Document-Based Question Answering System (RAG)

This document describes the design, architecture, and implementation of a Retrieval-Augmented Generation (RAG) system for question answering over PDF and DOCX documents using open-source models only.

1. Project Objective

The goal of this project is to build an end-to-end document question answering system that allows users to upload documents and ask natural language questions. The system retrieves relevant information from the documents and generates accurate answers strictly grounded in the document content.

2. High-Level Architecture

The system follows a modular Retrieval-Augmented Generation pipeline consisting of document ingestion, text processing, vectorization, retrieval, and answer generation.

1. Document Ingestion (PDF / DOCX)
2. Text Cleaning and Normalization
3. Text Chunking with Overlap
4. Embedding and Vector Indexing (FAISS)
5. Top-K Semantic Retrieval
6. Context Construction
7. LLM-based Answer Generation

3. Module Descriptions

3.1 Document Ingestion

The system supports PDF and DOCX documents. Each document is read using specialized libraries and converted into raw text. Basic metadata such as filename, file type, character length, and word count are collected.

3.2 Text Cleaning

The raw text is normalized by removing extra whitespace, non-ASCII characters, line breaks, and converting text to lowercase. This improves embedding quality and retrieval consistency.

3.3 Chunking

The cleaned text is split into overlapping chunks to preserve semantic continuity. Each chunk is stored with metadata including chunk ID, source document, and length.

3.4 Embedding and Vectorization

Each chunk is converted into a dense vector embedding using a Sentence Transformer model. Embeddings are normalized and stored in a FAISS vector index to enable efficient similarity search.

3.5 FAISS Index Storage

The FAISS index and chunk metadata are persisted to disk. This avoids recomputation and reduces memory usage, allowing the system to scale to larger document collections.

3.6 Retrieval Module

User queries are embedded using the same embedding model. The FAISS index is searched to retrieve the top-K most relevant chunks based on cosine similarity.

3.7 Context Construction

Retrieved chunks are sorted by relevance score and concatenated into a single context block, respecting a maximum token budget to prevent model overflow.

3.8 Generation Module

A lightweight open-source language model (e.g., TinyLlama or Mistral) generates answers using a structured prompt. The model is instructed to answer strictly based on the provided context.

4. Prompt Design

The prompt includes system instructions, retrieved context, the user question, and explicit constraints to prevent hallucination. Only information present in the context is allowed in the final answer.

5. Fine-Tuning Strategy (Planned)

The project supports Supervised Fine-Tuning (SFT) of the generation model using domain-specific question–answer pairs. Fine-tuning is implemented as a separate training module and does not interfere with the main inference pipeline.

6. Design Decisions

1. Open-source models only (no paid APIs)
2. Disk-based FAISS storage for memory efficiency
3. Strict context-grounded generation
4. Notebook-friendly modular implementation

7. Conclusion

This project demonstrates a complete, modular, and extensible RAG-based question answering system. It is suitable for academic use, experimentation, and future improvements such as fine-tuning, evaluation metrics, and web deployment.