RAG Pipeline Implementation – Q&A

Qus: What method or library did you use to extract the text, and why? Did you face any formatting challenges with the PDF content?

Ans: We used PyMuPDF (fitz) for rendering pages and pytesseract for OCR because the PDFs were image-based Bangla texts. Yes, line breaks and hyphenated words caused formatting issues.

Qus: What chunking strategy did you choose (e.g. paragraph-based, sentence-based, character limit)? Why do you think it works well for semantic retrieval?

Ans: We used RecursiveCharacterTextSplitter with 300-character chunks and 50-character overlap. This balances context retention and vector granularity for semantic search.

Qus: What embedding model did you use? Why did you choose it? How does it capture the meaning of the text?

Ans: We used 'intfloat/multilingual-e5-base' as it supports both Bangla and English. It maps semantically similar text into nearby vector space using contrastive learning.

Qus: How are you comparing the query with your stored chunks? Why did you choose this similarity method and storage setup?

Ans: We use FAISS with cosine similarity on SentenceTransformer embeddings. FAISS enables fast nearest-neighbor search for real-time retrieval.

Qus: How do you ensure that the question and the document chunks are compared meaningfully? What would happen if the query is vague or missing context?

Ans: Embedding both queries and chunks in the same vector space ensures semantic alignment. Vague queries may lead to low similarity and irrelevant chunk retrieval.

Qus: Do the results seem relevant? If not, what might improve them (e.g. better chunking, better embedding model, larger document)?

Ans: Most results are relevant. For improvement, we could try sentence-level chunking or use a stronger embedding model like multilingual-e5-large.