SMARTBRIDGE
Let's Bridge the Gap

Generative AI Project using IBM Cloud – HEALTHAI

Project Documentation Format

---

1. Introduction

- **Project Title: HEALTHAI: Intelligent Healthcare Assistant using IBM Granite (Generative AI with IBM Cloud) Team Members:**
    - **Divya Siva nagaMalleswari Dontuboina (Team Leader – Development & Integration):**
      Led the complete development of the HEALTHAI application, including IBMGranite integration, Streamlit-based UI design, module creation, and model API handling,project deployment.
    - **Akula Sujan (Model Interaction & Testing):**
      Contributed by assisting in prompt design, testing the AI model outputs across modules like Disease Prediction and Health Chat, and refining interactions with IBM Granite.
    - **Sri Vidya Lakshmi Althy (UI Structuring & Feature Enhancement):**
      Supported in designing user flow, organizing the Streamlit interface across all modules, and suggesting improvements in user interaction and feature behavior.

---

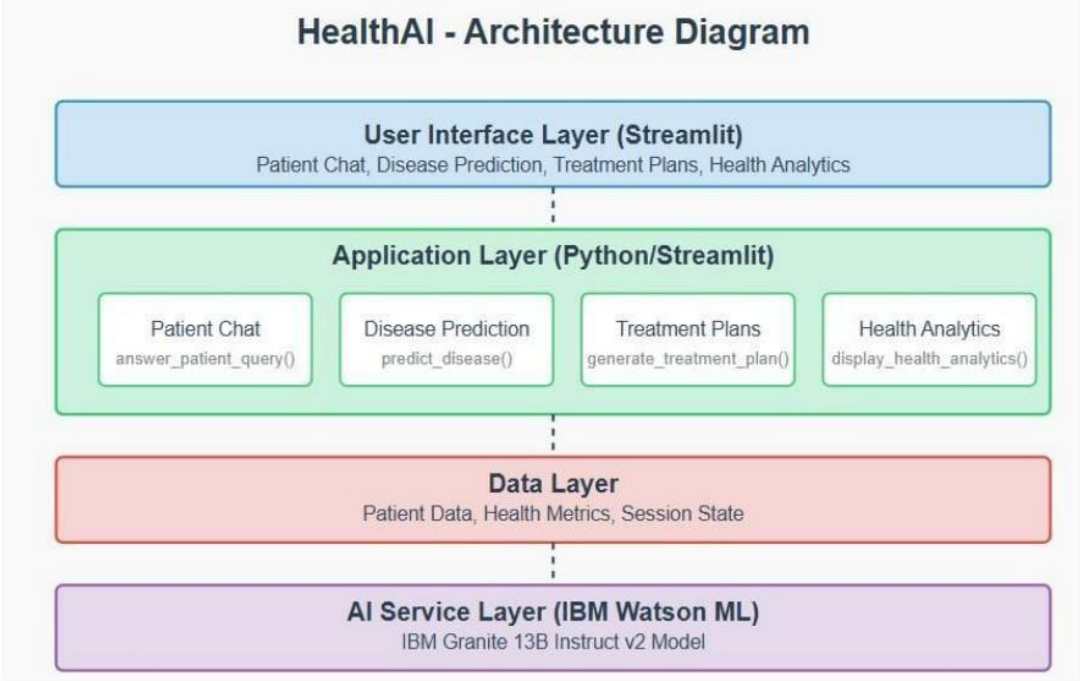
2. Project Overview

- **Purpose:**
  To build a Generative AI-based healthcare assistant using IBM Granite, capable of answering health queries, predicting diseases, suggesting treatments, and displaying analytics.
- **Features:**
    - AI Health Chat using IBM Granite
    - Disease Prediction from user symptoms
    - Treatment Plan Suggestions
    - Health Analytics Dashboard
    - Centralized shared model for performance optimization

---

3. Architecture

- **Frontend:**
  Built using **Streamlit** for a clean and responsive web interface. Each feature is modularized for easy navigation via sidebar.
- **Backend & Model:**
  - No traditional backend. All logic handled in Streamlit using Python.
  - Uses **IBM Granite 3-2b Instruct model** from IBM Watsonx: ibm-granite/granite-3.2b-instruct
  - Supports both API and **local model loading** (granite/ folder).
- **Shared Model Loader:**
  The shared_model.py file centrally loads and shares the AI model across modules to prevent memory crashes and redundancy.



---

4. Setup Instructions

Prerequisites

- Python 3.10+
- pip
- IBM cloud account and Streamlit Community Cloud account

- Installed model files if using local (granite/ folder)

Installation

git clone: https://github.com/divyareddy011/Health

ai.git

cd Health-ai pip install -r requirements.txt

Environment Variables

Create a .env file in the root folder:

IBM watsonx_API_Key= ESdPIiW78JY8LM32Vp6ujw_EVP1dySvqb5ZQ5lc7K4wi

✅ .env file must be excluded in .gitignore.

---

5. Folder Structure

Health-ai/

├— app.py # Main entry point

├— shared_model.py # Shared AI model instance

├— patient_chat.py # AI Health Chat module

├— disease_prediction.py # Disease Prediction logic

├— treatment_plans.py # Treatment Plan suggestions

├— health_analytics.py # Analytics module

├— requirements.txt # Python dependencies

├— .env # API token (not pushed to GitHub)

├— granite/ # [Optional] Local model folder

└— assets/ # Logos and screenshots

---

6. Running the Application

For IBM watsonx API:

streamlit run app.py

For Local Model:

Ensure granite/ folder contains the downloaded model and tokenizer files.

In shared_model.py, update: model_path = "./granite"

---

7. API Documentation

Endpoint: https://eu-de.ml.cloud.ibm.com/ml/v1/text/chat?version=2023-05-29

Headers:

```
{
 "Authorization": "Bearer <IBM _API_Key>",
 "Content-Type": "application/json"
}
```

Example Request:

```
{
 "inputs": "What are the symptoms of diabetes?"
}
```

Example Response:

```
{
 "generated_text": "Common symptoms of diabetes include frequent urination..."
}
```

---

8. Authentication

- Streamlit cloud Secrets is securely stored the .env credentials.
- .env is excluded via .gitignore
- App is currently public and stateless (no user login)
- HuggingFace or Firebase Auth can be added in future

---

9. User Interface

- Built entirely with **Streamlit**
- Sidebar for navigation
- Text/chat inputs for interaction

- Visual graphs and health tips in Analytics
- Centralized theme and branding

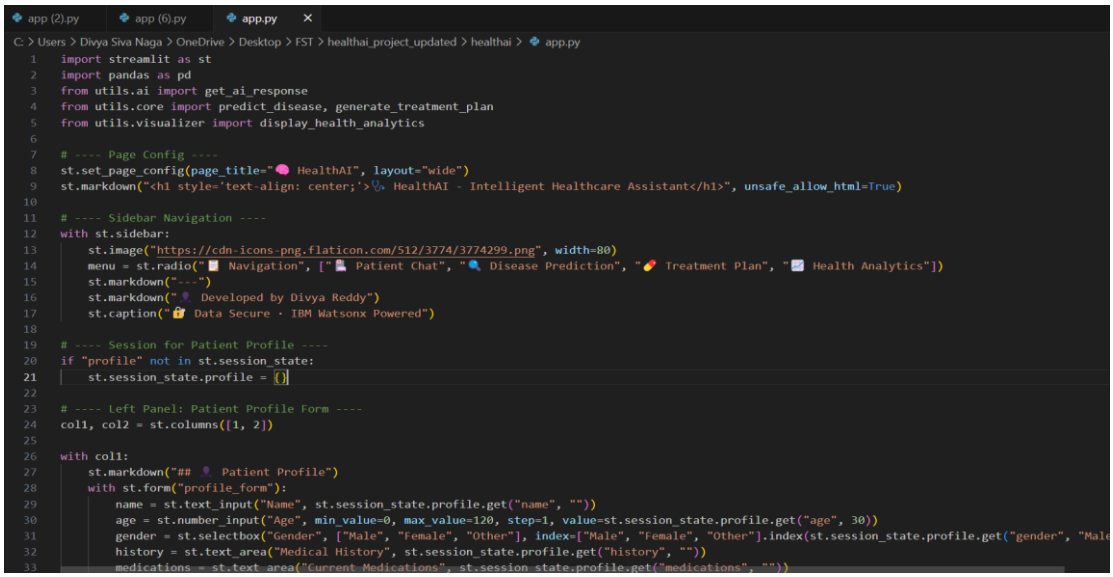
---

10. Testing

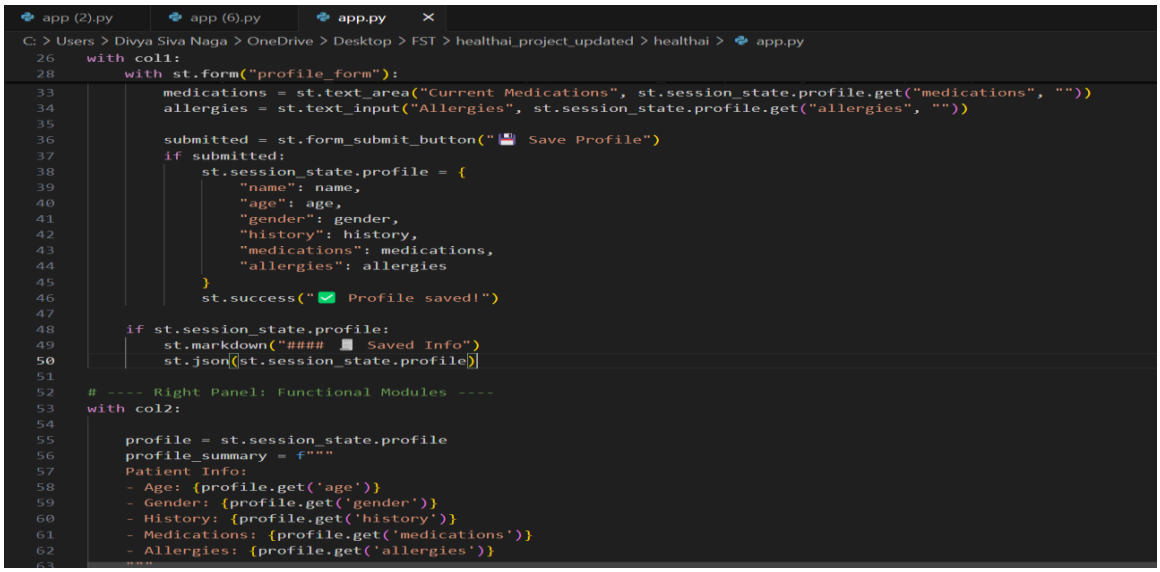
- ✅ Manual testing across all modules
- ✅ Model tested with varied prompts and edge cases
- ✅ Handled errors for invalid inputs and model timeouts

---

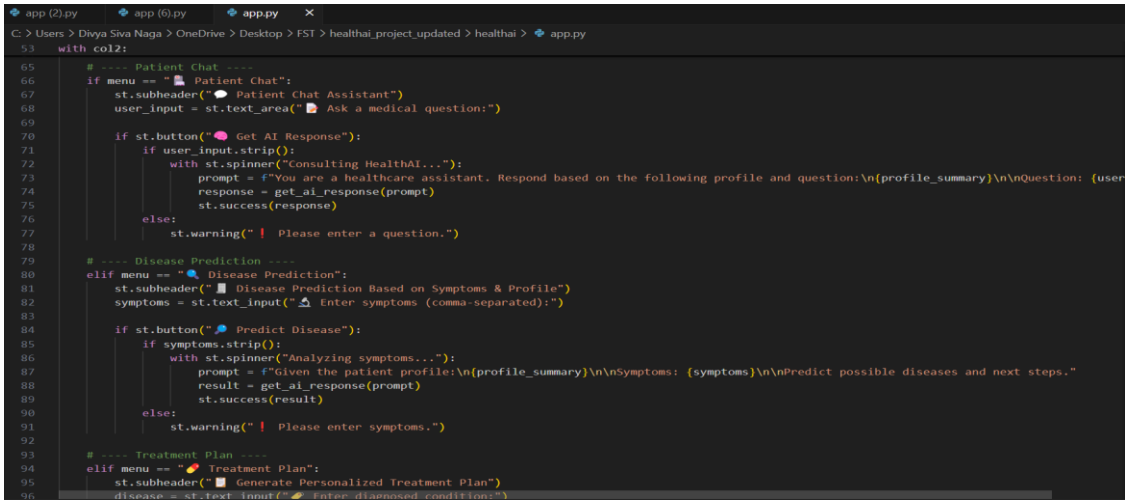
11. Screenshots or Demo



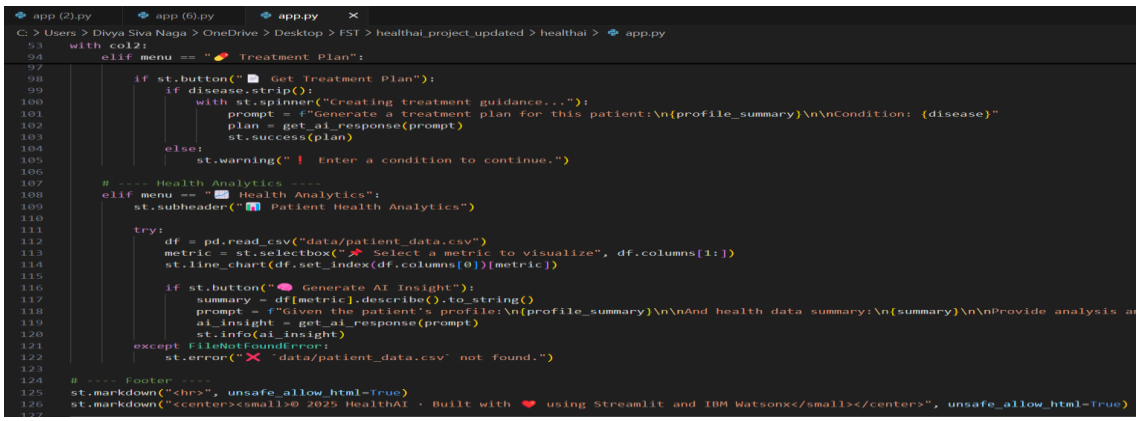
SMARTBRIDGE
Let's Bridge the Gap



```
app (2).py   app (6).py   app.py
C: > Users > Divya Siva Naga > OneDrive > Desktop > FST > healthai_project_updated > healthai > app.py
    with col1:
        with st.form("profile_form"):
            medications = st.text_area("Current Medications", st.session_state.profile.get("medications", ""))
            allergies = st.text_input("Allergies", st.session_state.profile.get("allergies", ""))

            submitted = st.form_submit_button("💾 Save Profile")
            if submitted:
                st.session_state.profile = {
                    "name": name,
                    "age": age,
                    "gender": gender,
                    "history": history,
                    "medications": medications,
                    "allergies": allergies
                }
                st.success("✅ Profile saved!")

        if st.session_state.profile:
            st.markdown("#### 📋 Saved Info")
            st.json(st.session_state.profile)

    # ---- Right Panel: Functional Modules ----
    with col2:

        profile = st.session_state.profile
        profile_summary = f"""
        Patient Info:
        - Age: {profile.get('age')}
        - Gender: {profile.get('gender')}
        - History: {profile.get('history')}
        - Medications: {profile.get('medications')}
        - Allergies: {profile.get('allergies')}
        """
```



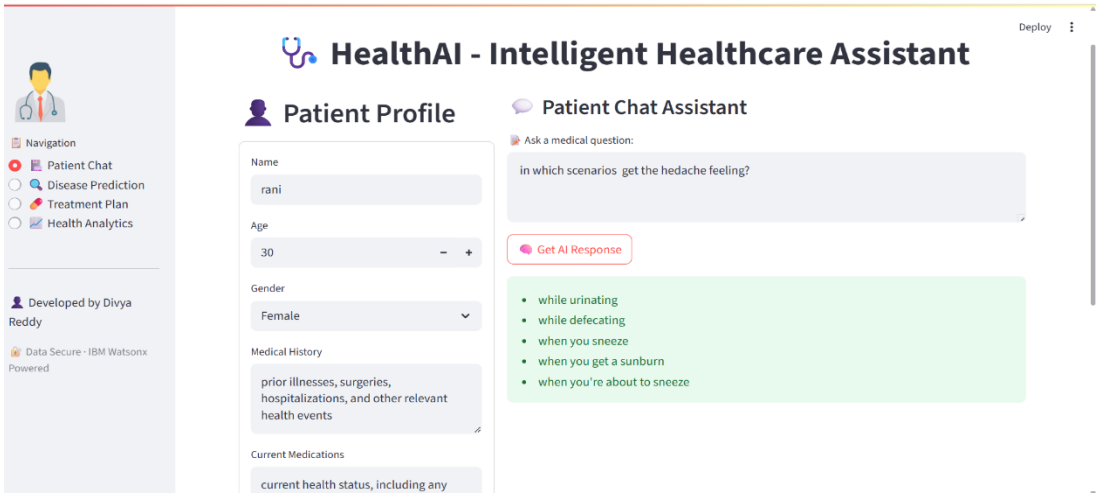
```
app (2).py   app (6).py   app.py
C: > Users > Divya Siva Naga > OneDrive > Desktop > FST > healthai_project_updated > healthai > app.py
    with col2:

        # ---- Patient Chat ----
        if menu == "🧑‍⚕️ Patient Chat":
            st.subheader("💬 Patient Chat Assistant")
            user_input = st.text_area("📝 Ask a medical question:")

            if st.button("🧠 Get AI Response"):
                if user_input.strip():
                    with st.spinner("Consulting HealthAI..."):
                        prompt = f"You are a healthcare assistant. Respond based on the following profile and question:\n{profile_summary}\n\nQuestion: {user
                        response = get_ai_response(prompt)
                        st.success(response)
                else:
                    st.warning("❗ Please enter a question.")

        # ---- Disease Prediction ----
        elif menu == "🔍 Disease Prediction":
            st.subheader("📊 Disease Prediction Based on Symptoms & Profile")
            symptoms = st.text_input("🔬 Enter symptoms (comma-separated):")

            if st.button("🔎 Predict Disease"):
                if symptoms.strip():
                    with st.spinner("Analyzing symptoms..."):
                        prompt = f"Given the patient profile:\n{profile_summary}\n\nSymptoms: {symptoms}\n\nPredict possible diseases and next steps."
                        result = get_ai_response(prompt)
                        st.success(result)
                else:
                    st.warning("❗ Please enter symptoms.")

        # ---- Treatment Plan ----
        elif menu == "💊 Treatment Plan":
            st.subheader("📋 Generate Personalized Treatment Plan")
            disease = st.text_input("🧾 Enter diagnosed condition:")
```

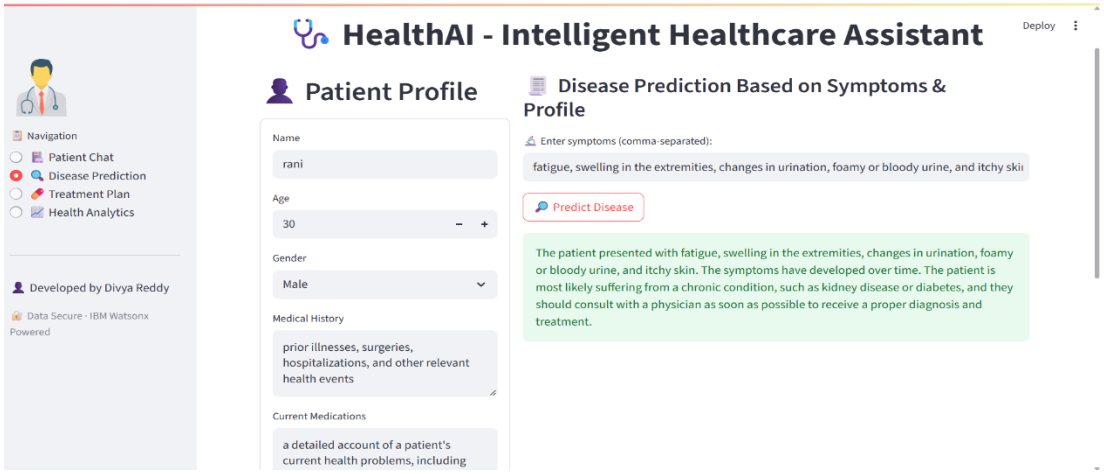


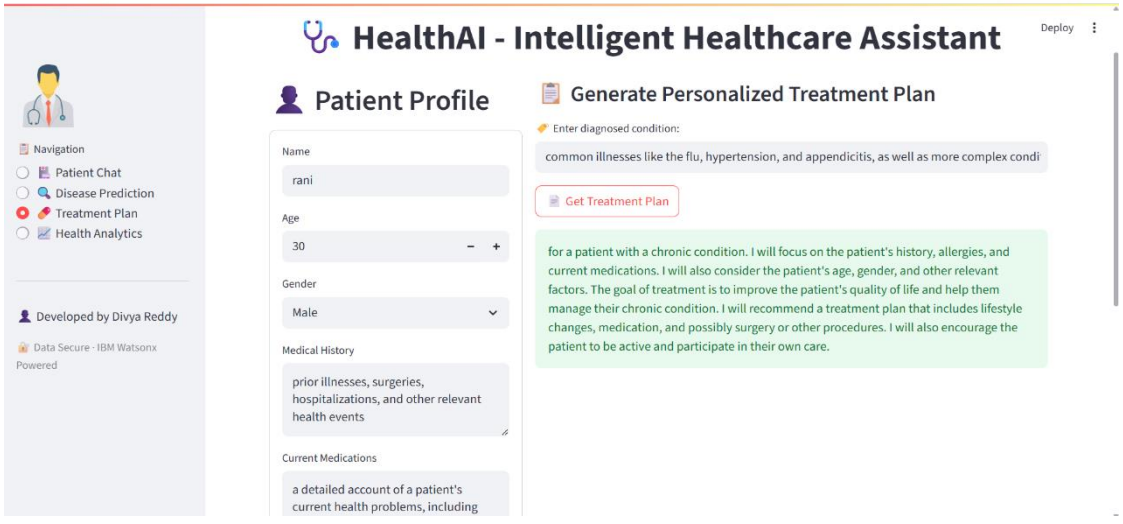
```
app (2).py   app (6).py   app.py
C: > Users > Divya Siva Naga > OneDrive > Desktop > FST > healthai_project_updated > healthai > app.py
    with col2:
        elif menu == "💊 Treatment Plan":

            if st.button("📄 Get Treatment Plan"):
                if disease.strip():
                    with st.spinner("Creating treatment guidance..."):
                        prompt = f"Generate a treatment plan for this patient:\n{profile_summary}\n\nCondition: {disease}"
                        plan = get_ai_response(prompt)
                        st.success(plan)
                else:
                    st.warning("❗ Enter a condition to continue.")

        # ---- Health Analytics ----
        elif menu == "📈 Health Analytics":
            st.subheader("📊 Patient Health Analytics")

            try:
                df = pd.read_csv("data/patient_data.csv")
                metric = st.selectbox("📌 Select a metric to visualize", df.columns[1:])
                st.line_chart(df.set_index(df.columns[0])[metric])

                if st.button("🧠 Generate AI Insight"):
                    summary = df[metric].describe().to_string()
                    prompt = f"Given the patient's profile:\n{profile_summary}\n\nAnd health data summary:\n{summary}\n\nProvide analysis a
                    ai_insight = get_ai_response(prompt)
                    st.info(ai_insight)
            except FileNotFoundError:
                st.error("❌ 'data/patient_data.csv' not found.")

    # ---- Footer ----
    st.markdown("<hr>", unsafe_allow_html=True)
    st.markdown("<center><small>© 2025 HealthAI · Built with ❤️ using Streamlit and IBM Watsonx</small></center>", unsafe_allow_html=True)
```

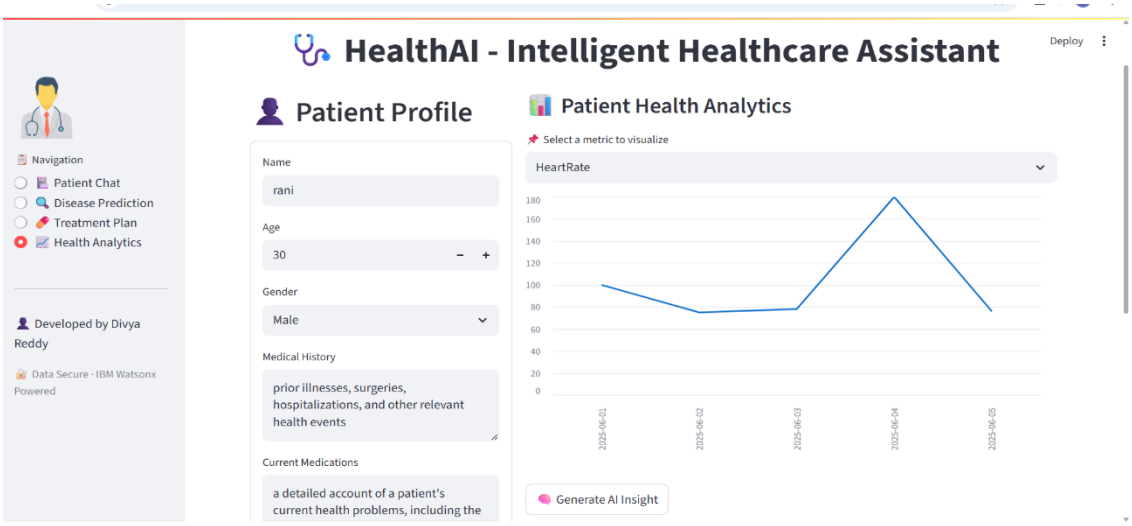
- **OUTPUT :**

SMARTBRIDGE
Let's Bridge the Gap









12. Known Issues

- □ Generic model outputs due to lack of medical domain fine-tuning
- □ Internet dependency when using IBM watsonx.
- □ No data persistence (currently stateless app)

---

13. Future Enhancements

- ☑ Add user authentication and patient record storage
- ☑ Multilingual prompt support
- ☑ Mobile version of the app
- ☑ Integrate with real-time health APIs or EHRs
