RAG PROTOTYPE- PLAN

1. Retrieval Index Construction

1.1 Vector Database Implementation

1. Technology: FAISS (Facebook AI Similarity Search)
2. Embedding Model: nomic-embed-text served through Ollama
3. Persistant Storage: faiss_index/ directory
4. Index Type: Flat L2 index for exact similarity search

1.2 Construction Process

1. Load PDF: Extract text from pdf using PyMuPDFLoader
2. Split Documents: Break into chunks using RecursiveCharacterTextSplitter
3. Generate Embeddings: Convert each chunk to vectors using embedding model
4. Build Index: Create FAISS index from document embeddings
5. Persist: Save index to disk at faiss_index/ for reuse

1.3 Retrieval Mechanism

- Returns top-k most relevant document chunks for each query
- Preserves metadata including page numbers
- Loads persisted index on startup
- Converts queries to embeddings for similarity search

2. Chunking Strategy

2.1 Configuration Parameters

- Chunk Size: 300 characters
- Chunk Overlap: 25 characters
- Splitter Type: RecursiveCharacterTextSplitter
- Separator Hierarchy: Paragraph,Line, Space, Character
- Source document reference, Page number, Chunk content with formatting is shown

3. Prompt Template Family for Audience Adaptation

Standard RAG Query: Use retrieved context, cite [Source X], format equations in LaTeX, avoid HTML tags, andadmit when the answer is not in the context.

Question Contextualization: *Reformulate follow-up into standalone by referencing chat history when needed, leave them unchanged if self-contained, and don't attempt to answer*

Refinement Query: *Review the previous output alongside feedback, enhance response additional context, return JSON with changes and reasons*