GENERATIVE AI WITH IBM CLOUD

1. Introduction

- **Project Title:** HEALTH AI : INTELLIGENT HEALTH CARE
- **Team Members:** Gowri Sri Charan Teja Burra[Team Leader]
  Akella Lakshmi Aiswarya [Team member]
  B Tulasi sai [Team member]

2. Project Overview

HealthAI is an AI-powered healthcare assistant designed to provide intelligent, personalized, and accessible medical support. It leverages IBM's **Granite-3.3-2B-Instruct** generative AI model to assist users in understanding their health better through symptom analysis, treatment suggestions, health data insights, and medical Q&A — all in real-time via an intuitive interface.

Built using **Gradio** and **transformers**, HealthAI bridges the gap between users and healthcare information with an empathetic, informative, and responsive system.

Purpose:

The primary goal of HealthAI is to:

- **Empower users** with AI-driven medical insights
- **Simplify access** to preliminary healthcare guidance
- **Enable early detection** of potential health issues through symptom analysis
- **Support better decision-making** with treatment and health trend analytics
- Provide a **safe, user-friendly** AI assistant for general health questions

- **Key Features:**
- **Patient Chat**:

Users can ask general medical questions.

AI responds with clear, empathetic, and factual medical information.

- **Disease Prediction:**

Users input their symptoms.

The AI predicts possible medical conditions along with risk levels and guidance.

- **Treatment Plans:**

Users enter a diagnosed condition.AI provides a suggested treatment plan including medications, lifestyle changes, and testing advice.

- **Health Analytics:**

Users input their vital signs (e.g., heart rate, blood pressure).

The AI generates insights, highlights health trends, and recommends improvements.

-

3. System Architecture

FRONTEND:

- **Tool Used:** Gradio
- **Purpose:** Provides a clean, user-friendly web interface for patients to interact with the HealthAI system.
- **Functionality:**
  - Users can input natural language queries (symptoms, questions, conditions). ○ The interface displays AI-generated responses instantly. ○ Organized into intuitive tabs: Patient Chat, Disease Prediction, Treatment Plans, and Health Analytics.
  - No installation needed—runs on browser via a shared link.

BACKEND:

- **Platform:** Google Colab (Python runtime environment) **Languages & Libraries Used:**
  - `Transformers` – for loading IBM Granite model from Hugging Face. ○ `Torch` – for model inference using CPU/GPU. ○ `Gradio` – for UI components and server launch. ○ *(Optional: `LangChain`)* – for chaining queries and maintaining context in multistep conversations.
- **Functionality:**
  - Accepts frontend input and forms a structured prompt.
  - Sends prompt to AI model for processing. ○ Receives and post-processes model output.
  - Sends back the response to the user in appropriate format.
  - Handles additional features like Health Analytics using Python's data libraries.

AI MODEL:

- **Model Used:** IBM Granite-3.3-2B-Instruct
- **Hosted via:** Hugging Face Hub
- **Role in the System:**
- Core intelligence for the application.
- Capable of:
    - Understanding health-related queries. ○ Analyzing symptoms and predicting potential conditions.
    - Recommending personalized treatment plans.
    - Generating empathetic and informative responses for patient chats. ○ Interpreting patient health data trends.

DATABASE HANDLING:

- **Current Setup:**
    - No permanent backend database; currently runs in-memory using Python variables and Google Colab's temporary runtime. **For Production Use (Recommended):**
    - Use cloud-based databases like Firebase (NoSQL) or PostgreSQL/MySQL for:
        - Patient session history.
        - Repeated queries.
        - User-specific health data and trends.
        - Authentication and role-based access (doctor vs. patient).
    - Integrate basic analytics and user logging tools.

4. Setup Instructions

- ⚙ Setup Instructions – HealthAI

☑ 1. Prerequisites:

- Before setting up the project, ensure the following are available:
- ☑ Google Account – To access and run Google Colab.
- ☑ Python (pre-installed in Colab) – Python 3.x runtime.
- ☑ Stable Internet Connection – For model loading and Gradio interface.

- ☑ GitHub Account (optional) – If cloning from a repository.
- ☑ Hugging Face Account – To access the IBM Granite model (if authentication is required).

📦 2. Required Libraries:

Install the following Python libraries (within Google Colab):

!pip install transformers gradio langchain

These are used for:

transformers – to run the AI model.

gradio – to create the web-based chat interface.

langchain – for optional advanced workflows.

🛠 3. Model Setup (IBM Granite 3.3 2B Instruct):

Load the model from Hugging Face: •

```
from transformers import AutoModelForCausalLM, AutoTokenizer
model_name = "ibm-granite/granite-3b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_nam
```

💡 Note: You may need a Hugging Face token if the model access is restricted.

💻 4. Gradio UI Setup:

Create a simple interface:

```
import gradio as gr
def chat_function(user_input):
# Pass input to model and return response (dummy example)
```

- return "This is a response to: " + user_input
- interface = gr.Interface(fn=chat_function, inputs="text", outputs="text") • interface.launch()

🔄 5. Running the Project:

- Open the code in Google Colab.
- Run all cells in order.
- A Gradio link will appear to interact with the assistant.

📝 Optional – Save Output & Logs:

- You can add code to save chat logs or analytics in:
- Google Sheets / CSV
- Firebase or MongoDB (if needed in future updates)

5. **Folder Structure**

```
HealthAI/
├── client/                # Frontend (Gradio interface)
│   ├── ui.py              # Main Gradio UI code
│   └── assets/            # (Optional) Images, logos, etc.
│
├── server/                # Backend (model + logic)
```

```
│   ├── model_handler.py      # Load & run the AI model
│   ├── predictor.py          # Disease prediction logic
│   ├── treatment_plan.py     # Treatment recommendation logic
│   └── utils.py              # Helper functions
│
├── config/                   # (Optional) Configurations
│   └── settings.py           # Model paths, Hugging Face tokens, etc.
│
├── requirements.txt          # Python dependencies
├── README.md                 # Project overview and setup guide
└── main.ipynb                # Google Colab notebook (entry point)
```

- Explanation:

🗀 client/ – Frontend

Handles user interaction using Gradio.

ui.py sets up the chat interface and connects to backend logic.

🗀 server/ – Backend

Handles the core AI logic, including:

Loading the IBM Granite model (model_handler.py)

Processing symptoms and generating predictions (predictor.py) Creating

treatment plans (treatment_plan.py)

Utility functions like data cleaning or formatting (utils.py)

🗀 config/ – (Optional)

Store configuration files such as API keys, model settings, etc.

main.ipynb

Colab notebook that integrates all pieces.

Used to run the project end-to-end (most likely your current working environment).

6. Running the Application

Step 1: Set Up the Environment

You'll be using Google Colab as your main development and runtime platform.

☑ Open a new notebook in Google Colab

☑ Install required libraries:

!pip install gradio transformers langchain ⚙

Step 2: Backend (AI Model Logic)

The backend is responsible for processing user input and generating responses.

☑ Load the IBM Granite Model:

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_name = "ibm-granite/granite-3b-instruct" tokenizer =
AutoTokenizer.from_pretrained(model_name) model =
AutoModelForCausalLM.from_pretrained(model_name)
```

☑ Define the prediction function:

```
def generate_response(prompt):
    inputs = tokenizer(prompt, return_tensors="pt")    outputs =
model.generate(**inputs, max_new_tokens=100)    return
tokenizer.decode(outputs[0], skip_special_tokens=True)
```

💻 Step 3: Frontend (Gradio Interface)

This is the interface users interact with.

☑ Set up Gradio UI import gradio as gr def chat_interface(user_input): response = generate_response(user_input) return response

interface = gr.Interface(fn=chat_interface, inputs="text", outputs="text", title="HealthAI Assistant") interface.launch()

> ☑ When you run this cell, it will generate a link. Clicking it will open the HealthAI chat assistant in your browser.

Step 4: Integrate Full Workflow

If you have different files/modules (e.g., for treatment plans or analytics), import and connect them in your Gradio function.

🌐 Summary:

Component	Tool Used	Run Method
Frontend	Gradio	interface.launch() in Colab
Backend	Python (Colab) + IBM Granite Model	Load model and define logic
Hosting	Temporary via Gradio public link (auto-generated)	

7. API Documentation

• 1. Predict Disease

Endpoint:

POST /predict

Description: Accepts a list of symptoms and returns a predicted disease name.

Request Body

(JSON):

```
{
 "symptoms":  "fever, cough, fatigue"
}
```

Response (JSON):

```
{
    "predicted_disease": "Flu or Viral Infection"
}
```

2. Generate Treatment Plan

Endpoint:

POST /treatment

Description: Returns a treatment suggestion based on the disease name.

Request Body (JSON):

```
{
  "disease": "Flu"
}
```

Response (JSON):

```
{
  "treatment": "Rest, fluids, antiviral medication if
```

severe."
}

- 3. Chat with AI Assistant

Endpoint:

POST /chat
Description: General health-related query chat endpoint.

Request Body (JSON):

```
{
  "query": "What should I do for a cold?"
}
```

Response (JSON):

```
{
  "response": "Stay hydrated, take rest, consider paracetamol if fever occurs."
```

🛠 Backend Logic (Under the Hood)

Each endpoint would internally:

Tokenize input using

AutoTokenizer

Generate response
Using
AutoModelForCaus
alLM

Return the decoded
result to the frontend

8. Authentication Flow

1. Basic Authentication (for Prototypes)

Simple login with username & password stored in memory or a file.

Can be implemented directly in Gradio or Flask/FastAPI backend.

Not recommended for production use due to lack of security.

2. OAuth2.0 / Social Login

Allow users to log in using Google, GitHub, or Microsoft accounts.

Easy to integrate with services like Firebase, Auth0, or Google Identity.

Secure and user-friendly.

3. Token-Based Authentication (JWT)

After login, user receives a JWT (JSON Web Token).

Token is included in every request header to access protected APIs.

Commonly used in FastAPI/Flask for backend APIs. Example:

GET /predict
Authorization: Bearer <your_token_here>

🧰 Tools for Authentication:

Tool Use Case

Firebase Auth For email/password or Google login
FastAPI OAuth2 For backend-secured APIs
Auth0 Enterprise-level authentication
Gradio login Limited, basic auth via Gradio

🔐 Recommended Auth Flow for HealthAI:

1. User visits Gradio frontend
2. Login prompt appears (Google login or email/password)
3. On success:
Store login token/session
Allow access to the chat, prediction, and analytics
4. On backend:
Verify token with each request

Return responses only to authenticated users

9.Testing

Purpose of Testing:

To ensure that the HealthAI system:
Works correctly across all features (chat, prediction, treatment)
Handles real-world user inputs reliably
Provides accurate and meaningful medical responses
Maintains stability under different conditions (e.g., long queries, empty inputs)

Types of Testing Used or Recommended:

1. Unit Testing

What it tests: Individual functions like generate_response(), predict_disease(), get_treatment_plan()

Tools: unittest or pytest in Python

Example:

```
def test_prediction():   assert predict_disease("fever, cough") ==
"Flu or Viral Infection"
```

2. Integration Testing

What it tests: Connection between backend and frontend (Gradio + model)

Ensures the whole workflow runs smoothly: input → model → output

Example: Entering symptoms in Gradio UI gives expected response

3. Functional Testing

What it tests: That each feature works as expected from a user's perspective

Disease prediction

Treatment generation

Chat interactions

Manual or automated UI checks using test cases

4. Edge Case Testing

Handles unexpected inputs like:

Empty text

Very long queries Non-
health queries

Ensures the model responds gracefully (e.g., "Please provide more specific symptoms.")

5. Performance Testing (Optional)

Test goals:

Measure average response time (e.g., under 2–3 seconds)

Handle multiple user inputs without delay

Useful for analytics and future scaling

Testing Tools You Can Use:

Tool	Purpose
pytest	Unit + integration testing
Postman	API endpoint testing (if REST API used)
Gradio preview	Manual UI functional testing
Google Colab	Test in live dev environment

---

11. Known Issues

1. No Real Medical Validation

Issue: The AI predictions and treatment suggestions are based on language models, not verified by licensed medical professionals.

Impact: Risk of incorrect or unsafe advice if used without doctor consultation.

Recommendation: Add a disclaimer or integrate with certified medical databas

2. No User Authentication or Data Privacy

Issue: No login or user protection system is currently implemented.

Impact: Data could be misused or sessions could be publicly visible.

Recommendation: Add authentication (Google/Firebase) and encrypt sensitive info.

3. Limited Context Understanding

Issue: The AI model may not fully understand complex or long patient queries.

Impact: Inaccurate or vague responses to multi-symptom or ambiguous input.
Recommendation: Use context-aware memory chains (like Langchain) or fine-tune the model.

4. No Persistent Storage / Database

Issue: User sessions, predictions, and analytics are not stored long-term.

Impact: No way to track health history, usage trends, or patient progress.

Recommendation: Integrate with a cloud database (e.g., Firebase, MongoDB).

5. Performance Limitations in Colab

Issue: Google Colab has limited runtime, memory, and no production deployment.

Impact: Temporary access; session resets lose state and functionality.

Recommendation: Move backend to a cloud service (e.g., AWS, Render, Hugging Face Spaces).

6. Gradio UI Is Temporary

Issue: Gradio interfaces in Colab shut down after inactivity or tab closure.

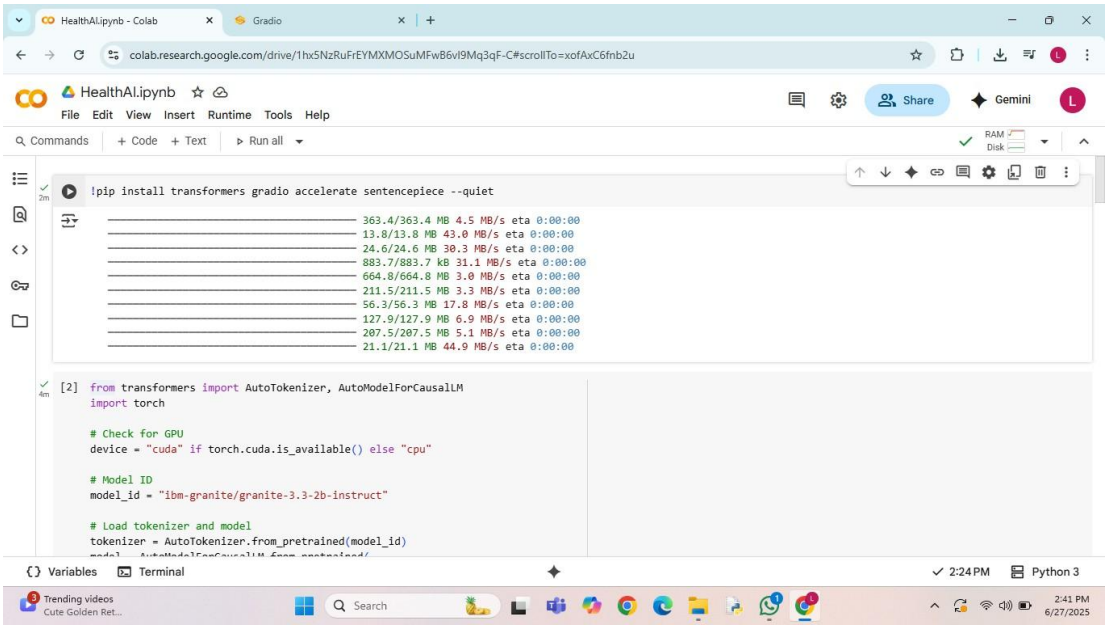
Impact: Not suitable for long-term or public access.

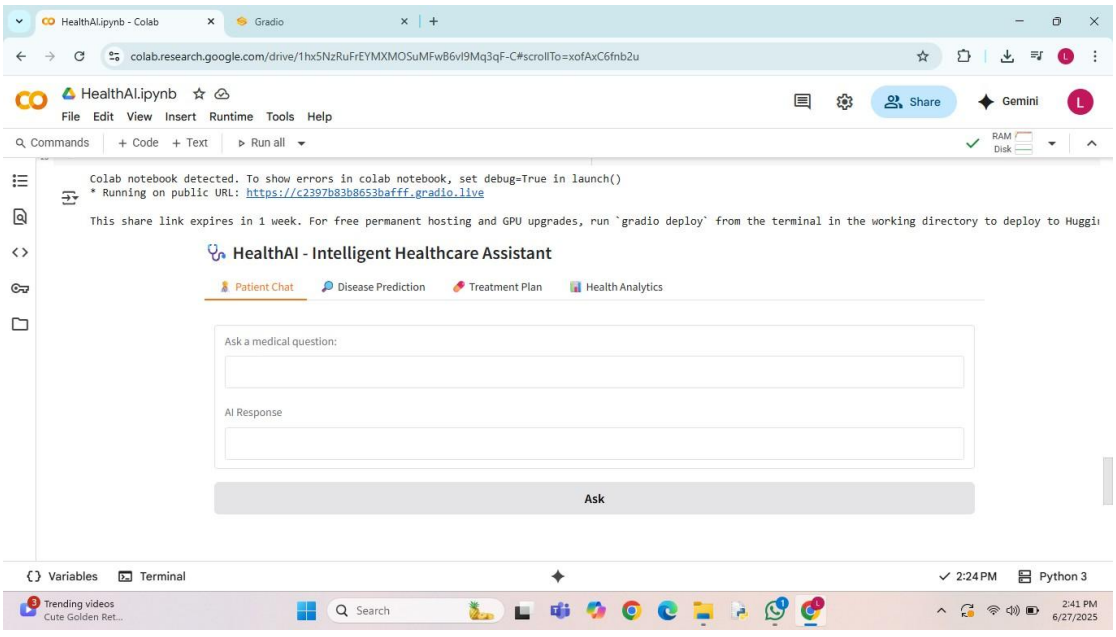
Recommendation: Deploy on Hugging Face Spaces or as a standalone web app.

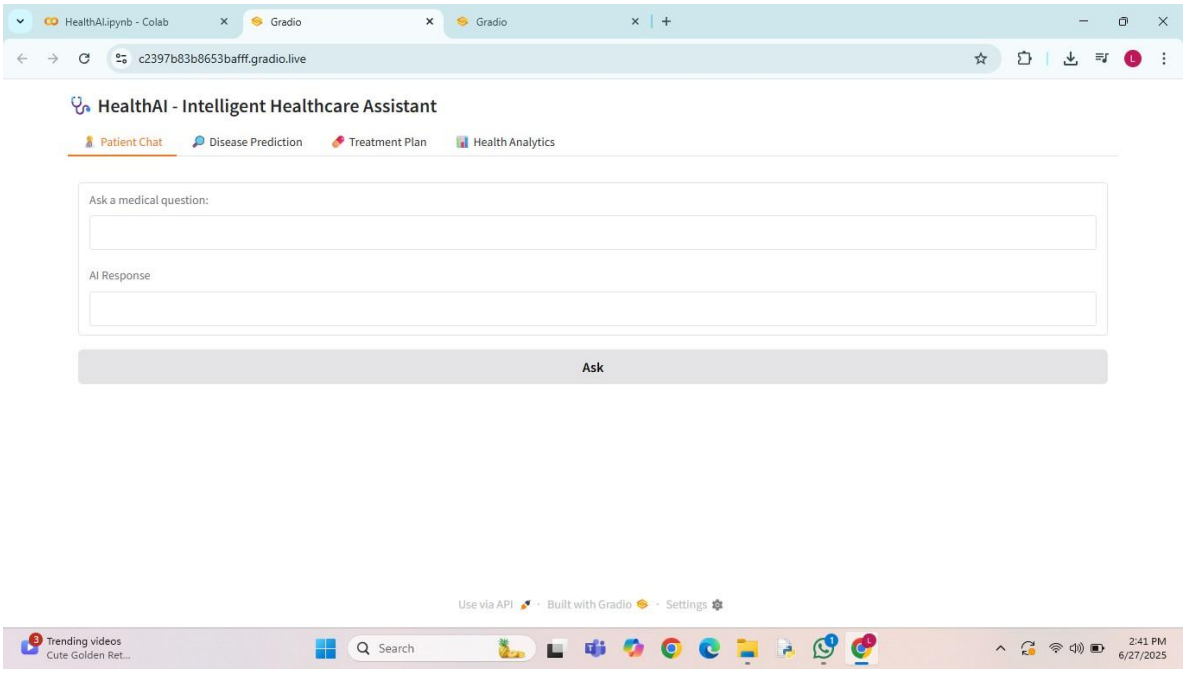
7. Model Size & Speed

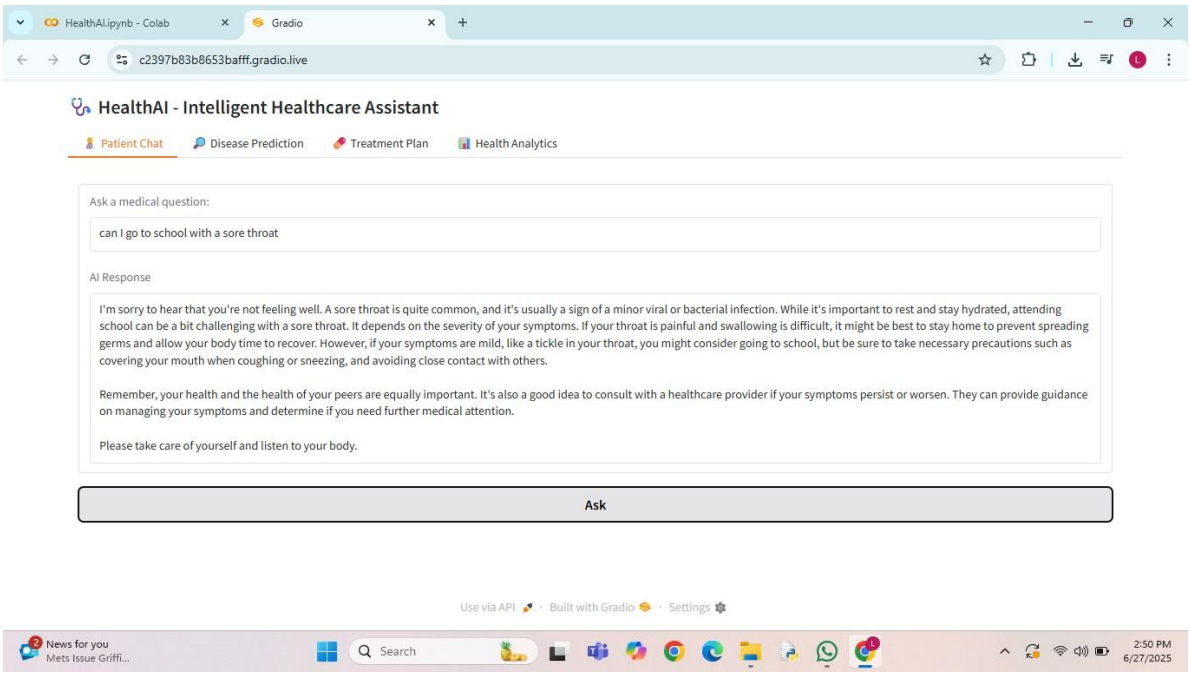
Issue: Large models like IBM Granite 3.3B can be slow to respond or load.
Impact: Increases wait time for users.

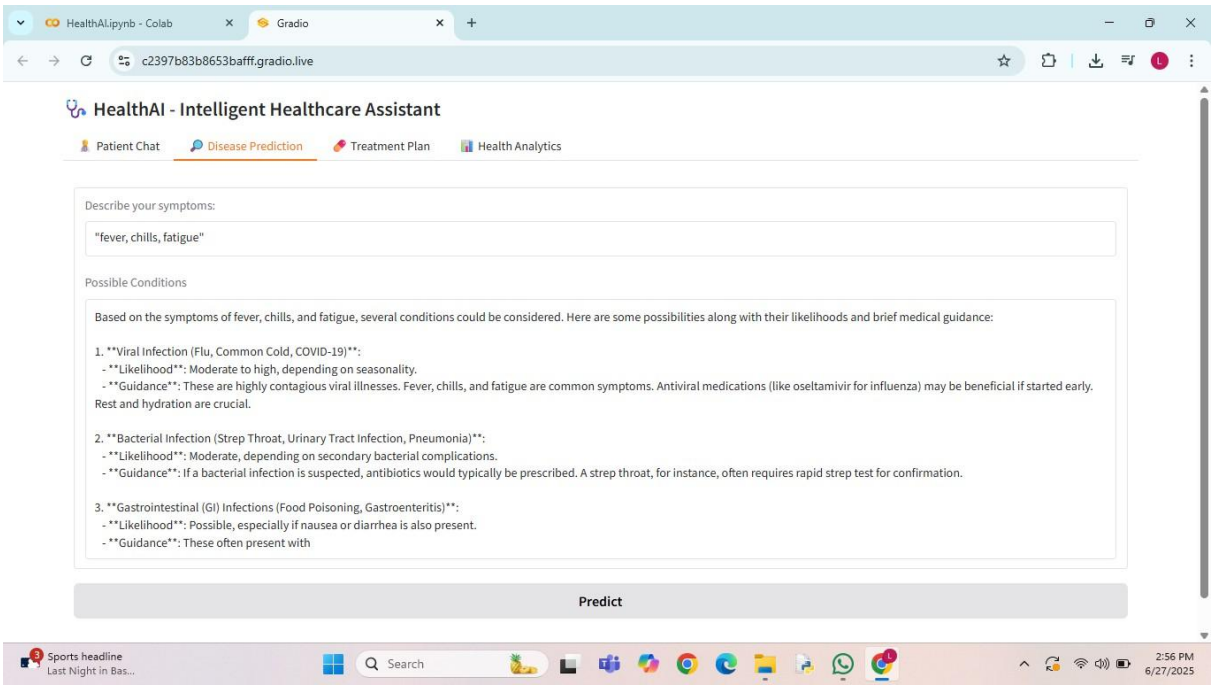
Recommendation: Use lighter models for faster performance or optimize prompts.

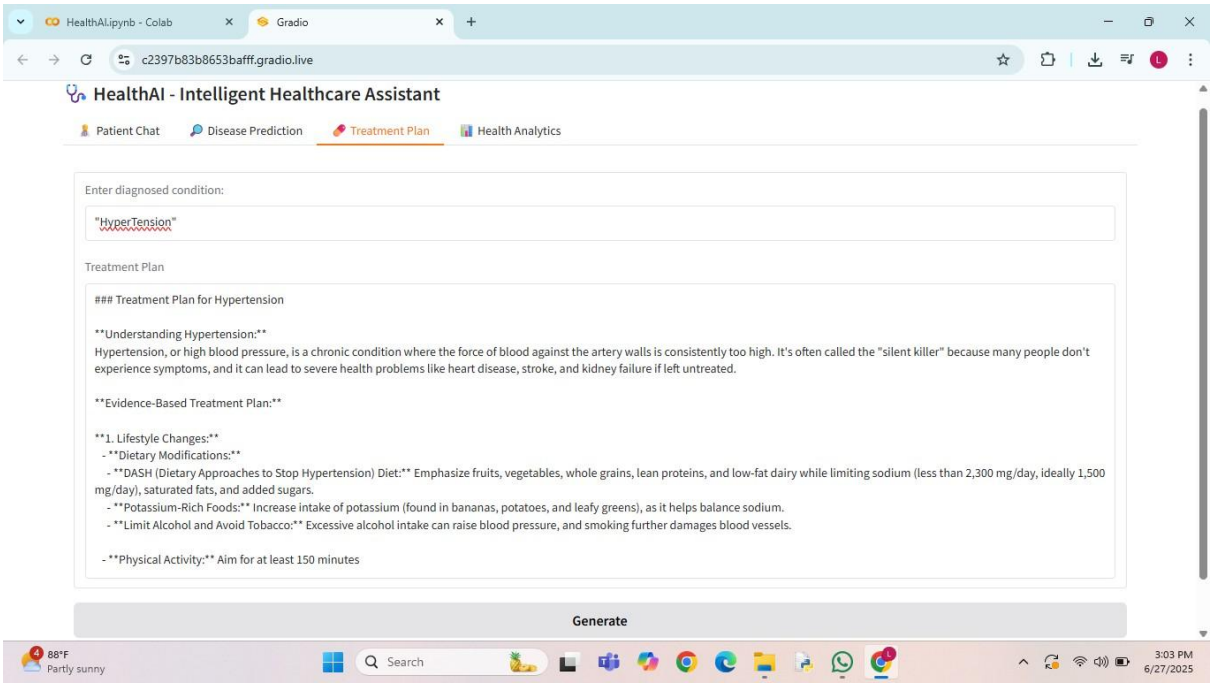

HealthAI.ipynb - Colab
Gradio
colab.research.google.com/drive/1hx5NzRuFrEYMXMOSuMFwB6vl9Mq3qF-C#scrollTo=xofAxC6fnb2u
HealthAI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
!pip install transformers gradio accelerate sentencepiece --quiet
363.4/363.4 MB 4.5 MB/s eta 0:00:00
13.8/13.8 MB 43.0 MB/s eta 0:00:00
24.6/24.6 MB 30.3 MB/s eta 0:00:00
883.7/883.7 kB 31.1 MB/s eta 0:00:00
664.8/664.8 MB 3.0 MB/s eta 0:00:00
211.5/211.5 MB 3.3 MB/s eta 0:00:00
56.3/56.3 MB 17.8 MB/s eta 0:00:00
127.9/127.9 MB 6.9 MB/s eta 0:00:00
207.5/207.5 MB 5.1 MB/s eta 0:00:00
21.1/21.1 MB 44.9 MB/s eta 0:00:00
[2]
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch
# Check for GPU
device = "cuda" if torch.cuda.is_available() else "cpu"
# Model ID
model_id = "ibm-granite/granite-3.3-2b-instruct"
# Load tokenizer and model
tokenizer = AutoTokenizer.from_pretrained(model_id)
Variables Terminal
2:24 PM Python 3
Trending videos
Cute Golden Ret...
Search
2:41 PM
6/27/2025

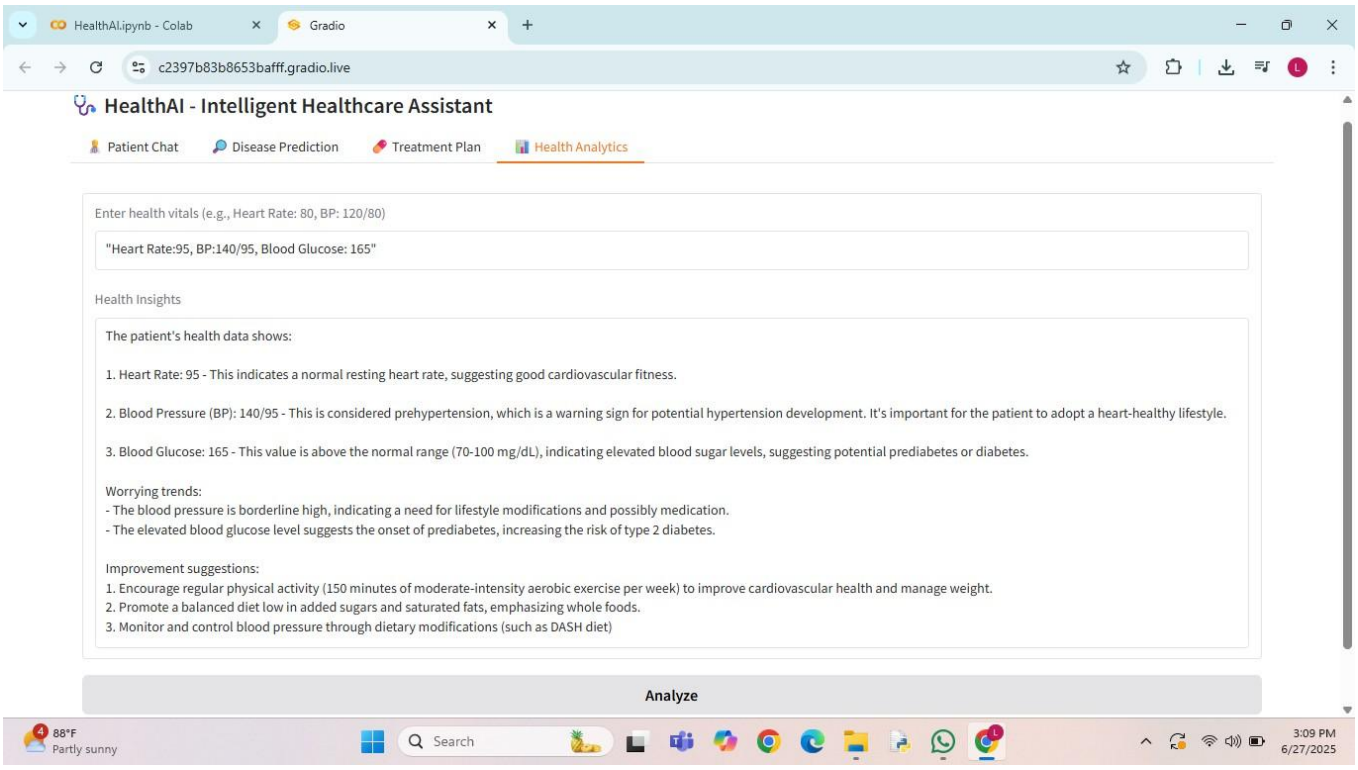

HealthAI.ipynb - Colab
Gradio
colab.research.google.com/drive/1hx5NzRuFrEYMXMOSuMFwB6vl9Mq3qF-C#scrollTo=xofAxC6fnb2u
HealthAI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://c2397b83b8653bafff.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Huggi
HealthAI - Intelligent Healthcare Assistant
Patient Chat
Disease Prediction
Treatment Plan
Health Analytics
Ask a medical question:
AI Response
Ask
Variables Terminal
2:24 PM Python 3
Trending videos
Cute Golden Ret...
Search
2:41 PM
6/27/2025


HealthAI.ipynb - Colab
Gradio
Gradio
c2397b83b8653bafff.gradio.live
HealthAI - Intelligent Healthcare Assistant
Patient Chat
Disease Prediction
Treatment Plan
Health Analytics
Ask a medical question:
AI Response
Ask
Use via API · Built with Gradio · Settings
Trending videos
Cute Golden Ret...
Search
2:41 PM
6/27/2025


HealthAI.ipynb - Colab
Gradio
c2397b83b8653bafff.gradio.live
HealthAI - Intelligent Healthcare Assistant
Patient Chat
Disease Prediction
Treatment Plan
Health Analytics
Ask a medical question:
can I go to school with a sore throat
AI Response
I'm sorry to hear that you're not feeling well. A sore throat is quite common, and it's usually a sign of a minor viral or bacterial infection. While it's important to rest and stay hydrated, attending school can be a bit challenging with a sore throat. It depends on the severity of your symptoms. If your throat is painful and swallowing is difficult, it might be best to stay home to prevent spreading germs and allow your body time to recover. However, if your symptoms are mild, like a tickle in your throat, you might consider going to school, but be sure to take necessary precautions such as covering your mouth when coughing or sneezing, and avoiding close contact with others.
Remember, your health and the health of your peers are equally important. It's also a good idea to consult with a healthcare provider if your symptoms persist or worsen. They can provide guidance on managing your symptoms and determine if you need further medical attention.
Please take care of yourself and listen to your body.
Ask
Use via API · Built with Gradio · Settings
News for you
Mets Issue Griffi...
Search
2:50 PM
6/27/2025


HealthAI.ipynb - Colab
Gradio
c2397b83b8653bafff.gradio.live
HealthAI - Intelligent Healthcare Assistant
Patient Chat
Disease Prediction
Treatment Plan
Health Analytics
Describe your symptoms:
"fever, chills, fatigue"
Possible Conditions
Based on the symptoms of fever, chills, and fatigue, several conditions could be considered. Here are some possibilities along with their likelihoods and brief medical guidance:
1. **Viral Infection (Flu, Common Cold, COVID-19)**:
- **Likelihood**: Moderate to high, depending on seasonality.
- **Guidance**: These are highly contagious viral illnesses. Fever, chills, and fatigue are common symptoms. Antiviral medications (like oseltamivir for influenza) may be beneficial if started early. Rest and hydration are crucial.
2. **Bacterial Infection (Strep Throat, Urinary Tract Infection, Pneumonia)**:
- **Likelihood**: Moderate, depending on secondary bacterial complications.
- **Guidance**: If a bacterial infection is suspected, antibiotics would typically be prescribed. A strep throat, for instance, often requires rapid strep test for confirmation.
3. **Gastrointestinal (GI) Infections (Food Poisoning, Gastroenteritis)**:
- **Likelihood**: Possible, especially if nausea or diarrhea is also present.
- **Guidance**: These often present with
Predict
Sports headline
Last Night in Bas...
Search
2:56 PM
6/27/2025


HealthAI.ipynb - Colab
Gradio
c2397b83b8653bafff.gradio.live
HealthAI - Intelligent Healthcare Assistant
Patient Chat
Disease Prediction
Treatment Plan
Health Analytics
Enter diagnosed condition:
"HyperTension"
Treatment Plan
Treatment Plan for Hypertension
Understanding Hypertension:
Hypertension, or high blood pressure, is a chronic condition where the force of blood against the artery walls is consistently too high. It's often called the "silent killer" because many people don't experience symptoms, and it can lead to severe health problems like heart disease, stroke, and kidney failure if left untreated.
Evidence-Based Treatment Plan:
1. Lifestyle Changes:
- **Dietary Modifications:**
- **DASH (Dietary Approaches to Stop Hypertension) Diet:** Emphasize fruits, vegetables, whole grains, lean proteins, and low-fat dairy while limiting sodium (less than 2,300 mg/day, ideally 1,500 mg/day), saturated fats, and added sugars.
- **Potassium-Rich Foods:** Increase intake of potassium (found in bananas, potatoes, and leafy greens), as it helps balance sodium.
- **Limit Alcohol and Avoid Tobacco:** Excessive alcohol intake can raise blood pressure, and smoking further damages blood vessels.
- **Physical Activity:** Aim for at least 150 minutes
Generate
88°F
Partly sunny
Search
3:03 PM
6/27/2025


HealthAI.ipynb - Colab
Gradio
c2397b83b8653bafff.gradio.live
HealthAI - Intelligent Healthcare Assistant
Patient Chat
Disease Prediction
Treatment Plan
Health Analytics
Enter health vitals (e.g., Heart Rate: 80, BP: 120/80)
"Heart Rate:95, BP:140/95, Blood Glucose: 165"
Health Insights
The patient's health data shows:
1. Heart Rate: 95 - This indicates a normal resting heart rate, suggesting good cardiovascular fitness.
2. Blood Pressure (BP): 140/95 - This is considered prehypertension, which is a warning sign for potential hypertension development. It's important for the patient to adopt a heart-healthy lifestyle.
3. Blood Glucose: 165 - This value is above the normal range (70-100 mg/dL), indicating elevated blood sugar levels, suggesting potential prediabetes or diabetes.
Worrying trends:
- The blood pressure is borderline high, indicating a need for lifestyle modifications and possibly medication.
- The elevated blood glucose level suggests the onset of prediabetes, increasing the risk of type 2 diabetes.
Improvement suggestions:
1. Encourage regular physical activity (150 minutes of moderate-intensity aerobic exercise per week) to improve cardiovascular health and manage weight.
2. Promote a balanced diet low in added sugars and saturated fats, emphasizing whole foods.
3. Monitor and control blood pressure through dietary modifications (such as DASH diet)
Analyze
88°F
Partly sunny
Search
3:09 PM
6/27/2025

12. Future Enhancements

1. Add User Authentication & Profiles
2. Implement secure login (Google, Email/Password)
3. Allow users to create profiles (patients/doctors)
4. Enable personalized tracking of symptoms, history, and recommendations

2. Database Integration

Store chat history, symptom logs, predictions, and analytics

Use cloud databases like Firebase, MongoDB, or PostgreSQL

Enable long-term patient monitoring and reporting

3. Doctor Panel / Admin Dashboard A separate interface for doctors to: Review patient inputs

Approve or modify AI suggestions

Track user cases and trends

4. Voice Input & Output

Add speech-to-text for input and text-to-speech for responses

Helps elderly or visually impaired users access healthcare advice easily

5. Multilingual Support

Enable the assistant to understand and reply in multiple languages (e.g., Hindi, Telugu, Tamil)

Useful for users in rural areas or those not fluent in English

6. Medical Database Integration

Connect with real-world medical databases (like WHO, CDC, or MedlinePlus)

Provide verified and up-to-date health information

7. Mobile App Development

Convert the Gradio UI into a cross-platform mobile app using tools like Flutter or React Native

Make HealthAI more accessible on smartphones

8. Symptom Image Input (Future AI Vision)

Allow users to upload images (e.g., skin rash, eye redness)

Use computer vision models to support diagnosis with visual input

9. Emergency Alert Feature

If AI detects severe or risky symptoms, notify emergency contacts or suggest urgent hospital visit

10. Model Fine-Tuning

Train the AI on healthcare-specific data for better accuracy

Improve disease prediction and treatment quality through real-world feedback

Would you like these enhancements formatted for a report or project presentation slide?