**Model Architecture and Approach**

**1. Model Architecture**

* **Sentence Transformer Model**:
  + **Model**: bert-base-nli-mean-tokens
  + **Purpose**: This model is used to encode textual data into high-dimensional vector embeddings. It utilizes BERT (Bidirectional Encoder Representations from Transformers) architecture, specifically fine-tuned for generating sentence embeddings. By converting sentences into fixed-size vectors, the model captures the semantic meaning and contextual information of the text. The resulting embeddings are 768-dimensional vectors that represent the content of each text chunk from the PDF.
* **Pinecone**:
  + **Purpose**: Pinecone is a vector database designed to manage and search high-dimensional embeddings efficiently. It serves as a key component in the retrieval process, allowing for fast similarity search over large volumes of vector data. Pinecone handles the indexing and querying of the document embeddings. The system supports various distance metrics, such as cosine similarity, to find the closest matches between query embeddings and stored document embeddings.
* **Cohere**:
  + **Purpose**: Cohere is used for generating natural language responses based on the context provided by the document chunks. It employs a pre-trained language model to produce coherent and contextually appropriate answers. By using the document chunks and the user's query as input, Cohere generates a response that synthesizes information from the relevant sections of the PDF. This model helps transform the structured data retrieval into a natural language format, enhancing the usability of the information extracted from the document.

**2. Approach to Retrieval**

* **Text Extraction**:
  + **Method**: The process begins with extracting text from the uploaded PDF file. This is accomplished using PyPDF2, a Python library that reads and processes PDF files. The text extraction involves converting each page of the PDF into a readable format, ensuring that the entire content of the document is available for further processing.
* **Text Chunking**:
  + **Method**: Given that long documents may be cumbersome to handle in one go, the extracted text is divided into smaller, manageable chunks. Each chunk is typically 512 characters long, allowing the system to process and encode text more efficiently. This chunking approach ensures that the document can be handled in pieces without losing context or relevance.
* **Embedding Generation**:
  + **Method**: Each text chunk is then converted into a vector representation using the Sentence Transformer model. This step involves encoding the text into 768-dimensional vectors that capture the semantic content of the text. These embeddings serve as a numerical representation of the document’s content, enabling efficient comparison and retrieval.
* **Indexing and Retrieval**:
  + **Method**: The embeddings are stored in Pinecone, where they are indexed for efficient retrieval. When a user submits a query, the system converts the query into an embedding and searches the Pinecone index for similar document embeddings. This retrieval process involves calculating similarity scores between the query embedding and stored embeddings to find the most relevant document chunks.
* **Generative Response Creation**:
  + **Method**: Once relevant document chunks are identified, a prompt is constructed that includes the user’s query and the retrieved chunks. This prompt is fed into Cohere, which generates a natural language response. The response is synthesized from the information provided by the relevant document chunks, delivering a coherent answer to the user’s query.