# Health AI: Intelligent Healthcare Assistant

# Project

Health AI : Intelligent Healthcare Assistant

# 1. Introduction

• Project title : Medical AI Assistant

• Team leader : Madhusri K H

• Team member : Madhumitha S

• Team member : Monica A

• Team member : Mithraa M J

• Team member : Mahalakshmi S

## 2. Project Overview

• Purpose :

The purpose of the Medical AI Assistant is to provide users with an AI-powered tool that can help analyze symptoms, suggest possible conditions, and generate general treatment plans. By leveraging IBM Granite LLM models through Hugging Face Transformers, this assistant is designed to enhance health awareness and provide guidance in an easy-to-use conversational interface. It is intended only for informational purposes and emphasizes the importance of consulting qualified healthcare professionals.

• Features:

Conversational Interface  
 Key Point: Natural language interaction  
 Functionality: Users can describe symptoms or conditions in plain language and receive AI-generated suggestions.

Disease Prediction  
 Key Point: Symptom-based analysis  
 Functionality: Provides possible conditions based on entered symptoms.

Treatment Plan Generator  
 Key Point: Personalized guidance  
 Functionality: Generates general treatment plans with home remedies and medication guidelines.

Disclaimer Integration  
 Key Point: Safety-first approach  
 Functionality: All outputs emphasize consulting a healthcare professional.

Gradio Interface  
 Key Point: User-friendly dashboard  
 Functionality: Provides an intuitive interface with tabs for disease prediction and treatment plans.

## 3. Architecture

Frontend (Gradio):  
  
The frontend is built with Gradio, providing an interactive web UI with tabs for Disease Prediction and Treatment Plan generation. Users can input symptoms, conditions, age, gender, and medical history, and receive outputs in real time.

Backend (Transformers + PyTorch):  
  
The backend uses Hugging Face Transformers and PyTorch for LLM inference. The IBM Granite instruct model is used for natural language understanding and generation. Prompt engineering ensures safety and clarity in medical guidance.

LLM Integration (IBM Granite):  
  
Granite LLM models are integrated to provide high-quality language generation, enabling medical symptom analysis and treatment recommendations.

Deployment:  
  
The app can be launched locally with Gradio’s sharing feature or deployed on cloud platforms for broader accessibility.

## 4. Setup Instructions

Prerequisites:  
o Python 3.9 or later  
o pip and virtual environment tools  
o PyTorch (with CUDA if GPU available)  
o Hugging Face Transformers  
o Gradio library

Installation Process:  
o Clone the repository  
o Install dependencies using pip install -r requirements.txt  
o Run the application script  
o Launch the Gradio interface in a browser

## 5. Folder Structure

app.py – Main application script containing Gradio Blocks implementation  
requirements.txt – List of dependencies  
models/ – Folder containing model integration logic if customized  
docs/ – Documentation and project reports

## 6. Running the Application

➢ Run the Python script to start the Gradio app.  
➢ Access the interface through the provided localhost or public share URL.  
➢ Use the 'Disease Prediction' tab to analyze symptoms.  
➢ Use the 'Treatment Plans' tab to generate general treatment suggestions.  
➢ Always follow disclaimer messages to consult a healthcare professional.

## 7. API Documentation

This version is designed as a Gradio interface and does not expose REST APIs directly. Future enhancements may include FastAPI-based endpoints such as:  
POST /predict-disease – Analyze symptoms  
POST /generate-plan – Generate treatment plan

## 8. Authentication

This demo runs in an open environment. For production deployment, security features can include:  
• Token-based authentication (JWT)  
• Role-based access (admin, doctor, patient)  
• Encrypted data transmission

## 9. User Interface

The Gradio interface is minimalist and functional with:  
• Tabs for Disease Prediction and Treatment Plans  
• Input boxes for symptoms, conditions, age, gender, and history  
• Output boxes displaying AI-generated guidance  
• Clear disclaimer messages to ensure safety

## 10. Testing

Testing was performed in multiple stages:  
• Unit Testing: Prompt and function outputs  
• Manual Testing: Symptom inputs and treatment outputs  
• Edge Case Handling: Empty inputs, long symptom lists, non-medical text