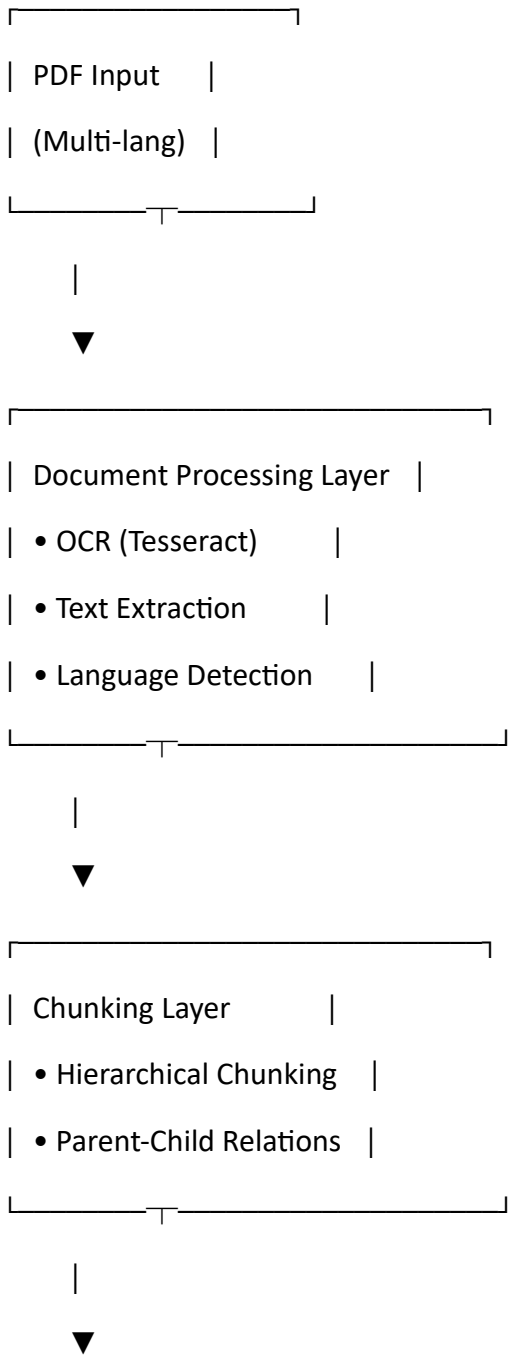
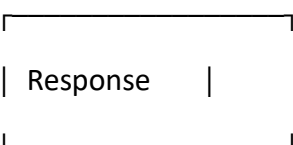
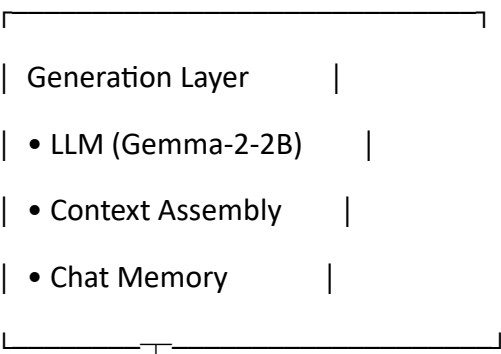
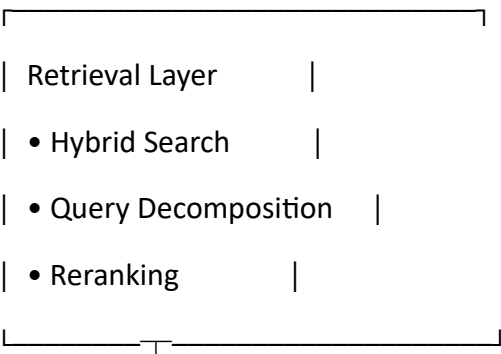
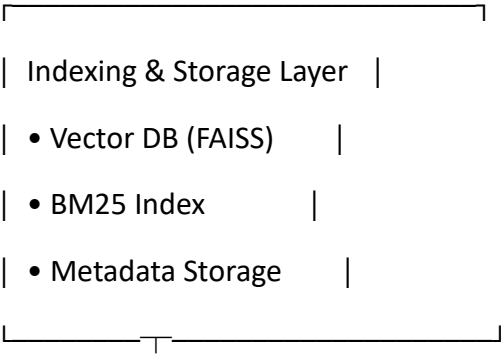


Technical Documentation: Multilingual PDF RAG System

1. System Architecture Overview

High-Level Architecture





2. Core Components

2.1 Document Processing

- **PDF Extractor:** Handles both digital and scanned PDFs
 - Digital extraction: PyMuPDF (fitz) for better RTL support
 - OCR: Tesseract with support for eng+hin+ben+chi_sim+chi_tra+ara+urd
 - Confidence threshold: 60% for OCR quality assurance
- **Language Detection:**
 - Script-based detection for RTL languages (Arabic/Urdu)
 - langdetect library with confidence scoring
 - Multi-language document support

2.2 Chunking Strategy

- **Hierarchical Chunking:**
 - Parent chunks: 2000 characters
 - Child chunks: 512 characters
 - Overlap: 100 characters
 - Structure-aware: Preserves document sections and headings
 - RTL-aware separators for better Arabic/Urdu handling

2.3 Embedding & Vector Storage

- **Model:** paraphrase-multilingual-MiniLM-L12-v2
 - Dimension: 384
 - Size: ~118M parameters
 - Supports 50+ languages
- **Vector Database:** FAISS (IndexFlatIP)
 - Normalized vectors for cosine similarity
 - In-memory for speed
 - Scalable to billions of vectors

2.4 Retrieval System

- **Hybrid Search:**
 - Semantic search (70% weight) via FAISS
 - Keyword search (30% weight) via BM25
 - Combined scoring for optimal results
- **Query Decomposition:**
 - Splits complex queries into sub-queries
 - Handles compound questions with "and", "also", "plus"
- **Reranking:**
 - Exact match boosting (2x)
 - Term overlap scoring
 - Length penalty normalization
 - Top-K filtering (default: 5)

2.5 Generation

- **LLM:** google/gemma-2-2b-it
 - Size: 2B parameters
 - 4-bit quantization (~1.5GB memory)
 - Context-aware generation
 - Temperature: 0.7 for balanced creativity
- **Chat Memory:**
 - Rolling window of 5 exchanges
 - Context injection for follow-up questions

3. Key Technical Decisions

3.1 Model Selection Rationale

1. **Embedding Model:** Chose multilingual model over language-specific to handle mixed-language documents
2. **LLM Size:** 2B model provides excellent quality-to-size ratio with 4-bit quantization
3. **Quantization:** 4-bit reduces memory by ~75% with minimal quality loss

3.2 Performance Optimizations

- Batch processing for embeddings (batch_size=32)
- FAISS inner product search for speed
- In-memory indexing for low latency
- Normalized vectors for efficient similarity computation

3.3 Scalability Considerations

- FAISS can handle billions of vectors
- Horizontal scaling possible via sharding
- Bottleneck is LLM generation, not retrieval
- Estimated capacity: 1TB+ with distributed setup

4. System Requirements

Hardware

- GPU: NVIDIA T4 or better (for inference)
- RAM: 16GB+ recommended
- Storage: Depends on document collection size

Software Dependencies

- Python 3.8+
- CUDA 11.8+ (for GPU acceleration)
- Tesseract OCR with language packs
- Key libraries: transformers, sentence-transformers, faiss-cpu, chromadb

5. Performance Characteristics

Current Metrics (29 PDFs, 2027 chunks)

- Average latency: ~19.2s per query
- Retrieval time: <1s
- Generation time: ~18s
- Relevance score: 0.99/1.0
- Fluency score: 0.92/1.0

Scalability Projections

- Linear scaling for retrieval up to millions of documents
- Memory usage: ~1.5GB for LLM + vector embeddings
- Throughput: Limited by sequential LLM generation

6. Future Enhancements

1. Performance:

- Query result caching
- Batch query processing
- Model distillation for faster inference

2. Accuracy:

- Cross-encoder reranking
- Advanced query decomposition
- Feedback loop for continuous improvement

3. Features:

- Multi-modal support (images, tables)
- Real-time document updates
- Distributed deployment for scale