HEALTH AI : Intelligent Healthcare Assistant Using IBM Granite

PROJECT DOCUMENTATION

1.Introduction

Project title : **Health AI**

Team leader : **Gayathri.B**

Team member : **Abitha.K**

Team member : **Abitha.S**

Team member :**Ayshwariyha.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.Setup Instructions Prerequisites:

- Python 3.x
- Gradio framework
- Transformers & Torch libraries
- Google Colab with T4 GPU access
- GitHub account **Installation Process:**

1. Open Google Colab and create a new notebook.
2. Change runtime type → select **T4 GPU**.

3. Install dependencies:
4. !pip install transformers torch gradio -q
5. Load and run the Citizen AI code.
6. Access the Gradio app link to interact with the model.
7. Download project files and upload them to GitHub for version control. **5. Folder Structure** citizen_ai

```
|— app.py             # Main Gradio application
|— model_config.py    # IBM Granite model setup
|— dashboard.py       # Dashboard logic
|— requirements.txt   # Required dependencies
└— README.md          # Project description
```

6.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.

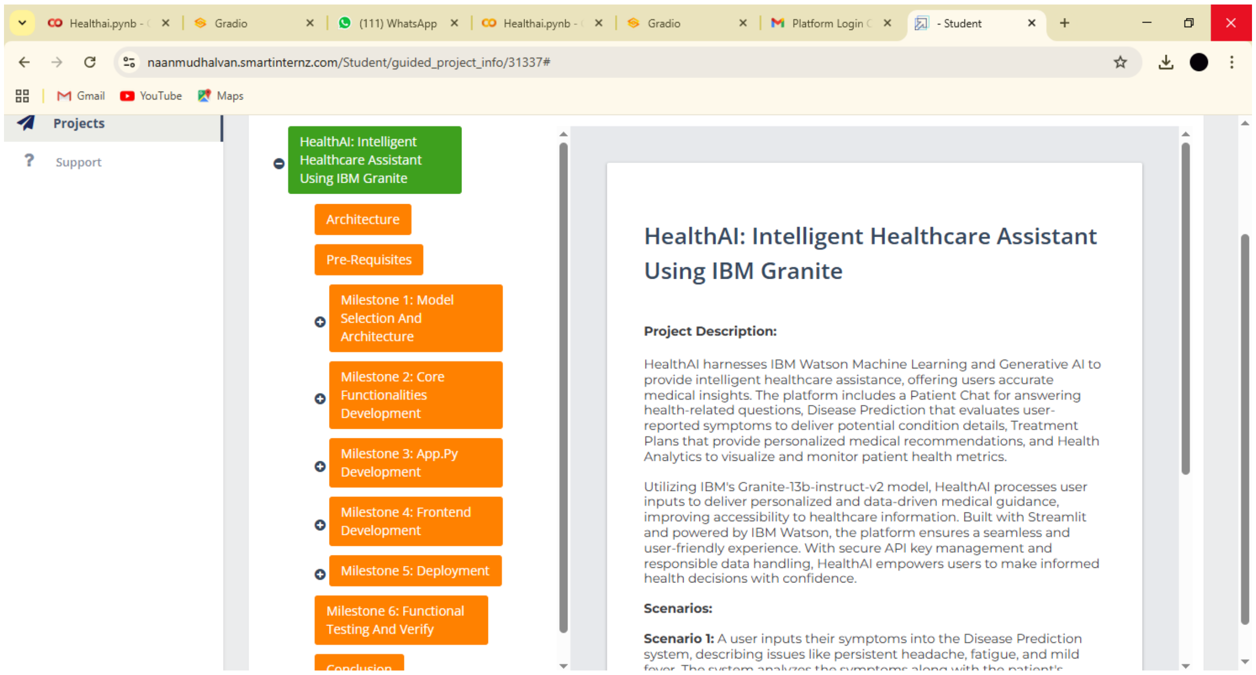
7.Project Workflow

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.


Healthai.pynb
Gradio
(111) WhatsApp
Healthai.pynb
Gradio
Platform Login
- Student
naanmudhalvan.smartinternz.com/Student/guided_project_info/31337#
Gmail
YouTube
Maps
Projects
Support
HealthAI: Intelligent Healthcare Assistant Using IBM Granite
Architecture
Pre-Requisites
Milestone 1: Model Selection And Architecture
Milestone 2: Core Functionalities Development
Milestone 3: App.Py Development
Milestone 4: Frontend Development
Milestone 5: Deployment
Milestone 6: Functional Testing And Verify
Conclusion

HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Project Description:

HealthAI harnesses IBM Watson Machine Learning and Generative AI to provide intelligent healthcare assistance, offering users accurate medical insights. The platform includes a Patient Chat for answering health-related questions, Disease Prediction that evaluates user-reported symptoms to deliver potential condition details, Treatment Plans that provide personalized medical recommendations, and Health Analytics to visualize and monitor patient health metrics.

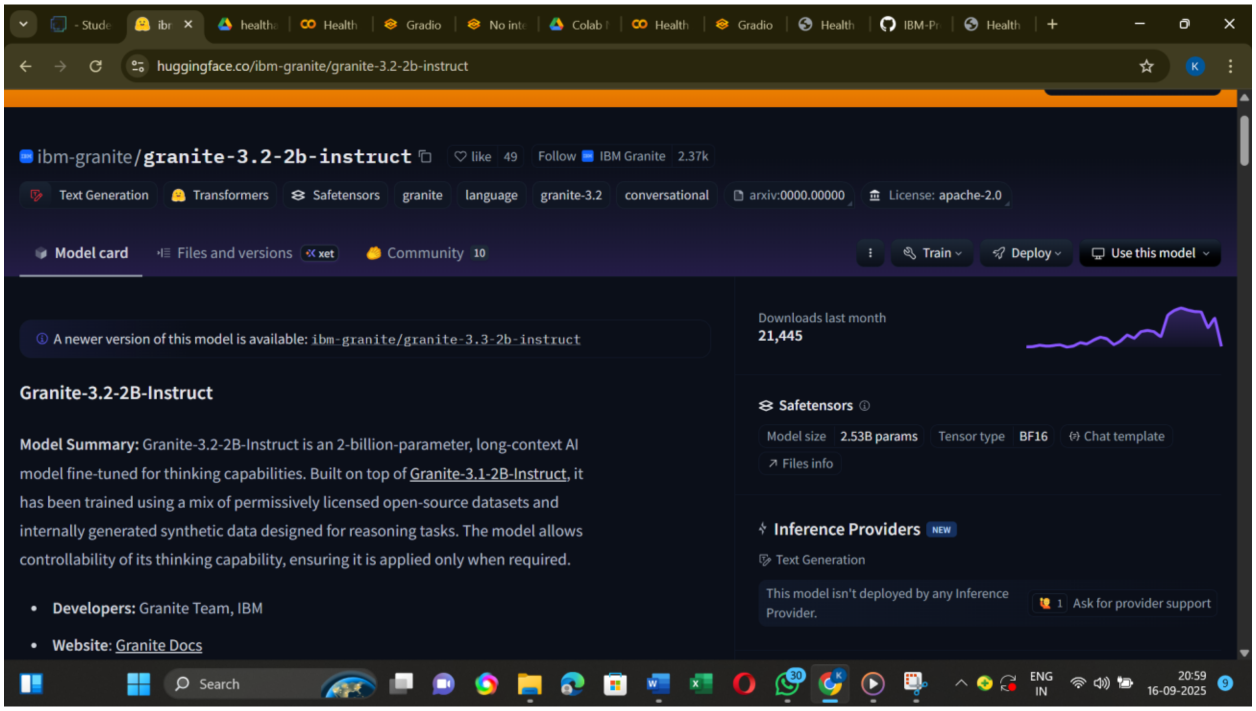
Utilizing IBM's Granite-13b-instruct-v2 model, HealthAI processes user inputs to deliver personalized and data-driven medical guidance, improving accessibility to healthcare information. Built with Streamlit and powered by IBM Watson, the platform ensures a seamless and user-friendly experience. With secure API key management and responsible data handling, HealthAI empowers users to make informed health decisions with confidence.

Scenarios:

Scenario 1: A user inputs their symptoms into the Disease Prediction system, describing issues like persistent headache, fatigue, and mild

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.


huggingface.co/ibm-granite/granite-3.2-2b-instruct

ibm-granite/**granite-3.2-2b-instruct** like 49 Follow IBM Granite 2.37k

Text Generation · Transformers · Safetensors · granite · language · granite-3.2 · conversational · arxiv:0000.00000 · License: apache-2.0

Model card · Files and versions · Community 10 · Train · Deploy · Use this model

A newer version of this model is available: `ibm-granite/granite-3.3-2b-instruct`

Granite-3.2-2B-Instruct

Model Summary: Granite-3.2-2B-Instruct is an 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. Built on top of Granite-3.1-2B-Instruct, it has been trained using a mix of permissively licensed open-source datasets and internally generated synthetic data designed for reasoning tasks. The model allows controllability of its thinking capability, ensuring it is applied only when required.

- **Developers:** Granite Team, IBM
- **Website:** Granite Docs

Downloads last month 21,445

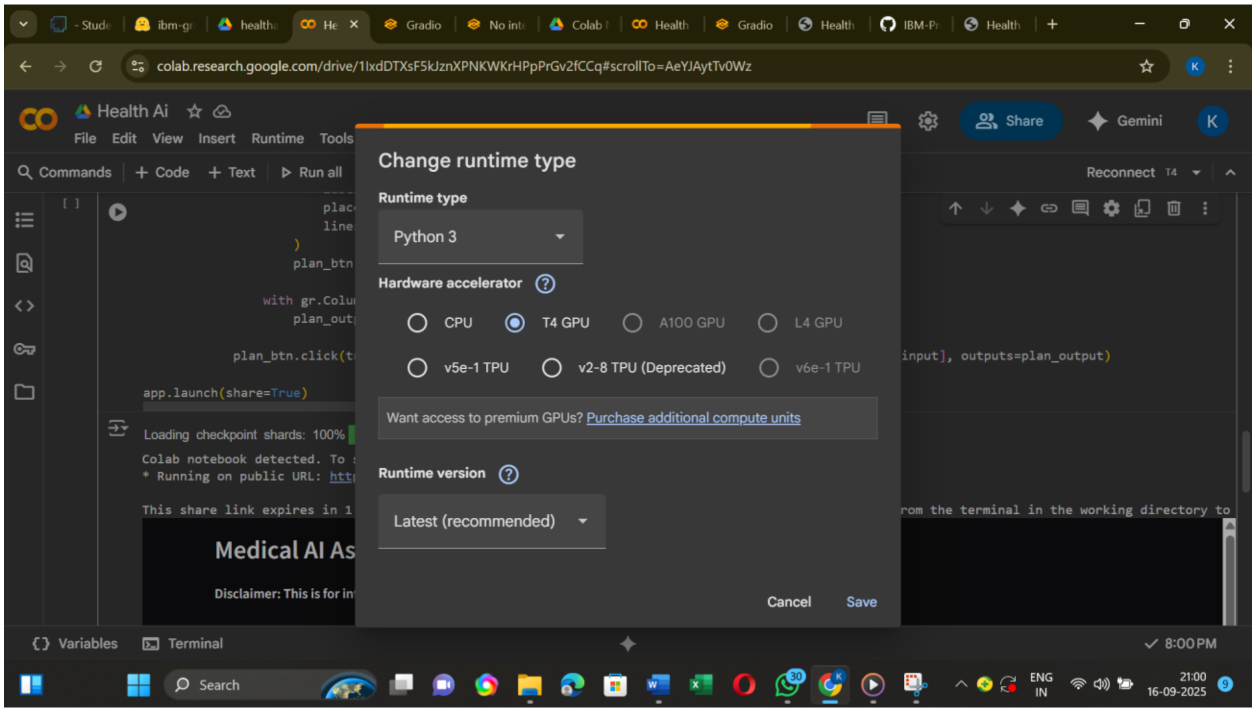
Safetensors — Model size 2.53B params · Tensor type BF16 · Chat template · Files info

Inference Providers NEW — Text Generation

This model isn't deployed by any Inference Provider. Ask for provider support

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.


colab.research.google.com/drive/1IxdDTXsF5kJznXPNKWKrHPpPrGv2fCCq#scrollTo=AeYJAytTv0Wz

Health Ai

```
                plan_btn
            with gr.Colu
                plan_out
        plan_btn.click(t                                    input], outputs=plan_output)

app.launch(share=True)
```

```
Loading checkpoint shards: 100%
Colab notebook detected. To
* Running on public URL: htt
This share link expires in 1                        rom the terminal in the working directory to
```

Medical AI As

Disclaimer: This is for in

Change runtime type

Runtime type: Python 3

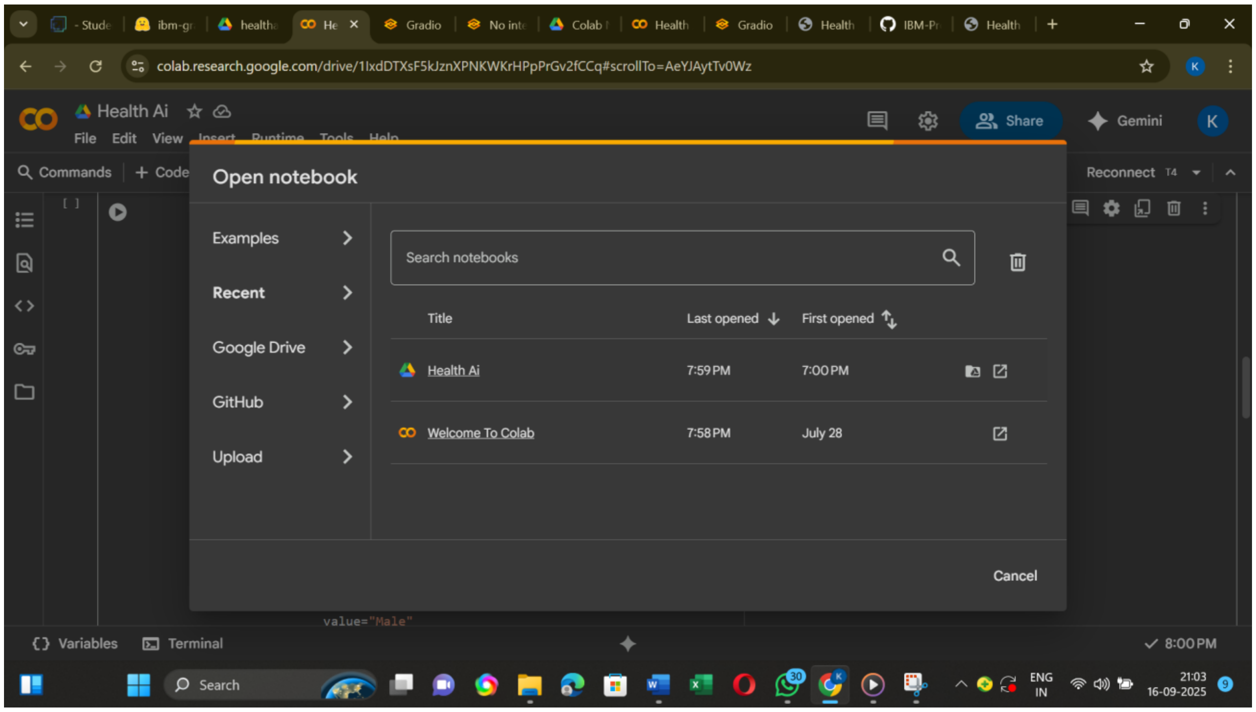
Hardware accelerator: ( ) CPU [✓ T4 GPU] ( ) A100 GPU ( ) L4 GPU ( ) v5e-1 TPU ( ) v2-8 TPU (Deprecated) ( ) v6e-1 TPU

Want access to premium GPUs? Purchase additional compute units

Runtime version: Latest (recommended)

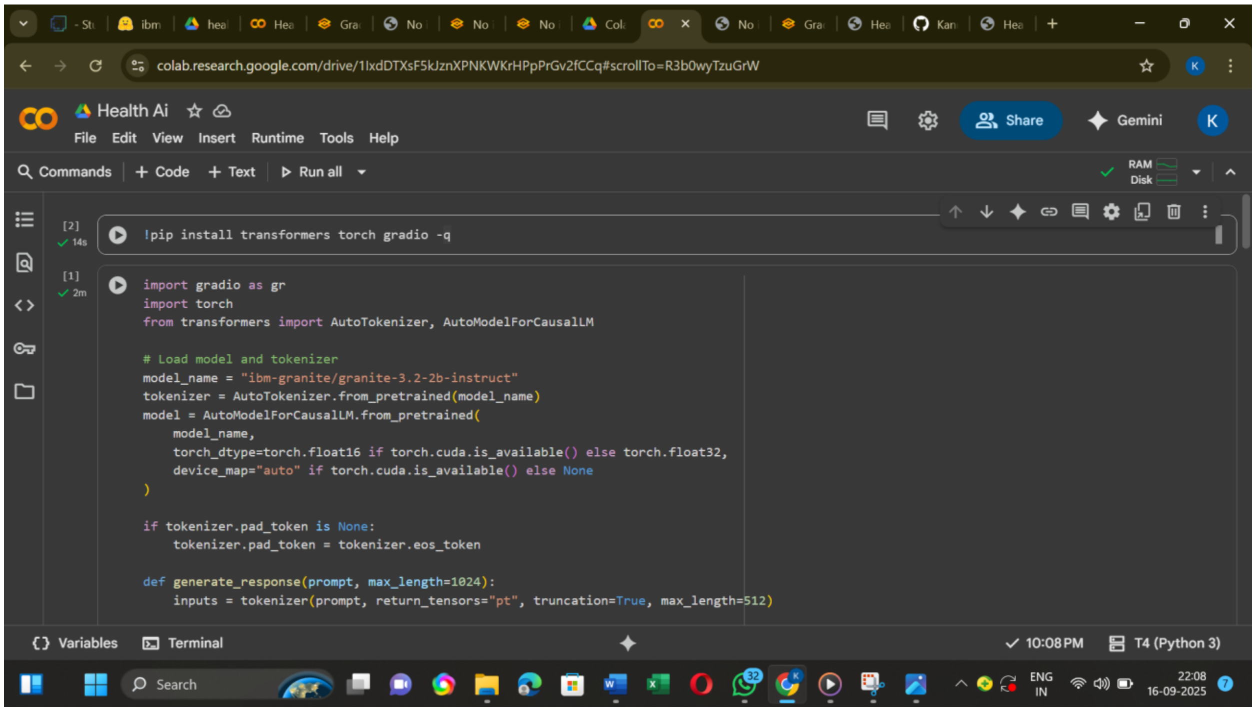
Cancel · Save

- Go to file, click Open notebook.



Title	Last opened	First opened
Health Ai	7:59 PM	7:00 PM
Welcome To Colab	7:58 PM	July 28

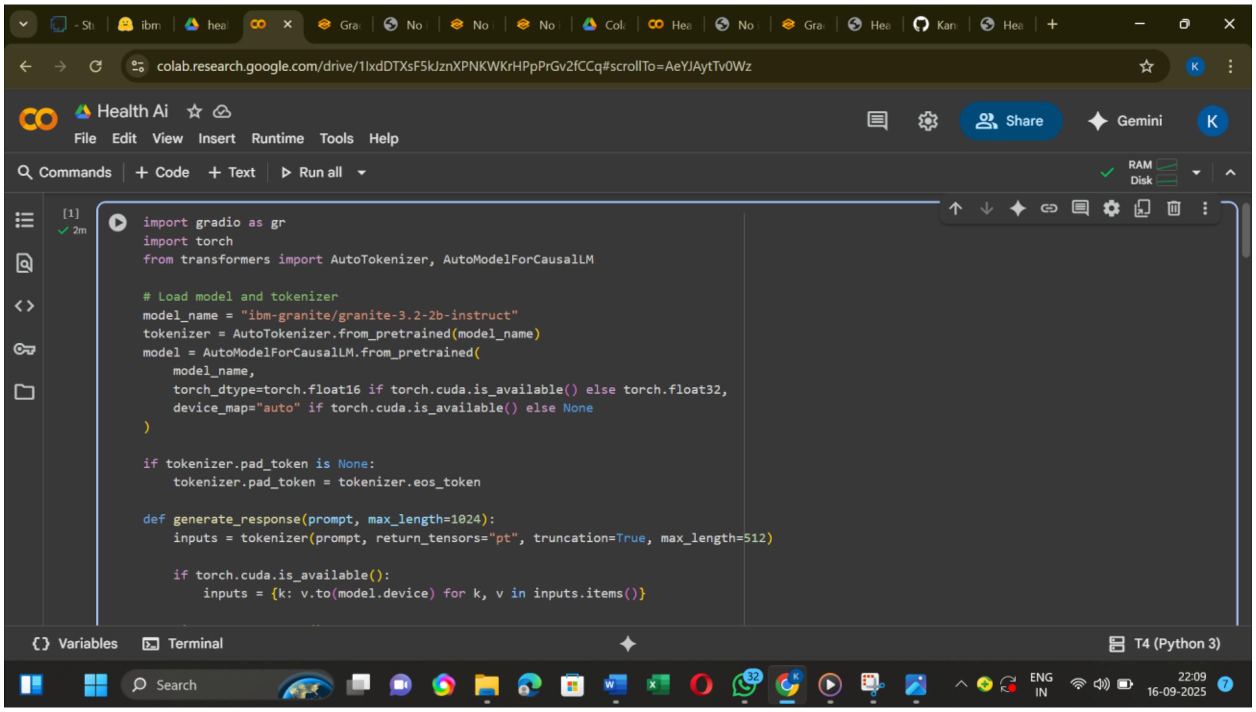
- Required libraries like transformers, torch, and gradio were installed.


Health Ai
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Variables Terminal
T4 (Python 3)

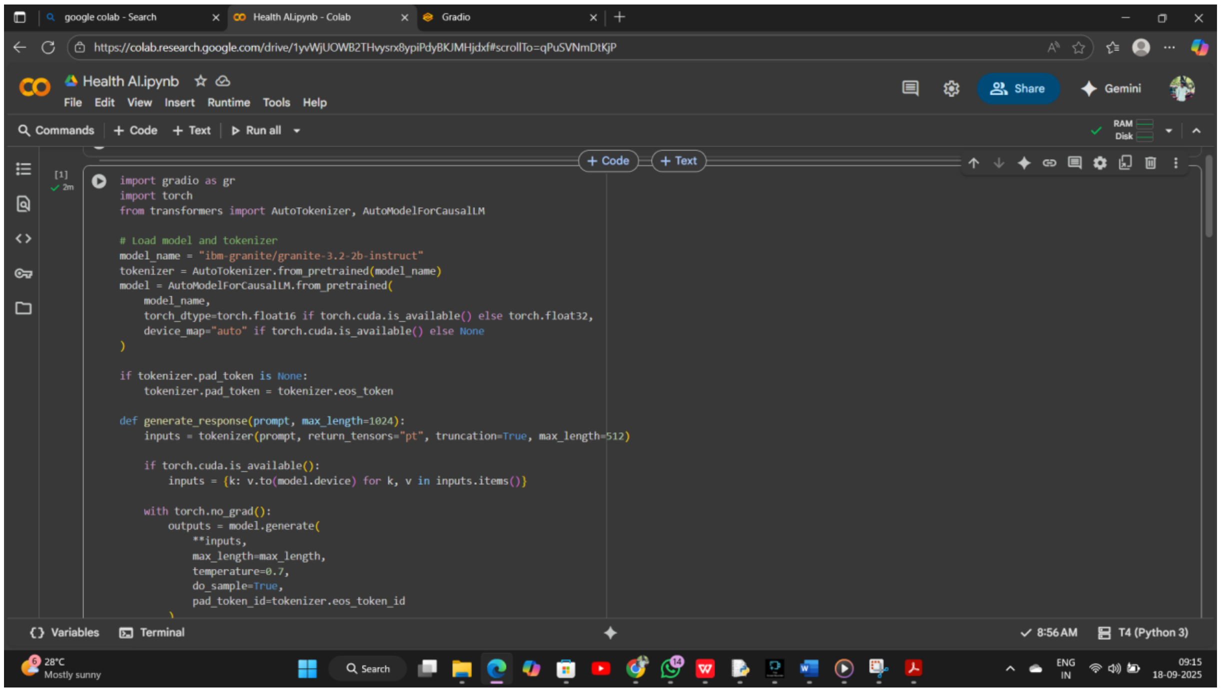
```
!pip install transformers torch gradio -q
```

```
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)
```

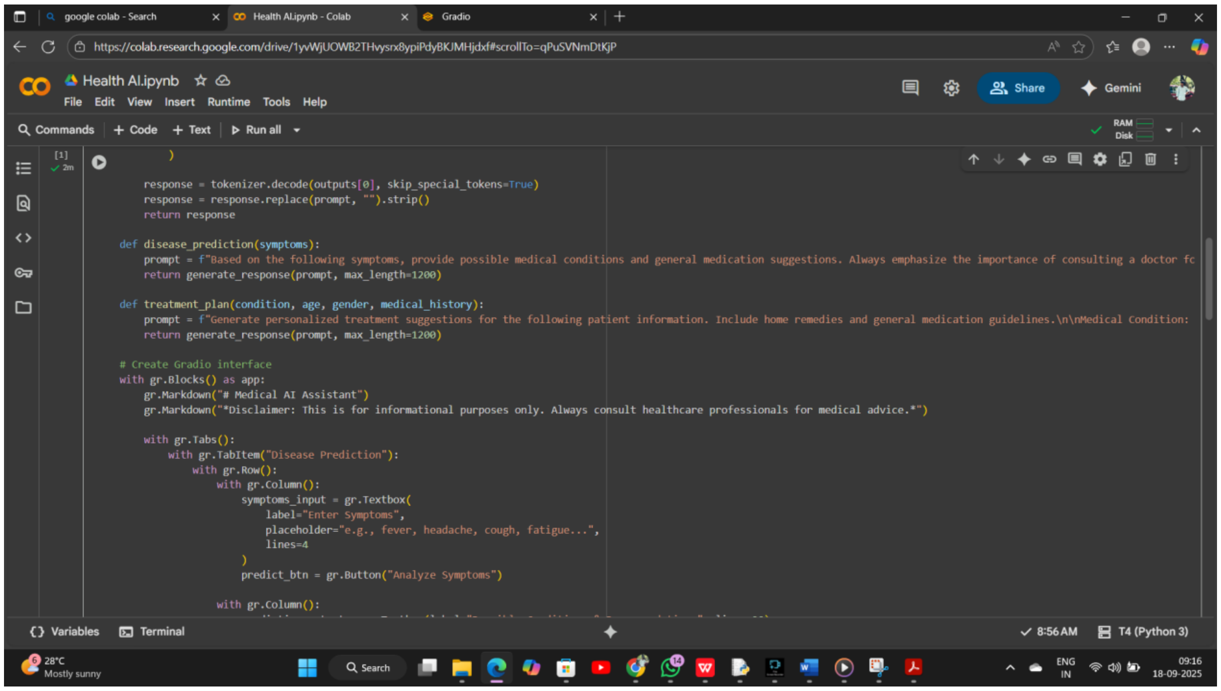
- The Python code, which handles loading the model, defining the generation functions, and creating the Gradio interface, was added to the notebook and executed.



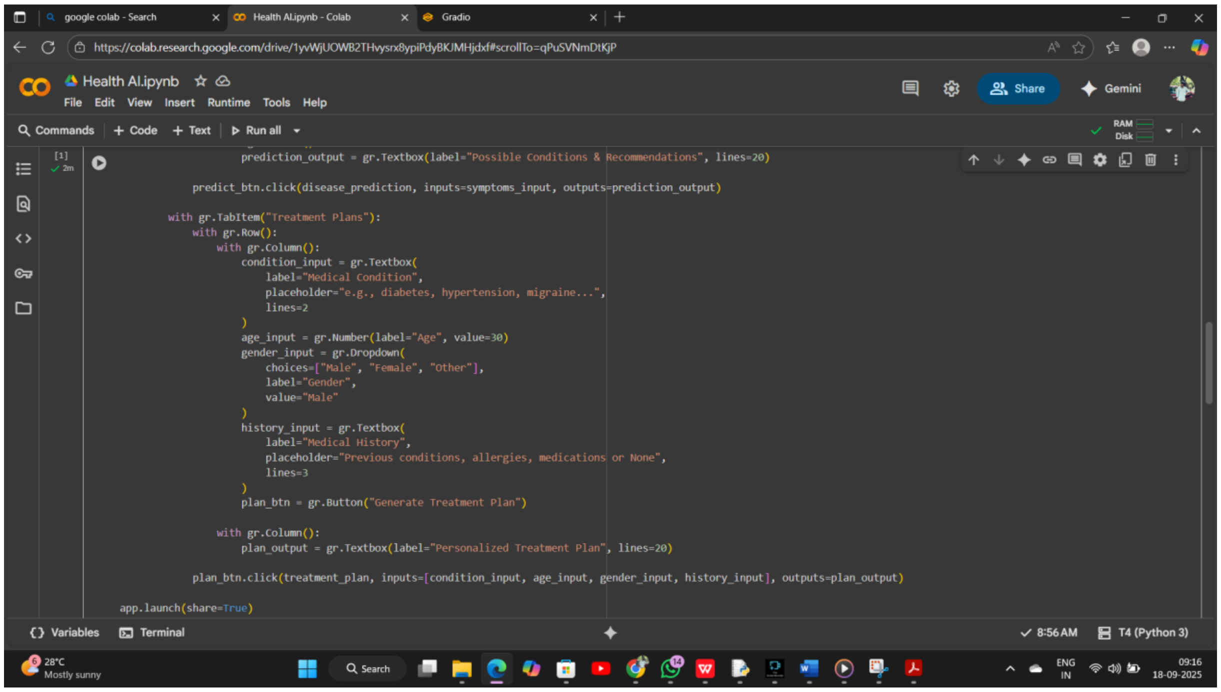
```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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()}
```



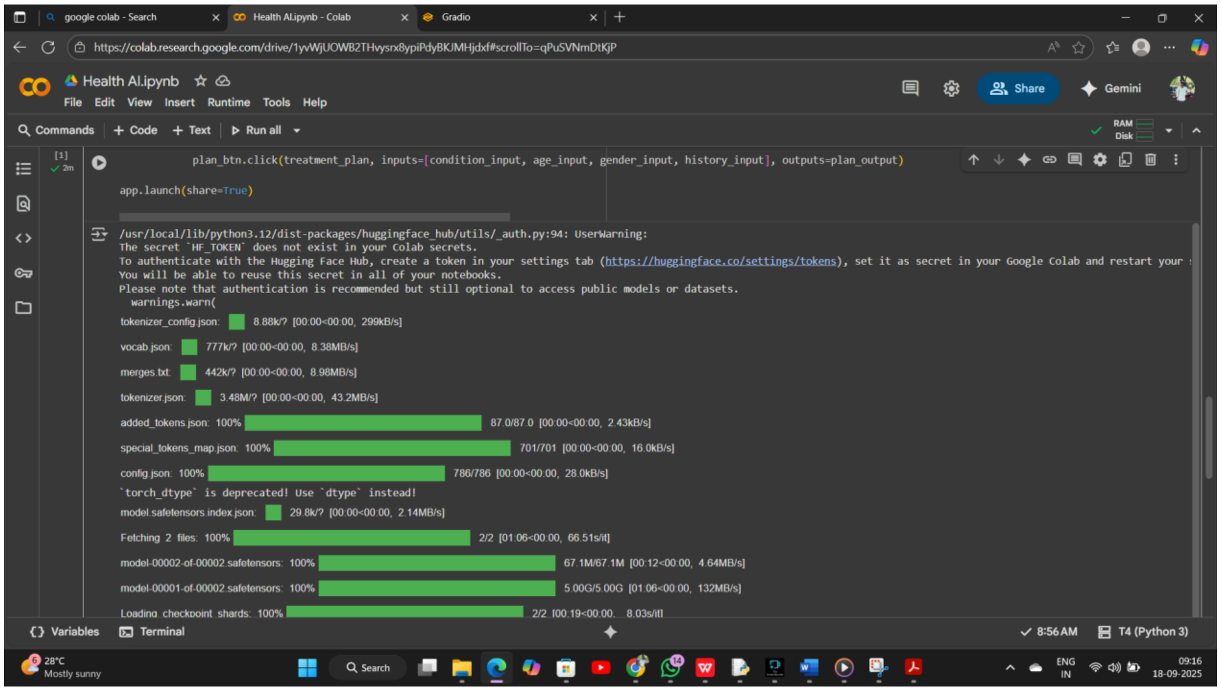
```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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 fc
        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:
        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():
```

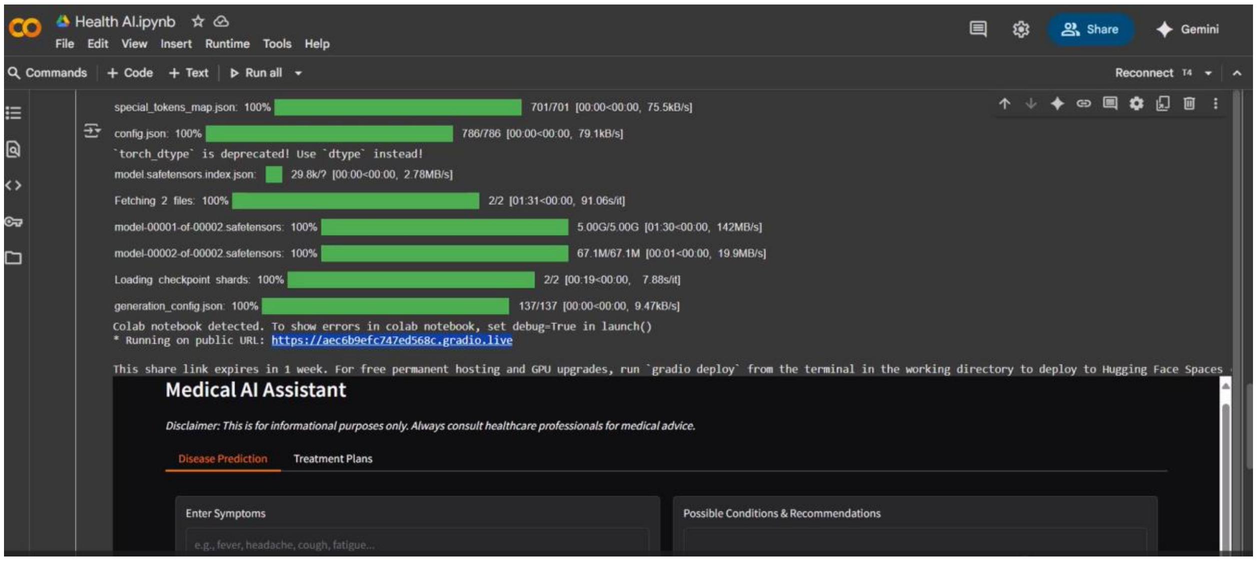


```
                        prediction_output = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)

                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)
```


google colab - Search
Health AI.ipynb - Colab
Gradio
https://colab.research.google.com/drive/1yvWjUOWB2THvysrx8ypiPdyBKJMHjdxf#scrollTo=qPuSVNmDtKjP
Health AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
Variables
Terminal
8:56 AM
T4 (Python 3)
28°C
Mostly sunny
Search
ENG
IN
09:16
18-09-2025

```
        plan_btn.click(treatment_plan, inputs=[condition_input, age_input, gender_input, history_input], outputs=plan_output)

app.launch(share=True)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your :
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 🟩 8.88k/? [00:00<00:00, 299kB/s]
vocab.json: 🟩 777k/? [00:00<00:00, 8.38MB/s]
merges.txt: 🟩 442k/? [00:00<00:00, 8.98MB/s]
tokenizer.json: 🟩 3.48M/? [00:00<00:00, 43.2MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.43kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 16.0kB/s]
config.json: 100% 786/786 [00:00<00:00, 28.0kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 🟩 29.8k/? [00:00<00:00, 2.14MB/s]
Fetching 2 files: 100% 2/2 [01:06<00:00, 66.51s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:12<00:00, 4.64MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:06<00:00, 132MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 8.03s/it]
```


Health AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
Reconnect T4

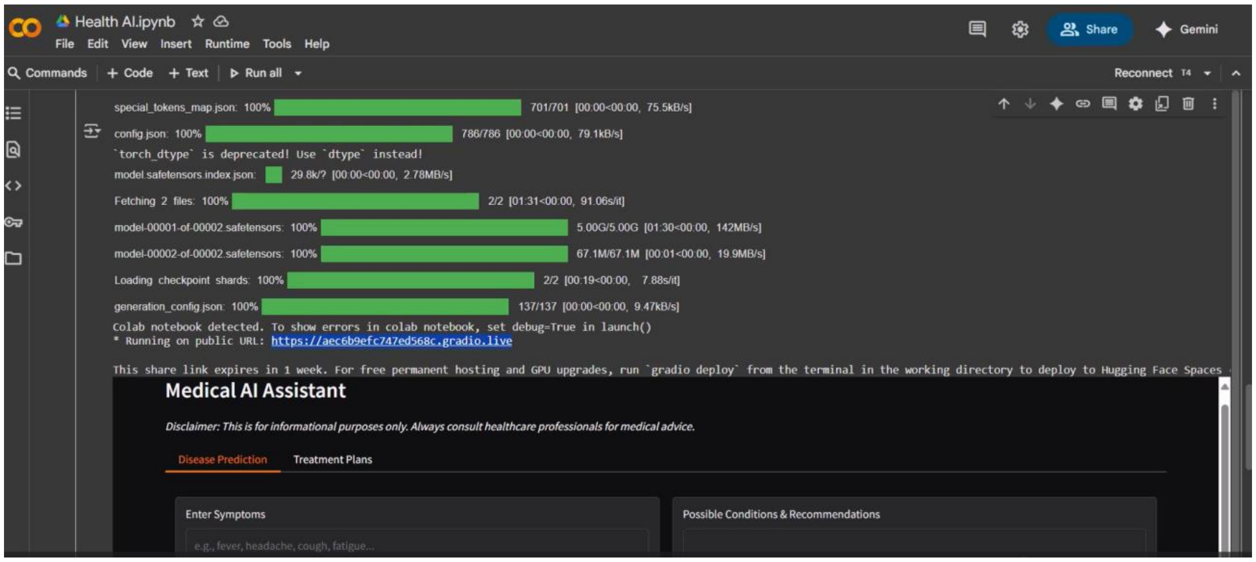
```
special_tokens_map.json: 100% 701/701 [00:00<00:00, 75.5kB/s]
config.json: 100% 786/786 [00:00<00:00, 79.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 🟩 29.8k/? [00:00<00:00, 2.78MB/s]
Fetching 2 files: 100% 2/2 [01:31<00:00, 91.06s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:30<00:00, 142MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 19.9MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 7.88s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 9.47kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://aec6b9efc747ed568c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```

Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

Disease Prediction
Treatment Plans
Enter Symptoms
e.g., fever, headache, cough, fatigue...
Possible Conditions & Recommendations

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

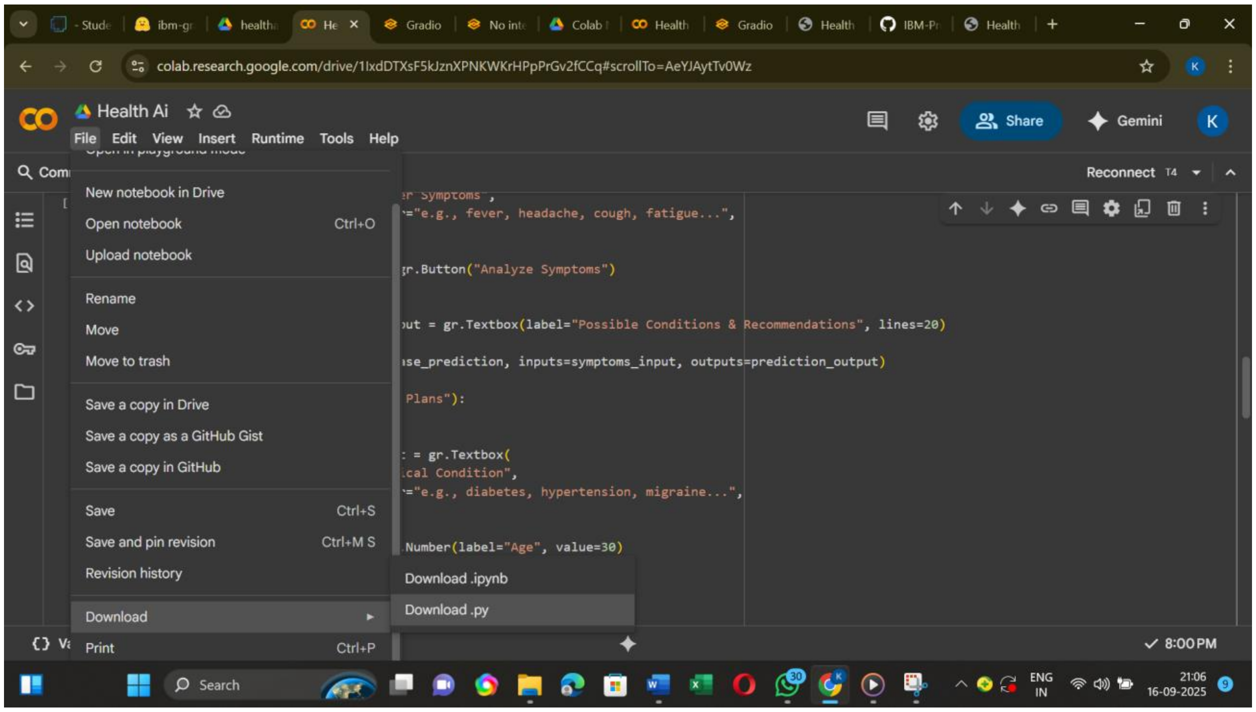


```
special_tokens_map.json: 100% 701/701 [00:00<00:00, 75.5kB/s]
config.json: 100% 786/786 [00:00<00:00, 79.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.78MB/s]
Fetching 2 files: 100% 2/2 [01:31<00:00, 91.06s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:30<00:00, 142MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 19.9MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00,  7.88s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 9.47kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://aec6b9efc747ed568c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```

Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

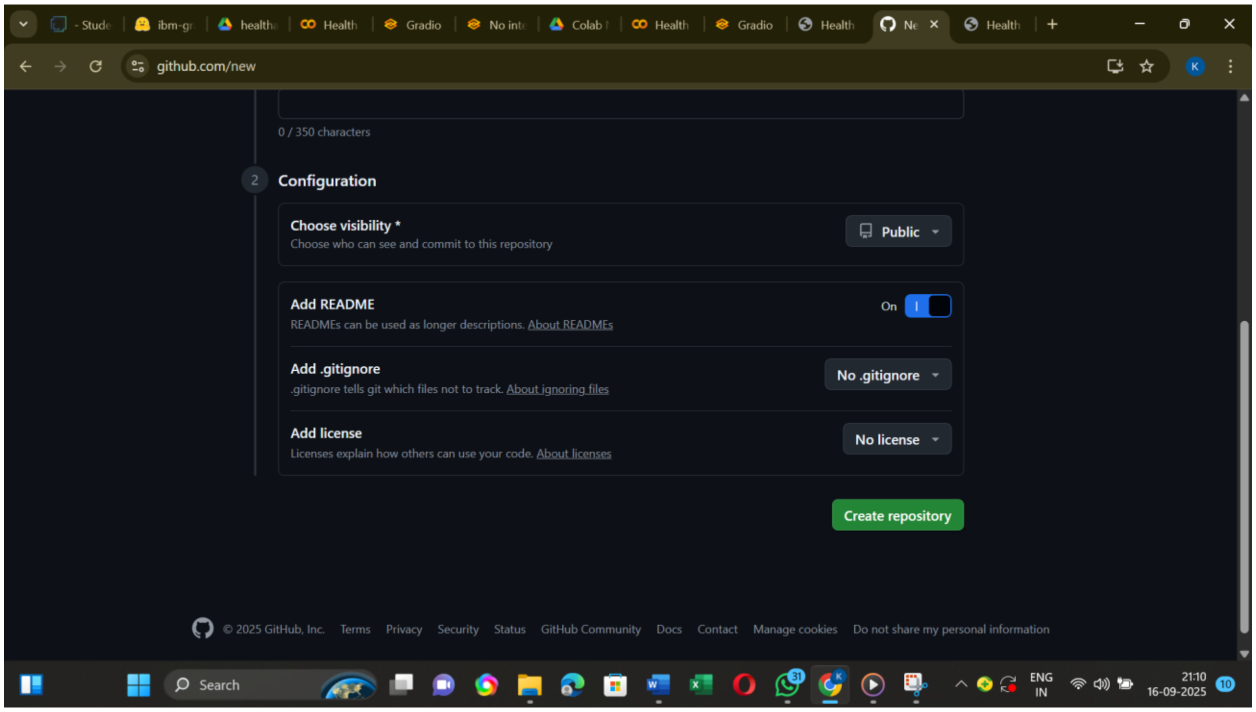
- Go to file and download the project as Download.py.



```
er Symptoms",
="e.g., fever, headache, cough, fatigue...",

gr.Button("Analyze Symptoms")

out = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)

se_prediction, inputs=symptoms_input, outputs=prediction_output)

 Plans"):

: = gr.Textbox(
cal Condition",
="e.g., diabetes, hypertension, migraine...",

Number(label="Age", value=30)
```

8.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.



0 / 350 characters

2 Configuration

Choose visibility *
Choose who can see and commit to this repository
Public

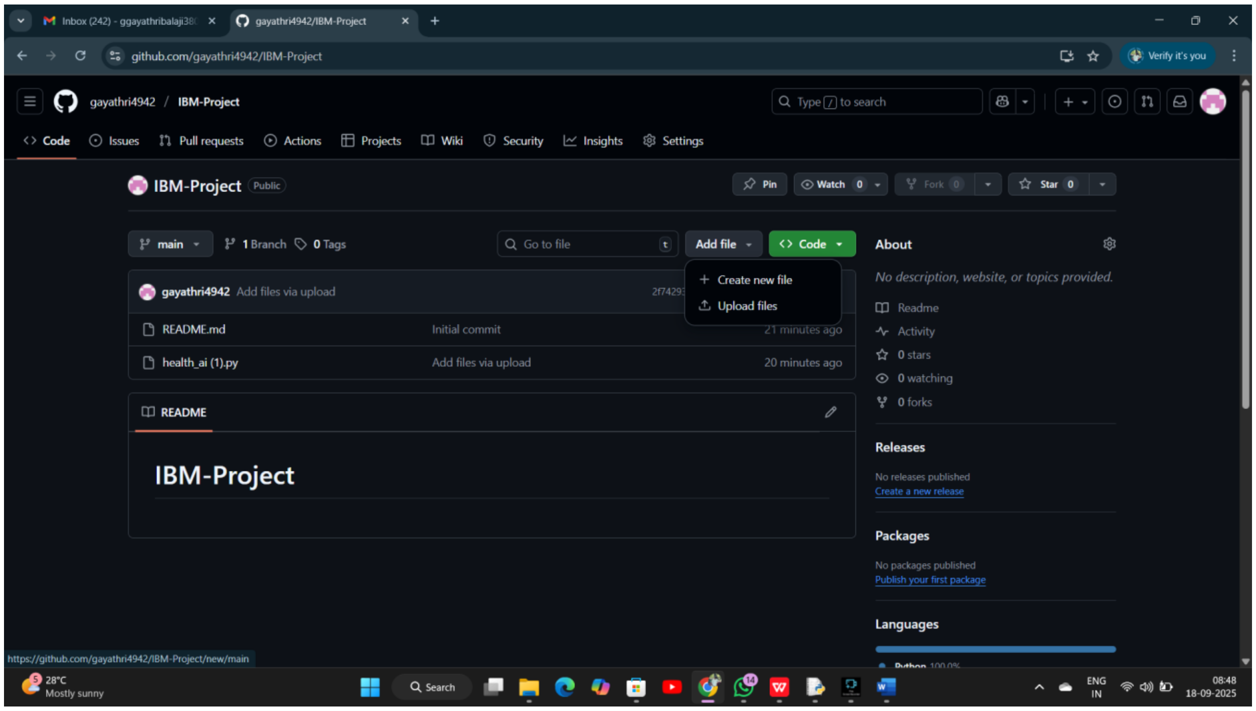
Add README
READMEs can be used as longer descriptions. About READMEs
On

Add .gitignore
.gitignore tells git which files not to track. About ignoring files
No .gitignore

Add license
Licenses explain how others can use your code. About licenses
No license

github.com/new
Create repository
© 2025 GitHub, Inc. Terms Privacy Security Status GitHub Community Docs Contact Manage cookies Do not share my personal information
Search
21:10
16-09-2025

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



gayathri4942	Add files via upload	
README.md	Initial commit	21 minutes ago
health_ai (1).py	Add files via upload	20 minutes ago

9.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:

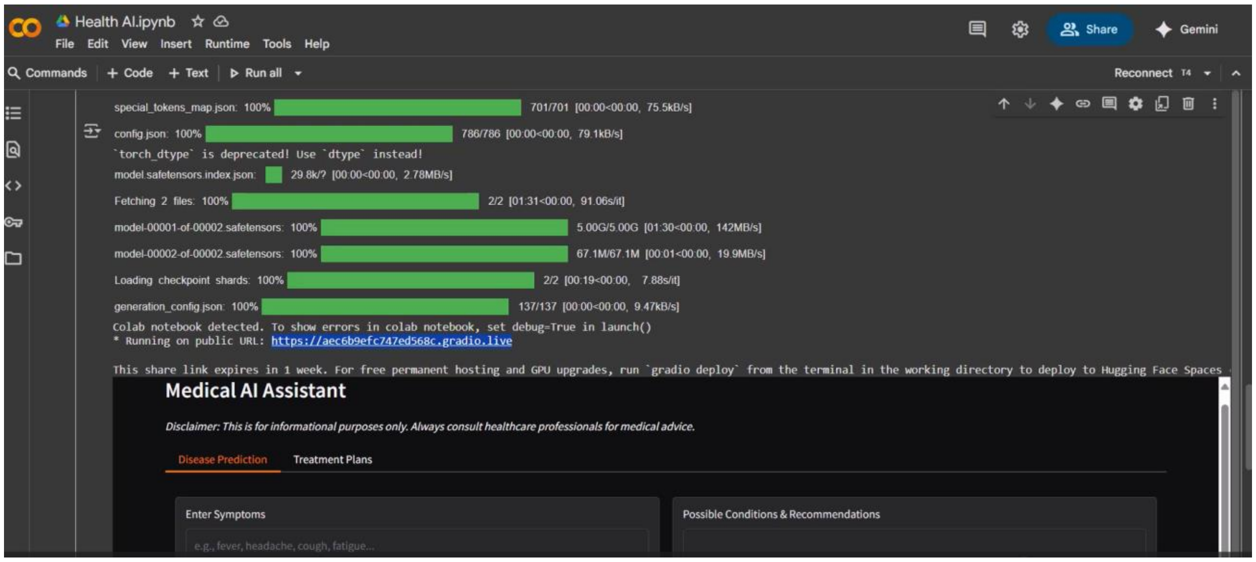
The application is structured using gr.Blocks(), which serves as a container for the entire UI. The main page features a prominent title and a crucial medical disclaimer to ensure responsible use.

- **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.



```
special_tokens_map.json: 100% 701/701 [00:00<00:00, 75.5kB/s]
config.json: 100% 786/786 [00:00<00:00, 79.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.78MB/s]
Fetching 2 files: 100% 2/2 [01:31<00:00, 91.06s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:30<00:00, 142MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 19.9MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 7.88s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 9.47kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://aec6b9efc747ed568c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```

Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

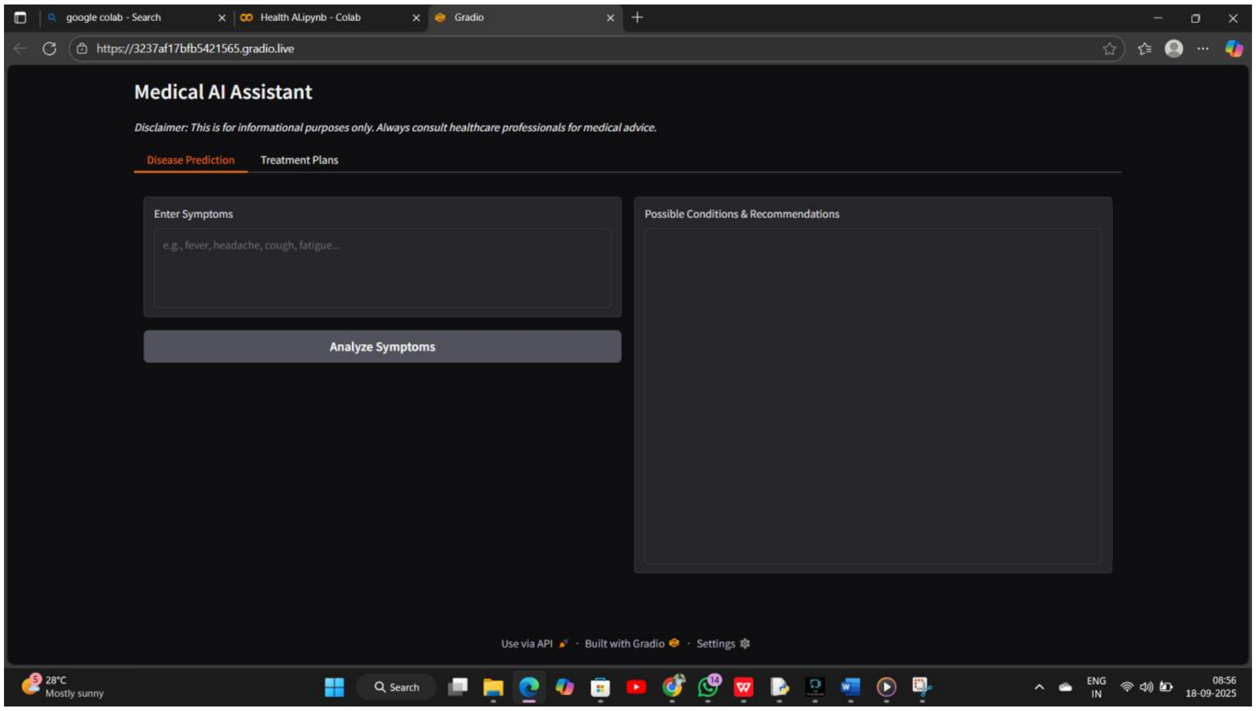
Disease Prediction
Treatment Plans
Enter Symptoms
e.g., fever, headache, cough, fatigue...
Possible Conditions & Recommendations

- 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.


google colab - Search
Health AI.ipynb - Colab
Gradio
https://3237af17bfb5421565.gradio.live
Medical AI Assistant
Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.
Disease Prediction
Treatment Plans
Enter Symptoms
e.g., fever, headache, cough, fatigue...
Analyze Symptoms
Possible Conditions & Recommendations
Use via API · Built with Gradio · Settings
28°C Mostly sunny
Search
ENG IN
08:56 18-09-2025

- Button (gr.Button):

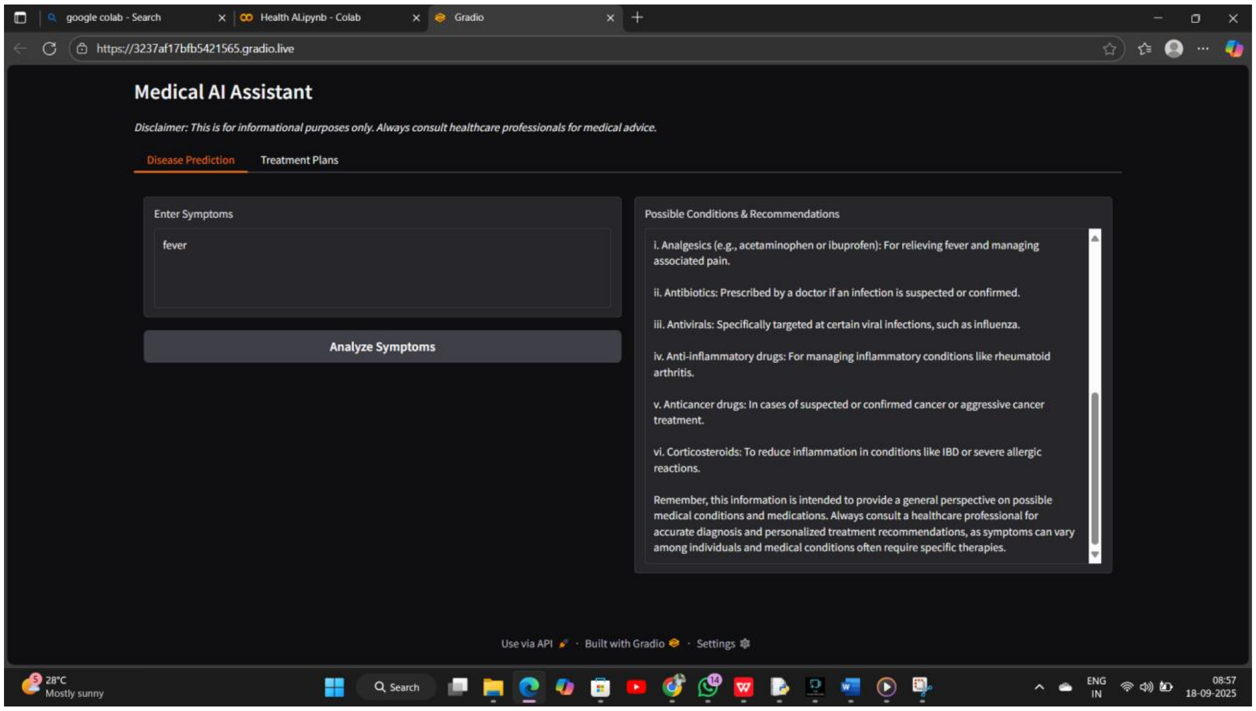
Label: "Analyze Symptoms"

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.


Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

Enter Symptoms

fever

Possible Conditions & Recommendations

i. Analgesics (e.g., acetaminophen or ibuprofen): For relieving fever and managing associated pain.

ii. Antibiotics: Prescribed by a doctor if an infection is suspected or confirmed.

iii. Antivirals: Specifically targeted at certain viral infections, such as influenza.

iv. Anti-inflammatory drugs: For managing inflammatory conditions like rheumatoid arthritis.

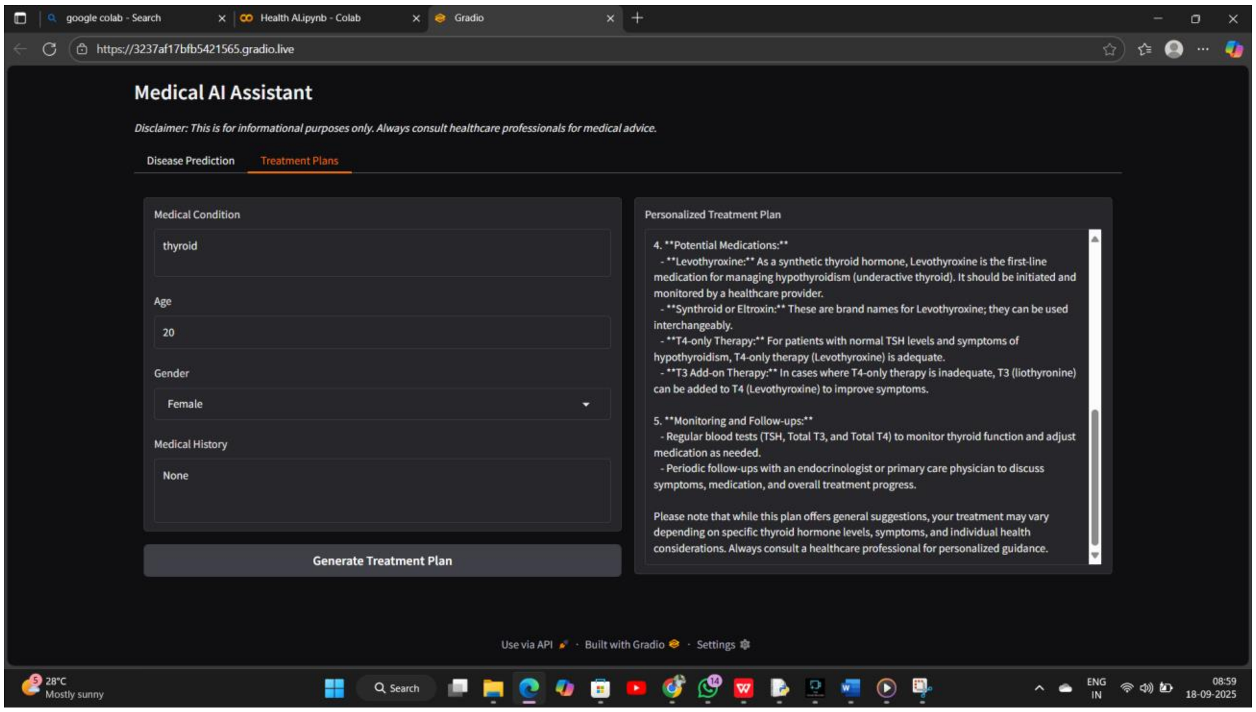
v. Anticancer drugs: In cases of suspected or confirmed cancer or aggressive cancer treatment.

vi. Corticosteroids: To reduce inflammation in conditions like IBD or severe allergic reactions.

Remember, this information is intended to provide a general perspective on possible medical conditions and medications. Always consult a healthcare professional for accurate diagnosis and personalized treatment recommendations, as symptoms can vary among individuals and medical conditions often require specific therapies.

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.


Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

Medical Condition

thyroid

Age

20

Gender

Female

Medical History

None

Personalized Treatment Plan

```
4. **Potential Medications:**
   - **Levothyroxine:** As a synthetic thyroid hormone, Levothyroxine is the first-line medication for managing hypothyroidism (underactive thyroid). It should be initiated and monitored by a healthcare provider.
   - **Synthroid or Eltroxin:** These are brand names for Levothyroxine; they can be used interchangeably.
   - **T4-only Therapy:** For patients with normal TSH levels and symptoms of hypothyroidism, T4-only therapy (Levothyroxine) is adequate.
   - **T3 Add-on Therapy:** In cases where T4-only therapy is inadequate, T3 (liothyronine) can be added to T4 (Levothyroxine) to improve symptoms.

5. **Monitoring and Follow-ups:**
   - Regular blood tests (TSH, Total T3, and Total T4) to monitor thyroid function and adjust medication as needed.
   - Periodic follow-ups with an endocrinologist or primary care physician to discuss symptoms, medication, and overall treatment progress.

Please note that while this plan offers general suggestions, your treatment may vary depending on specific thyroid hormone levels, symptoms, and individual health considerations. Always consult a healthcare professional for personalized guidance.
```

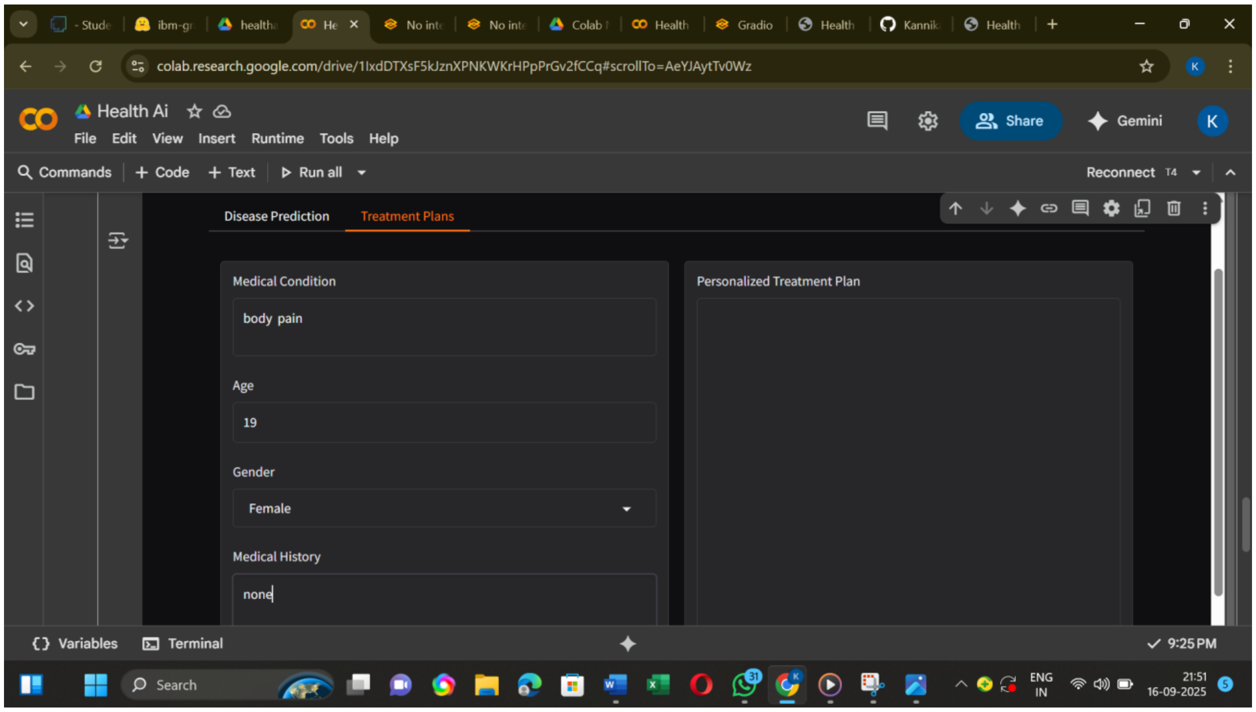
- Input Fields (gr.Textbox, gr.Number, gr.Dropdown):

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.


colab.research.google.com/drive/1IxdDTXsF5kJznXPNKWKrHPpPrGv2fCCq#scrollTo=AeYJAytTv0Wz
Health Ai
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
Code
Text
Run all
Reconnect T4
Disease Prediction
Treatment Plans
Variables
Terminal
9:25 PM

Medical Condition
body pain

Age
19

Gender
Female

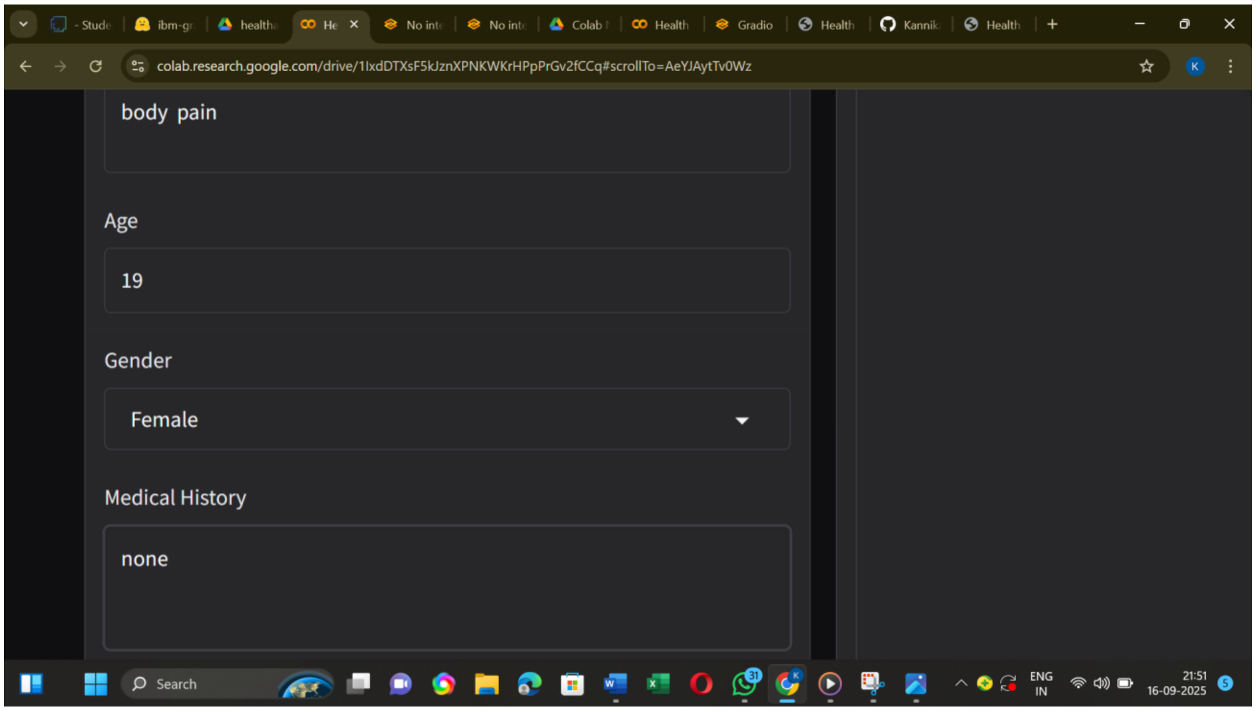
Medical History
none

Personalized Treatment Plan

Age (gr.Number):

Label: "Age"

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


body pain

Age

19

Gender

Female

Medical History

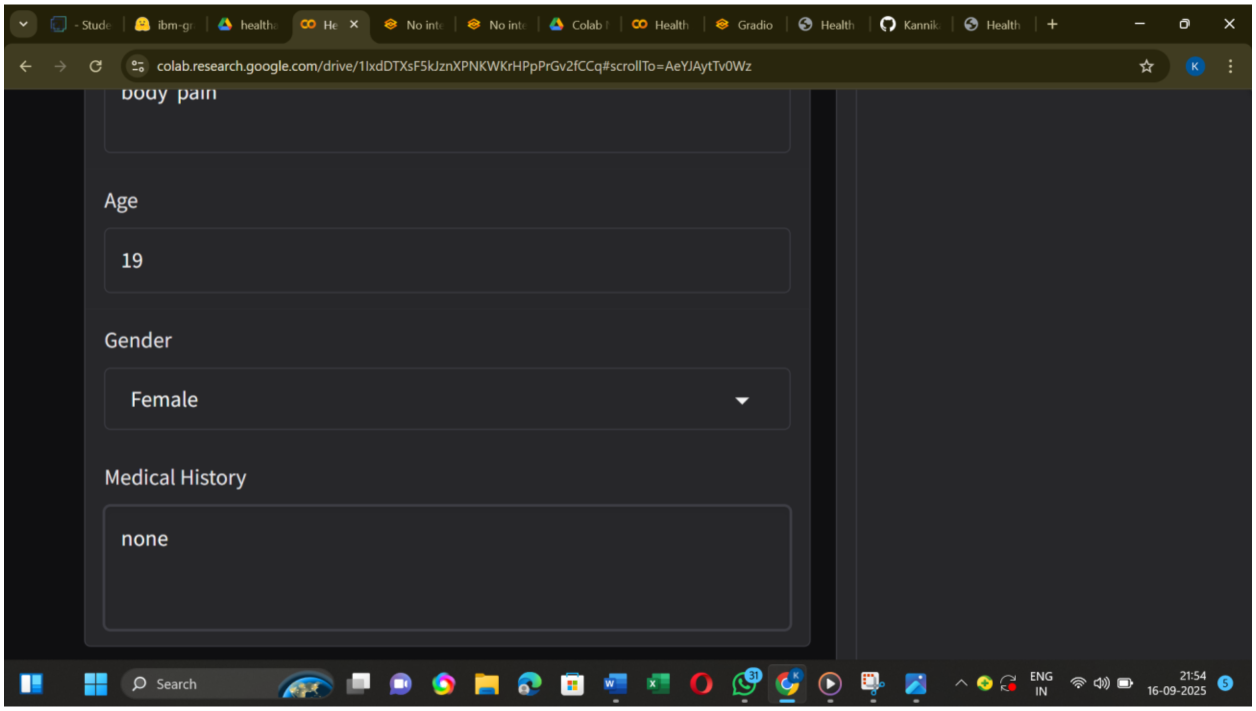
none

Gender (gr.Dropdown):

Label: "Gender"

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

Functionality: A dropdown menu to select gender.


body pain

Age

19

Gender

Female

Medical History

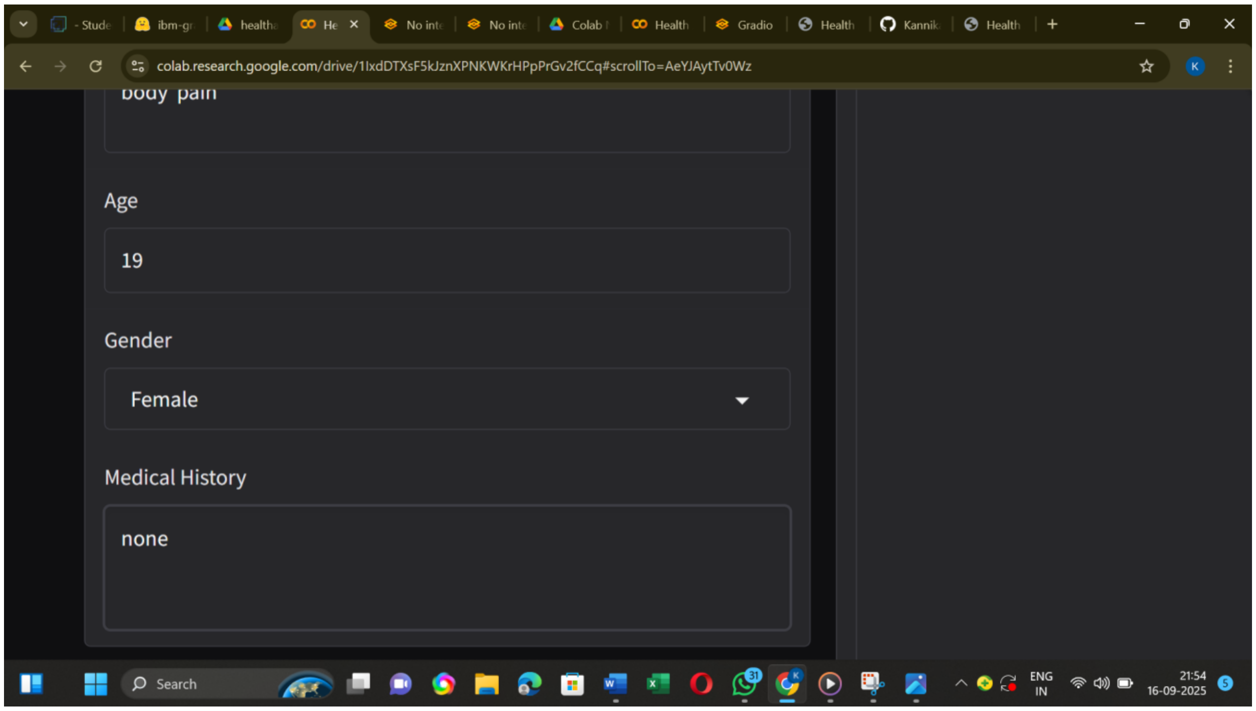
none

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.


body pain

Age

19

Gender

Female

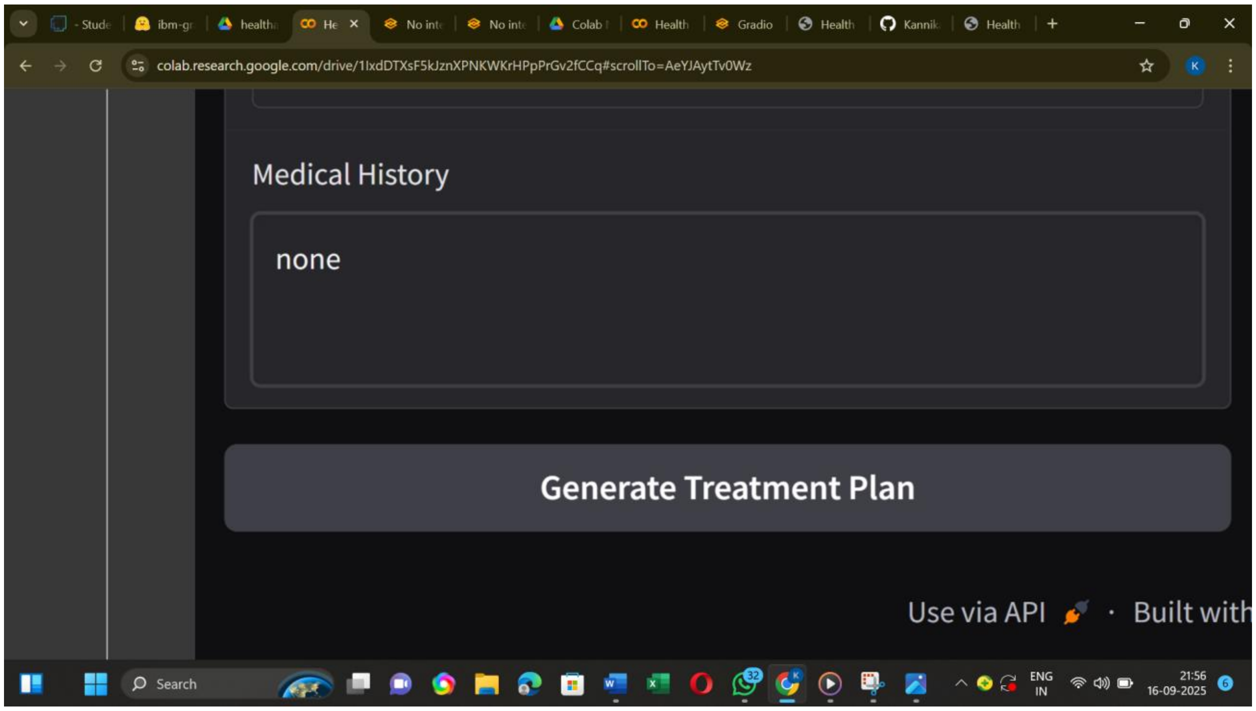
Medical History

none

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.

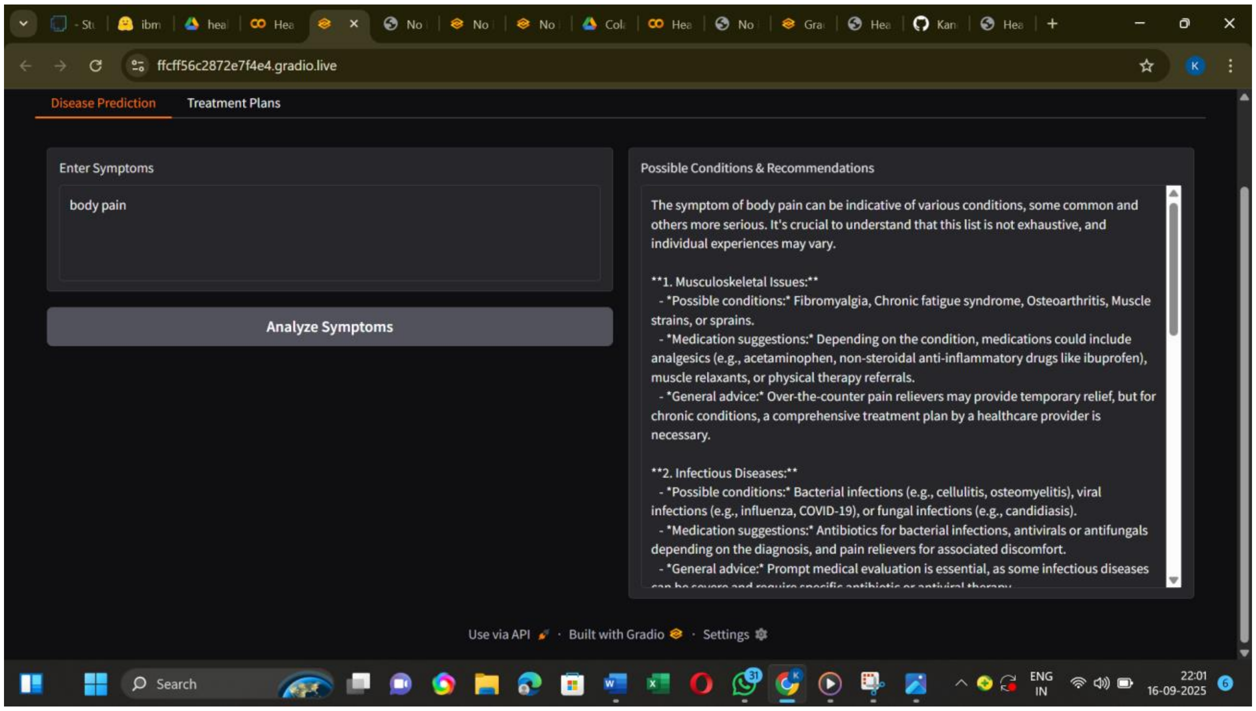

Medical History

none

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.


ffcff56c2872e7f4e4.gradio.live
Disease Prediction
Treatment Plans
Enter Symptoms
Analyze Symptoms
Possible Conditions & Recommendations
Use via API · Built with Gradio · Settings

body pain

```
The symptom of body pain can be indicative of various conditions, some common and
others more serious. It's crucial to understand that this list is not exhaustive, and
individual experiences may vary.

**1. Musculoskeletal Issues:**
  - *Possible conditions:* Fibromyalgia, Chronic fatigue syndrome, Osteoarthritis, Muscle
strains, or sprains.
  - *Medication suggestions:* Depending on the condition, medications could include
analgesics (e.g., acetaminophen, non-steroidal anti-inflammatory drugs like ibuprofen),
muscle relaxants, or physical therapy referrals.
  - *General advice:* Over-the-counter pain relievers may provide temporary relief, but for
chronic conditions, a comprehensive treatment plan by a healthcare provider is
necessary.

**2. Infectious Diseases:**
  - *Possible conditions:* Bacterial infections (e.g., cellulitis, osteomyelitis), viral
infections (e.g., influenza, COVID-19), or fungal infections (e.g., candidiasis).
  - *Medication suggestions:* Antibiotics for bacterial infections, antivirals or antifungals
depending on the diagnosis, and pain relievers for associated discomfort.
  - *General advice:* Prompt medical evaluation is essential, as some infectious diseases
```

10.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.

11. Known Issues (HealthAI)

- **Limited Dashboard Customization**: HealthAI's current dashboards provide only basic visualizations with restricted flexibility for customization based on user needs.
- **Dependency on Colab GPU Resources**: The application relies on Google Colab's free GPU, which can be inconsistent and may limit scalability and uptime.
- **No Role-Based Access System**: There is currently no authentication system to differentiate between citizens (patients) and officials (health workers/administrators), which restricts secure access to certain features.

12. Future Enhancements (HealthAI)

- **Role-Based Access Control (RBAC)**: Develop a secure authentication and authorization system to allow differentiated access for citizens (patients) and officials (e.g., healthcare professionals, administrators).
- **Multilingual Interface Support**: Integrate support for multiple languages to make HealthAI more inclusive and accessible to a broader population.
- **Enhanced Analytics Dashboards**: Expand the analytics capabilities with advanced, interactive, and customizable dashboards to better support decision-making.
- **Cloud Platform Deployment**: Migrate HealthAI from Google Colab to more robust cloud platforms such as IBM Cloud or AWS to enhance performance, reliability, and scalability.