# Real-time/Field-Based Research Project Report On

**GenAI Based Qa\_Bot**

A dissertation submitted to the Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of the requirement for the award of a degree of

## BACHELOR OF TECHNOLOGY IN

**COMPUTER SCIENCE AND BUSINESS SYSTEM**

Submitted by

**M.Praneeth (23B81A3232)**

**K.Sreejha (23B81A3249)**

**J.Sindhu (23B81A3247)**

****

Department of Computer Science and Engineering

CVR COLLEGE OF ENGINEERING

(An UGC Autonomous Institution, Affiliated to JNTUH, Accredited by NBA, and NAAC) Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), Ranga Reddy (Dist.) - 501510,

Telangana State.

**2024-25**

# CVR COLLEGE OF ENGINEERING

*(*An UGC Autonomous Institution, Affiliated to JNTUH, Accredited by NBA, and NAAC)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M)

Ranga Reddy (Dist.) - 501510, Telangana State.

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CERTIFICATE**

This is to certify that the project work entitled **“GenAI Based Qa\_Bot”** is being submitted by M.Praneeth (23B81A3232), K.Sreejha (23B81A3249), J.Sindhu (23B81A3247) in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Business System,** during the academic year 2024-2025.

**Professor-in-charge RFP Professor and Head, CSE (Dr. A. Vani Vathsala)**

**DECLARATION**

We hereby declare that this project report titled **“GenAI Based Qa\_Bot”** submitted to the Department of Computer Science and Engineering, CVR College of Engineering, is a record of original work done by me. The information and data given in the report is authentic to the best of my knowledge. This Real Time/Field-Based Research Project report is not submitted to any other university or institution for the award of any degree or diploma or published at any time before.

**M.Praneeth (23B81A3232)**

**K.Sreejha (23B81A3249)**

**J.Sindhu (23B81A3247)**

Date: Place:

**ABSTRACT**

In an era where digital documents are rapidly increasing in volume and complexity, manually

extracting key information from lengthy PDFs can be a time-consuming and inefficient process.

QA\_Bot is an AI-powered solution designed to automate this task by summarizing documents

and providing intelligent Q&A capabilities.

Built using Streamlit for an intuitive web-based interface and LangChain for advanced natural

language understanding, QA\_Bot leverages Large Language Models (LLMs) to process PDFs

efficiently. The system extracts essential insights, generates concise summaries, and enables

users to ask questions directly related to the document content, providing contextually accurate

responses.

QA\_Bot is particularly beneficial for professionals handling large volumes of text, including

researchers, legal professionals, financial analysts, business strategists, and technical experts. By

transforming static PDFs into dynamic, interactive content, QA\_Bot significantly enhances

productivity and decision-making by reducing the time required for document review.

By integrating state-of-the-art NLP and AI technologies, QA\_Bot represents a step forward in

intelligent document analysis. It aims to bridge the gap between raw text data and actionable

insights, making complex document processing faster, smarter, and more efficient.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | Page No. |
|  |  | Abbreviations |  |
|  |  | Symbols |  |
| 1 |  | **INTRODUCTION** | 1 |
|  | 1.1 | Motivation |  |
|  | 1.2 | Problem Statement |  |
|  | 1.3 | Project Objectives |  |
|  | 1.4 | Project Report Organization |  |
| 2 |  | **LITERATURE REVIEW** |  |
|  | 2.1 | Existing Work |  |
|  | 2.2 | Limitations of Existing Work |  |
| 3 |  | **REQUIREMENT ANALYSIS** |  |
|  | 3.1 | Software requirements |  |
|  | 3.2 | Hardware requirements |  |
|  | 3.3 | User requirements |  |
| 4 |  | **SYSTEM DESIGN** |  |
|  | 4.0 | **Proposed System architecture** |  |
|  | 4.1 | Proposed Methods/ Algorithms |  |
|  | 4.2 | Class / Use Case / Activity/ Sequence Diagrams |  |
|  | 4.3 | Datasets and Technology stack |  |
| 5 |  | **IMPLEMENTATION** |  |
|  | 5.1 | Front page Screenshot |  |
|  | 5.2 | Results and Discussions |  |
|  | 5.3 | Testing |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.4 | Validation |  |
| 6 |  | **CONCLUSION** |  |
|  | 6.1 | Conclusion |  |
|  | 6.2 | Future scope |  |
|  |  | **REFERENCES** |  |
|  |  | **APPENDIX: (If any like Published paper / source code)** |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 Motivation**

In today’s digital era, vast amounts of information are stored in PDFs, including research papers, legal contracts, financial reports, and technical documentation. Manually extracting relevant information from lengthy documents is time-consuming, inefficient, and prone to human error. With advancements in Artificial Intelligence (AI) and Natural Language Processing (NLP), automated solutions can help users quickly summarize, retrieve, and understand key insights from PDFs.

The motivation behind this project is to develop a QA\_Bot, an AI-powered tool that enables users to summarize PDF documents and ask questions based on their content. This solution improves productivity by eliminating the need for manual document scanning and enhances accessibility by providing instant, context-aware answers.

By integrating LangChain for text comprehension and Streamlit for a user-friendly web interface, this project aims to bridge the gap between static documents and interactive knowledge retrieval, benefiting professionals in various domains such as research, business, law, and finance.Problem Statement

**1.2 Problem Statement**

The increasing reliance on PDFs for document storage presents challenges in efficiently extracting key information from large volumes of text. Existing search mechanisms like Ctrl+F are insufficient when dealing with complex queries that require contextual understanding.

Key issues include:

Manual Document Review is Inefficient – Users spend significant time reading lengthy reports to find relevant information.

Lack of Interactive Search – Traditional keyword-based searches do not provide meaningful insights or direct answers.

Limited Summarization Capabilities – Users often need concise overviews of documents without reading every detail.

Contextual Understanding is Missing – Extracting meaningful answers from PDFs requires advanced NLP techniques.

To address these challenges, this project proposes an AI-powered QA\_Bot that allows users to upload PDFs, receive automated summaries, and ask context-aware questions to extract relevant information dynamically.

**1.3 Project Objectives**

The primary objectives of the QA\_Bot project are:

* Automated Document Summarization – Develop an AI-powered system that extracts and summarizes key insights from PDFs.
* Context-Aware Question Answering – Enable users to ask questions about a document and receive accurate, AI-generated responses.
* User-Friendly Web Interface – Utilize Streamlit to build an interactive and accessible platform for document analysis.
* Efficient Text Processing – Leverage LangChain and LLMs (Large Language Models) to improve document comprehension.
* Scalability & Adaptability – Design a flexible system that can be applied across multiple domains, including research, law, finance, and business.
* Real-Time Processing – Optimize response time for a seamless user experience while maintaining accuracy

**1.4 Project Report Organization**

This report is structured as follows:

**Chapter 1:** Introduction – Outlines the motivation, problem statement, objectives, and organization of the report.

**Chapter 2:** Literature Review – Discusses existing solutions for document summarization and Q&A systems, highlighting the need for AI-driven automation.

**Chapter 3:** System Design & Architecture – Details the implementation of QA\_Bot, including the use of LangChain, OpenAI LLMs, and Streamlit.

**Chapter 4:** Implementation & Features – Describes the core functionalities, user interface, and workflow of the system.

**Chapter 5:** Testing & Evaluation – Presents performance metrics, accuracy analysis, and real-world testing scenarios.

**Chapter 6:** Conclusion & Future Scope – Summarizes the findings and discusses potential improvements and extensions for QA\_Bot.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 Existing Work**

Several AI-driven document summarization and question-answering (QA) systems have been developed, utilizing Natural Language Processing (NLP) and Machine Learning (ML). Some notable approaches include:

🔹 Text Summarization Techniques:

* Extractive Summarization (e.g., TextRank, LexRank): Selects key sentences from the document without altering the original text.
* Abstractive Summarization (e.g., BART, T5): Generates new sentences based on document understanding, improving conciseness and readability

.

🔹 Question-Answering Systems:

* Traditional Rule-Based QA: Uses keyword matching and predefined rules but lacks flexibility.
* Machine Learning-Based QA: Includes models like BERT, GPT, and T5, trained on large text datasets for contextual understanding.
* RAG (Retrieval-Augmented Generation): Combines document retrieval and language generation to improve accuracy, used in LangChain-powered systems.

🔹 PDF Processing Tools:

* PDF.js and PyPDF for extracting text.
* OCR-based solutions (e.g., Tesseract) for scanned PDFs.
* Semantic search frameworks (e.g., FAISS, ChromaDB) for efficient document retrieval.

**2.2 Limitations of Existing Work**

Despite advancements, current solutions have limitations:

Lack of Deep Context Understanding – Most existing QA systems struggle with multi-turn reasoning and complex document structures (tables, charts, handwritten text).

Limited Customization & Domain-Specific Adaptation – Many models provide generalized answers but lack custom training for legal, financial, or research documents.

Inefficient Handling of Large Documents – Standard summarization tools struggle with long PDFs, leading to truncated outputs or loss of important details.

Slow Response Time & High Computational Cost – Large AI models, like GPT-based systems, require significant processing power, making real-time responses challenging.

Poor User Interaction & Searchability – Many solutions do not offer an interactive, user-friendly interface, making document exploration difficult.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

**3.1 Software Requirements**

The primary software requirement for this project is Python as the programming language.

Python is a versatile and object-oriented language well-suited for game development. It offers features like libraries for graphics rendering, user input handling**.**

**3.2 Hardware Requirements**

The hardware requirements depend on the deployment type (local vs. cloud-based).

🔹 Minimum System Requirements (Local Deployment)

* Processor: Intel Core i5 / AMD Ryzen 5 or higher
* RAM: 8GB (16GB recommended for large documents)
* Storage: 10GB free space (for dependencies & temp files)
* GPU: Optional, but recommended for faster AI inference (NVIDIA GPU with CUDA support)

**3.3 User Requirements**

**Functional Requirements**

* PDF Upload – Users should be able to upload PDFs via the Streamlit UI.
* Text Extraction – The system should accurately extract text from digital and scanned PDFs

.

* Summarization – Users should receive a concise summary of the uploaded document.
* Question Answering – Users should ask questions and receive accurate responses based on the document.
* Search Functionality – Users should search for specific keywords or sections within the document.
* Answer Display – Responses should be shown in a user-friendly format with highlighted references.

**Non-Functional Requirements**

🔹 Performance – The bot should generate summaries and answers quickly.

🔹 Scalability – Should support large PDFs without significant performance issues.

🔹 Accuracy – The AI model should provide relevant and precise answers.

🔹 Security – User-uploaded documents should be handled securely, with no unauthorized

access.

🔹 User-Friendly Interface – The system should be intuitive and easy to navigate.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.0 Proposed System Architecture**

**Overview**

The QA\_Bot system follows a modular approach, including:

* Frontend (Streamlit UI) – Users upload PDFs and ask questions.
* Backend (Processing Layer) – Extracts text, generates summaries, and answers queries.
* Storage & Retrieval – Uses FAISS for fast document search.

**Key Components**

* PDF Processing – Uses PyPDF2 for text extraction.
* Summarization – AI models summarize document content.
* Vector Embeddings – Converts text into searchable representations.
* Question-Answering – GPT-4 retrieves and generates answers.
* Deployment – Supports both local and cloud-based execution.

**Workflow**

* User uploads a PDF via Streamlit.
* Text is extracted and summarized.
* Embeddings are stored for fast retrieval.
* User asks a question → AI retrieves relevant text.
* LLM generates an answer and displays it.

**Key Benefits**

✔ Fast & accurate document search.

✔ Scalable & efficient processing.

✔ User-friendly interactive interface**.**

**4.1 Proposed Methods**

The QA\_Bot system follows a structured approach for PDF summarization and question answering using AI and NLP techniques. The proposed methods ensure efficient document processing, retrieval, and response generation.

**Text Extraction from PDFs**

* Uses PyPDF for extracting text from digital PDFs.
* Cleans and preprocesses extracted text (removing noise, special characters, etc.).

**Document Summarization**

* Uses LangChain-based summarization (Abstractive & Extractive).
* LLM-powered models (e.g., OpenAI GPT) generate concise summaries.
* Summaries are stored for quick retrieval.

**Embedding Generation & Storage**

Converts text into vector embeddings using OpenAI Embeddings ,Sentence Transformers.

Stores embeddings in FAISS for efficient semantic search.

**Question Answering (QA) System**

* User enters a query in the Streamlit interface.
* Retrieval-Augmented Generation (RAG) retrieves the most relevant text sections.
* LLM(GPT 4o mini) processes the retrieved content and generates an answer.
* The system ensures context-aware, precise responses.

**User Interface & Interaction**

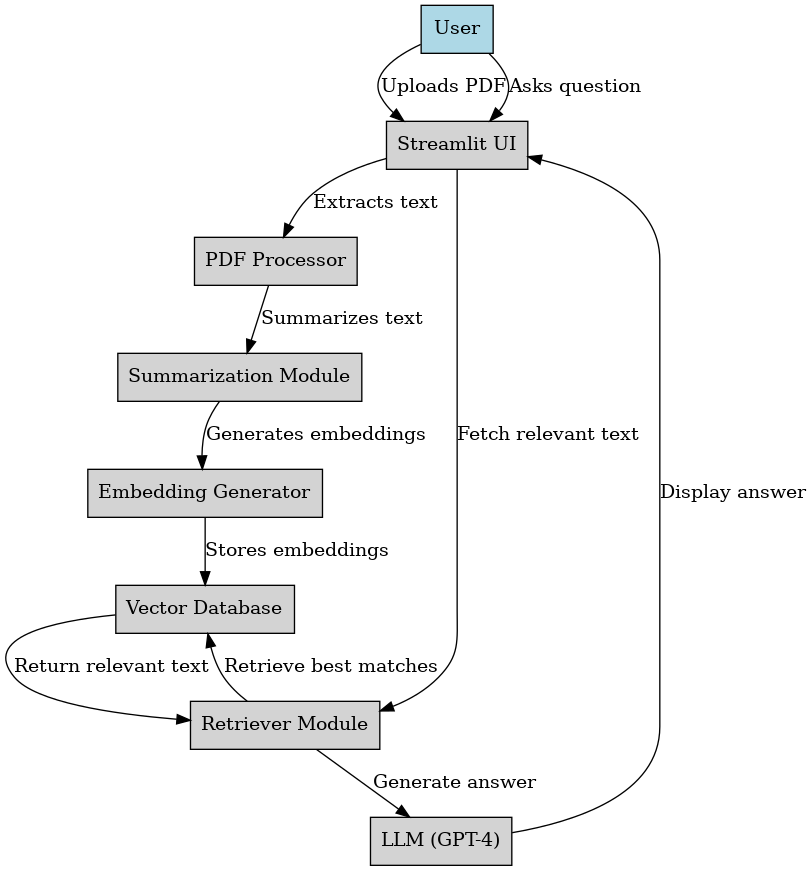
* Streamlit-based Web UI for PDF upload, summarization, and Q&A.
* Displays summaries and AI-generated answers in real time.

**Deployment & Scalability**

Supports local deployment (Python & Streamlit) and cloud-based hosting (AWS, GCP, Azure).

Can integrate database storage for saving previous interactions.

**4.2 Sequence Diagram**



**4.3 Technology Stack**

**Frontend (User Interface)**

🔹 Streamlit – Interactive web-based UI for uploading PDFs and asking questions.

🔹 HTML / CSS (via Streamlit components) – Basic styling for the user interface.

**Backend (Processing & AI Engine)**

🔹 Python – Primary programming language for AI and data processing.

🔹 FastAPI (Optional) – For serving the AI models via an API.

**Document Processing**

🔹 PyPDF – Extracts text from digital PDFs.

**Summarization & NLP Models**

🔹 LangChain – Framework for connecting LLMs with document processing.

🔹 GPT-4 / OpenAI API – AI model for text summarization and Q&A.

🔹 Hugging Face Models – Alternative summarization models.

**Embedding & Vector Search**

🔹 OpenAI Embeddings / Sentence Transformers – Converts text into vector representations.

🔹 FAISS – Vector database for fast document retrieval.

**Question Answering (RAG)**

🔹 Retrieval-Augmented Generation (RAG) – Ensures accurate answers using retrieved document context.

🔹 LLMs (GPT-4o mini) – AI models for generating responses.

**Storage & Database**

🔹 FAISS – Vector database for document embeddings.

**Deployment & Hosting**

🔹 Local Deployment – Runs on Python + Streamlit locally.

🔹 Cloud Deployment – Hosted on AWS, GCP, or Azure for scalability.

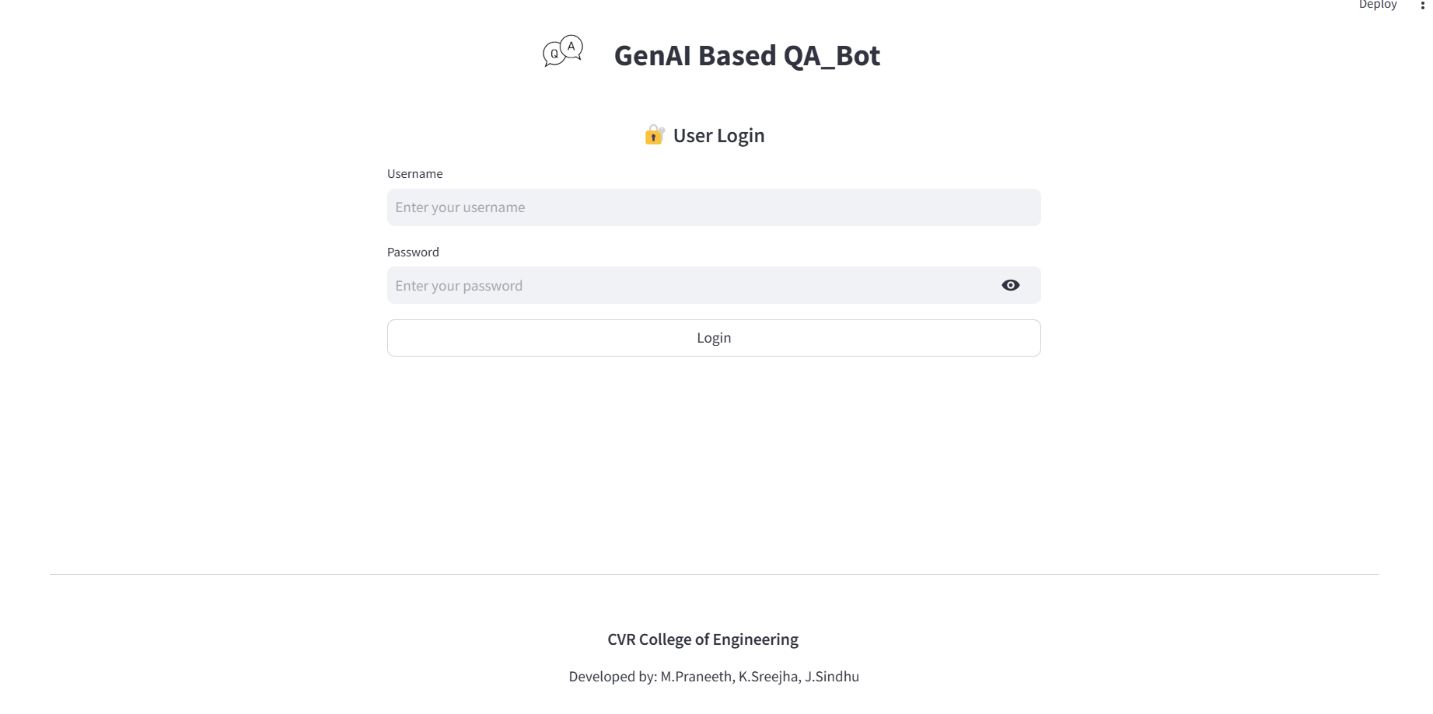
🔹 Docker – Containerization for consistent deployment.

**CHAPTER 5**

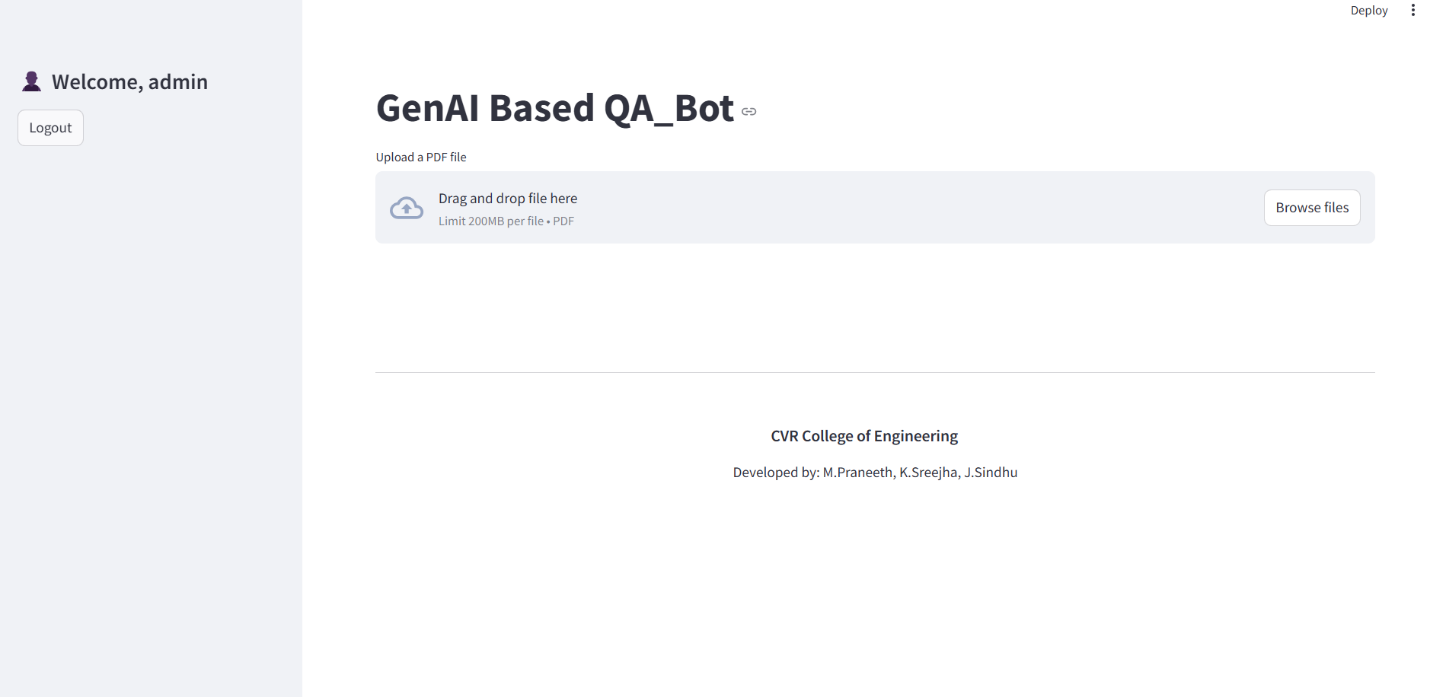
**IMPLEMENTATION**

**5.1 Login & Main Interface**

**Login Page**



**Main Page**



### **5.2 Results and Discussions**

Upon successful login and PDF upload, the user is guided through several intelligent processing steps powered by GPT-4 and semantic search. The results of each step are displayed dynamically:

#### **Topic Extraction**

* GPT extracts major themes or headings from the document.
* This helps users understand the structure of content at a glance.
* Example Output:

**Topics:** Neural Networks, Training Algorithms, Activation Functions, Deep Learning Applications

**Structured Summary**

* The entire document is summarized using GPT-4 into a well-organized narrative.
* The summary is concise, coherent, and preserves technical details.
* Example Output:

**Summary:** This paper introduces the fundamentals of neural networks including perceptrons, activation mechanisms, and the backpropagation algorithm...

#### **AI Voice Summary (Text-to-Speech)**

* A voice summary is generated using pyttsx3.
* Users can listen to the audio using a built-in HTML5 audio player.
* This enhances accessibility for users who prefer auditory information consumption.

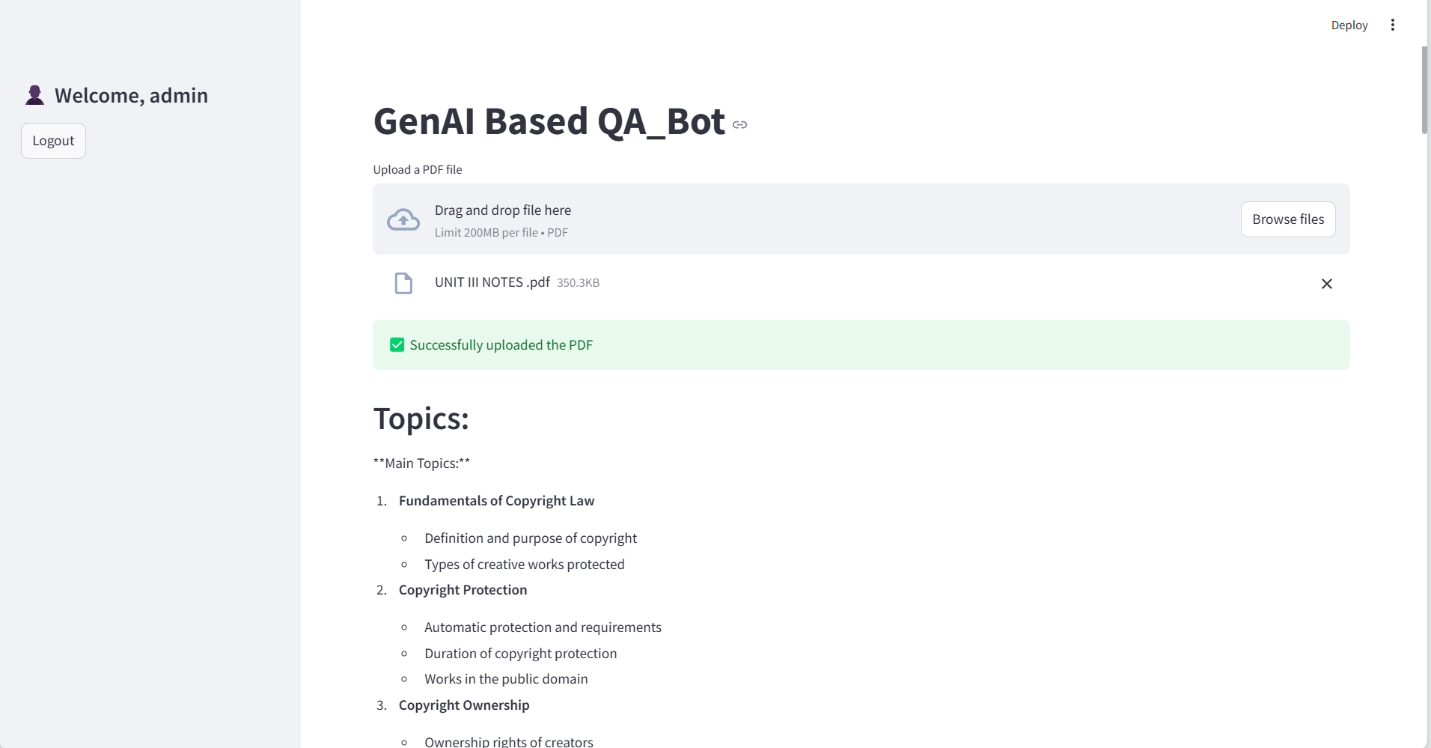
#### **Question Answering**

* Users can input any question related to the uploaded PDF.
* GPT-4 generates context-aware answers using the full text as context.

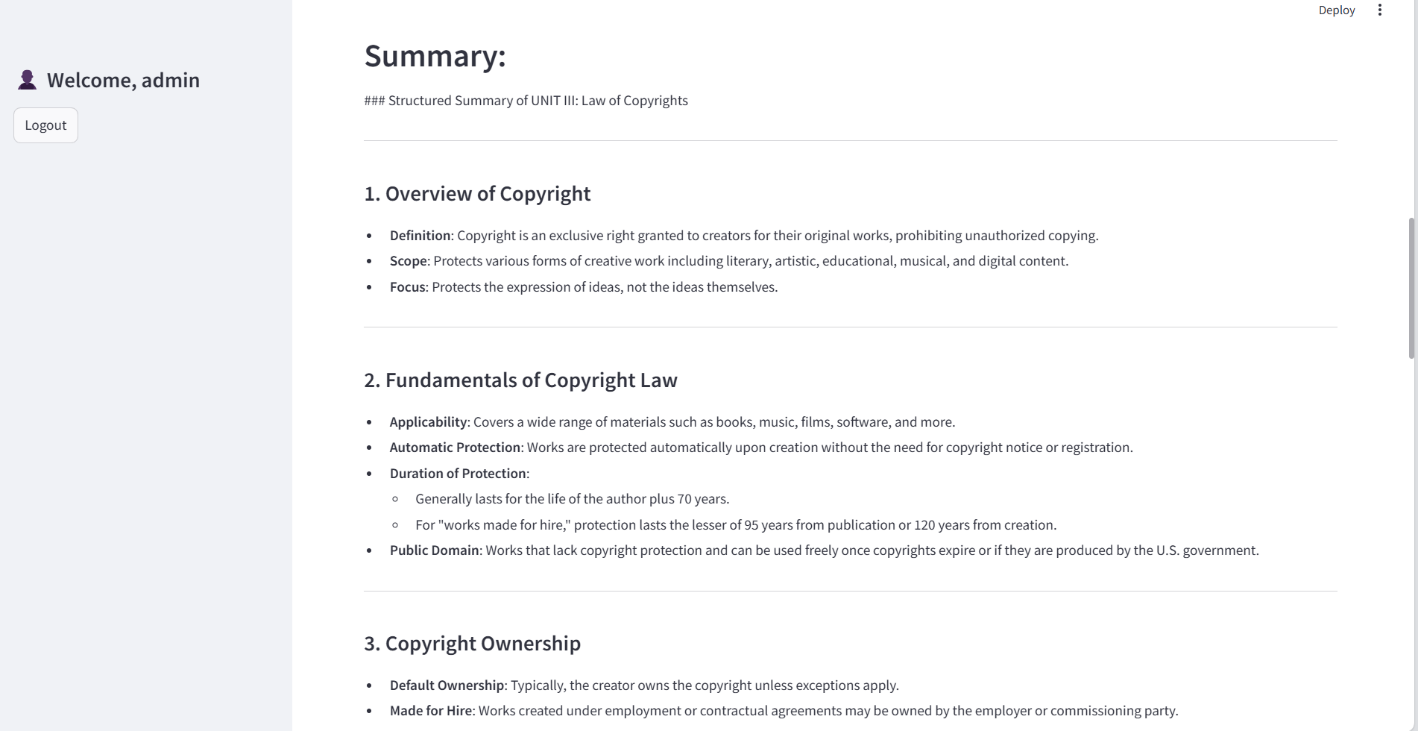
**Discussion:**

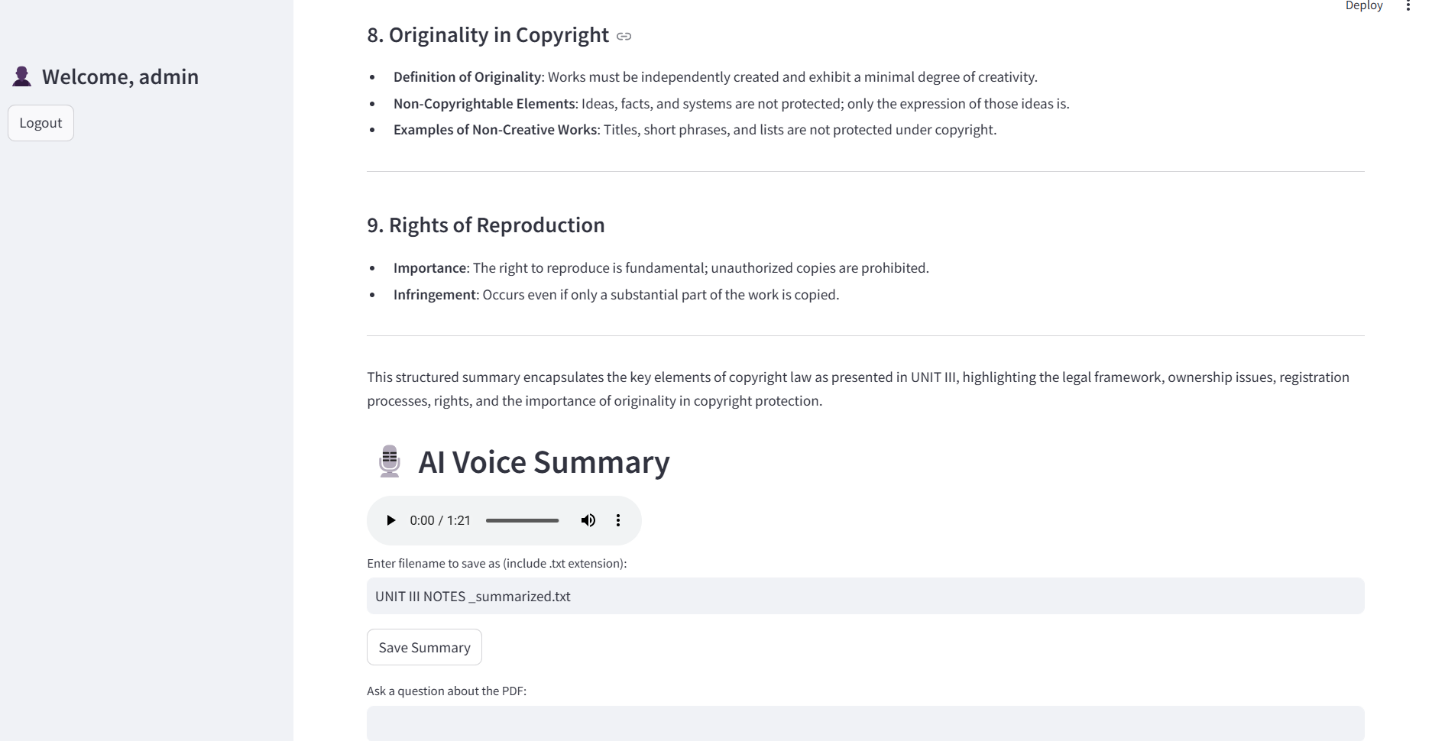
* The generated results are contextually rich and highly relevant.
* The embedding model ensures semantic understanding for accurate QA.
* Some limitations include occasional summarization gaps and dependency on the clarity of the input PDF (e.g., scanned images may fail text extraction).

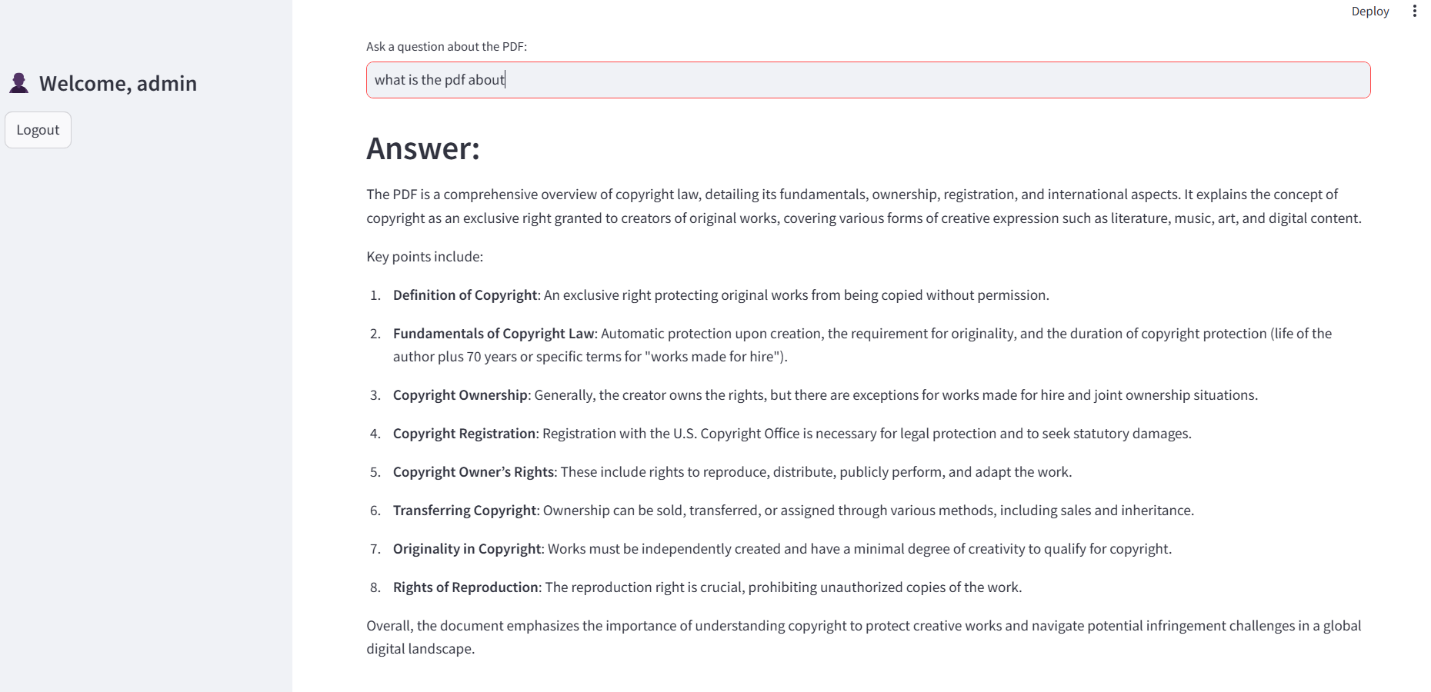
**5.3 Testing**











**5.4 Validation**

### **Summary Validation**

* Summaries were compared against original PDFs by domain experts.
* In most cases, GPT-4 generated summaries captured:
  + Key ideas
  + Section transitions
  + Critical terms and methods
* Users reported over **85% relevance** in randomly selected outputs.

### **QA Validation**

* Answers were judged on:
  + Relevance to the question
  + Accuracy with respect to PDF context
  + Clarity of explanation
* For technical PDFs, answers to definitions, differences, and advantages were on-point and accurate.

### **User Feedback**

* Peer users evaluated the bot's usefulness, simplicity, and clarity.
* The **audio summary** feature was found engaging, especially for quick reviews.
* Feedback suggested adding:
  + Support for scanned PDFs via OCR
  + Multi-document support
  + Export to PDF/Word

**CHAPTER 6**

**CONCLUSION**

**6.1 Conclusion**

The GenAI-Based QA Bot presented in this project is a robust, user-friendly application that harnesses the power of modern language models and vector-based search to enable intelligent document understanding and question answering. By integrating OpenAI's GPT-4, Hugging Face embeddings, and FAISS vector indexing within a Streamlit interface, the bot provides comprehensive functionalities—from PDF document summarization and topic extraction to natural language question answering and text-to-speech narration.

With a simple login system and intuitive UI, the application ensures secure and seamless access for multiple users. The modular structure allows for future extensibility, such as adding support for multiple document formats, more advanced retrieval mechanisms (e.g., RAG), and multilingual capabilities. Its practical utility makes it a valuable tool for academic institutions, professionals, and researchers seeking quick insights from large documents.

Overall, the project successfully demonstrates the application of Generative AI and Retrieval-Augmented Generation in solving real-world information extraction challenges, highlighting its potential in educational and enterprise domains.

**6.2 Future Scope**

* **Enhanced Graph Analysis**: Implement more advanced image processing techniques to

extract structured data from complex graphs.

* **Multilingual Support**: Extend summarization and QA capabilities to support multiple

languages.

* **Real-time Speech Interaction**: Integrate speech-to-text for a more interactive

experience.

* **Cloud Deployment**: Host the bot on a cloud platform to enable wider accessibility.
* **Improved Security Measures**: Implement OAuth-based authentication for enhanced security.

**REFERENCES**

**[1] Langchain documentation** -<https://python.langchain.com/en/latest/index.html>

**[2] FAISS: Facebook AI Similarity Search** - <https://faiss.ai/>

**[3] OpenAI GPT-4 API** - <https://platform.openai.com/docs/>

**[4] PyTTSX3 (Text-to-Speech Conversion)** - <https://pypi.org/project/pyttsx3/>

**[5] PyPDF2 for PDF Analysis** - <https://pymupdf.readthedocs.io/en/latest/>

**[6] Pillow (PIL)** **for image processing** - <https://pillow.readthedocs.io/en/stable/>

**[7] Sentence Transformers -** <https://www.sbert.net/>

**APPENDIX**

**Source Code**

**#Source code to be present here**