**Project Documentation**

**1. Introduction**

**Project Title:**

Health AI

**Team Members:**

* S. Gokul (Team leader)
* K. Sachin
* J. Bala Krishnan
* A. Naina Mohammed

**Introduction**:

The Health AI is an innovative healthcare-focused AI application that leverages large language models (LLMs) to analyze user inputs such as symptoms, conditions, and medical history, and return generalized insights in the form of disease predictions and treatment suggestions. The system is designed to empower individuals with preliminary health information while strongly emphasizing that professional healthcare providers must be consulted for actual diagnosis and treatment.

This project demonstrates how AI in healthcare can assist in reducing uncertainty, improving health awareness, and offering personalized suggestions, all while being guided by responsible AI principles.

---

**2. Project Overview**

**Purpose:**

The purpose of the Health AI is to bridge the gap between individuals seeking quick health-related guidance and the overwhelming amount of unstructured medical information online. Unlike random search engine results, this tool provides structured, conversational, and context-aware responses tailored to user input.

**Benefits:**

* Helps users better understand their symptoms.
* Provides a starting point for health discussions with doctors.
* Increases awareness about general home remedies and medication guidelines.
* Accessible to people without medical expertise through natural language queries.

**Features:**

**1. Disease Prediction**

* Input: User-provided symptoms (e.g., fever, cough, fatigue)
* Output: A list of possible medical conditions with general lifestyle or medication recommendations.
* Value: Provides initial guidance and narrows down possible causes for further medical consultation.

**2. Treatment Plan Generator**

* Input: Medical condition, age, gender, and medical history.
* Output: Personalized treatment suggestions, including lifestyle changes, dietary advice, and general medication guidelines.
* Value: Helps individuals prepare before visiting a doctor and encourages proactive health management.

**3. Conversational Interface**

* Built using Gradio, offering a user-friendly interface with tabbed navigation.
* Interactive buttons and textboxes provide a smooth, chat-like experience.

**4. AI-Powered Responses**

* Powered by IBM Granite LLM, capable of natural language understanding and response generation.
* Ensures coherent, structured outputs instead of raw model predictions

**5. Ethical Design**

* Clear disclaimers are shown in both UI and responses.
* Emphasis on informational use only, avoiding overconfidence in medical accuracy.

**3. Architecture**

**Frontend (Gradio):**

* Tab-based design with dedicated sections for disease prediction and treatment plan generation.
* Accepts structured inputs such as symptoms, patient details, and medical history.
* Provides outputs in multi-line formatted textboxes.
* Lightweight, responsive, and easily shareable via share=True option.

**Backend (PyTorch & Transformers):**

* Uses Hugging Face Transformers to load the IBM Granite LLM and tokenizer.
* Handles tokenization, tensor creation, and decoding of model outputs.
* Automatically detects GPU availability and optimizes for CUDA if present.
* Manages prompt construction to guide the model toward safe, contextual responses.

**LLM Integration (IBM Granite LLM):**

* IBM Granite 3.2 2B Instruct model is optimized for instruction-following tasks.
* Prompt engineering ensures disclaimers and medically responsible outputs.
* Supports temperature-based sampling for balanced creativity and accuracy.

**4. Setup Instructions**

**Prerequisites:**

* Python 3.9 or later

**Installed dependencies:**

* pip install torch transformers gradio
* GPU (optional, recommended for large models)
* Internet access to load the model from Hugging Face

**Installation Process:**

1. Clone or download the repository.

2. Navigate into the project directory.

3. Install dependencies from requirements.txt.

4. Run the application:

python app.py

5. Access the Gradio web interface on:

Localhost (http://127.0.0.1:7860)

Shareable public link (if share=True is enabled).

**5. Folder Structure**

project/

│

├── app.py # Main Gradio application

├── requirements.txt # Required dependencies

├── README.md # Documentation and usage guide

└── assets/ # (Optional) Icons, images, or logos for UI

This simple structure ensures readability, maintainability, and easy deployment.

**6. Running the Application**

**1. Start the application with:**

Google collab

**2. Once launched, Gradio will provide:**

* A local link for testing on your machine.
* A shareable link for accessing the app remotely.

**3. Navigate through the two main tabs:**

Disease Prediction: Enter symptoms and click Analyze Symptoms to get possible conditions.

Treatment Plans: Enter patient details and click Generate Treatment Plan for suggestions.

**4. Results are displayed in clear, formatted text areas for easy reading and copying.**

**7. API Documentation**

Currently, the application is designed as a UI-based system. However, future enhancements can add REST APIs for external integration. Example APIs could include:

POST /predict-disease

Input: JSON containing user symptoms.

Output: JSON with possible conditions and general recommendations.

POST /treatment-plan

Input: JSON with patient condition, age, gender, and history.

Output: JSON with treatment plan suggestions.

These endpoints would allow integration into mobile apps, EHR systems, and telehealth platforms.

**8. Authentication**

Current Version: No authentication (open demo).

Future Plans:

Token-based authentication (JWT, API keys).

Role-based access for patients, doctors, and admins.

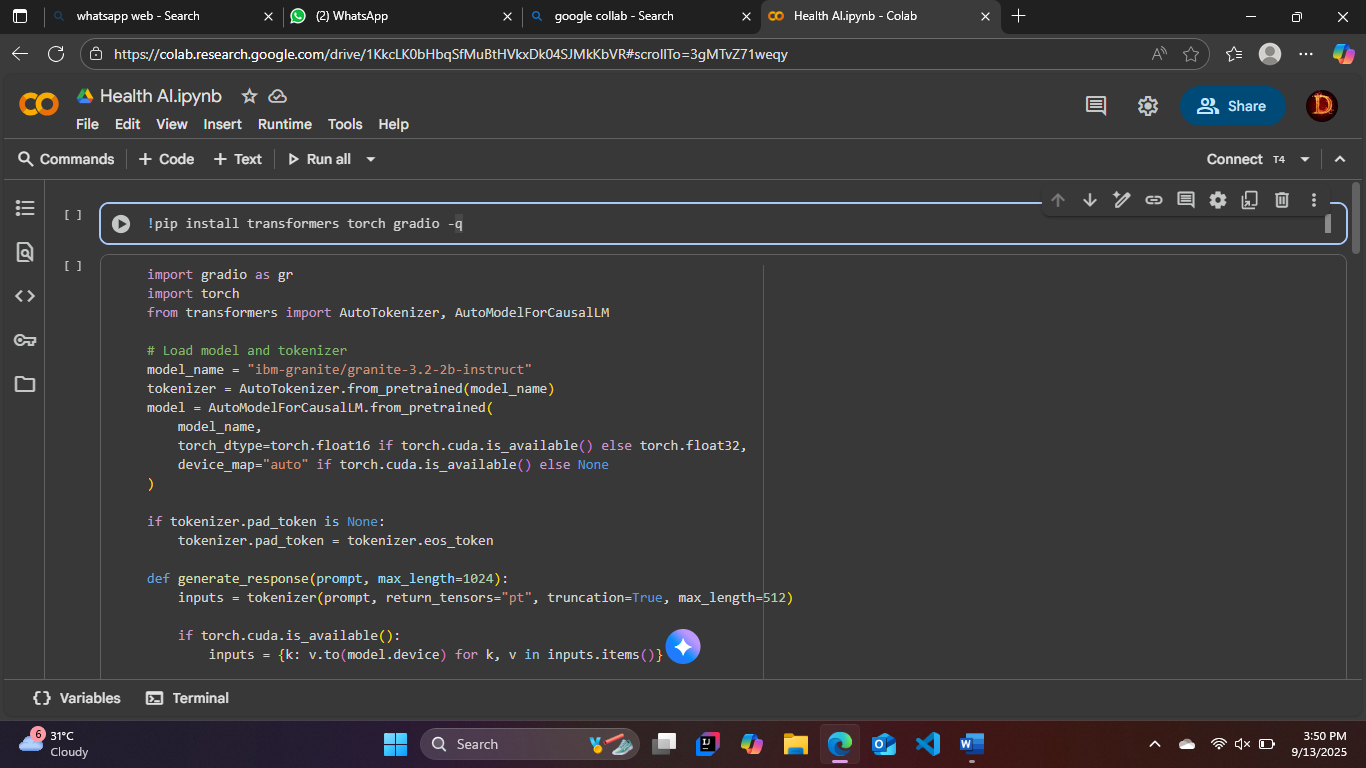
Secure storage for patient interaction logs.

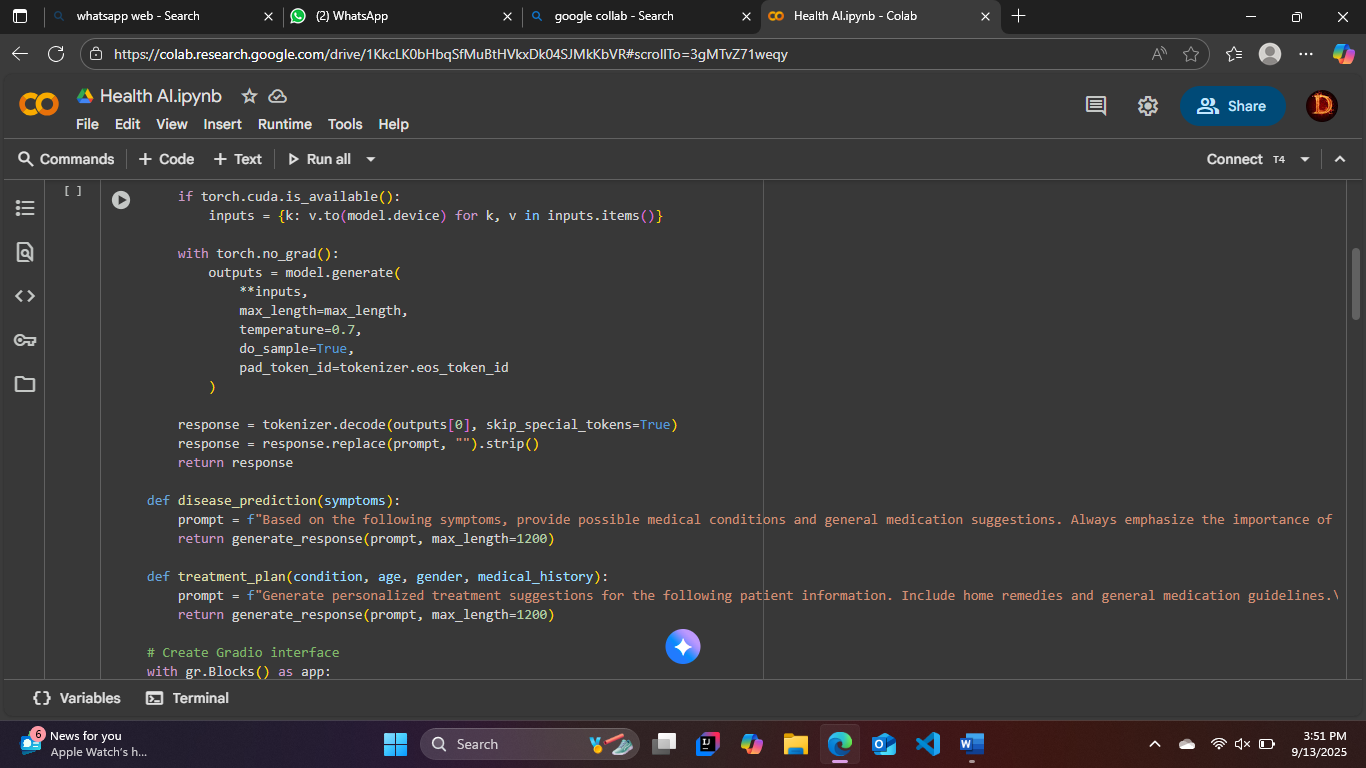
**9. User Interface**

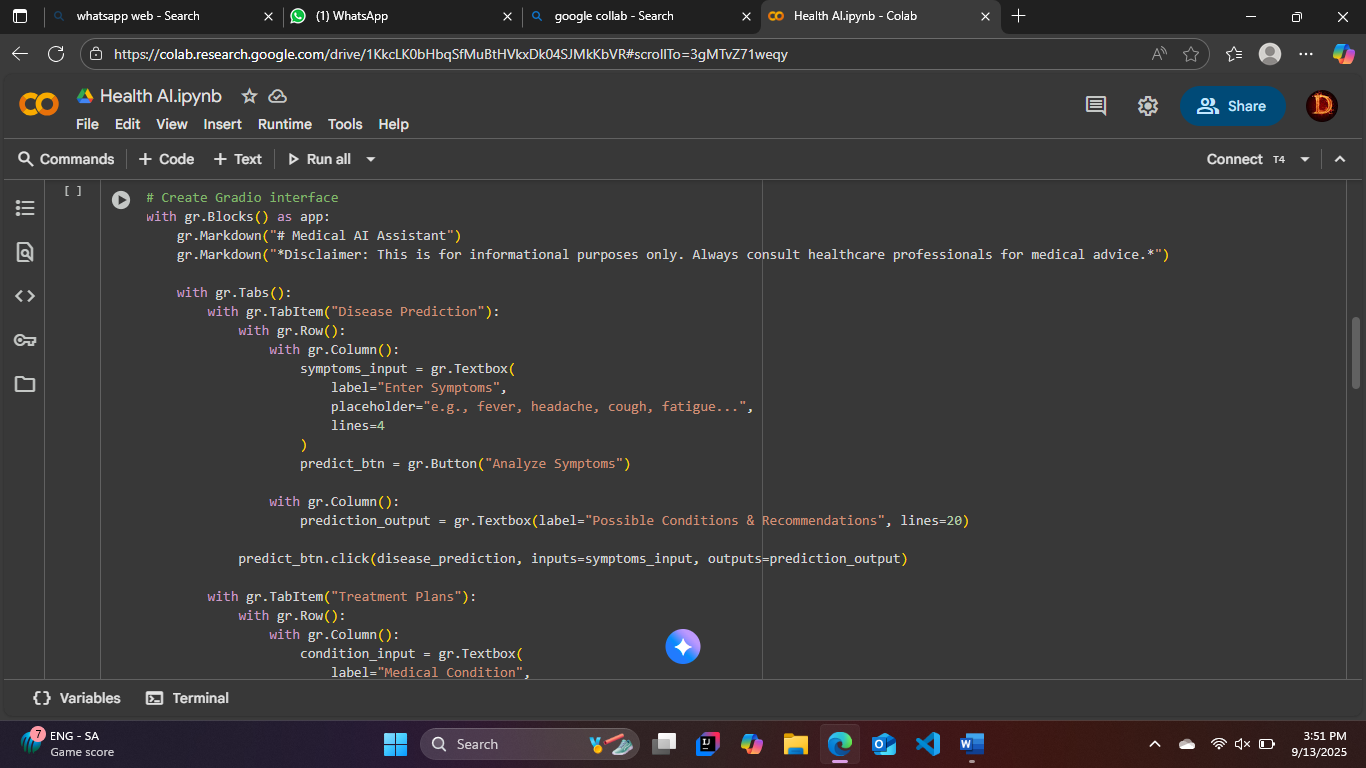
Tabbed Layout: Two tabs for separate workflows (Disease Prediction & Treatment Plans).

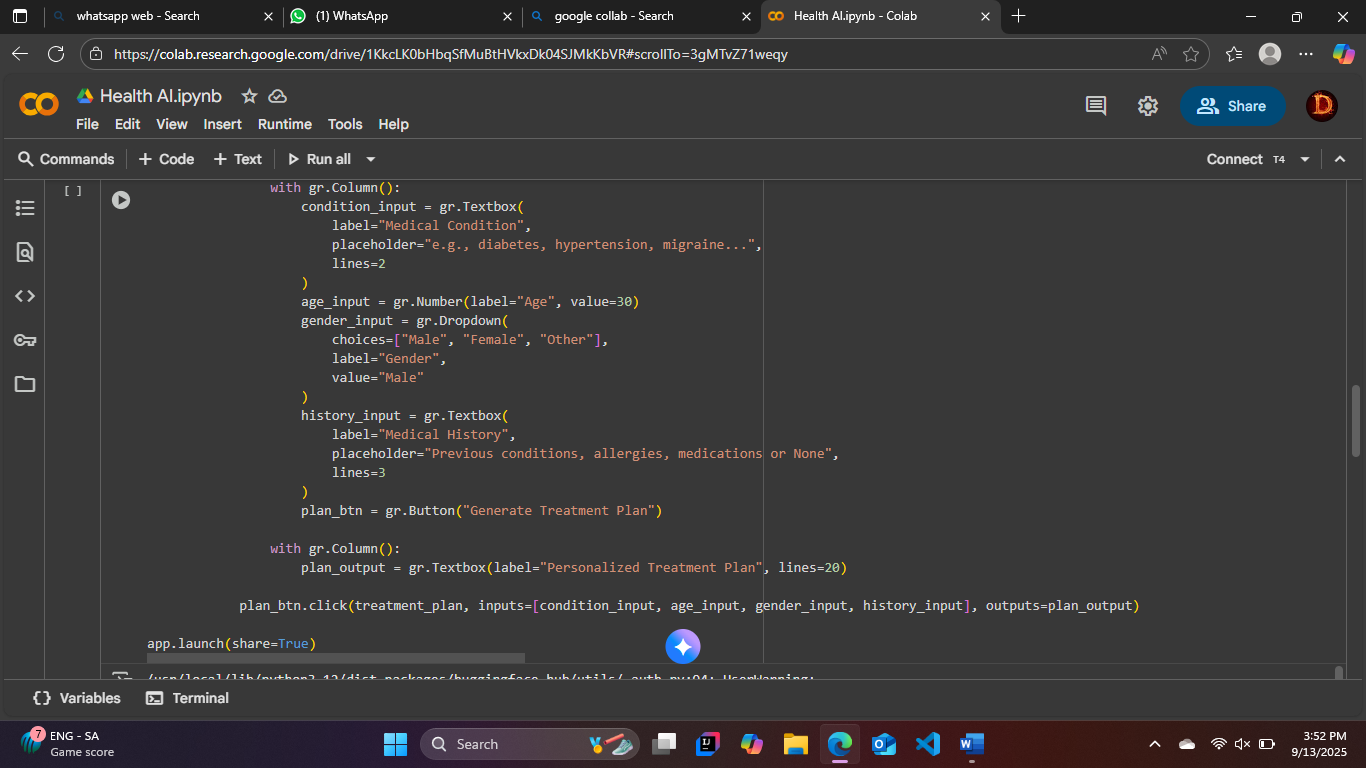
Inputs:

* Textbox for symptoms.
* Dropdown for gender.
* Number field for age.
* Textbox for medical history.



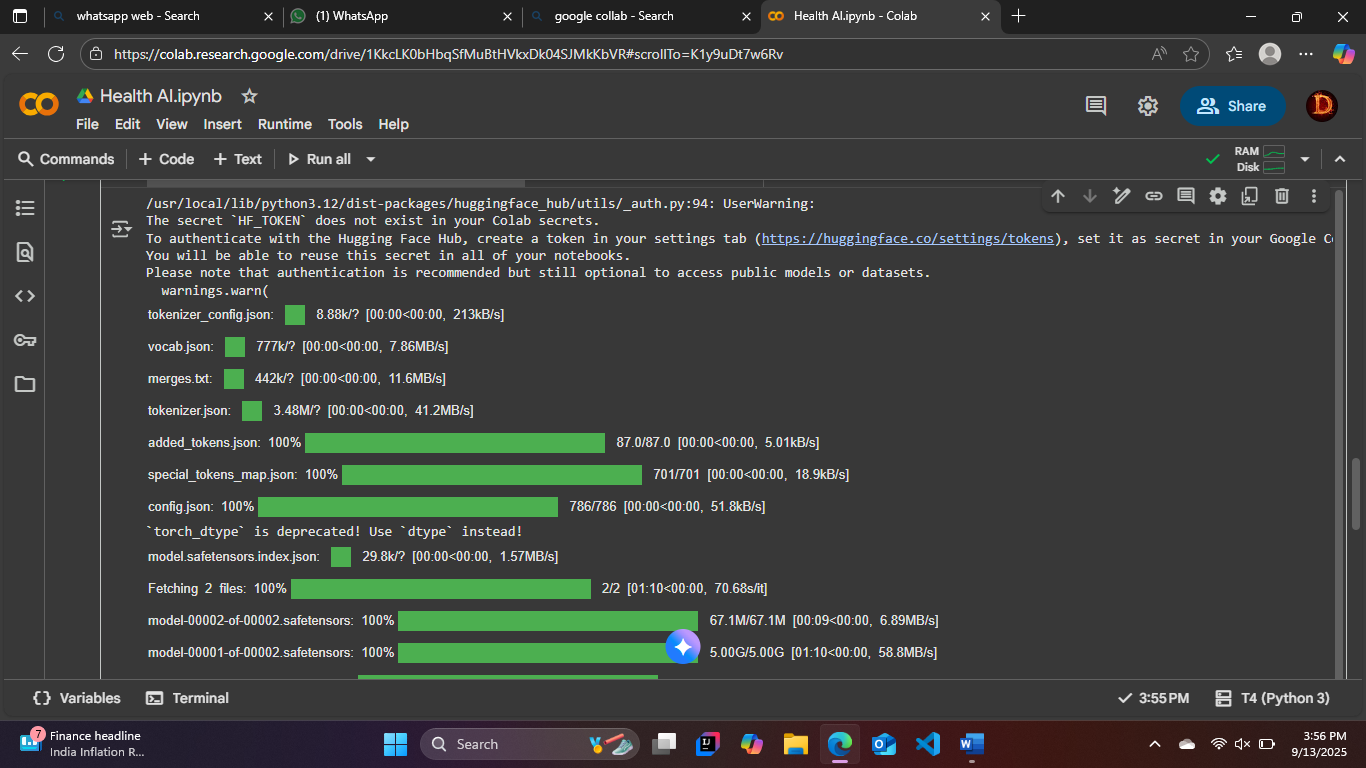


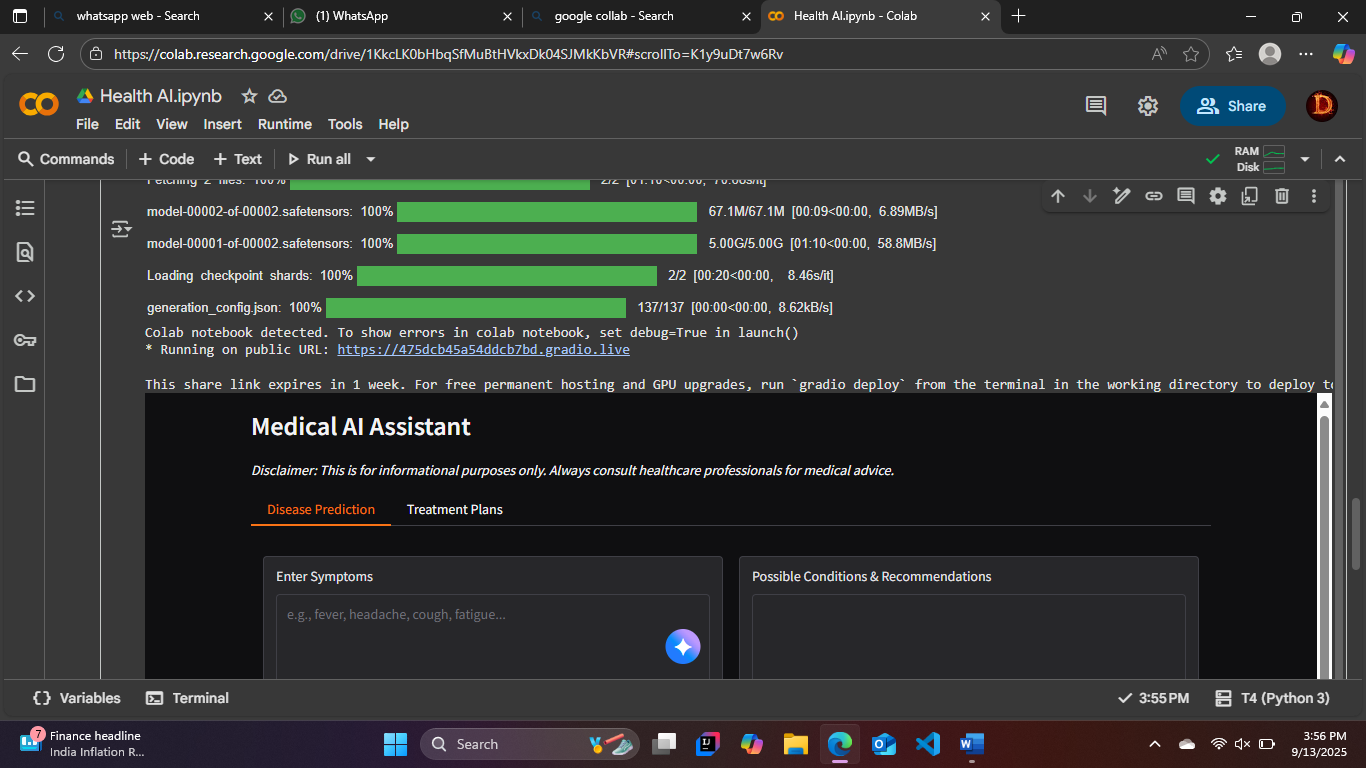


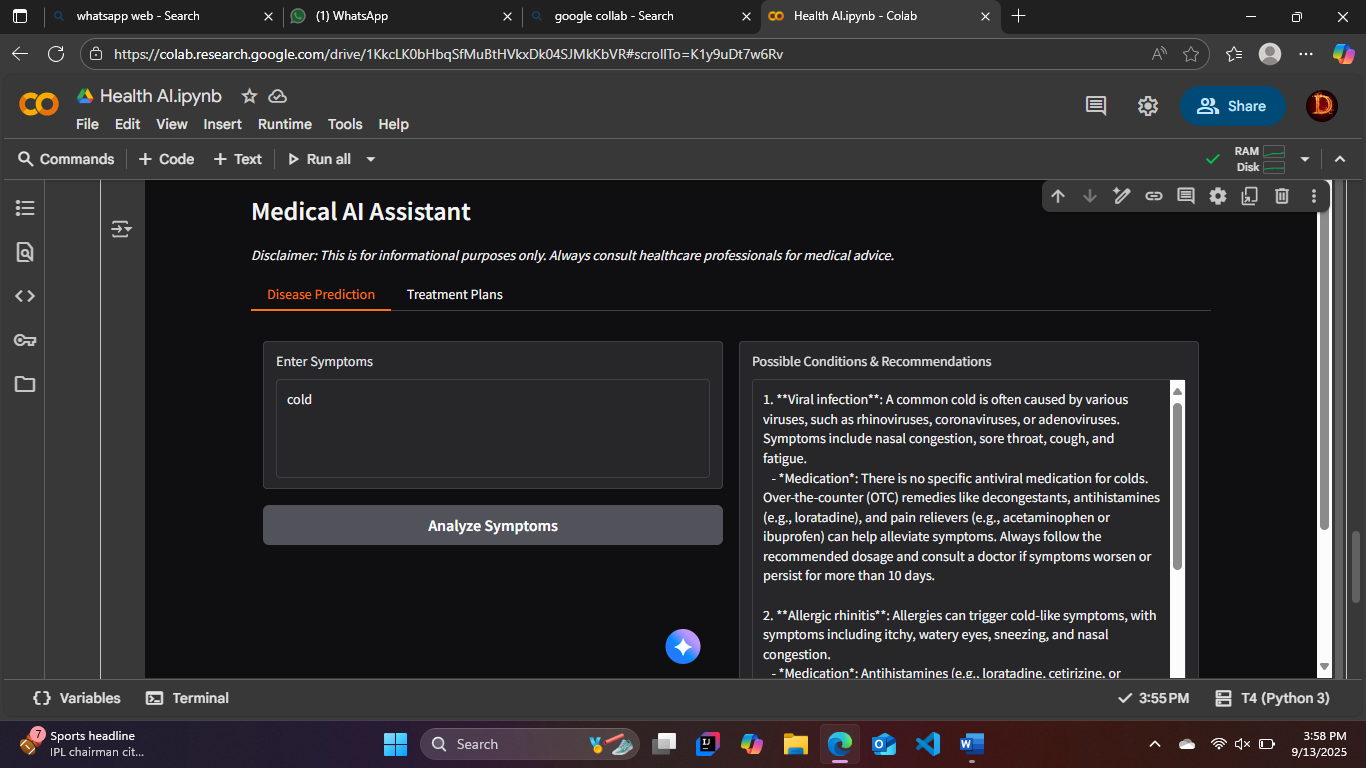


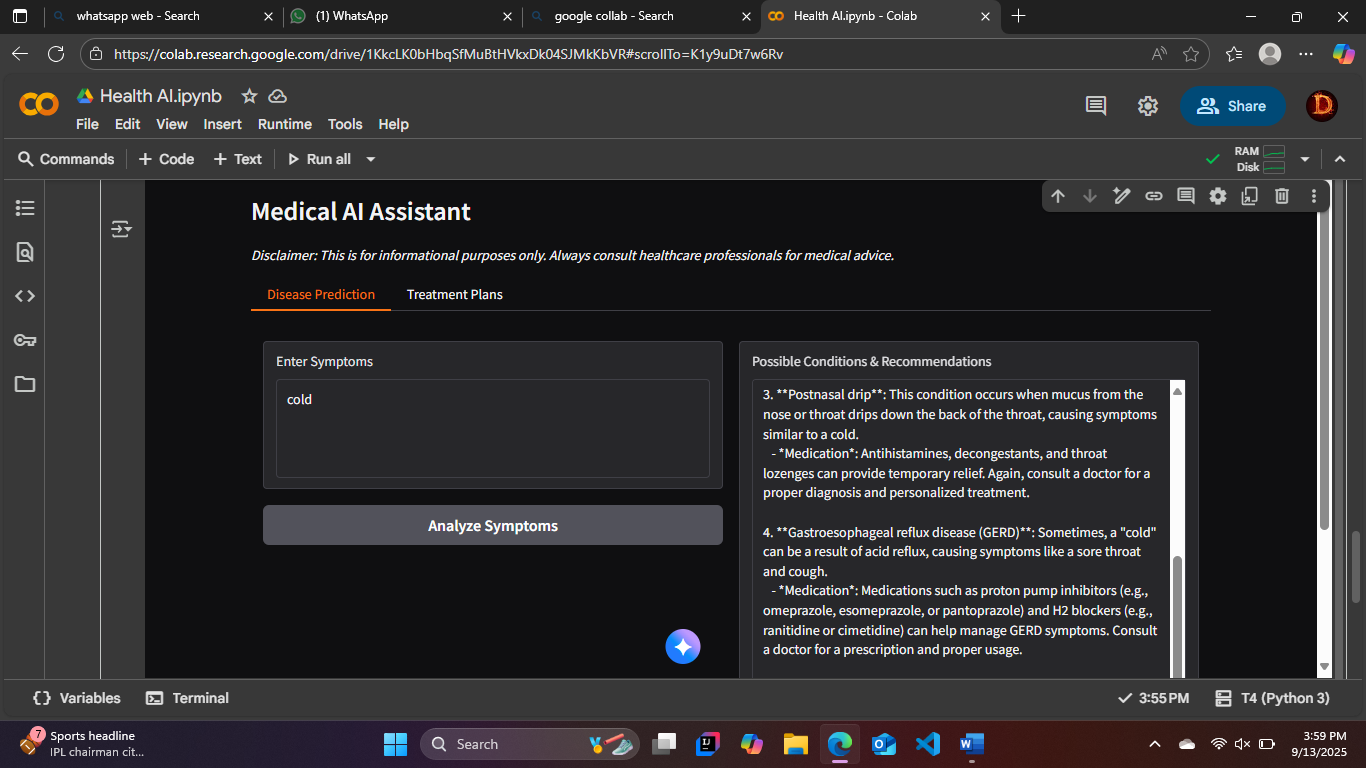
Outputs:

Multi-line textboxes for results.









**Interaction:**

* Simple, intuitive, and mobile-friendly.
* Emphasis on clarity and accessibility.

**10. Testing**

* Testing performed includes:
* Unit Testing: Validation of prompt construction and tokenization.
* Manual Testing: Testing across different symptom/condition inputs.

Performance Testing: Comparing GPU vs CPU performance.

**Edge Case Handling:**

* Empty or incomplete inputs.
* Extremely long text entries.
* Non-standard characters.

Outcome: The system reliably generates responses but requires disclaimers for accuracy.

**11. Known Issues**

* AI responses may vary depending on input phrasing.
* Outputs may be too generic for complex or rare medical conditions.
* Long prompts sometimes exceed model token limits, leading to truncated responses.
* LLM knowledge may not reflect latest medical research.

**12. Future Enhancements**

* Integration with Medical Knowledge Bases (e.g., WHO, CDC, PubMed).
* Patient Session History Tracking for continuous monitoring.
* Multi-Language Support to expand accessibility globally.
* Risk Level Classification to flag urgent cases.
* Voice-based Interaction for hands-free usability.
* API Exposure for mobile/enterprise healthcare systems.
* Fine-tuned Models trained specifically on medical literature.