# Project Title : Health AI Assistant

## Project Documentation

## Introduction :


- **Team Leader :** Nithiya shree M P
- **Team member :** Harshamitha J
- **Team member :** pooja M (2005)
- **Team member :** Jannathul firthous B


## Project Overview :

The Health AI Assistant is an AI-powered tool that assists in providing medical-related information based on user inputs. This tool is designed to help predict possible medical conditions based on user-reported symptoms and generate personalized treatment plans. However, it’s crucial to emphasize that this tool is **not a substitute for professional medical advice**, and users should always consult a healthcare provider for diagnosis and treatment.

The tool utilizes **IBM's Granite AI model** to analyze symptoms and conditions, providing responses to aid in healthcare decision-making, thereby supporting doctors, patients, and healthcare professionals.

## Objective :

The objective of this project is to create an intelligent assistant capable of:

1. Disease Prediction : Identifying potential medical conditions based on the symptoms provided.
2. Treatment Plan Generation : Creating personalized treatment plans, including medication guidelines and home remedies based on the patient's condition, age, gender, and medical history.

## Technologies Used :

1. **Programming Language :** Python

2. **Libraries/Frameworks :**

**PyTorch** : Deep learning library used to load and run the pre-trained model.
**Transformers** : A library by Hugging Face for natural language processing tasks, used to load the AI model and tokenizer.
**Gradio** : A Python library that enables quick creation of user interfaces for machine learning models.
**Torch** : A framework that accelerates tensor computations and model inference.

### How It Works :

### Model Loading :

The application uses a pre-trained language model
**"ibm-granite/granite-3.2-2b-instruct"
from the Hugging Face library. The model is loaded with either float16 or float32
precision based on the hardware (GPU or CPU) available.

```
python
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
)
```

## Generating Responses :

The AI generates responses by processing the user inputs (either symptoms or condition details) and providing relevant output, either possible medical conditions or treatment plans. The response is generated using **beam search** and **temperature sampling** for diverse output generation.

```python
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)
    return response
```

## Disease Prediction :

When users input their symptoms, the model analyzes the provided text and returns possible medical conditions along with general medication suggestions. An example input could be:

**Input :** "fever, headache, sore throat"
**Output :** "Possible conditions: flu, cold, strep throat. Recommended: Rest, hydration, over-the-counter fever medication. **Important: Consult a healthcare provider for a proper diagnosis.**"

## Treatment Plan Generation :

For users providing their medical condition, age, gender, and medical history, the system generates a personalized treatment plan. The treatment includes

general medication recommendations and home remedies based on the patient's data.

**Input :**

* Medical Condition: "Hypertension"
* Age: "45"
* Gender: "Male"
* Medical History: "None"

**Output :**

"A treatment plan for Hypertension could include lifestyle changes such as reduced salt intake, regular exercise, and medications like ACE inhibitors or calcium channel blockers."

## Gradio Interface :

The interface is developed using **Gradio**, which provides an easy-to-use web-based interface to interact with the AI model. The user can access two tabs:

1. **Disease Prediction :** For entering symptoms and receiving possible conditions and recommendations.

2. **Treatment Plans :** For providing details about a medical condition, patient demographics, and medical history to generate a treatment plan.

## Tab 1 : Disease Prediction

**Symptoms Input :** A user types symptoms like "headache, dizziness, nausea."

**Click Analyze Symptoms :** The AI processes the symptoms and returns a list of possible conditions like "migraine, dehydration, anxiety." It also suggests over-the-counter medications and emphasizes consulting a doctor for an accurate diagnosis.

### Tab 2: Treatment Plans

**Condition Input :** A user provides the condition "Asthma," their age, gender, and medical history.

**Click Generate Treatment Plan :** The AI generates a personalized treatment plan that includes medication suggestions (e.g., bronchodilators) and home remedies like using a humidifier or avoiding allergens.

### Gradio Interface Code :

The core of the user interface is created using Gradio. Here is an overview of how the interface is structured:

```python
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", 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)
```

---

## Deployment :

The app is deployed using **Gradio's share functionality**, which allows the app to be hosted and shared through a public URL. This allows users to interact with the model without needing to install any dependencies locally.

```python
app.launch(share=True)
```

## Ethical Considerations :

**Accuracy of Results :** The results provided by the AI are based on the data it has been trained on and are not guaranteed to be accurate. Always consult a healthcare professional for a diagnosis.

**Privacy:** No personal data is collected or stored beyond the scope of the user session, ensuring user privacy.

**Disclaimers :** The system emphasizes that the responses are for informational purposes only, encouraging users to seek proper medical advice.

**Future Improvements :**

1. Integration with Real-Time Medical Databases  Incorporating real-time health data or expert-curated databases to improve the accuracy of predictions and treatment plans.

2. **User Feedback :** Adding a feedback system where users can rate the accuracy of the suggestions and provide corrections to improve the model's performance.
3. **Extended User Demographics :** Expanding the user input options to include additional medical parameters such as weight, lifestyle habits, and more comprehensive medical history.

## Conclusion:

This project provides a simple yet powerful AI assistant to help people get informed about potential medical conditions and treatment plans based on their symptoms or conditions. It uses state-of-the-art language models to process natural language inputs and generate useful, context-aware recommendations. However, it is essential to remind users that the tool is for informational purposes only and cannot replace professional medical consultation