HEALTH AI ASSISTANT

PROJECT DOCUMENTATION

INTRODUCTION:

PROJECT TITLE : HEALTH AI: INTELLIGENT HEALTHCARE ASSISTANT

* TEAM ID: NM2025TMID02718

- **Team Leader:** Sakthidharan. U (DOCUMENTATION) (NM ID: 501030A8B5B24A92A14E335041ED167)
- Team Member: Saranraj. M (VOICE OVER& CREATER)
- (NM ID: A229C1DB4188D3B817848A9401432D15)
- Team Member: Kishore. R (VEDIO ANALYSER)
 (NM ID: 784BCD165FC36FAE9844F6D2FFC7E8B)
- **Team Member:** Karthir Kanth. Y. K (UPLOADER)
- (NM ID: 3D1F967851E67E3B415459F6D24FD96F)
- **Team Member:** Sam. M (EDITOR)
- (NM ID:98DEE501C7852B79422428587ED601A4)

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:

- app.py \rightarrow Main application file.
- requirements.txt \rightarrow 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):
 - o Input: Symptom text (e.g., "fever, headache, cough").
 - o Output: Possible conditions and general recommendations.
- treatment_plan(condition, age, gender, history):
 - o 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:

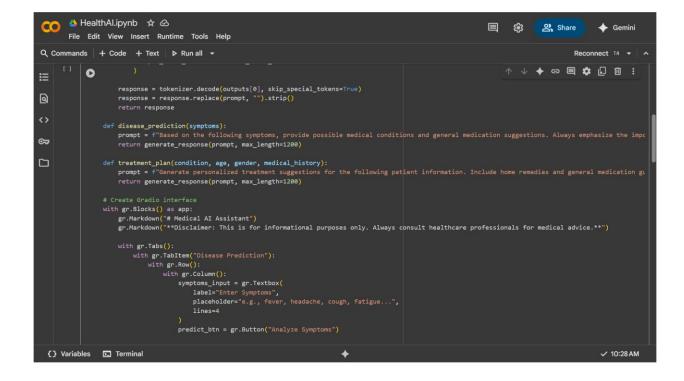
- Unit Testing: Functions like disease_prediction and treatment_plan were tested with sample inputs.
- **Integration Testing:** Verified that the UI, backend, and AI model worked seamlessly.
- **User Testing:** Simulated real-world inputs to evaluate usability and accuracy.

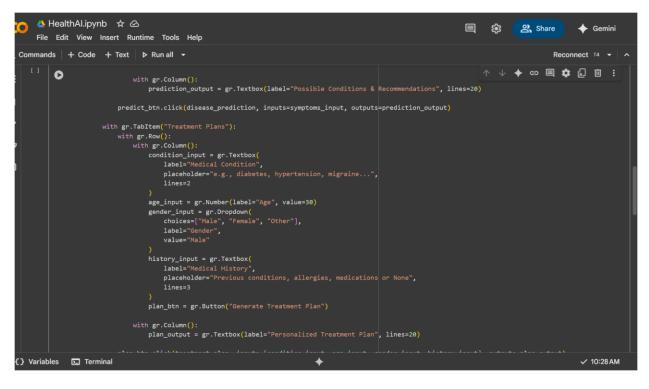
Testing confirmed that the system produced meaningful and consistent outputs, reinforcing its reliability as an academic project.

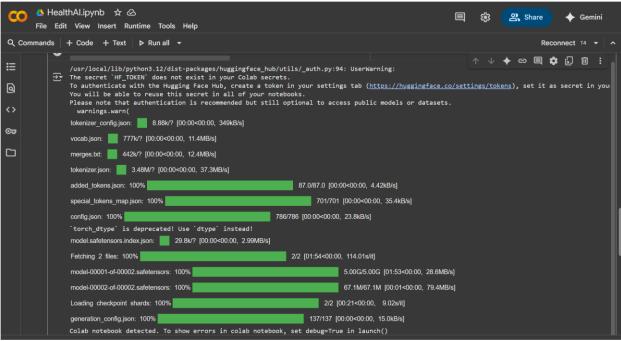
SCREEN SHOTS:

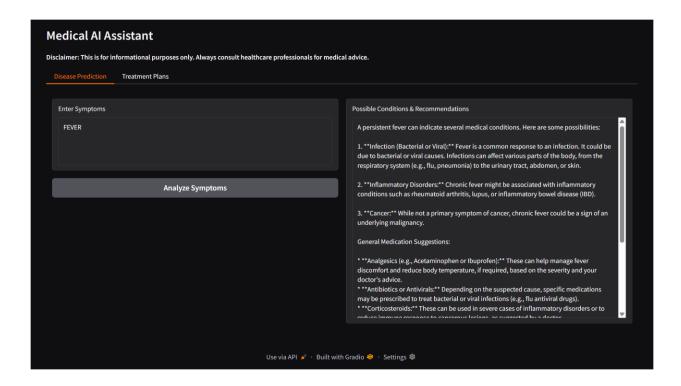
The Code of the program in google colab:

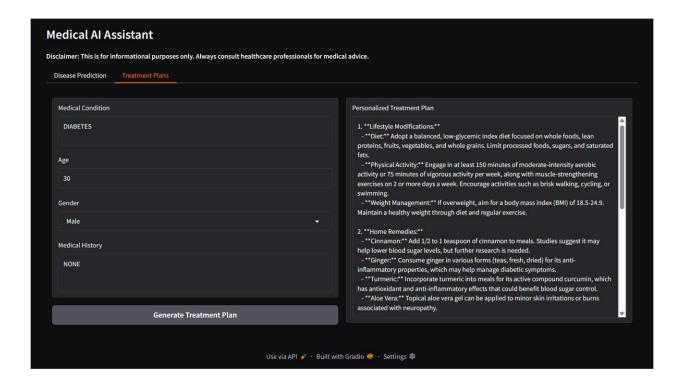
```
↑ ↓ ♦ ⊖ 🗏 🛊 🖟 🔟 :
              0
                    from transformers import AutoTokenizer, AutoModelForCausalLM
Q
                    model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
                    model = AutoModelForCausalLM.from_pretrained(
⊙ಾ
                        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32, device_map="auto" if torch.cuda.is_available() else None
def generate_response(prompt, max_length=1024):
                        inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
                        with torch.no_grad():
                            outputs = model.generate(
    **inputs,
                                  max_length=max_length,
                                  pad_token_id=tokenizer.eos_token_id
  {} Variables 🔼 Terminal
                                                                                                                                                                               ✓ 10:28 AM
```

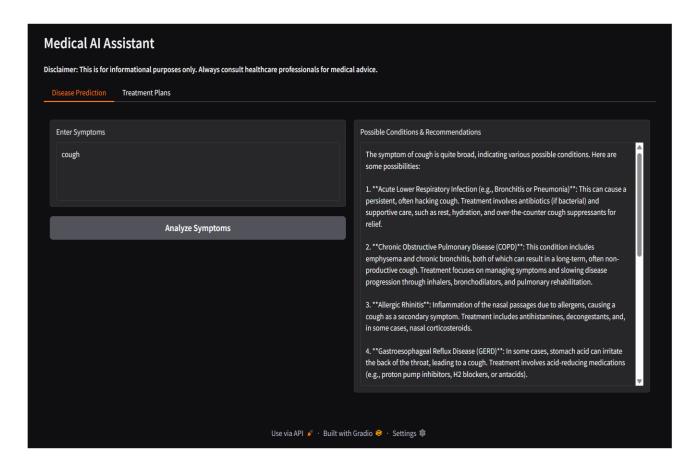


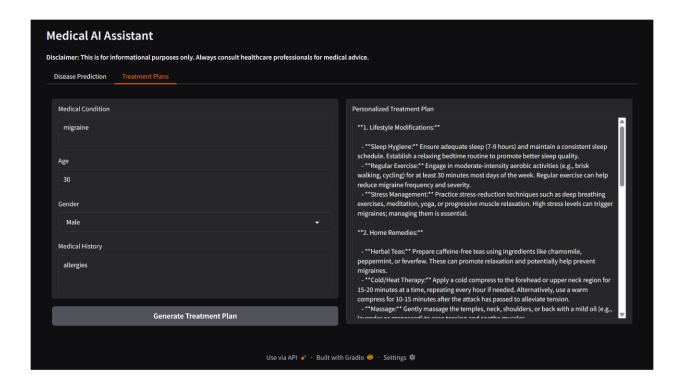












KNOWN ISSUES:

- 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.