**HealthAI – Intelligent Healthcare Assistant**

**Project Documentation**

**1.Introduction**

**• Project title : HealthAI – Intelligent Healthcare Assistant**

**• Team member : SANDHIYA K**

**• Team member : SANGEETHA K N**

**• Team member : SARULATHA M**

**• Team member : SHAKERA BEGAM S R**

**Healthcare is one of the most essential aspects of human life, yet millions of people across the globe face barriers in accessing timely and reliable medical information. In rural areas and developing countries, the shortage of healthcare professionals often leaves patients with limited support, while in urban centers, overburdened hospitals and clinics can delay consultation. Additionally, misinformation on the internet often misguides people into incorrect self-diagnoses, sometimes causing panic or leading them to neglect serious conditions.**

**With the rapid advancement of Artificial Intelligence (AI), there is now an opportunity to create systems that provide accessible, affordable, and intelligent healthcare assistance to people anywhere, anytime. HealthAI is one such solution, designed as a smart assistant that leverages Granite models from Hugging Face to deliver conversational, patient-friendly, and medically relevant insights.**

**The project is deployed in Google Colab, which ensures that users can run it without requiring specialized hardware or complex setup. HealthAI is not intended to replace doctors but to serve as an educational tool and a first-level guidance system. By interacting with the assistant, users can describe their symptoms, ask questions about diseases, and receive suggestions for basic treatment options.**

**This project documentation explains the objectives, design, architecture, and functionality of HealthAI in detail. It also discusses testing, limitations, and potential future enhancements, offering a comprehensive view of how AI can transform healthcare accessibility.**

**2. Objectives**

**The primary goal of HealthAI is to empower users with smart, easy-to-understand medical guidance using natural language processing and machine learning. More specifically, the objectives are:**

1. **Conversational Healthcare Assistance – Provide patients with a chatbot interface to ask health-related queries in plain language and receive informative responses.**
2. **Disease Prediction – Analyze symptoms described by users and predict possible health conditions using AI models.**
3. **Treatment Plan Suggestions – Provide preventive care advice, management strategies, and lifestyle recommendations.**
4. **Accessible Deployment – Run the entire system in Google Colab to ensure it is easy to use, free, and platform-independent.**
5. **Scalability – Build the system in a modular way so that additional functionalities like medicine lookup, doctor appointment reminders, or wearable integration can be added in the future.**

**By meeting these objectives, HealthAI aims to reduce the gap between healthcare knowledge and patients, particularly for those who struggle to access professional advice.**

**3. Problem Statement**

**Healthcare systems worldwide are under significant pressure due to increasing population, chronic disease prevalence, and uneven distribution of medical professionals. Many people rely on internet searches for medical guidance, but search results often lack personalization and reliability. This creates confusion and, in many cases, harmful self-medication practices.**

**Moreover, traditional healthcare access is limited by cost, geographical barriers, and long waiting times. There is a pressing need for AI-driven, conversational healthcare tools that can provide initial guidance, help people understand their symptoms, and encourage them to seek professional care when necessary.**

**HealthAI addresses this challenge by combining large language model capabilities with a healthcare focus, providing an easy-to-use conversational assistant that can simulate the first stage of medical interaction.**

**4. Scope of the Project**

**The scope of HealthAI is focused on providing preliminary healthcare support and education. It is not designed to diagnose or prescribe medications but rather to:**

* **Answer general medical questions.**
* **Provide condition awareness based on symptoms.**
* **Suggest lifestyle modifications and preventive care strategies.**
* **Serve as an educational platform for students and researchers interested in healthcare AI.**

**The system does not integrate with hospital databases, medical prescriptions, or real-time emergency services, as those would require strict compliance with healthcare regulations such as HIPAA or GDPR. Instead, HealthAI serves as a research-oriented and educational healthcare assistant, demonstrating how AI can enhance accessibility.**

**5. Literature Review / Related Work**

**AI in healthcare has gained significant attention in the past decade. Systems such as IBM Watson Health, Ada Health, and Buoy Health have demonstrated the potential of AI-powered assistants in clinical decision support and patient triage.**

* **IBM Watson Health focuses on supporting doctors by analyzing medical literature and providing treatment recommendations. However, it is complex and mostly available in enterprise settings.**
* **Ada Health provides a mobile-based symptom checker, widely used by individuals, but its algorithms are proprietary and not easily customizable for research projects.**
* **Buoy Health is another commercial AI chatbot that helps in patient triage, but again, it is a closed-source system.**

**Compared to these, HealthAI is designed as an open, educational, and modular system. It uses Granite LLM from Hugging Face, which ensures transparency, customizability, and adaptability for academic and research purposes. Unlike commercial tools, HealthAI can be deployed freely in Google Colab, making it highly accessible.**

**6. Project Overview**

**HealthAI integrates Granite, a large language model trained on diverse datasets, with healthcare-specific logic to provide assistance. The system allows users to:**

* **Chat naturally about health concerns.**
* **Input symptoms to predict possible conditions.**
* **Receive suggestions for treatment and preventive steps.**
* **Access health tips for general well-being.**

**The project is modular, with separate components for chat, disease prediction, and treatment generation. It can be extended with future features such as:**

* **Medication side-effect explanations.**
* **Health record analysis.**
* **Integration with wearable devices for monitoring vitals.**

**By structuring the project in this way, HealthAI remains flexible and scalable while still being practical for demonstration and learning.**

**7. Features**

**7.1 Patient Chat**

**The conversational interface is the core feature of HealthAI. Users can describe their symptoms or ask general medical questions in natural language. For example, a user might type, “I have a sore throat and fever, what could this be?” and the system responds with a possible explanation, while advising the user to consult a professional if symptoms persist.**

**7.2 Disease Prediction**

**Based on user-reported symptoms, HealthAI predicts potential conditions. While it cannot provide definitive diagnoses, it helps narrow down possibilities, reducing patient anxiety and guiding them toward appropriate care.**

**7.3 Treatment Plans**

**The system suggests preventive measures and simple treatment options. For example, for common cold symptoms, it might suggest rest, hydration, and over-the-counter remedies, while reminding users to see a doctor if symptoms worsen.**

**7.4 Extendable Modules**

**The design of HealthAI allows for additional functionalities, including:**

* **Medicine information lookup.**
* **Preventive health tips.**
* **Doctor appointment scheduling.**
* **Integration with fitness apps and devices.**

**8. System Architecture**

**The architecture of HealthAI consists of four main layers:**

1. **User Interface Layer (Google Colab / Gradio):  
   Provides a simple and interactive platform for users to input queries and view responses.**
2. **Application Layer:  
   Contains logic for different modules (chat, prediction, treatment generation).**
3. **Model Layer (Granite LLM):  
   Processes natural language input and generates contextually relevant medical responses.**
4. **Deployment Layer (Google Colab):  
   Ensures accessibility and portability, requiring only a browser and internet connection.**

***(Here, a diagram showing interaction flow between User → UI → Granite Model → Response can be inserted.)***

**9. Technology Stack**

* **Programming Language: Python 3.9+**
* **Core Libraries: Transformers, Torch**
* **Frontend Interface: Gradio or Colab Notebook Cells**
* **Backend Model: Granite (Hugging Face)**
* **Deployment: Google Colab**
* **Visualization Tools (optional): Pandas, Matplotlib**

**10. Module Description**

1. **Chat Module: Handles patient interaction and response generation.**
2. **Prediction Module: Uses symptom descriptions to suggest possible conditions.**
3. **Treatment Module: Generates treatment and preventive care guidance.**
4. **Extensions: Modules for medicine lookup, lifestyle tips, or health records can be added.**

**11. Setup Instructions**

**Prerequisites**

* **Python 3.9+**
* **Hugging Face model access**
* **Google Colab account**

**Steps**

1. **Open healthai\_colab.ipynb in Google Colab.**
2. **Install required libraries using:**

**python**

**!pip install transformers torch gradio**

**3.Load the Granite model from Hugging Face.**

**4.Run the notebook cells to start the chatbot interface.**

**12. Folder Structure**

**HealthAI/**

**│── healthai\_colab.ipynb # Main notebook**

**│── modules/ # Disease prediction, treatment logic**

**│── docs/ # Documentation files**

**│── data/ # Sample symptom datasets (if used)**

**│── README.md # Overview**

**│── requirements.txt # Dependencies**

**13. Running the Application**

**To run HealthAI:**

1. **Launch the Colab notebook.**
2. **Install dependencies.**
3. **Load the Granite model.**
4. **Interact with the chatbot by typing questions or symptoms.**
5. **View predictions and treatment suggestions in real time.**

**14. API Documentation**

**Currently, HealthAI runs in Colab, but it can be extended into APIs using FastAPI:**

* **POST /chat → Takes user input and returns AI-generated response.**
* **POST /predict-disease → Accepts symptoms, predicts possible conditions.**
* **GET /treatment-plan → Returns preventive measures and treatment suggestions.**

**15. Authentication & Security**

* **Current version: Open access in Colab for research use.**
* **Future deployment can include:**
  + **Token-based authentication (JWT).**
  + **Role-based access (patients, doctors, researchers).**
  + **Encrypted data handling for sensitive health information.**

**16. User Interface Design**

**The user interface in Colab is text-based, but a Gradio app can provide a more interactive experience with:**

* **Chat bubbles.**
* **Symptom selection dropdowns.**
* **Result cards showing predictions and treatment suggestions.**

**This ensures that non-technical users can easily interact with the system.**

**17. Testing & Validation**

**HealthAI was tested through:**

* **Unit Testing: Ensuring prompt engineering and model response functions worked correctly.**
* **Manual Testing: Querying common health issues (e.g., cold, fever, headache).**
* **Edge Case Testing: Handling incomplete or vague user inputs.**

**Results showed that while HealthAI provides reliable general guidance, it cannot yet handle very rare or complex medical queries.**

**18. Results & Discussion**

**The system successfully demonstrated:**

* **Clear, conversational responses to medical queries.**
* **Symptom-based predictions that align with common medical knowledge.**
* **Treatment suggestions that emphasized safe and preventive care.**

**Feedback indicated that HealthAI can help reduce anxiety for patients and guide them toward professional care when needed.**

**19. Limitations**

* **Not a replacement for professional diagnosis.**
* **Predictions may not always be accurate for rare conditions.**
* **Limited scope in medical databases compared to enterprise solutions.**
* **Dependent on internet connectivity.**

**20. Future Enhancements**

* **Integration with speech recognition and voice output.**
* **Expansion to multi-language support for wider accessibility.**
* **Connection with wearable devices to track vitals like heart rate and oxygen levels.**
* **Development of a mobile app or web dashboard for real-world use.**
* **Integration with medical image analysis for X-rays or scans.**

**21. Conclusion**

**HealthAI demonstrates the power of artificial intelligence in enhancing healthcare accessibility. By providing a conversational assistant capable of predicting diseases and suggesting treatments, it empowers users with knowledge and encourages them to seek timely medical care.**

**While the project has limitations, it lays the foundation for future healthcare AI solutions that could be deployed at scale, reducing the pressure on hospitals and improving patient education worldwide.**

**22. References**

1. **Hugging Face Documentation – https://huggingface.co**
2. **IBM Watson Health – AI in Healthcare Reports**
3. **Ada Health – Symptom Checker Application**
4. **WHO Reports on Digital Health and AI (2022)**
5. **Research papers on AI-driven healthcare assistants (IEEE, Springer)**

**23.Python Code**

**import gradio as gr**

**import torch**

**from transformers import AutoTokenizer, AutoModelForCausalLM**

**# Load model and tokenizer**

**model\_name = "ibm-granite/granite-3.2-2b-instruct"**

**tokenizer = AutoTokenizer.from\_pretrained(model\_name)**

**model = AutoModelForCausalLM.from\_pretrained(**

**model\_name,**

**torch\_dtype=torch.float16 if torch.cuda.is\_available() else torch.float32,**

**device\_map="auto" if torch.cuda.is\_available() else None**

**)**

**if tokenizer.pad\_token is None:**

**tokenizer.pad\_token = tokenizer.eos\_token**

**def generate\_response(prompt, max\_length=1024):**

**inputs = tokenizer(prompt, return\_tensors="pt", truncation=True, max\_length=512)**

**if torch.cuda.is\_available():**

**inputs = {k: v.to(model.device) for k, v in inputs.items()}**

**with torch.no\_grad():**

**outputs = model.generate(**

**\*\*inputs,**

**max\_length=max\_length,**

**temperature=0.7,**

**do\_sample=True,**

**pad\_token\_id=tokenizer.eos\_token\_id**

**)**

**response = tokenizer.decode(outputs[0], skip\_special\_tokens=True)**

**response = response.replace(prompt, "").strip()**

**return response**

**def disease\_prediction(symptoms):**

**prompt = f"Based on the following symptoms, provide possible medical conditions and general medication suggestions. Always emphasize the importance of consulting a doctor for proper diagnosis.\n\nSymptoms: {symptoms}\n\nPossible conditions and recommendations:\n\n\*\*IMPORTANT: This is for informational purposes only. Please consult a healthcare professional for proper diagnosis and treatment.\*\*\n\nAnalysis:"**

**return generate\_response(prompt, max\_length=1200)**

**def treatment\_plan(condition, age, gender, medical\_history):**

**prompt = f"Generate personalized treatment suggestions for the following patient information. Include home remedies and general medication guidelines.\n\nMedical Condition: {condition}\nAge: {age}\nGender: {gender}\nMedical History: {medical\_history}\n\nPersonalized treatment plan including home remedies and medication guidelines:\n\n\*\*IMPORTANT: This is for informational purposes only. Please consult a healthcare professional for proper treatment.\*\*\n\nTreatment Plan:"**

**return generate\_response(prompt, max\_length=1200)**

**# Create Gradio interface**

**with gr.Blocks() as app:**

**gr.Markdown("# Medical AI Assistant")**

**gr.Markdown("\*\*Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.\*\*")**

**with gr.Tabs():**

**with gr.TabItem("Disease Prediction"):**

**with gr.Row():**

**with gr.Column():**

**symptoms\_input = gr.Textbox(**

**label="Enter Symptoms",**

**placeholder="e.g., fever, headache, cough, fatigue...",**

**lines=4**

**)**

**predict\_btn = gr.Button("Analyze Symptoms")**

**with gr.Column():**

**prediction\_output = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)**

**predict\_btn.click(disease\_prediction, inputs=symptoms\_input, outputs=prediction\_output)**

**with gr.TabItem("Treatment Plans"):**

**with gr.Row():**

**with gr.Column():**

**condition\_input = gr.Textbox(**

**label="Medical Condition",**

**placeholder="e.g., diabetes, hypertension, migraine...",**

**lines=2**

**)**

**age\_input = gr.Number(label="Age", value=30)**

**gender\_input = gr.Dropdown(**

**choices=["Male", "Female", "Other"],**

**label="Gender",**

**value="Male"**

**)**

**history\_input = gr.Textbox(**

**label="Medical History",**

**placeholder="Previous conditions, allergies, medications or None",**

**lines=3**

**)**

**plan\_btn = gr.Button("Generate Treatment Plan")**

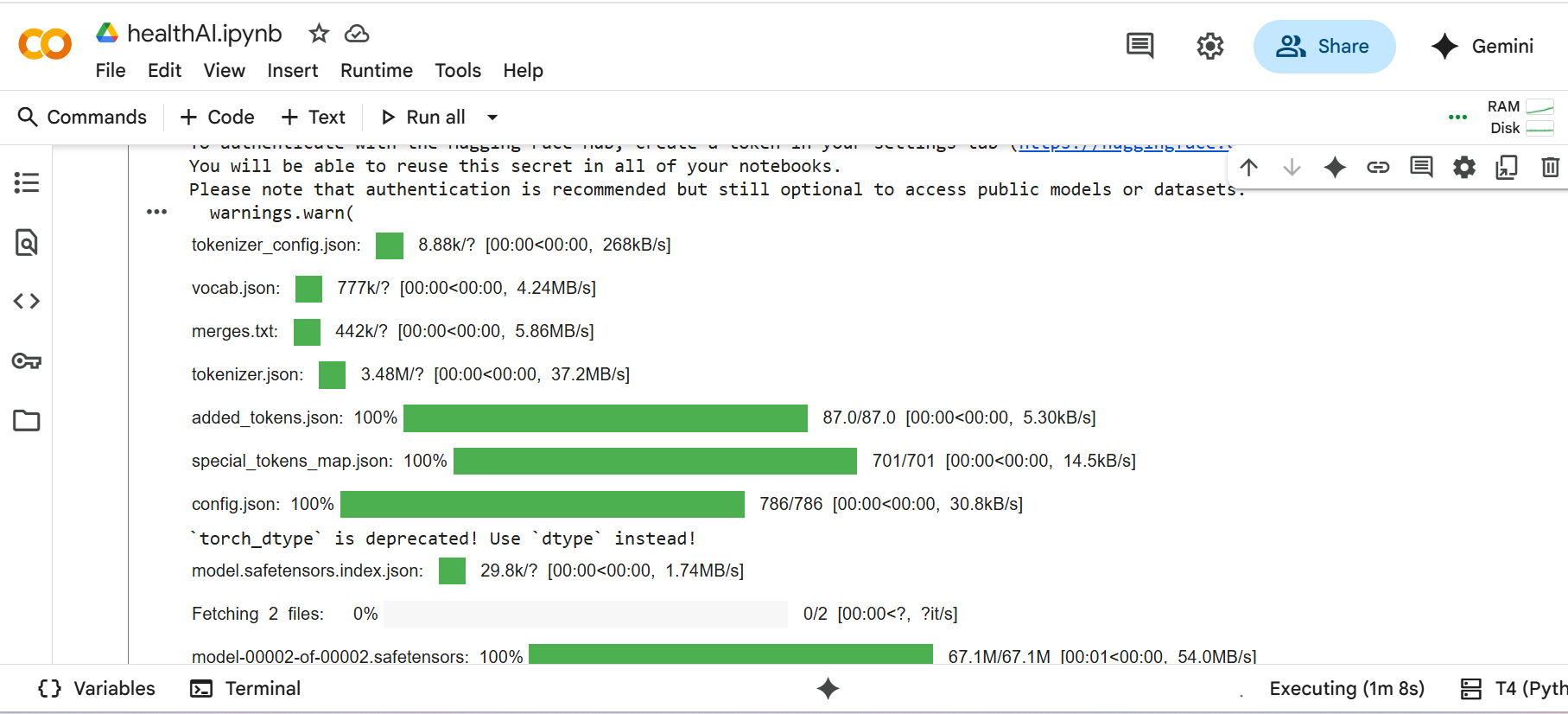
**with gr.Column():**

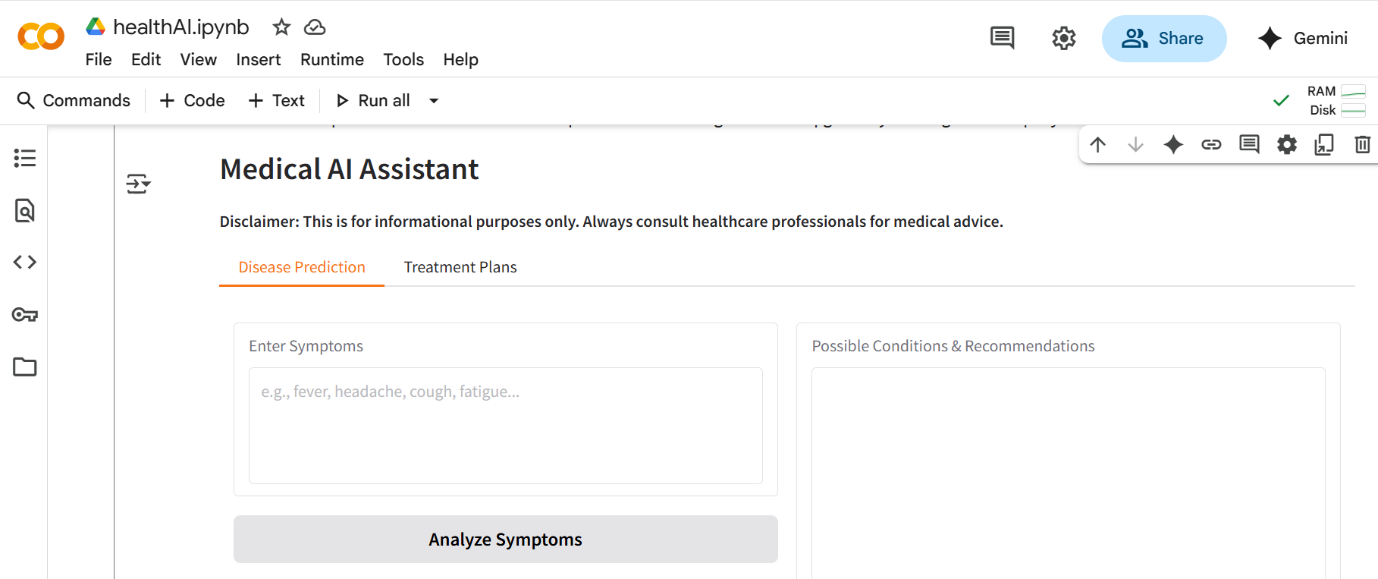
**plan\_output = gr.Textbox(label="Personalized Treatment Plan", lines=20)**

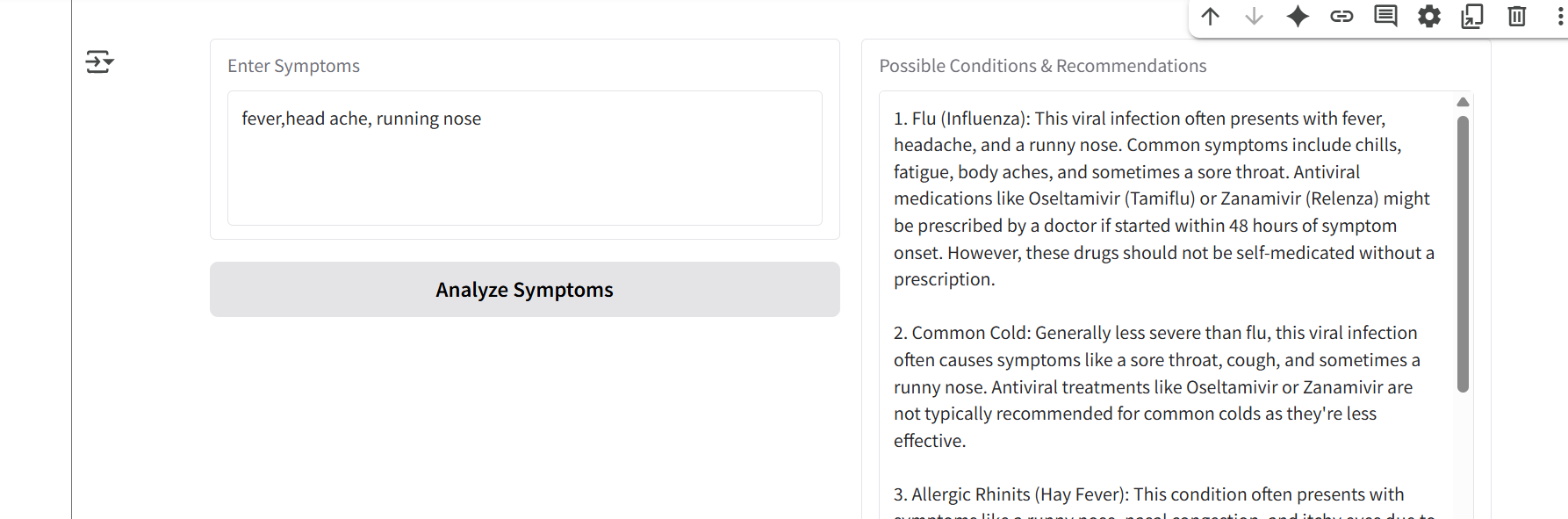
**plan\_btn.click(treatment\_plan, inputs=[condition\_input, age\_input, gender\_input, history\_input], outputs=plan\_output)**

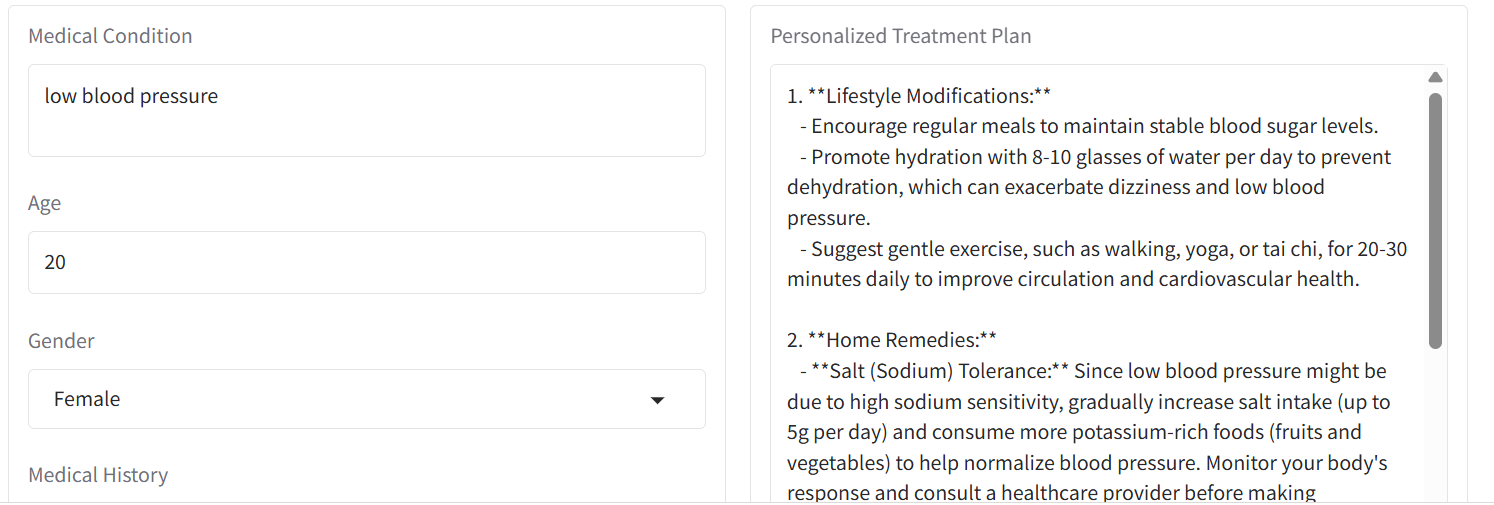
**app.launch(share=True)**

**24.Program execution screen shots**

****

****

****

****