# Metis RAG Project Planning

## Overview

Metis RAG is an advanced Retrieval Augmented Generation system designed to provide accurate, contextual responses by combining conversational AI with document retrieval capabilities. This document outlines the high-level architecture, design decisions, and technical specifications that guide the development of the system.

## Architecture

### Core Components

1. **FastAPI Backend**
   * RESTful API endpoints for chat, document management, and system operations
   * Asynchronous request handling for improved performance
   * Middleware for authentication, logging, and error handling
2. **RAG Engine**
   * Document retrieval system using vector embeddings
   * Context-aware query processing
   * Response generation with source attribution
   * Memory system for conversation persistence
3. **Vector Database (ChromaDB)**
   * Storage for document embeddings
   * Similarity search capabilities
   * Metadata filtering and retrieval
   * Optimized caching for performance
4. **Relational Database (PostgreSQL)**
   * User and authentication data
   * Document metadata storage
   * Conversation and message history
   * Memory buffer for explicit information storage
5. **LLM Integration (Ollama)**
   * Text generation capabilities
   * Context-aware responses
   * Support for multiple models
   * Streaming response generation
6. **Background Task System**
   * Asynchronous document processing
   * Task prioritization and management
   * Resource monitoring and allocation
   * Task dependency handling
7. **Authentication System**
   * JWT-based authentication
   * Role-based access control
   * Resource ownership validation
   * Secure password handling

## Technical Stack

* **Backend**: Python 3.10+, FastAPI, Pydantic
* **Database**: PostgreSQL with SQLAlchemy ORM, Alembic for migrations
* **Vector Store**: ChromaDB with optimized caching
* **LLM Integration**: Ollama API with enhanced prompt engineering
* **Document Processing**: LangChain with custom chunking strategies
* **Frontend**: HTML, CSS, JavaScript with fetch API
* **Deployment**: Docker, Docker Compose
* **Testing**: Pytest, unittest

## Design Principles

1. **Accuracy First**: The system prioritizes factual accuracy and source attribution over generative capabilities.
2. **Explicit Memory**: The system maintains explicit memory of user information and conversation context.
3. **Modularity**: Components are designed to be modular and replaceable.
4. **Scalability**: The architecture supports horizontal scaling for increased load.
5. **Security**: Authentication and authorization are integrated at all levels.
6. **Performance**: Optimized for low latency responses and efficient resource usage.
7. **Transparency**: Clear source attribution and audit trails for all responses.

## Key Features

1. **Dynamic Chunking Strategy**
   * Intelligent document analysis for optimal chunking
   * Adaptive parameters based on document type and content
   * Preservation of semantic meaning and document structure
2. **Query Refinement and Retrieval Enhancement**
   * Query analysis and refinement for improved retrieval
   * Relevance evaluation of retrieved chunks
   * Re-ranking based on query relevance
3. **Response Quality Pipeline**
   * Response synthesis with proper source attribution
   * Factual accuracy verification
   * Hallucination detection and removal
   * Iterative refinement for quality improvement
4. **Conversation Memory**
   * Persistent memory across conversation turns
   * Explicit memory commands for information storage
   * Implicit memory retrieval for contextual awareness
   * Memory categorization and prioritization
5. **Multi-User Support**
   * User authentication and access control
   * Resource ownership and permission management
   * Isolated data and conversation contexts

## Constraints and Considerations

1. **Performance Constraints**
   * Response time targets: < 3 seconds for typical queries
   * Document processing time: < 30 seconds for standard documents
   * Memory usage: < 4GB RAM for core services
2. **Scalability Considerations**
   * Horizontal scaling for increased user load
   * Vector database partitioning for large document collections
   * Background task distribution for processing-intensive operations
3. **Security Requirements**
   * JWT token-based authentication
   * Password hashing with bcrypt
   * CORS protection and security headers
   * Input validation and sanitization
4. **Deployment Requirements**
   * Docker containerization for all components
   * Environment-based configuration
   * Health checks and monitoring
   * Backup and recovery procedures

## Development Roadmap

1. **Phase 1: Core RAG Functionality**
   * Basic document processing and retrieval
   * Simple chat interface
   * Initial LLM integration
2. **Phase 2: Enhanced RAG Capabilities**
   * Dynamic chunking strategies
   * Query refinement
   * Response quality improvements
3. **Phase 3: Memory and Conversation Persistence**
   * Conversation history tracking
   * Explicit memory commands
   * Context-aware responses
4. **Phase 4: Authentication and Multi-User Support**
   * User registration and login
   * Role-based access control
   * Resource ownership
5. **Phase 5: Performance Optimization and Scaling**
   * Caching improvements
   * Asynchronous processing
   * Resource monitoring and management

## Implementation Guidelines

1. **Code Organization**
   * Follow repository pattern for data access
   * Use dependency injection for service composition
   * Implement clear separation of concerns
2. **Error Handling**
   * Comprehensive exception handling
   * Detailed error logging
   * User-friendly error messages
3. **Testing Strategy**
   * Unit tests for core components
   * Integration tests for service interactions
   * End-to-end tests for critical user flows
   * Performance tests for optimization
4. **Documentation**
   * API documentation with OpenAPI
   * Code documentation with docstrings
   * User guides and tutorials
   * Architecture and design documentation

## Conclusion

This planning document serves as a high-level guide for the development of the Metis RAG system. It outlines the architecture, technical stack, design principles, and key features that define the system. Developers should refer to this document when making implementation decisions to ensure alignment with the overall vision and goals of the project.