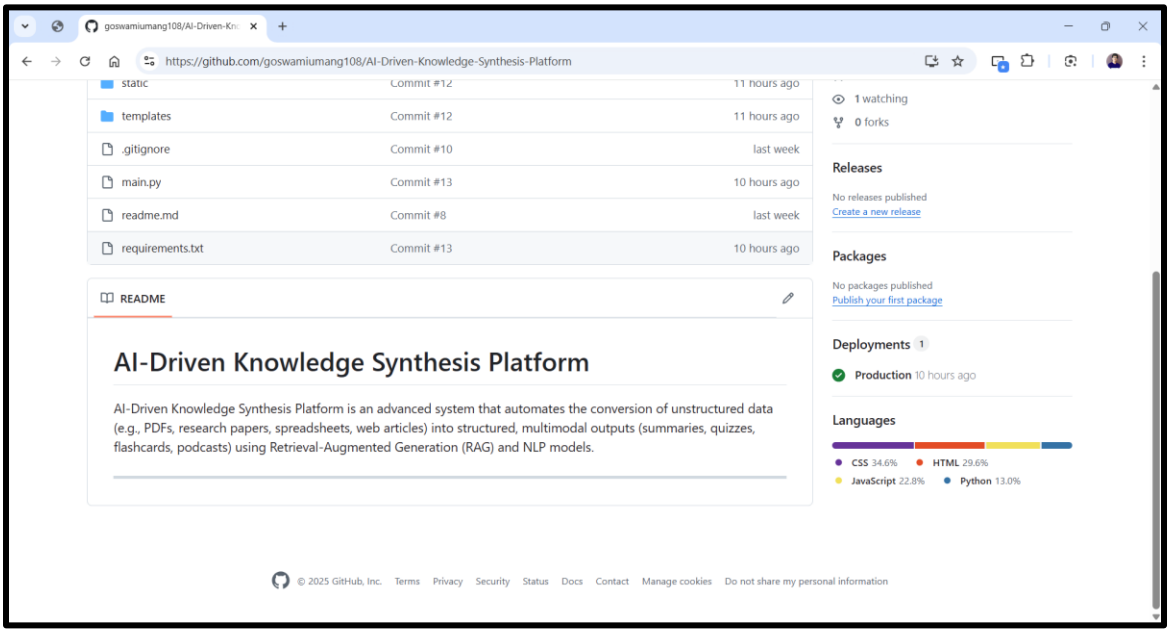

Figure 26 – The GitHub Repository of our page

4. Documentation

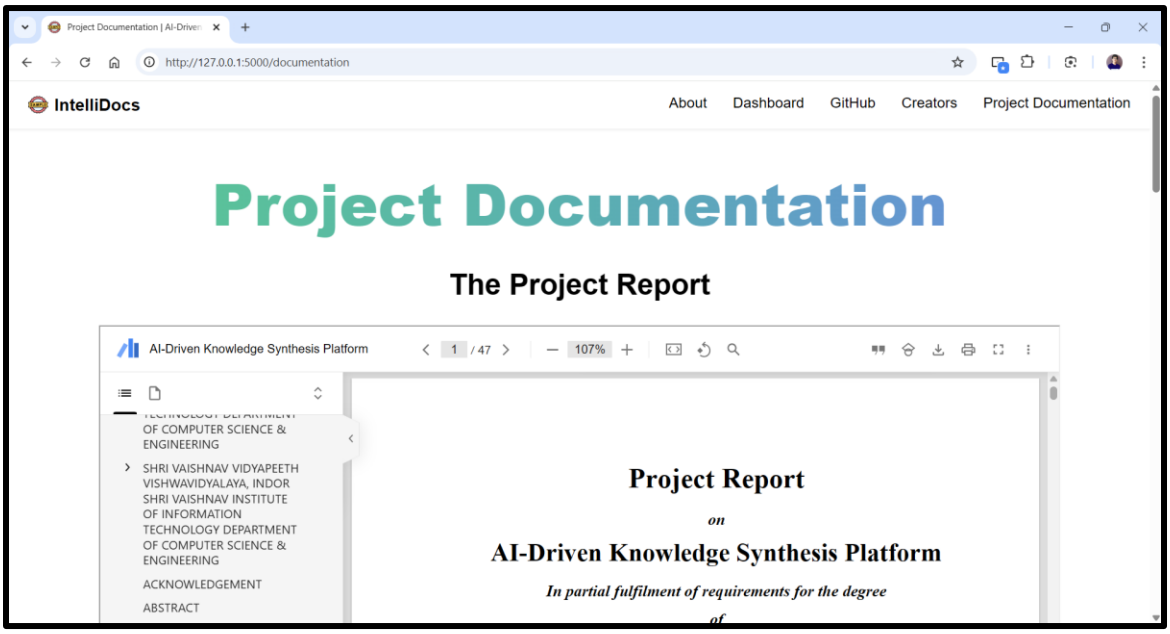

Figure 27 – The Project Documentation page showing Project Report

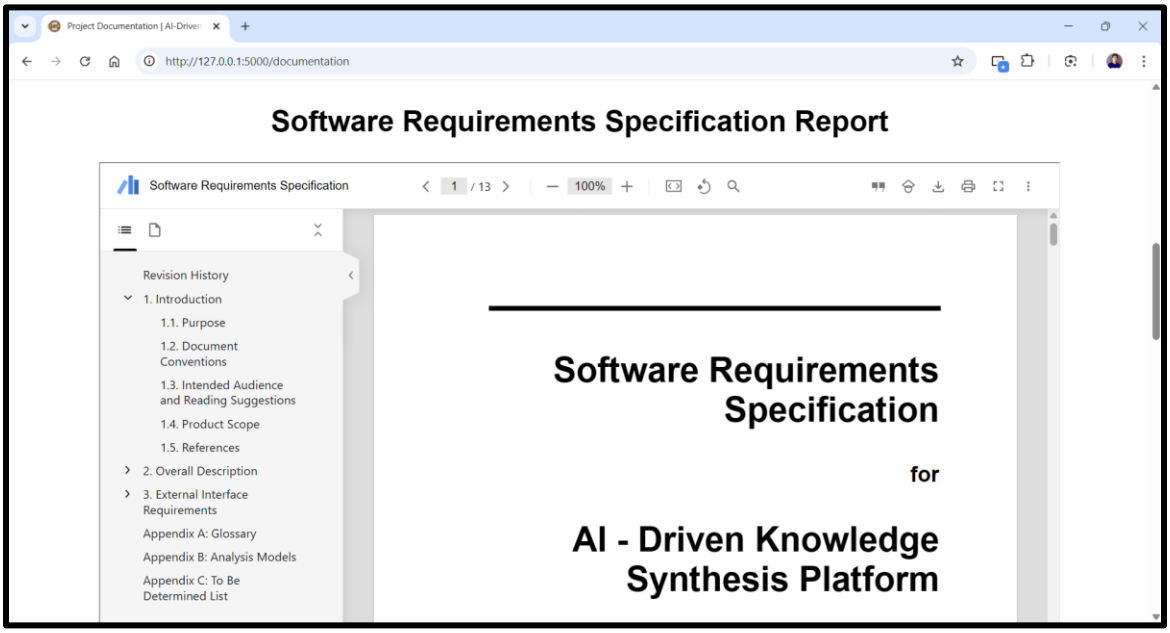

Figure 28 – The Project Documentation page showing SRS Report

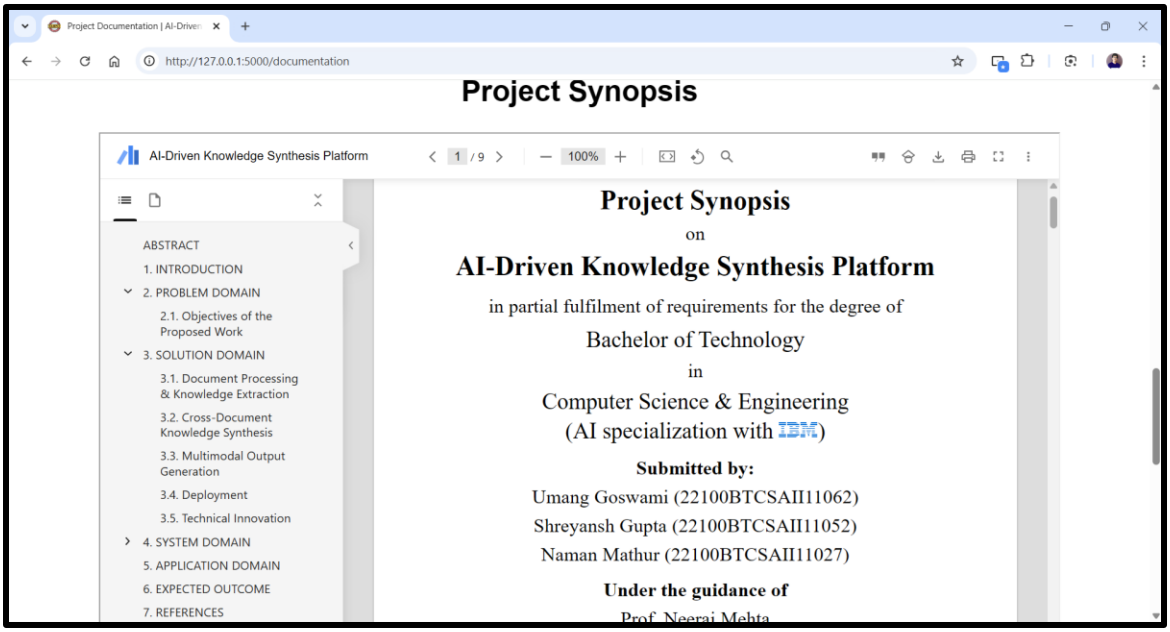

Figure 29 – The Project Documentation page showing the Project Synopsis

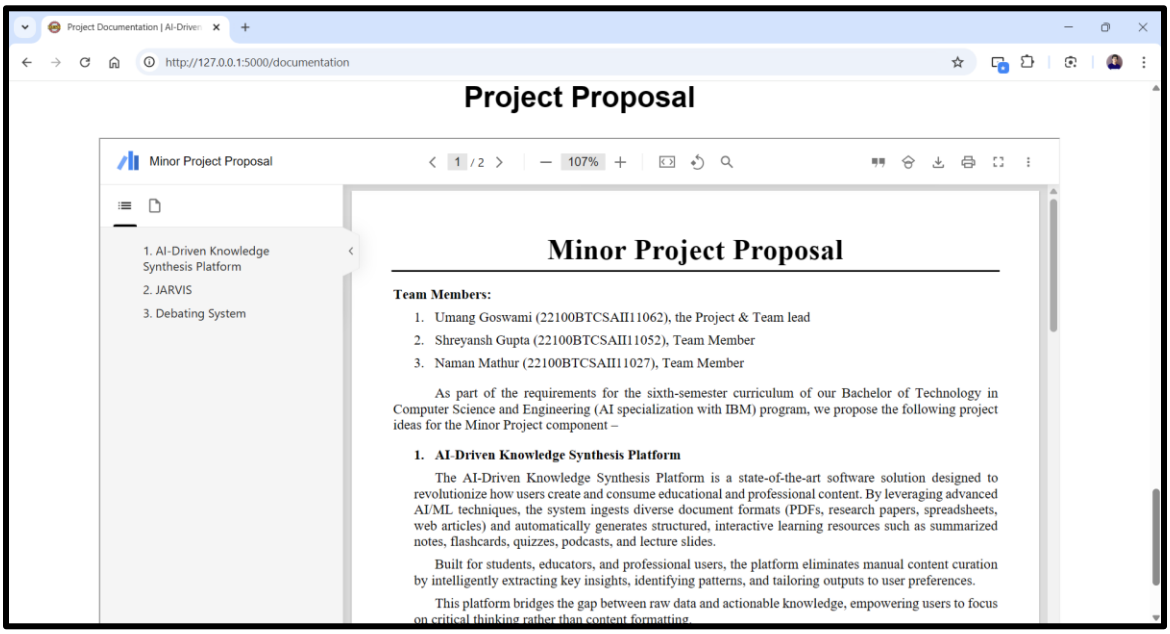

Figure 30 – The Project Documentation page showing the Project Proposal