Mission UpSkill India Hackathon

Team:

DeepThinkers

Project Title:

Document Q & A System

Team Members:

Atmika M Banerjee

Grace Ann Mathew

Praisy Sharon Varghese

Date of Submission:

07/09/2025

TABLE OF CONTENTS

SL.NO	TOPIC	PAGE NO.
1.	Problem Statement	1
2.	Objectives	1
3.	Proposed Solution	2
4.	System Architecture	2-3
5.	Features of the Solution	4
6.	Technology Stack	4
7.	Implementation Details	5
8.	Result & Demo Screenshots	6-7
9.	Impact & Future Scope	8
10.	Conclusion	8
11.	References	9

PROBLEM STATEMENT :

Accessing and understanding information from large documents such as PDFs, research papers, product manuals, and wikis is often time-consuming and overwhelming. Users typically need to manually search through lengthy files or rely on inefficient keyword-based search, which does not always return contextual or accurate results.

This problem is important because it affects a wide range of people:

- Students who need to quickly grasp concepts from long textbooks or research papers.
- Employees who must query internal documentation, technical manuals, or wikis to find solutions and work efficiently.
- Customers who struggle to navigate product manuals or FAQs to solve their issues without external support.

Without an intelligent system to retrieve and summarize information, users waste valuable time, face frustration, and risk missing critical details.

OBJECTIVES:

The primary objectives of this project are:

1. **Enable Natural Language Q&A**
   Allow users to upload documents (PDF, Markdown, HTML) and ask questions in plain English, receiving accurate and context-aware answers.
2. **Efficient Information Retrieval**
   Implement a vector-based search using FAISS and embeddings (Sentence-Transformers) to retrieve the most relevant document chunks.
3. **Leverage Advanced Language Models**
   Use LLaMA-2 (7B, quantized) to generate high-quality, human-like responses grounded in the retrieved context.
4. **Provide Source Transparency**
   Display citations from the original document chunks alongside each answer, so users can trust and verify the response.
5. **Offer a Simple, User-Friendly Interface**
   Build an interactive Gradio interface that makes it easy for non-technical users to upload files and interact with the system.
6. **Scalable & Extendable Solution**
   Design the solution so that it can be extended to other document formats, larger models, or integrated into enterprise applications.

PROPOSED SOLUTION:

To address the challenge of efficiently retrieving knowledge from large and complex documents, our team has built an AI-powered Document Q&A System using Retrieval-Augmented Generation (RAG).

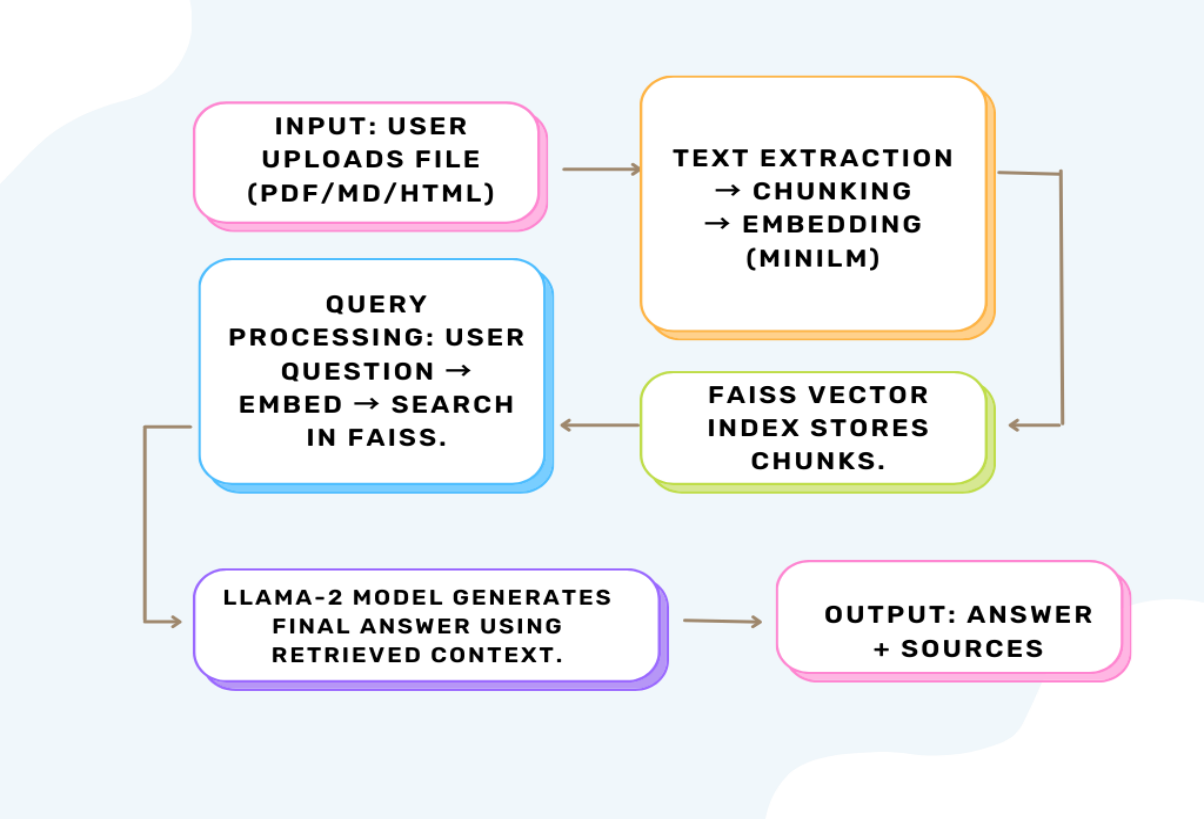
The solution allows users to:

- Upload documents in formats such as PDF, Markdown, and HTML.
- Automatically extract and chunk the text into manageable pieces.
- Store embeddings of these chunks in a FAISS vector database for efficient semantic search.
- Use SentenceTransformers for generating embeddings and LLaMA-2 (7B Chat) as the reasoning model for contextual answers.
- Provide natural language answers to user queries along with source citations, ensuring accuracy and traceability.
- Offer an easy-to-use Gradio interface where users can ask questions directly without technical knowledge.

This approach combines the speed of vector similarity search with the reasoning power of large language models, enabling fast, accurate, and explainable document Q&A.

SYSTEM ARCHITECTURE:

High Level Architecture Diagram:



Explanation of each component:

1) **User Interface (Gradio Frontend)**

- Provides a simple interface for users to upload files and ask questions.
- Displays generated answers and cites relevant sources.

2) **Document Upload**

- Accepts PDF, Markdown, and HTML files.
- Ensures flexibility for different types of content.

3) **Text Extraction & Preprocessing**

- PyPDF2 → extracts text from PDF files.
- Markdown2 + BeautifulSoup → converts Markdown/HTML into clean text.
- Removes noise (scripts, styles) and prepares text for processing.

4) **Chunking & Embedding**

- Long text is split into chunks (~1500 characters with overlap).
- SentenceTransformers (all-MiniLM-L6-v2) generates dense embeddings.
- Ensures semantic meaning is captured, not just keyword matching.

5) **Vector Database (FAISS)**

- Stores embeddings of all document chunks.
- Enables fast semantic search **to** retrieve the most relevant passages.

6) **LLaMA-2 Model (7B Chat, 4-bit Quantized)**

- Acts as the reasoning engine.
- Combines retrieved context with user query to generate accurate answers.
- Provides citations to increase trust and reliability.

7) **Answer with Sources**

- The final response is displayed to the user along with the document page/section it was retrieved from.
- Ensures transparency, accuracy, and traceability**.**

FEATURES OF THE SOLUTION:

• **Multi-format Document Support** – Upload and process documents in PDF, Markdown, and HTML formats.

• **Automated Text Extraction** – Uses PyPDF2, Markdown2, and BeautifulSoup to cleanly extract text from different file types.

• **Smart Text Chunking** – Splits documents into manageable chunks with overlap, ensuring context is preserved.

• **Semantic Search with FAISS** – Stores embeddings in a vector database for fast and accurate retrieval based on meaning, not just keywords.

• **Powerful Embeddings** – Uses SentenceTransformers (all-MiniLM-L6-v2) for high-quality embeddings.

• **AI-Powered Answer Generation** – Employs LLaMA-2 (7B Chat, quantized) for generating human-like, context-aware answers.

• **Source Citation** – Each generated answer includes references to original document sections/pages for transparency.

• **User-Friendly Interface** – Built with Gradio, enabling non-technical users to easily upload files and ask questions.

• **Lightweight & Efficient** – Optimized with 4-bit quantization for reduced memory usage and faster inference.

• **Scalable Design** – Architecture can handle multiple document uploads and be extended to enterprise-scale use cases.

TECHNOLOGY STACK:

- Our solution is built using a combination of modern AI, NLP, and web technologies that make it both powerful and user-friendly. At its core, the system is written in Python, which serves as the backbone for text processing, model integration, and the overall workflow.
- For document parsing, we rely on specialized libraries: PyPDF2 for PDFs, Markdown2 for markdown files, and BeautifulSoup for HTML content. This ensures that no matter the file type, the text is extracted cleanly and structured for further processing.
- Once extracted, the text is chunked and embedded using SentenceTransformers (all-MiniLM-L6-v2), which generates high-quality vector embeddings. These embeddings are stored and searched using FAISS, a lightweight yet highly efficient vector database that enables fast similarity search.

- For the language model, we use LLaMA-2 (7B Chat, quantized to 4-bit), accessed through Hugging Face Transformers. Quantization makes the model run faster and more efficiently on limited hardware without losing much accuracy.
- On the frontend side, we implemented an intuitive Gradio interface, allowing users to upload files and ask natural language questions directly, without requiring technical expertise.

Altogether, this stack combines robust document handling, efficient vector search, and powerful LLM-driven reasoning into a seamless, end-to-end pipeline.

IMPLEMENTATION DETAILS:

The development of our solution followed a step-by-step approach, starting from document handling and moving toward an end-to-end interactive system.

We began by designing the document ingestion pipeline. Using PyPDF2, Markdown2, and BeautifulSoup, we ensured that text could be extracted cleanly from PDFs, Markdown exports, and HTML files. Once the raw text was available, we implemented a chunking strategy that splits the text into segments of around 1,500 characters with a slight overlap. This overlap preserved context so that no important details were lost between chunks.

The next stage was embedding generation. We used SentenceTransformers (all-MiniLM-L6-v2) to convert each chunk into a numerical vector representation. These embeddings were normalized and stored in a FAISS index, which allowed us to efficiently retrieve the most semantically relevant chunks for any user query.

For the retrieval-augmented generation (RAG) step, we loaded LLaMA-2 (7B Chat) in 4-bit quantized form using Hugging Face Transformers. This reduced the model's memory footprint, making it more efficient to run. When a user submits a question, the system encodes the query into an embedding, searches FAISS for the top-k matching chunks, and then provides both the context and the question to LLaMA-2. The model generates a contextual answer, citing the relevant sources.

To make the solution accessible, we built a Gradio interface. Users can upload documents, type questions, and receive answers along with references in a simple, chat-like interface.

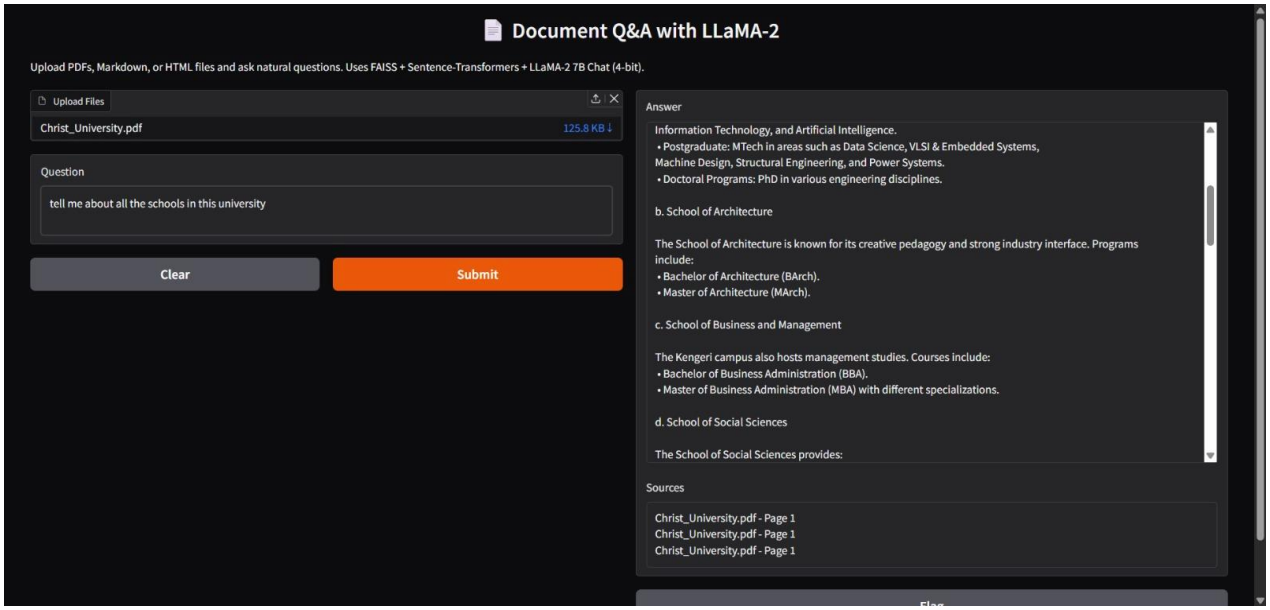
Challenges Faced & Solutions:

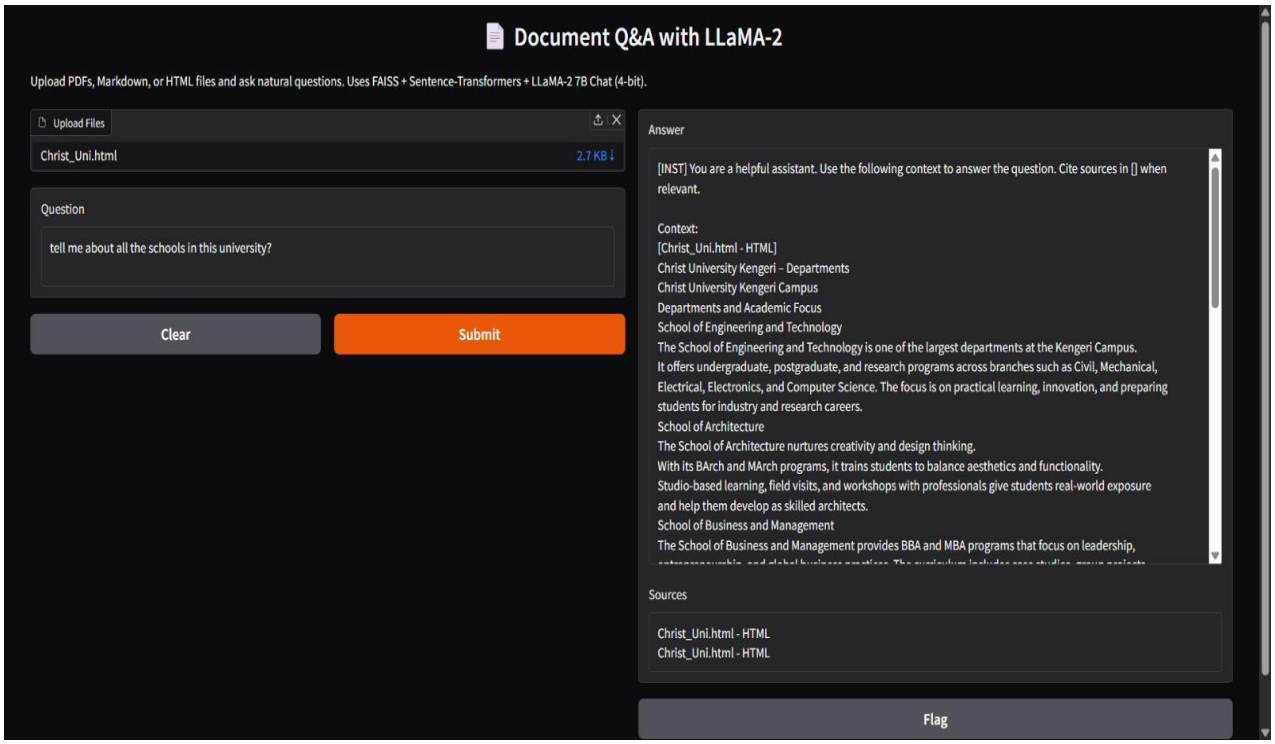
- **Handling different file formats:** Extracting text from PDFs, Markdown, and HTML posed challenges because of formatting inconsistencies. We addressed this by writing separate parsing functions for each type.
- **Maintaining context in long documents:** Splitting text risked losing meaning across sections. We solved this using overlapping chunks to ensure continuity.
- **Model efficiency:** Running large models like LLaMA-2 requires heavy computation. We optimized by using **4-bit quantization** and efficient device mapping.
- **Accuracy of answers:** Initially, answers were generic. Incorporating **retrieved context** from FAISS improved both precision and relevance.
- **User interface simplicity:** Hackathon users may not be technical. We solved this by designing a clean Gradio UI with minimal steps — upload, ask, and get an answer.

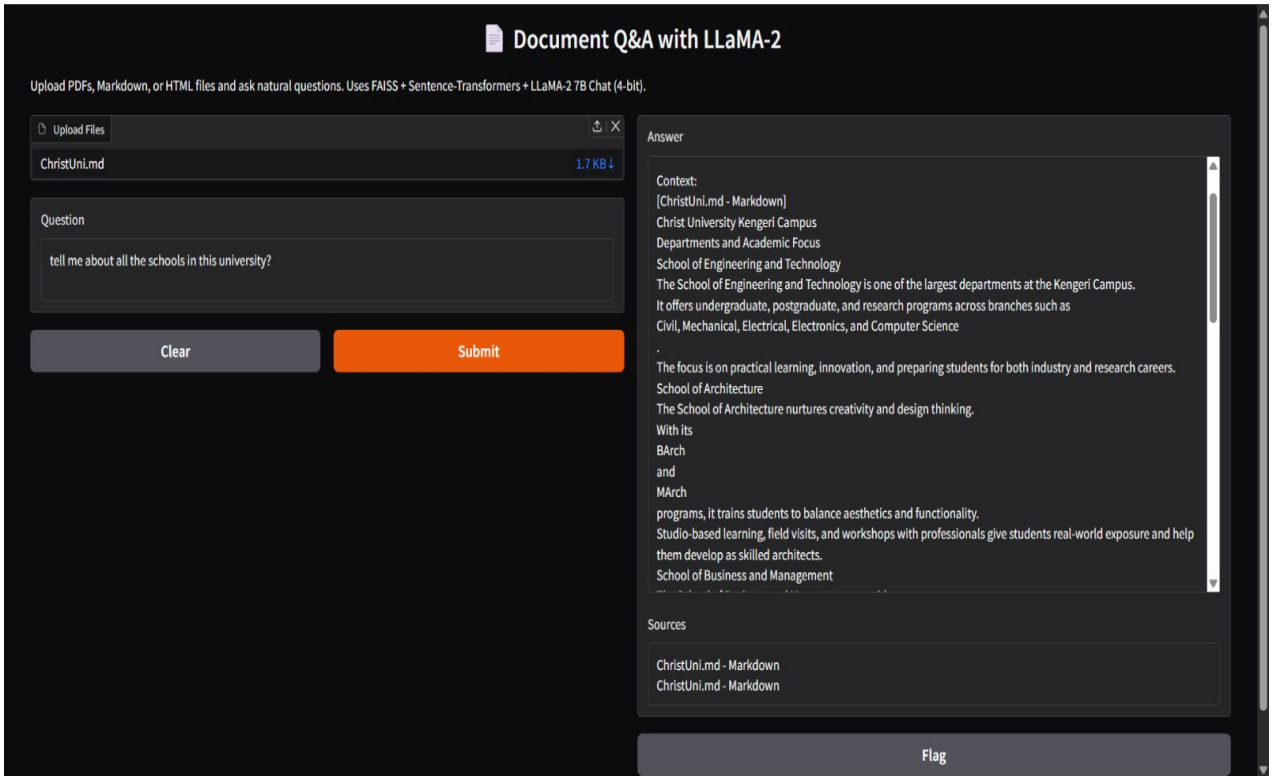
Through this structured pipeline, we successfully implemented a system that combines document retrieval, semantic search, and generative AI into one smooth user experience.

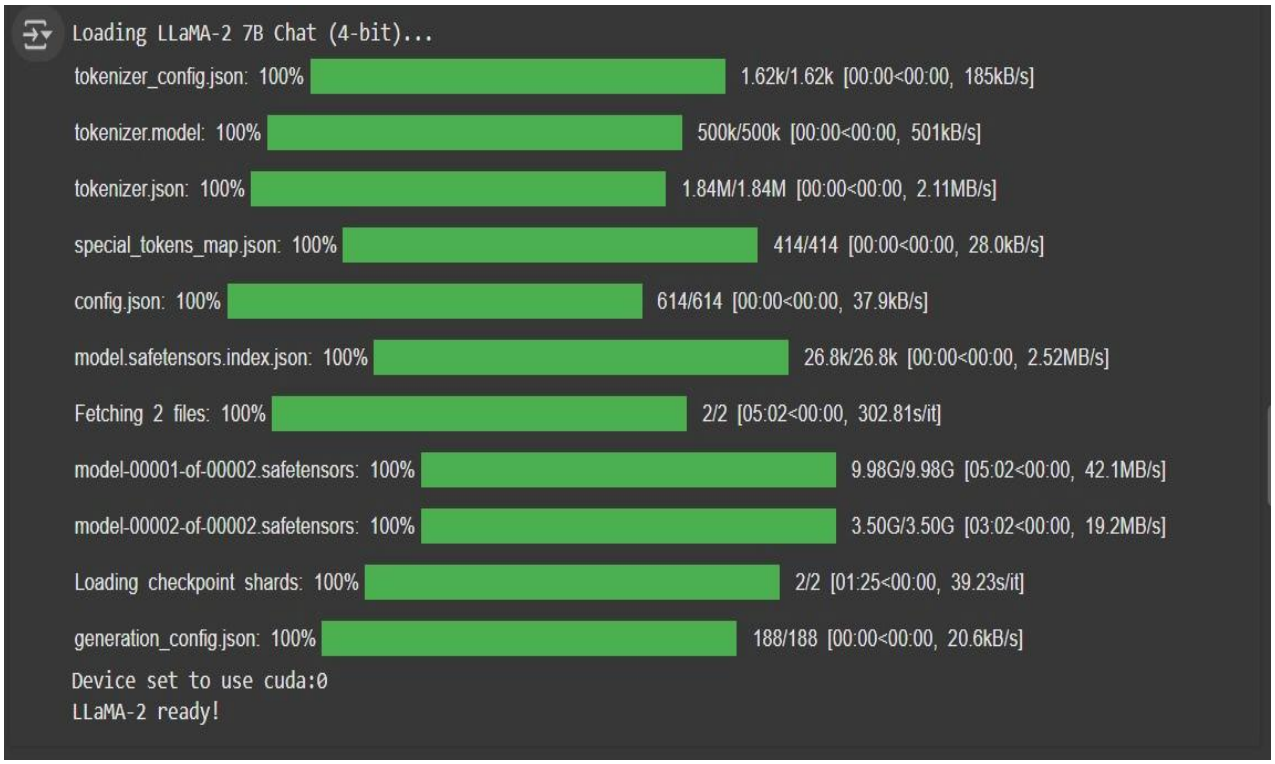
RESULTS & DEMO SCREENSHOTS:

Our system was successfully implemented and tested with different types of documents (PDF, Markdown, and HTML). The results demonstrate that the application can extract content, perform semantic search, and generate accurate, contextual answers with source references.






Document Q&A with LLaMA-2
Upload PDFs, Markdown, or HTML files and ask natural questions. Uses FAISS + Sentence-Transformers + LLaMA-2 7B Chat (4-bit).
Upload Files
ChristUni.md
1.7 KB
Question
tell me about all the schools in this university?
Clear
Submit
Answer
Context:
[ChristUni.md - Markdown]
Christ University Kengeri Campus
Departments and Academic Focus
School of Engineering and Technology
The School of Engineering and Technology is one of the largest departments at the Kengeri Campus.
It offers undergraduate, postgraduate, and research programs across branches such as
Civil, Mechanical, Electrical, Electronics, and Computer Science
.
The focus is on practical learning, innovation, and preparing students for both industry and research careers.
School of Architecture
The School of Architecture nurtures creativity and design thinking.
With its
BArch
and
MArch
programs, it trains students to balance aesthetics and functionality.
Studio-based learning, field visits, and workshops with professionals give students real-world exposure and help them develop as skilled architects.
School of Business and Management
Sources
ChristUni.md - Markdown
ChristUni.md - Markdown
Flag


Loading LLaMA-2 7B Chat (4-bit)...
tokenizer_config.json: 100% 1.62k/1.62k [00:00<00:00, 185kB/s]
tokenizer.model: 100% 500k/500k [00:00<00:00, 501kB/s]
tokenizer.json: 100% 1.84M/1.84M [00:00<00:00, 2.11MB/s]
special_tokens_map.json: 100% 414/414 [00:00<00:00, 28.0kB/s]
config.json: 100% 614/614 [00:00<00:00, 37.9kB/s]
model.safetensors.index.json: 100% 26.8k/26.8k [00:00<00:00, 2.52MB/s]
Fetching 2 files: 100% 2/2 [05:02<00:00, 302.81s/it]
model-00001-of-00002.safetensors: 100% 9.98G/9.98G [05:02<00:00, 42.1MB/s]
model-00002-of-00002.safetensors: 100% 3.50G/3.50G [03:02<00:00, 19.2MB/s]
Loading checkpoint shards: 100% 2/2 [01:25<00:00, 39.23s/it]
generation_config.json: 100% 188/188 [00:00<00:00, 20.6kB/s]
Device set to use cuda:0
LLaMA-2 ready!

IMPACT AND FUTURE SCOPE:

Our solution directly addresses the challenge of navigating large, complex documents by providing quick, accurate, and contextual answers. This has a broad real-world impact across multiple domains:

- Education: Students can save valuable study time by querying long research papers, lecture notes, or textbooks.
- Corporate Environments: Employees gain faster access to organizational knowledge bases, wikis, or technical manuals, improving productivity.
- Customer Support: Customers can resolve queries independently by interacting with product manuals and FAQs through natural language.

Beyond immediate benefits, the project has scalable potential. Future improvements may include:

- Support for additional formats like Word, Excel, and scanned image-based PDFs (with OCR).
- Integration with Notion, Google Docs, and enterprise knowledge bases for seamless use in workplaces.
- Enhancements such as answer highlighting, confidence scoring, and multi-turn conversational memory.
- Deployment as a cloud-based SaaS product that can scale to enterprise-level usage.

This makes the project not just a hackathon prototype but a foundation for real-world applications.

CONCLUSION:

In this project, we successfully built an intelligent Document Q&A system powered by Retrieval-Augmented Generation (RAG). The system combines document parsing, semantic search, and large language models into a single workflow that allows users to interact with their documents naturally.

Through testing, we demonstrated that the solution can process multi-format documents, retrieve contextually relevant information, and generate accurate answers with supporting references. By optimizing with quantization and providing a user-friendly interface, we made the system both efficient and accessible.

This project highlights the potential of AI in transforming how we access and interact with information, paving the way for faster learning, better workplace efficiency, and improved customer experiences.

REFERENCES:

- Meta AI. *LLaMA-2: Open Foundation and Fine-Tuned Chat Models*. https://ai.meta.com/llama

- Hugging Face. *Transformers Library Documentation*. https://huggingface.co/docs/transformers/index

- Sentence Transformers. *Pretrained Models and Documentation*. https://www.sbert.net/docs/pretrained_models.html

- Facebook AI Research. *FAISS: A Library for Efficient Similarity Search*. https://github.com/facebookresearch/faiss

- Gradio. *Quickstart Guide*. https://www.gradio.app/guides/quickstart

- PyPDF2. *Official Documentation*. https://pypdf2.readthedocs.io/en/latest/

- BeautifulSoup4. *Documentation*. https://www.crummy.com/software/BeautifulSoup/bs4/doc/