

Project Title: Medical AI Scheduling Agent

Author Name: Shahnawaz Anwar

Phone: +91 8750191902

Email: shaikhmdanwar80@gmail.com

GitHub: github.com/shahnawazanwar

LinkedIn: <https://www.linkedin.com/in/shaikh-shahnawaz-anwar-137683150>

Table

Title & Author Info

Introduction

Features

System Architecture (diagram + short explanation)

Directory Structure

Dependencies

Code Files Overview

Data Files (patients.csv, doctor_schedules.xlsx)

Setup & Run Instructions

Example Usage

Limitations & Future Work

Introduction:

The Medical AI Scheduling Agent is an intelligent medical appointment management system that automates the process of scheduling, rescheduling, and confirming patient–doctor appointments.

Built using LangGraph and Google Gemini, the agent integrates seamlessly with multiple tools such as a patient database, doctor schedules, and communication services. It ensures accurate, real-time scheduling while simulating professional email and SMS confirmations.

This project aims to:

Reduce manual effort in appointment booking.

Provide a smooth and interactive Streamlit-based UI for patients and staff.

Demonstrate the power of LLM-driven workflow orchestration in real-world healthcare use cases.

By combining state management, tool-driven decision making, and mock data generation, This project serves as a practical case study in applying AI agents to healthcare scheduling challenges.

Feature:

Patient Lookup – Identify whether a patient is new or returning.

Doctor Listing – Fetch available doctors dynamically.

Smart Scheduling – Check real-time availability and book slots (30 min for returning, 60 min for new patients).

Communication Tools – Simulated Email & SMS confirmations after booking.

Streamlit UI – Easy-to-use interface for patients and admins.

Data Generator – Create mock patients and doctor schedules for testing.

Logging – Automatically store confirmed appointments in Excel logs.

System Architecture Explanation:

The architecture is designed to be modular and tool-driven, ensuring flexibility and maintainability:

User – Interacts with the system through natural language.

Streamlit UI – Provides a simple and user-friendly interface for appointment booking.

LangGraph Agent (Gemini-powered) – Acts as the brain of the system:

Understands user requests.

Decides when and how to use tools.

Manages conversation flow.

Tools – Specialized modules supporting scheduling tasks:

Patient DB – Looks up patients to identify new or returning status.

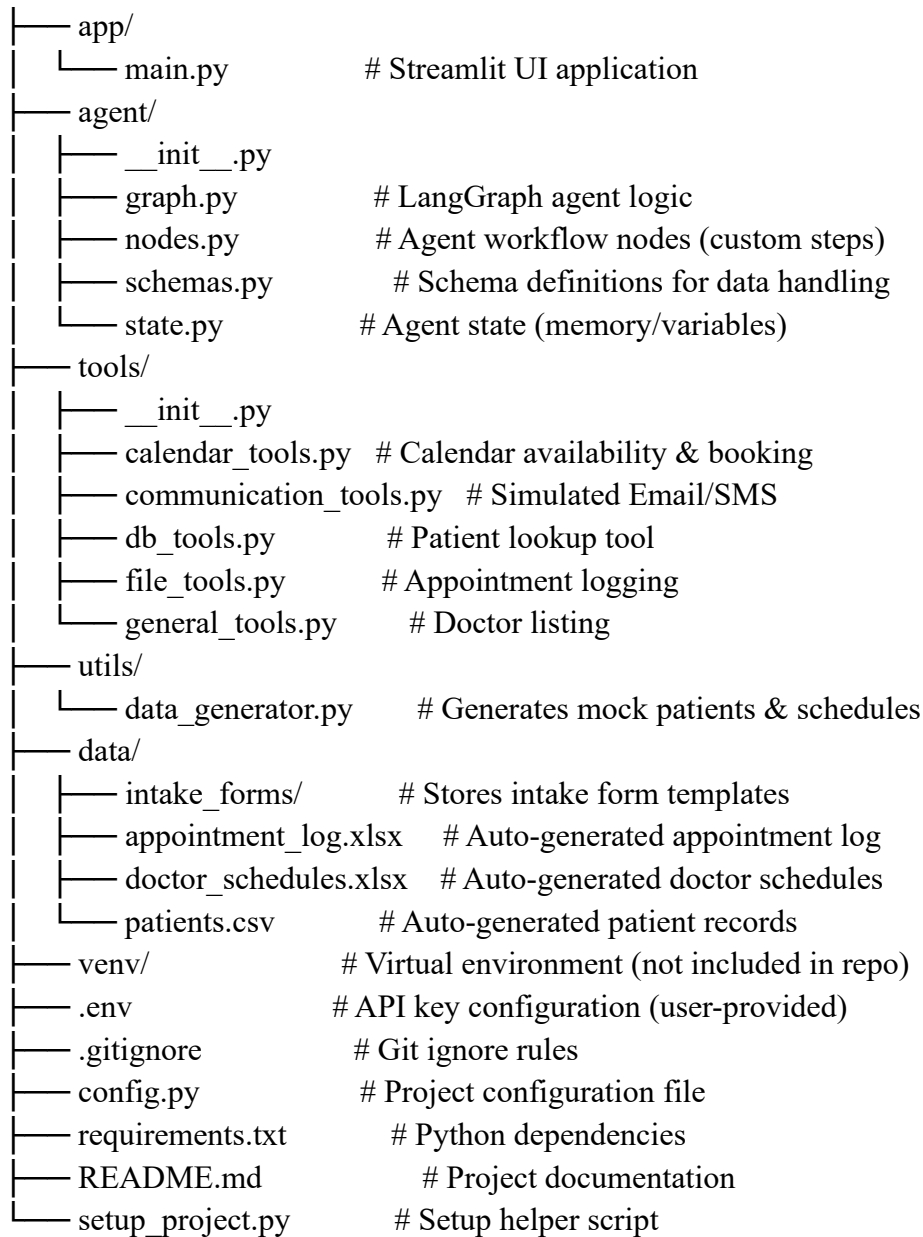
Calendar – Checks doctor availability and books slots.

Communication – Sends simulated Email/SMS confirmations.

This workflow ensures accurate scheduling, clear communication, and scalable integration with real-world healthcare systems in the future.

Directory Structure

ai_scheduling_agent/



Short Notes:

app/ → Streamlit UI (main.py)

agent/ → Core AI agent logic

graph.py → LangGraph workflow

state.py → Agent memory/state

nodes.py → Workflow nodes

schemas.py → Data schemas

tools/ → Agent helper tools

db_tools.py → Patient lookup

calendar_tools.py → Check & book slots

general_tools.py → Doctor listing

communication_tools.py → Email/SMS simulation

Project Root Files

.env → Stores API key (not shared in repo).

.gitignore → Hides sensitive/unnecessary files (venv/, .env, __pycache__/, etc.).

config.py → Central configuration for project settings.

requirements.txt → Lists Python dependencies.

README.md → Main project documentation.

setup_project.py → Script to automate environment or data setup.

venv/ → Local Python virtual environment (excluded from repo).

Dependencies:

The project relies on the following Python libraries:

langchain, langgraph, langchain-google-genai → Core agent framework and Gemini integration.

streamlit → Interactive UI for patients and staff.

pandas, openpyxl → Data handling, patient records, and schedule storage.

Faker → Generate mock patient and schedule data.

python-dotenv → Manage environment variables securely (.env file).

Code Files Overview:

app/main.py → Streamlit app for user interaction.

agent/state.py → Defines agent state (patient info, doctor, date, etc.).

agent/graph.py → LangGraph workflow logic using Gemini.

agent/nodes.py → Custom nodes for workflow steps.

agent/schemas.py → Data schemas for validation.

tools/db_tools.py → Patient lookup (new/returning).

tools/calendar_tools.py → Check availability & book slots.

tools/general_tools.py → List available doctors.

tools/communication_tools.py → Simulated Email/SMS confirmations.

tools/file_tools.py → Export appointments to Excel logs.

utils/data_generator.py → Generates mock patients and schedules.

config.py → Project configuration file.

setup_project.py → Script for environment/data setup.

Data Files:

patients.csv → Stores mock patient records (ID, name, DOB, phone). Generated by data_generator.py.

doctor_schedules.xlsx → Contains doctors' schedules, availability, and time slots. Generated by data_generator.py.

appointment_log.xlsx → Stores booked appointments with patient, doctor, date, time, and insurance details. Logged automatically when booking.

intake_forms/ → Placeholder folder for patient intake forms (future use)

Setup & Run Instructions:

Create Virtual Environment:

```
python3 -m venv venv
```

```
source venv/bin/activate # On Linux/Mac
```

```
venv\Scripts\activate # On Windows
```

Install Dependencies

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

Set Up API Key

Create a .env file in the project root.

Add your Google API key:

```
GOOGLE_API_KEY="your_api_key_here"
```

Generate Mock Data

```
python utils/data_generator.py
```

Run Application

```
streamlit run app/main.py
```

Example Conversation:

Agent: Hello Shahnawaz! I'm your AI medical scheduling assistant. How may I help you today?

User: I want to book a doctor appointment.

Agent: Sure! Please provide your full name and date of birth (YYYY-MM-DD).

User: My name is Shahnawaz Anwar, DOB 1995-03-15.

Agent: Thanks, Shahnawaz. I found your record as a returning patient.

Here are the available doctors:

- Dr. Smith

- Dr. Jones

- Dr. Williams

User: I want to book with Dr. Smith on 2025-09-15.

Agent: Dr. Smith is available on 2025-09-15.

The available slots are:

- 10:00

- 10:30

- 11:00

Which time would you prefer?

User: Book 10:30. My mobile number is 8750191902.

Agent: Appointment confirmed for Shahnawaz Anwar with Dr. Smith on 2025-09-15 at 10:30.

A confirmation Email + SMS have been sent to 8750191902.

Limitations & Future Work:

Limitations

Works with mock data only (patients & schedules generated with Faker).

Email & SMS are simulated (not connected to real services).

No real-time calendar sync (e.g., Google/Outlook).

Basic error handling, may fail with unexpected inputs.

Future Work:

Integration with real EHR systems and hospital databases.

Sync with Google Calendar / Outlook APIs.

Use real communication APIs (Twilio, SendGrid).

Add authentication & secure data storage.

Dashboard for doctors/admins to view bookings & reports.