Technical Specification: Offline Multimodal Retrieval-Augmented Generation System

Document Version: *1.0*

Team ChittiGeNN

Table of Contents

1. Executive Summary

1.1 Project Purpose

The Offline Multimodal Retrieval-Augmented Generation (MMRAG) system is designed to provide a privacy-preserving, locally-operated knowledge management platform that integrates multiple data formats including documents (PDF, DOCX), images, audio recordings, and textual notes into a unified semantic knowledge base. The system enables natural language querying with citation-backed responses while maintaining complete data privacy through offline operation.

1.2 Bussiness Value(Optional)

The system addresses critical needs in sectors requiring data confidentiality including government agencies, healthcare institutions, legal firms, and research organizations. Key value propositions include:

- ***Data Privacy:*** *Complete offline operation ensures sensitive information never leaves the local environment*

- ***Multimodal Intelligence:*** *Unified semantic understanding across text, image, and audio modalities*
- ***Source Verification:*** *All responses include verifiable citations to original sources*
- ***Operational Efficiency:*** *Natural language interface reduces time-to-insight for complex information retrieval tasks*

1.3 Scope

This specification covers the development of a desktop application capable of processing, indexing, and retrieving information from heterogeneous data sources using advanced machine learning techniques, all while operating entirely offline on standard consumer hardware.

2. Project Overview

2.1 Vision Statement

To develop a comprehensive offline-first multimodal knowledge assistant that enables users to interact naturally with their document collections, regardless of format, while maintaining absolute data privacy and providing transparent, citation-backed responses.

2.2 Objectives

Primary Objectives:

- *Establish robust ingestion pipeline for PDF, DOCX, images, and audio files*
- *Implement unified semantic indexing across all supported modalities*
- *Deploy natural language query interface with voice input capability*
- *Ensure all responses include numbered citations with source navigation*
- *Maintain offline operation with local storage and processing*

Secondary Objectives:

- *Enable cross-modal linking and relationship discovery*
- *Provide temporal-based retrieval using metadata timestamps*
- *Implement conflict detection across different sources*
- *Support multilingual content processing and translation*

2.3 Target Users

Primary Users:

- *Government Intelligence Analysts*

- *Healthcare Researchers*
- *Legal Professionals*
- *Academic Researchers*
- *Corporate Security Teams*

Secondary Users:

- *Graduate Students*
- *Corporate Knowledge Workers*
- *Healthcare Practitioners*
- *Compliance Officers*

3. Requirements

3.1 Functional Requirements

3.1.1 Content Ingestion

- ***REQ-001:*** *System shall support drag-and-drop file upload for PDF, DOCX, JPG, PNG, MP3, WAV, and MP4 formats*
- ***REQ-002:*** *System shall monitor designated folders for automatic file ingestion*
- ***REQ-003:*** *System shall extract text content from PDF and DOCX files with metadata preservation*
- ***REQ-004:*** *System shall perform Optical Character Recognition (OCR) on image files*
- ***REQ-005:*** *System shall transcribe audio files using Automatic Speech Recognition (ASR)*
- ***REQ-006:*** *System shall extract and preserve file metadata including timestamps, author information, and creation dates*

3.1.2 Content Processing

- ***REQ-007:*** *System shall generate semantic embeddings for all text content using transformer-based models*
- ***REQ-008:*** *System shall create descriptive captions for images using vision-language models*
- ***REQ-009:*** *System shall perform speaker diarization for multi-speaker audio content*
- ***REQ-010:*** *System shall detect and translate non-English content to English while preserving original text*
- ***REQ-011:*** *System shall chunk large documents using semantic boundaries rather than arbitrary length limits*

3.1.3 Query Processing

- ***REQ-012:*** *System shall accept natural language queries through text input interface*
- ***REQ-013:*** *System shall support voice queries using offline speech-to-text processing*
- ***REQ-014:*** *System shall perform semantic similarity search across all indexed content*
- ***REQ-015:*** *System shall implement metadata-based filtering including date ranges and content types*
- ***REQ-016:*** *System shall support complex multi-hop queries requiring information synthesis*

3.1.4 Response Generation

- ***REQ-017:*** *System shall generate contextually appropriate responses using local large language models*
- ***REQ-018:*** *System shall include numbered citations for all factual claims in responses*
- ***REQ-019:*** *System shall provide direct navigation to source documents at specific pages or timestamps*
- ***REQ-020:*** *System shall indicate confidence levels for retrieved information*
- ***REQ-021:*** *System shall detect and report conflicting information across sources*

3.2 User Interface Requirements

3.2.1 Primary Interface

- ***REQ-022:*** *System shall provide intuitive search interface with auto-completion*
- ***REQ-023:*** *System shall display results with clear modality indicators (text, image, audio)*
- ***REQ-024:*** *System shall enable inline document viewing without external applications*
- ***REQ-025:*** *System shall provide audio playback controls with timestamp navigation*
- ***REQ-026:*** *System shall display image metadata and EXIF information*

3.2.2 File Management

- ***REQ-027:*** *System shall display ingestion progress with per-file processing status*
- ***REQ-028:*** *System shall provide file deletion with complete index cleanup*
- ***REQ-029:*** *System shall show storage utilisation and index statistics*
- ***REQ-030:*** *System shall enable bulk operations for file management*

4. Technical Architecture

4.1 System Architecture Overview

The system follows a modular architecture with clear separation between ingestion, processing, storage, and retrieval components. All components operate locally without external dependencies.

4.1.1 Core Components

- ***Ingestion Engine:*** *Handles file upload, monitoring, and initial preprocessing*
- ***Processing Pipeline:*** *Manages OCR, ASR, embedding generation, and indexing*
- ***Vector Store:*** *Maintains semantic embeddings and metadata using local vector database*
- ***Query Engine:*** *Processes natural language queries and performs similarity search*
- ***Response Generator:*** *Utilizes local LLM for answer synthesis with citation insertion*
- ***User Interface:*** *Web-based frontend for query input and result presentation*

4.1.2 Data Flow Architecture

File Input → Content Extraction → Semantic Processing → Vector Indexing → Query Processing → Response Generation → User Interface

4.2 Technology Stack

4.2.1 Core Technologies

- ***Programming Language:*** *Python 3.10+*
- ***Web Framework:*** *FastAPI for backend API, React for frontend interface*
- ***Vector Database:*** *FAISS or Chroma for local vector storage*
- ***Machine Learning Framework:*** *PyTorch for model inference*
- ***Database:*** *SQLite for metadata storage*

4.2.2 Machine Learning Models

- ***Text Embeddings:*** *sentence-transformers/all-mpnet-base-v2 or similar*
- ***Image Processing:*** *BLIP-2 or CLIP for image understanding*
- ***Speech Recognition:*** *OpenAI Whisper (small/medium variants)*
- ***Language Model:*** *Llama 2/3 7B or Mistral 7B (quantized versions)*
- ***OCR Engine:*** *PaddleOCR or Tesseract with language pack support*

5. Functional Requirements

5.1 Content Processing Requirements

5.1.1 Document Processing

The system must accurately extract text content from PDF and DOCX files while preserving formatting information and metadata. For PDF files, the system should handle both text-based and image-based PDFs, applying OCR when necessary.

5.1.2 Image Processing

All image files must undergo OCR processing to extract embedded text content. Additionally, the system should generate descriptive captions that capture the semantic content of images, enabling cross-modal retrieval between textual queries and visual content.

5.1.3 Audio Processing

Audio files require transcription using state-of-the-art ASR models, with additional processing for speaker identification in multi-speaker scenarios. The system should preserve timing information to enable timestamp-based navigation.

5.2 Retrieval Requirements

5.2.1 Semantic Search

The system must implement advanced semantic search capabilities that go beyond keyword matching to understand conceptual relationships between query terms and indexed content across all modalities.

5.2.2 Cross-Modal Retrieval

Users should be able to query using one modality and receive relevant results from other modalities. For example, a textual query about "financial projections" should return relevant charts, audio discussions, and document sections.

5.2.3 Temporal Filtering

The system must support time-based queries, allowing users to filter results by creation date, modification date, or content-specific temporal references.

5.3 Response Generation Requirements

5.3.1 Citation Integration

All generated responses must include inline citations that reference specific source documents, pages, timestamps, or image regions. Citations should be numbered and allow direct navigation to the referenced content.

5.3.2 Conflict Detection

When contradictory information exists across sources, the system must identify these conflicts and present them transparently to users rather than attempting to resolve them automatically.

5.3.3 Confidence Assessment

Each retrieved result should include a confidence score based on semantic similarity, source reliability, and content quality indicators.

6. Non-Functional Requirements

6.1 Performance Requirements

6.1.1 Response Time

- ***Query Processing:*** *Maximum 5 seconds for semantic search and response generation*
- ***File Ingestion:*** *Maximum 30 seconds per MB for document processing*
- ***Index Building:*** *Maximum 10 minutes for 1000 document corpus*

6.1.2 Throughput

- ***Concurrent Queries:*** *Support minimum 5 concurrent query sessions*
- ***Batch Processing:*** *Process minimum 100 files per hour during ingestion*

6.1.3 Resource Utilization

- ***Memory Usage:*** *Maximum 8GB RAM for standard operation*
- ***Storage Efficiency:*** *Index size should not exceed 20% of original content size*
- ***CPU Usage:*** *Maximum 80% utilization during active processing*

6.2 Scalability Requirements

6.2.1 Content Volume

- ***Document Capacity:*** *Support a minimum 10,000 documents*
- ***Image Capacity:*** *Support a minimum 50,000 images*
- ***Audio Capacity:*** *Support a minimum 1,000 hours of audio content*

6.2.2 Performance Scaling

- ***Linear Scaling:*** *Search performance degradation should be logarithmic with content volume*
- ***Incremental Processing:*** *New content addition should not require full index rebuild*

6.3 Reliability Requirements

6.3.1 Availability

- ***System Uptime:*** *99.9% availability during operational hours*
- ***Error Recovery:*** *Automatic recovery from processing failures*
- ***Data Integrity:*** *Zero data loss during normal operations*

6.3.2 Fault Tolerance

- ***Graceful Degradation:*** *System should continue operating with reduced functionality during component failures*
- ***Error Handling:*** *Comprehensive error reporting with actionable recovery suggestions*

6.4 Security Requirements

6.4.1 Data Protection

- ***Encryption at Rest:*** *All stored data must be encrypted using AES-256 encryption*
- ***Memory Protection:*** *Sensitive data in memory should be cleared after processing*
- ***Access Control:*** *Role-based access control for multi-user scenarios*

6.4.2 Privacy Compliance

- ***Data Locality:*** *All processing must occur locally without external communication*
- ***Audit Trail:*** *Complete logging of data access and modification activities*
- ***Secure Deletion:*** *Cryptographic deletion of user-requested data removal*

7. System Design

7.1 Component Architecture

7.1.1 Ingestion Layer

The ingestion layer handles file input through multiple channels including drag-and-drop interface, folder monitoring, and API endpoints. Each file type follows a specialized processing pipeline:

PDF Processing Pipeline:

1. *Text extraction using PyMuPDF or pdfplumber*
2. *Image extraction for embedded figures*
3. *Metadata extraction including creation date, author, and title*
4. *Page-level segmentation for precise citation references*

Image Processing Pipeline:

1. *Format normalisation and compression*
2. *OCR processing using PaddleOCR with multi-language support*
3. *Image captioning using BLIP-2 or a similar vision-language model*
4. *EXIF metadata extraction for temporal information*

Audio Processing Pipeline:

1. *Format conversion to standard WAV format*
2. *Noise reduction and audio enhancement*
3. *Speech-to-text using the Whisper model*
4. *Speaker diarization using pyannote-audio*
5. *Language detection and optional translation*

7.1.2 Processing Layer

The processing layer transforms raw content into searchable semantic representations:

Text Processing:

- *Semantic chunking using sentence boundaries and topic modelling*
- *Named entity recognition for improved metadata*
- *Embedding generation using transformer models*
- *Language detection and translation as needed*

Cross-Modal Processing:

- *Image-text alignment using CLIP embeddings*
- *Audio-text synchronisation with timestamp preservation*
- *Multi-modal fusion for unified semantic representation*

7.1.3 Storage Layer

The storage layer manages both vector embeddings and original content:

Vector Storage:

- *FAISS index for high-performance similarity search*
- *Metadata database using SQLite for structured information*
- *Incremental index updates without full rebuilds*

Content Storage:

- *Original files preserved with encryption*
- *Processed content cached for quick access*
- *Backup and recovery mechanisms*

7.1.4 Retrieval Layer

The retrieval layer handles query processing and result ranking:

Query Processing:

- *Natural language understanding using sentence transformers*
- *Query expansion using synonyms and related terms*
- *Multi-modal query support (text, image, audio inputs)*

Ranking and Fusion:

- *Semantic similarity scoring*
- *Cross-encoder reranking for improved precision*
- *Diversity algorithms to avoid redundant results*
- *Temporal relevance scoring*

7.1.5 Response Layer

The response layer generates final answers with citations:

Language Model Integration:

- *Local LLM deployment using llama.cpp or similar*
- *Prompt engineering for citation-aware responses*
- *Context window management for long documents*
- *Hallucination detection and mitigation*

7.2 Data Flow Design

7.2.1 Ingestion Flow

User Input → File Validation → Content Extraction → Preprocessing → Embedding Generation → Index Storage → Status Update

7.2.2 Query Flow

User Query → Query Processing → Similarity Search → Result Ranking → Context Assembly → LLM Generation → Citation Integration → Response Delivery

7.2.3 Cross-Modal Linking Flow

Content Analysis → Temporal Alignment → Semantic Correlation → Relationship Mapping → Link Storage → Query-Time Fusion

8. Data Management

8.1 Data Models

8.1.1 Document Model

Document {

document_id: UUID

filename: String

file_path: String

content_type: Enum[PDF, DOCX, IMAGE, AUDIO, TEXT]

original_size: Integer

processed_size: Integer

creation_date: DateTime

ingestion_date: DateTime

last_modified: DateTime

metadata: JSON

processing_status: Enum[PENDING, PROCESSING, COMPLETED, FAILED]

checksum: String

}

8.1.2 Content Chunk Model

ContentChunk {

chunk_id: UUID

document_id: UUID (Foreign Key)

content_text: Text

chunk_type: Enum[TEXT, IMAGE_CAPTION, AUDIO_TRANSCRIPT]

start_position: Integer

end_position: Integer

page_number: Integer (nullable)

timestamp: Float (nullable)

embedding_vector: Float[]

confidence_score: Float

language: String

metadata: JSON

}

8.1.3 Cross-Modal Link Model

CrossModalLink {

link_id: UUID

source_chunk_id: UUID (Foreign Key)

target_chunk_id: UUID (Foreign Key)

relationship_type: Enum[TEMPORAL, SEMANTIC, REFERENCE]

confidence_score: Float

evidence: JSON

creation_date: DateTime

}

8.2 Storage Architecture

8.2.1 File System Organisation

```
/data/
  /documents/
    /originals/      # Original uploaded files
    /processed/      # Processed content cache
    /thumbnails/     # Image thumbnails
  /indexes/
    /vectors/        # FAISS vector indexes
    /metadata/       # SQLite database files
  /models/
    /embeddings/     # Pre-trained embedding models
    /llm/            # Local language models
  /logs/             # Application and audit logs
  /backups/          # Automated backup storage
```

8.2.2 Vector Index Structure

The vector index maintains separate subindices for different content types while enabling unified search across modalities:

- ***Text Index:*** *Semantic embeddings of document chunks*
- ***Image Index:*** *Visual embeddings and caption embeddings*
- ***Audio Index:*** *Transcript embeddings with temporal markers*
- ***Unified Index:*** *Cross-modal embeddings for integrated search*

8.2.3 Metadata Database Schema

SQLite database with optimised schema for quick metadata retrieval:

- ***Documents Table:*** *Core document information and processing status*
- ***Chunks Table:*** *Content segments with position and embedding references*
- ***Links Table:*** *Cross-modal relationships and temporal alignments*
- ***Queries Table:*** *Query history and performance metrics*

- ***Users Table:*** *User sessions and access control (if applicable)*