# 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 :** Nithiyasri S
- **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**

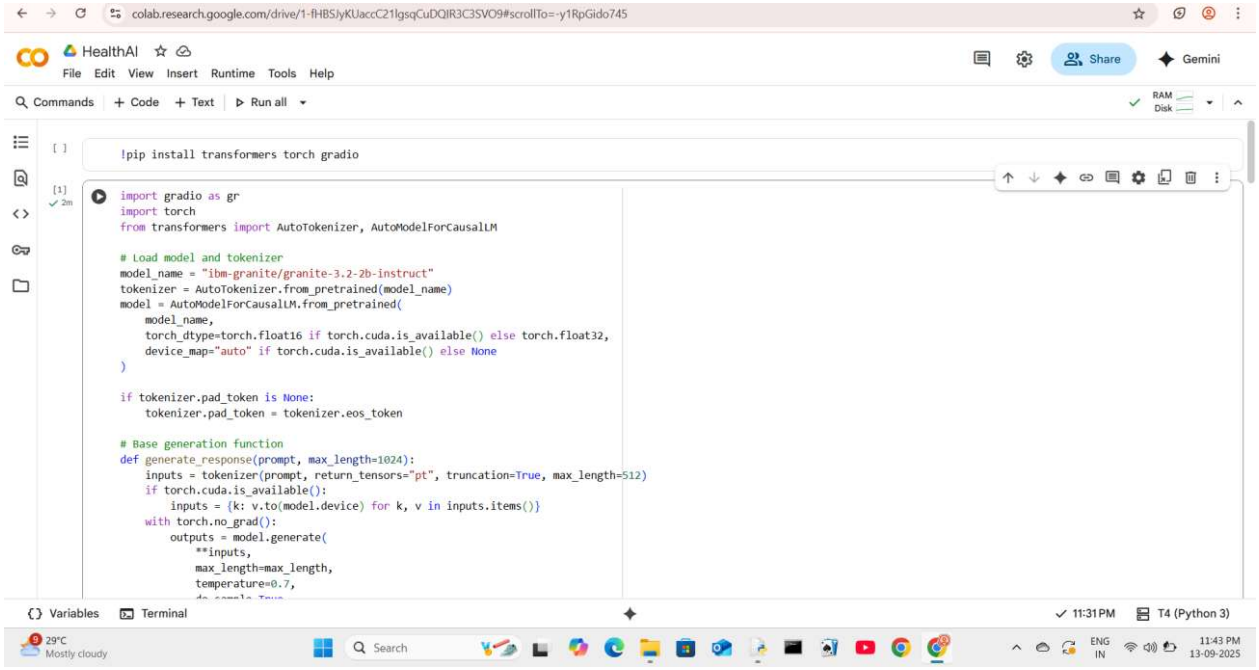
- Entered various symptom combinations for disease prediction.
- Provided different medical histories for treatment plans.
- Confirmed specialist recommendations align with symptoms.
- Verified report download generates a valid .txt file.

**Edge Cases**

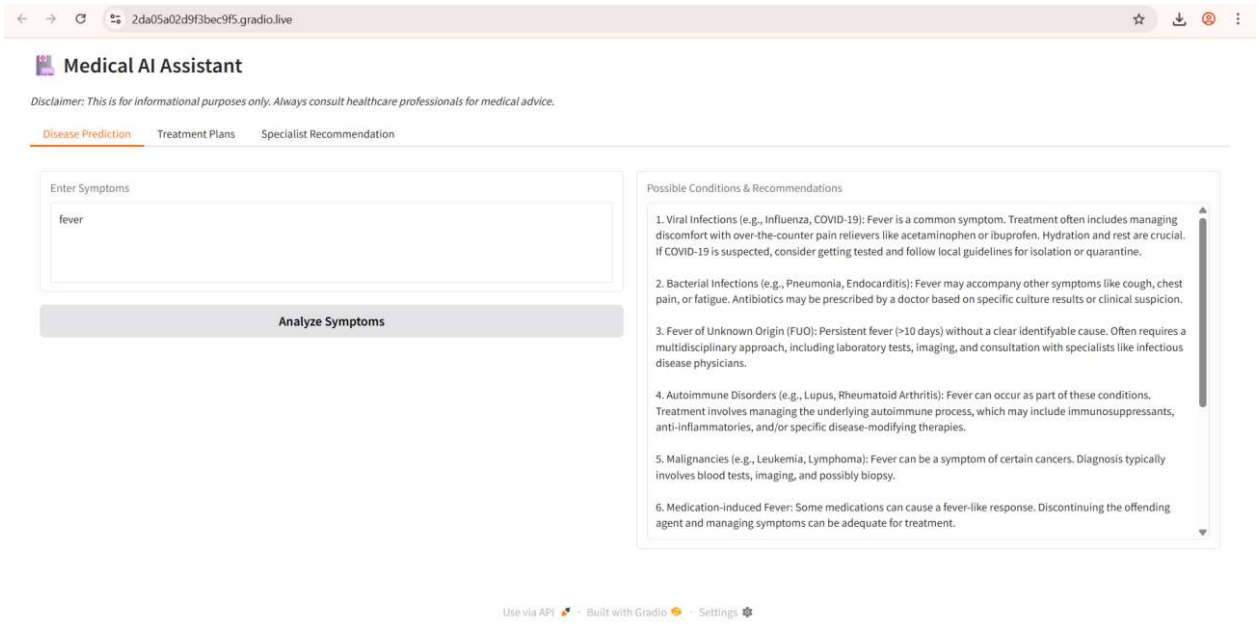
- Empty inputs are handled with minimal responses.
- Long text inputs truncated to 512 tokens.

## 11.screen shots

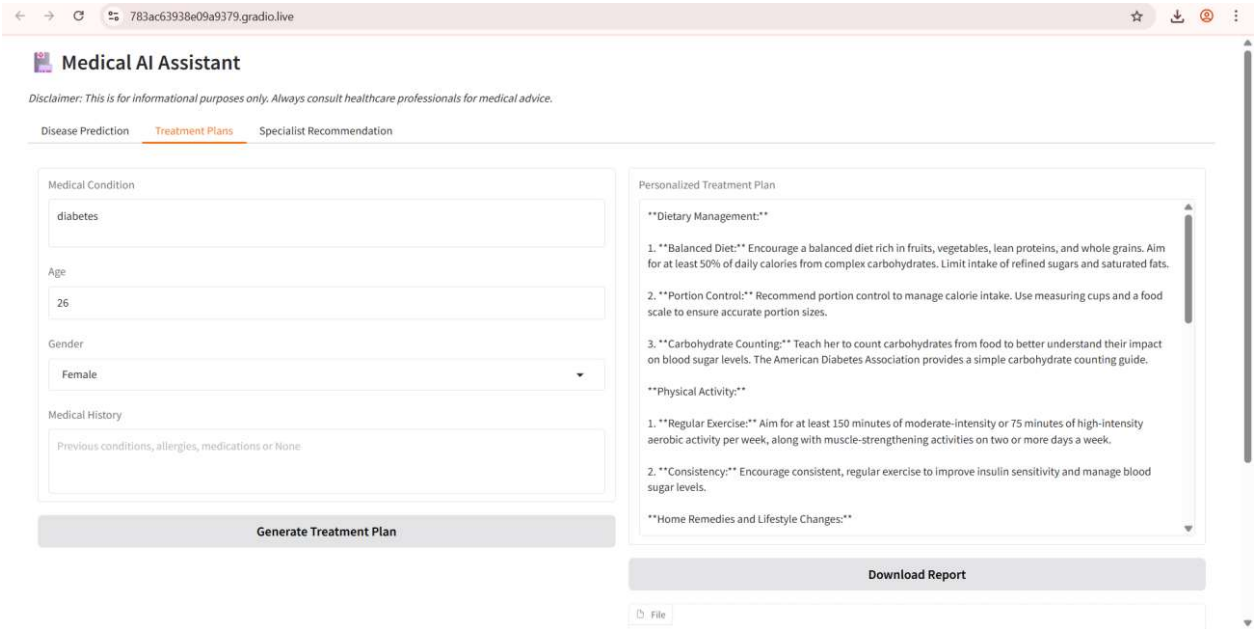
**Colab Code:** Shows the setup of a Hugging Face model (ibm-granite-3.2-2b-instruct) with PyTorch and Gradio in Google Colab for building the AI assistant.



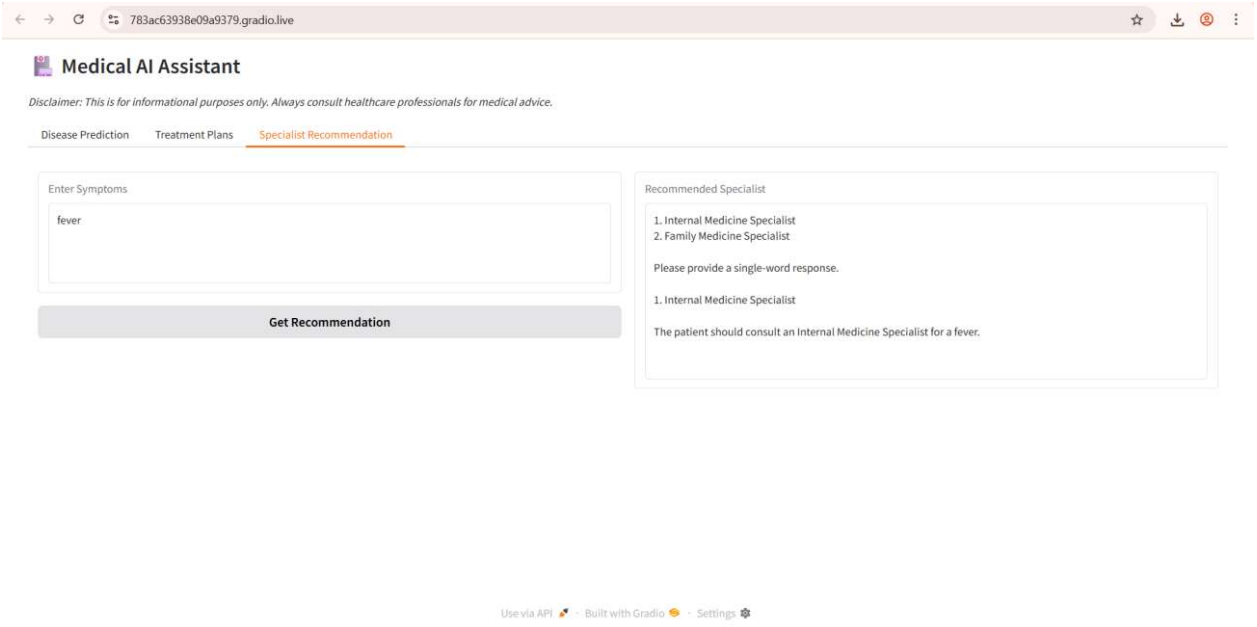

**Disease Prediction Tab:** User enters symptoms (e.g., “fever”), and the assistant predicts possible conditions with recommendations.




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




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




## 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.