Health AI: Intelligent Healthcare Assistant Project Documentation

Project Documentation

1.Introduction

- **Project title :** Health AI: Intelligent Healthcare Assistant Project Documentation
- **Team ID :** NM2025TMID02155
- **Team Leader :** LAKSHMANA ARJUN S
- **Team member :** BENAYA BLESSING D
- **Team member :** CHANDUR E
- **Team member :** ABISHEK G

2. project overview

Purpose:

To create an AI-powered medical assistant that helps users by:

- Predicting possible diseases based on entered symptoms.
- Suggesting treatment plans (general medication guidelines + home remedies) based on patient details.
- To provide quick, accessible, and informational medical guidance for users who want an initial idea about their health.
- To build a user-friendly web application with Gradio, where people can interact with the model easily.
- To show how Large Language Models (LLMs) like IBM Granite can be applied in the healthcare domain for informational support.
- To emphasize safe AI usage by always including a disclaimer: "This is for informational purposes only. Consult a doctor for proper diagnosis and treatment

Key Features

1. Disease Prediction

Users input symptoms (e.g., fever, cough, fatigue).The AI suggests possible conditions and general recommendations.Emphasizes visiting a doctor for confirmation.

2. Treatment Plan Generator

Takes inputs like:

Medical condition

Age

Gender

Medical history (allergies, past diseases, medications)

Generates personalized treatment suggestions:

Home remedies

General medication guidelines

Always includes a safety disclaimer.

3. Interactive Web Interface (Gradio)

Simple tab-based UI with two sections:

Disease Prediction

Treatment Plans

Textboxes and dropdowns for easy input.

Outputs displayed in large text areas.

Tech Stack

Python (main programming language)

Gradio (for creating the user-friendly web interface)

Hugging Face Transformers (for loading IBM Granite model)

PyTorch (for model execution with GPU/CPU support)

How It Works

1. User enters symptoms or patient details.

2. The system converts input into a prompt.

3. The Granite LLM processes the prompt and generates a text response.

4. The response is displayed in the Gradio UI.

3. Architecture

1. Model & Tokenizer Layer

Model Used: ibm-granite/granite-3.2-2b-instruct

Library: Hugging Face Transformers (AutoModelForCausalLM, AutoTokenizer)

Framework: PyTorch (with GPU/CPU support)

Loads the model and tokenizer. Handles text input → converts to tokens → generates AI response.

2. Response Generation Layer

Function: generate_response()

Converts the user’s input into a prompt.

Sends it to the model for text generation.

Decodes the model output into readable text.

Ensures safe response with temperature control and padding.

3. Application Logic Layer

Functions:

disease_prediction(symptoms) → Creates a medical prompt for symptoms.

treatment_plan(condition, age, gender, medical_history) → Creates a treatment plan prompt.

Both functions call the model via generate_response() and return AI-generated suggestions.

4. User Interface Layer (Gradio)

Framework: Gradio (gr.Blocks, gr.Tabs, gr.Textbox, gr.Button, etc.)

Two main tabs:

1. Disease Prediction Tab – User enters symptoms → gets possible conditions & recommendations.

2. Treatment Plan Tab – User enters condition + details → gets personalized plan.

UI is interactive and user-friendly.

5. Deployment Layer

app.launch(share=True) → Launches the web app and creates a shareable public link.

Runs locally or can be hosted online (Hugging Face Spaces, Colab, etc.).

✅ High-Level Flow (Architecture Diagram in Words)

User Input (Symptoms / Condition Details)

|

▼

Gradio UI (Textbox, Dropdown, Buttons)

|

▼

Application Logic (disease_prediction / treatment_plan functions)

|

▼

Prompt Generator (formats input as prompt)

|

▼

LLM (IBM Granite model via Hugging Face + PyTorch)

|

▼

Response Decoder (generate_response function)

|

▼

Gradio UI Output (Displays Conditions / Treatment Plan to user)

4. Setup Instructions

Step 1: Install Dependencies

Make sure you have Python 3.9+ installed, then run:

```
pip install gradio torch transformers
```

Step 2: Save the Code

Save your Python file as:

medical_ai_assistant.py

Step 3: Run the Application

Run the script:

```
python medical_ai_assistant.py
```

Step 4: Access the App

The terminal will show a local URL (e.g., http://127.0.0.1:7860)

And a public share link (because of share=True)

5. Folder Structure

```
Medical-AI-Assistant/
│
├── medical_ai_assistant.py     # Main Python script
├── requirements.txt            # Project dependencies
├── README.md                   # Project documentation
│
├── data/                       # (Optional) Store sample input/output data
│   └── sample_symptoms.txt
│
├── docs/                       # Documentation
```

```
|   └── architecture.png         # Architecture diagram (if created)
|
└── models/                  # (Optional) Store custom models if used
```

6. Running the Application

1. Open a terminal in the project folder.

2. Run the app:

```
python medical_ai_assistant.py
```

3. The terminal will display two links:

Local URL (e.g., http://127.0.0.1:7860) → runs on your computer.

Public Share URL → can be shared with others to test online.

4. Open the link in your browser.

5. Enter symptoms or patient details → get results instantly.

7. API Documentation

Although this is a Gradio UI app, the code can also be treated as an API service.

Endpoints (Functions)

1. disease_prediction(symptoms: str) -> str

Description: Analyzes symptoms and suggests possible conditions with recommendations.

Input:

symptoms (string) – comma-separated symptoms.

Output:

String containing conditions & recommendations.

8. Authentication

1. Simple Authentication – Use auth=("username", "password") in app.launch().

2. Multiple Users – Use auth=[("user1", "pass1"), ("user2", "pass2")].

3. Custom Function – Define an authenticate(username, password) function and pass it to auth.

4. No Authentication – Default (app.launch(share=True)) means anyone can access.

5. Recommendation – For medical apps, enable authentication for security.

9. User Interface

1. Framework – The UI is built using Gradio Blocks.

2. Tabs – Two main sections:

Disease Prediction – Accepts symptoms input, shows possible conditions.

Treatment Plans – Accepts patient details (condition, age, gender, history), shows treatment plan.

3. Input Fields –

Textbox for symptoms/conditions/history.

Number input for age.

Dropdown for gender.

4. Output Fields – Large textboxes to display AI-generated analysis or treatment plan.

5. Buttons – "Analyze Symptoms" and "Generate Treatment Plan" trigger AI functions.

10. Testing

1. Functional Testing –

- Enter sample symptoms (e.g., fever, cough, fatigue) → Check if disease prediction output is meaningful.
- Enter sample condition (e.g., Diabetes, Age: 45, Gender: Male, History: hypertension) → Check treatment plan.

2. UI Testing –

- Ensure tabs switch properly.
- Buttons respond correctly.
- Outputs display without cutting text.

3. Performance Testing –

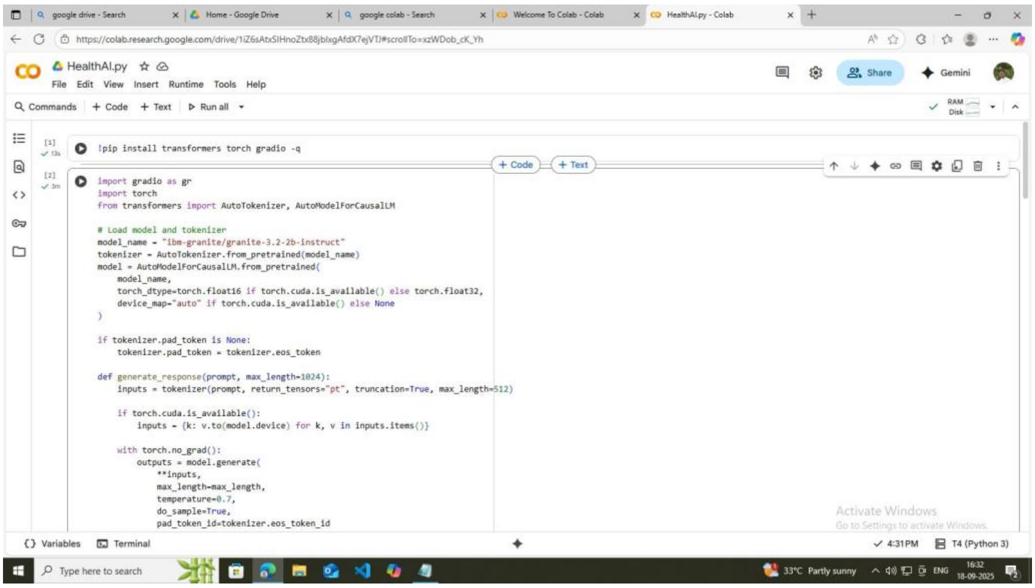
- Check response time with different inputs.
- Test on CPU vs GPU (if available).

4. Error Handling Testing –

- Leave fields blank → Ensure model still runs or shows a safe response.
- Enter long text → Verify truncation works
  (limited to 512 tokens).

11.Screen shots

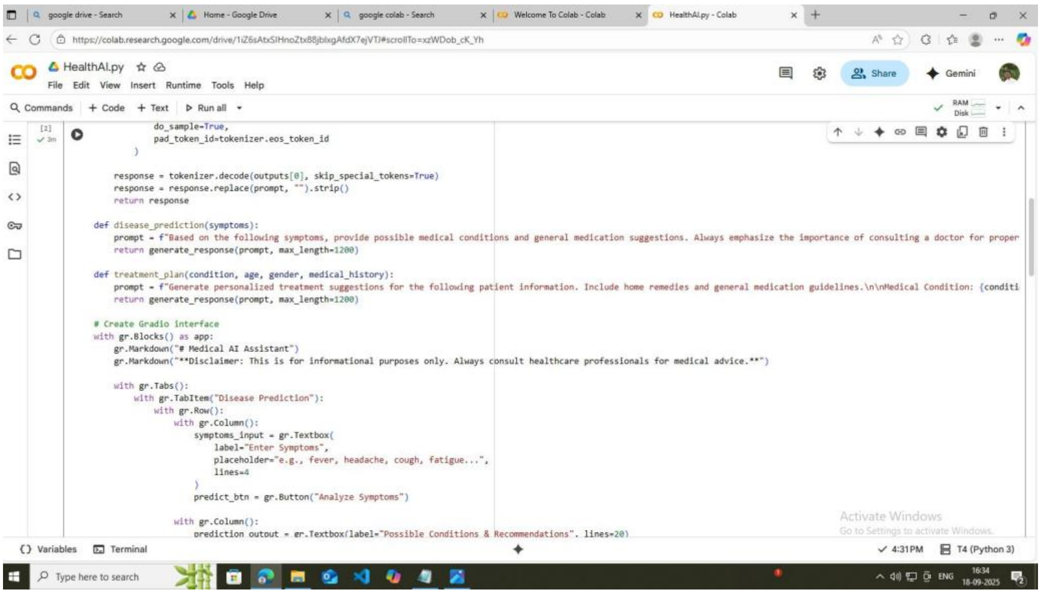
1.Input:



HealthAI.py
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini
Variables Terminal
4:31PM T4 (Python 3)

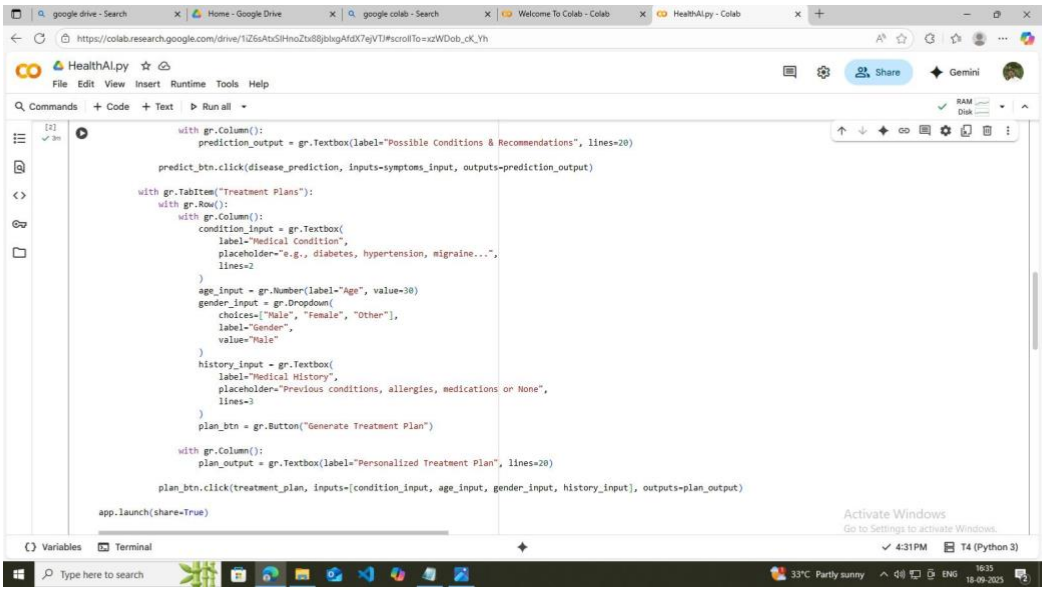
```
!pip install transformers torch gradio -q
```

```
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)

    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
```

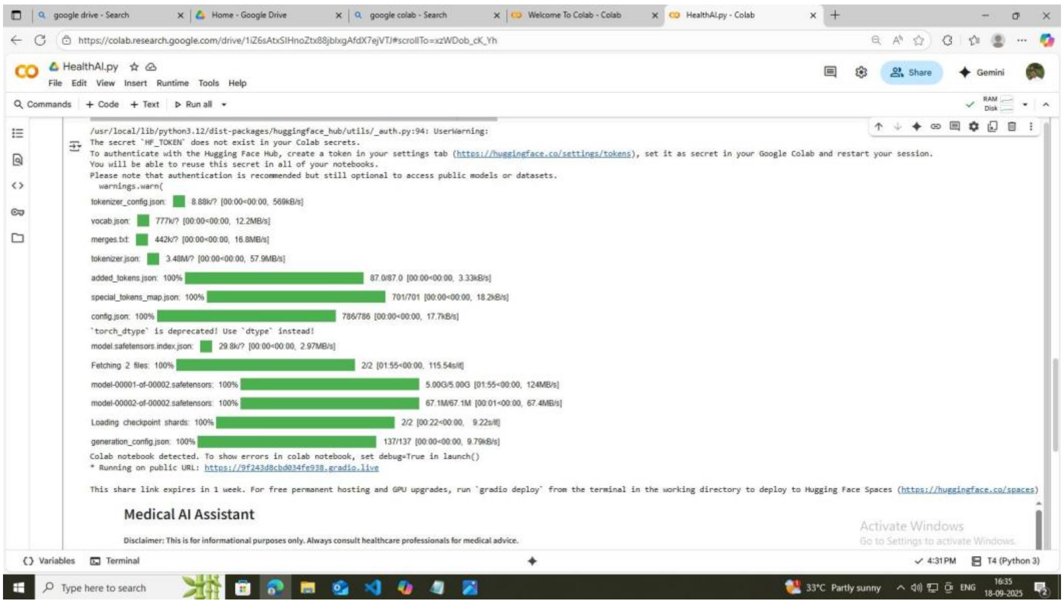


HealthAI.py
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini
Variables Terminal
4:31PM T4 (Python 3)

```
            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
    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: {conditi
    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)
```


HealthAI.py - Colab
https://colab.research.google.com/drive/1iZ6sAtxSIHnoZtx88jblxgAfdX7ejVTJ#scrollTo=xzWDob_cK_Yh
HealthAI.py
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
Gemini
Variables
Terminal
4:31PM
T4 (Python 3)

```
            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)
```

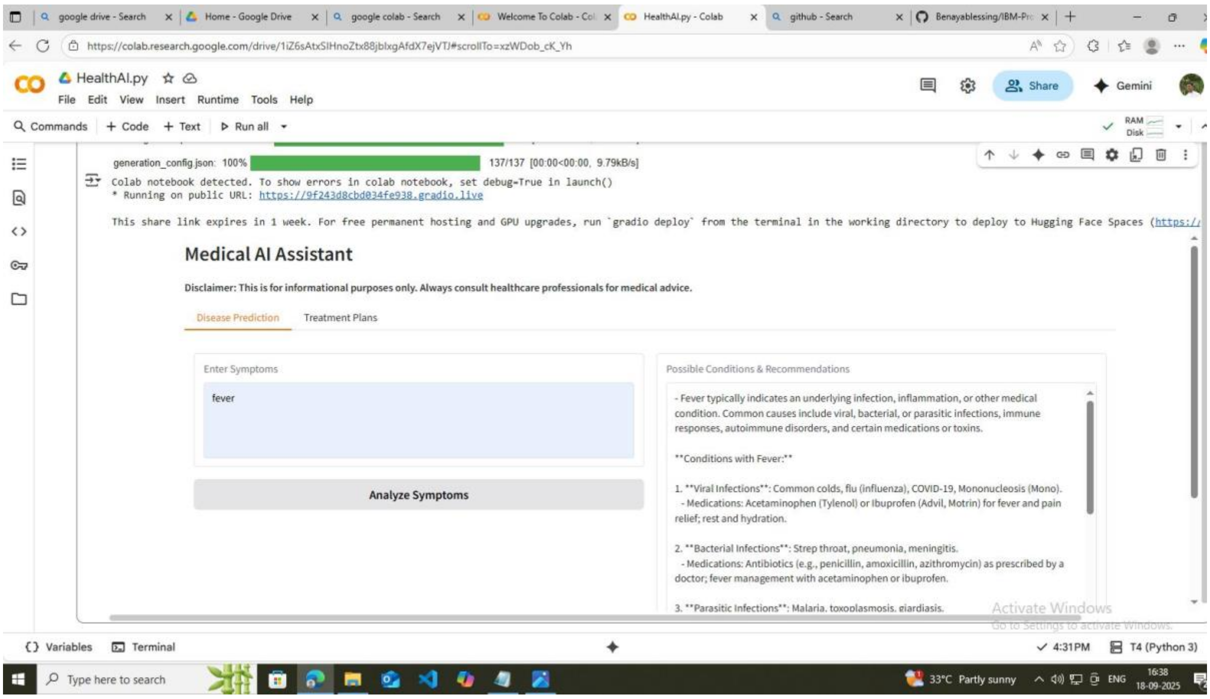

HealthAI.py - Colab
https://colab.research.google.com/drive/1iZ6sAtxSIHnoZtx88jblxgAfdX7ejVTJ#scrollTo=xzWDob_cK_Yh
HealthAI.py
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
Gemini
Variables
Terminal
4:31PM
T4 (Python 3)

```
/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 session.
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, 569kB/s]
vocab.json:  777k/? [00:00<00:00, 12.2MB/s]
merges.txt:  442k/? [00:00<00:00, 16.8MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 57.9MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 3.33kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 18.2kB/s]
config.json: 100%  786/786 [00:00<00:00, 17.7kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.97MB/s]
Fetching 2 files: 100%  2/2 [01:55<00:00, 115.54s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [01:55<00:00, 124MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:01<00:00, 67.4MB/s]
Loading checkpoint shards: 100%  2/2 [00:22<00:00, 9.22s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 9.79kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://9f243d8cbd034fe938.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 (https://huggingface.co/spaces)
```

Medical AI Assistant

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

Output:



```
generation_config.json: 100% 137/137 [00:00<00:00, 9.79kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://9f243d8cbd034fe938.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 (https://
```

Medical AI Assistant

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

Disease Prediction | Treatment Plans

Enter Symptoms: fever

Analyze Symptoms

Possible Conditions & Recommendations

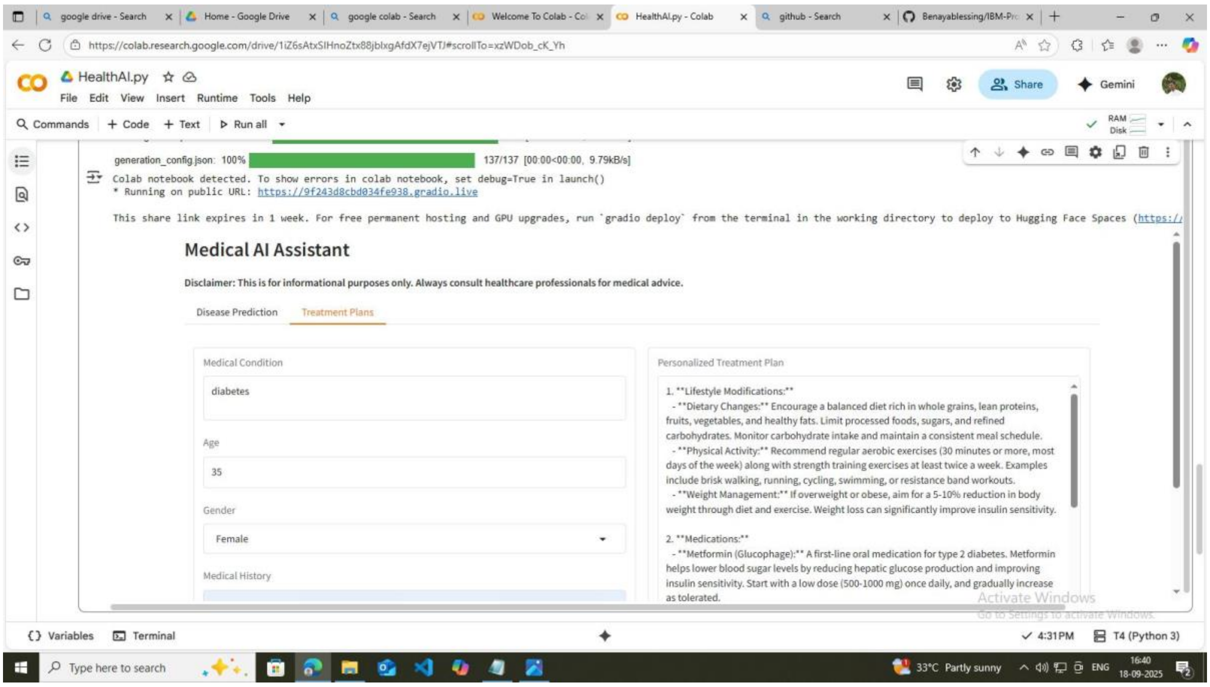
- Fever typically indicates an underlying infection, inflammation, or other medical condition. Common causes include viral, bacterial, or parasitic infections, immune responses, autoimmune disorders, and certain medications or toxins.

Conditions with Fever:

1. **Viral Infections**: Common colds, flu (influenza), COVID-19, Mononucleosis (Mono).
 - Medications: Acetaminophen (Tylenol) or Ibuprofen (Advil, Motrin) for fever and pain relief; rest and hydration.

2. **Bacterial Infections**: Strep throat, pneumonia, meningitis.
 - Medications: Antibiotics (e.g., penicillin, amoxicillin, azithromycin) as prescribed by a doctor; fever management with acetaminophen or ibuprofen.

3. **Parasitic Infections**: Malaria, toxoplasmosis, giardiasis.



```
generation_config.json: 100% 137/137 [00:00<00:00, 9.79kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://9f243d8cbd034fe938.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 (https://
```

Medical AI Assistant

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

Disease Prediction | Treatment Plans

Medical Condition: diabetes

Age: 35

Gender: Female

Medical History

Personalized Treatment Plan

1. **Lifestyle Modifications:**
 - **Dietary Changes:** Encourage a balanced diet rich in whole grains, lean proteins, fruits, vegetables, and healthy fats. Limit processed foods, sugars, and refined carbohydrates. Monitor carbohydrate intake and maintain a consistent meal schedule.
 - **Physical Activity:** Recommend regular aerobic exercises (30 minutes or more, most days of the week) along with strength training exercises at least twice a week. Examples include brisk walking, running, cycling, swimming, or resistance band workouts.
 - **Weight Management:** If overweight or obese, aim for a 5-10% reduction in body weight through diet and exercise. Weight loss can significantly improve insulin sensitivity.

2. **Medications:**
 - **Metformin (Glucophage):** A first-line oral medication for type 2 diabetes. Metformin helps lower blood sugar levels by reducing hepatic glucose production and improving insulin sensitivity. Start with a low dose (500-1000 mg) once daily, and gradually increase as tolerated.