Agentic Healthcare Assistant for Medical Task Automation

Course-end Project 3 — LangGraph Multi-Agent + RAG + Memory + Streamlit

1. Problem Scenario

Healthcare administration requires coordinating multiple tasks such as appointment scheduling, maintaining patient histories, and retrieving disease information. These tasks are often handled in siloed systems with limited automation, causing inefficiencies and fragmented patient experiences.

2. Project Objectives

This project develops an Agentic Healthcare Assistant that can:

• Book medical appointments by discovering slots and scheduling based on intent and availability

• Manage medical records by adding/updating structured/unstructured history

• Retrieve medical histories by summarizing past diagnoses, treatments, and alerts

• Perform medical information retrieval using a RAG knowledge base built from trusted documents (and extensible to WHO/Medline sources)

3. Solution Overview

The solution uses an agentic architecture orchestrated with LangGraph. Specialized agents handle intent routing, appointment booking, record updates, history retrieval, and medical knowledge retrieval (RAG). A final response agent consolidates outputs and produces a safe, user-friendly response.

4. System Architecture

High-level workflow:

1) Intent Router Agent — classifies the message into appointment/records/history/medical info

2) Sentiment Agent — detects tone for personalization

3) Task Agent — executes the appropriate action:

• Appointment Agent: slot discovery + booking into DB

• Records Agent: add/update patient history

• History Agent: retrieve & summarize patient records

• Medical Info Agent: triggers RAG retrieval from medical PDFs

4) Retrieval Node (RAG) — similarity search over indexed medical documents

5) Response Agent — generates final response using intent, sentiment, action output, and retrieved context

5. Data & Storage

• SQLite database stores patients, doctors, appointments, and patient records.

• Demo data seeding loads sample patient reports (PDF) as imported records.

• Knowledge base uses FAISS vector store built from PDFs in data/pdf_sources.

6. RAG Pipeline

PDF documents are loaded, split into chunks, embedded with SentenceTransformers, and stored in FAISS. At runtime, medical info questions trigger similarity search to retrieve the most relevant passages, which ground the LLM response and reduce hallucinations.

7. Safety & Guardrails

The assistant follows healthcare-safe guidelines:

• No diagnosis or prescriptive medical decisions

• Educational information only, with recommendation to consult clinicians

• Emergency symptom guidance (seek urgent care) when needed

• Patient data shown only from the local database records

8. User Interface

A Streamlit app provides:

• Assistant tab: chat input and outputs (intent, sentiment, action output, final response)

• Dashboard tab: view patients and appointments

• Knowledge Base tab: list RAG source PDFs and KB status

9. Evaluation / Testing Scenarios

Recommended test cases:

• Appointment: “Book an appointment with Cardiology next Monday morning for Anjali.”

• Update record: “Add record for patient Anjali: Diabetes diagnosed 2019, on Metformin.”

• Retrieve history: “Show medical history for patient Ramesh.”

• Medical info: “What are symptoms and treatment options for hypertension?”

Outputs should be correct, grounded in DB/KB, and safety-compliant.

10. Conclusion & Future Enhancements

This project demonstrates how agentic AI + RAG can automate healthcare workflows and improve patient experience. Future enhancements include integration with FHIR/HL7 EHR systems, role-based access control, and live medical source retrieval from WHO/Medline APIs.