Submitted by:

Smriti Tiwari – 2201195

Nandini Modi – 2201128

Nisha – 2201132

CS683 PROJECT REPORT

Question Answering System over Custom Documents

This project presents a **Question Answering (QA) System** designed to operate over **custom user-provided documents**. The system enables users to upload PDF files, ask natural language questions, and receive contextually accurate answers extracted directly from those documents.
It combines **document embeddings**, **vector-based retrieval (FAISS)**, and **large language models (LLMs)** to provide precise, domain-specific responses.
Built using **Streamlit**, **LangChain**, **HuggingFace**, and **Groq LLM**, the system demonstrates how retrieval-augmented generation (RAG) can make AI assistants work effectively with private data sources.

INTRODUCTION

In today's world, vast textual information is often locked within documents that are not accessible through standard AI models.

This project aims to bridge that gap by enabling **natural language interaction** with **custom, private documents**.

The system reads and processes user-uploaded PDF files, converts them into vector embeddings, and uses a retrieval mechanism to identify the most relevant content for any user query.

The retrieved context is then passed to a **large language model (Groq Llama-3.1)**, which formulates a natural, coherent answer.

Objectives:

- To build an interactive question-answering system over user documents.
- To apply NLP-based retrieval techniques for document understanding.
- To demonstrate RAG (Retrieval-Augmented Generation) in practice.

SYSTEM ARCHITECTURE

The system follows an **8-step pipeline**:

1. **User Uploads PDFs** – The Streamlit interface allows multiple PDFs to be uploaded.
2. **Text Extraction** – Using PyPDFLoader, textual content is extracted from each file.
3. **Text Chunking** – The text is divided into smaller chunks using RecursiveCharacterTextSplitter.

4. **Embeddings Generation** – Each chunk is converted into a vector embedding using **HuggingFace MiniLM**.
5. **Vector Indexing** – The embeddings are stored in a **FAISS vector database** for fast retrieval.

6. **User Query Input** – The user asks a question in natural language.
7. **Context Retrieval** – The system retrieves semantically similar chunks from FAISS.
8. **Answer Generation** – The **Groq Llama-3.1 model** generates a concise, context-based answer.

This pipeline ensures high accuracy, scalability, and relevance of responses.

METHODOLOGY

a. Data Preprocessing

Uploaded PDF files are converted to text and segmented into chunks to manage long contexts efficiently.

b. Embedding Generation

Each text chunk is embedded into a fixed-length dense vector using:

model = "sentence-transformers/all-MiniLM-L6-v2"

This representation captures semantic meaning and allows similarity search.

c. Vector Database

FAISS (Facebook AI Similarity Search) is used to store and retrieve the document embeddings efficiently.

It enables millisecond-level nearest-neighbor search among thousands of document vectors.

d. Retrieval Chain

When a question is asked:

- The query is embedded into the same vector space.
- FAISS retrieves the top similar chunks.
- These chunks are fed into the language model through LangChain's **retrieval chain**.

e. Answer Generation

The **ChatGroq** model (llama-3.1-8b-instant) processes both the query and retrieved context to produce a natural answer.

The system ensures that responses are **context-aware** and **document-grounded**.

IMPLEMENTATION DETAILS

Core Components:

- **Streamlit:** For building the interactive web interface.
- **LangChain:** Provides the framework for chaining retrieval and generation steps.
- **HuggingFace Embeddings:** Used for semantic vector representation.

- **FAISS:** Enables efficient similarity-based retrieval.
- **Groq LLM:** Provides powerful reasoning and text generation.
- **Python-dotenv:** Manages environment variables securely.

Important Functions:

- compute_bytes_hash(): Detects file changes via hashing.
- create_new_index_from_paths(): Creates FAISS index for all uploaded documents.
- create_stuff_documents_chain(): Builds the question-context-answer pipeline.
- retrieval_chain.invoke(): Executes retrieval and LLM query generation.

RESULTS AND DISCUSSION

The system performs efficiently, answering queries within seconds.

It correctly identifies relevant sections from multiple documents and generates coherent, contextual answers.

Example:

Uploaded File: "Can Thought Exist Without Language?"

Query: "What does Vygotsky argue about thought and language?"

Answer: "Vygotsky suggests that thought and language develop independently and later merge to enable complex reasoning."

This demonstrates the system's ability to handle **academic or domain-specific** content effectively.

ADVANTAGES

- Works on **private, domain-specific documents**.
- **Contextual answers** grounded in the provided text.
- **Fast and scalable** due to FAISS-based retrieval.
- **Model-agnostic** – supports multiple LLMs.
- **User-friendly interface** with Streamlit.
- Automatically rebuilds the FAISS index if files are updated.

FUTURE ENHANCEMENTS

- Add **conversational memory** to support follow-up questions.
- Integrate **user authentication** for private data access.
- Support **multiple LLMs** (Groq, OpenAI, Gemini, etc.).
- Provide **visual analytics** of questions and answers.
- Deploy on cloud for broader accessibility.

CONCLUSION

This project successfully implements an end-to-end **Question Answering System** using NLP, vector retrieval, and large language models.

It showcases how **retrieval-augmented generation (RAG)** can make AI systems intelligent, explainable, and grounded in user-provided data.

The approach is versatile and can be extended to knowledge management, education, and enterprise application.

REFERENCES

1. LangChain Documentation — https://python.langchain.com
2. FAISS Library — https://github.com/facebookresearch/faiss
3. Groq API Documentation — https://console.groq.com/docs
4. Sentence Transformers — https://www.sbert.net
5. Streamlit — https://streamlit.io