Architecture Document

Introduction

Retrieval-Augmented Generation (RAG) is an emerging tech that combines information retrieval with large language models (LLMs) to produce context-aware, grounded answers. Unlike pure LLM-based approaches, RAG systems leverage external knowledge bases, ensuring outputs are factually accurate, domain-specific, and less prone to hallucination. This project implements a modular, Naive RAG system as a baseline to support enhancements, experimentation, and evaluation in subsequent phases.

The system is designed around three pillars: embedding, retrieval, and generation. It employs all-MiniLM-L6-v2 and all-mpnet-base-v2 embeddings from Sentence-Transformers, stores them in a vector database (Milvus Lite), and retrieves relevant chunks for query-conditioned generation. This document describes the architecture, design decisions, trade-offs, and extensibility considerations.

System Architecture

1. High-Level Workflow
   Document Ingestion: Raw text is preprocessed, segmented into smaller chunks, and normalized.
   Embedding Generation: Each chunk is embedded into a fixed-size dense vector representation using all-MiniLM-L6-v2.
   Vector Indexing: Embeddings are inserted into Milvus Lite for similarity search.
   Query Encoding: User queries are embedded into the same vector space.
   Retrieval: Top-k most similar document chunks are retrieved via cosine similarity.
   Response Generation: Retrieved evidence is injected into a prompt template, and an LLM generates the final answer.

2. System Components
   Embedding Layer
   Library: HuggingFace sentence-transformers.
   Model: all-MiniLM-L6-v2 and all-mpnet-base-v2
   Embedding sizes tested: 384 (default) and 784 (experimental).

   Vector Database
   Milvus Lite - scalable, distributed, production-oriented.

   Retriever
   Query embeddings compared with stored vectors using cosine similarity.
   Configurable top-k retrieval (baseline = 1).

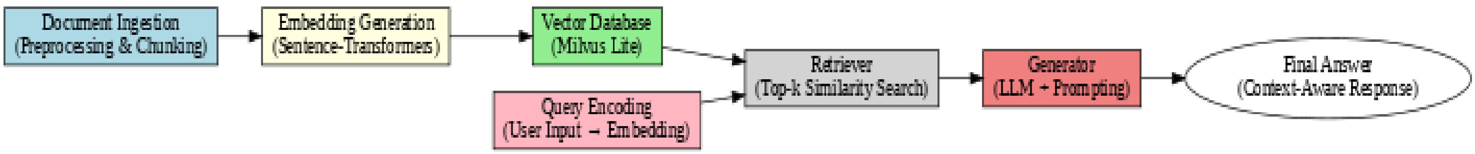
   Generator
   Takes retrieved document(s) and reformulates them into an answer.
   Supports various prompting strategies (Instruction, Chain-of-Thought, Persona).

Design Decisions and Trade-offs

1. Embedding Model
   all-MiniLM-L6-v2: 384 Embedding Size
   All-mpnet-base-v2: 784 Embedding Size

2. Vector Store Choice:
   Milvus Lite: Production-friendly, distributed retrieval, persistent storage.
   Options: ChromaDB, FAISS

3. Retrieval Strategy
   Baseline: Top k = 1
   Experimentation: Top k =3, 5




```
Documents --> Preprocess & Chunk --> Embedding Generation --> Vector DB
                                   ^                          |
                                   |                          v
Query --> Query Encoding --------------> Retriever --------------> Generator --> Final Answer
```