1. 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 pdfplumber to extract text from the PDF. It was chosen because it reliably handles text extraction from PDFs with complex layouts, unlike some other libraries (e.g., PyMuPDF) which cause import or usage issues in our environment. pdfplumber allows extraction page by page, preserving most textual structure.
Challenges:
Some formatting issues were encountered such as inconsistent newlines, mixed fonts, or stray characters, especially in Bengali script. To mitigate this, we applied custom cleaning and normalization — removing unwanted characters and normalizing whitespace.

2. 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 a fixed-size chunking strategy based on word count, grouping approximately 300 words per chunk.

Reason:
- Paragraph boundaries in PDFs can be inconsistent or missing, especially after extraction.
- Sentence tokenization for Bengali can be tricky due to limited tools.
- Fixed word chunks ensure consistent-sized inputs for embedding models, improving retrieval stability.
This size balances semantic coherence (enough context in each chunk) and manageable embedding vector sizes. It helps the retrieval system find relevant chunks without too much noise or overly broad content.

3. What embedding model did you use? Why did you choose it? How does it capture the meaning of the text?
Ans: We used the sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 model.

Reason:
- It supports multilingual embeddings, including both Bengali and English, which suits the multilingual requirement.
- It is lightweight and efficient, allowing faster embedding on typical hardware.
- It produces semantically meaningful sentence-level embeddings that capture the contextual meaning, rather than just keywords.
This model encodes text into dense vectors where semantically similar sentences or paragraphs are closer in vector space, enabling meaningful similarity search.

4. How are you comparing the query with your stored chunks? Why did you choose this similarity method and storage setup?
Ans: We compare queries and stored chunks by:

- Encoding the user query with the same embedding model.
- Using FAISS with L2 distance (Euclidean distance) to find the top-k closest chunk embeddings to the query vector.

Reason:
- FAISS is a highly efficient, scalable similarity search library optimized for large vector datasets.
- L2 distance works well with the MiniLM embedding vectors for semantic similarity.
- Storing embeddings in a FAISS index enables very fast nearest neighbor retrieval, crucial for responsive systems.

5. 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: Meaningful comparison is ensured by:
- Using the same embedding model and vector space for both queries and chunks.
- Preprocessing text consistently (cleaning, tokenization) before embedding.
- Chunking text into semantically coherent units to preserve context.

Limitations:
- If a query is very vague or lacks context, the retrieval may return chunks that are only loosely related or too general.
- The model cannot infer missing background knowledge, so retrieval depends heavily on how well the chunks cover the document topics.

6. Do the results seem relevant? If not, what might improve them (e.g. better chunking, better embedding model, larger document)?
Ans: Current observations:
- The system provides reasonably relevant answers for well-formed queries in both Bengali and English.
- Some answers may be generic or incomplete if the retrieved chunks do not contain precise info.

Potential improvements:
- Better chunking: Use semantic-aware chunking (e.g., paragraph or sentence boundaries, or overlap chunks) to improve retrieval granularity.
- Stronger embedding models: Use larger multilingual models like LaBSE or mUSE for richer semantic representation.
- Expand document corpus: More documents or multiple sources can increase knowledge coverage.
- Advanced reranking: Add a re-ranking step after retrieval to prioritize the most relevant chunks.
- Query expansion: Preprocess queries to add context or clarify vague questions.