Health AI: Intelligent Healthcare Assistant

Project Documentation:

1. Introduction

Project Title: Health AI: Intelligent Healthcare Assistant

Team Members:

Member 1:S.Abinaya

Member 2:J.Sofiya Nusrath

Member 3:J.Shapharmhatee

Member 4:N.Vasundra devi

2. Project Overview

Purpose:

The purpose of Health AI is to provide smart, accessible, and secure healthcare assistance using IBM’s Granite AI models. It delivers patient-centric services such as interactive chat, disease prediction, and personalized treatment plan recommendations. The system supports healthcare professionals by offering insights and suggestions while empowering patients with clear, easy-to-understand medical information.

Features:

Patient Chat

Key Point: Conversational interface for healthcare queries

Functionality: Patients can ask health-related questions in natural language and receive AI-generated responses.

Disease Prediction

Key Point: AI-driven risk assessment

Functionality: Uses patient data to predict possible health conditions.

Treatment Plans

Key Point: Personalized care recommendations

Functionality: Suggests treatment steps or guidelines based on symptoms and predictions.

Integration with IBM Granite Models

Key Point: Powerful, scalable AI backend

Functionality: Uses Granite models from Hugging Face for natural language understanding and generation.

Deployment in Google Colab

Key Point: Fast, accessible environment

Functionality: Runs the application on GPU-enabled notebooks for quick prototyping.

Gradio UI

Key Point: User-friendly healthcare dashboard

Functionality: Provides a simple, interactive interface for patients and doctors.

3. Architecture

Frontend (Gradio):

Interactive UI built using Gradio to host patient chat, prediction forms, and treatment suggestions. Users can run the interface locally or in Google Colab.

Backend (IBM Granite Models):

Granite models (e.g., granite-3.2-2b-instruct) from Hugging Face power natural language processing for healthcare queries.

Workflow Integration:

Data flows from the Gradio UI to the backend model, which processes input and returns health insights.

4. Setup Instructions

Prerequisites:

Python 3.9 or later

Gradio framework installed (pip install gradio)

Transformers & Torch libraries

IBM Granite model from Hugging Face

Google Colab T4 GPU (optional for acceleration)

GitHub account for code hosting

Installation Process:

1. Open Google Colab and create a new notebook named “Health AI”.

2. Change runtime to T4 GPU under Runtime > Change runtime type.

3. Run:

```
!pip install transformers torch gradio -q
```

4. Load the IBM Granite model from Hugging Face.

5. Run the application code and click the generated URL to open the Gradio app.

5. Folder Structure:

app/ – Core logic for AI healthcare assistance.

ui/ – Gradio-based UI components.

health_ai.py – Main entry script for running the app in Colab or locally.

model_integration.py – Handles communication with IBM Granite models.

prediction_module.py – Disease prediction logic.

treatment_planner.py – Suggests treatment steps.

6. Running the Application:

Launch Google Colab and open your Health AI notebook.

Install prerequisites.

Run the cells to start the backend and Gradio frontend.

Click the link to open the live Gradio app.

Interact with the app: ask questions, upload data, and view predictions.

7. API Documentation:

The Gradio app runs locally, but backend endpoints can be exposed using FastAPI if needed:

POST /chat/ask – Accepts patient query and responds with AI-generated answer.

POST /upload-data – Uploads patient data for prediction.

GET /predict-disease – Returns disease risk predictions.

GET /treatment-plan – Provides AI-generated treatment suggestions.

8. Authentication:

For demo purposes, the app runs openly in Google Colab. For production:

Token-based authentication for API calls.

OAuth2 integration with healthcare provider credentials.

Role-based access for patients, doctors, and admins.

9. User Interface:

The interface is designed for both patients and healthcare staff. It includes:

Sidebar with navigation between chat, prediction, and treatment modules.

Real-time output display.

Easy export of reports and results.

10. Testing:

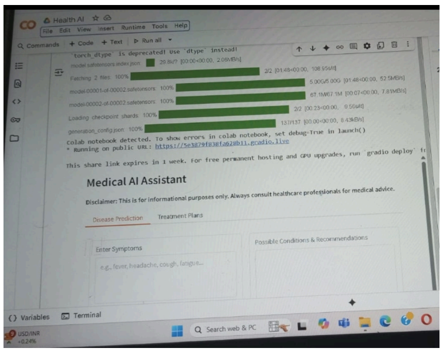
Unit Testing: Validate disease prediction and treatment plan modules.

Integration Testing: Test Gradio UI with backend models.

Manual Testing: Ensure smooth workflow in Google Colab.

Edge Cases: Handle large datasets or missing patient information.

11. Screenshots:



```
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.06MB/s]
Fetching 2 files: 100%  2/2 [01:48<00:00, 108.95s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [01:48<00:00, 52.5MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:07<00:00, 7.81MB/s]
Loading checkpoint shards: 100%  2/2 [00:23<00:00, 9.56s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 8.43kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://5e3879f838fa628b11.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` fr
```

Medical AI Assistant

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

12. Known Issues:

Limited medical dataset integration.

Requires stable internet for Hugging Face model loading.

13. Future Enhancements

Integrate with electronic health records (EHRs).

Add multilingual support for patient interaction.

Include voice-based queries.

Implement secure deployment on cloud platforms.