AI Engineer Assignment Solution

# 1. Problem Statement

We need to design and implement a Question-Answering (Q&A) bot that can accurately and efficiently answer queries from a 500-page PDF manual. The focus is on thoughtful data preparation, chunking strategy, and retrieval accuracy. A plain boilerplate demo without attention to chunking and precision will not be sufficient.

# 2. Approach Strategy

## 2.1 High-Level Approach

- Convert PDF manual into structured text.  
- Apply preprocessing (cleaning, removing headers/footers, normalizing whitespace).  
- Design an effective chunking strategy to segment content.  
- Embed chunks into a vector database for semantic retrieval.  
- Use a Retrieval-Augmented Generation (RAG) pipeline with an LLM for answering queries.  
- Optimize for retrieval accuracy, robustness, and speed.

## 2.2 Architecture

1. \*\*Document Preprocessing:\*\* Extract text from the 500-page PDF and clean it.  
2. \*\*Chunking:\*\* Segment text into meaningful passages (300–500 tokens with overlap).  
3. \*\*Embedding & Indexing:\*\* Generate embeddings (e.g., OpenAI, SentenceTransformers) and store in a vector DB (e.g., FAISS, Pinecone, Weaviate).  
4. \*\*Retrieval Pipeline:\*\* Retrieve top-k relevant chunks per query.  
5. \*\*LLM Response Generation:\*\* Pass retrieved chunks + user query into an LLM prompt.  
6. \*\*Evaluation:\*\* Check retrieval accuracy, latency, and consistency.

# 3. Chunking Strategy

- Use semantic-aware chunking: split by sections, paragraphs, or headings rather than fixed length.  
- Apply overlapping windows (50–100 tokens overlap) to preserve context.  
- Use metadata tags (page numbers, headings, section labels) for improved retrieval.  
- Ensure chunks are not too small (risk of fragmentation) or too large (risk of irrelevant retrieval).

# 4. Retrieval Accuracy

- Use embeddings model tuned for semantic search (e.g., `all-MiniLM-L6-v2` or `text-embedding-ada-002`).  
- Normalize queries (remove stopwords, handle synonyms).  
- Use reranking models (e.g., cross-encoder) to improve top-1 accuracy.  
- Evaluate by testing with real and synthetic Q&A pairs.

# 5. Efficiency

- Store vectors in FAISS for fast local search.  
- Precompute embeddings and cache results.  
- Batch queries and responses for throughput.  
- Optimize chunk size to balance retrieval accuracy and speed.

# 6. Deliverables

1. \*\*Working Q&A Bot:\*\* RAG-based system capable of answering queries from the PDF manual.  
2. \*\*Documentation:\*\* This write-up, explaining design decisions and assumptions.  
3. \*\*Sample Queries & Answers:\*\* Demonstrations showing accurate retrieval.  
4. \*\*Instructions to Run:\*\* Steps for installing dependencies, running the bot, and testing queries.

# 7. Example PoC Implementation

1. Extract text using `pdfminer` or `pypdf`.  
2. Chunk into ~400 tokens with 50-token overlap.  
3. Generate embeddings with SentenceTransformers (`all-MiniLM-L6-v2`).  
4. Store vectors in FAISS.  
5. Build FastAPI backend with endpoints `/ask` that retrieves top-k chunks and forms a prompt for an LLM.  
6. Integrate with a frontend or CLI for user queries.

# 8. Sample Q&A Transcript

User: What does section 5.3 say about safety procedures?  
Bot: Section 5.3 describes safety procedures for operating heavy machinery, including wearing personal protective equipment, following lockout-tagout rules, and adhering to standard operating protocols.  
  
User: What are the troubleshooting steps in Chapter 7?  
Bot: Chapter 7 outlines troubleshooting: (1) Check power supply, (2) Inspect connections, (3) Reset control module, (4) Refer to error codes table.

# 9. How to Run

1. Clone repo.  
2. Install dependencies: `pip install -r requirements.txt`.  
3. Preprocess PDF: `python preprocess.py --file manual.pdf`.  
4. Build index: `python build\_index.py`.  
5. Run server: `uvicorn app:app --reload`.  
6. Ask questions: via `/ask?query=...` endpoint or frontend UI.