Documentation: Model Architecture, Retrieval Approach, and Generative Responses

1. Model Architecture:

The solution employs a **Retrieval-Augmented Generation (RAG)** architecture, combining a retrieval system with a generative model. Here's how each part works:

- **Retrieval**: A vector database (Pinecone DB) is used to store document embeddings. The embeddings are vectors representing the content of documents or segments, which allow for efficient semantic search.
- **Generation**: Once relevant information is retrieved, a generative model, such as facebook/bart-large-cnn, is used to produce answers. The generative model takes the retrieved context and a user query to generate coherent, contextually appropriate responses.
- **Combined Workflow**:
    - **Embedding Creation**: The document is first split into smaller text chunks, and each chunk is converted into a vector using the SentenceTransformer model.
    - **Vector Storage and Query**: The vectors are stored in Pinecone DB. When a query is made, the system retrieves the most relevant chunks based on the similarity of their vector representations to the query's embedding.
    - **Answer Generation**: The retrieved text chunks serve as context for the generative model to construct the final answer.

2. Approach to Retrieval:

The retrieval process is based on **semantic search** using embeddings. Pinecone DB is responsible for efficient storage and retrieval of these embeddings:

- **Document Embedding**: The SentenceTransformer model is used to convert both documents and queries into vector embeddings. These embeddings are high-dimensional representations of the text.
- **Semantic Search**: When a user enters a query, it is also transformed into an embedding. Pinecone DB then performs a similarity search by comparing the query embedding with the document embeddings. It retrieves the top-k most relevant document chunks based on the cosine similarity of their vectors.
- **Efficient Retrieval**: The use of a vector database like Pinecone allows for quick retrieval even for large document sets. It handles the scaling of embedding-based search for complex queries.

3. Generative Responses:

The generative response mechanism works by incorporating the retrieved context into a question-answering framework:

- **Model**: A text2text-generation pipeline is used with the BART model (facebook/bart-large-cnn), which excels at summarization and answer generation tasks. The model receives the following prompt:

- **Prompt Construction**: "Context: {retrieved_chunks}\n\nQuestion: {query}\n\nAnswer:"
- **Response Generation**: The model generates a coherent response by combining the context with the query. It aims to answer the question using the information provided in the retrieved chunks.

4. Example Flow:

1. **Document Upload**: A PDF is uploaded, and text is extracted.
2. **Embedding Creation**: Text chunks are created and converted into embeddings.
3. **Storage**: Embeddings are stored in Pinecone DB.
4. **Query Input**: User submits a question.
5. **Retrieval**: Relevant text chunks are retrieved from Pinecone based on query similarity.
6. **Answer Generation**: The BART model generates a response using the retrieved text as context.

Conclusion:

The **RAG architecture** allows the system to not only retrieve relevant document sections but also generate comprehensive, coherent responses based on this information. This combination of retrieval and generation ensures that users receive accurate, context-aware answers to their questions.