HEALTH AI ASSISTANT

PROJECT DOCUMENTATION

INTRODUCTION :

- **Team Leader:** Sakthidharan. U
- **Team Member:** Saranraj. M
- **Team Member:** Kishore. R
- **Team Member:** Karthir Kanth. Y. K
- **Team Member:** Sam. M

PROJECT OVERVIEW:

The Health AI Assistant is designed with **two main functionalities**:

1. **Disease Prediction** – Users input symptoms (e.g., fever, cough, fatigue), and the system suggests possible medical conditions along with general recommendations.
2. **Treatment Plan Generation** – Users provide details such as medical condition, age, gender, and medical history. The system then generates a personalized treatment guideline that includes general medication information and home remedies.

The application is built using **Gradio** for the interface, **HuggingFace Transformers** for natural language processing, and **PyTorch** for deep learning execution. The backend integrates seamlessly with the frontend, ensuring smooth input-output processing.

Key goals of the project include:

- Providing **informational healthcare support** in a simple format.
- Highlighting the role of **AI in digital health transformation**.
- Developing a system that is **scalable, modular, and user-friendly**.

The project demonstrates the use of AI in **non-critical healthcare assistance**, ensuring ethical boundaries are respected by including disclaimers and encouraging consultation with professionals

ARCHITECTURE:

The Health AI Assistant follows a **three-tier modular architecture**:

1. **User Interface (UI):** Built with Gradio, the UI allows users to input symptoms or health details and receive responses in an accessible format.
2. **Backend Processing Layer:** Implemented in Python, this layer manages tokenization, model inference, and response formatting.
3. **AI Model Layer:** Powered by IBM Granite's pre-trained instruction model (via HuggingFace), this layer generates meaningful outputs.

The architecture ensures **modularity, scalability, and maintainability**. Each component can be independently updated or replaced without affecting the rest of the system.

Communication between the layers is handled using structured prompts and tokenized inputs. The GPU acceleration support in PyTorch enhances performance, enabling faster response generation.

This architecture demonstrates how **NLP models can be integrated into real-world applications** through a well-defined software pipeline.

SETUP INSTRUCTIONS:

To set up the Health AI Assistant, the following steps must be followed:

1. **Install Dependencies:**

```
pip install torch transformers gradio
```

2. **Clone Project Repository:**

```
git clone <repository-link>
cd HealthAI
```

3. **Run Application:**

```
python app.py
```

The application automatically launches a Gradio interface, which can be accessed locally or shared via a public link. GPU acceleration is recommended but optional.

The setup requires Python 3.9+ and a stable internet connection for downloading pre-trained models. These instructions make deployment simple for developers and evaluators.

FOLDER STRUCTURE:

The project follows an **organized folder structure** for clarity and scalability:

```
HealthAI/
|── app.py
|── requirements.txt
|── README.md
|── /notebooks
|   └── HealthAI.ipynb
|── /docs
|   └── Project_Report.pdf
|── /models
|   └── pretrained/
|── /tests
```

```
│   └── unit_tests.py
```

- **app.py** → Main application file.
- **requirements.txt** → Dependency list.
- **notebooks/** → Demonstration and experiments.
- **docs/** → Documentation and reports.
- **models/** → Storage for pre-trained/fine-tuned models.
- **tests/** → Unit and integration test scripts.

This structure ensures **clean code management** and supports future expansion.

RUNNING THE APPLICATIONS:

The Health AI Assistant can be run in two ways:

1. **Local Execution:** Run `python app.py`, and a Gradio web interface opens in the browser.
2. **Google Colab:** Execute the `HealthAI.ipynb` notebook to run the project in a cloud environment.

The system generates a **shareable link** when hosted via Gradio, enabling remote testing.

The dual execution modes ensure accessibility for both developers and non-technical users.

API DOCUMENTATION:

The project exposes two primary backend functions:

- **disease_prediction(symptoms):**
    - Input: Symptom text (e.g., "fever, headache, cough").
    - Output: Possible conditions and general recommendations.
- **treatment_plan(condition, age, gender, history):**

    - Input: Patient details.
    - Output: Personalized treatment plan including remedies and medication guidelines.

Both functions are designed to be **modular and reusable**, allowing easy integration into other platforms.

AUTHENTICATION:

Currently, authentication is **not implemented**, since this is a proof-of-concept academic project. However, in production, authentication is critical to protect patient data.

Future enhancements may include:

- User accounts with login credentials.
- API token-based authentication.
- End-to-end encryption for sensitive data.

This ensures compliance with **healthcare data privacy standards**.

USER INTERFACE :

The Gradio-based user interface emphasizes **simplicity and accessibility**.

- **Tab 1: Disease Prediction** → Input symptoms and receive possible conditions.
- **Tab 2: Treatment Plan** → Enter medical details and receive personalized recommendations.

The UI is designed to be minimal, intuitive, and mobile-friendly. Its clarity ensures that even **non-technical users** can interact with the system effectively.

TESTING :

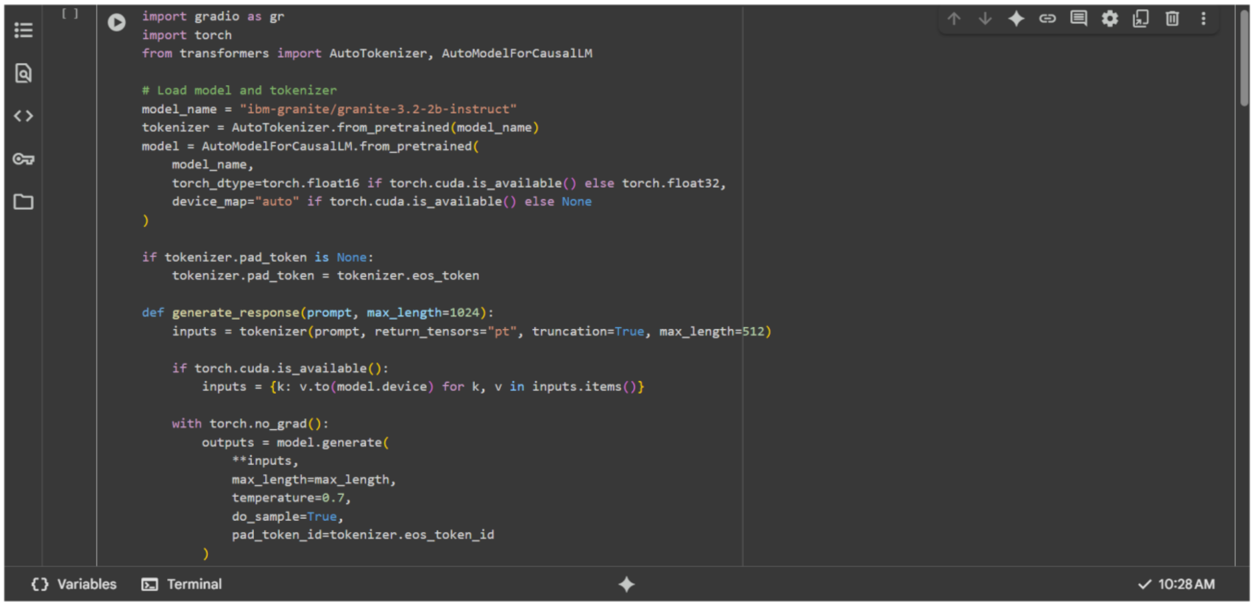
Testing was carried out at multiple levels:

- **Unit Testing:** Functions like `disease_prediction` and `treatment_plan` were tested with sample inputs.
- **Integration Testing:** Verified that the UI, backend, and AI model worked seamlessly.
- **User Testing:** Simulated real-world inputs to evaluate usability and accuracy.

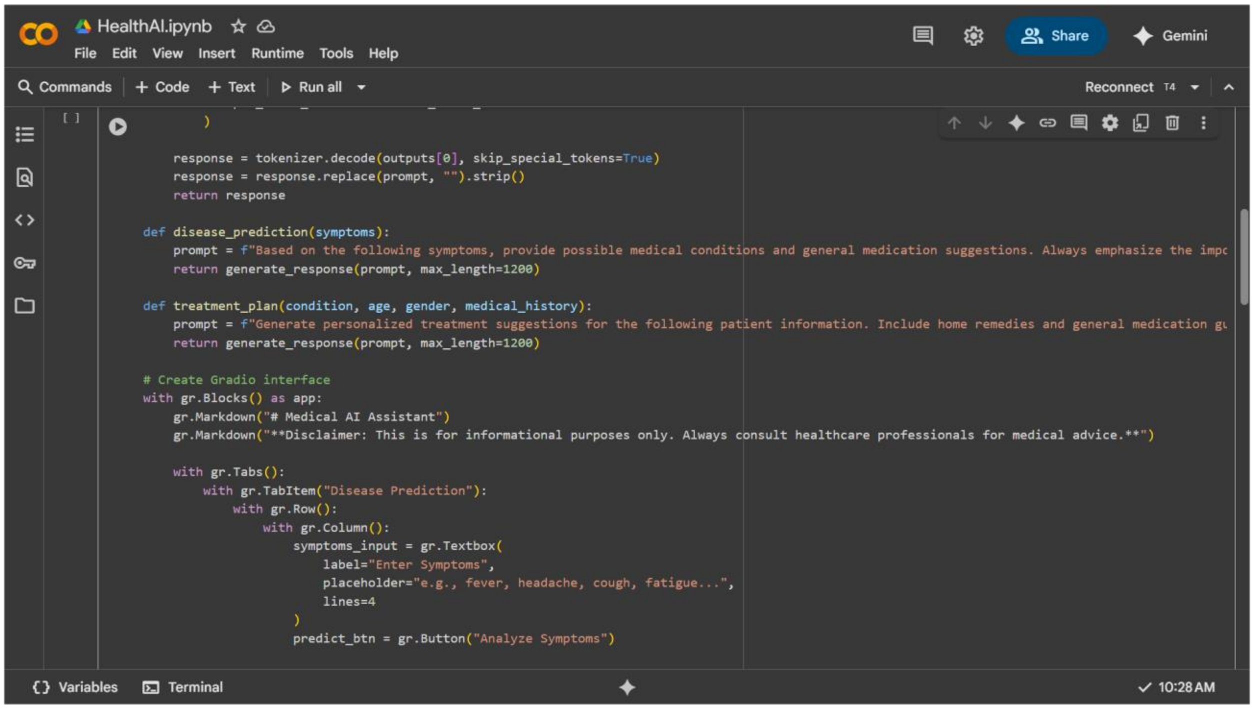
Testing confirmed that the system produced meaningful and consistent outputs, reinforcing its reliability as an academic project.

SCREEN SHOTS :

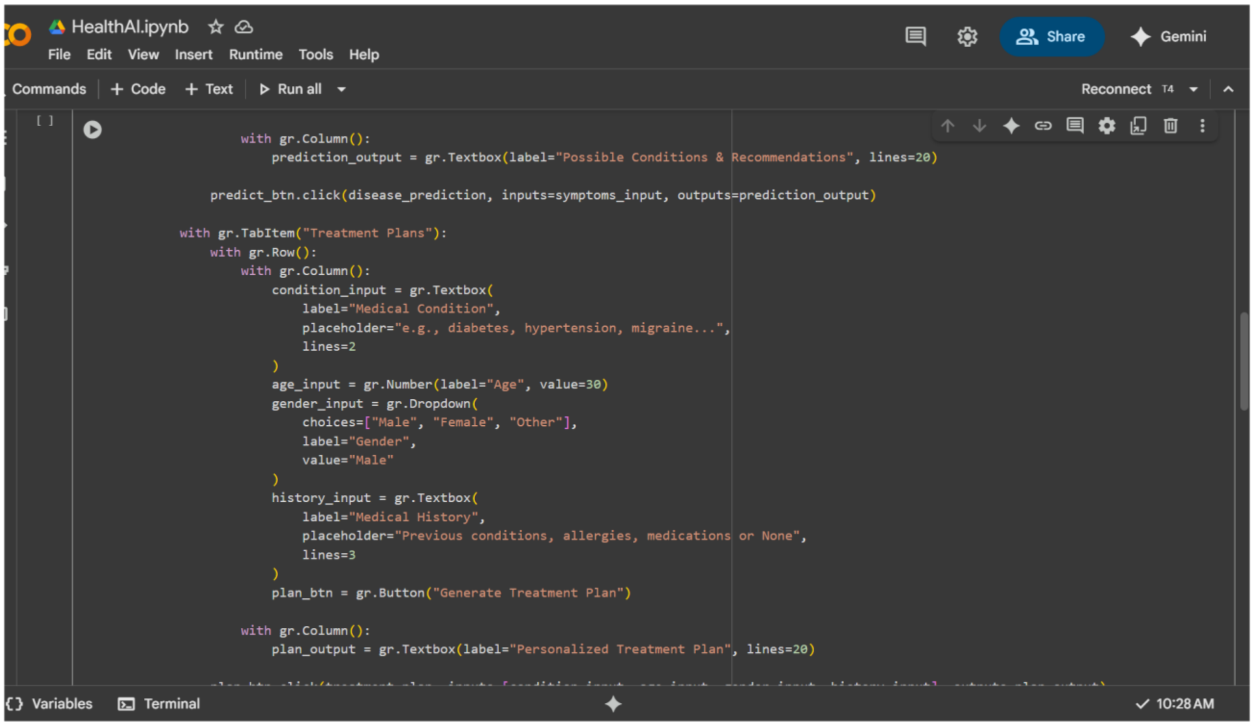
The Code of the program in google colab :



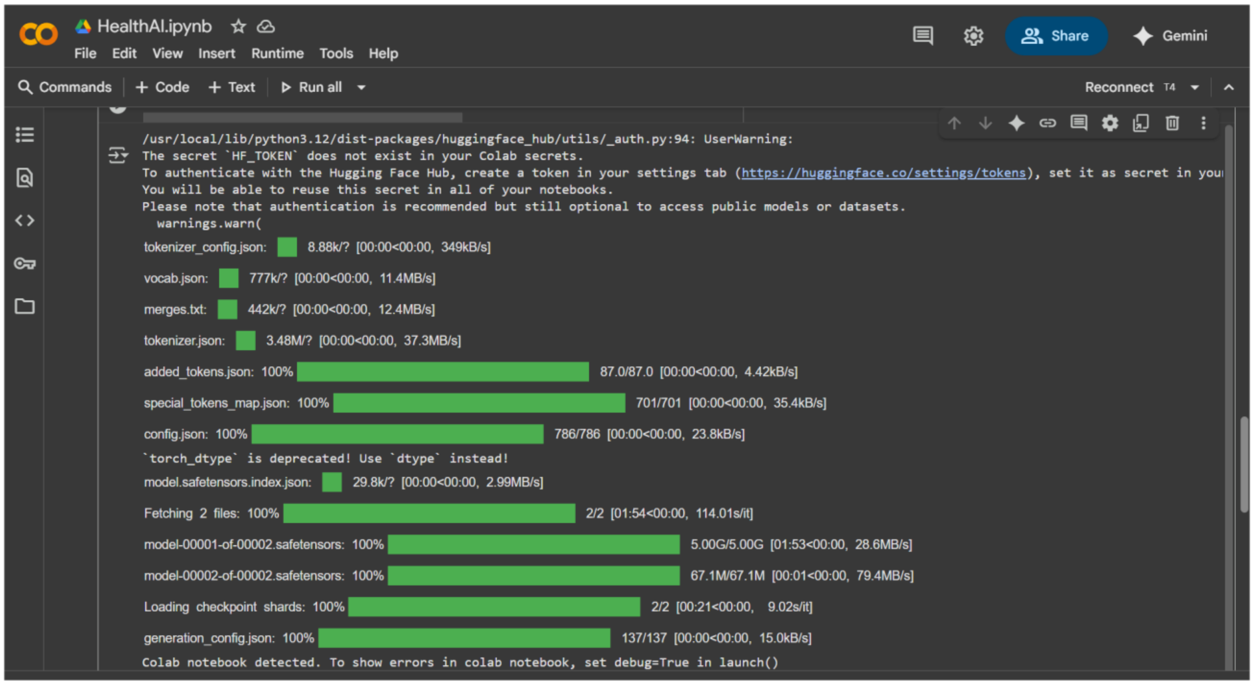
```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini Reconnect T4
Variables Terminal
10:28 AM

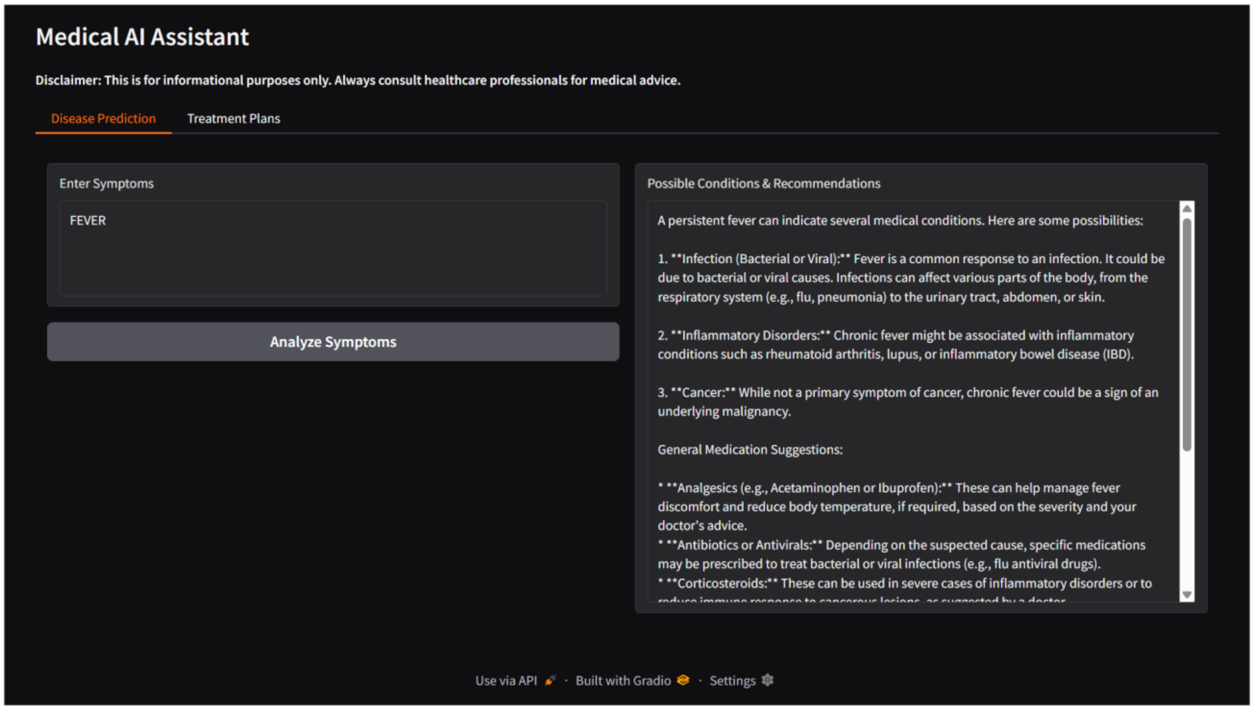
```
        )

    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 impo
    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 gu
    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")
```


HealthAI.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini Reconnect T4
Variables Terminal
10:28 AM

```
                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)
```


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

```
/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 you
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, 349kB/s]
vocab.json:  777k/? [00:00<00:00, 11.4MB/s]
merges.txt:  442k/? [00:00<00:00, 12.4MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 37.3MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 4.42kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 35.4kB/s]
config.json: 100%  786/786 [00:00<00:00, 23.8kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.99MB/s]
Fetching 2 files: 100%  2/2 [01:54<00:00, 114.01s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [01:53<00:00, 28.6MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:01<00:00, 79.4MB/s]
Loading checkpoint shards: 100%  2/2 [00:21<00:00, 9.02s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 15.0kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
```



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

Enter Symptoms

FEVER

Possible Conditions & Recommendations

A persistent fever can indicate several medical conditions. Here are some possibilities:

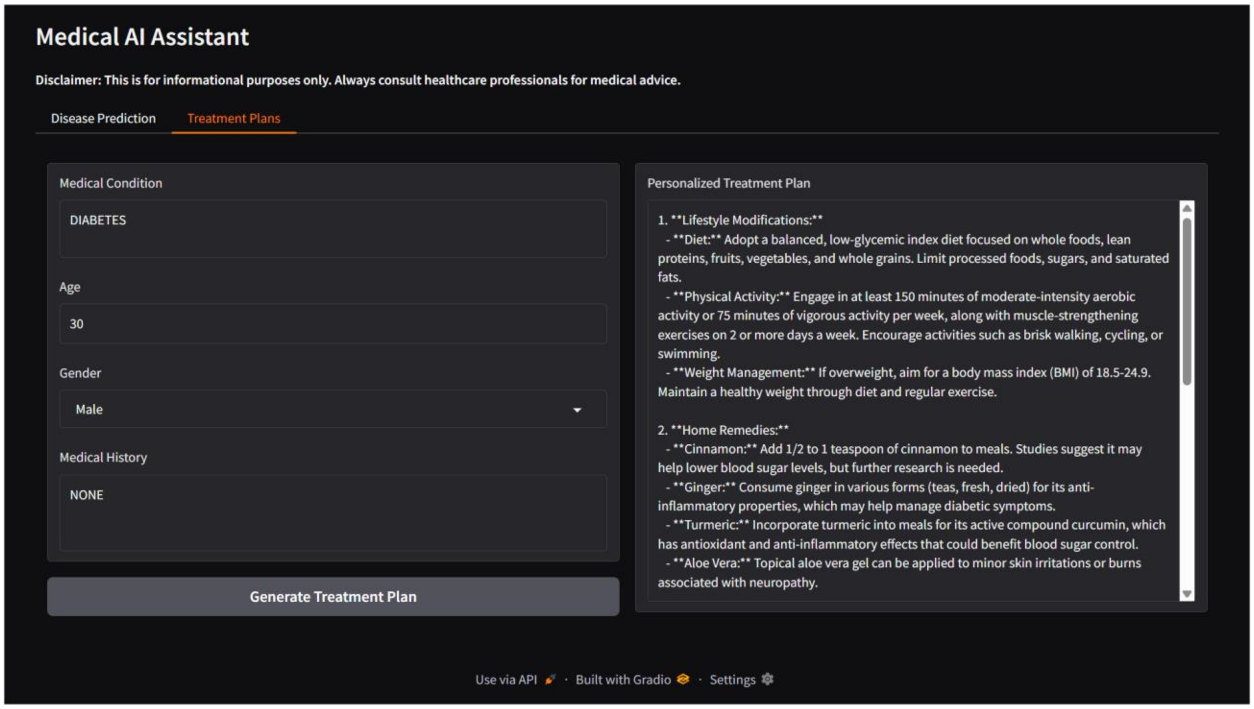
1. **Infection (Bacterial or Viral):** Fever is a common response to an infection. It could be due to bacterial or viral causes. Infections can affect various parts of the body, from the respiratory system (e.g., flu, pneumonia) to the urinary tract, abdomen, or skin.

2. **Inflammatory Disorders:** Chronic fever might be associated with inflammatory conditions such as rheumatoid arthritis, lupus, or inflammatory bowel disease (IBD).

3. **Cancer:** While not a primary symptom of cancer, chronic fever could be a sign of an underlying malignancy.

General Medication Suggestions:

* **Analgesics (e.g., Acetaminophen or Ibuprofen):** These can help manage fever discomfort and reduce body temperature, if required, based on the severity and your doctor's advice.
* **Antibiotics or Antivirals:** Depending on the suspected cause, specific medications may be prescribed to treat bacterial or viral infections (e.g., flu antiviral drugs).
* **Corticosteroids:** These can be used in severe cases of inflammatory disorders or to reduce immune response to cancerous lesions, as suggested by a doctor.



Medical AI Assistant

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

Medical Condition: DIABETES

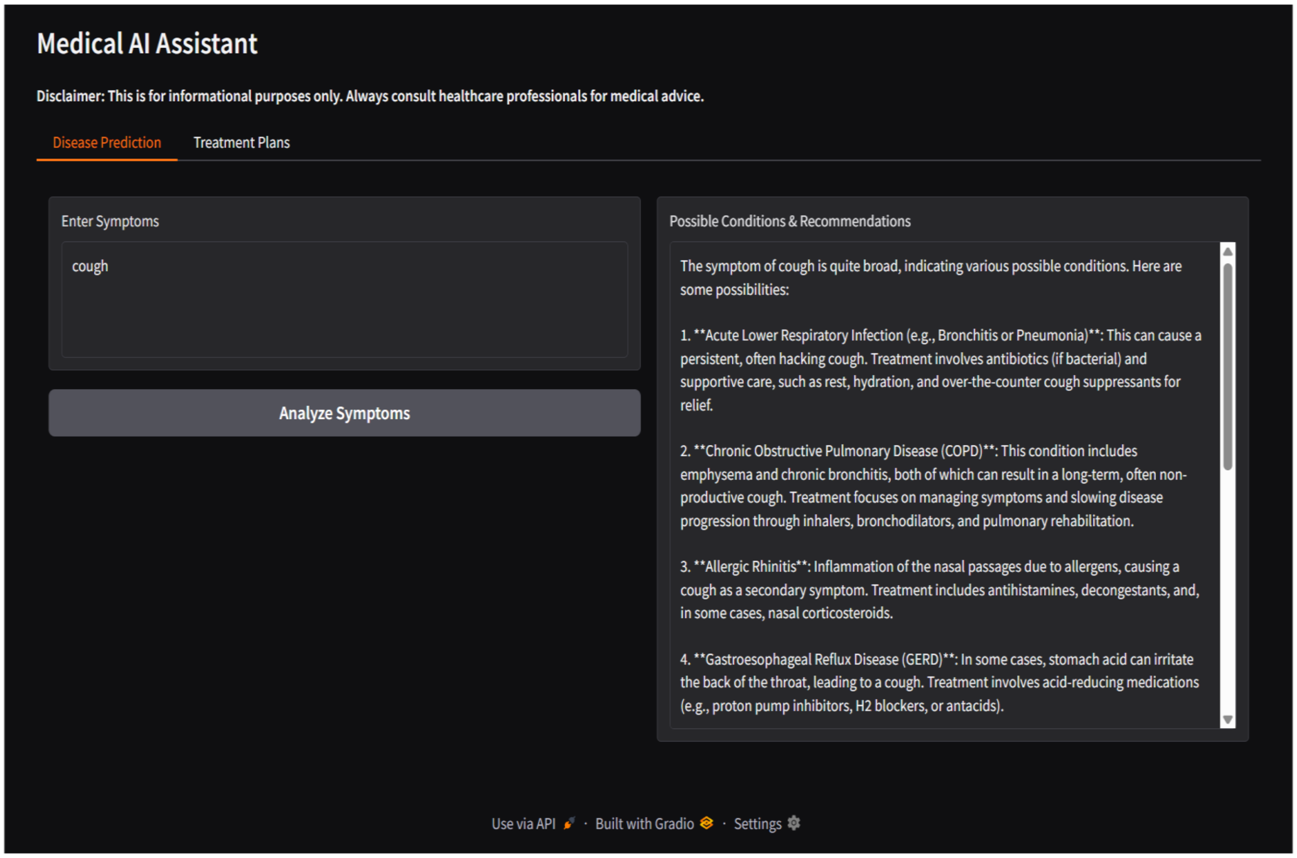
Age: 30

Gender: Male

Medical History: NONE

Personalized Treatment Plan

```
1. **Lifestyle Modifications:**
  - **Diet:** Adopt a balanced, low-glycemic index diet focused on whole foods, lean proteins, fruits, vegetables, and whole grains. Limit processed foods, sugars, and saturated fats.
  - **Physical Activity:** Engage in at least 150 minutes of moderate-intensity aerobic activity or 75 minutes of vigorous activity per week, along with muscle-strengthening exercises on 2 or more days a week. Encourage activities such as brisk walking, cycling, or swimming.
  - **Weight Management:** If overweight, aim for a body mass index (BMI) of 18.5-24.9. Maintain a healthy weight through diet and regular exercise.

2. **Home Remedies:**
  - **Cinnamon:** Add 1/2 to 1 teaspoon of cinnamon to meals. Studies suggest it may help lower blood sugar levels, but further research is needed.
  - **Ginger:** Consume ginger in various forms (teas, fresh, dried) for its anti-inflammatory properties, which may help manage diabetic symptoms.
  - **Turmeric:** Incorporate turmeric into meals for its active compound curcumin, which has antioxidant and anti-inflammatory effects that could benefit blood sugar control.
  - **Aloe Vera:** Topical aloe vera gel can be applied to minor skin irritations or burns associated with neuropathy.
```



Medical AI Assistant

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

Disease Prediction | Treatment Plans

Enter Symptoms

cough

Analyze Symptoms

Possible Conditions & Recommendations

The symptom of cough is quite broad, indicating various possible conditions. Here are some possibilities:

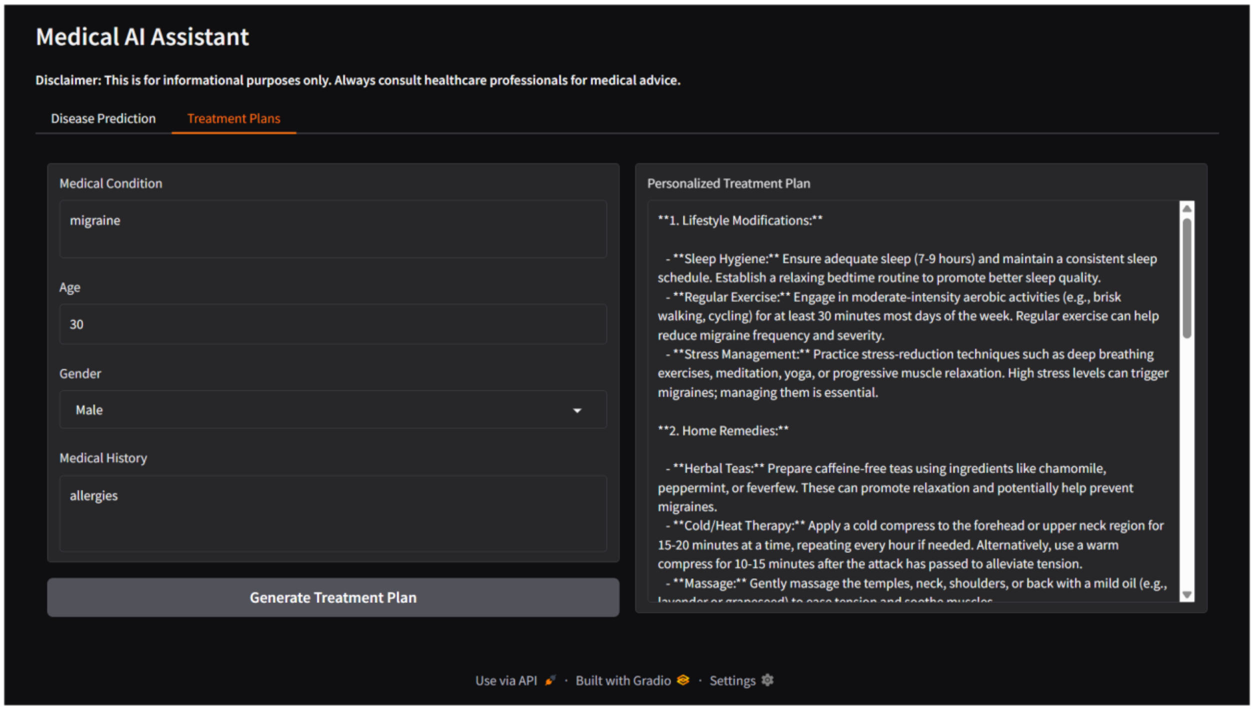
1. **Acute Lower Respiratory Infection (e.g., Bronchitis or Pneumonia)**: This can cause a persistent, often hacking cough. Treatment involves antibiotics (if bacterial) and supportive care, such as rest, hydration, and over-the-counter cough suppressants for relief.

2. **Chronic Obstructive Pulmonary Disease (COPD)**: This condition includes emphysema and chronic bronchitis, both of which can result in a long-term, often non-productive cough. Treatment focuses on managing symptoms and slowing disease progression through inhalers, bronchodilators, and pulmonary rehabilitation.

3. **Allergic Rhinitis**: Inflammation of the nasal passages due to allergens, causing a cough as a secondary symptom. Treatment includes antihistamines, decongestants, and, in some cases, nasal corticosteroids.

4. **Gastroesophageal Reflux Disease (GERD)**: In some cases, stomach acid can irritate the back of the throat, leading to a cough. Treatment involves acid-reducing medications (e.g., proton pump inhibitors, H2 blockers, or antacids).

Use via API · Built with Gradio · Settings



Medical AI Assistant

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

Disease Prediction | Treatment Plans

Medical Condition

migraine

Age

30

Gender

Male

Medical History

allergies

Generate Treatment Plan

Personalized Treatment Plan

1. Lifestyle Modifications:

 - **Sleep Hygiene:** Ensure adequate sleep (7-9 hours) and maintain a consistent sleep schedule. Establish a relaxing bedtime routine to promote better sleep quality.
 - **Regular Exercise:** Engage in moderate-intensity aerobic activities (e.g., brisk walking, cycling) for at least 30 minutes most days of the week. Regular exercise can help reduce migraine frequency and severity.
 - **Stress Management:** Practice stress-reduction techniques such as deep breathing exercises, meditation, yoga, or progressive muscle relaxation. High stress levels can trigger migraines; managing them is essential.

2. Home Remedies:

 - **Herbal Teas:** Prepare caffeine-free teas using ingredients like chamomile, peppermint, or feverfew. These can promote relaxation and potentially help prevent migraines.
 - **Cold/Heat Therapy:** Apply a cold compress to the forehead or upper neck region for 15-20 minutes at a time, repeating every hour if needed. Alternatively, use a warm compress for 10-15 minutes after the attack has passed to alleviate tension.
 - **Massage:** Gently massage the temples, neck, shoulders, or back with a mild oil (e.g., lavender or grapeseed) to ease tension and soothe muscles.

Use via API · Built with Gradio · Settings

KNOWN ISSUES:

- **Medical Accuracy**
- The AI model may provide incomplete, outdated, or inaccurate medical information.
- Suggestions generated are **not a substitute for professional medical advice**.
- The system should **not be used for emergencies** or critical medical decision-making.
- **Bias and Generalization**
- The model might produce **biased or culturally insensitive recommendations** since it was trained on general datasets.
- Treatment plans may not account for region-specific healthcare practices, availability of medicines, or local regulations.
- **Data Privacy**
- User-entered symptoms and medical history are not encrypted or stored securely unless additional data protection measures are implemented.
- There is currently **no HIPAA/GDPR compliance** in this version.
- **Model Limitations**
- The Granite 3.2-2B-Instruct model is relatively small and may lack **deep medical knowledge** compared to specialized healthcare models.
- Outputs may sometimes be **too generic, repetitive, or verbose**.
- Occasionally, the model may fail to follow instructions like focusing only on home remedies or medication guidelines.
- **Performance Issues**
- On CPU-only systems, **response time may be slow** and less efficient.
- High GPU memory usage can limit scalability on smaller devices.
- **Prompt Sensitivity**
- Small changes in user input wording may drastically alter results.
- The model sometimes **hallucinates conditions or treatments** that are not medically valid.
- **Ethical & Legal Concerns**

- This assistant should not be deployed for **commercial medical practice** without regulatory approval.
- Legal liability exists if users misunderstand outputs as official prescriptions.
- **UI/UX Limitations**
- Only text-based interaction is available; no voice or image-based diagnosis.
- Long responses may overflow the textbox or require manual scrolling.

FUTURE ENHANCEMENT :

1. **Integration with Verified Medical Databases**
   a. Connect the system with trusted sources such as **WHO, CDC, PubMed, or government health portals** to ensure accurate, up-to-date medical guidance.
   b. Regular updates to reflect the latest research, treatment guidelines, and approved medications.
2. **Multi-Modal Input**
   a. Expand input methods beyond text to include **voice-based symptom reporting** and **image uploads** (e.g., rashes, medical reports, lab results).
   b. Incorporate Natural Language Speech-to-Text for users who prefer conversational interaction.
3. **Personalized Health Monitoring**
   a. Enable integration with **wearable devices** (smartwatches, fitness trackers) to analyze real-time health data like heart rate, blood oxygen, and sleep patterns.
   b. Provide more **dynamic treatment and lifestyle recommendations** based on live health metrics.
4. **Localization & Multi-Language Support**
   a. Extend support to multiple languages (e.g., Hindi, Tamil, French, Spanish) to make healthcare information more accessible.
   b. Adapt recommendations based on local healthcare practices, diet, and available medicines.
5. **Emergency Handling & Alerts**

   a. Add a feature to **detect emergency symptoms** (e.g., chest pain, difficulty breathing) and immediately advise contacting emergency services.
   b. Option to connect users directly to **telemedicine platforms** or nearby hospitals.

6. **Enhanced Security & Privacy**
   a. Implement **end-to-end encryption** for user data.
   b. Ensure compliance with **HIPAA, GDPR, and other healthcare privacy standards** for safe handling of sensitive medical information.
7. **Adaptive Learning & Personalization**
   a. Build a feedback loop where user ratings improve the AI's recommendations.
   b. Store non-sensitive health history to offer **personalized and evolving treatment suggestions** over time.
8. **Expanded Medical Use Cases**
   a. Add specialized modules for **mental health screening, diet/nutrition planning, chronic disease management (diabetes, hypertension, asthma)**.
   b. Develop predictive health risk models for preventive care.
9. **Scalability & Cloud Deployment**
   a. Deploy the application on cloud platforms (AWS, Azure, IBM Cloud) for **global accessibility**.
   b. Enable support for **mobile applications (Android/iOS)** with offline capability for rural areas with limited internet.
10. **Collaboration with Medical Experts**

- Partner with doctors and medical institutions to **validate treatment suggestions**.
- Create a **hybrid AI + human-in-the-loop model** where doctors review AI-generated plans. ....