**Health AI: Intelligent Healthcare Assistant**

**Title**

Health AI : Intelligent Healthcare Assing IBM Granite Model

# Team Leadrer: RISHI P

Team member: ABINESH S

Team member: NAVEEN M

Team member:PRASATH S

# Abstract

The Health AI project leverages IBM Granite models from Hugging Face to provide an intelligent healthcare assistant. It integrates patient chat, disease prediction, and treatment planning into a user-friendly application. The project runs on Google Colab, ensuring accessibility, GPU-accelerated performance, and real-time responses. By combining Gradio, Python, and Granite, the assistant delivers secure and smart medical guidance.

# Objectives

* To design an AI-powered healthcare assistant.
* To enable patient interaction via chat for health-related queries.• To provide basic disease prediction and treatment suggestions.
* To deploy the model in an accessible platform (Google Colab).
* To integrate IBM Granite models for efficient and accurate performance.

# Problem Statement

Healthcare systems often face challenges in providing timely medical guidance due to lack of availability, high costs, and geographical barriers. Many individuals seek preliminary medical advice online, but reliability and accuracy remain issues. There is a need for an accessible, AI-driven assistant that can provide quick and reliable healthcare information.

# Methodology

* Model Selection: Choose an IBM Granite model (e.g., granite-3.2-2b-instruct) from Hugging Face.
* Development Environment: Use Google Colab with T4 GPU for development and execution.
* Framework: Implement Gradio for building the interactive user interface.
* Coding: Develop Python scripts to integrate Granite model with Gradio app.
* Deployment: Run and test the application in Colab, then upload project files to GitHub.

# Tools & Technologies

* IBM Granite Models (Hugging Face)
* Gradio Framework
* Python Programming
* Google Colab with T4 GPU
* Git & GitHub (for version control and hosting)

# Project Workflow

* Explore Naan Mudhalvan Smart Interz portal and enroll in Health AI project.
* Choose and configure IBM Granite model from Hugging Face.
* Run the application in Google Colab with necessary dependencies.
* Deploy and test the application using Gradio interface.
* Upload completed project to GitHub for submission and collaboration.

# Expected Outcomes

* Basic disease prediction and treatment suggestions.
* Deployment-ready project hosted in Google Colab.
* Repository in GitHub for project files and code.
* Foundation for A functional healthcare assistant capable of interacting with patients.
* expanding functionalities in the healthcare domain.

**Program Code**

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 for proper diagnosis.\n\nSymptoms: {symptoms}\n\nPossible conditions and recommendations:\n\n\*\*IMPORTANT: This is for informational purposes only. Please consult a healthcare professional for proper diagnosis and treatment.\*\*\n\nAnalysis:"

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: {condition}\nAge: {age}\nGender: {gender}\nMedical History: {medical\_history}\n\nPersonalized treatment plan including home remedies and medication guidelines:\n\n\*\*IMPORTANT: This is for informational purposes only. Please consult a healthcare professional for proper treatment.\*\*\n\nTreatment Plan:"

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)

Output

A screenshot of a computer

AI-generated content may be incorrect.