The following PDF contains the required questions and answers for the assessment.

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

Ans: PyMuPDF library has been used to extract text from PDFs because it is fast, and efficient. It preserves the original formatting, including punctuation marks like । (Dari), which are important for Bangla sentence structure. However there were some challenges in detecting যুক্তবর্ণ and vowel signs (স্বরচিহ্ন). Apart from that the model seemed to struggle to understand সাধু ভাষা . This might be due to the low resources on bangla for model training.

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: I used LlamaIndex's SentenceSplitter with a Bangla-aware approach, setting the chunk size to 600 tokens and an overlap of 60 tokens. This method splits the text at Bangla punctuation marks such as "।", "?", and "!" to preserve sentence meaning. Additionally, it uses a regular expression ([^,.;।]+[,.;।]?) to avoid breaking text in the middle of a sentence.This strategy works well for semantic retrieval because it maintains context by overlapping chunks. It is more effective than fixed-size splitting, as it respects natural language boundaries and preserves semantic coherence.

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

Ans: I chose the paraphrase-multilingual-mpnet-base-v2 model from Sentence Transformers since it supports multilingual text, including Bangla. It is trained on parallel texts, which allows it to capture semantic meaning across different languages effectively. Compared to smaller models like paraphrase-multilingual-MiniLM-L12-v2, it offers improved performance, especially for non-English queries.

In my experiments, this model yielded better results on the given Bangla document than both sagorsarker/bangla-bert-base and paraphrase-multilingual-MiniLM-L12-v2, making it the most suitable choice for semantic retrieval in a bilingual RAG setup.

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

Ans: I used cosine similarity which is default in ChromaDB and FAISS, to compare query and chunk embeddings. For storage, I used ChromaDB due to ease of use, and smooth integration with LangChain.This approach enables semantic search by retrieving chunks with similar meaning rather than relying on keyword matches. I also experimented with reranking using cross-encoder/ms-marco-MiniLM-L-6-v2, but the results were not up to the mark in my case.

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: To manage vague queries, I used chat history to add context and clarify user's intent. Still, if the query is too general or the document doesn't contain enough relevant details, the system might provide generic responses or occasionally produce answers based on information from outside sources.

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

Ans: Some results are relevant, and the model is able to retrieve text passages containing the answers. However, the quality of the generated responses needs improvement. Due to quota limitations, I was unable to use OpenAI models, and other more powerful models require a GPU. It's important to note that the effectiveness of retrieval and answer generation largely depends on the choice of the LLM. The results could improve significantly if I had access to paid LLMs like OpenAI (gpt-4) or other high-capacity models that need GPU support.