🩺 Doctor Appointment Assistant

A sophisticated AI-powered healthcare scheduling system that uses LangGraph's multi-agent architecture to handle doctor appointments intelligently. The system combines a supervisor agent pattern with specialized nodes for information retrieval and booking management.

🌐 Live Demo

Try the live application: http://3.86.149.92:8501
The system is currently deployed on AWS EC2 and ready for testing. Simply enter a user ID (e.g., 12345) and start chatting with the AI assistant!

🚀 What This Project Is

The Doctor Appointment Assistant is an intelligent conversational agent that helps patients:

- Check doctor availability by name or specialization
- Book new appointments
- Cancel existing appointments
- Reschedule appointments
- Get information about healthcare services

The system uses natural language processing to understand user requests and automatically routes them to the appropriate specialized agent for handling.

🧠 Key Concepts & Technologies

LangGraph Multi-Agent Architecture

- **Supervisor Agent Pattern**: Central coordinator that routes requests to specialized nodes
- **State Management**: Persistent conversation state across interactions
- **Conditional Routing**: Dynamic decision-making based on user intent

Agent Specialization

- **Information Node**: Handles availability queries and general information
- **Booking Node**: Manages appointment operations (set, cancel, reschedule)
- **Tool Integration**: Each node has access to specific tools for their domain

Memory & Persistence

- Thread-based conversation memory using LangGraph's MemorySaver
- CSV-based data persistence for appointment scheduling
- Session state management in the frontend

🏗️ System Architecture

```
┌──────────────────┐     ┌──────────────────┐     ┌──────────────────┐
│  Streamlit UI    │◄───►│   FastAPI        │◄───►│   LangGraph      │
│  (Frontend)      │     │   (Backend)      │     │   Agent          │
└──────────────────┘     └──────────────────┘     └──────────────────┘
                                                            │
                                                            ▼
                                             ┌──────────────────────────────┐
                                             │      Supervisor Node         │
                                             │  (Routes based on intent)    │
                                             └──────────────────────────────┘
                                                            │
                                              ┌─────────────┴─────────────┐
                                              ▼                           ▼
                                        ┌───────────────┐
┌───────────────┐
│                                       │ Information   │          │  Booking
│                                       │    Node       │          │   Node
│                                       │               │          │
│                                       │ • Check       │          │ • Set appt
│                                       │   availability│          │ • Cancel
│                                       │ • Doctor info │          │ • Reschedule
└───────────────┘                       └───────────────┘
                                              │                           │
                                              ▼                           ▼
                                        ┌───────────────┐
┌───────────────┐
│                                       │   Tools:      │          │   Tools:
│                                       │ • check_by_   │          │ • set_appt
│                                       │   doctor      │          │ • cancel
│                                       │ • check_by_   │          │ • reschedule
│                                       │   specialty   │          │
└───────────────┘                       └───────────────┘
```

🛠️ Tools & Technologies

Backend Stack

- **FastAPI**: REST API framework for handling HTTP requests
- **LangGraph**: Multi-agent workflow orchestration
- **LangChain**: LLM integration and tool management
- **Pydantic**: Data validation and serialization
- **Pandas**: Data manipulation for appointment records

Frontend Stack

- **Streamlit**: Interactive web interface
- **Session Management**: Persistent chat history and user state

AI/ML Stack

- **LLM Integration**: Support for OpenAI and Groq models
- **Structured Output**: Type-safe LLM responses using Pydantic models
- **Tool Calling**: Function calling capabilities for database operations

Data Storage

- **CSV Database**: Simple file-based storage for appointment data
- **In-Memory State**: LangGraph memory saver for conversation persistence

📊 Data Structure

The system uses a CSV-based database (`doctor_availability.csv`) with the following structure:

```
date_slot,specialization,doctor_name,is_available,patient_to_attend
05-08-2025 08:00,general_dentist,john doe,True,
05-08-2025 08:30,general_dentist,john doe,False,1000082.0
05-08-2025 09:00,general_dentist,john doe,False,1000048.0
05-08-2025 09:30,general_dentist,john doe,False,1000036.0
05-08-2025 10:00,general_dentist,john doe,False,1000024.0
05-08-2025 10:30,general_dentist,john doe,False,1000011.0
05-08-2025 11:00,general_dentist,john doe,False,1000061.0
```

Available Doctors

- John Doe, Jane Smith, Emily Johnson, Michael Green, Sarah Wilson, Daniel Miller, Susan Davis, Robert Martinez, Lisa Brown, Kevin Anderson

Specializations

- General Dentist, Cosmetic Dentist, Prosthodontist, Pediatric Dentist, Emergency Dentist, Oral Surgeon, Orthodontist

🔄 Behind the Scenes - Agent Workflow

1. Supervisor Node Decision Making

```
# The supervisor analyzes user input and routes to appropriate node
Router Response:
{
  "next": "information_node" | "booking_node" | "FINISH",
  "reasoning": "User wants to check availability..."
}
```

2. Information Node Processing

- Receives queries about doctor availability
- Uses specialized tools to query the database
- Returns availability information in natural language
- Handles follow-up questions about scheduling

3. Booking Node Operations

- Manages appointment lifecycle (create, update, delete)
- Validates user permissions using ID numbers
- Updates database state atomically
- Provides confirmation messages

4. State Management

```
AgentState = {
    "messages": [...],           # Conversation history
    "id_number": 1234567,         # User identification
    "next": "booking_node",     # Next node to execute
    "query": "...",             # Current user query
    "current_reasoning": "...", # Agent's reasoning
    "follow_up_needed": False   # Whether more input needed
}
```

🔄 Behind the Scenes - Frontend/Backend Flow

Request Flow

1. **User Input**: User types message in Streamlit chat interface
2. **Validation**: System checks for required user ID
3. **API Call**: POST request to FastAPI `/execute` endpoint
4. **Agent Invocation**: FastAPI triggers LangGraph agent with user state
5. **Processing**: Agent routes through supervisor → specialized node → tools
6. **Response**: Agent returns updated state with AI response
7. **UI Update**: Streamlit displays response and updates chat history

Session Management

- **Thread ID**: Unique identifier for conversation persistence
- **User ID**: Patient identification for appointment management
- **Message History**: Maintained both in frontend and agent memory

Error Handling

- Network timeouts with user-friendly messages
- Database operation failures with graceful degradation
- Input validation at multiple layers

🏃 How to Run Locally

Prerequisites

- Python 3.11
- pip package manager

Installation Steps

1. **Clone the repository**

```
git clone https://github.com/aaditey932/operationalizing-ai-weekly-projects.git
cd final-project
```

2. **Install dependencies**

```
pip install -r requirements.txt
```

3. **Set up environment variables**
   Create a `.env` file in the root directory:

```
OPENAI_API_KEY=your_openai_api_key_here
GROQ_API_KEY=your_groq_api_key_here
TAVILY_API_KEY=your_tavily_api_key_here
```

4. **Prepare the data**
   Ensure `data/doctor_availability.csv` exists with proper format

5. **Start the FastAPI backend**

```
uvicorn main:app --host 127.0.0.1 --port 8003 --reload
```

6. **Launch the Streamlit frontend**

```
streamlit run streamlit_ui.py
```

7. **Access the application**
   Open your browser and navigate to `http://localhost:8501`

Testing the System

- Enter a user ID (e.g., 1234567) in the sidebar
- Try queries like:
  - "Check availability for Dr. John Doe on January 15th"
  - "I need to book an appointment with a dentist"
  - "Cancel my appointment on January 16th"

🚀 Deployment Details

AWS EC2 Configuration

- **Instance Type**: t2.micro
- **Memory**: 16GB
- **Operating System**: Linux (Amazon Linux 2 or Ubuntu)
- **Storage**: 8GB+ SSD

📝 Usage Examples

Check Availability

```
User: "Is Dr. John Doe available tomorrow?"
System: "Let me check Dr. John Doe's availability for January 16th, 
2025..."
```

Book Appointment

```
User: "I need to book an appointment with a dentist for next Monday at 2 
PM"
System: "I'll help you book an appointment. Let me check 
availability..."
```

Cancel Appointment

```
User: "I need to cancel my appointment on January 15th"
System: "I'll help you cancel your appointment. Let me find your
```

```
booking..."
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/new-feature`)
3. Commit your changes (`git commit -am 'Add new feature'`)
4. Push to the branch (`git push origin feature/new-feature`)
5. Create a Pull Request