04 Medical RAG Chatbot Project

Advanced Medical RAG Chatbot Project

Problem Statement

Build an intelligent **multi-modal medical assistant** using **Retrieval Augmented Generation (RAG)** that can answer health-related questions through multiple interaction modes. Your system should process medical documents, create a searchable knowledge base, and provide contextually relevant answers via text, voice, and image inputs. The project includes two main components: a **Document-based RAG Chatbot** and an **AI Doctor Assistant** with multi-modal capabilities.

Learning Objectives

By completing this project, you will:

- ✅ Implement a complete RAG pipeline from scratch
- ✅ Master LangChain/LangGraph for orchestrating AI workflows
- ✅ Build vector-based knowledge retrieval systems
- ✅ Create multi-modal AI applications with text, voice, and image processing
- ✅ Handle large document processing and text chunking strategies
- ✅ Integrate speech-to-text and text-to-speech technologies
- ✅ Implement vision-language models for medical image analysis
- ✅ Build interactive applications using Streamlit

Core Requirements

App 1: Document-Based RAG Chatbot

Phase 1: Document Processing & Knowledge Base Creation

1. **PDF Document Ingestion**
   - Load and extract text from medical PDF documents
   - Clean and filter extracted content
   - Handle multiple document formats gracefully
2. **Text Chunking Strategy**
   - Implement recursive character text splitting
   - Configure chunk size: 500 tokens with 20-token overlap
   - Preserve context across chunk boundaries
   - Demonstrate different chunking strategies and their impact

Phase 2: Vector Knowledge Base Setup

3. **Embedding Generation**

- Use any embedding model of your choice (e.g., Sentence Transformers, OpenAI embeddings, etc.)
- Convert text chunks to vector embeddings
- Implement batch processing for efficiency

4. **Vector Storage & Retrieval**
   - Set up ChromaDB vector database (local/persistent)
   - Store embeddings with metadata
   - Implement semantic similarity search (top-k retrieval)
   - Add functionality to update/expand the knowledge base

Phase 3: RAG Pipeline Implementation

5. **LangChain Integration**
   - Build complete RAG chain using LangChain/LangGraph
   - Implement query-to-vector conversion
   - Create context-aware prompt templates
   - Handle LLM integration (any LLM of your choice: OpenAI, Ollama, HuggingFace, etc.)
6. **Response Generation**
   - Generate contextually relevant medical responses
   - Implement proper citation of source chunks
   - Add confidence scoring for answers

App 2: AI Doctor Assistant (Multi-Modal)

Phase 4: Voice Integration

7. **Speech-to-Text Processing**
   - Implement real-time voice input capture
   - Convert speech to text using models like Whisper, SpeechRecognition, or cloud APIs
   - Handle multiple languages and medical terminology
   - Add noise filtering and audio preprocessing
8. **Text-to-Speech Response**
   - Convert AI responses to natural speech
   - Use TTS libraries like gTTS, Azure Speech, or ElevenLabs
   - Implement different voice options and speaking rates
   - Add audio playback controls in the interface

Phase 5: Vision and Image Analysis

9. **Medical Image Processing**
   - Accept various image formats (JPG, PNG, DICOM, etc.)
   - Implement image preprocessing and enhancement
   - Handle medical imaging standards and metadata
   - Add image resize and optimization for model processing

10. **Vision-Language Model Integration**
    - Integrate vision-language models (GPT-4V, LLaVA, BLIP-2, etc.)
    - Process medical images with contextual understanding
    - Generate detailed medical observations and analysis
    - Combine image analysis with text-based knowledge retrieval
11. **Multi-Modal Response Generation**
    - Combine text queries, voice input, and image analysis
    - Generate comprehensive responses using multiple information sources
    - Provide visual annotations and highlighting on uploaded images
    - Create detailed medical reports with image references

Phase 6: Integrated Interface

12. **Advanced Streamlit Application**
    - Create tabbed interface for different interaction modes
    - Text chat interface with conversation history
    - Voice recording and playback functionality
    - Image upload and display with analysis results
    - Multi-modal conversation flow (text + voice + image)
    - Real-time processing indicators and progress bars

Bonus Challenges (Extra Features)

Advanced RAG Features

1. **Conversational Memory**
    - Implement conversation buffer memory across all interaction modes
    - Maintain context across text, voice, and image conversations
    - Create persistent user sessions and conversation history
2. **Multi-Modal RAG Processing**
    - Extract and process images/tables from PDF documents
    - Combine textual and visual information retrieval
    - Implement cross-modal similarity search
3. **Advanced Retrieval Techniques**
    - Implement re-ranking of retrieved documents
    - Add query expansion and reformulation
    - Create hybrid retrieval combining text and image embeddings

AI Doctor Enhancement

4. **Medical Specialization**
    - Implement specialty-specific AI doctors (cardiology, dermatology, radiology, etc.)
    - Create domain-specific knowledge bases and models
    - Add specialty-specific image analysis capabilities

5. **Clinical Decision Support**
   - Implement symptom checker functionality
   - Add differential diagnosis suggestions
   - Create treatment recommendation systems (with appropriate disclaimers)
6. **Medical Image Analysis**
   - Implement specific medical image analysis (X-rays, MRIs, skin conditions, etc.)
   - Add image annotation and markup capabilities
   - Create comparative analysis with similar cases

Performance & User Experience

7. **Real-Time Processing**
   - Implement streaming responses for long queries
   - Add real-time voice conversation capabilities
   - Create progressive image analysis with immediate feedback
8. **Multi-Language Support**
   - Support multiple languages for voice and text input
   - Implement medical terminology translation
   - Add language-specific medical knowledge bases
9. **Advanced UI/UX Features**
   - Create mobile-responsive design
   - Add dark/light theme options
   - Implement keyboard shortcuts and accessibility features
   - Add export functionality for conversations and reports

Implementation Guidelines

The project should be structured with clean, modular code organization:
medical-rag-chatbot/ ├── src/ │ ├── document_processor.py # PDF loading & chunking │ ├── knowledge_base.py # Vector DB operations │ ├── rag_pipeline.py # LangChain RAG implementation │ ├── prompts.py # Prompt templates │ └── utils.py # Helper functions ├── data/ │ └── medical_documents/ # PDF files ├── app.py # Streamlit interface ├── requirements.txt ├── .env # API keys (not in repo) └── README.md

---

Sample Test Cases

Test Case 1: Text-Based RAG Query

Input: "What are the symptoms of diabetes?"
Expected Behavior:

1. Convert query to vector embedding
2. Retrieve top-3 relevant chunks about diabetes
3. Generate contextual response using LLM
4. Return answer with source citations

Output Format:

```
{
    "answer": "Diabetes symptoms include frequent urination, excessive thirst ... ",
    "sources": ["Medical_Book_Chapter_5_Page_123",
"Medical_Book_Chapter_8_Page_89"],
    "confidence_score": 0.87,
    "response_type": "text"
}
```

Test Case 2: Voice Query Processing

Input: Audio file with spoken question "What should I do for a headache?"

Expected Behavior:

1. Convert speech to text using STT
2. Process text query through RAG or direct LLM
3. Generate text response
4. Convert response to speech using TTS
5. Return both text and audio response

Test Case 3: Medical Image Analysis

Input: Image of a skin condition + text query "What could this rash be?"

Expected Behavior:

1. Process uploaded image through vision-language model
2. Extract visual features and medical observations
3. Combine image analysis with text query
4. Retrieve relevant medical knowledge if needed
5. Generate comprehensive analysis with visual and textual information

Output Format:

```
{
    "image_analysis": "The image shows a red, scaly rash with defined borders ... ",
    "text_response": "Based on the image and your description ... ",
    "confidence_score": 0.82,
    "recommendations": ["Consult dermatologist", "Avoid scratching"],
    "response_type": "multi_modal"
}
```

Test Case 4: Multi-Modal Conversation Flow

Scenario: User uploads chest X-ray image, asks via voice "Is this pneumonia?", then follows up with text "What treatment options are available?"

Expected: Maintain conversation context across different input modalities and provide coherent, contextual responses.

Technical Specifications

Required Libraries

```
# Core Libraries
langchain==0.1.0
langchain-community==0.0.10
streamlit==1.29.0
chromadb==0.4.22
pypdf==3.17.4
python-dotenv==1.0.0

# LLM & Embedding Models (Choose any)
# Examples: openai, ollama, huggingface-hub, sentence-transformers, etc.

# Utilities
pandas==2.1.4
numpy==1.26.2
```

Evaluation Criteria

Code Quality

- Clean, modular code structure
- Proper error handling
- Good documentation and comments
- Following Python best practices

Functionality

- All core features working correctly
- Proper RAG pipeline implementation
- Accurate retrieval and generation
- User interface functionality

Innovation

- Creative solutions to challenges
- Implementation of bonus features
- Optimization techniques
- Novel approaches to common problems

Documentation

- Clear README with setup instructions
- Code documentation
- Usage examples
- Performance analysis

Deliverables

1. **Complete Codebase**
   - All source files with proper modular organization
   - Working Streamlit application with both RAG and AI Doctor modes
   - Requirements.txt with exact versions
   - Configuration files and environment setup
2. **Demo Data & Models**
   - At least 2 medical PDF documents for RAG knowledge base
   - Sample medical images for testing vision capabilities
   - Sample audio files for voice testing (optional)
   - Documentation of model choices and setup instructions
3. **Documentation**
   - Comprehensive README with setup and usage instructions
   - Technical report (3-5 pages) explaining:
     - Architecture decisions for both systems
     - Choice of LLM, embedding, vision, and voice models
     - Multi-modal integration approach
     - Challenges faced and solutions implemented
     - Performance analysis and evaluation results
     - Future improvement suggestions
4. **Demo Materials**
   - **Video Demo** (5-8 minutes) showing:
     - RAG chatbot functionality with document queries
     - Voice input/output demonstration
     - Image analysis capabilities
     - Multi-modal conversation example
     - UI navigation and features
   - **Live Demo Setup** (optional): Instructions for live demonstration

Getting Started

1. **Environment Setup**

```
# Create virtual environment
conda create -n advanced-medical-ai python=3.11
conda activate advanced-medical-ai

# Install dependencies
pip install -r requirements.txt

# For audio processing (may require system-level installation)
# On Windows: Install PyAudio wheel
# On Linux: sudo apt-get install portaudio19-dev
# On Mac: brew install portaudio
```

2. **Model Setup**

```
# Download Whisper model (if using local)
python -c "import whisper; whisper.load_model('base')"

# Setup local LLMs (if using Ollama)
ollama pull llama2  # or your preferred model
```

3. **Configuration**

```
# Create .env file with your API keys
OPENAI_API_KEY=your_key_here
# Add other API keys as needed
```

4. **Run Application**

```
streamlit run app.py
```

Application Structure

The application should have a main interface with two primary modes:

🔍 RAG Chatbot Mode

- Document-based question answering
- Knowledge base management
- Source citation and references
- Conversation history

👨‍⚕️ AI Doctor Mode

- **Text Chat**: Traditional text-based medical consultation
- **Voice Chat**: Speech-to-text input with text-to-speech responses
- **Image Analysis**: Upload and analyze medical images
- **Multi-Modal**: Combined text, voice, and image interaction

Good luck building your advanced medical AI assistant! 🚀

Note: This project focuses on educational purposes. Any medical information generated should not be used for actual medical decisions. Always consult qualified healthcare professionals for medical advice.