**Health Assistant — Project Documentation**

**Project Overview**

**Health Assistant** is a web application designed to empower users to understand their health better through AI-powered tools. It offers a Symptom Checker that analyzes user symptoms to suggest possible diagnoses and health insights, and an AI Health Assistant chatbot to answer health-related queries conversationally.

This project leverages the Gemini 2.5 Flash AI model from Google AI Studio to provide intelligent and fast healthcare insights with a conversational interface.

**Features**

**Symptom Checker**

* Input symptoms to receive potential diagnoses and insights.
* Powered by Gemini AI for intelligent and probabilistic outputs.

**AI Health Assistant Chatbot**

* Real-time interaction with an AI agent that answers health-related questions.
* Maintains conversation history for contextual continuity.

**Clean User Interface**

* Intuitive, responsive React.js interface.
* Accessible chat button to interact with the health assistant.

**Real-time API Integration**

* FastAPI backend ensures fast, reliable communication.
* Gemini API integration for symptom analysis and NLP-based conversation.

**Innovation & Impact**

* **Novel Use of AI**: Integrates Gemini 2.5 Flash AI to intelligently analyze symptoms and enable contextual conversations.
* **Healthcare Empowerment**: Bridges the gap between basic health queries and clinical diagnosis.
* **Scalability**: Designed to be extended into telemedicine, multilingual access, and advanced diagnostics.
* **Responsible AI Use**: Provides disclaimers and avoids overstepping medical advice boundaries.

**Technical Depth & Functionality**

* **Modular Backend Architecture**:
  + Symptom Analyzer API (feature1\_fastapi.py) using FastAPI.
  + Health Assistant API (chatbot.py) for managing chats.
  + Server manager script (server\_manager.py) to run multiple services.
* **Frontend Stack**:
  + React.js, Node.js v14+, Vite
  + Styled for usability and real-time interaction.
* **Gemini AI Integration**:
  + Uses Gemini 2.5 Flash AI via API key from Google AI Studio.
  + Responds with formatted output: bold keywords, explanations, and segmented content.
* **Functionality Highlights**:
  + Symptom analysis via POST /analyze
  + Chat interaction via POST /chat
  + Session reset and health check endpoints.

**Efficiency & User Experience**

* **Optimized API Communication**: Uses Uvicorn and FastAPI for high-performance backend communication.
* **Interactive UI**: Users can intuitively analyze symptoms and chat in real-time.
* **Rapid Feedback**: Responses are formatted and generated with low latency.
* **Lightweight Deployment**: Uses scripts to start and manage servers easily.

**Demo & Presentation Strategy**

* **Walkthrough Ready**: Demo begins with a symptom input followed by chatbot use.
* **Explanation Highlights**: AI responses begin with "Explanation:" for clarity.
* **Clarity in Communication**: Chat includes bold text and sectioning for better user understanding.
* **Professional Design**: No clutter, intuitive access points, and well-documented instructions.

**Technology Stack**

* **Backend**: Python 3.8+, FastAPI, Uvicorn
* **Frontend**: React.js, Node.js 14+, npm/yarn
* **AI Integration**: Google Gemini 2.5 Flash AI
* **Communication**: RESTful API endpoints

**Setup and Installation**

**Prerequisites**

* Python 3.8+
* Node.js 14+
* npm or yarn

**Backend Setup**

pip install -r requirements.txt

* Get your API key from Google AI Studio and insert it in the relevant config or frontend file.

**Frontend Setup**

npm install

npm run dev

**Running the Application**

**Option 1 (Recommended):** Use start\_servers.bat or start\_servers.sh for backend, and npm run dev for frontend.

**Option 2:** Run python server\_manager.py and frontend separately.

**Option 3:** Run both FastAPI apps manually and then start frontend.

**API Endpoints**

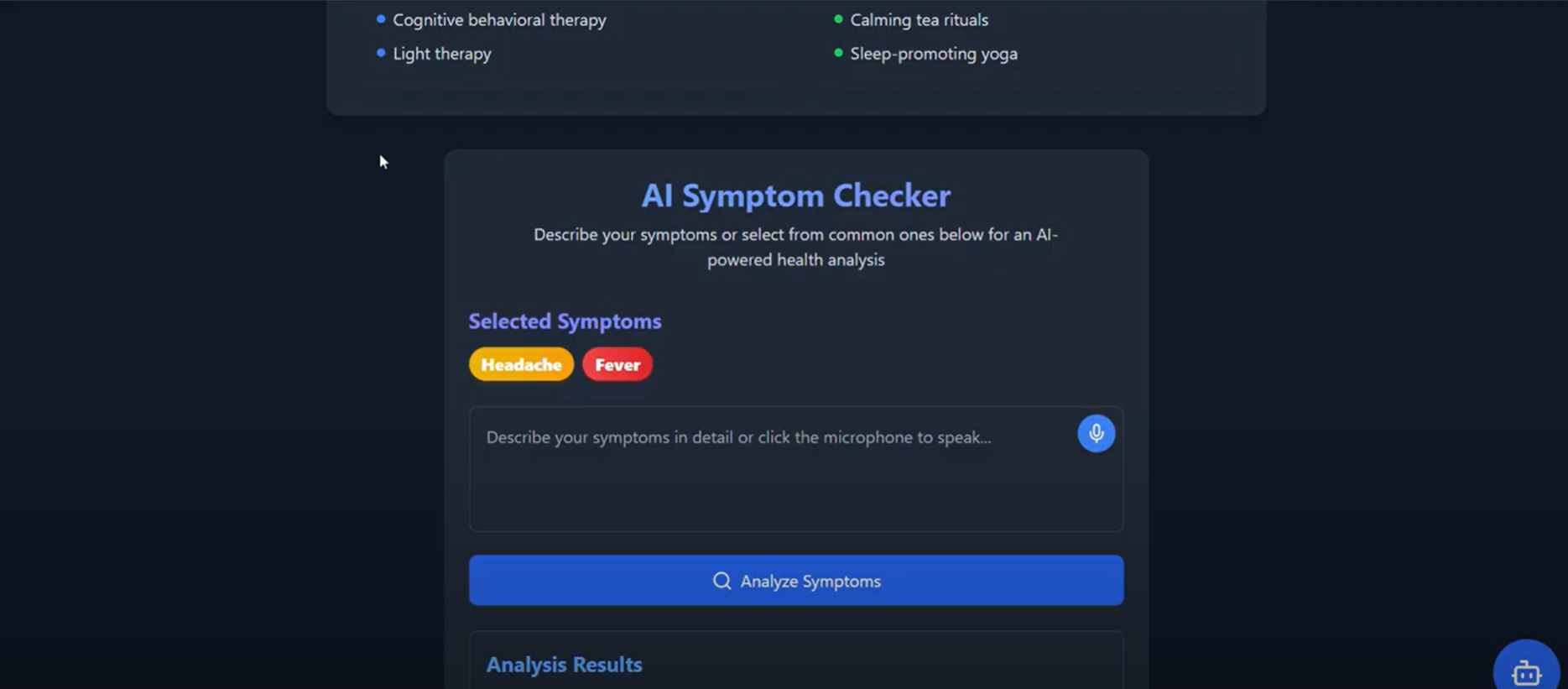
**Symptom Analyzer API (Port 8000)**

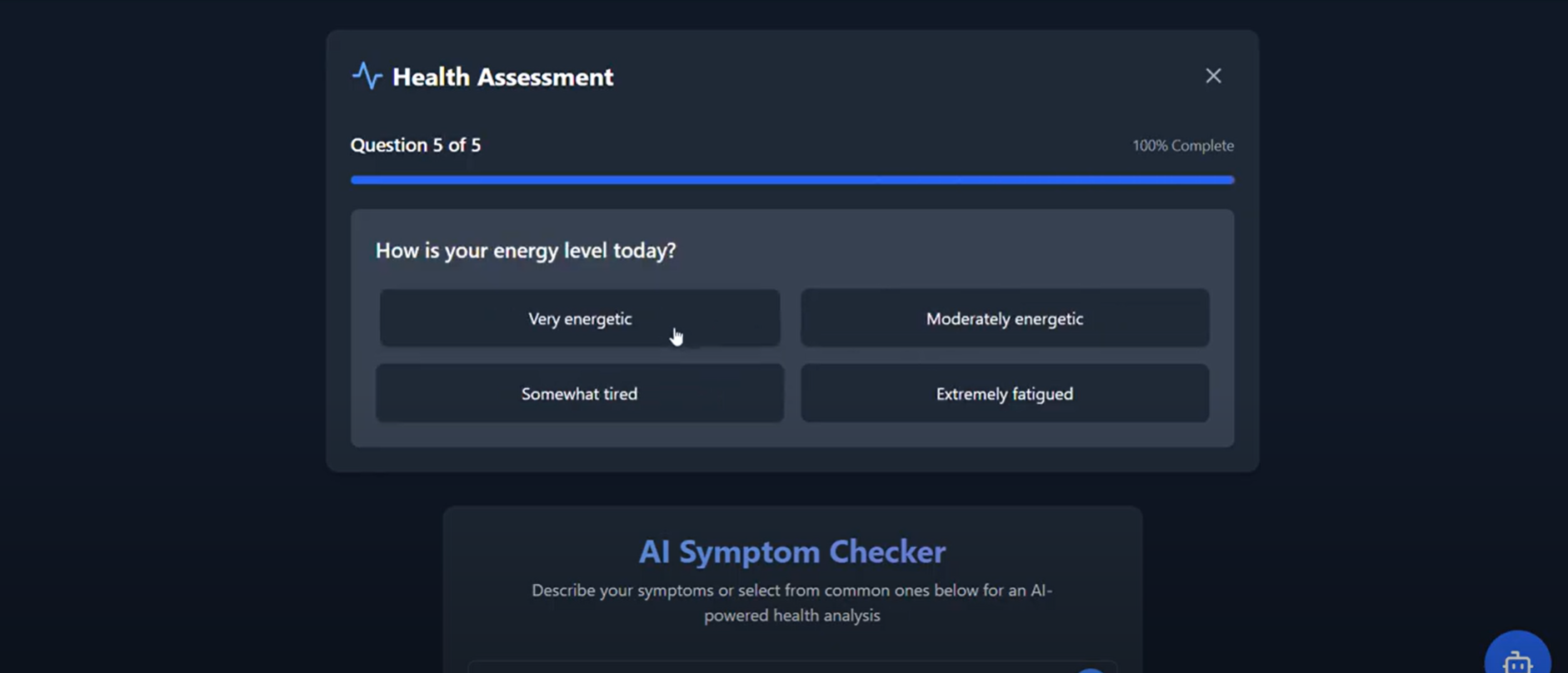
|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| POST | /analyze | Analyze symptoms |
| GET | /health | Health check |

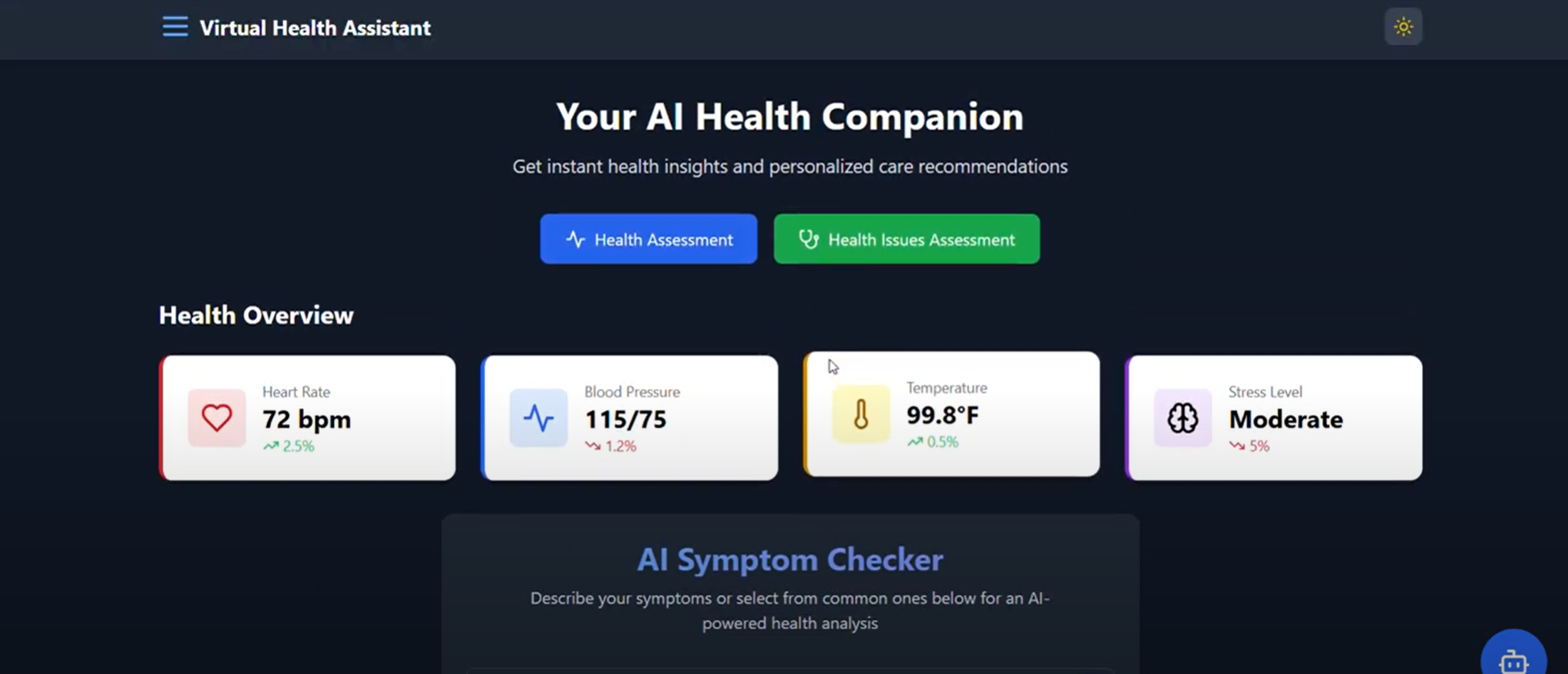
**Health Assistant API (Port 8001)**

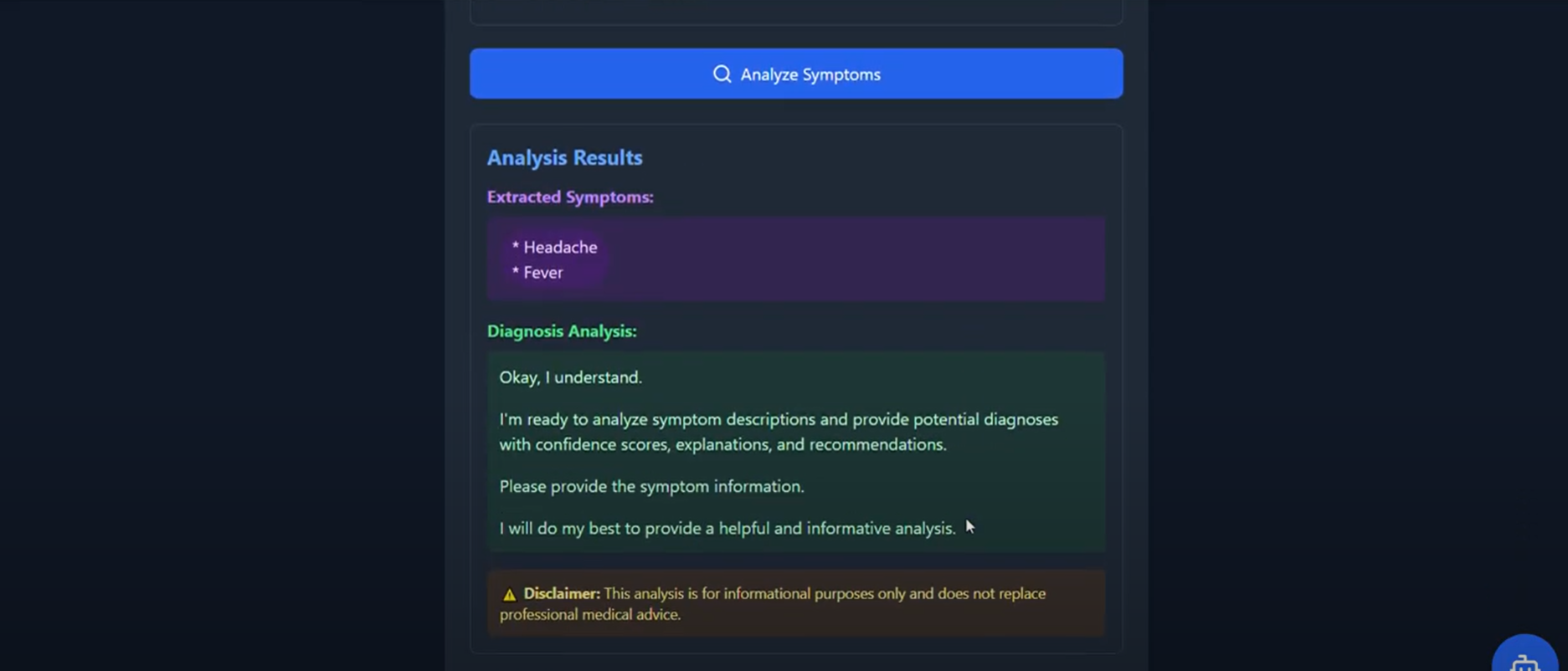
|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| POST | /chat | Chat with AI assistant |
| POST | /reset | Reset conversation history |
| GET | /health | Health check |

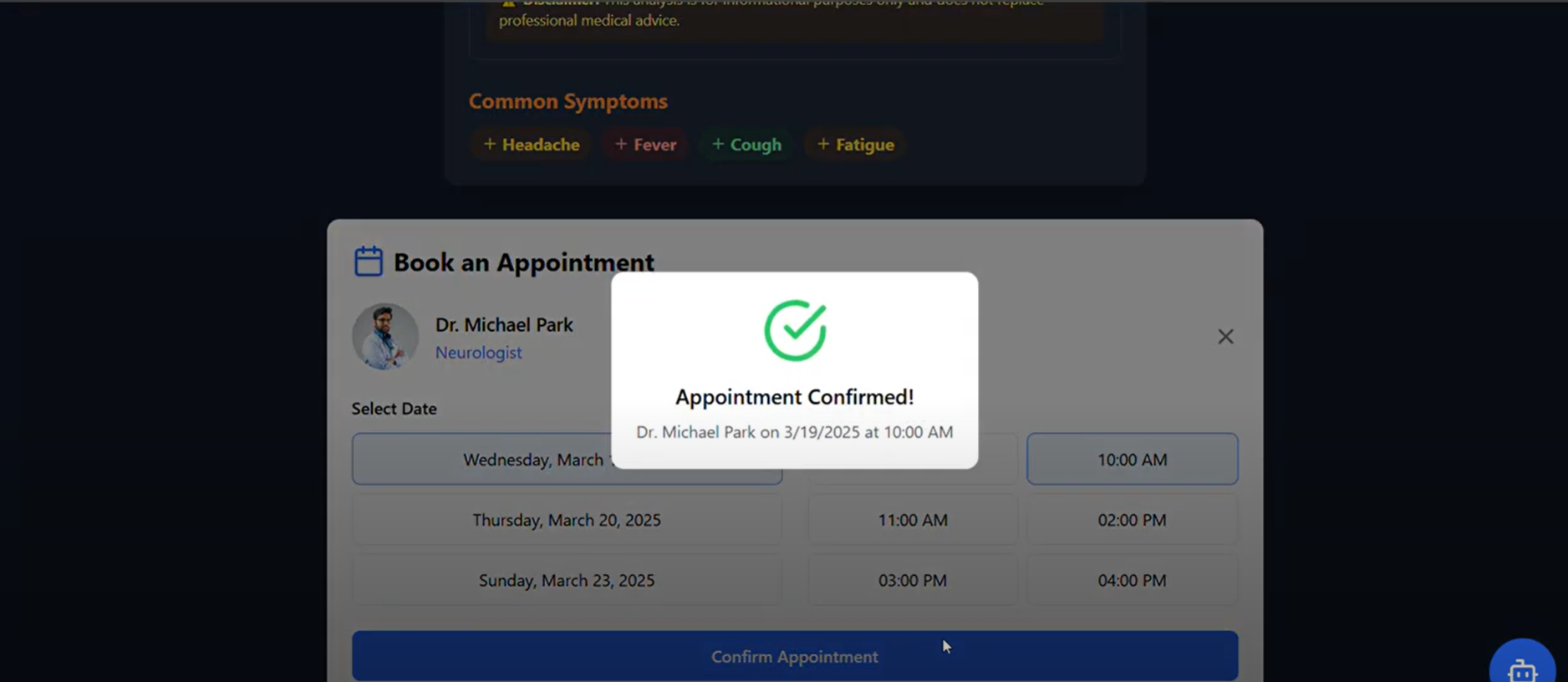
**Output**

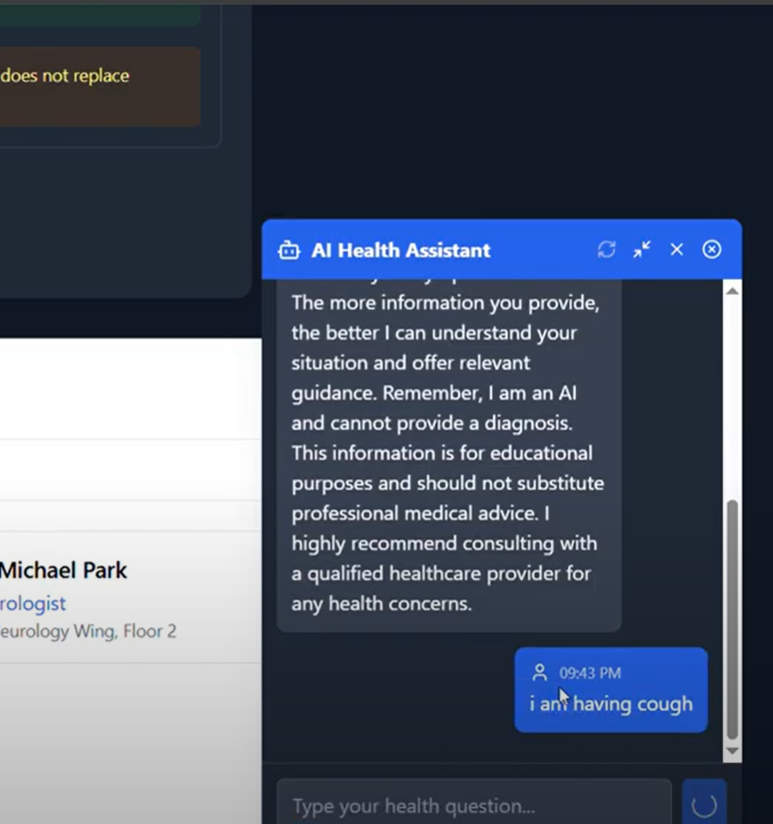












**AI HEALTH CHATBOT**

**Troubleshooting Guide**

**Common Issues:**

* **Connection Errors**: Use troubleshoot.bat or troubleshoot.py to debug issues.
* **Port Conflicts**: Kill processes using 8000/8001 or update port configs.
* **Missing Files**: Ensure required Python scripts exist in root directory.
* **Server Fails to Start**: Check console output for tracebacks or missing dependencies.

**Manual API Testing:**

curl http://localhost:8000/health

curl <http://localhost:8001/health>

**Disclaimer**

This application is for informational purposes only and does not replace professional medical advice. Always consult a healthcare provider for any medical condition.

**Future Enhancements**

* Add multilingual support
* Include user authentication and history tracking
* Integrate richer medical databases and diagnostic tools
* Expand into mental health, diet, and fitness recommendations

**Contact & Contribution**

For support, queries, or contributions, contact: **[]**