**Technical Documentation of the Multilingual RAG System**

**System Overview**

The Multilingual Retrieval-Augmented Generation (RAG) System is designed to handle documents and queries in multiple languages, including Hindi, Bengali, Chinese, and English. The system processes both scanned and digital PDFs, extracts text, and provides accurate responses to queries. The architecture leverages advanced NLP models, hybrid search techniques, and scalable vector databases to deliver high-performance retrieval and generation capabilities.

**Components and Architecture**

**1. Text Extraction**

**1.1 OCR for Scanned PDFs**

* Tool: Tesseract OCR
* Functionality: Extracts text from scanned PDF images.
* Process: Preprocesses images (e.g., noise reduction, binarization), performs OCR, and outputs text.

**1.2 Text Extraction for Digital PDFs**

* Tool: PyMuPDF (Fitz) and pdfplumber
* Functionality: Extracts text from digitally created PDFs.
* Process: Directly parses and extracts textual content without the need for OCR.

**2. Document Chunking**

**2.1 Recursive Character Text Splitter**

* Tool: Custom Implementation using RecursiveCharacterTextSplitter
* Functionality: Splits documents into manageable chunks for efficient processing and retrieval.
* Process: Recursively divides text based on character limits, ensuring chunks are contextually coherent.

**3. Hybrid Search Implementation**

**3.1 Sparse Vector Creation**

* Model: BERT-Base Multilingual Uncased
* Process: Generates sparse vectors by removing token IDs for CLS, SEP, MASK, and padding tokens, capturing keyword-based information.

**3.2 Dense Vector Embeddings**

* Model: BERT-Base Multilingual Uncased
* Process: Creates dense vector embeddings representing the semantic content of document chunks.

**4. Vector Database**

**4.1 Pinecone DB**

* Functionality: Stores hybrid search vectors (both sparse and dense) and enables efficient retrieval.
* Process: Indexes vectors, performs nearest neighbour search, and retrieves relevant chunks based on user queries.

**5. Query Processing**

**5.1 Query Decomposition**

* Model: Facebook/xglm-564M
* Functionality: Decomposes complex queries into simpler sub-queries for more accurate retrieval.
* Process: Breaks down queries into key components and processes each part independently.

**6. Information Retrieval and Generation**

**6.1 Retrieval**

* Tool: Pinecone DB Query Interface
* Functionality: Retrieves relevant document chunks based on the hybrid search vectors.
* Process: Combines results from both sparse and dense searches, ranks them, and returns the most relevant chunks.

**6.2 Generation**

* Model: Facebook/xglm-564M
* Functionality: Generates responses, summaries, and answers to questions based on retrieved information.
* Process: Uses the model to synthesize information from retrieved chunks and generate coherent outputs.

**7. Summarization and Question Answering**

**7.1 Summarization**

* Model: Facebook/xglm-564M
* Functionality: Summarizes large documents or sets of documents.
* Process: Extracts key points and generates concise summaries.

**7.2 Question Answering**

* Model: Facebook/xglm-564M
* Functionality: Answers specific questions based on document content.
* Process: Uses retrieved information to generate precise and relevant answers.

**8. Scalability**

* Data Handling: Capable of processing up to 1TB of data.
* Performance: Optimized for low-latency retrieval and high relevance of search results.