

# **HEALTH AI : Intelligent Learning Healthcare Assistant Using IBM Granite**

## **PROJECT DOCUMENTATION**

### **1.Introduction**

Project title : **Health AI**

Team leader : **Kannika parameswari.T**

Team member : **Padma Sri .P.K**

Team member : **Dharshini.A**

Team member : **Jenipher.V**

### **2. Project Overview**

The "Health AI" project is a generative AI application that provides basic medical insights and information. Built on IBM's Granite model, the application serves as a healthcare assistant, offering functionalities like disease prediction and personalized treatment plan suggestions. The project's core purpose is to make health information more accessible and user-friendly, while a crucial disclaimer emphasizes that the information is for guidance only and not a substitute for professional medical advice

### **3.Features:**

#### **Disease Prediction:**

Users input a list of symptoms, and the model analyzes them to suggest possible medical conditions and general medication. This feature leverages the model's reasoning capabilities to connect disparate symptoms to likely diagnoses.

## **Personalized Treatment Plans:**

Given a medical condition, along with the user's age, gender, and medical history, the AI generates a customized treatment plan. This includes both home remedies and medication guidelines, showcasing the model's ability to generate tailored, context-aware responses

## **4.Core Functionalities**

The application has two main features:

### **Disease Prediction:**

Users can enter a list of symptoms, and the AI model will provide a list of possible medical conditions and general medication suggestions.

### **Treatment Plans:**

Based on a specified medical condition, age, gender, and medical history, the application generates a personalized treatment plan, including home remedies and general medication guidelines.

It's important to note that the application includes a clear disclaimer that the information provided is for informational purposes only and users should always consult a healthcare professional for a proper diagnosis and treatment.

## **5.Technology Stack**

The project leverages a number of powerful tools and libraries to function.

### **Generative AI Model:**

- The core of the application is the IBM Granite model, specifically the granite-3.2-2b-instruct version, which is a lightweight and fast model fine-tuned for reasoning tasks. The model processes user inputs to generate personalized and data-driven medical guidance.

## **Frameworks:**

- **Gradio:**  
The user interface (UI) is built with the Gradio framework, which makes it easy to create and share interactive web applications for machine learning models. The UI consists of two tabs: "Disease Prediction" and "Treatment Plans".
- **Hugging Face transformers:**  
This library is used to load and work with the IBM Granite model and its tokenizer.
- **Python:**  
The entire application logic is written in Python, using libraries like torch for handling the model's tensors and GPU compatibility.
- **Deployment:**
  - The application is designed to be deployed on Google Colab. Google Colab provides a free environment with access to a T4 GPU, which is essential for running the large AI model efficiently.

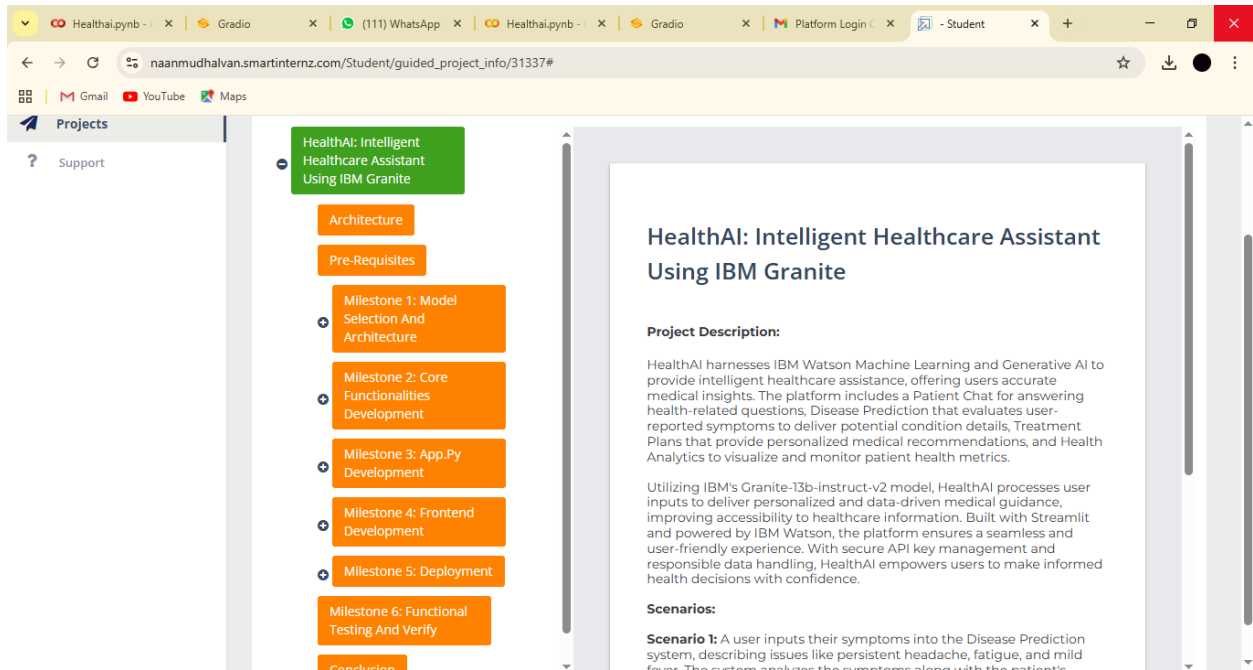
## **6.Project Workflow**

Building this project involved a clear, step-by-step process:

- i. Exploring the Project Portal
- ii. Choosing the AI Model.
- iii. Running the Application on Google Collab
- iv. Uploading the Project to GitHub

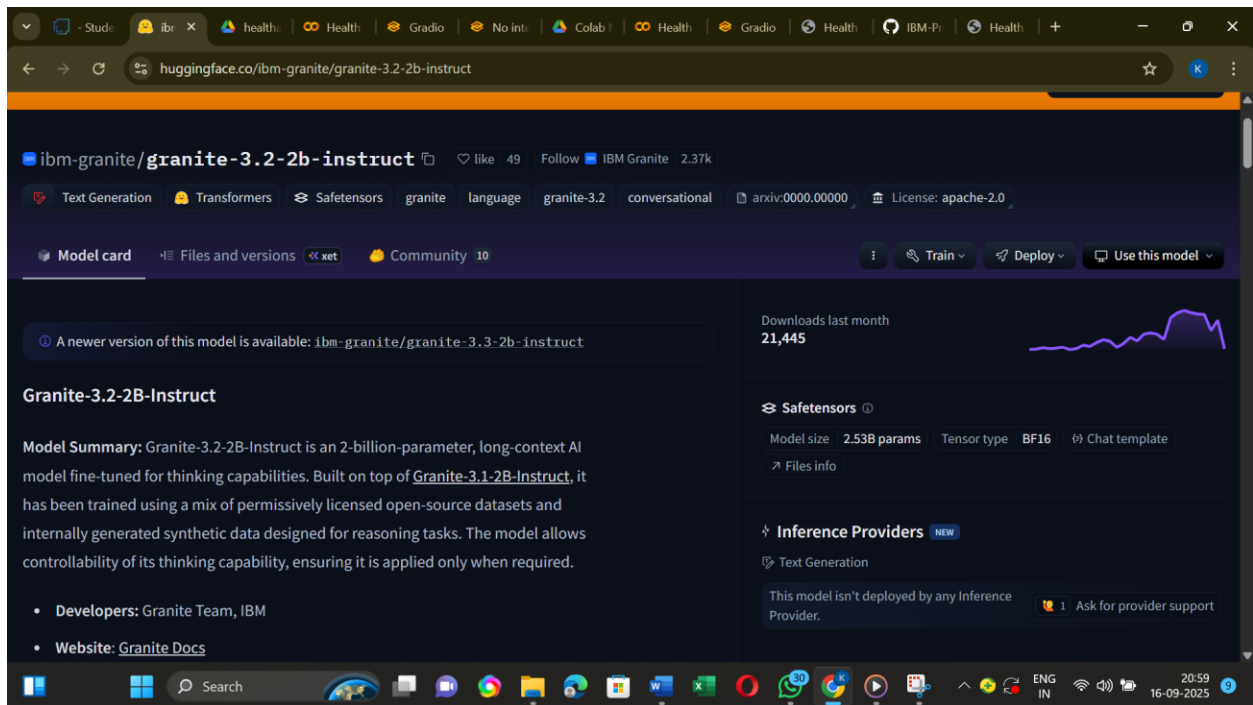
### **1.Exploring the Project Portal:**

The project began by accessing the "Naan Mudhalvan - SmartInternz" portal to understand the project requirement and details.



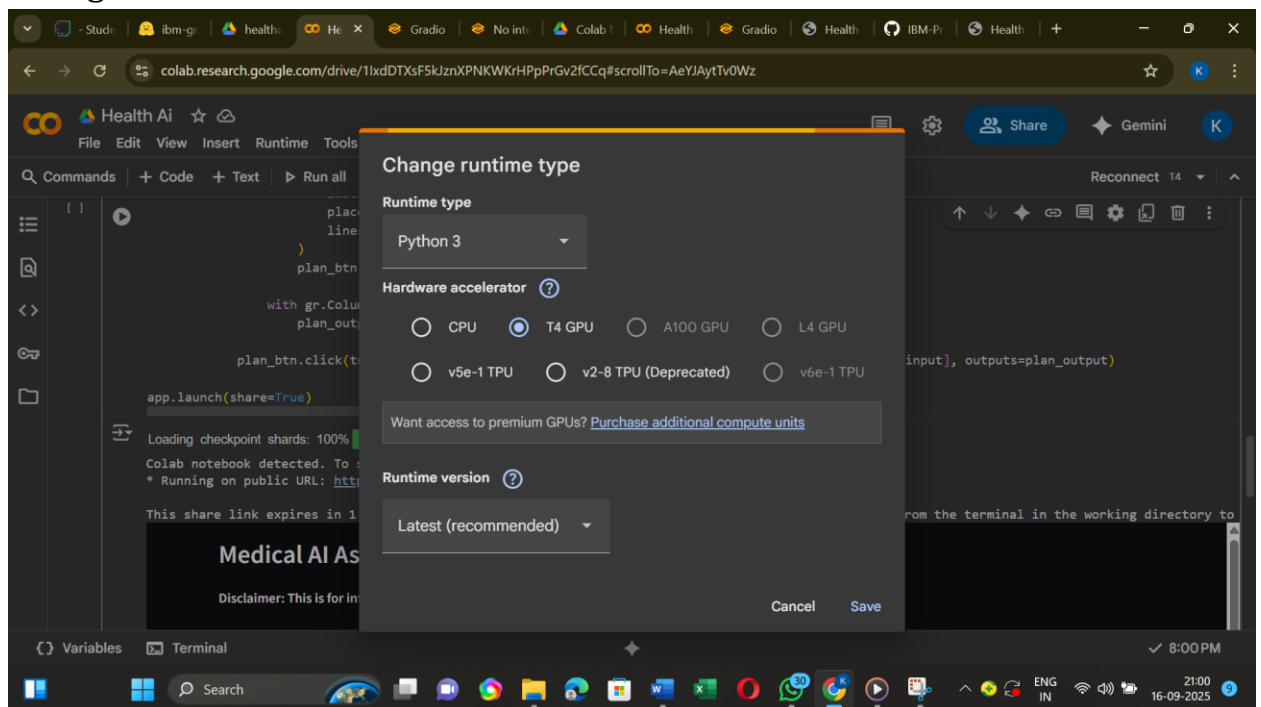
## 2.Choosing the AI Model:

An IBM Granite model was selected from Hugging Face, a platform that hosts and allows collaboration on public AI models and datasets. The ibm-granite/granite-3.2-2b-instruct model was chosen for its performance and efficiency.

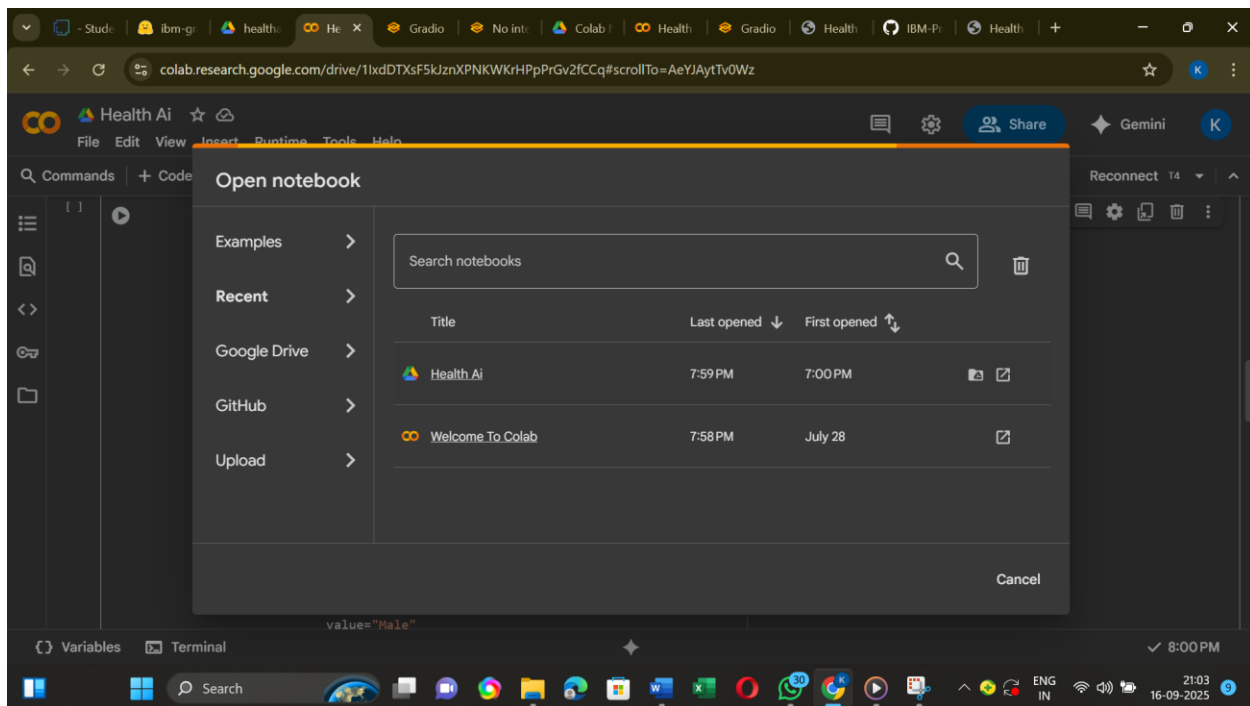


### 3. Running the Application on Google Colab:

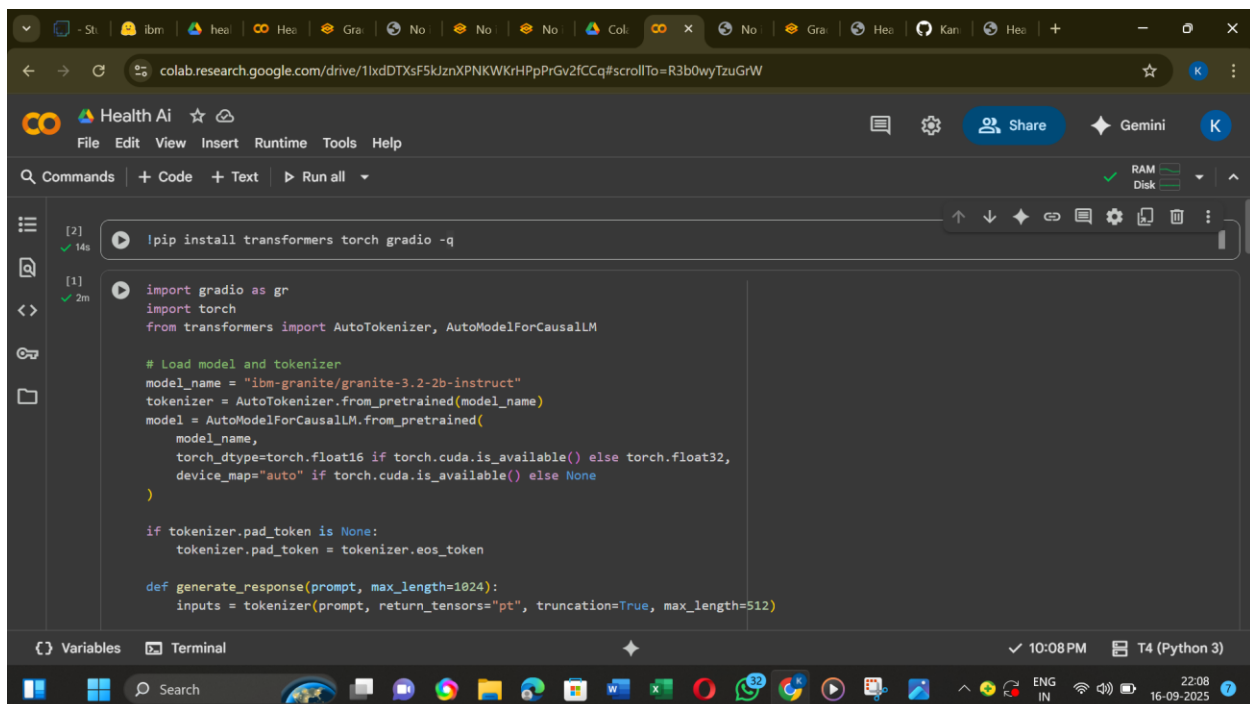
- A new notebook was created in Google Colab and the runtime was configured to use a T4 GPU. Then Save.



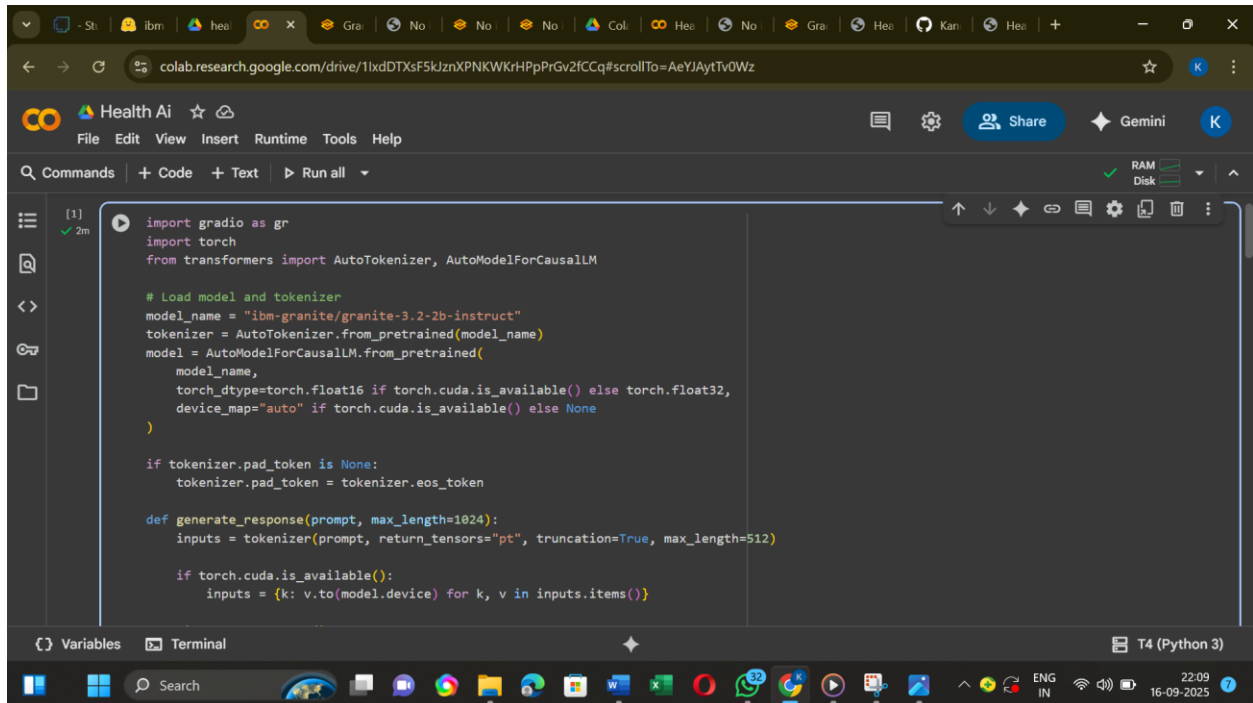
- Go to file, click Open notebook.



- Required libraries like transformers, torch, and gradio were installed.



- The Python code, which handles loading the model, defining the generation functions, and creating the Gradio interface, was added to the notebook and executed.



The screenshot shows a Google Colab notebook titled "Health Ai". The code in the notebook is as follows:

```
[1] ✓ 2m
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
)

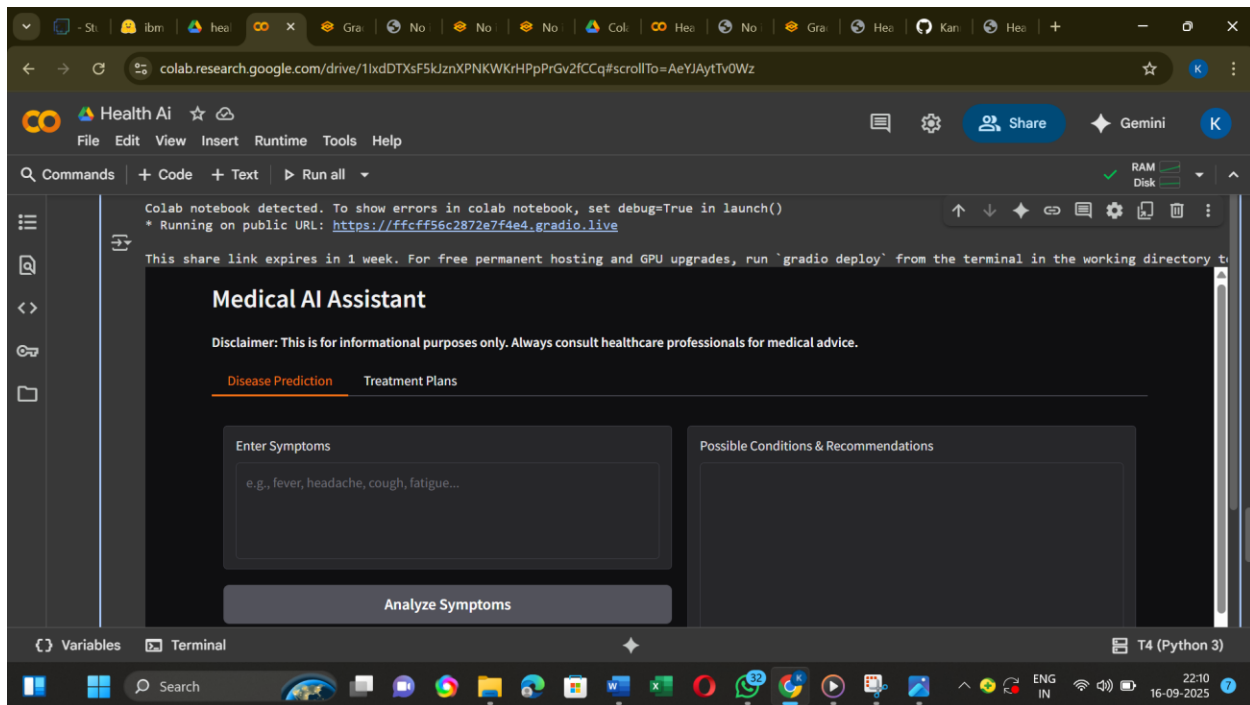
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

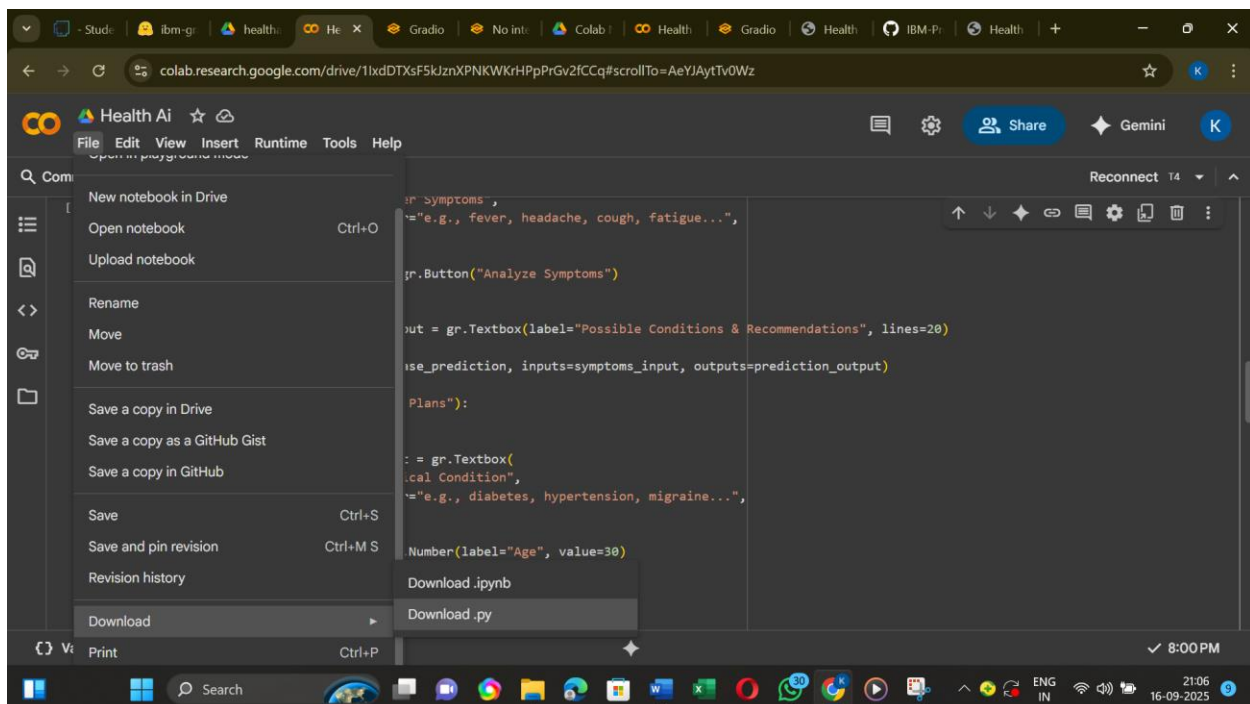
    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}
```

The interface includes a top bar with "Health Ai" and a "Share" button. The bottom bar shows "T4 (Python 3)" and the system clock "22:09 16-09-2025".

- The application launched, providing a public URL to access the "Health AI" interface.



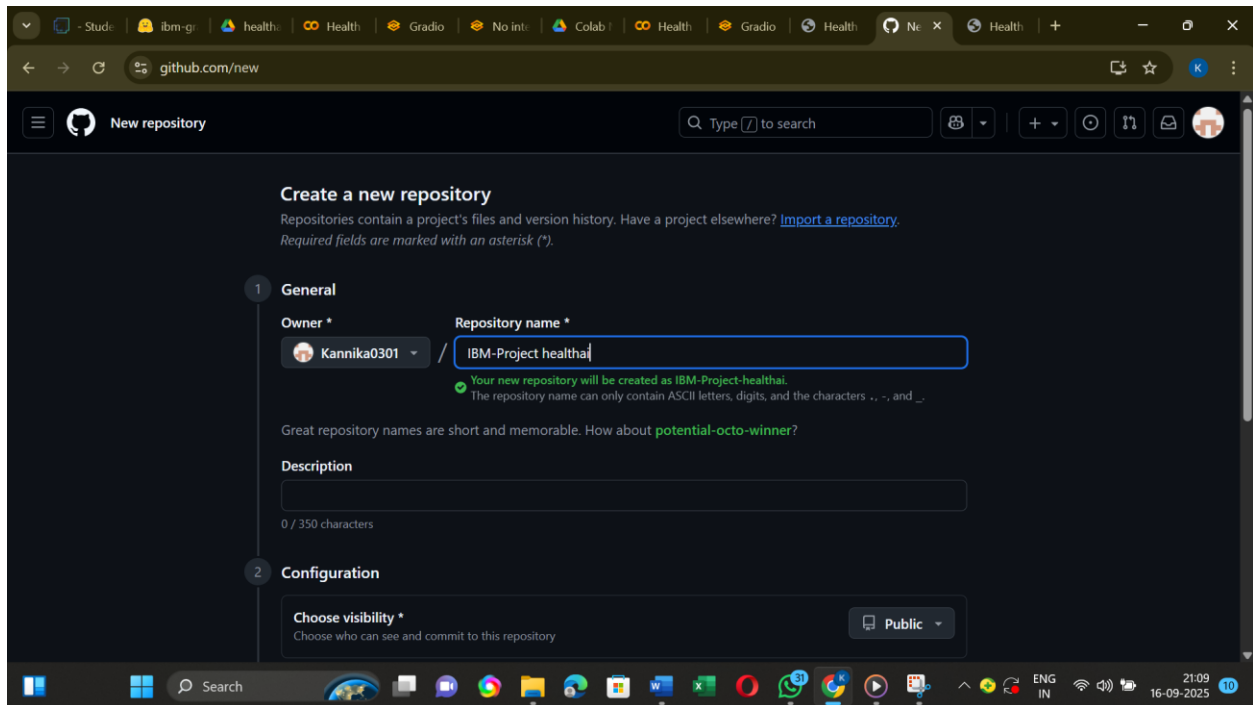
- Go to file and download the project as Download.py.

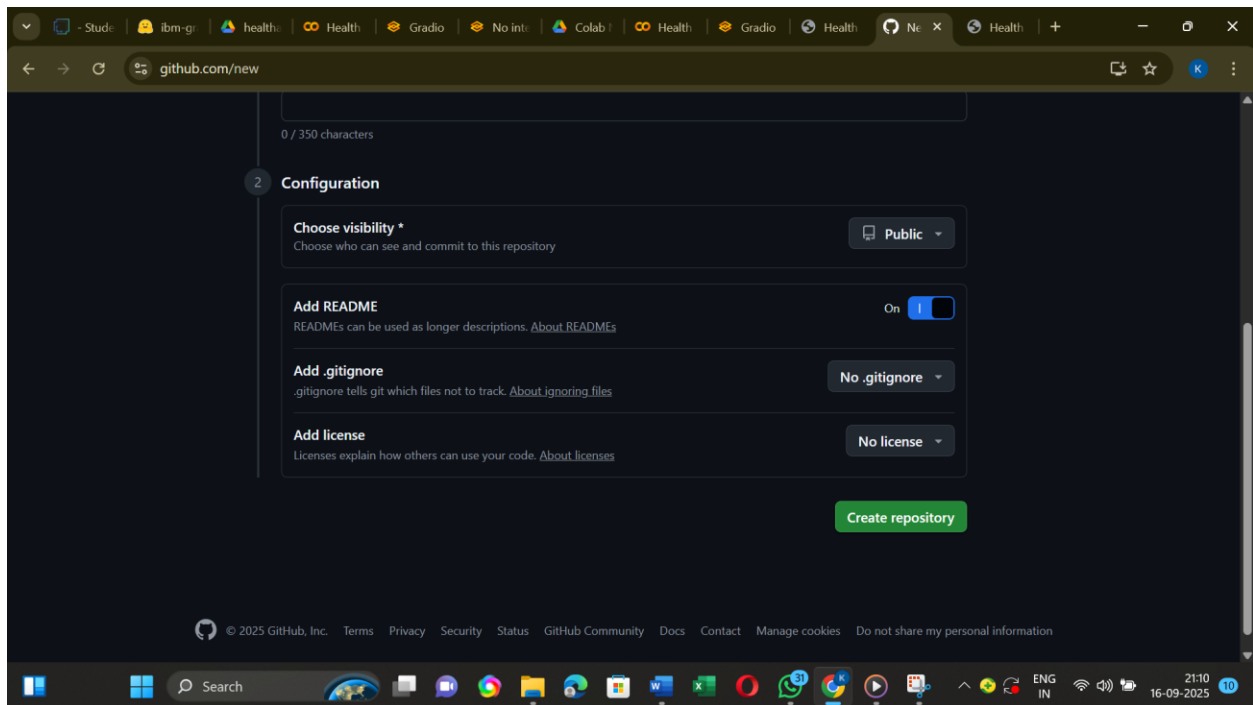




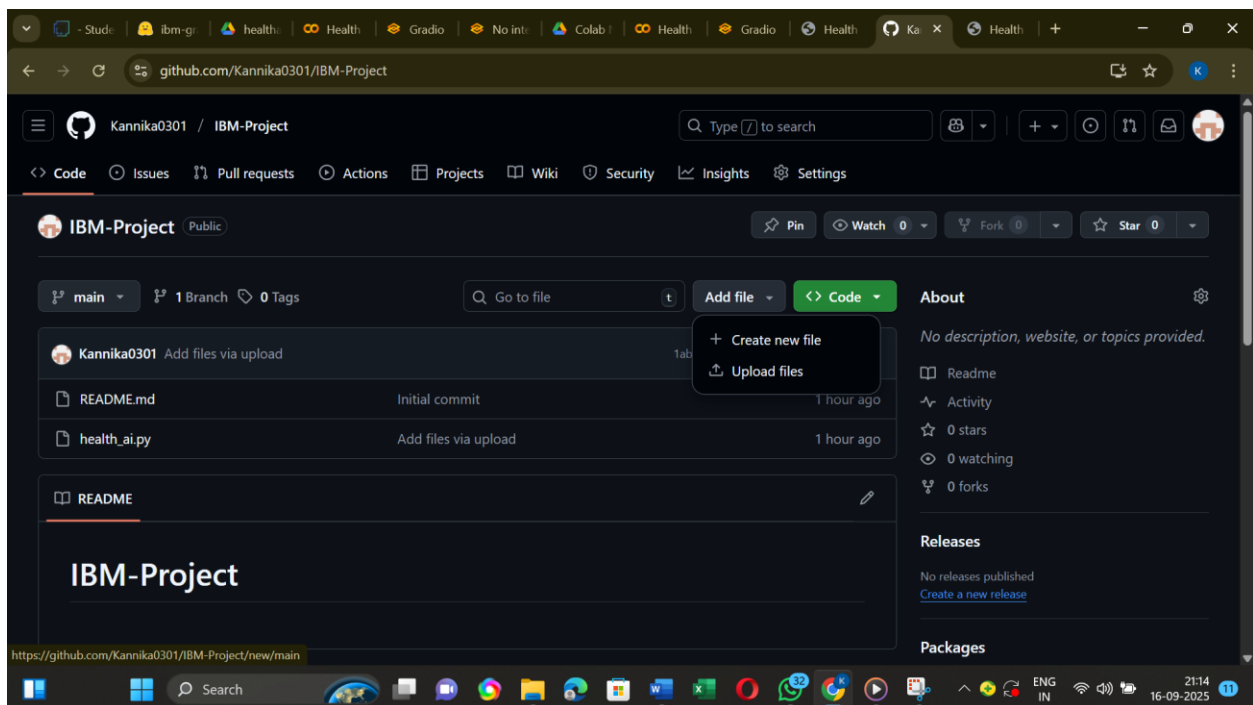
## 4.Uploading the Project to GitHub:

- The Python code from the Google Colab notebook was downloaded as a .py file.
- Use your own repository name.
- In configuration, ADDREADME should be turned On.
- A new repository was created on GitHub to store the project files.





- The downloaded Python file was then uploaded to the GitHub repository.



## 7.Interface: A Gradio-based Medical AI Assistant

The user interface for the "Health AI: Intelligent Healthcare Assistant" is built using the Gradio framework. Gradio provides a clean, easy-to-navigate, and interactive web application that allows users to interact with the underlying AI model without needing any coding knowledge.

### **Overall Layout:**

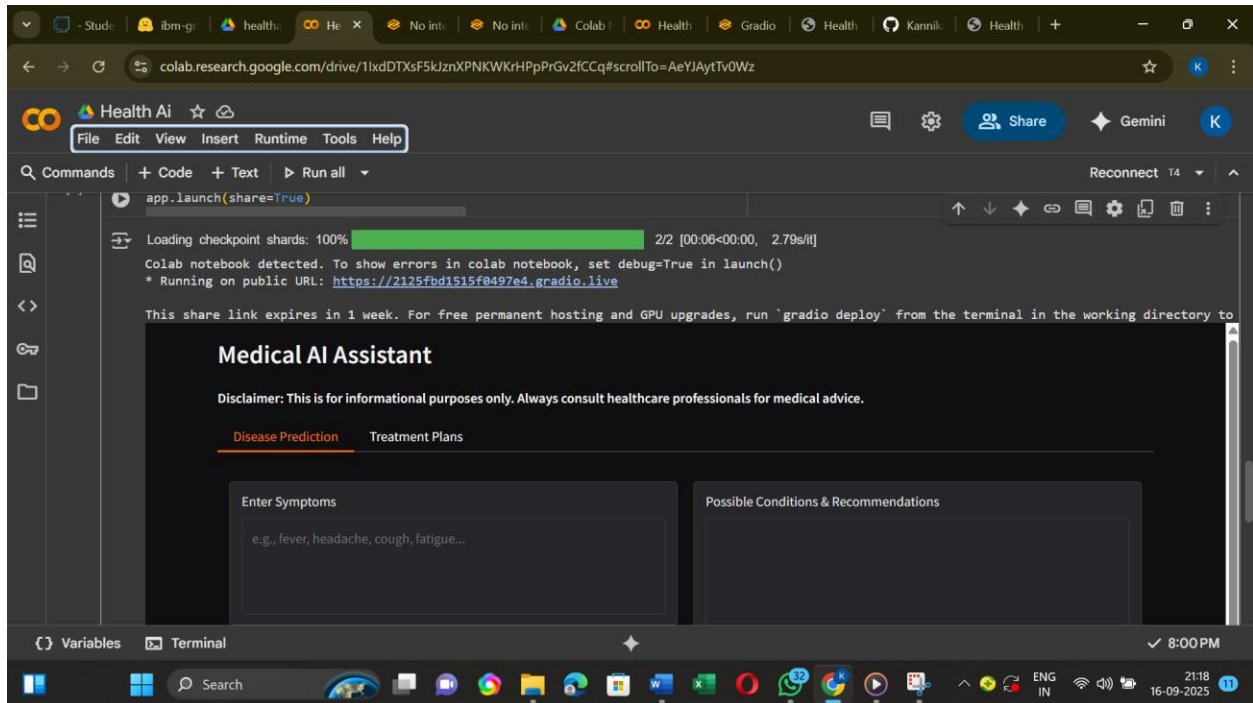
The application is structured using `gr.Blocks()`, which serves as a container for the entire UI. The main page features a prominent title and a crucial medical disclaimer to ensure responsible use.

- **Title:** A large, centered heading reads "# Medical AI Assistant".
- **Disclaimer:** A markdown text below the title clearly states: "Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice." This is a vital component that sets appropriate user expectations from the start.

The core functionality is organized into a tabbed layout using `gr.Tabs()`, which separates the two main features of the application.

### **Tab 1: "Disease Prediction"**

This tab is designed for users who want to understand possible medical conditions based on their symptoms. The layout is simple and intuitive, with an input area and a corresponding output area.

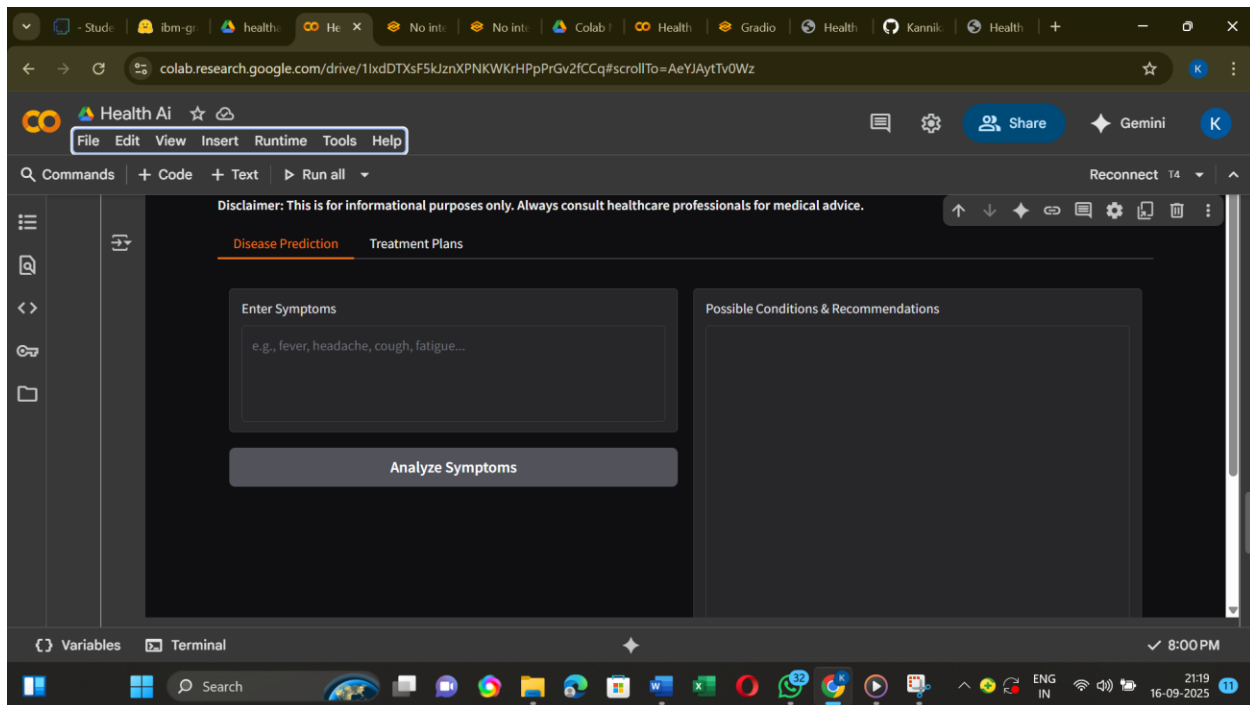


- Input (gr.Textbox):

Label: "Enter Symptoms"

Placeholder: "e.g., fever, headache, cough, fatigue..."

Functionality: This is a multi-line text box where the user can type in a list of their symptoms. The placeholder text provides clear examples to guide the user.



- Button (gr.Button):

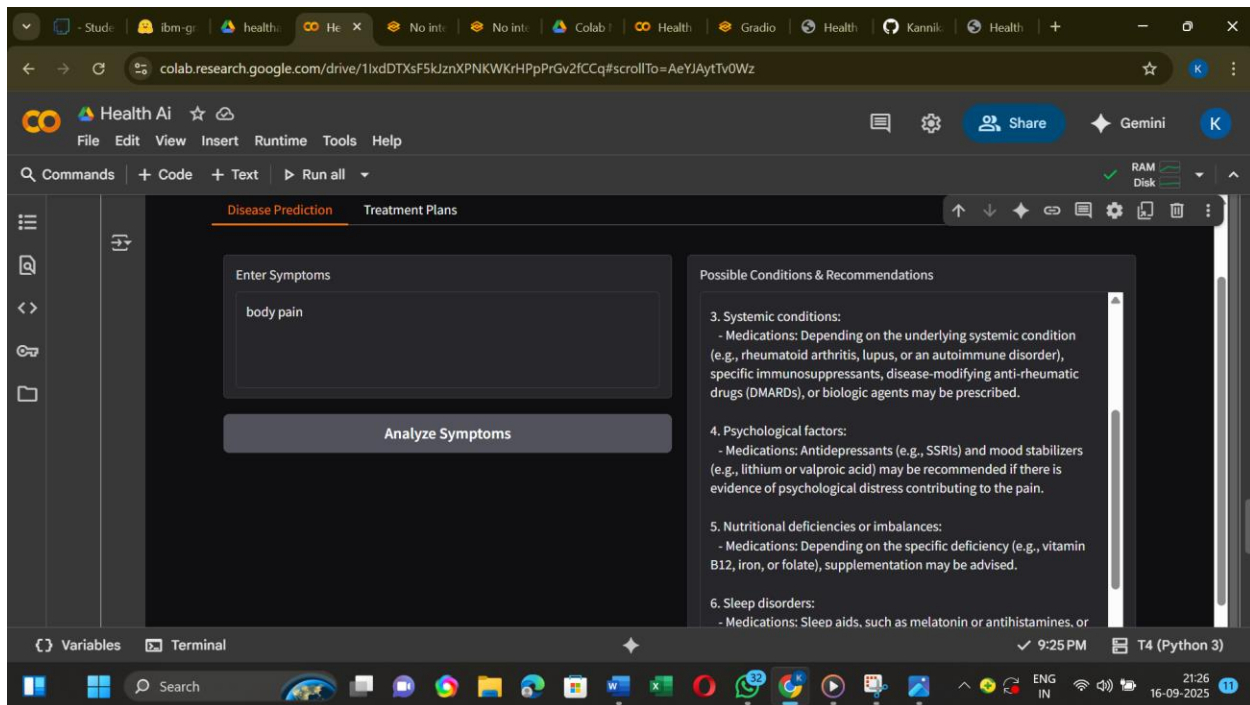
Label: "Analyze Symptoms"

Functionality: Clicking this button sends the text from the symptoms input box to the disease\_prediction function in the backend.

- Output (gr.Textbox):

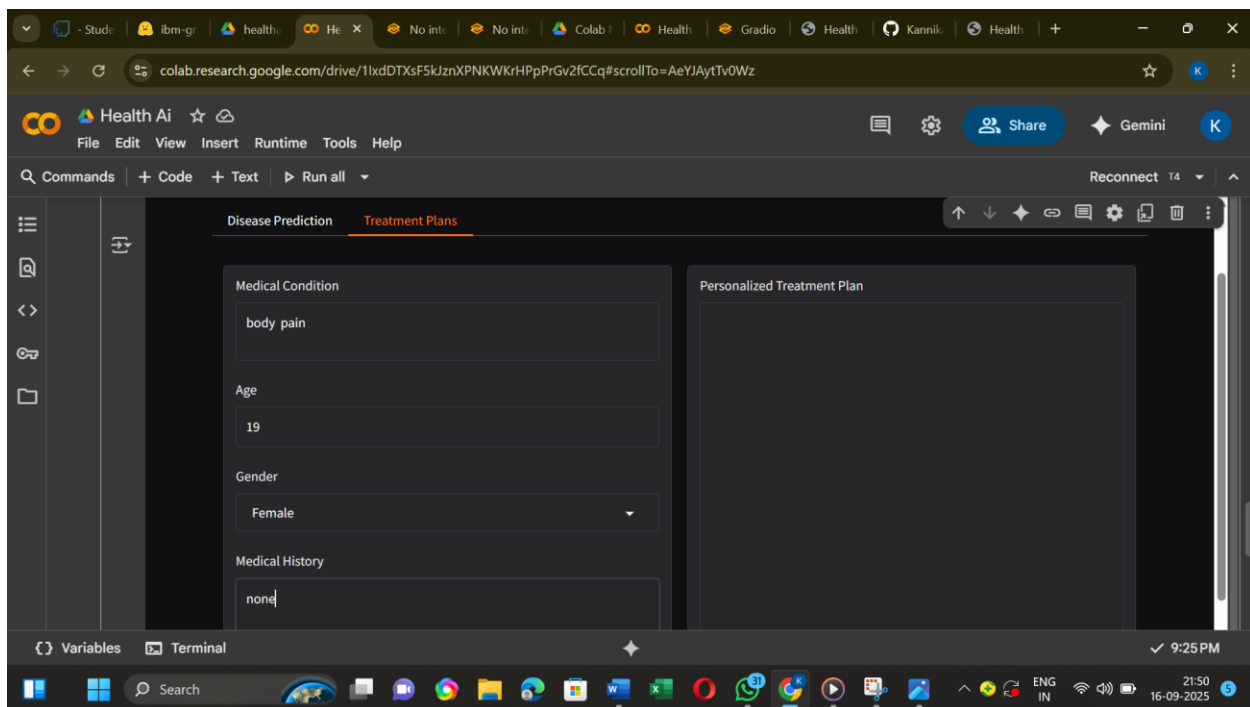
Label: "Possible Conditions & Recommendations"

Functionality: This large, read-only text box displays the response generated by the AI model. The model's output, which includes a list of potential conditions and general recommendations, appears here for the user to read.



## Tab 2: "Treatment Plans"

This tab is for users who have a diagnosed or known condition and want to explore personalized treatment suggestions. It collects more specific information to provide a tailored response.



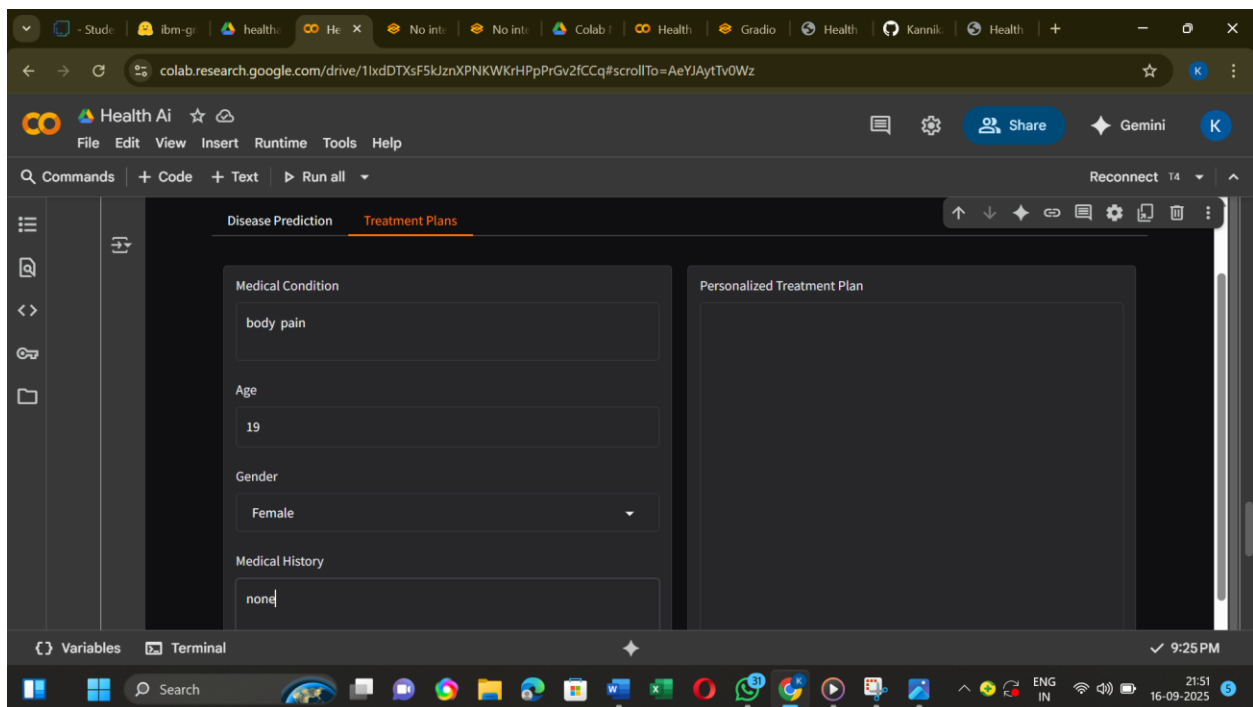
- Input Fields (gr.Textbox, gr.Number, gr.Dropdown):

*Medical Condition (gr.Textbox):*

Label: "Medical Condition"

Placeholder: "e.g., diabetes, hypertension, migraine..."

Functionality: A text box for the user to enter their specific medical condition.



*Age (gr.Number):*

Label: "Age"

Functionality: A numerical input field for the user's age, which is a key factor in many medical treatments.

A screenshot of a Google Colab notebook interface. The browser's address bar shows a Google Drive link. The notebook contains a form with the following fields:

- body pain**: A text input field containing the text "body pain".
- Age**: A text input field containing the number "19".
- Gender**: A dropdown menu currently displaying "Female".
- Medical History**: A text input field containing the text "none".

The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 21:51 on 16-09-2025.

*Gender (gr.Dropdown):*

Label: "Gender"

Choices: "Male", "Female", "Other"

Functionality: A dropdown menu to select gender.



The screenshot shows a web browser window with a Google Drive link in the address bar. The page displays a form with the following fields:

- body pain**: A text input field.
- Age**: A text input field containing the value "19".
- Gender**: A dropdown menu with "Female" selected.
- Medical History**: A multi-line text input field containing the value "none".

The Windows taskbar is visible at the bottom, showing the search bar and various application icons. The system clock indicates the time is 21:54 on 16-09-2025.

*Medical History (gr.Textbox):*

Label: "Medical History"

Placeholder: "Previous conditions, allergies, medications or None"

Functionality: A multi-line text box for the user to provide details about their health background.

A screenshot of a web browser displaying a form for generating a treatment plan. The form is titled "body pain" and contains the following fields:

- Age: 19
- Gender: Female (dropdown menu)
- Medical History: none

The browser's address bar shows the URL: [colab.research.google.com/drive/1lxdDTXsF5kJznXPnKWkrHPpPrGv2fCCq#scrollTo=AeYJAytTv0Wz](https://colab.research.google.com/drive/1lxdDTXsF5kJznXPnKWkrHPpPrGv2fCCq#scrollTo=AeYJAytTv0Wz). The Windows taskbar at the bottom shows the time as 21:54 on 16-09-2025.

*Button (gr.Button):*

Label: "Generate Treatment Plan"

Functionality: This button triggers the `treatment_plan` function in the backend, passing all the entered information to the AI model.

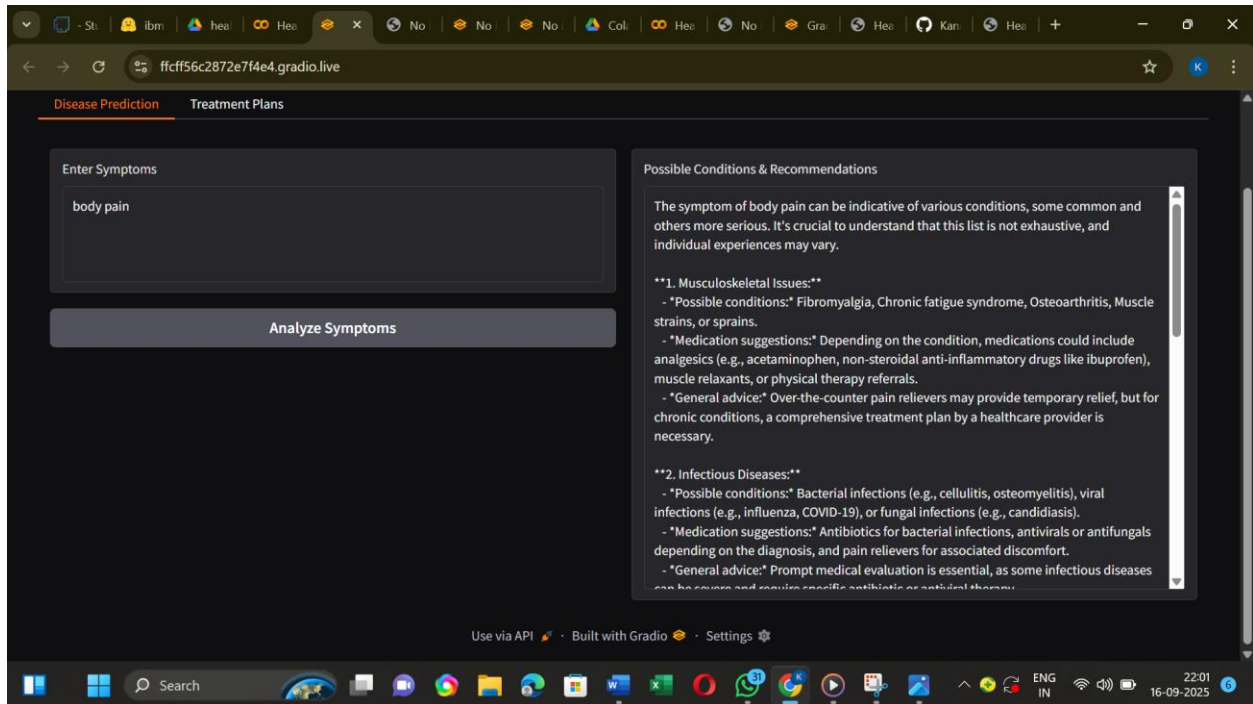
A screenshot of the same web browser displaying the form, but with the "Generate Treatment Plan" button highlighted. The button is a large, dark gray rectangle with the text "Generate Treatment Plan" in white. The form fields are still visible above the button.

The browser's address bar shows the same URL: [colab.research.google.com/drive/1lxdDTXsF5kJznXPnKWkrHPpPrGv2fCCq#scrollTo=AeYJAytTv0Wz](https://colab.research.google.com/drive/1lxdDTXsF5kJznXPnKWkrHPpPrGv2fCCq#scrollTo=AeYJAytTv0Wz). The Windows taskbar at the bottom shows the time as 21:56 on 16-09-2025.

*Output (gr.Textbox):*

Label: "Personalized Treatment Plan"

Functionality: A large, scrollable text box that displays the AI-generated treatment plan, including home remedies and medication guidelines.



## 8.User Flow:

1. The user lands on the application page.
2. They can choose between the "Disease Prediction" and "Treatment Plans" tabs.
3. In the chosen tab, they fill out the input fields with their specific information.
4. They click the "Analyze Symptoms" or "Generate Treatment Plan" button.
5. The Gradio interface sends this information to the Python backend, where the IBM Granite model processes the request.
6. The AI's response is returned and displayed in the designated output textbox.

7. The user can then modify their inputs and generate new responses as needed.