HealthAI: Intelligent Healthcare Assistant Using IBM Granite

A Project Report
Submitted to the Naan Mudhalvan course for the award of degree of

BACHELOR OF COMPUTER SCIENCE

Submitted By

(Team-Id: NM2025TMID04699)

Name	Reg. No
DIVYA A	222305234
GEETHANJALI R	222305235
KARTHIKEYAN I	222305213
PONPUGAZH P	222305216

Department of Computer Science

PACHAIYAPPA'S COLLEGE

(Affiliated to the University of Madras)

CHENNAI – 600030.

November – 2025.

PACHAIYAPPA'S COLLEGE

(Affiliated to the University of Madras)

CHENNAI-600 030.

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the Naan Mudhalvan project work entitled **"HealthAI: Intelligent Healthcare Assistant Using IBM Granite"** is the Bonafide record of work done by **(Divya - Geethanjali - Karthikeyan - Ponpugazh)** in partial fulfillment for the award of the degree of **Bachelor of Computer Science,** under our guidance and supervision, during the academic year 2025-2026.

Head of the Department
Dr. J. KARTHIKEYAN

Staff-in-Charge
Mr. K. DINESHKUMAR

Submitted for Viva-Voce Examination held on…….a Pachaiyappa's College, Chennai-30.

Internal Examiner

External Examiner

HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Table of Contents

1. Introduction

Project Title: HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Team Members:

S. No.	Name	Reg. No
1	DIVYA A	222305234
2	GEETHANJALI R	222305235
3	KARTHIKEYAN I	222305213
4	PONPUGAZH P	222305216

Program Background:

The *IBM Naan Mudhalvan Smart Internz Program* is an initiative designed to empower students and early professionals by providing real-world project experience with IBM's cutting-edge technologies. The program integrates *IBM Watsonx*, AI models, and cloud-native development practices into hands-on guided projects.

As part of this program, our team developed *Health AI*, a healthcare-focused assistant that leverages generative AI, predictive analytics, and anomaly detection to provide intelligent, accessible, and real-time health-related insights.

2. Project Overview

2.1 Purpose

The primary purpose of *Health AI* is to create an **intelligent digital healthcare assistant** that leverages artificial intelligence to support patients, healthcare professionals, and caregivers in accessing reliable, real-time medical insights. With the growing demand for **personalized healthcare solutions**, this project integrates **conversational AI, predictive analytics, and anomaly detection** to bridge the gap between medical knowledge and end-user accessibility.

By combining **natural language processing, machine learning, and multimodal data analysis**, the system empowers users to:

- Understand complex medical information through simplified summaries.
- Predict health trends and risks using forecasting models.
- Detect anomalies in patient health metrics for early intervention.
- Receive actionable wellness tips tailored to individual health profiles.

Ultimately, *Health AI* seeks to **enhance decision-making, improve preventive care, and foster patient engagement** by providing an AI-powered platform that is both **scalable and user-friendly**.

2.2 Features

The system incorporates a comprehensive set of features designed to provide **end-to-end healthcare assistance**:

1. **Conversational Chat Interface**
    - Natural language-based chat system for answering medical questions.
    - Powered by *IBM Watsonx Granite LLM* for reliable and contextual responses.
2. **Medical Document Summarization**
    - Converts lengthy medical reports, prescriptions, or research papers into concise, understandable summaries.
    - Supports patients in comprehending complex medical terminology.
3. **Health Data Forecasting**
    - Time-series forecasting to predict future health trends such as blood pressure, glucose levels, or patient admission rates.
    - Useful for preventive care and resource planning in healthcare facilities.

4. **Healthcare Tips Generator**
    - Provides AI-driven, personalized wellness tips based on user health data and lifestyle inputs.
    - Encourages preventive healthcare practices.
5. **Feedback and Improvement Loop**
    - Allows patients and healthcare workers to submit feedback about the assistant's performance.
    - Data collected is used to refine system accuracy and reliability.
6. **KPI Tracking**
    - Tracks key performance indicators such as patient recovery rate, medication adherence, and vital sign fluctuations.
    - Displays insights in dashboard visualizations.
7. **Anomaly Detection**
    - Detects abnormal patterns in patient health data, such as sudden spikes in heart rate or irregular blood sugar readings.
    - Raises early alerts for potential risks.
8. **Multimodal Input Support**
    - Accepts **text, CSV files, and PDFs** as input for analysis.
    - Flexible support ensures both structured (numerical data) and unstructured (clinical notes) inputs can be processed.
9. **User Interface (UI)**
    - Developed using **Streamlit**, ensuring a clean, accessible, and intuitive design.
    - Provides sidebar navigation, KPI dashboards, report downloads, and interactive visualizations.

3. Architecture

The architecture of *Health AI* integrates **generative AI models, a lightweight user interface, and structured prompt engineering** to deliver healthcare insights. The system is modular, with clear separation between the **model inference layer**, the **user interaction layer**, and the **prompt logic** that defines the healthcare use cases.

3.1 Model Layer – IBM Granite LLM

- The core intelligence of *Health AI* comes from the **IBM Granite 3.2 2B Instruct model**, accessed via Hugging Face Transformers.
- Components:
    - **Tokenizer**: Converts user input into tokens that can be processed by the model.
    - **Model**: AutoModelForCausalLM is used for **text generation** with prompt conditioning.
    - **Inference Settings**: Configurable parameters such as max_length, temperature, and do_sample allow control over creativity vs. accuracy.
- This layer handles **language understanding, reasoning, and response generation**.

3.2 Prompt Engineering Layer

- Defines *specific healthcare tasks* by wrapping user input into carefully structured prompts.
- Implemented as Python functions:
    - **Disease Prediction**: Takes a list of symptoms and generates possible medical conditions and general suggestions.
    - **Treatment Plan Generator**: Takes condition, age, gender, and medical history to generate a personalized care plan.
- Each prompt explicitly includes a **medical disclaimer** to reinforce that the output is *informational only* and not a replacement for professional care.

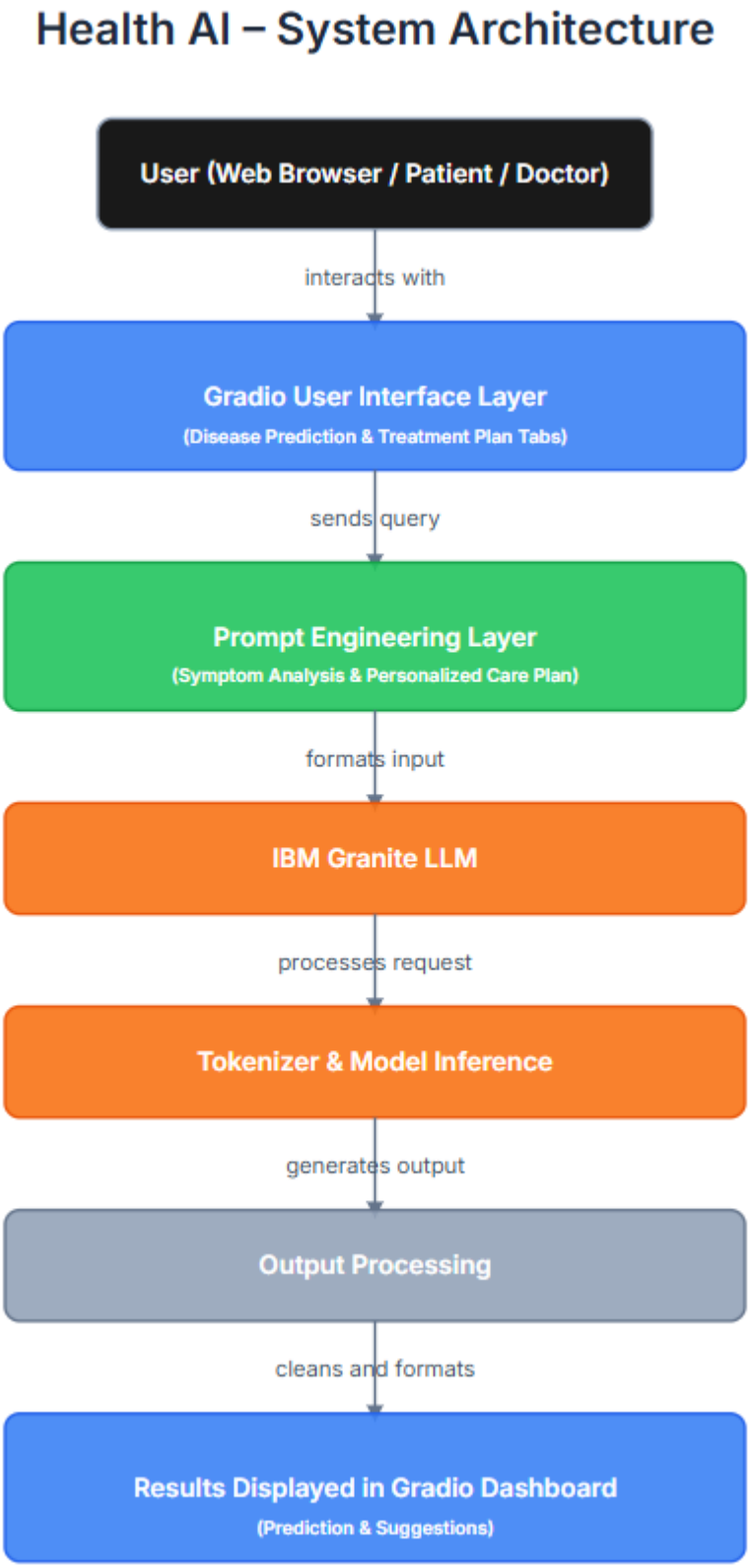
3.3 Application Layer – Gradio Interface

- The user interface is developed using **Gradio Blocks**, providing a **web-based, tabbed application** with interactive widgets.
- Key elements:
    - **Tabs**:
        - *Disease Prediction* → Accepts symptoms and displays conditions.
        - *Treatment Plan* → Collects patient demographics + condition and returns care suggestions.
    - **Input Widgets**:
        - Textboxes for symptoms, condition, and history.
        - Number field for age.
        - Dropdown for gender.
    - **Output Widgets**:
        - Large textbox areas for results and recommendations.
- Advantages:
    - Lightweight deployment (can run on local machine or Colab).
    - Built-in support for hosting (app.launch(share=True)).

3.4 System Flow

1. **User Input**
    - The user provides symptoms (for prediction) or condition details (for treatment plan).
2. **Prompt Construction**
    - Input is passed to the prompt engineering layer, which wraps it into a structured query with disclaimers.
3. **Model Inference**
    - The IBM Granite LLM processes the input and generates a response.
4. **Output Processing**
    - Raw model text is decoded, cleaned, and presented in a user-friendly format.
5. **Result Display**
    - Gradio interface displays the AI-generated analysis in real time.

3.5 Architecture Diagram



4. Setup Instructions

The *Health AI* project requires a Python environment with support for Hugging Face Transformers, PyTorch, and Gradio. The following setup guide ensures smooth installation and execution across local machines or cloud environments such as **Google Colab**.

4.1 Prerequisites

Before installation, ensure the following requirements are met:

- **Python Environment**: Python 3.8 or higher.
- **Package Manager**: pip (latest version recommended).
- **Hardware**:
    - CPU (works, but slower).
    - GPU (recommended – CUDA-enabled device or Google Colab T4 GPU).
- **Internet Access**: Required to download the IBM Granite model from Hugging Face.
- **Git**: For cloning repositories (optional but recommended).

4.2 Installation Steps

Step 1: Clone the Repository

If your code is hosted on GitHub:

git clone <your-repo-url>

cd health-ai

If not, download the project folder manually and open it in your preferred IDE (VS Code, PyCharm, Jupyter Notebook, etc.).

Step 2: Create and Activate a Virtual Environment *(recommended)*

python -m venv venv

Windows

venv\Scripts\activate

macOS/Linux

source venv/bin/activate

Step 3: Install Dependencies

Install all required Python libraries:

pip install torch transformers gradio

💡 If running on GPU with CUDA, ensure the appropriate PyTorch version is installed. Refer to PyTorch Installation Guide.

Step 4: Configure Model Access

- The project uses **IBM Granite models** hosted on Hugging Face.
- You may need to **create a Hugging Face account** and accept model usage terms.
- If authentication is required, run the following command and enter your Hugging Face token:
- huggingface-cli login

Step 5: Run the Application

Execute the Gradio application:

python app.py

This will launch a local server. By default, you will see output like:

Running on local URL: http://127.0.0.1:7860

Running on public URL: https://xxxx.gradio.live

- Open the **local URL** in your browser to use the system.
- If running on **Google Colab**, click the **public Gradio link** to access the app.

4.3 Google Colab Setup (Optional)

If you prefer running in Colab with GPU support:

1. Open Google Colab.
2. Create a new notebook and set **Runtime → Change runtime type → GPU (T4)**.
3. Install dependencies:
4. !pip install torch transformers gradio
5. Upload your project file (app.py) or paste the code into a cell.
6. Run the cell to start the Gradio interface.
7. Click the generated **shareable link** to open the application.

5. Folder Structure

```
health-ai/
├── app.py                  # Main entry point for the Gradio application
├── requirements.txt        # Python dependencies for the project
├── README.md               # Project description and usage instructions
│
├── /models                 # Model-related utilities and configuration
│   ├── model_loader.py     # Handles loading IBM Granite LLM and tokenizer
│   └── prompts.py          # Stores prompt templates for disease prediction, treatment, etc.
│
├── /services               # Core backend services
│   ├── disease_service.py  # Functions for disease prediction based on symptoms
│   ├── treatment_service.py# Functions for personalized treatment planning
│   └── utils.py            # Helper utilities (response formatting, cleaning, etc.)
│
├── /ui                     # User interface logic (Gradio components)
│   ├── layout.py           # Defines the structure of tabs, textboxes, and buttons
│   ├── components.py       # Reusable UI elements (text areas, dropdowns, disclaimers)
│   └── styles.css          # (Optional) Custom CSS for styling the Gradio interface
│
├── /data                   # Sample input data for testing
│   ├── symptoms_examples.txt
│   └── patient_profiles.csv
│
└── /docs                   # Documentation resources
    ├── architecture.png    # Architecture diagram (placeholder)
    ├── screenshots/        # Screenshots of UI for the report
    └── report.md           # Extended project documentation
```

6. Running the Application

Once the setup is complete, the *Health AI* system can be executed in both **local environments** and **cloud-based environments** (e.g., Google Colab). The following steps describe how to launch and interact with the application.

6.1 Running Locally (on your computer)

1. **Activate Virtual Environment** (if created):
2. # Windows
3. venv\Scripts\activate
4. # macOS/Linux
5. source venv/bin/activate
6. **Run the Application**
   Execute the main script:
7. python app.py
8. **Access the Application**
   After execution, Gradio will generate links:
9. Running on local URL: http://127.0.0.1:7860
10. Running on public URL: https://xxxx.gradio.live
    - Open the **local URL** in your browser if running on your machine.
    - Use the **public URL** (auto-generated by Gradio) to share the app with others.

6.2 Running on Google Colab (with GPU)

1. Open Google Colab.
2. Create a new notebook and set **Runtime → Change Runtime Type → GPU (T4 recommended)**.
3. Install dependencies:
4. !pip install torch transformers gradio
5. Upload your app.py file or paste the code into a notebook cell.
6. Run the notebook. Gradio will output a **public shareable link** (e.g., https://xxxx.gradio.live) to access the interface.

6.3 Interacting with the Application

Once launched, the application provides a **tabbed dashboard** with the following features:

- **Disease Prediction Tab**
    - Input: Symptoms (e.g., “fever, cough, headache”).
    - Output: Possible medical conditions and general recommendations.
- **Treatment Plan Tab**
    - Input: Medical condition, age, gender, and medical history.
    - Output: Personalized treatment suggestions, home remedies, and general medication guidelines.

Each response includes a **medical disclaimer** emphasizing that outputs are for **informational purposes only** and must not replace professional medical consultation.

7. API Documentation

The *Health AI* application provides functionality through structured functions that act as service endpoints within the Gradio interface. While the system is primarily deployed as a **web app via Gradio**, the internal logic can also be exposed as APIs if integrated with a REST framework (e.g., FastAPI).

Below is the documentation of the core functional endpoints:

7.1 Disease Prediction API

Functionality:
Analyzes user-provided symptoms and suggests possible medical conditions along with general recommendations.

Method: POST
Endpoint (logical): /disease/predict
Request Parameters:

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

Response Example:

```
{
  "conditions": [
    "Common Cold",
    "Viral Fever",
    "Influenza"
  ],
  "recommendations": "Stay hydrated, take adequate rest, and consult a healthcare professional if symptoms persist."
}
```

- **Disclaimer:** Output is for **informational purposes only**. Must not be used for professional medical diagnosis.

7.2 Treatment Plan API

Functionality:
Generates a personalized treatment suggestion based on patient demographics and medical history.

Method: POST
Endpoint (logical): /treatment/plan
Request Parameters:

```
{
  "condition": "Hypertension",
  "age": 45,
  "gender": "Male",
  "medical_history": "Diabetes, on medication"
}
```

Response Example:

```
{
  "treatment_plan": "Maintain a low-salt diet, engage in light exercise such as walking, monitor blood pressure daily. Consult a physician regarding continuation or adjustment of medication."
}
```

- **Disclaimer:** This is **informational guidance only**. Professional consultation is mandatory.

7.3 Shared Functionalities

Both APIs rely on the **IBM Granite LLM** via Hugging Face Transformers:

- **Model:** ibm-granite/granite-3.2-2b-instruct
- **Tokenizer:** Hugging Face AutoTokenizer
- **Generation Settings:**
    - max_length: up to 1200 tokens
    - temperature: 0.7 (balanced creativity and accuracy)
    - do_sample: True (varied responses)

7.4 Swagger/OpenAPI (Planned)

Currently, the system is **only accessible via the Gradio UI**.
For future iterations, we plan to expose these functionalities as REST APIs with:

- **FastAPI + Swagger UI** for endpoint testing.
- **Authentication (JWT / OAuth2)** for security.
- **Role-based access** for patients, doctors, and administrators.

8. Authentication

Currently, the *Health AI* application runs in an **open environment** where users can directly access the Gradio interface without login or identity verification. This approach is suitable for **prototype development, testing, and demonstration purposes**. However, for production deployments in **healthcare settings**, authentication is a critical requirement to ensure **security, privacy, and compliance with healthcare regulations**.

8.1 Current Implementation

- **Access Mode:** Open, no login required.
- **Reasoning:**
  - Simplicity for project demonstration under the IBM Naan Mudhalvan Smart Internz Program.
  - Facilitates quick testing without configuration overhead.
- **Risks:**
  - Any user with the application link can access the system.
  - No user-level tracking or data protection mechanisms.

8.2 Planned Enhancements

To transition from a prototype to a **secure healthcare-grade solution**, the following authentication mechanisms are planned:

1. **JWT (JSON Web Token) Authentication**
   - Each user receives a signed token upon login.
   - Token is included in all subsequent requests to verify session identity.
   - Lightweight and scalable for web-based applications.
2. **OAuth2 Authentication**
   - Integration with third-party identity providers (e.g., Google, IBM Cloud Identity, Microsoft Azure AD).
   - Provides secure login and simplifies credential management.
   - Supports **role-based access control** (e.g., Patient, Doctor, Admin).
3. **Role-Based Access Control (RBAC)**
   - Patients → Can use symptom checker and treatment plan modules.
   - Doctors → Can access extended features, analytics, and feedback review.
   - Admins → Manage system settings, monitor activity logs, and configure model access.

4. **Session Management**
    - User sessions tracked securely.
    - Automatic logout after inactivity.
    - Option to maintain session history for repeat users.

8.3 Data Privacy & Compliance Considerations

For healthcare-related applications, **authentication must align with medical data regulations** such as:

- **HIPAA (Health Insurance Portability and Accountability Act)** – for handling patient records in the U.S.
- **GDPR (General Data Protection Regulation)** – for protecting user data in the EU.
- **Consent Management** – ensuring users acknowledge that the system provides *informational guidance only*.

9. User Interface

The *Health AI* application adopts a **minimalist and user-friendly design philosophy**, prioritizing accessibility and clarity for both technical and non-technical users. Built using the **Gradio Blocks framework**, the interface provides a web-based dashboard that runs directly in a browser, eliminating the need for complex installations or external applications.

9.1 Design Principles

- **Simplicity:** A clean layout with clearly labeled input and output areas ensures ease of use.
- **Accessibility:** The interface is browser-based, requiring no additional setup for the end-user.
- **Guidance:** Embedded disclaimers and placeholder text guide users on how to interact responsibly with the system.
- **Responsiveness:** Automatically adjusts to different screen sizes, making it usable on desktops, laptops, and tablets.

9.2 Layout Overview

The interface is organized into **two primary tabs**, accessible via a navigation bar:

1. **Disease Prediction Tab**
    - **Inputs:**
        - Textbox for entering symptoms (e.g., “fever, cough, headache”).
        - Button: *“Analyze Symptoms”*.
    - **Outputs:**
        - Large textbox displaying predicted possible conditions.
        - General recommendations (e.g., rest, hydration).
    - **Disclaimers:** Emphasizes that the output is for *informational purposes only*.
2. **Treatment Plan Tab**
    - **Inputs:**
        - Textbox for *Medical Condition*.
        - Number field for *Age*.
        - Dropdown for *Gender* (Male, Female, Other).
        - Textbox for *Medical History* (e.g., allergies, chronic conditions).
        - Button: *“Generate Treatment Plan”*.

- **Outputs:**
    - Detailed textbox containing a personalized treatment plan, lifestyle guidance, and home remedies.
- **Disclaimers:** Reinforces that this is not a replacement for medical advice.

9.3 Additional Features

- **Disclaimers:** Prominently displayed at the top of the dashboard, ensuring ethical and safe usage of AI-generated outputs.
- **Interactive Widgets:** Input fields (textboxes, dropdowns, number inputs) make it easy for users to enter structured data.
- **Real-time Processing:** Responses are displayed instantly after clicking the action buttons.
- **Public Access Links:** When deployed on **Google Colab or local environments**, Gradio generates a shareable URL for remote access.

10. Testing

Testing was conducted systematically to ensure that the *Health AI* system functions reliably across its core features—**disease prediction, treatment planning, and user interaction through the Gradio interface**. The testing strategy combined **unit testing, functional validation, API-level checks, and manual testing of the user interface**.

10.1 Testing Strategy

The following approaches were employed:

1. **Unit Testing**
    - Verified the correctness of backend functions such as:
        - disease_prediction() – ensures proper construction of prompts and return values.
        - treatment_plan() – checks formatting of treatment suggestions and disclaimers.
        - Model loading and tokenizer initialization.
    - Tools used: Python's built-in unittest framework.
2. **Functional Testing**
    - Validated the **end-to-end workflow** from user input → prompt generation → model inference → output display.
    - Checked response length, coherence, and presence of disclaimers.
    - Ensured Gradio interface elements (buttons, textboxes, dropdowns) function as intended.
3. **API Testing (Logical)**
    - Even though the system is deployed as a **Gradio app**, the core functions were tested as if they were **API endpoints**.
    - Example: Passing sample symptom strings into disease_prediction() and verifying the expected structure of results.
    - Ensured consistent response formatting (JSON-like outputs).
4. **Manual Testing**
    - Conducted exploratory testing by interacting directly with the Gradio UI.
    - Tested multiple combinations of inputs:
        - Short vs. long symptom descriptions.
        - Different demographic profiles (age, gender, medical history).
        - Edge-case conditions (rare or conflicting symptoms).

5. **Edge Case & Error Handling**
    - Invalid inputs (e.g., empty textboxes, nonsensical strings).
    - Very long symptom descriptions exceeding token limits.
    - GPU vs. CPU execution (ensured fallback to CPU works).
    - Network issues during Hugging Face model download.

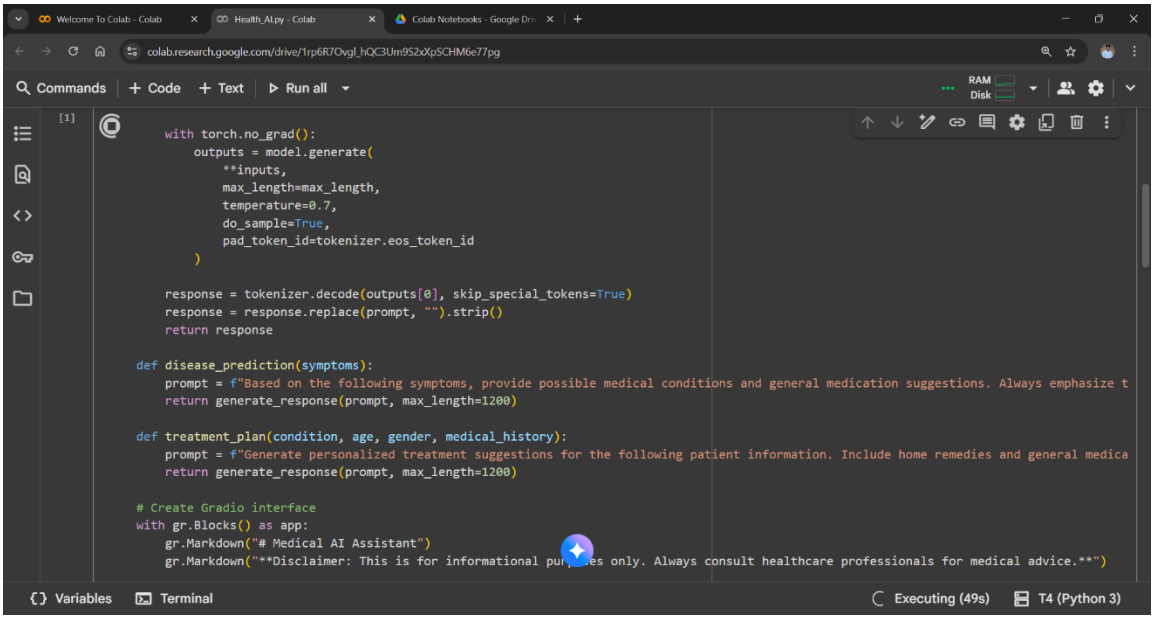
10.2 Observations

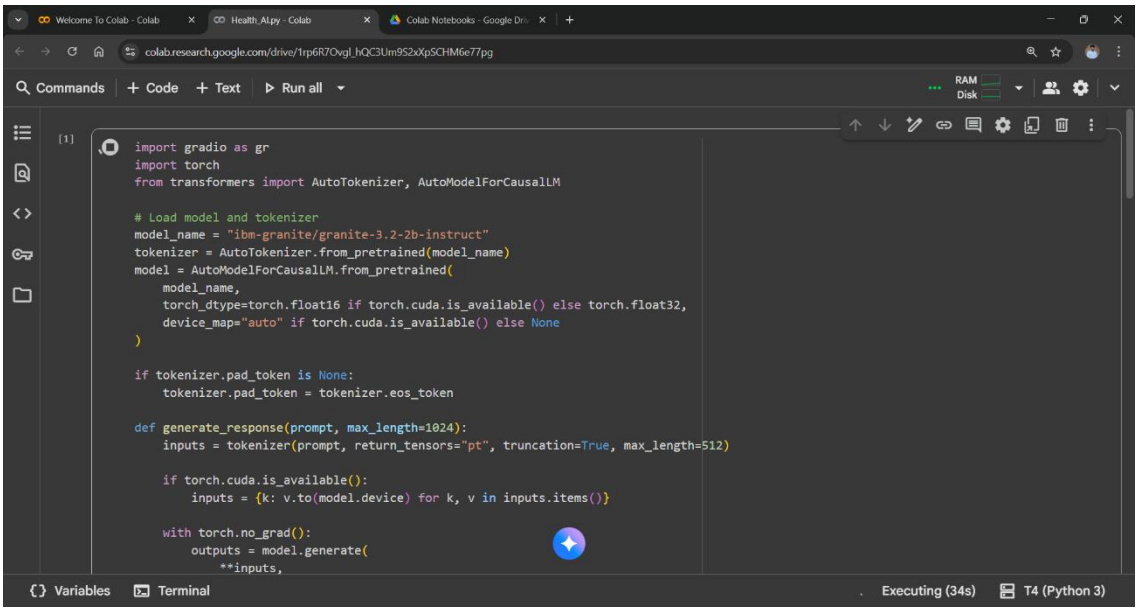
- The model consistently generated **relevant, medically plausible outputs**, but variability exists due to the **stochastic nature of language models**.
- The **disclaimer prompt engineering** worked reliably—outputs always contained safety notices.
- Performance was **significantly faster on GPU (Colab T4)** compared to CPU.
- UI remained responsive and stable even with longer inputs.

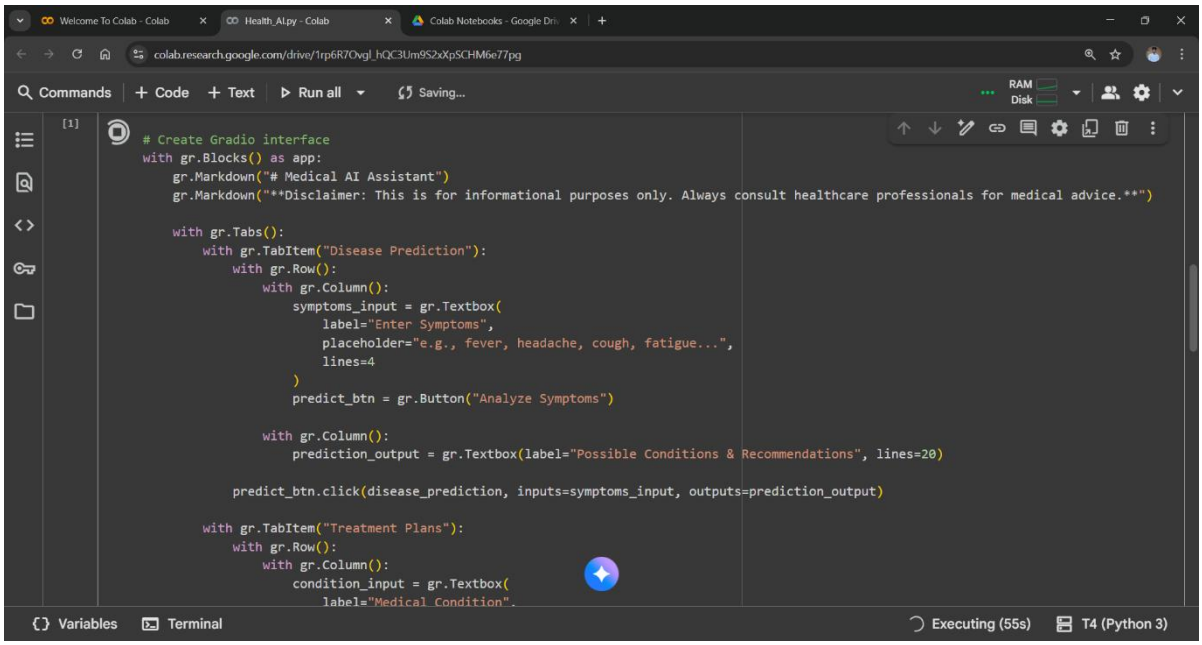
10.3 Limitations in Testing

- Outputs were **not clinically validated** by medical professionals (informational use only).
- Lack of automated load testing for concurrent users (single-user focus).
- Hugging Face model download dependency introduces variability in execution times during fresh setups.

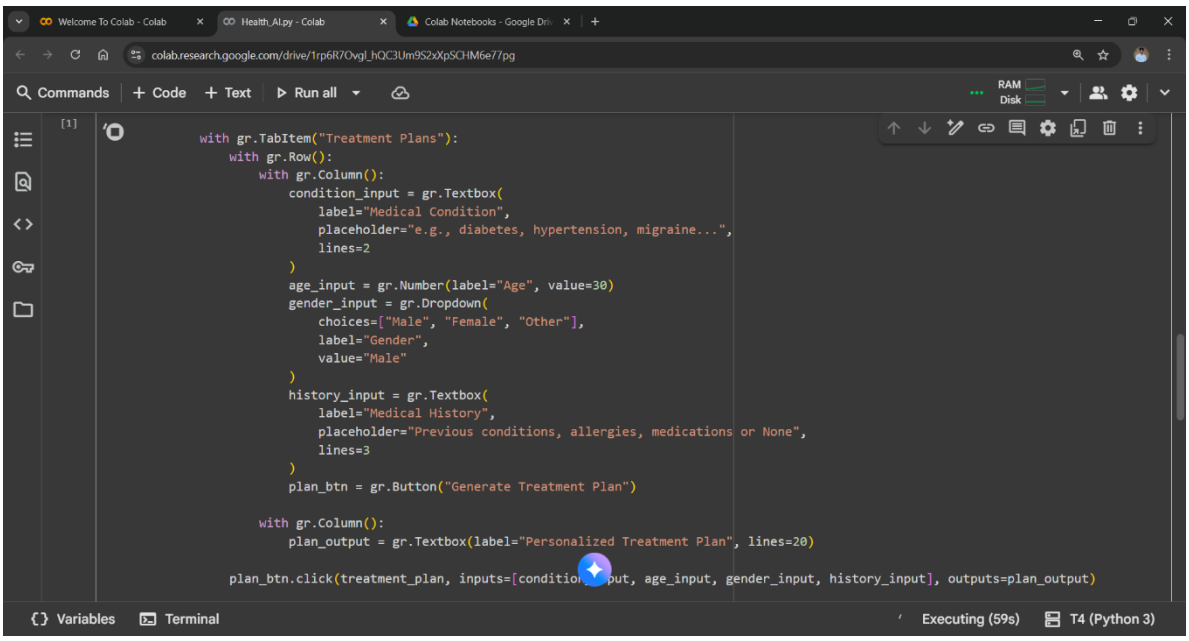
11. Screenshots



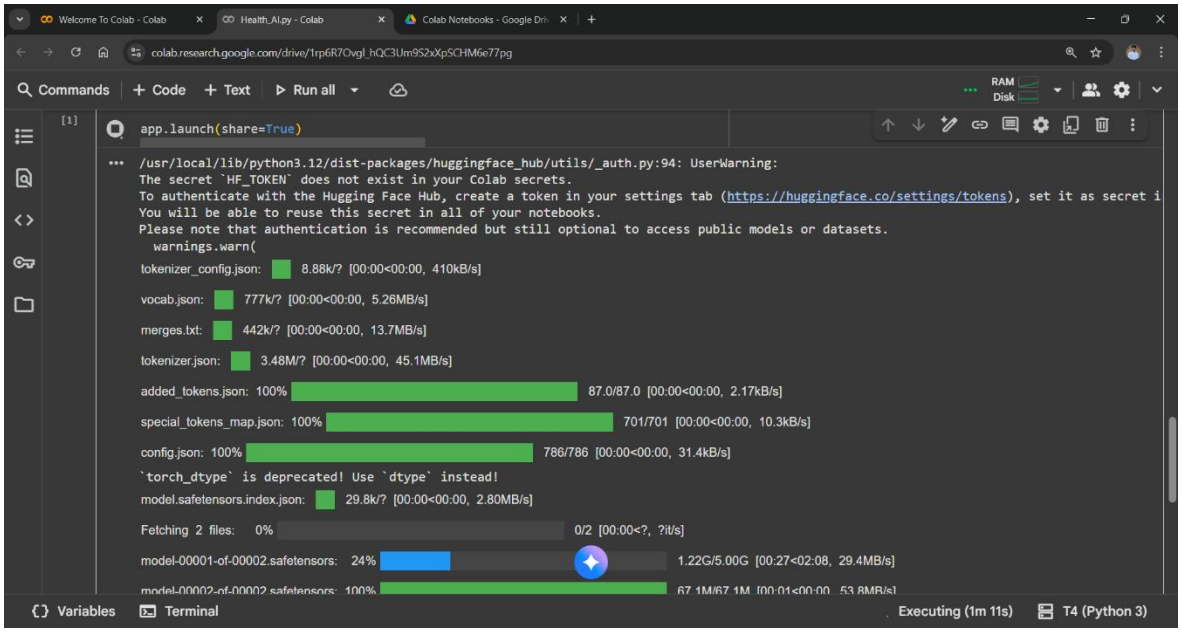


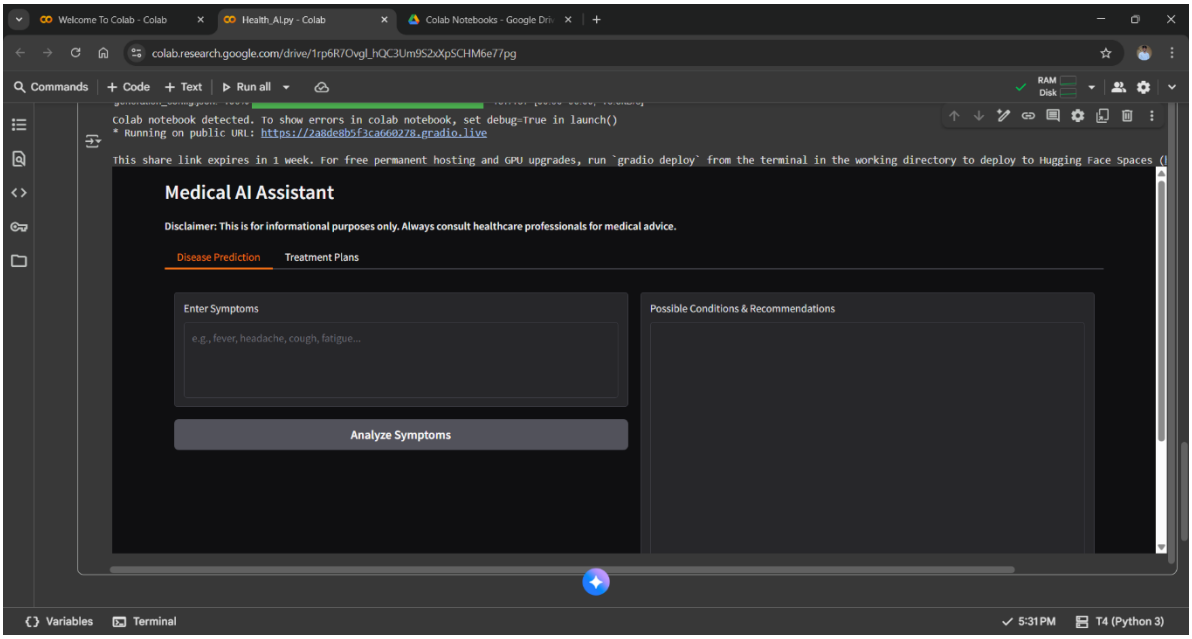


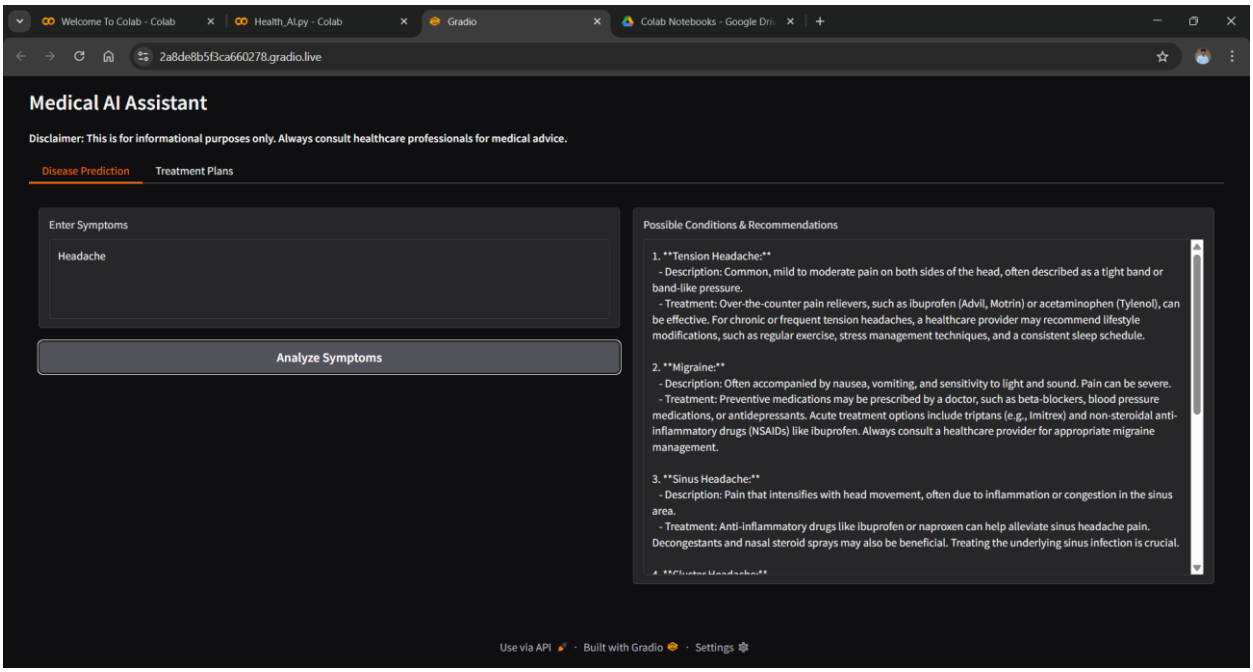
```
# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Medical AI Assistant")
    gr.Markdown("**Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.**")

    with gr.Tabs():
        with gr.TabItem("Disease Prediction"):
            with gr.Row():
                with gr.Column():
                    symptoms_input = gr.Textbox(
                        label="Enter Symptoms",
                        placeholder="e.g., fever, headache, cough, fatigue...",
                        lines=4
                    )
                    predict_btn = gr.Button("Analyze Symptoms")

                with gr.Column():
                    prediction_output = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)

            predict_btn.click(disease_prediction, inputs=symptoms_input, outputs=prediction_output)

        with gr.TabItem("Treatment Plans"):
            with gr.Row():
                with gr.Column():
                    condition_input = gr.Textbox(
                        label="Medical Condition",
```



```
        with gr.TabItem("Treatment Plans"):
            with gr.Row():
                with gr.Column():
                    condition_input = gr.Textbox(
                        label="Medical Condition",
                        placeholder="e.g., diabetes, hypertension, migraine...",
                        lines=2
                    )
                    age_input = gr.Number(label="Age", value=30)
                    gender_input = gr.Dropdown(
                        choices=["Male", "Female", "Other"],
                        label="Gender",
                        value="Male"
                    )
                    history_input = gr.Textbox(
                        label="Medical History",
                        placeholder="Previous conditions, allergies, medications or None",
                        lines=3
                    )
                    plan_btn = gr.Button("Generate Treatment Plan")

                with gr.Column():
                    plan_output = gr.Textbox(label="Personalized Treatment Plan", lines=20)

            plan_btn.click(treatment_plan, inputs=[condition_input, age_input, gender_input, history_input], outputs=plan_output)
```


Welcome To Colab - Colab
Health_AI.py - Colab
Colab Notebooks - Google Dri
colab.research.google.com/drive/1rp6R7Ovgl_hQC3Um9S2xXpSCHM6e77pg
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
app.launch(share=True)
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret i
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 410kB/s]
vocab.json: 777k/? [00:00<00:00, 5.26MB/s]
merges.txt: 442k/? [00:00<00:00, 13.7MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 45.1MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.17kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 10.3kB/s]
config.json: 100% 786/786 [00:00<00:00, 31.4kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.80MB/s]
Fetching 2 files: 0% 0/2 [00:00<?, ?it/s]
model-00001-of-00002.safetensors: 24% 1.22G/5.00G [00:27<02:08, 29.4MB/s]
Variables
Terminal
Executing (1m 11s)
T4 (Python 3)


Welcome To Colab - Colab
Health_AI.py - Colab
Colab Notebooks - Google Dri
colab.research.google.com/drive/1rp6R7Ovgl_hQC3Um9S2xXpSCHM6e77pg
Commands
+ Code
+ Text
Run all
RAM
Disk
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2a8de8b5f3ca660278.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 Hugging Face Spaces
Medical AI Assistant
Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.
Disease Prediction
Treatment Plans
Enter Symptoms
e.g., fever, headache, cough, fatigue...
Possible Conditions & Recommendations
Analyze Symptoms
Variables
Terminal
5:31PM
T4 (Python 3)


Welcome To Colab - Colab
Health_AI.py - Colab
Gradio
Colab Notebooks - Google Dri
2a8de8b5f3ca660278.gradio.live
Medical AI Assistant
Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.
Disease Prediction
Treatment Plans
Enter Symptoms
Headache
Analyze Symptoms
Possible Conditions & Recommendations
1. **Tension Headache:**
- Description: Common, mild to moderate pain on both sides of the head, often described as a tight band or band-like pressure.
- Treatment: Over-the-counter pain relievers, such as ibuprofen (Advil, Motrin) or acetaminophen (Tylenol), can be effective. For chronic or frequent tension headaches, a healthcare provider may recommend lifestyle modifications, such as regular exercise, stress management techniques, and a consistent sleep schedule.
2. **Migraine:**
- Description: Often accompanied by nausea, vomiting, and sensitivity to light and sound. Pain can be severe.
- Treatment: Preventive medications may be prescribed by a doctor, such as beta-blockers, blood pressure medications, or antidepressants. Acute treatment options include triptans (e.g., Imitrex) and non-steroidal anti-inflammatory drugs (NSAIDs) like ibuprofen. Always consult a healthcare provider for appropriate migraine management.
3. **Sinus Headache:**
- Description: Pain that intensifies with head movement, often due to inflammation or congestion in the sinus area.
- Treatment: Anti-inflammatory drugs like ibuprofen or naproxen can help alleviate sinus headache pain. Decongestants and nasal steroid sprays may also be beneficial. Treating the underlying sinus infection is crucial.
Use via API
Built with Gradio
Settings

12. Known Issues

While *Health AI* successfully demonstrates the integration of **IBM Granite LLM, Gradio interface, and healthcare-focused AI prompts**, several limitations were identified during development and testing. These issues are important to acknowledge for transparency and to guide future enhancements.

12.1 Model Limitations

- The **IBM Granite LLM** provides **general healthcare information** but is **not domain-specialized** for clinical medicine.
- Generated outputs may vary in accuracy and depth depending on input phrasing.
- The model cannot guarantee **clinically validated or evidence-based recommendations**.

12.2 Data & Input Handling

- Inconsistent or incomplete symptom descriptions can lead to **ambiguous or overly broad outputs**.
- Very long text inputs risk **truncation** due to token length limitations.
- Lack of structured medical data (e.g., lab test results, vital sign streams) limits **forecasting and anomaly detection** features in the current version.

12.3 User Interface Constraints

- The **Gradio interface** is functional but has limited customization compared to frameworks like Streamlit or React-based dashboards.
- Visualization features (charts, KPIs, anomaly alerts) are currently minimal.
- No integrated export options beyond copy-paste; formal PDF/CSV report download is not yet available.

12.4 Performance Issues

- Model inference on **CPU-only environments** is slow, affecting user experience.
- Reliance on **Hugging Face model downloads** may cause delays during initial setup.
- Limited optimization for handling **concurrent users**; designed primarily for single-user interaction.

12.5 Security & Authentication

- The current version is deployed in an **open environment without authentication**.
- No encryption or role-based access control for sensitive medical inputs.
- Makes the system suitable for demonstration purposes only, not for production use.

13. Future Enhancements

The current version of *Health AI* serves as a **prototype and proof of concept**, demonstrating the potential of generative AI in healthcare assistance. To evolve into a **production-grade intelligent healthcare system**, the following enhancements are planned:

13.1 Model and AI Improvements

- **Domain-Specific Fine-Tuning**
    - Train or fine-tune IBM Granite models on **healthcare-specific datasets** to improve accuracy, reliability, and domain expertise.
- **Multilingual Support**
    - Extend the system to handle **regional languages**, enabling wider accessibility for diverse patient populations.
- **Expanded Use Cases**
    - Add new modules such as:
        - *Diet and Nutrition Planner*
        - *Mental Health Wellness Guidance*
        - *Medication Reminder & Tracker*

13.2 Advanced Analytics

- **KPI Dashboards**
    - Introduce real-time visualizations of patient metrics (vital signs, glucose levels, etc.) through charts and graphs.
- **Anomaly Detection Upgrade**
    - Implement advanced ML and deep learning algorithms (e.g., LSTMs, autoencoders) for **early health risk detection**.
- **Predictive Forecasting**
    - Enhance forecasting models to support **patient flow predictions** in hospitals and **resource utilization planning**.

13.3 User Interface Enhancements

- **Rich Visual Dashboards**
  - Replace or extend Gradio with **Streamlit, React, or Flutter** for advanced UI capabilities.
- **Report Generation**
  - Enable automated **PDF/CSV exports** of disease predictions, treatment plans, and KPI summaries.
- **Mobile Integration**
  - Develop a **cross-platform mobile app** for patient convenience and real-time health monitoring.

13.4 Security and Compliance

- **Authentication & Access Control**
  - Implement **JWT/OAuth2 authentication** and **role-based access control** (Patients, Doctors, Admins).
- **Data Privacy**
  - Ensure compliance with healthcare regulations such as **HIPAA (U.S.)** and **GDPR (EU)**.
- **Secure Cloud Deployment**
  - Host on **IBM Cloud** or other HIPAA-compliant cloud platforms with encryption for sensitive data.

13.5 Integration with Healthcare Systems

- **Electronic Health Records (EHR) Integration**
  - Connect with existing hospital EHR systems for **automatic retrieval and analysis** of patient history.
- **IoT & Wearables Support**
  - Incorporate real-time health data from **wearables (smartwatches, glucose monitors, BP monitors)**.
- **Telemedicine Integration**
  - Provide **doctor-patient video consultations** integrated into the AI assistant.