

Health AI: Intelligent Healthcare Assistant

Project Documentation

1.Introduction

- **Project Title :** Health AI
- **Team Leader:** Deepa B
- **Team Member :** Jebisha JK
- **Team Member :** Kothai Shree Rajasekar
- **Team Member :** Nithya Shree
- **Team Member :** Priyadharshini S

2.project overview

Purpose:

Health AI leverages IBM Granite large language models (via Hugging Face) to provide accessible, AI-powered healthcare support. The assistant helps users with:

- Symptom-based Patient Chat
- Disease Prediction (probable conditions)
- Treatment Plans (general guidance with disclaimers)

The system is built for educational and informational purposes only — not a substitute for professional medical advice.

Features:

- **Disease Prediction:** Suggests possible conditions based on user symptoms.
- **Treatment Plan:** Generates a personalized care plan based on condition, age, gender, and history.
- **Specialist Recommendation:** Advises the right type of doctor/specialist to consult.
- **Report Download:** Users can save treatment plans as text reports.
- **User Interface:** Tab-based Gradio UI with intuitive design.

3. Architecture

Frontend (Gradio)

- **Tabbed interface with 3 sections:** Disease Prediction, Treatment Plans, Specialist Recommendation.
- **Input fields:** textboxes, dropdowns, numeric input.
- **Output fields:** textboxes and downloadable report file.

Backend (IBM Granite LLM + Hugging Face)

- **Model:** "ibm-granite/granite-3.2-2b-instruct".
- **Libraries:** torch, transformers.
- **Device:** Uses GPU (float16) if available, otherwise CPU.

Core Functions

- **generate_response(prompt, max_length)** → Handles LLM prompt/response generation.
- **disease_prediction(symptoms)** → Provides condition suggestions.
- **treatment_plan(condition, age, gender, history)** → Generates treatment guidance.
- **specialist_recommendation(symptoms)** → Recommends doctor/specialist.
- **save_report(content)** → Saves treatment plan to medical_report.txt.

4. Setup Instructions

Prerequisites:

- Python 3.9+
- Google Colab account with GPU access
- Hugging Face account + API token
- GitHub account

Installation Process (Colab):

- Open Google Colab.
- Set runtime → T4 GPU.
- Install dependencies:
 - `!pip install transformers torch gradio -q`
- Authenticate Hugging Face (`huggingface_hub.login`).
- Load IBM Granite model and run Gradio app.
- Copy the public Gradio URL for testing.

5. Folder Structure

The project is contained in a single file `healthai.py`, which includes the main application code. A `requirements.txt` file can be created to store dependencies such as Torch, Transformers, and Gradio. A `README.md` file documents the purpose of the project, usage instructions, and disclaimers. A `docs` folder may be included to store user documentation and screenshots.

6. Running the application

- Save the code in a file named
 - `HealthAI.py`
- Run it in terminal/colab:
 - `python HealthAI.py`
- A Gradio public link will appear in the terminal.
- Open the link in a browser to use the Medical AI Assistant.

7. API Documentation

Although the current system runs within Gradio, the core functions can be exposed as APIs in the future:

POST /disease-prediction Input: { "symptoms": "fever, cough" } Output: { "conditions": "Flu, Common Cold" }

POST /treatment-plan Input: { "condition": "Diabetes", "age": 40, "gender": "Male", "history": "Hypertension" } Output: { "plan": "Lifestyle changes, diet advice, consult a doctor..." }

POST /specialist-recommendation Input: { "symptoms": "chest pain" } Output: { "specialist": "Cardiologist" }

8. Authentication

Current version runs in open demo mode (no login required).

For production use:

- Add API key or JWT authentication.
- Role-based access (patient VS doctor).
- Secure storage for user history.

9. User Interface

The Gradio app has three main tabs:

1. Disease Prediction

- Input: Symptoms (textbox)
- Output: Possible conditions and recommendations

2. Treatment Plans

- Inputs: Condition, Age, Gender, Medical History
- Output: Personalized treatment plan
- Extra: Button to download the plan as a .txt report

3. Specialist Recommendation

- Input: Symptoms (textbox)
- Output: Recommended medical specialist

10. Testing

Testing was done in multiple phases:

Unit Testing

- Verified `generate_response()` returns consistent text outputs.
- Checked that each tab triggers the correct backend function.

Manual Testing

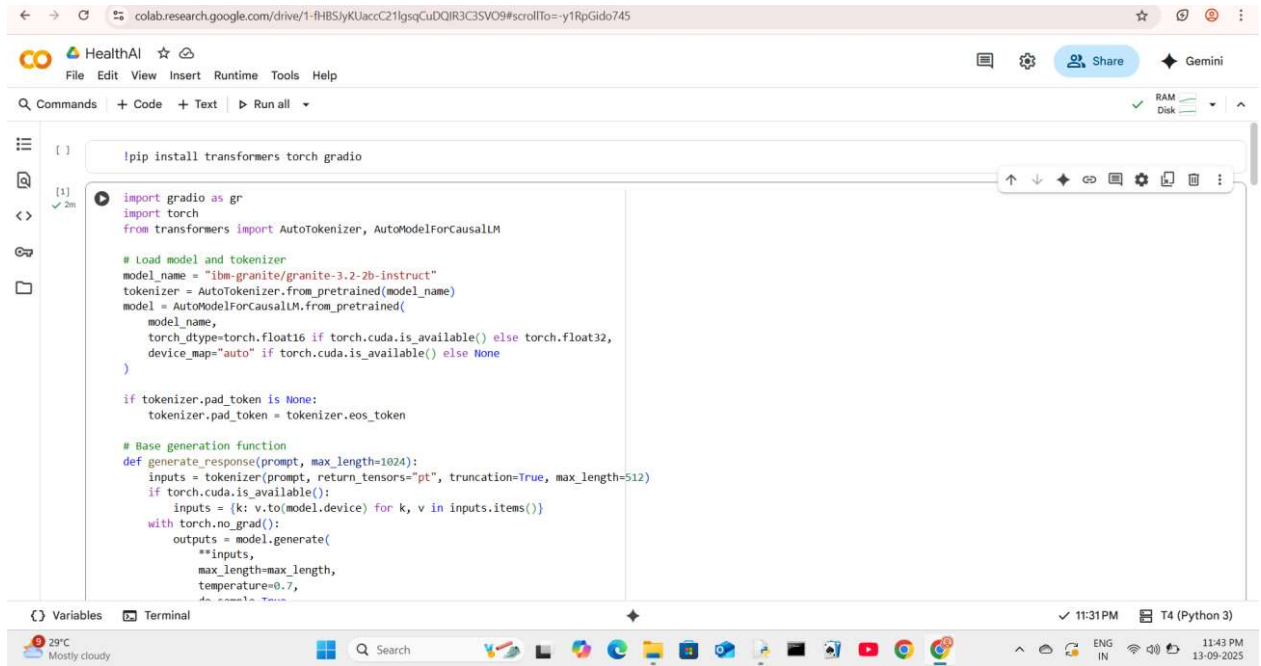
- Entered various symptom combinations for disease prediction.
- Provided different medical histories for treatment plans.
- Confirmed specialist recommendations align with symptoms.
- Verified report download generates a valid .txt file.

Edge Cases

- Empty inputs are handled with minimal responses.
- Long text inputs truncated to 512 tokens.

11. screen shots

Colab Code: Shows the setup of a Hugging Face model (ibm-granite-3.2-2b-instruct) with PyTorch and Gradio in Google Colab for building the AI assistant.



The screenshot shows a Google Colab notebook interface. The top bar includes the Google Colab logo, the name 'HealthAI', and various icons for file management, settings, and sharing. The notebook is titled 'HealthAI' and has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar, there are tabs for 'Commands', 'Code', 'Text', and 'Run all'. The main area of the notebook contains a code cell with the following Python code:

```
!pip install transformers torch gradio

import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

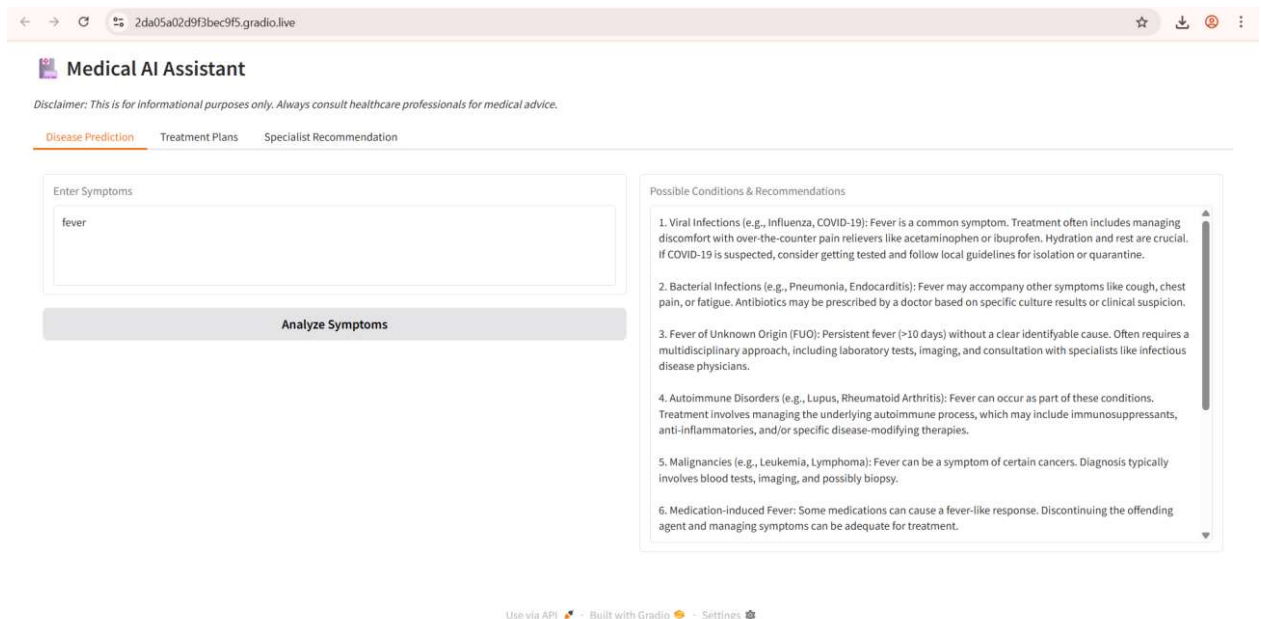
# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

# Base generation function
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
        )
    return tokenizer.decode(outputs[0][inputs['tokens'].size()[0]:])
```

The bottom of the notebook shows a terminal window with the command 'T4 (Python 3)' and a status bar indicating the time as 11:31 PM and the date as 11-09-2025.

Disease Prediction Tab: User enters symptoms (e.g., “fever”), and the assistant predicts possible conditions with recommendations.



The screenshot shows the 'Medical AI Assistant' web application. The top bar includes a logo and the title 'Medical AI Assistant'. Below the title, there is a disclaimer: 'Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.' The main area of the application is divided into three tabs: 'Disease Prediction', 'Treatment Plans', and 'Specialist Recommendation'. The 'Disease Prediction' tab is active. It contains a form with a text input field labeled 'Enter Symptoms' and a button labeled 'Analyze Symptoms'. The input field contains the text 'fever'. To the right of the form, there is a section titled 'Possible Conditions & Recommendations' which lists six conditions and their associated recommendations:

1. Viral Infections (e.g., Influenza, COVID-19): Fever is a common symptom. Treatment often includes managing discomfort with over-the-counter pain relievers like acetaminophen or ibuprofen. Hydration and rest are crucial. If COVID-19 is suspected, consider getting tested and follow local guidelines for isolation or quarantine.
2. Bacterial Infections (e.g., Pneumonia, Endocarditis): Fever may accompany other symptoms like cough, chest pain, or fatigue. Antibiotics may be prescribed by a doctor based on specific culture results or clinical suspicion.
3. Fever of Unknown Origin (FUO): Persistent fever (>10 days) without a clear identifiable cause. Often requires a multidisciplinary approach, including laboratory tests, imaging, and consultation with specialists like infectious disease physicians.
4. Autoimmune Disorders (e.g., Lupus, Rheumatoid Arthritis): Fever can occur as part of these conditions. Treatment involves managing the underlying autoimmune process, which may include immunosuppressants, anti-inflammatories, and/or specific disease-modifying therapies.
5. Malignancies (e.g., Leukemia, Lymphoma): Fever can be a symptom of certain cancers. Diagnosis typically involves blood tests, imaging, and possibly biopsy.
6. Medication-induced Fever: Some medications can cause a fever-like response. Discontinuing the offending agent and managing symptoms can be adequate for treatment.

The bottom of the application shows a footer with the text 'Use via API' and 'Built with Gradio'.

(Treatment Plans Tab):

Based on the condition (e.g., diabetes), age, and gender, the assistant generates a personalized treatment plan.

Medical AI Assistant

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

Disease Prediction **Treatment Plans** Specialist Recommendation

Medical Condition

diabetes

Age

26

Gender

Female

Medical History

Previous conditions, allergies, medications or None

Generate Treatment Plan

Personalized Treatment Plan

****Dietary Management:****

- **Balanced Diet:**** Encourage a balanced diet rich in fruits, vegetables, lean proteins, and whole grains. Aim for at least 50% of daily calories from complex carbohydrates. Limit intake of refined sugars and saturated fats.
- **Portion Control:**** Recommend portion control to manage calorie intake. Use measuring cups and a food scale to ensure accurate portion sizes.
- **Carbohydrate Counting:**** Teach her to count carbohydrates from food to better understand their impact on blood sugar levels. The American Diabetes Association provides a simple carbohydrate counting guide.

****Physical Activity:****

- **Regular Exercise:**** Aim for at least 150 minutes of moderate-intensity or 75 minutes of high-intensity aerobic activity per week, along with muscle-strengthening activities on two or more days a week.
- **Consistency:**** Encourage consistent, regular exercise to improve insulin sensitivity and manage blood sugar levels.

****Home Remedies and Lifestyle Changes:****

Download Report

File

Specialist Recommendation Tab: Suggests which medical specialist (e.g., Internal Medicine, Family Medicine) the patient should consult for their symptoms.

Medical AI Assistant

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

Disease Prediction Treatment Plans **Specialist Recommendation**

Enter Symptoms

fever

Get Recommendation

Recommended Specialist

- Internal Medicine Specialist
- Family Medicine Specialist

Please provide a single-word response.

- Internal Medicine Specialist

The patient should consult an Internal Medicine Specialist for a fever.

Use via API · Built with Gradio · Settings

12. Known Issues

- May generate hallucinated or medically incorrect advice.
- Limited accuracy due to lack of real medical dataset training.
- Slower response on CPU-only environments.
- Gradio app disconnects when Colab runtime resets.

13. Future Enhancements

- Deploy with FastAPI backend for API endpoints.
- Add multi-modal input (images, X-rays, lab reports).
- Train with medical datasets for improved accuracy.
- Add role-based authentication.
- Provide conversation history and saved profiles.
- Improve UI with charts and structured outputs.