Research Paper Explainer & Blog Generator

Introduction:

1. Project Overview.
2. Project Library Installation and Setup.
3. Block Diagram.
4. Tech Stack.
5. RAG Chain & Blog Generation.
6. Steps.
7. Frontend- UI Design.

Project Overview: Simplifying Research Paper for Broader Accessibility

- **Objective:-** The project aims to bridge the gap between complex research and general understanding by making scholarly content more accessible to a wider audience.
- **Problem Statement:-** Research papers, especially those from sources like Google Scholar and academic conferences, are often written in dense, technical language that is difficult for non-experts to interpret.
- **Solution Approach:-** We developed a tool that takes a PDF of a research paper as input and produces a simplified, blog-style explanation as output.
- **Simplified Output:-** The generated blog post is written in plain, conversational language designed for easy comprehension by readers without a technical background.

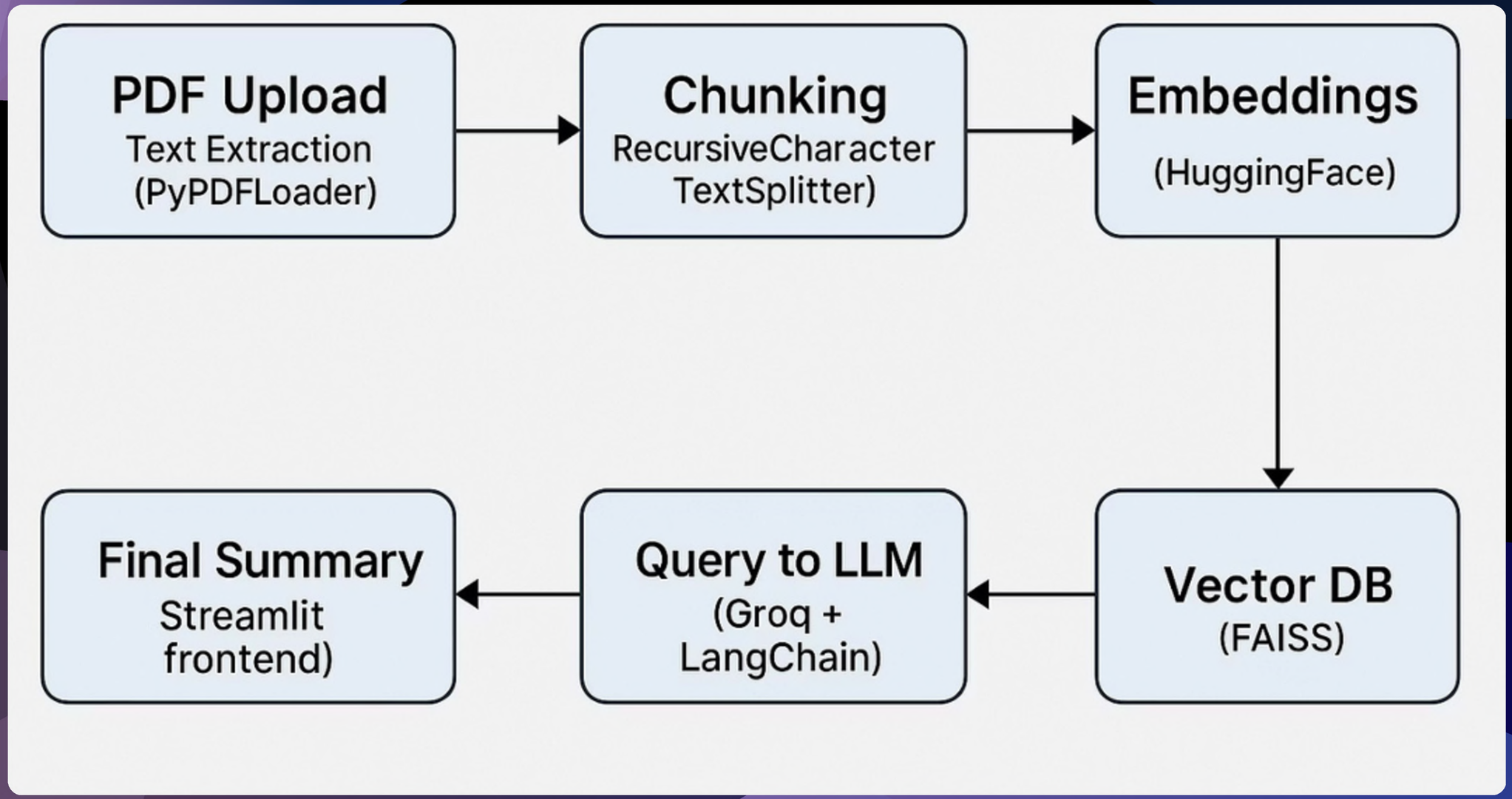
Project Overview:

- **Focus Area:-** Each simplified blog post highlights three essential components of the original paper:
  - The problem or research question being addressed.
  - The methodology or approach used in the study.
  - The key findings or conclusions drawn.

- **Target Audience:-** This tool is especially beneficial for students, professionals from non-academic fields, and curious general readers who want to engage with cutting-edge research without struggling with technical jargon.

- **End Goal:** By translating academic research into more approachable formats, we aim to foster a more informed public, promote interdisciplinary learning, and encourage broader participation in scholarly discourse.

Block Diagram



Tech-Stack:

- **LangChain:**
  → An AI orchestration framework that chains together LLMs, retrievers, and tools to process documents intelligently.

- **HuggingFace Embeddings:**
  → Converts text chunks into dense vector embeddings for similarity search and retrieval.

- **FAISS (Facebook AI Similarity Search):**
  → A fast vector database used to index and retrieve semantically similar documents chunks.

- **ChatGroq:**
  → Integrates Groq's ultra-fast hardware-accelerated inference with Meta's LLaMA 3 (70B) language model to generate intelligent, natural responses.

- **RecursiveCharacterTextSplitter:**
  → Splits large documents into overlapping text chunks suitable for embedding and retrieval.

- **RetrievalQA Chain (LangChain):**
  → Combines a retriever and a language model to answer questions based on document context, enabling contextual summarization.

- **PyPDFLoader:**
  → Extracts text from PDF documents and converts it into a structured format for further processing.

Steps

1. Library Installation & Setup

```
!pip install -q langchain langchain-community langchain-core faiss-cpu PyMuPDF
!pip install -q langchain-groq
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 26.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31.3/31.3 MB 28.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.1/24.1 MB 29.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.2/45.2 kB 2.8 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.9/50.9 kB 2.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.8/130.8 kB 3.6 MB/s eta 0:00:00
```

2. PDF Upload & API Key Setup

```
[17] import os
     os.environ["GROQ_API_KEY"] = "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"
```

```
from google.colab import files
uploaded = files.upload()
pdf_path = list(uploaded.keys())[0]
```

Choose files Train_Crash...System.pdf

- **Train_Crash_Avoidance_System.pdf**(application/pdf) - 862444 bytes, last modified: 01/07/2025 - 100% done

Saving Train_Crash_Avoidance_System.pdf to Train_Crash_Avoidance_System (2).pdf

Steps

3. Extracting & Chunking PDF Text

```
[19] from langchain.document_loaders import PyPDFLoader
     from langchain.text_splitter import RecursiveCharacterTextSplitter

     loader = PyPDFLoader(pdf_path)
     documents = loader.load()

     splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
     chunks = splitter.split_documents(documents)

     print(f"Total chunks: {len(chunks)}")

     Total chunks: 26
```

4. Embedding & Vector Store Creation

```
[20] from langchain_community.vectorstores import FAISS
     from langchain.embeddings import HuggingFaceEmbeddings

     embedding = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
     vectorstore = FAISS.from_documents(chunks, embedding)
```

RAG Chain & Blog Generation

LLM Setup with GROQ

```
from langchain_groq.chat_models import ChatGroq
from langchain.chains import RetrievalQA

llm = ChatGroq(
    model="llama-3.3-70b-versatile",
    temperature=0.3,
    api_key=os.environ["GROQ_API_KEY"]
)

retriever = vectorstore.as_retriever()
qa_chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)
```

The LLaMA 3 model (70B version) is loaded from GROQ using the ChatGroq class, with temperature set to 0.3 for more focused, accurate responses. This model will later be used to generate the blog-style summary.

RAG Chain & Blog Generation

Building the RAG Pipeline

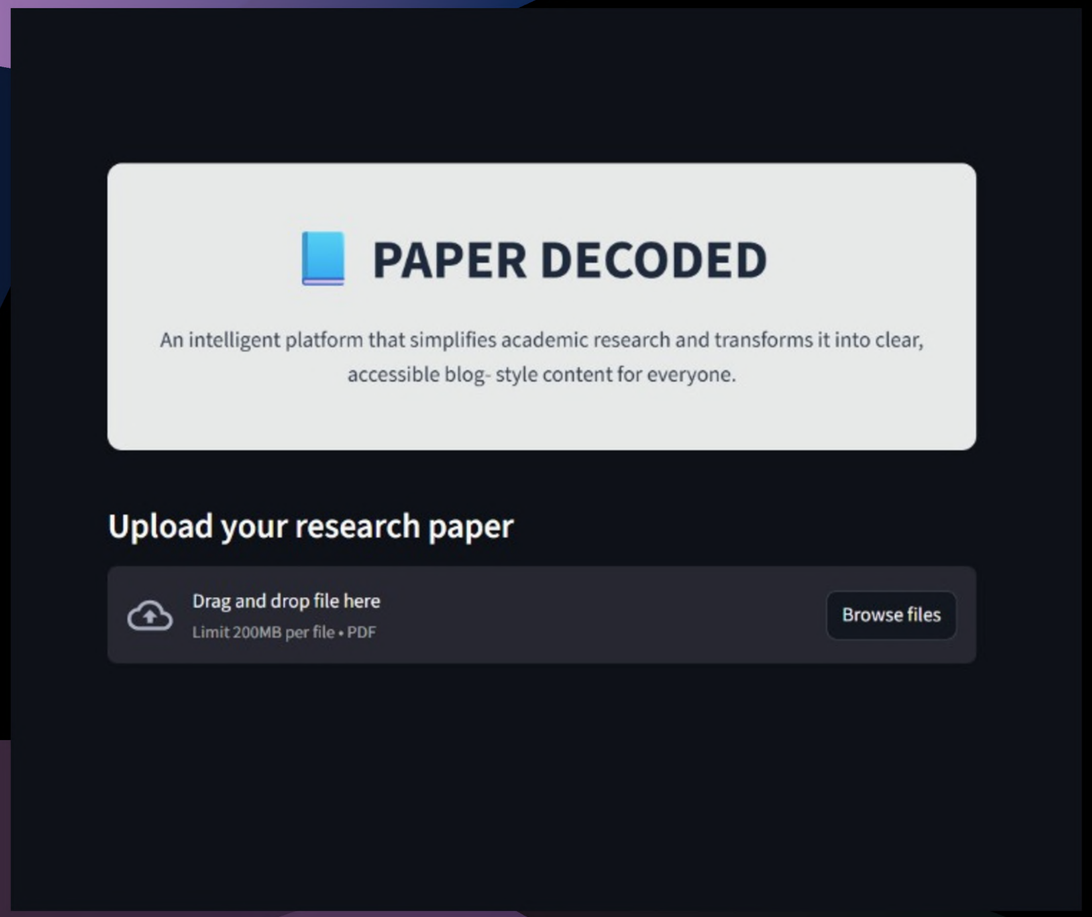
TheFAISS vector store is turned into a retriever, which helps find the most relevant chunks based on the query. We then connect this retriever with the LLM using LangChain's RetrievalQA, which completes the RAG (Retrieval-Augmented Generation) setup.

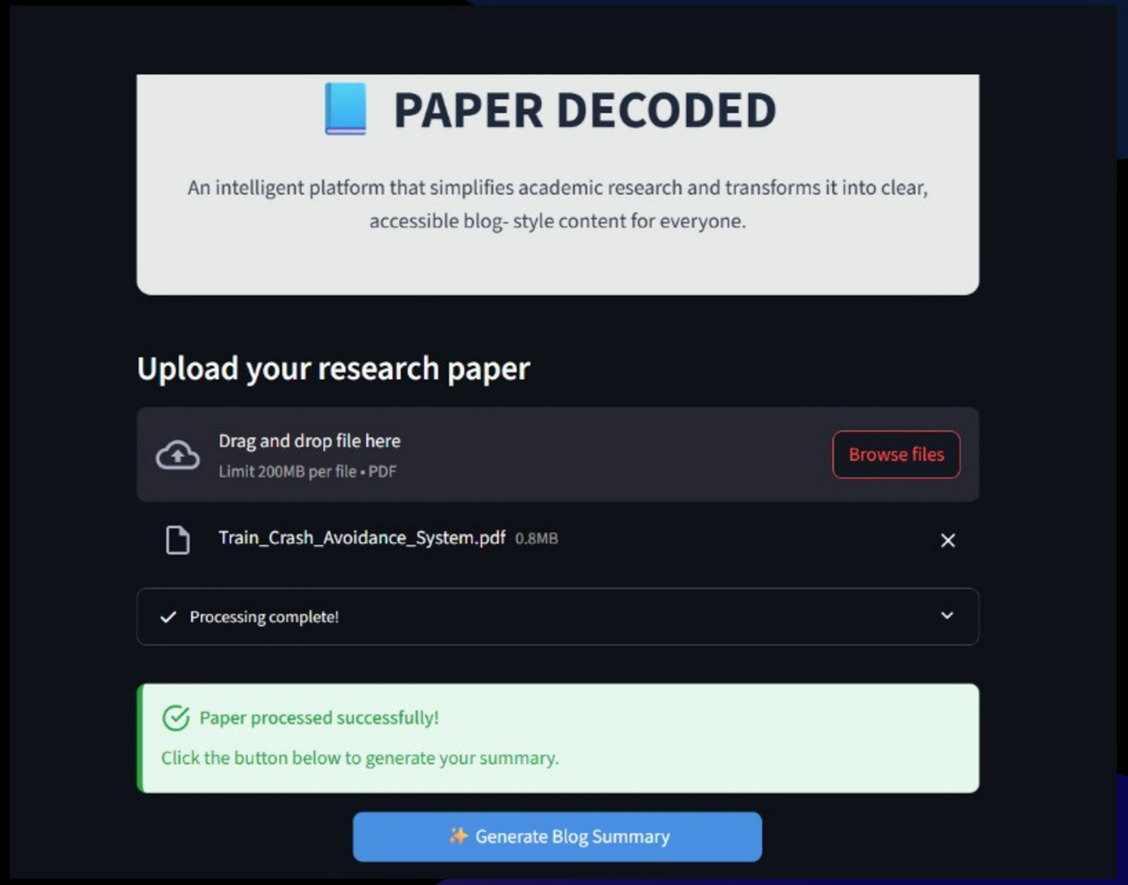
```
from langchain_groq.chat_models import ChatGroq
from langchain.chains import RetrievalQA

llm = ChatGroq(
    model="llama-3.3-70b-versatile",
    temperature=0.3,
    api_key=os.environ["GROQ_API_KEY"]
)

retriever = vectorstore.as_retriever()
qa_chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)
```

Frontend


PAPER DECODED
An intelligent platform that simplifies academic research and transforms it into clear, accessible blog- style content for everyone.
Upload your research paper
Drag and drop file here
Limit 200MB per file • PDF
Browse files


PAPER DECODED
An intelligent platform that simplifies academic research and transforms it into clear, accessible blog- style content for everyone.
Upload your research paper
Drag and drop file here
Limit 200MB per file • PDF
Browse files
Train_Crash_Avoidance_System.pdf 0.8MB
Processing complete!
Paper processed successfully!
Click the button below to generate your summary.
Generate Blog Summary