

HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Project Documentation

1. Introduction

Project Title: HealthAI: Intelligent Healthcare Assistant

Team Members:

Team Leader : Pavuluri Sai Lithan

Team member : Nischintala Jai Manikanta Chari

Team member : Niharika Sreekakulapu

Team member : P B V S N Mahitha

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 Prediction:** Evaluates user-reported symptoms (e.g., persistent headache, fatigue, mild fever) along with patient profile and health data to provide potential condition predictions, including likelihood assessments and recommended next steps.
- **Treatment Plans:** Generates personalized, evidence-based medical recommendations for a diagnosed condition, including medications, lifestyle modifications, and follow-up testing.

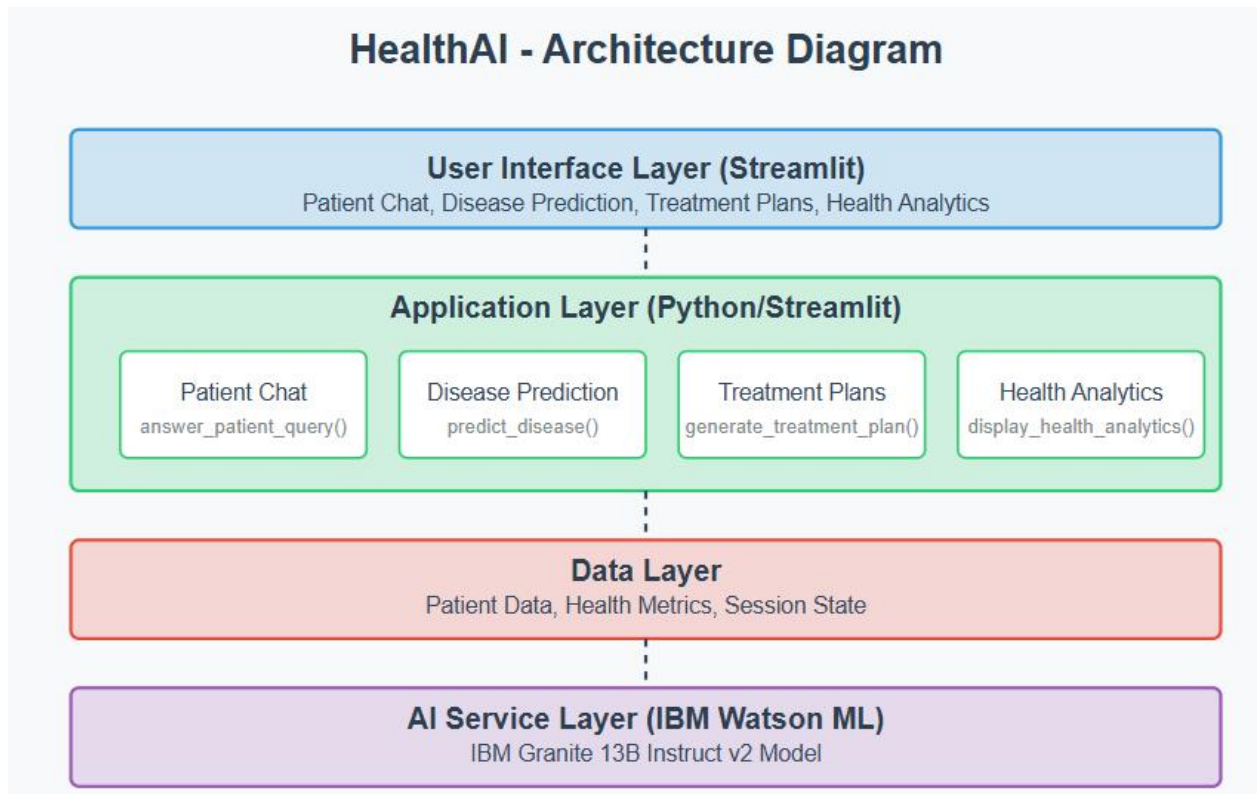
- **Health Analytics:** Visualizes vital signs over time (heart rate, blood pressure, blood glucose) and provides AI-generated insights about potential health concerns and improvement recommendations.

3. Architecture

The HealthAI application follows a layered architecture, integrating Streamlit for the user interface and application logic, with IBM Watson Machine Learning serving as the core AI service.

- **User Interface Layer (Streamlit):** 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 (Python/Streamlit):** Written in Python using the Streamlit framework, this layer contains the main application logic. It processes user inputs, manages session state, and orchestrates calls to the AI Service Layer. Key functions include `answer_patient_query()`, `predict_disease()`, `generate_treatment_plan()`, and `display_health_analytics()`.
- **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 Watson ML):** 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 streamlit pandas numpy plotly ibm-watson-machine-learning python-dotenv
```

➤ 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_WATSON_ML_API_KEY="YOUR_ACTUAL_API_KEY"
```

```
IBM_WATSON_ML_URL="YOUR_ACTUAL_WATSON_ML_URL"
```

- 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 Streamlit applications:

- **HealthAI/ (Root Directory)**
- **.env:** Contains environment variables for API keys and URLs.
- **app.py:** The main Streamlit application file, containing UI layout, feature logic, and AI integration calls. 15
- **utils.py:** Contains helper functions, including the `init_granite_model()` function for initializing the IBM Watson Machine Learning model, and functions to generate sample patient data and profiles.
- **venv/:** The Python virtual environment directory (created during setup).
- **data/:** (Optional, but recommended) A directory for any static data files or more complex patient data storage if implemented.

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\spava\OneDrive\Desktop\Ai
- **Activate your virtual environment:**
 - **Windows:** `.\venv\Scripts\activate`
- **Start the Streamlit server:**
`streamlit run app.py`

This command will open the application in your default web browser (usually at <http://localhost:8501>).

5. API Documentation

The backend "API" in this project is primarily the interface with the IBM Watson Machine Learning service. There are no traditional REST API endpoints exposed by a separate backend server (like Node.js/Express.js) as this is a Streamlit-based application directly interacting with the AI service.

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:** API Key and Endpoint URL, managed securely via .env file and loaded by python-dotenv.

Internal AI Interactions (through `utils.py` and `app.py`):

- **Patient Chat:**

- **Function:** `granite_model.generate_text(prompt=query_prompt)`
- **Prompt Structure (example):**

As a healthcare AI assistant, provide a helpful, accurate, and evidence-based response to the following patient question:

PATIENT QUESTION: {query}

Provide a clear, empathetic response that:

- Directly addresses the question
- Includes relevant medical facts
- Acknowledges limitations (when appropriate)
- Suggests when to seek professional medical advice
- Avoids making definitive diagnoses
- Uses accessible, non-technical language

RESPONSE:

- **Disease Prediction:**

- **Function:** `granite_model.generate_text(prompt=prediction_prompt)`

- Prompt Structure (example):

As a medical AI assistant, predict potential health conditions based on the following patient data:

Current Symptoms: {symptoms}

Age: {age}

Gender: {gender}

Medical History: {medical_history}

Recent Health Metrics:

- Average Heart Rate: {avg_heart_rate} bpm

- Average Blood Pressure: {avg_systolic_bp}/{avg_diastolic_bp} mmHg

- Average Blood Glucose: {avg_blood_glucose} mg/dL

- Recently Reported Symptoms: {symptoms}

Format your response as:

1. Potential condition name
2. Likelihood (High/Medium/Low)
3. Brief explanation
4. Recommended next steps

Provide the top 3 most likely conditions based on the data provided.

- **Treatment Plans:**

- **Function:** `granite_model.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: {medical_history}

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

1. Recommended medications (include dosage guidelines if appropriate)
2. Lifestyle modifications
3. Follow-up testing and monitoring
4. Dietary recommendations
5. Physical activity guidelines
6. Mental health considerations

Format this as a clear, structured treatment plan that follows current medical guidelines while being personalized to this patient's specific needs.

○

- **Health Analytics Insights:**

- **Function:** `granite_model.generate_text(prompt=analytics_prompt)`
- **Prompt Structure (example):** The prompt would be dynamically generated to summarize the displayed health metrics (Heart Rates, Systolic BPs, Diastolic BPs, Blood Glucoses) and ask the model to provide a brief summary of potential health observations or recommendations based on these trends.

6. Authentication

Authentication with the IBM Watson Machine Learning service is handled via an API Key.


- The API Key is obtained from the IBM Cloud account associated with the Watson Machine Learning service instance.
- It is securely stored as an environment variable in the `.env` file within the project directory.
- The `python-dotenv` library loads this API key at application startup.

- The `ibm_watson_machine_learning.credentials.Credentials` class uses this API key along with the service URL to authenticate with the IBM Watson Machine Learning API when initializing the `Model` object. This ensures secure communication without hardcoding sensitive information.

7. User Interface

The user interface is developed using the Streamlit framework, focusing on a responsive and intuitive design.

❖ Main Application Layout:

- Configurable page title ("HealthAI: Intelligent Healthcare Assistant"), icon (), and wide layout.
- A prominent sidebar for navigation between features and displaying patient profile details.
- Custom CSS can be added for enhanced visual appearance (though not explicitly implemented in the provided code snippets, it's a capability of Streamlit).

❖ Feature-Specific Interfaces:

- Patient Chat: A conversational chat-style interface with user input and message history display.
- Disease Prediction: An input form for symptoms, age, gender, and medical history, with a dedicated area for displaying prediction results.
- Treatment Plans: An input form for the diagnosed condition and patient details, with a structured output area for the generated treatment plan.
- Health Analytics: Features interactive charts for health metrics and a summary section for key indicators, along with AI-generated insights.

❖ Dynamic Visualizations (via Plotly):

- Health Metric Charts: Includes a heart rate trend line chart, a blood pressure dual-line chart (systolic and diastolic), and a blood glucose trend line chart (potentially with reference lines for normal ranges).
- Metrics Summary: Displays key health indicators with trend deltas from the previous day/period.

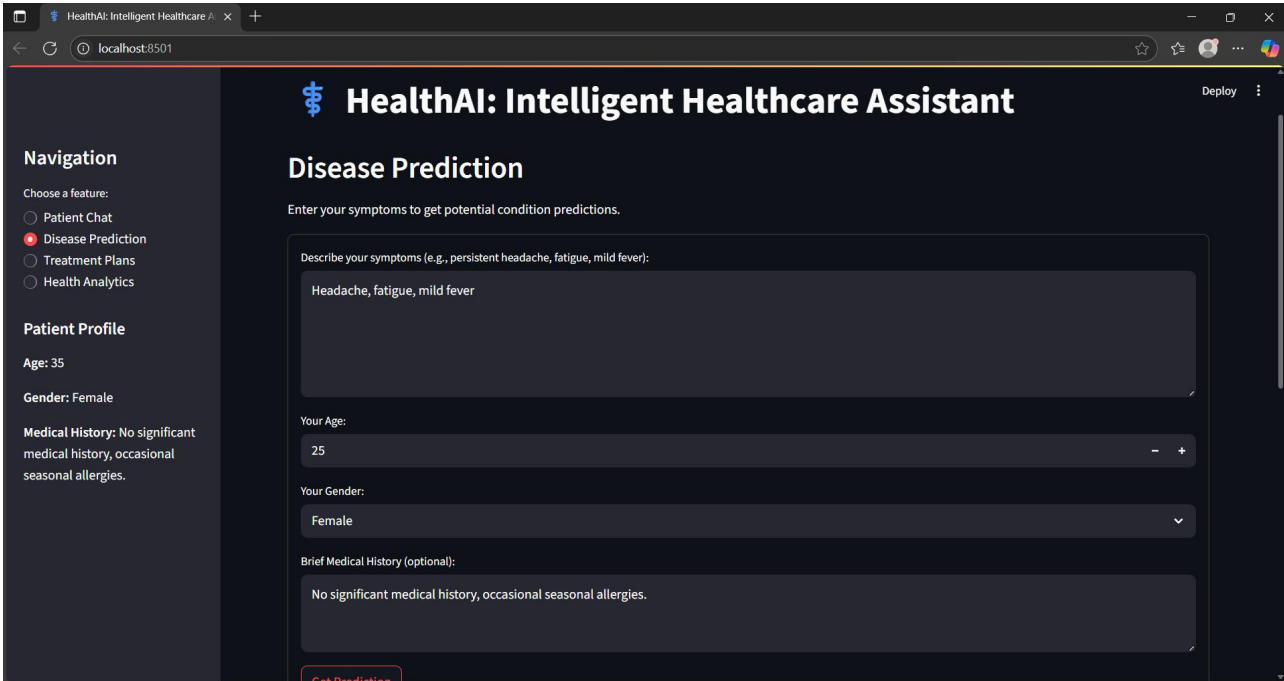
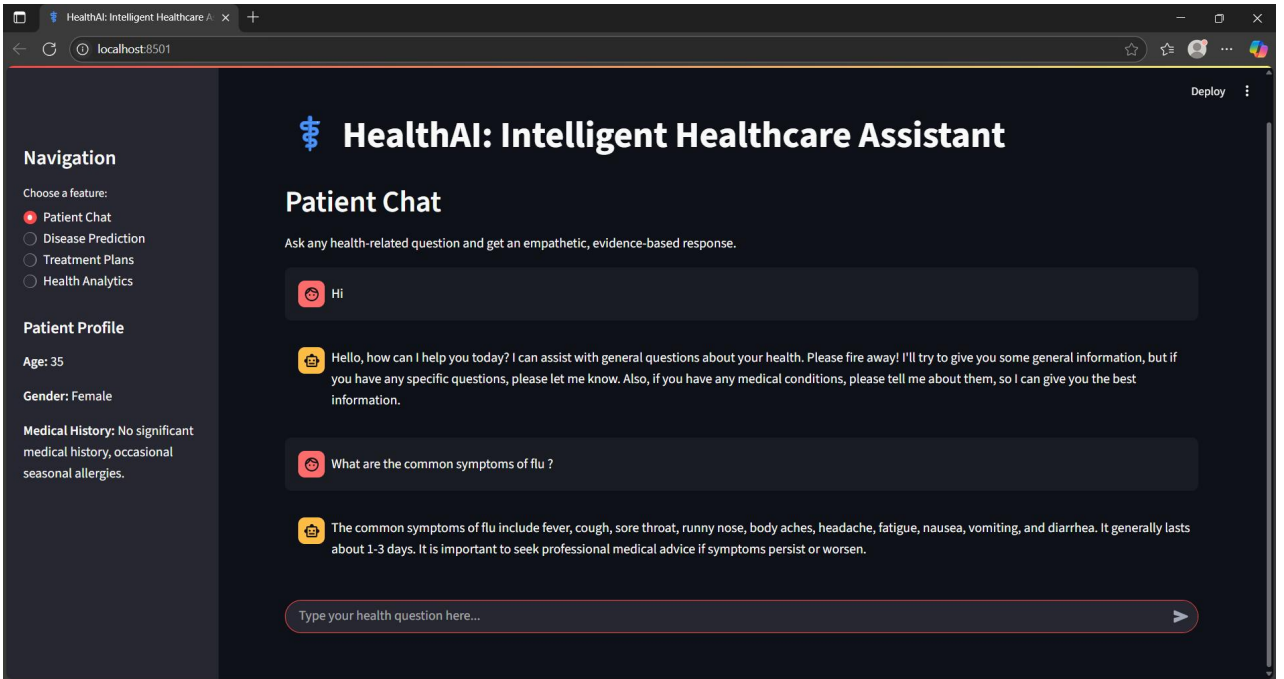
- Interactive tooltip information is supported by Plotly charts.
- A conceptual symptom frequency pie chart can be integrated if detailed symptom logging is implemented.

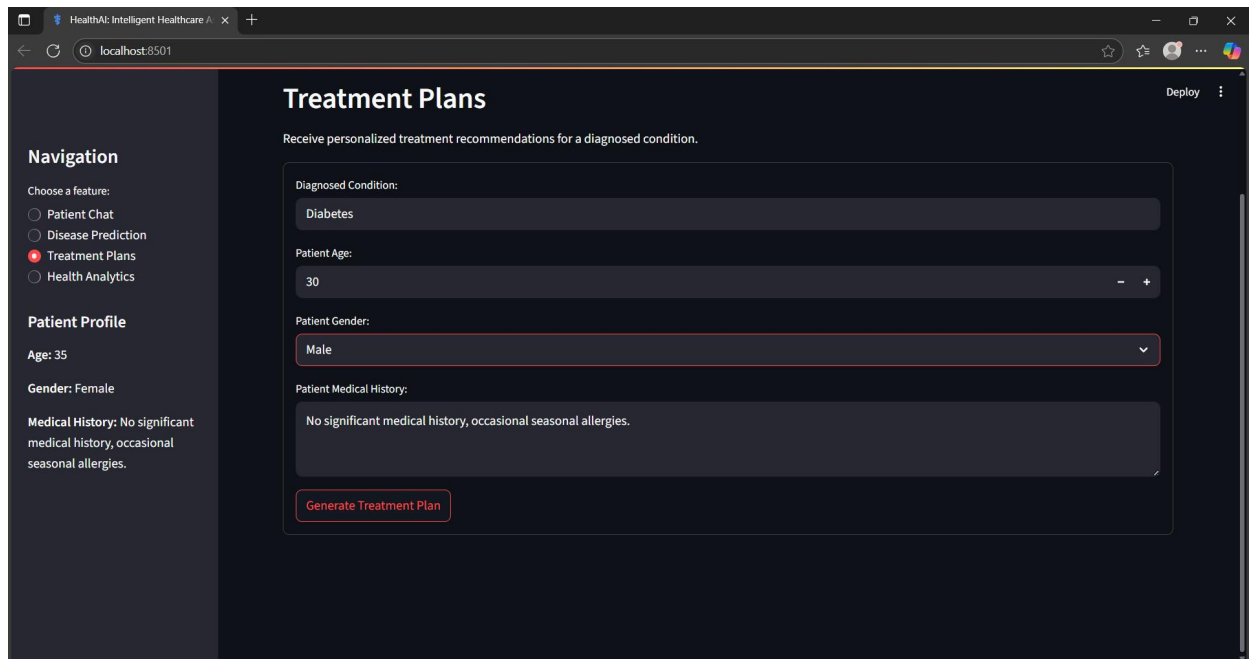
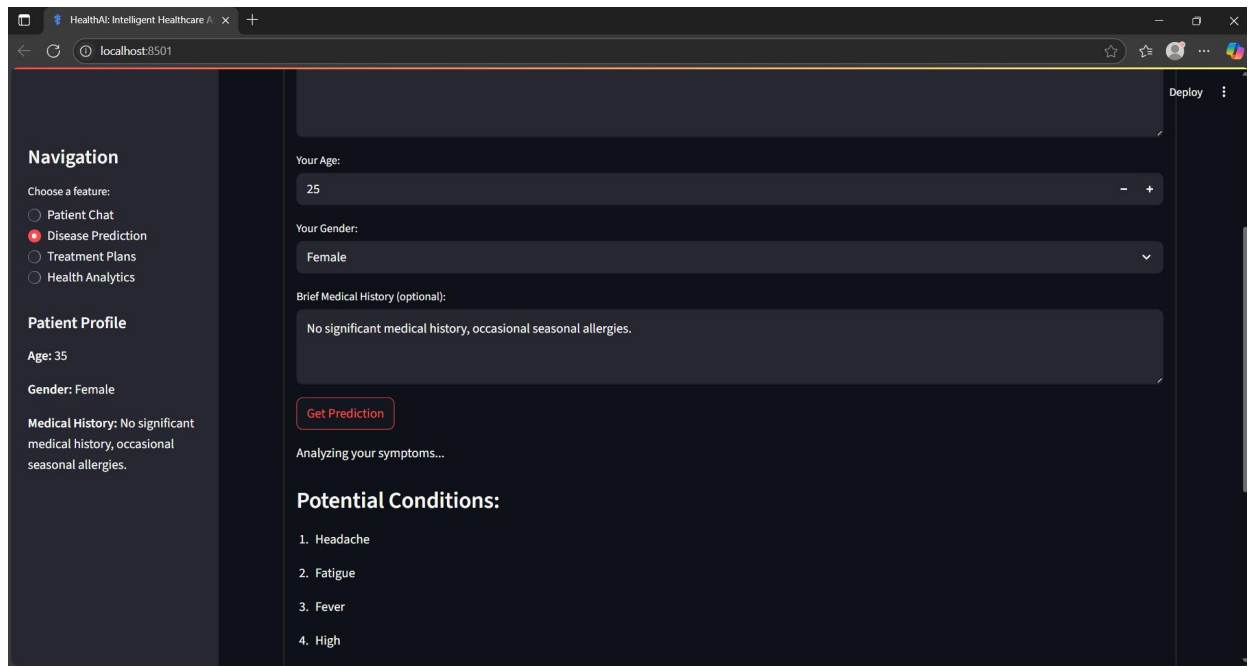
8. Testing

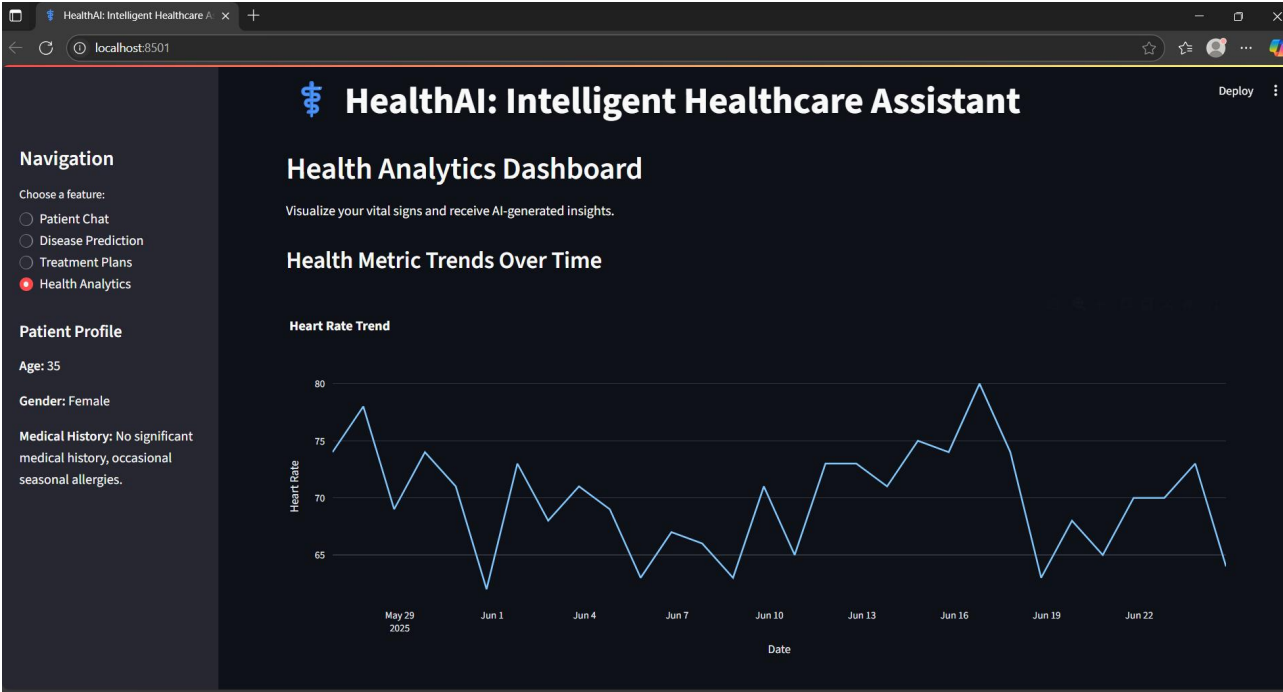
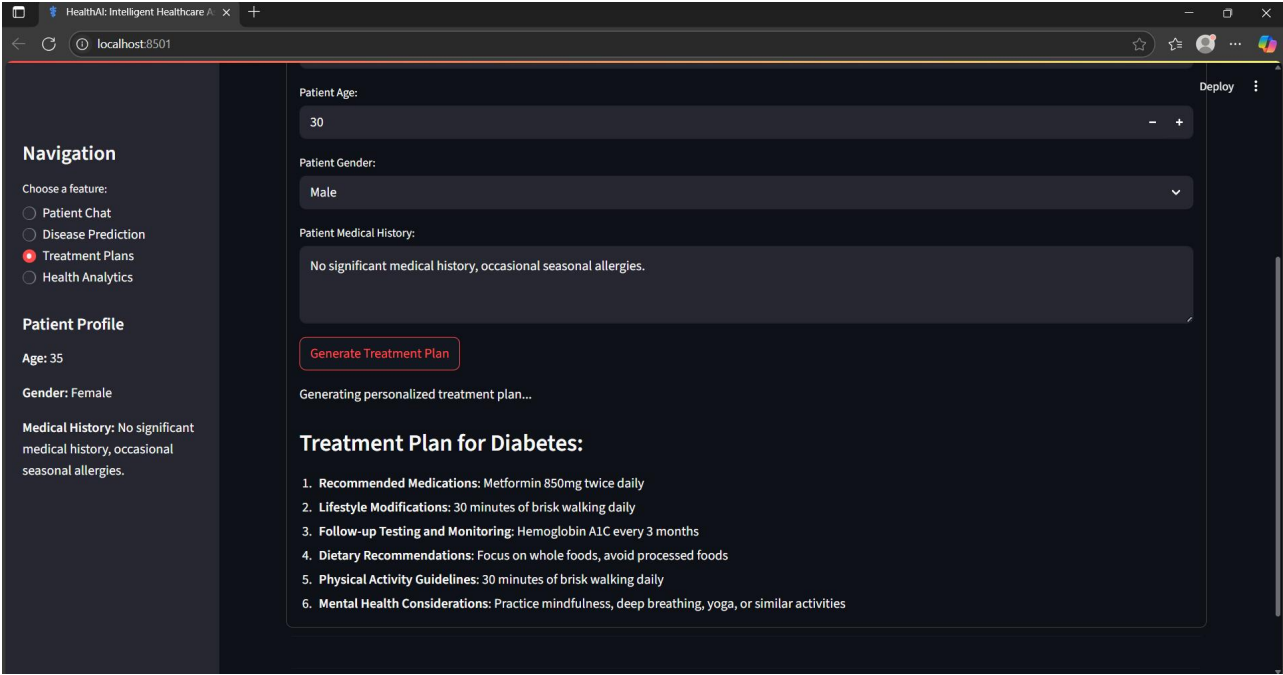
The project's testing strategy involves:

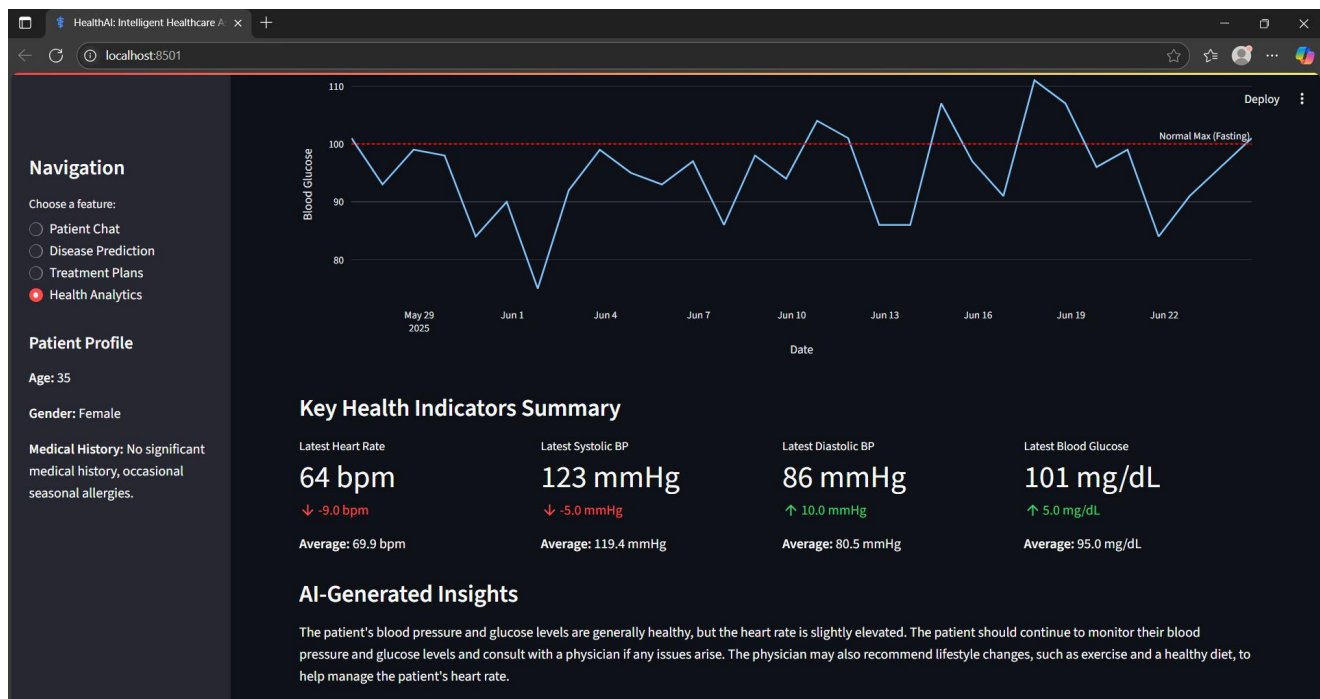
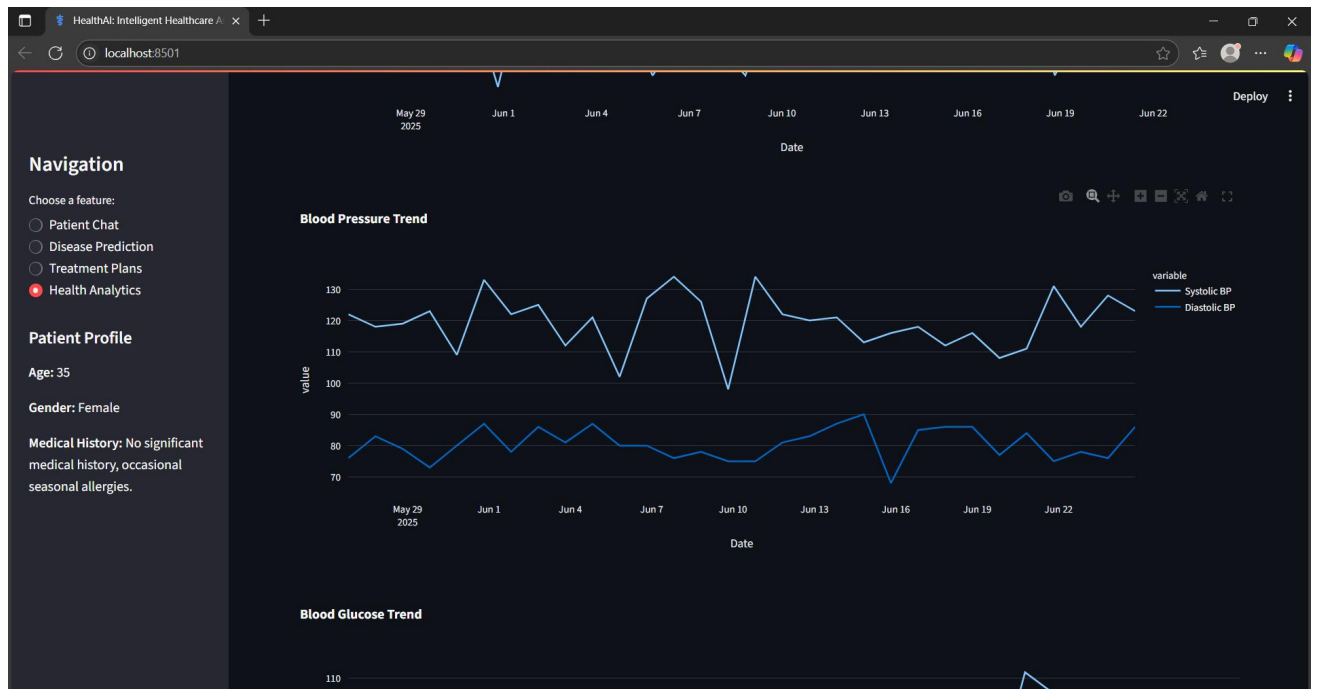
- **Unit Testing (Conceptual):** Testing individual functions within `utils.py` (e.g., `init_granite_model`, `get_sample_patient_data`) to ensure they return expected outputs.
- **Integration Testing:** Verifying the seamless interaction between the Streamlit frontend and the IBM Watson Machine Learning backend. This includes checking if prompts are correctly formatted and if responses from the Granite model are received and processed as expected for each feature.
- **User Acceptance Testing (UAT):** Manual testing by users to ensure the application meets the specified requirements and provides a satisfactory user experience, particularly for the empathetic and evidence-based nature of the AI responses.
- **Error Handling:** Testing the error handling mechanisms, especially for API connectivity issues or invalid credentials (e.g., the `try-except` block around `init_granite_model`).

9. Screenshots









10. Known Issues

- **Sample Data Limitation:** The current version uses randomly generated sample patient data. For a real-world application, this would need to be replaced with persistent storage and retrieval from a database.

- **Rate Limits:** Depending on your IBM Cloud plan, there might be API call rate limits for the Watson Machine Learning service, which could affect responsiveness under heavy usage.
- **Model Specificity:** While Granite-13b-instruct-v2 is a capable model, its responses are general-purpose. For highly specialized medical advice, fine-tuning or a more domain-specific model might be required, which is beyond the scope of this initial project.
- **No User Authentication/Persistence:** The current application does not include user login or persistent storage of individual patient chat histories or profiles. Session state is used for the current session only.

11. Future Enhancements

- **Database Integration:** Implement a robust database (e.g., MongoDB, PostgreSQL) to store real patient data, health metrics, and chat history for persistent user profiles.
- **User Authentication and Authorization:** Add a secure user login system to manage individual patient accounts and ensure data privacy.
- **Advanced Data Analytics:** Integrate more sophisticated data analysis techniques to provide deeper, more personalized insights into health trends, including anomaly detection and predictive analytics beyond basic trends.
- **Integration with Wearable Devices:** Allow users to connect health tracking wearables (e.g., smartwatches) to automatically sync vital signs and activity data.
- **Customizable Alerts:** Enable users to set up personalized alerts for specific health metrics (e.g., high blood pressure readings, unusual heart rate).
- **Appointment Scheduling:** Integrate with a calendaring system to help users schedule appointments with healthcare professionals based on AI recommendations.
- **Multi-modal Input:** Explore incorporating voice input for the patient chat.
- **Model Fine-tuning:** Fine-tune the IBM Granite model or explore other specialized medical LLMs for even more accurate and nuanced healthcare responses.
- **Mobile Responsiveness:** Enhance the Streamlit application's responsiveness for a better experience on mobile devices.