HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Project Documentation

1. Introduction

Project Title: HealthAI: Intelligent Healthcare Assistant

Team Members:

Team Leader : Nikku Chandana Sai Durga

Team member : Mudili Lakshmi Srimannarayana Charan Teja

Team member : Muppalla Sindhu

Team member : Motukuri Shrivalli

2. Project Overview

Purpose: HealthAI harnesses IBM Watson Machine Learning and Generative AI to provide intelligent healthcare assistance, offering users accurate medical insights. The primary goal is to improve accessibility to healthcare information by delivering personalized and data-driven medical guidance.

Features:
The HealthAI platform includes four key features:

- **Patient Chat:** A chat-style interface for answering health-related questions, providing clear, empathetic responses with relevant medical facts, acknowledging limitations, and suggesting when to seek professional medical advice.
- **Disease Identifier:** Evaluates user-reported symptoms (e.g., severe headache, fatigue, mild fever) along with patient profile and health data to provide potential condition predictions, including likelihood assessments and recommended next steps.
- **Home Remedies:** Suggests natural remedies for symptoms or known diseases in a Easy-to-read, bullet-point formatted results.
- **Treatment Plan Generator:** Generates personalized, evidence-based medical recommendations for a diagnosed condition, including medications, lifestyle modifications, and follow-up testing.

- **Health Dashboard:** Visualizes vital signs over time (heart rate, blood pressure, blood glucose) and provides AI-generated insights about the potential health concerns and improvement recommendations. Plots Health trends using Chart.js

- **Health Tips:** Generates 6 personalised daily health Tips, displayed in a seperate cards using a clean UI which helps in enhancing the user health.

3. Architecture

The HealthAI application follows a modular and layered architecture, integrating FastAPI as the backend API layer, a Single page Application (SPA) for the Frontend, and IBM WatsonX Foundation Models (Granite 13B Chat V2) as the AI Engine.

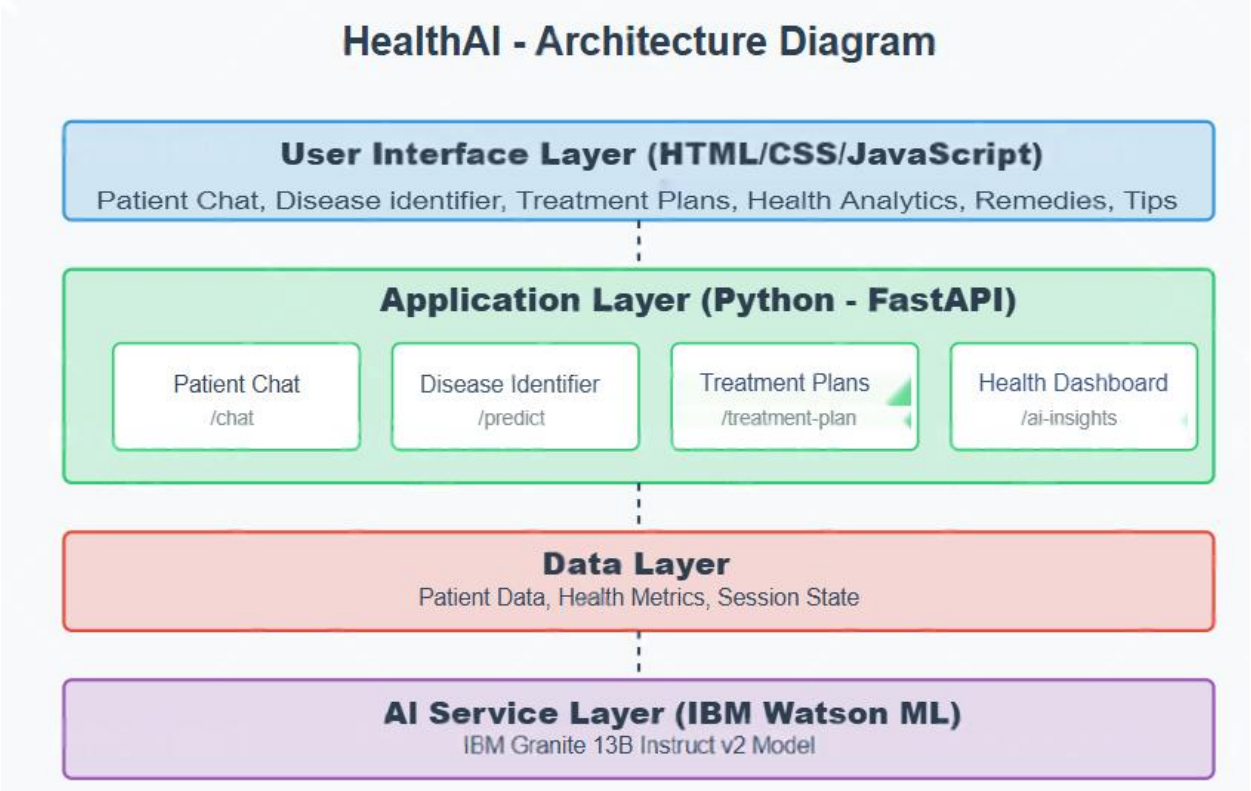
- **User Interface Layer (HTML/CSS/JavaScript):** This layer handles all user interactions and displays information. It consists of the Patient Chat, Disease Prediction, Treatment Plans, and Health Analytics interfaces.

- **Application Layer (FastAPI - Python):** Written in Python using the FastAPI framework, this layer contains the main application logic. It processes user inputs, manages session state, and acts as orchestrates for routing , user input processing, and calling AI services. Key routes include: /predict, /remedies, /tips, /chat, /treatment-plan, /ai-insights.

- **Data Layer:** Manages patient-related data, health metrics (like heart rate, blood pressure, blood glucose), and application session state. For this project, sample patient data is generated for demonstration.

- **AI Service Layer (IBM WatsonX Granite 13B v2):** This layer is powered by the IBM Granite 13B Instruct v2 model. It receives prompts from the Application Layer and generates intelligent, data-driven responses for medical guidance.

Architectural Diagram:



4. Setup Instructions

Prerequisites:

- Python: Python 3.8 or higher.
- Pip: Python package installer (comes with Python).
- IBM Cloud Account: With access to IBM Watson Machine Learning service.
- Virtual Environment: Recommended for dependency management.
- Git: For version control.

Installation:

- **Create a Virtual Environment:**

  python -m venv venv

- **Activate the Virtual Environment:**
  - **Windows:** .\venv\Scripts\activate

- **Install Required Libraries:**

  pip install fastapi uvicorn python-dotenv ibm-watsonx-ai jinja2

- **Set Up Environment Variables (**.env **file):**
  - Create a file named .env in the root of your project directory (HealthAI/).

  - Obtain your apikey and url from your IBM Watson Machine Learning service credentials in IBM Cloud.

  - Add the following lines to your .env file, replacing the placeholders with your actual credentials:

    IBM_API_KEY="YOUR_ACTUAL_API_KEY"

    IBM_GRANITE_ENDPOINT="YOUR_ACTUAL_WATSON_ML_URL"

    IBM_MODEL_ID="YOUR_ACTUAL_IBM_MODEL_ID"

    IBM_PROJECT_ID="YOUR_ACTUAL_IBM_PROJECT_ID"

Ensure there are no spaces around the = sign and values are in double quotes. This securely manages your API credentials.

5. Folder Structure

The project has a straightforward structure suitable for FastAPI applications:

- HealthAI/ **(Root Directory)**
- .env: Contains environment variables for IBM API key, endpoint, project ID, model ID.
- main.py: The Main FastAPI application with all backend routes and AI integration.
- venv/: The Python virtual environment directory (created during setup).
- templates/: Main Single Page Application interface and Health Analytics panel content (integrated inside SPA layout)
- static/: UI styling for all components and JavaScript logic: navigation, fetch API calls, dynamic updates.

6. Running the Application

To run the HealthAI application locally:

- **Open your terminal or command prompt.**
- **Navigate to the project's root directory (**HealthAI/**):**
  C:\Users\91994\Documents\HealthAIFinal\HealthAI
- **Activate your virtual environment:**
  - **Windows:** .\venv\Scripts\activate
- **Start the FastAPI server using Uvicorn:**
  uvicorn main:app --reload

This command will launch the application in your default web browser (usually at http://127.0.0.1:8000/).

7. API Documentation

The backend "API" in this project is implemented using **FastAPI** and primarily interfaces with the **IBM Watson Machine Learning** service. There are no third-party microservices or external backend platforms (like Node.js). Instead, the application directly communicates with IBM's foundation model via secure internal function calls.

The primary "API" calls are made internally to the IBM Watson Machine Learning service:

- **Service**: IBM Watson Machine Learning
- **Model Used**: IBM Granite 13B Instruct v2 (ibm/granite-13b-instruct-v2)
- **Authentication**: Managed securely via .env file and loaded using python-dotenv
- **Libraries Used**: ibm-watsonx-ai, fastapi, uvicorn, jinja2, python-dotenev

Internal AI Interactions (via main.py):

- **Patient Chat:**

- **Route**: /chat (POST)
- **Function Call**: generate_text(prompt=chat_prompt)
- **Prompt Structure (Example)**:

  You are a smart and helpful health assistant.

  Respond to the user's health questions clearly and accurately.

  User: {user_query}
- **Output Behavior**:
  - Direct, empathetic, medically informed
  - Avoids self-diagnosis
  - Encourages follow-up with a real doctor when appropriate
  - Displayed as chatbot bubbles in the frontend.

- **Disease Prediction:**

- **Route**: /predict (POST)
- **Function Call**: generate_text(prompt=prediction_prompt)
- **Prompt Structure (Example)**:

  Symptoms: {symptom_input}

What disease could it be? Give 3 lines of explanation also in bullet points.

- **Output Behavior**:
  - Three-point condition explanation
  - Rendered in bullets using 👉 icon
  - Example:
    - 👉 Likely viral infection
    - 👉 Symptom cluster indicates dehydration
    - 👉 Consult physician if persists

➢ **Home Remedies:**

- **Route**: /remedies (POST)
- **Function Call**: generate_text(prompt=remedy_prompt)
- **Prompt Structure (Example)**:

  Disease or Symptoms: {user_input}

  Give a list of natural remedies (max 8) in points, line by line.
- **Output Behavior**:
  - Stripped numbers from model output using regex
  - Each point rendered on a new line or inside styled card blocks

➢ **Treatment Plan Generator:**

- **Route**: /treatment (POST)
- **Function Call**: generate_text(prompt=treatment_prompt)
- **Prompt Structure (Example)**:

  As a medical AI assistant, generate a personalized treatment plan for the following scenario:

  Patient Profile:

  - Condition: {condition}

  - Age: {age}

  - Gender: {gender}

  - Medical History: {history}

  Create a comprehensive, evidence-based treatment plan that includes:

1. Medications (with dosage)

2. Lifestyle changes

3.Follow-up tests

- **Output Behavior**:
  - Structured in numbered or bullet points
  - Cards displayed in the "Treatment" panel

➢ **Health Analytics Insights:**

- **Route**: /analytics (POST)
- **Function Call**: generate_text(prompt=analytics_prompt)
- **Prompt Structure (Example)**:

  Based on the following patient metrics:

  - Heart Rate: {data}

  - BP: {data}

  - Glucose: {data}

  Summarize health trends and provide actionable AI insights or warnings.

- **Output Behavior**:
  - Generated summary in 4–5 lines
  - Rendered under the chart section
  - Tips about fluctuations, averages, or medical alerts

➢ **Health Tips Generator:**

- **Route**: /tips (GET)
- **Function Call**: generate_text(prompt="Give 6 health tips")
- **Prompt Structure (Example)**:

  Give 6 general daily health tips.

  Present them in short, easy-to-read sentences.

- **Output Behavior**:
  Rendered as hoverable health tip cards

One card per tip using the 💡 symbol

8. Authentication

Authentication with the **IBM Watson Machine Learning** service is securely handled using an **API Key-based approach**.

➢ **IBM Cloud API Key** is obtained from the IBM Watson Machine Learning service instance.

- It is stored securely in a .env file located at the project root to avoid hardcoding sensitive credentials.
- The python-dotenv package is used to **load environment variables** into the FastAPI application at runtime.

9. User Interface

The UI is a **Single Page Application (SPA)** built using **HTML, CSS, and vanilla JavaScript**, running within a FastAPI backend.

❖ Main Layout

- A **top navbar** displays the app title: **"HealthAI: Intelligent Healthcare Assistant"**.
- A **sidebar** provides navigation to core features: Patient Chat, Symptom Predictor, Remedies, Tips, Analytics, and Treatment Plan.
- The design uses **responsive layouts**, modern **card-based components**, and **blurred background effects** for a clean look.

❖ Feature Interfaces

- **Patient Chat**: WhatsApp-style chat UI. User messages appear on the left, bot replies on the right with preserved history.
- **Symptom Predictor**: Text input for symptoms. The AI returns possible conditions with explanations and next steps.
- **Remedies**: Accepts symptom/disease input. Displays 6–8 natural remedies as clean bullet-point tips in individual cards.
- **Treatment Plan**: Inputs include condition, age, gender, and history. Returns medications, lifestyle tips, and follow-up care in a structured format.
- **Analytics Dashboard**: Users enter vitals (HR, BP, glucose). Visualized via **Chart.js**, with a "Generate Insights" button for AI analysis.
- **Health Tips**: One-click generates 6 daily wellness tips, each shown in a hoverable card with a 💡 icon.

❖ Visual Elements

- **Chart.js** is used for line graphs (heart rate, BP, glucose).
- **CSS** animations, **emoji markers**, and responsive design improve UX.

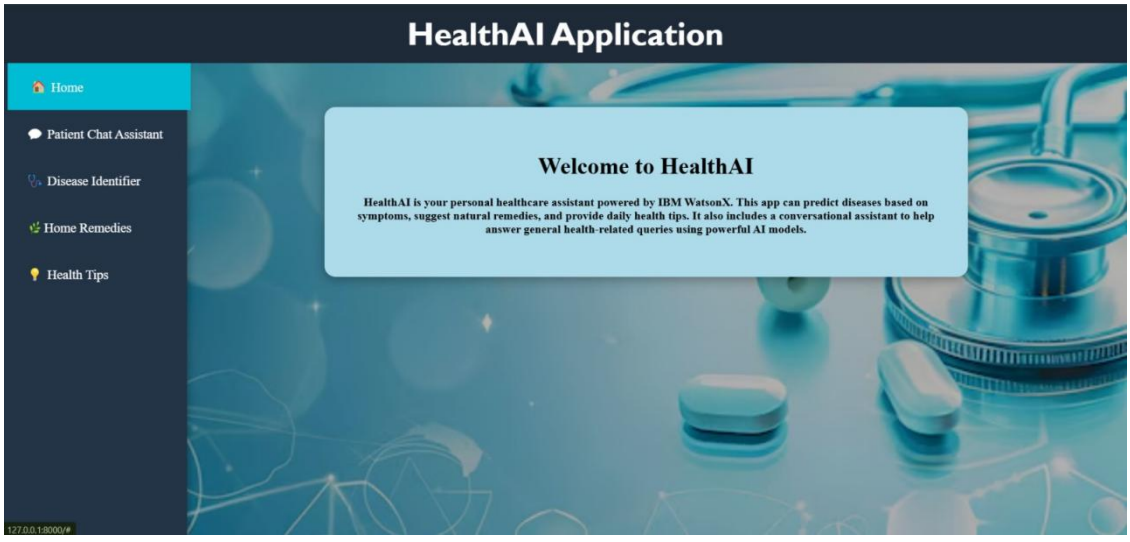
- **Result boxes** and cards are styled with shadows, hover effects, and readable fonts.

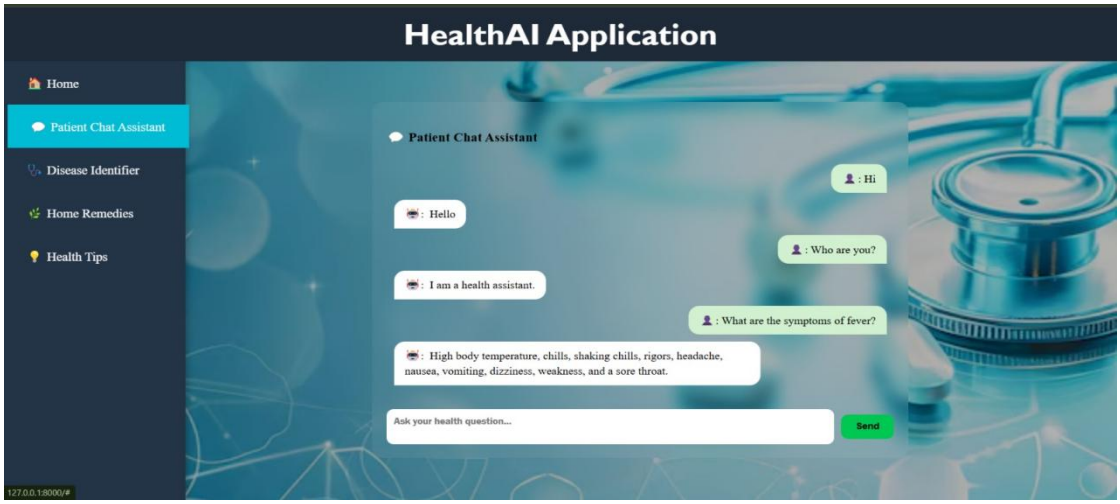
10. Testing

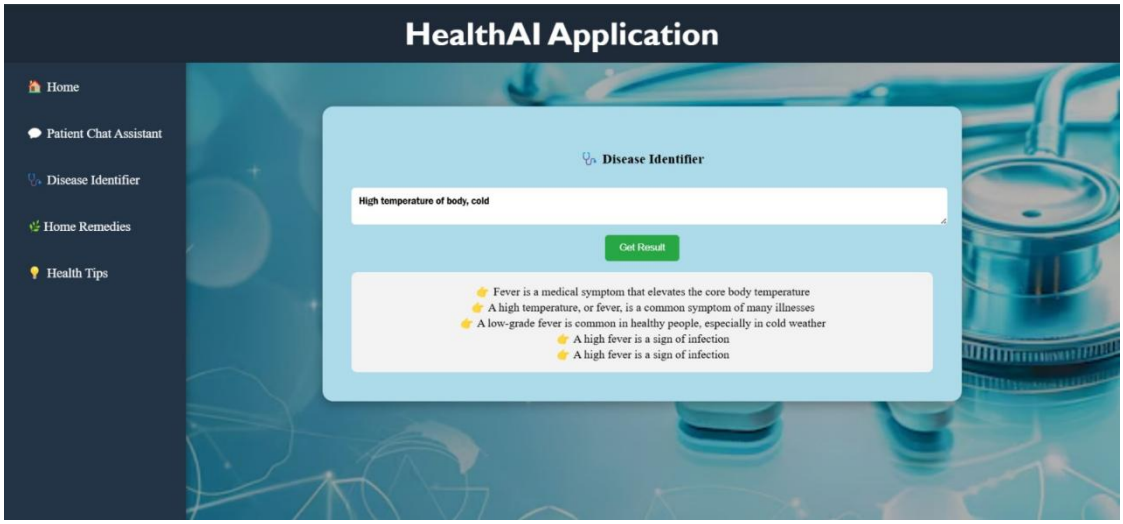
The testing strategy for the HealthAI application includes the following components:

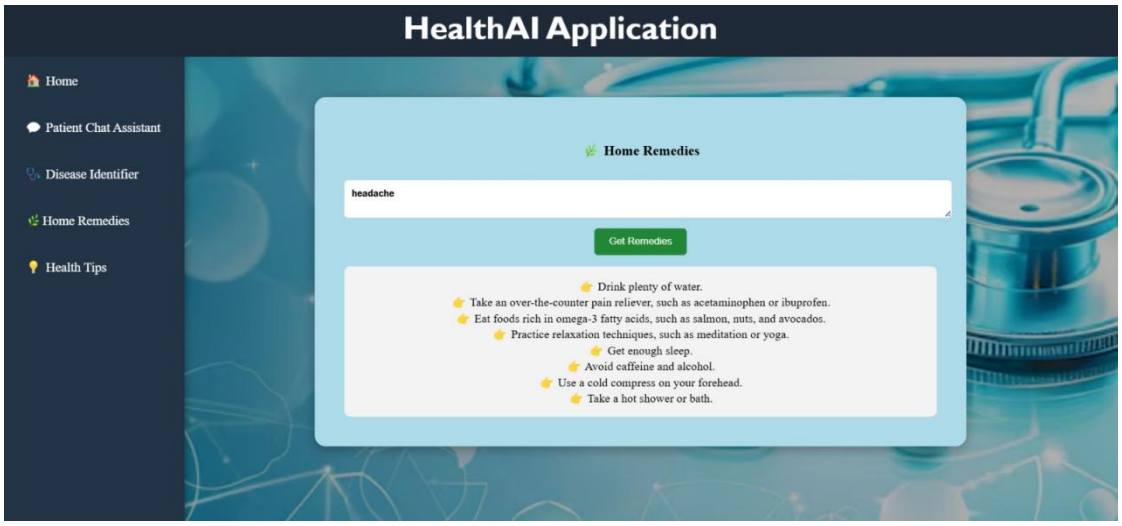
- **Unit Testing (Conceptual):** Test core utility functions such as query_model() to ensure correct prompt formatting and expected response structure from the IBM Granite model.
- **Integration Testing:** Verify interactions between the FastAPI backend and the HTML/CSS/JS frontend, including form submissions and fetch API calls. Ensure that routes like /predict, /chat, /remedies, and /tips return properly structured responses.
- **User Acceptance Testing (UAT):** Manually test each feature to ensure it meets expected functionality and user experience standards. Confirm that AI-generated responses are clear, medically informative, and appropriately styled.
- **Error Handling Tests:** Validate behavior when environment variables or API credentials are missing or incorrect. Test edge cases such as empty input or unexpected model responses to ensure the system handles errors gracefully and provides helpful feedback.

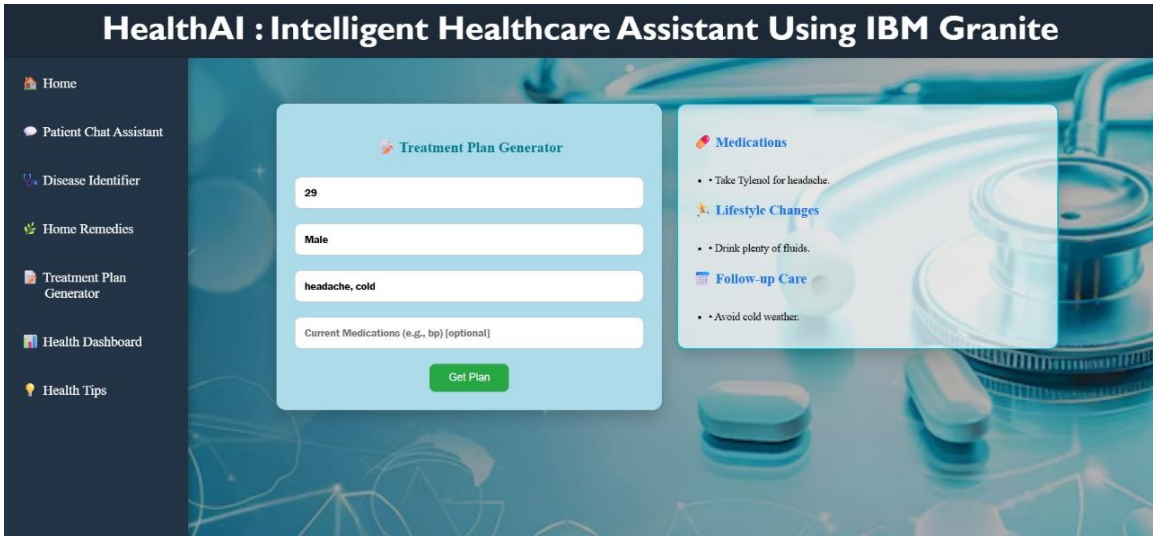
11. Screenshots

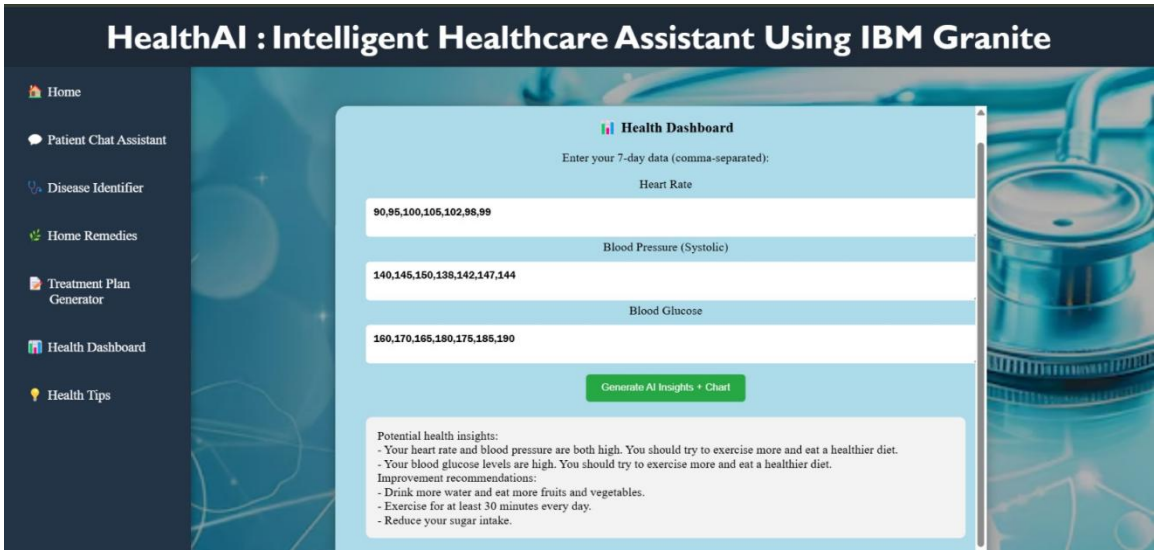


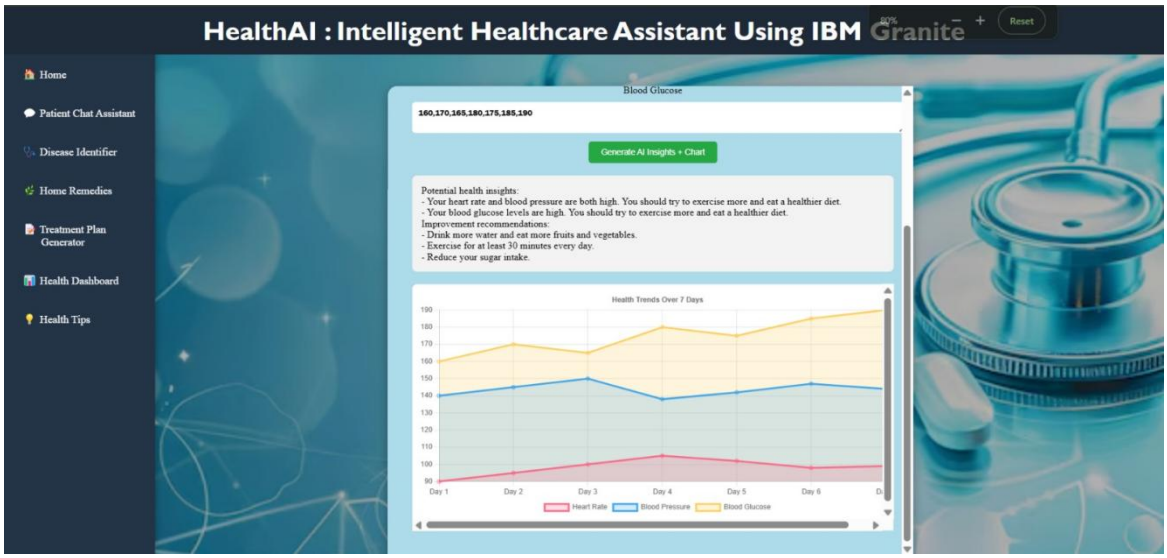

HealthAI Application
Home
Patient Chat Assistant
Disease Identifier
Home Remedies
Health Tips
Patient Chat Assistant
: Hi
: Hello
: Who are you?
: I am a health assistant.
: What are the symptoms of fever?
: High body temperature, chills, shaking chills, rigors, headache, nausea, vomiting, dizziness, weakness, and a sore throat.
Ask your health question...
Send
127.0.0.1:8000/#

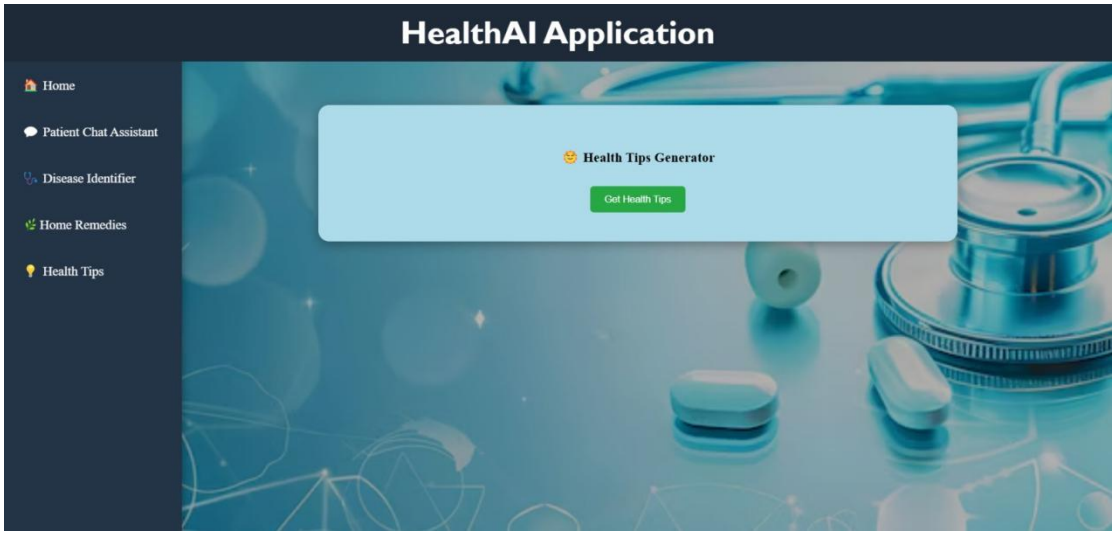

HealthAI Application
Home
Patient Chat Assistant
Disease Identifier
Home Remedies
Health Tips
Disease Identifier
High temperature of body, cold
Get Result
Fever is a medical symptom that elevates the core body temperature
A high temperature, or fever, is a common symptom of many illnesses
A low-grade fever is common in healthy people, especially in cold weather
A high fever is a sign of infection
A high fever is a sign of infection

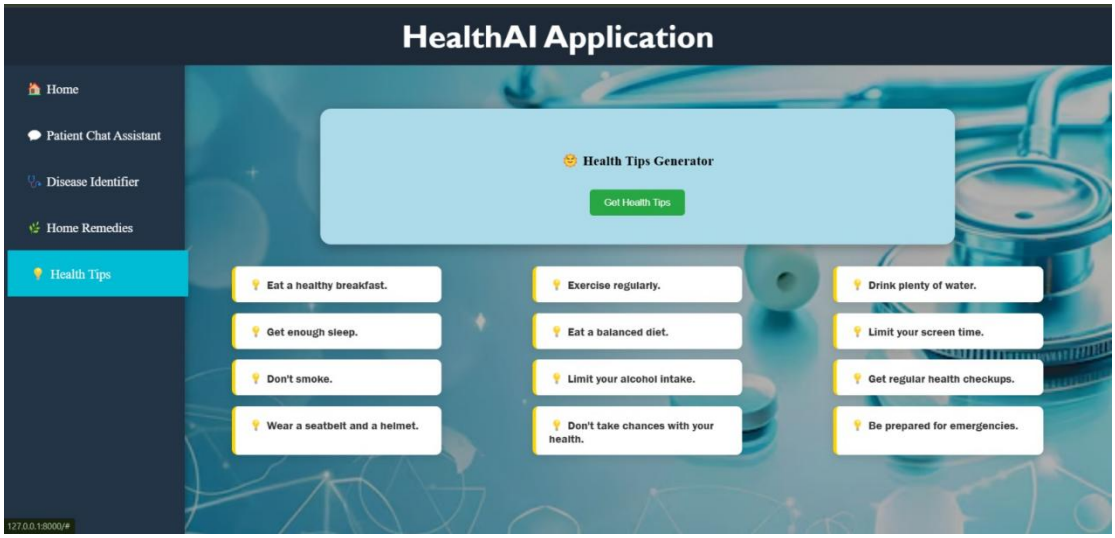

HealthAI Application
Home
Patient Chat Assistant
Disease Identifier
Home Remedies
Health Tips
Home Remedies
headache
Get Remedies
Drink plenty of water.
Take an over-the-counter pain reliever, such as acetaminophen or ibuprofen.
Eat foods rich in omega-3 fatty acids, such as salmon, nuts, and avocados.
Practice relaxation techniques, such as meditation or yoga.
Get enough sleep.
Avoid caffeine and alcohol.
Use a cold compress on your forehead.
Take a hot shower or bath.


HealthAI : Intelligent Healthcare Assistant Using IBM Granite
Home
Patient Chat Assistant
Disease Identifier
Home Remedies
Treatment Plan Generator
Health Dashboard
Health Tips
Treatment Plan Generator
29
Male
headache, cold
Current Medications (e.g., bp) [optional]
Get Plan
Medications
• Take Tylenol for headache.
Lifestyle Changes
• Drink plenty of fluids.
Follow-up Care
• Avoid cold weather.


HealthAI : Intelligent Healthcare Assistant Using IBM Granite
Home
Patient Chat Assistant
Disease Identifier
Home Remedies
Treatment Plan Generator
Health Dashboard
Health Tips
Health Dashboard
Enter your 7-day data (comma-separated):
Heart Rate
90,95,100,105,102,98,99
Blood Pressure (Systolic)
140,145,150,138,142,147,144
Blood Glucose
160,170,165,180,175,185,190
Generate AI Insights + Chart
Potential health insights:
- Your heart rate and blood pressure are both high. You should try to exercise more and eat a healthier diet.
- Your blood glucose levels are high. You should try to exercise more and eat a healthier diet.
Improvement recommendations:
- Drink more water and eat more fruits and vegetables.
- Exercise for at least 30 minutes every day.
- Reduce your sugar intake.


HealthAI : Intelligent Healthcare Assistant Using IBM Granite
Reset
Home
Patient Chat Assistant
Disease Identifier
Home Remedies
Treatment Plan Generator
Health Dashboard
Health Tips
Blood Glucose
160,170,165,180,175,185,190
Generate AI Insights + Chart
Potential health insights:
- Your heart rate and blood pressure are both high. You should try to exercise more and eat a healthier diet.
- Your blood glucose levels are high. You should try to exercise more and eat a healthier diet.
Improvement recommendations:
- Drink more water and eat more fruits and vegetables.
- Exercise for at least 30 minutes every day.
- Reduce your sugar intake.
Health Trends Over 7 Days
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Heart Rate
Blood Pressure
Blood Glucose





12. Known Issues

- **Sample Data Only:** The application currently uses hardcoded or dummy health metrics for charts and analytics. Real-world deployment requires integration with persistent storage
- **Rate Limitations:** The IBM WatsonX service may impose limits on the number of API calls depending on your IBM Cloud plan, which can affect app performance under high load
- **Model Generalization:** The IBM Granite 13B model provides general responses and is not specialized for clinical use. Responses should not be treated as definitive medical advice.
- **No Authentication or User History:** There's no user login or saved chat/history. All inputs and outputs are session-based and lost on refresh.

13. Future Enhancements

- **Database Integration:** Connect a database (e.g., MongoDB or PostgreSQL) to store user details, health data, and chat history.
- **User Authentication:** Implement secure login/signup features with JWT or OAuth2 to support personalized and protected data access.
- **Real-Time Health Data Input:** Build forms or integrations to accept real patient vitals (e.g., via smart devices or manual entry) for analytics.
- **Expanded Analytics:** Use machine learning or statistical techniques to detect abnormal trends and give predictive warnings.
- **Health Alerts & Recommendations:** Enable personalized alert systems and suggest actions when metrics cross thresholds (e.g., high BP).
- **Treatment Plan Refinement:** Add more structured output formatting, dosage suggestions, and follow-up logic to the treatment module.
- **Mobile Optimization:** Improve responsiveness and layout on small screens for better mobile usability.
- **Voice Support:** Add optional voice input using Web Speech API for chat-based health interaction.
- **LLM Fine-tuning:** Explore fine-tuning IBM Granite or integrating other medically-tuned models like MedPaLM for higher clinical accuracy.