Agentic AI Doctor Appointment System

A comprehensive full-stack **agentic AI** platform for doctor appointment scheduling and management. The system uses **MCP (Model Context Protocol)** so the LLM can autonomously decide which tools to call based on natural-language inputs.

> **Stack:** FastAPI · PostgreSQL · SQLAlchemy · WebSockets · OpenAI (GPT-4/4.1 compatible) · Next.js 15 (React/TypeScript) · Tailwind CSS · shadcn/ui · Google Calendar (optional) · SendGrid/SMTP (optional)

🎯Assignment Checklist

- [x] **Clean, modular codebase** (separated backend/frontend, services, models, routes)
- [x] **README** with setup steps, sample prompts, and **API usage summary**
- [x] **Prompt-based appointment booking** (multi-turn)
- [x] **Real-time doctor notifications** + **summarized stats**
- [x] Optional: short demo video + screenshots

🎞️Architecture Overview

This is a true agentic architecture: the LLM interprets free-form requests, selects MCP tools, executes them, and composes responses. Context is preserved across turns.

```
User ↔️ Next.js UI (chat + dashboards)
          ↓
      FastAPI /api/chat — Session & state mgmt
          ↓
      Agent Orchestrator (LLM + MCP client)
          ↓
     ┌──────────────────────────────────────────┐
     │              MCP Tools                   │
     │ 1) check_doctor_availability            │
     │ 2) schedule_appointment                 │
     │ 3) get_appointment_stats                │
     │ 4) search_patients_by_symptom           │
     │ 5) get_doctor_schedule                  │
     └──────────────────────────────────────────┘

          ↓                          ↓
    PostgreSQL (SQLAlchemy)     Integrations
                                 • Google Calendar
```

```
                                    • Email (SendGrid/SMTP)
                                    • WebSockets → doctor UIs
```

🖌️ Key Features

Agentic AI

- **Natural-language understanding** of complex appointment requests
- **Autonomous tool selection** via MCP
- **Multi-turn conversations** with memory
- **Intelligent suggestions** & follow-ups

Core Functionality

- **Appointment scheduling** with availability checks
- **Doctor management** (profiles, availability)
- **Patient management** (records, history)
- **Real-time notifications** for doctors (WebSockets)
- **Calendar sync** (Google Calendar)
- **Email confirmations & reminders**

Repository Structure

```
/ (root)
├─ backend/
│  ├─ app/
│  │  ├─ main.py                # FastAPI entry
│  │  ├─ core/                  # config, logging, deps
│  │  ├─ api/                   # routers (chat, mcp, doctors, patients)
│  │  ├─ services/              # agent, notifications, calendar, email
│  │  ├─ mcp_tools/             # tool impls (5 tools)
│  │  ├─ db/                    # models, schemas, session
│  │  └─ tests/                 # pytest suites
│  └─ requirements.txt
│
├─ frontend/
│  ├─ app/                      # Next.js (App Router)
│  ├─ components/               # UI (chat, schedules, stats)
│  ├─ lib/                      # api client, websocket hooks
│  └─ package.json
│
├─ scripts/
│  ├─ 01_create_database_schema.sql
```

```
│   ├─ 02_seed_sample_data.sql
│   ├─ test_complete_system.py
│   ├─ test_agent.py
│   └─ test_mcp_tools.py
│
├─ .env.example
└─ README.md (this file)
```

🧩 Prerequisites

- **Python** 3.9+
- **Node.js** 18+
- **PostgreSQL** 12+
- **OpenAI API key**
- Optional: **Google Calendar** credentials, **Email** service credentials

---

Configuration (.env)

```
# Database
DATABASE_URL=postgresql://user:password@localhost/doctor_appointments

# OpenAI
OPENAI_API_KEY=your-openai-api-key

# Google Calendar (Optional)
GOOGLE_CALENDAR_CREDENTIALS_FILE=path/to/credentials.json
GOOGLE_CALENDAR_TOKEN_FILE=path/to/token.json

# Email Service (Optional)
SENDGRID_API_KEY=your-sendgrid-key
SMTP_SERVER=smtp.gmail.com
SMTP_PORT=587
SMTP_USERNAME=your-email@gmail.com
SMTP_PASSWORD=your-app-password

# App
DEBUG=true
SESSION_TIMEOUT_MINUTES=30
ALLOWED_ORIGINS=["http://localhost:3000"]
```

🖌️ Quick Start

1) Database

```
# Create database & schema
psql -U postgres -c "CREATE DATABASE doctor_appointments;"
psql -U postgres -d doctor_appointments -f scripts/01_create_database_schema.sql
psql -U postgres -d doctor_appointments -f scripts/02_seed_sample_data.sql
```

2) Backend (FastAPI)

```
cd backend
pip install -r requirements.txt
cp ../.env.example .env
# edit .env values
python -m app.main   # or: uvicorn app.main:app --reload
```

3) Frontend (Next.js 15)

```
cd frontend
npm install
npm run dev
```

4) Run Tests

```
cd scripts
python test_complete_system.py
python test_agent.py
python test_mcp_tools.py
```

🧨 Testing Coverage

The comprehensive suite validates: - DB connectivity & models - MCP tool behavior - Agent responses & tool selection - REST API endpoints - WebSocket notifications - External integrations (Calendar/Email) — mocked in CI

🎂 Sample Prompts (Copy-paste)

Patient — Multi-turn

Patient 1: "I want to check **Dr. Ahuja's** availability for **Friday afternoon**."
AI: "Checking Dr. Ahuja's schedule for Friday afternoon. Here are the available slots:
• 2:30 PM • **3:00 PM** • 4:15 PM
Which time works for you?"

Patient 2: "Please book the **3 PM** slot."
AI: "Great. Booking **Friday 3:00 PM** with Dr. Ahuja... ✅ **Appointment confirmed.** You'll receive an email shortly."

Doctor — At-a-glance Analytics

- "**How many patients visited yesterday?**"
- "**How many appointments do I have today and tomorrow?**"
- "**How many patients with fever** this month?"

Other Useful Patient Prompts

- "Reschedule my appointment with Dr. Johnson to **Monday 2 PM**."
- "Cancel my appointment with **Dr. Smith** tomorrow."
- "Show my **past 3 visits** for headaches."

🎄 API Summary

Chat

```
POST /chat
Content-Type: application/json
{
  "message": "I want to book an appointment with Dr. Smith tomorrow at 2 PM",
  "session_id": "optional-session-id",
  "user_type": "patient"  // or "doctor"
}
```

MCP Tool Execution

```
POST /mcp/execute
Content-Type: application/json
{
  "tool_name": "check_doctor_availability",
  "parameters": {
```

```
    "doctor_name": "Dr. Smith",
    "date": "2024-01-15",
    "time_preference": "afternoon"
  }
}
```

WebSocket Notifications (Doctor UI)

```
const ws = new WebSocket('ws://localhost:8000/ws/notifications/1');
ws.onmessage = (event) => {
  const notification = JSON.parse(event.data);
  console.log('New notification:', notification);
};
```

🧩 MCP Tools

1. **check_doctor_availability**
2. **Params:** `doctor_name`, `date`, `time_preference`
3. **Returns:** array of available time slots
4. **schedule_appointment**
5. **Params:** `doctor_name`, `patient_name`, `patient_email`, `appointment_date`, `appointment_time`, `symptoms`
6. **Returns:** confirmation (DB row), Calendar event (optional), email (optional)
7. **get_appointment_stats**
8. **Params:** `doctor_name`, `date_range`, `filter_by`
9. **Returns:** aggregate stats (counts, trends)
10. **search_patients_by_symptom**
11. **Params:** `symptom`, `doctor_name`, `date_range`
12. **Returns:** list of patients matching criteria
13. **get_doctor_schedule**
14. **Params:** `doctor_name`, `start_date`, `end_date`
15. **Returns:** full schedule for range

🏓 Frontend UX Notes

- **Chat** panel for agent conversations (patient/doctor roles)
- **Doctor Dashboard**: today/tomorrow counts, yesterday visits, symptom filters (e.g., "fever")
- **Live notifications** chip/badge for: `new_appointment`, `appointment_cancelled`, `appointment_reminder`, `patient_message`, `system_alert`
- **Calendar view** (optional) synced with Google Calendar

Notification Types

- `new_appointment` — High priority, includes patient and time
- `appointment_cancelled` — Medium
- `appointment_reminder` — High, 30-minute reminder
- `patient_message` — Medium
- `system_alert` — Variable priority

🪃 Demo Script (for Video/Screenshots)

1) **Patient flow (booking):** - Prompt: *"I want to check Dr. Ahuja's availability for Friday afternoon."* - Agent shows slots → user picks → **booking confirmed** - **Screenshot**: Chat showing prompt + slot list + confirmation

2) **Doctor notifications:** - Receive **real-time** `new_appointment` notification in doctor UI - **Screenshot**: notification toast/card with appointment details

3) **Doctor summary stats:** - Prompt: *"How many patients visited yesterday? How many appointments do I have today and tomorrow?"* - Agent returns counts + list - **Screenshot**: stats widget + chat response

---

Troubleshooting

Database: confirm Postgres is running; check `DATABASE_URL`
OpenAI: key valid, quota available
WebSockets: CORS origins in `.env`, firewall open
Email: verify SMTP/SendGrid creds
Calendar: check OAuth creds and scopes

Enable verbose logging:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

🥎 Deployment

- **DB**: Managed PostgreSQL (RDS/Cloud SQL)
- **Backend**: FastAPI via Uvicorn/Gunicorn
- **Frontend**: Next.js on Vercel/Render/Netlify
- **HTTPS**: enable TLS for WebSockets

Docker (Backend)

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Contributing

1. Fork
2. Feature branch
3. Add tests
4. PR

License

MIT

🎺 Support

1. Read this README
2. Run the test suite
3. Check server logs
4. Open an issue