Healthcare Appointment Scheduler AI: A Deep Dive into Design, Architecture, and Implementation

Index

1. Introduction

1.1 Overview of the System

AI Healthcare Appointment Assistant
Powered by Mistral 7B (via OpenRouter) + FastAPI + LangChain

The Healthcare Appointment Scheduler AI is an intelligent assistant built using LangChain, FastAPI, and Streamlit. It facilitates scheduling appointments by leveraging a Mistral 7B Instruct Model for natural language understanding and decision-making. The system interacts with users in a conversational format, retrieving available doctors, slots, and booking appointments dynamically.

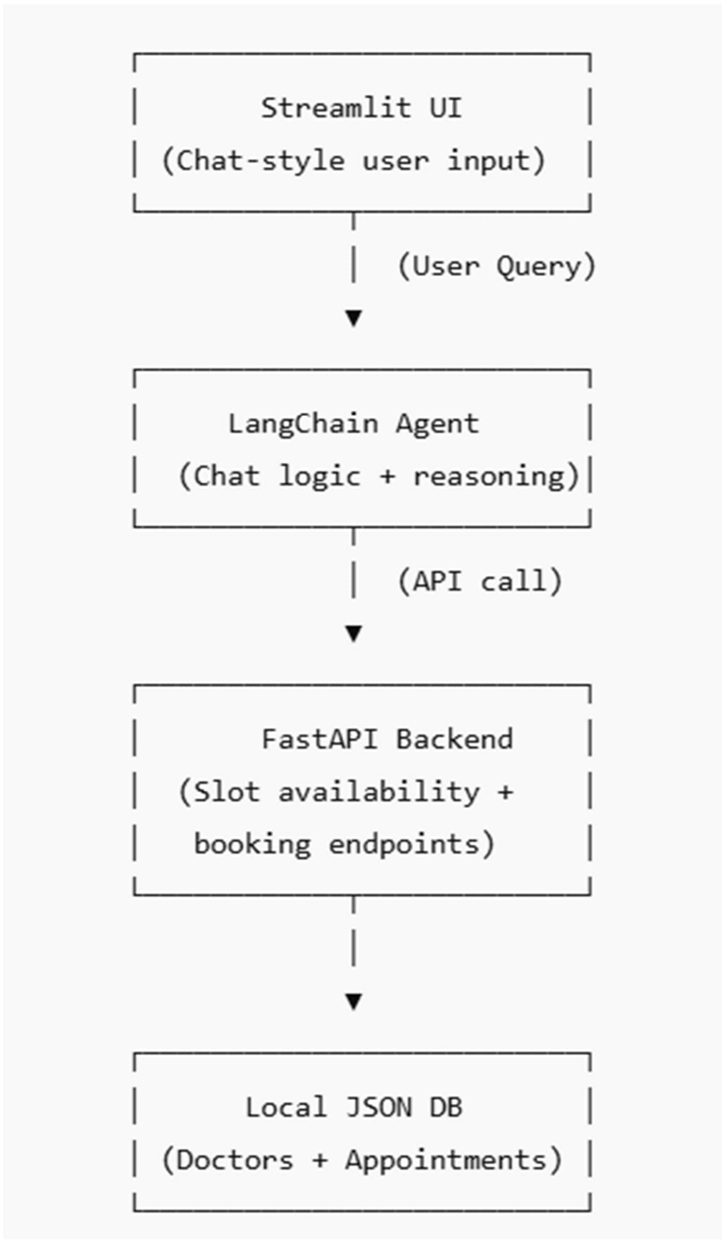
1.2 Purpose and Scope

The system aims to automate the appointment scheduling process for healthcare providers and patients. By using AI-driven reasoning, it provides a user-friendly interface for checking available doctors and booking appointments in real-time.

1.3 Benefits of the System

- Efficiency: Automates the entire scheduling process.
- User-Friendly: Easy-to-use conversational interface.
- Real-Time Updates: Provides up-to-date availability of doctors and slots.

2. System Architecture and Design



```
┌─────────────────────────────┐
│         Streamlit UI        │
│ (Chat-style user input)     │
└─────────────────────────────┘
              │
              │   (User Query)
              ▼
┌─────────────────────────────┐
│       LangChain Agent       │
│  (Chat logic + reasoning)   │
└─────────────────────────────┘
              │
              │   (API call)
              ▼
┌─────────────────────────────┐
│         FastAPI Backend     │
│  (Slot availability +       │
│   booking endpoints)        │
└─────────────────────────────┘
              │
              │
              ▼
┌─────────────────────────────┐
│        Local JSON DB        │
│ (Doctors + Appointments)    │
└─────────────────────────────┘
```

2.1 Overview of the Architecture

The system follows a client-server architecture, where the client (Streamlit UI) interacts with the server (FastAPI) via API endpoints. LangChain is used for intelligent reasoning to handle user queries and provide answers dynamically.

2.2 Key Components and Technologies

- FastAPI: Backend framework for handling API requests.
- LangChain: Framework for building AI-driven agents.

- Mistral 7B Model: Language model used for natural language understanding.
- Streamlit: Framework for building the frontend UI.

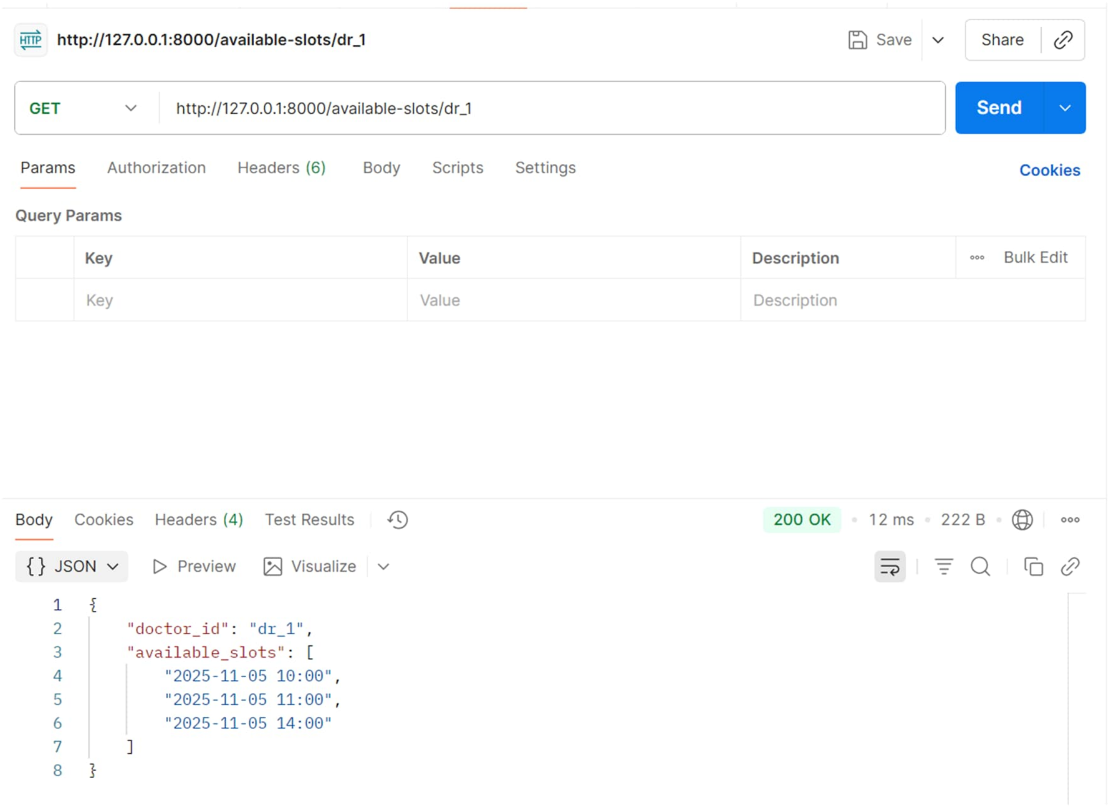
2.3 Flow of Data and Interactions

The user interacts with the Streamlit UI, which sends requests to the FastAPI backend. The backend processes these requests (e.g., checking available doctors or booking appointments). The LangChain AI agent is invoked to reason through the request and decide the appropriate action.

2.4 System Diagram

Include a high-level architecture diagram here (explained below in the next section).

3. Backend Implementation (FastAPI)



http://127.0.0.1:8000/available-slots/dr_1

GET http://127.0.0.1:8000/available-slots/dr_1

Query Params

Key	Value	Description
Key	Value	Description

200 OK • 12 ms • 222 B

```
{
    "doctor_id": "dr_1",
    "available_slots": [
        "2025-11-05 10:00",
        "2025-11-05 11:00",
        "2025-11-05 14:00"
    ]
}
```

3.1 API Design and Endpoints

The FastAPI backend provides several key endpoints:
- `/doctors`: Lists available doctors.
- `/available-slots/{doctor_id}`: Returns available appointment slots for a specified doctor.
- `/book-appointment`: Books an appointment for the user with the specified doctor and time.

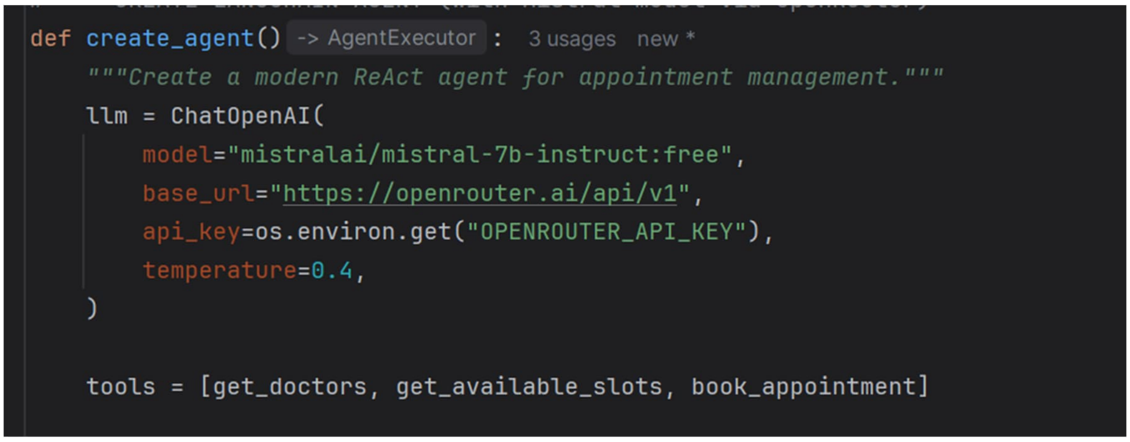
3.2 Doctor and Appointment Data Storage

Doctor data and appointment details are stored in a JSON file (`data.json`). Each doctor has an ID, name, and specialty, and each appointment contains the doctor's ID, patient name, and time.

3.3 Slot Availability and Booking Logic

The logic for checking available slots and booking appointments is simple. It checks whether a requested slot is available and adds the appointment to the list if it is.

4. AI Agent (LangChain)



```
def create_agent() -> AgentExecutor :  3 usages  new *
    """Create a modern ReAct agent for appointment management."""
    llm = ChatOpenAI(
        model="mistralai/mistral-7b-instruct:free",
        base_url="https://openrouter.ai/api/v1",
        api_key=os.environ.get("OPENROUTER_API_KEY"),
        temperature=0.4,
    )

    tools = [get_doctors, get_available_slots, book_appointment]
```

4.1 Agent Overview

The LangChain agent serves as the intelligent reasoning engine behind the system. It is capable of processing user queries and selecting the right actions (e.g., fetch doctor data, check available slots, book an appointment).

4.2 LangChain Framework

LangChain is a framework designed to facilitate building language model-driven applications by chaining multiple tasks. The key feature used in this system is the ReAct agent, which reasons through the user’s request and interacts with various tools dynamically.

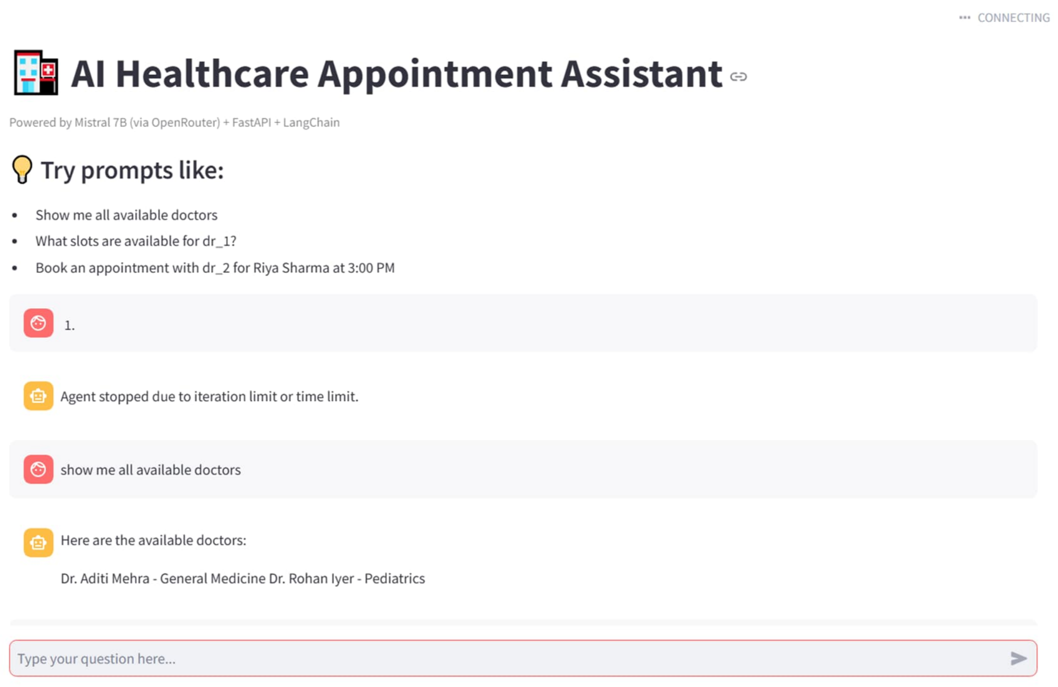
4.3 Tools for Agent Reasoning

The agent uses three main tools:
1. `get_doctors`: Fetches the list of available doctors.
2. `get_available_slots`: Checks available slots for a specified doctor.
3. `book_appointment`: Books an appointment.

4.4 ReAct Agent Design

The ReAct agent follows a thought-action-observation loop, where it decides what to do based on the user's request, executes an action, observes the result, and then generates a response.

5. Frontend Implementation (Streamlit UI)



CONNECTING

AI Healthcare Appointment Assistant

Powered by Mistral 7B (via OpenRouter) + FastAPI + LangChain

Try prompts like:

- Show me all available doctors
- What slots are available for dr_1?
- Book an appointment with dr_2 for Riya Sharma at 3:00 PM

1.

Agent stopped due to iteration limit or time limit.

show me all available doctors

Here are the available doctors:

Dr. Aditi Mehra - General Medicine Dr. Rohan Iyer - Pediatrics

Type your question here...

5.1 Building the Chat Interface

The UI is built using Streamlit, which provides a simple way to create interactive web apps. The chat interface allows users to input their queries and receive responses from the agent.

5.2 Handling User Input

Streamlit’s chat_input() function is used to capture user queries. The input is sent to the LangChain agent, which processes it and returns the appropriate response.

5.3 Connecting the UI with the Backend

The frontend communicates with the backend by sending HTTP requests to the FastAPI endpoints (e.g., to fetch doctors or available slots).

6. Integration of AI with Backend and Frontend

6.1 Tool Selection and Dynamic Action Flow

The agent is designed to dynamically select tools based on the user query. If the user asks for doctors, the agent will call the get_doctors tool. If the user asks for available slots, it will first ask for clarification and then call the get_available_slots tool.

6.2 Error Handling and User Clarification

The system includes error handling to ensure smooth interaction. For example, if the user asks for slots without specifying a doctor, the agent will prompt the user for clarification.

6.3 Testing the Full Workflow

The system has been tested with various use cases:
- Displaying available doctors.
- Showing available slots for specific doctors.
- Booking appointments.

7. Challenges and Solutions

7.1 Handling Missing Data

One challenge is ensuring that the backend data (e.g., doctors and slots) is up-to-date. This is handled by regularly updating the data.json file.

7.2 Ensuring Model Reliability

Ensuring that the LangChain agent consistently follows the correct format (i.e., Thought-Action-Observation) was a challenge. We solved this by refining the prompt template and adding fallback logic.

7.3 Improving User Experience

The user experience was enhanced by providing clear prompts and dynamically adjusting the system's behavior based on user input.

8. Conclusion

8.1 Summary of the System

This system successfully integrates AI-driven scheduling with real-time doctor availability, offering a user-friendly interface.

8.2 Future Enhancements

Multilingual support: Add support for multiple languages.
Real-time availability: Integrate with external databases to fetch real-time slot data.

8.3 Final Thoughts

This system demonstrates the power of AI in automating healthcare tasks and improving efficiency.

9. References

LangChain Documentation: https://langchain.com/docs
FastAPI Documentation: https://fastapi.tiangolo.com/
Streamlit Documentation: https://streamlit.io/docs