Technical Report: Agentic RAG System with LangGraph

System Architecture and Design Decisions

1. Overall Architecture

The system follows a modular, layered architecture designed for scalability and maintainability:

- **Frontend Layer**: Streamlit-based web interface for user interaction
- **Orchestration Layer**: System manager coordinating all components
- **Agentic Layer**: LangGraph-based RAG agent with autonomous workflows
- **Processing Layer**: Document processing and vector storage
- **LLM Layer**: Multi-provider language model management

2. Key Design Decisions

2.1 LangGraph Integration

- **Rationale**: LangGraph provides robust workflow management and state persistence
- **Implementation**: 5-node workflow (retrieve → analyze → generate → evaluate → improve)
- **Benefits**: Autonomous decision-making, conditional routing, and self-correction

2.2 Multi-LLM Strategy

- **Primary**: Google AI Studio/Gemini API (free tier, high quality)
- **Fallback**: OpenAI API (reliability, consistency) or Ollama Mistral (local, offline)
- **Implementation**: Automatic provider switching with retry logic
- **Benefits**: Redundancy, cost optimization, performance flexibility

2.3 Vector Storage Selection

- **Choice**: Chroma (in-memory with persistence)
- **Alternatives Considered**: FAISS (performance), Qdrant (scalability)
- **Decision Factors**: Ease of integration, metadata support, development speed

2.4 Document Processing Strategy

- **Chunking**: Recursive character splitting with overlap
- **Size**: 1000 characters (optimal for medical text comprehension)
- **Overlap**: 200 characters (maintains context continuity)
- **Metadata**: Comprehensive tracking for source attribution

3. Pipeline Components and Interactions

3.1 Document Processing Pipeline

3.2 Query Processing Pipeline

3.3 Component Interactions

- **SystemManager** orchestrates the entire pipeline
- **DocumentProcessor** handles PDF extraction and chunking
- **VectorStore** manages embeddings and similarity search
- **RAGAgent** executes the LangGraph workflow
- **LLMManager** provides LLM access with fallback mechanisms

4. System Limitations

4.1 Functional Limitations

- **Language Support**: Optimized for English/German medical texts
- **Document Types**: Currently limited to PDF format
- **Query Complexity**: Best suited for factual and analytical queries
- **Medical Accuracy**: Responses should be verified by professionals

4.2 Operational Limitations

- **Cost**: API usage costs for LLM services
- **Network**: Requires stable internet connection

- **Maintenance**: Regular updates needed for dependencies
- **Expertise**: Requires technical knowledge for deployment

Conclusion

The Agentic RAG System successfully demonstrates the potential of LangGraph for building autonomous, self-improving document analysis systems. The modular architecture provides flexibility for future enhancements, while the multi-LLM approach ensures reliability and performance. The system effectively processes medical documents and provides evidence-based responses with proper source citations, making it suitable for research and educational purposes in the medical domain.

Key achievements include:

- Successful implementation of agentic behavior using LangGraph
- Robust document processing and vector storage
- Intelligent fallback mechanisms for LLM providers
- Comprehensive source tracking and citation management
- User-friendly Streamlit interface for testing and evaluation

The system serves as a foundation for more advanced medical AI applications and demonstrates the viability of autonomous RAG systems for complex document analysis tasks.