NephroConnect-AI

1. Introduction & Project Overview

The primary objective of this project was to build a functional, multi-agent AI chatbot system to assist patients with post-discharge care. The system is designed to provide personalized information based on a patient's discharge report and answer general medical questions related to their condition (nephrology) using a Retrieval-Augmented Generation (RAG) pipeline.

This POC successfully demonstrates core GenAI capabilities, including:

- **Multi-Agent Orchestration:** A Receptionist Agent for patient intake and a Clinical AI Agent for specialized queries.
- **Retrieval-Augmented Generation (RAG):** Answering questions using a dedicated nephrology knowledge base.
- **Tool Usage:** Integration of a patient data retrieval tool and a live web search tool.
- **End-to-End System:** A complete, interactive application with a simple web frontend and a robust backend.

---

2. System Flow

$$\begin{tikzcd}\text{Web Client} \arrow[r] & \text{FastAPI App} \arrow[r] & \text{Agent System}\end{tikzcd}$$

How It Works: The User Journey

1. **Initial Interaction:** The user opens the web interface (HTML/CSS/JS) and is greeted by the system. They enter their name.
2. **API Call:** The frontend sends the user's message to the backend FastAPI server.
3. **Receptionist Agent:** The request is routed to the Receptionist Agent. It identifies the user's intent is to provide their name.
4. **Patient Data Retrieval:** The Receptionist Agent uses the PatientDataRetrievalTool to look up the user's name in the SQLite database.
5. **Personalized Greeting:** Upon successfully finding the patient's discharge report, the agent composes a personalized greeting, confirming their diagnosis and asking a follow-up question (e.g., "How are you feeling today?").

6. **Query Routing:**
    - If the user asks a non-medical, conversational question, the Receptionist Agent handles it.
    - If the user asks a medical question (e.g., "Why are my legs swelling?"), the Receptionist Agent recognizes the need for clinical expertise and hands the conversation over to the Clinical AI Agent.
7. **Clinical AI Agent:** This agent receives the query and the patient's context (e.g., their diagnosis of Chronic Kidney Disease).
    - **RAG:** For questions about the patient's condition, the agent uses the RAG Tool. This tool queries the FAISS vector store containing the nephrology reference material to find relevant information and generate a cited answer.
    - **Web Search:** If the query is about recent news or information not found in the static knowledge base (e.g., "latest research on SGLT2 inhibitors"), the agent uses the Web Search Tool to get up-to-date information.
8. **Response to User:** The generated response, complete with citations and a medical disclaimer, is sent back through the API to the frontend and displayed to the user.
9. **Logging:** Every step of this process—from user input to agent decisions and tool outputs—is recorded in a timestamped log file for debugging and analysis.

---

3. Component Implementation Details

Data Setup

- **Patient Reports:** 25+ dummy patient reports were created in JSON format, following the structure provided in the assignment. These were loaded into a **JSON** database table for structured, efficient retrieval.
- **Knowledge Base:** A nephrology reference text was processed. It was split into semantic chunks of manageable size (e.g., 500 characters with overlap).
- **Vector Embeddings:** Each text chunk was converted into a numerical vector using **Google's GoogleGenerativeAIEmbeddings** model. This model was chosen for its high performance and compatibility with the Google LLM ecosystem.
- **Vector Store:** The generated embeddings and their corresponding text chunks were stored in a **FAISS (Facebook AI Similarity Search)** index. FAISS is an in-

memory library, making it extremely fast for the semantic search required by the RAG system.

Multi-Agent System (LangChain)

A custom multi-agent system was orchestrated using LangChain.

- **Receptionist Agent:**
    - **Purpose:** Manages patient intake, identification, and initial interaction.
    - **Tools:** Patient Data Retrieval Tool.
    - **Logic:** Its core prompt instructs it to be friendly, ask for the patient's name, use its tool to fetch their data, and route any medical questions to the Clinical Agent.
- **Clinical AI Agent:**
    - **Purpose:** To provide accurate, evidence-based answers to medical questions.
    - **Tools:** RAG Tool (querying FAISS) and Web Search Tool.
    - **Logic:** Its prompt guides it to first attempt to answer questions using the internal RAG knowledge base. It is instructed to use the web search tool only for topics requiring recent information. It must always provide citations and include a medical disclaimer in its final answer.

Frontend

A simple, clean user interface was built using **HTML, CSS, and JavaScript**.

- A chat box allows users to input messages.
- JavaScript handles the fetch API calls to the FastAPI backend.
- The conversation history is dynamically updated in the browser, clearly distinguishing between user messages and AI responses.
- A medical disclaimer is permanently visible on the interface.

4. Conclusion

This Proof of Concept successfully fulfills all the requirements outlined in the assignment. It demonstrates a functional multi-agent system capable of personalized patient interaction and evidence-based medical Q&A through a RAG pipeline. The architectural choices prioritize rapid development and core functionality, resulting in a robust and well-documented system.