SRI VENKATESWARA COLLEGE OF ENGINEERING

Tirupati

INTERSHIP DOCUMENTATION

ON

“HealthAI: Intelligent Healthcare Assistant Using IBM Granite”

Submitted in partial fulfillment of the requirements of the

Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

KADIYALA UDAY KIRAN (22BFA02055)

SHAIK THASNEEM (22BFA05386)

JALADHI MUNI NAGESH (22BFA02049)

GUTTHA NADHIYA(22BFA02047)

P RUCHITHA(22BFA05370)

TEAM ID:

LTVIP2025TMID33215

1. Project Overview

HealthAI is a next-generation, AI-powered healthcare assistant designed to provide users with intelligent, personalized, and accessible medical support. The platform leverages the capabilities of **IBM Watson Machine Learning** and **IBM Granite-13b-instruct-v2**, a powerful generative language model, to process user inputs, analyze symptoms, and generate medical recommendations.

Built with **Streamlit**, HealthAI features an intuitive web interface where users can engage in natural language conversations, predict possible diseases, receive AI-generated treatment plans, and monitor health trends through dynamic visualizations. It transforms how individuals interact with healthcare technology by providing real-time support that is not only data-driven but also conversational and personalized.

1.1 PURPOSE:

The purpose of **HealthAI** is to empower individuals with intelligent, accessible, and personalized healthcare support by utilizing advanced generative AI models. HealthAI serves as a virtual healthcare assistant that offers users preliminary diagnostics, treatment recommendations, interactive health analytics, and an AI-driven medical chat—all from a single, easy-to-use platform.

HealthAI addresses the growing need for reliable, real-time health guidance in scenarios where immediate access to medical professionals may be limited. The platform bridges the gap between health awareness and medical consultation by providing users with actionable insights and clear, empathetic communication powered by IBM Granite and Watson Machine Learning.

1.2 FEATURES:

1. Patient Chat System

- An intelligent chat interface where users can ask health-related questions.
- Real-time, context-aware responses powered by IBM Granite-13b-instruct-v2.
- Maintains session-based message history for continuous conversations.

2. Disease Prediction

- Users describe symptoms in natural language.
- The AI analyzes the input and predicts possible conditions.
- Provides likelihood indicators and recommended next steps.

- Includes medical disclaimers to encourage real consultation.

💊 3. Treatment Plan Generator

- Users input a known medical condition.
- AI generates a comprehensive treatment plan, including:
  - Medication suggestions
  - Lifestyle modifications
  - Monitoring and follow-up care
- Structured in easy-to-read, collapsible sections.

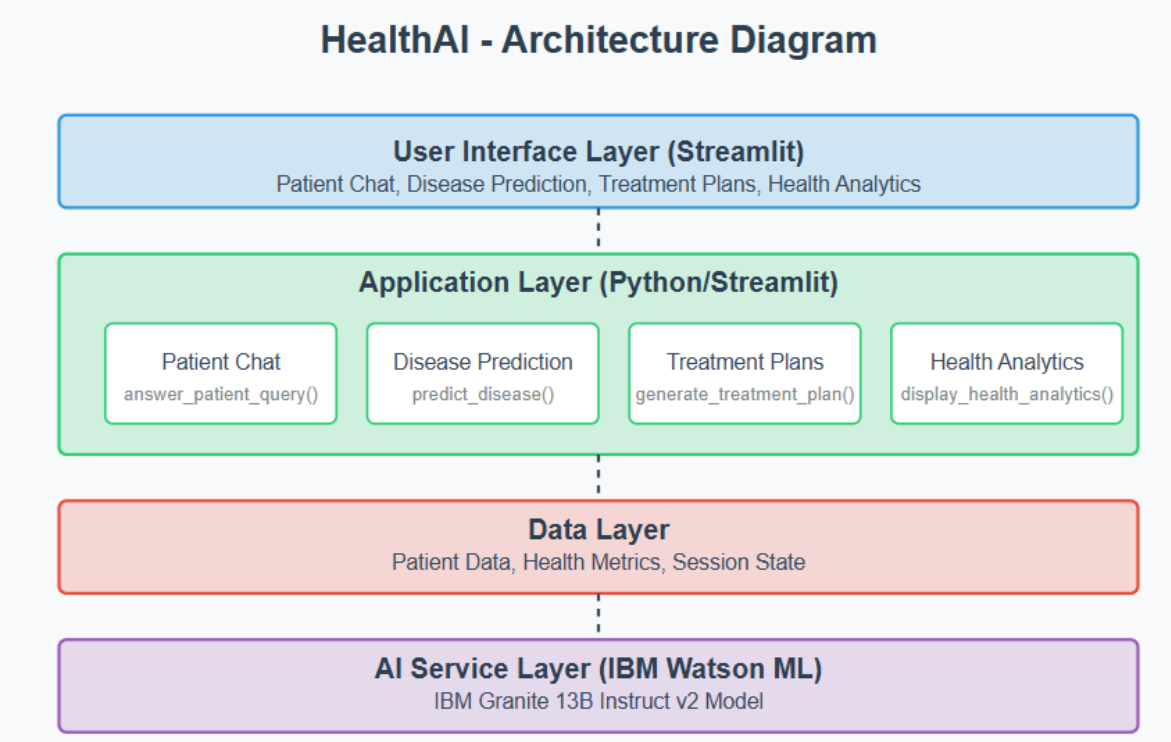
📊 4. Health Analytics Dashboard

- Visualizes patient metrics like heart rate, blood pressure, and glucose levels.
- Interactive line charts and symptom frequency pie charts.
- Trend summaries with color-coded indicators (e.g., red for abnormal, green for normal).
- Provides AI-generated insights and health improvement suggestions.

👤 5. Patient Profile Management

- Users can manage personal health details (age, gender, medical history).
- Profile data is used to tailor predictions and insights.
- Ensures persistent session-based interaction and secure data handling.

2.ARCHITECTURE



2.1 FRONTEND (User Interface)

Technology: Streamlit (Python-based UI Framework)

Responsibilities:

- Provide an interactive, clean, and accessible user interface.
- Enable user interactions via:
    - Symptom entry for disease prediction
    - Text input for patient chat
    - Condition input for treatment plans
    - Patient data uploads and visualizations
- Maintain chat history and session state.
- Render dynamic data visualizations (line charts, pie charts, health metrics).

UI Modules:

- Home Page: Intro and navigation
- Patient Chat Interface: Chat-style interface with message flow
- Disease Prediction: Symptom form + output card
- Treatment Plans: Input box + structured treatment layout
- Health Analytics: Charts + metric summaries

2.2 BACKEND (Business Logic & AI Integration)

Technology: Python

AI Model APIs:

- IBM Granite-13b-instruct-v2 (LLM for patient chat, treatment plans)
- IBM Watson Machine Learning (for disease prediction)

Responsibilities:

- Process and validate user inputs.
- Generate prompts for IBM Granite API calls.

- Integrate IBM Watson ML deployment for condition prediction.
- Handle JSON responses and format for UI display.
- Compute and analyze health metrics from CSV data.

Core Backend Modules:

- granite_llm.py: Handles LLM calls for chat and treatments.
- mock_predictor.py or watson_predictor.py: AI prediction based on symptoms.
- analytics_utils.py: Calculates trends, averages, and flag indicators.

2.3 DATABASE (Data Management Layer)

Technology:CSV-basedmockdata

Alternative (For Production): SQLite / Firebase / PostgreSQL

Responsibilities:

- Manage patient profiles and input records.
- Store uploaded patient metrics (heart rate, glucose, BP).
- Serve time-series data for analytics.
- Maintain chat history and session variables (Streamlit's session state).

Sample Data Files:

- patient_metrics.csv: Stores historical vitals.

3.SETUP INSTRUCTIONS

3.1 Pre-requisites:

1. Streamlit Framework Knowledge: Streamlit Documentation
2. IBM Watson Machine Learning: IBM Watson ML Documentation
3. Python Programming Proficiency: Python Documentation
4. Data Visualization Libraries: Plotly Documentation
5. VersionControlwithGit: GitDocumentation

3.2 Installations:

Development Environment Setup: Flask Installation Guide

4.FOLDER STRUCTURE

4.1 Client (Frontend - Streamlit UI)

Technology: Streamlit

Responsibilities:

- Render UI components (forms, inputs, charts)
- Handle user interactions:
  - Chat queries
  - Symptom descriptions
  - Condition input
  - Profile details
- **Display:**
  - Model predictions
  - Health recommendations
  - Analytics visualizations
- Maintain stateful sessions (chat, inputs, profile)
- Responsive design for desktop/mobile

Pages:

- 🏠 Home
- 💬 Patient Chat
- 🩺 Disease Prediction
- 💊 Treatment Plan Generator
- 📊 Health Analytics Dashboard

4.2 Server (Backend Logic & AI Services)

Technology: Python functions, IBM Granite API, IBM Watson ML, optionally OpenAI

Responsibilities:

- Process inputs from frontend
- Build prompt structures
- Call IBM Granite LLM API for:
    - Treatment suggestions
    - Patient Q&A
- Call IBM Watson ML for:
    - Disease predictions
- Return structured JSON responses
- Handle error checking and rate limits
- Perform data trend calculations for visualizations

5. RUNNING THE APPLICATION

5.1 FRONTEND (Streamlit UI) :

“streamlit run app.py” in the clint directory.

5.2 BACKEND (FastAPI) :

“uvicorn api:app –reload” in the server directory.

6 . API DOCUMENTATION

HealthAI – API Documentation

🔗 Base URL: http://localhost:8000

1. POST /predict_disease

Purpose: Predict disease based on symptoms

Input:

{ "symptoms": "fever, headache" }

Response:

{ "condition": "Viral Infection", "confidence": "High" }

2. POST /generate_treatment

Purpose: Generate treatment for a condition

Input:

{ "condition": "Diabetes" }

Response:

{ "treatment_plan": "Use Metformin, follow diet, regular checkups" }

---

3. POST /patient_chat

Purpose: Ask a health question

Input:

{ "message": "What causes high BP?" }

Response:

{ "reply": "High blood pressure can be caused by diet, stress, or genetics." }

7 . AUTHENTICATION

Purpose:

To **protect** user-specific data like:

- Health questions
- Predictions
- Personal insights

And to make sure only **valid users** can access sensitive features like:

- Health analytics
- Chat history
- Profile updates
-

🔑 **1. Authentication (Who can access)**

OptionUsed: SimpleUsername+Password(Session-based)
→ Suitable for **Streamlit apps** with internal or demo usage.

🔧 **How it works:**

- A login form is created using st.text_input() and st.session_state.
- When a valid user logs in, we store:

st.session_state["authenticated"] = True

st.session_state["username"] = "udaykiran"

Example:

```
if not st.session_state.get("authenticated"):
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")
    if st.button("Login") and username == "admin" and password == "1234":
        st.session_state["authenticated"] = True
        st.success("Logged in successfully!")
```

🚪 **2. Authorization (What user can do)**

You check for session before showing protected pages.

Example:

```
if st.session_state.get("authenticated"):
    st.write("Welcome to Health Analytics!")
else:
    st.warning("Please log in to view this page.")
```

🔐 **3. API Key Management (for AI Models)**

To access IBM Granite or OpenAI models:

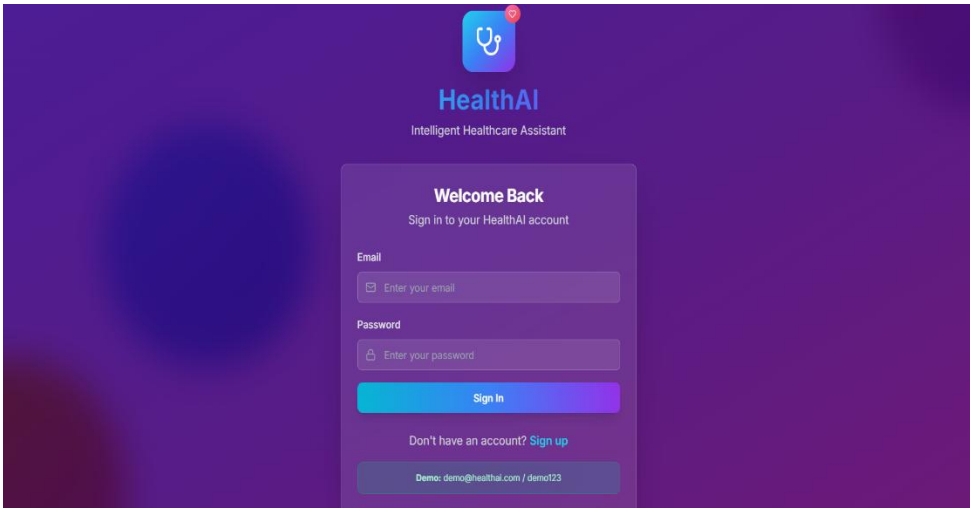
- Store keys in .env file:
- OPENAI_API_KEY=your-api-key
- Load in your Python file:

```
import os
from dotenv import load_dotenv
load_dotenv()
```

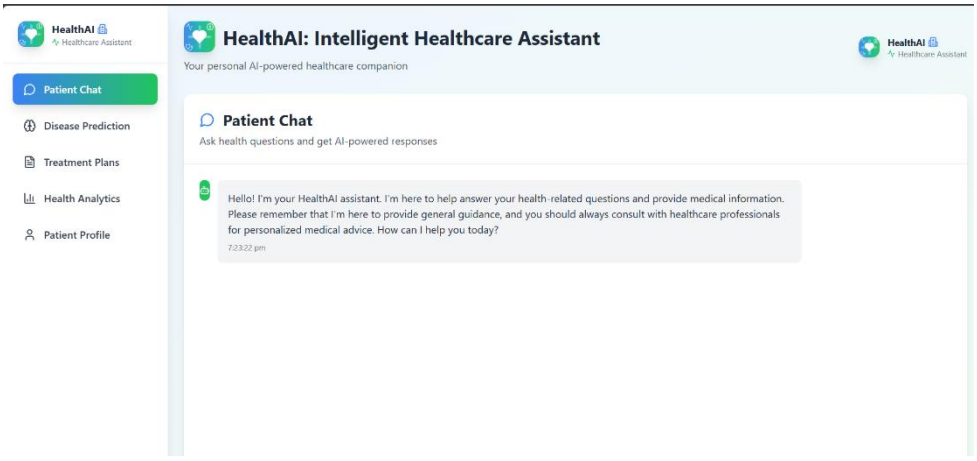
api_key = os.getenv("OPENAI_API_KEY")

8 . USER INTERFACE (UI DESIGN

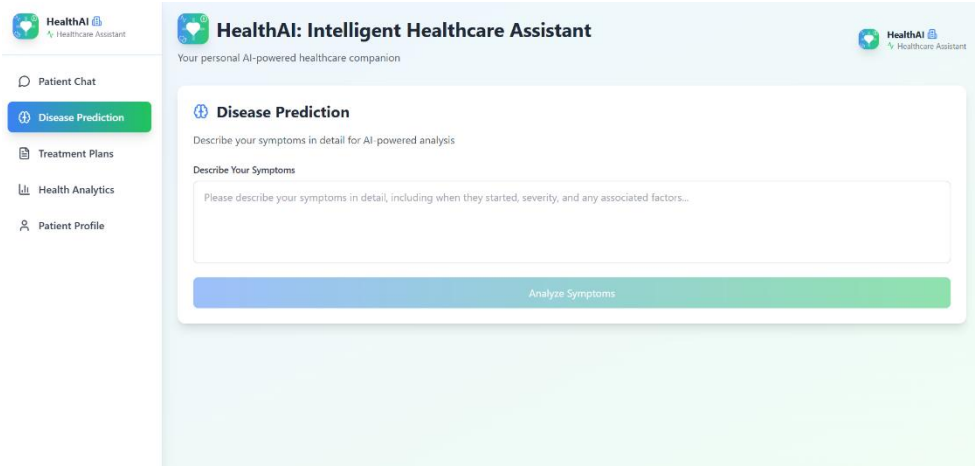
The webpage designs:-



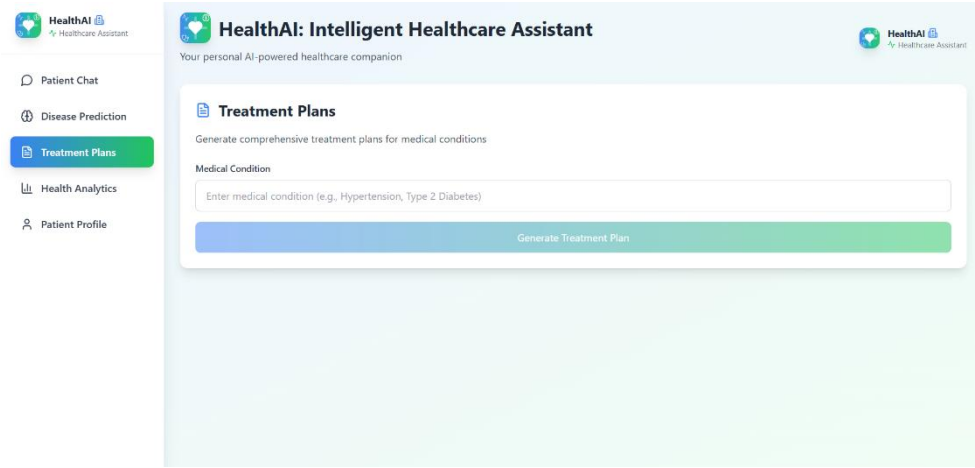
User Authentication UI



Patient chat AI chat UI



Disease prediction UI



Treatment plans UI

Health Analytics UI

Patient profile UI

9. TESTING

Testing Strategy

HealthAI uses a **manual + functional testing** approach, suitable for data-driven, AI-integrated web applications. The testing focuses on ensuring the **core functionalities**, **interface flow**, and **API responses** work as expected.

Key Areas Tested:

Module	What's Tested
Patient Chat	Response clarity, context handling, edge question input
Disease Prediction	Input validation, prediction accuracy, response speed
Treatment Plans	Valid conditions generate meaningful recommendations
Health Analytics	CSV upload, chart rendering, metric correctness
Session Management	Login/logout behavior, state persistence

🧪 2. Types of Testing Performed

- **Manual Testing**
  - Through the Streamlit interface (UI clicks, text inputs)
  - Symptom and treatment workflows
- **Functional Testing**
  - Checking if each feature returns the **correct expected output**
- **API Testing (Optional)**

    - Postman used to test endpoints (if using FastAPI backend)
    - JSON request/response validation
- **Data Testing**
    - Uploaded patient_metrics.csv tested for:
        - Format errors
        - Missing values
        - Graph rendering
- **Security Testing**
    - API key hidden using .env
    - Authentication state validation (st.session_state)

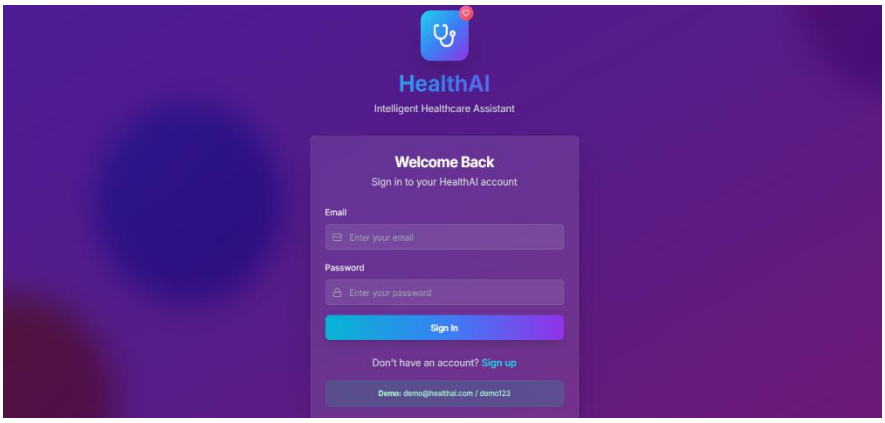
🧰 3. Tools Used

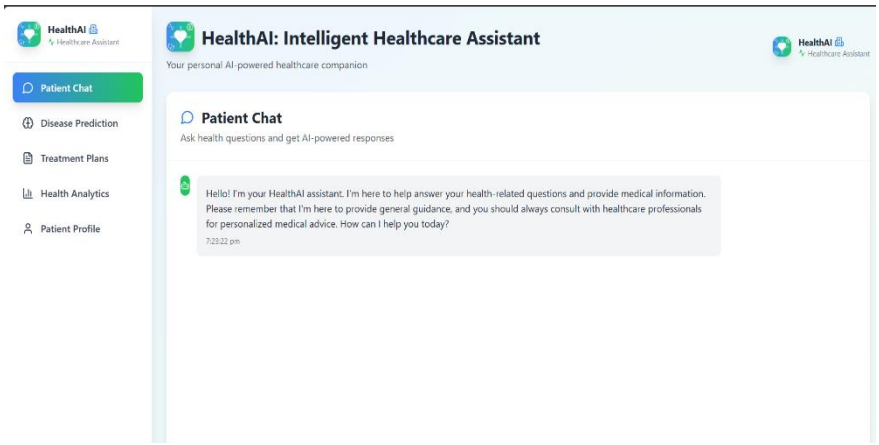
Tool	Purpose
Streamlit	UI testing and state management
Postman	API endpoint testing (if REST used)
Python assert	Unit-like checks in functions
VS Code / PyCharm	Manual input/output observation
Matplotlib	Validate analytics visualization
.env file	Manage API key security

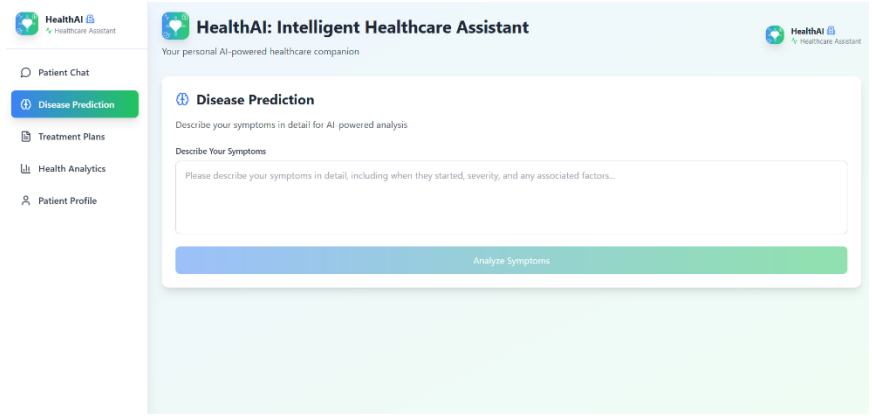
📋 Example Test Case

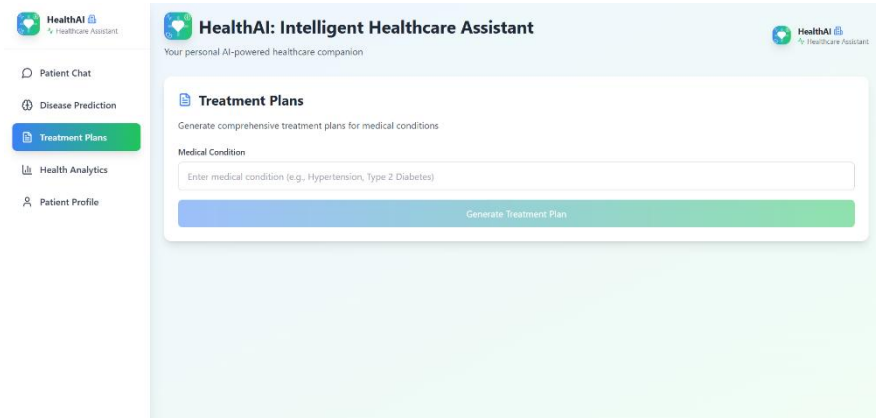
Test Name	Input	Expected Output
Disease Prediction Test	"Fever, fatigue, cough"	Returns Viral Infection, confidence %
Chat Test	"What causes high BP?"	Returns relevant explanation
Chart Render Test	Upload patient_metrics.csv	Shows heart rate and glucose trend
Login Test	Correct username & password	App navigates to dashboard

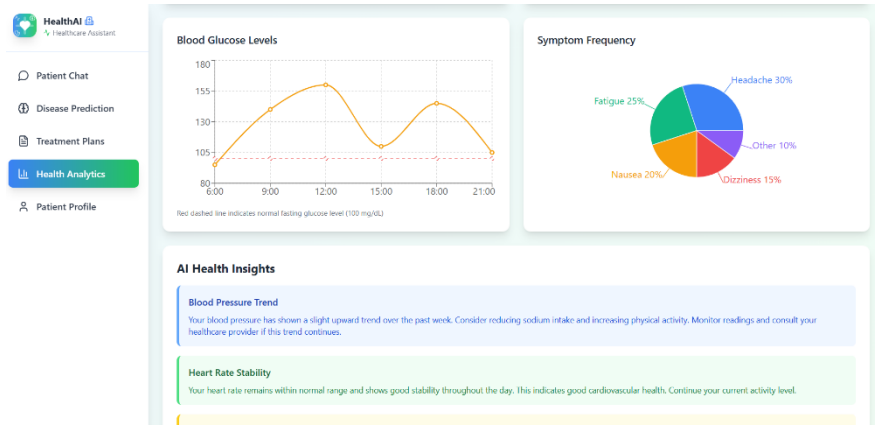
10. SCREENSHOTS or DEMO











website DEMO Link:- [healthai-insights-dashboard]

11. KNOWN ISSUES

1. ❗ **No Real Medical Validation**
   - **Issue:** The disease predictions and treatment plans are AI-generated and not medically certified.
   - **Impact:** Users might interpret suggestions as real diagnoses.
   - **Recommended Fix:** Include disclaimers on all pages. Add a mandatory acknowledgment checkbox.

2. 🔑 **API Key Hardcoding (if not using .env)**
   - **Issue:** If API keys are exposed in code (e.g., openai.api_key = "sk-..."), this poses a security risk.
   - **Impact:** Risk of API misuse or quota exhaustion.

- **Recommended Fix:** Always use .env files or environment variables to store sensitive data.

3. 🤯 LLM Hallucinations (Granite/OpenAI)

- **Issue:** The AI may sometimes generate plausible but factually incorrect medical information.
- **Impact:** May mislead users if not verified.
- **Recommended Fix:** Add visible warnings like "AI-generated, not a substitute for doctor's advice."

4. 📊 Graph Errors with Invalid CSV

- **Issue:** If the uploaded patient_metrics.csv contains missing or malformed data, charts may not render.
- **Impact:** Crashes or blank graphs.
- **Recommended Fix:** Validate CSV structure before processing. Handle missing values gracefully.

5. 👥 No User Role Management

- **Issue:** All users have equal access; no admin vs user roles.
- **Impact:** Limits control in multi-user environments.
- **Recommended Fix:** Add user roles and access restrictions in future versions.

6. 🔄 Session Loss on Refresh

- **Issue:** Streamlit's session state resets on browser refresh.
- **Impact:** User chat history or input may be lost.
- **Recommended Fix:** Store state in a local database or temporary file if needed.

7. 🌐 No Internet = App Breaks

- **Issue:** IBM Granite, Watson ML, and LLMs require live API access.
- **Impact:** App becomes non-functional offline.
- **Recommended Fix:** Use cached fallback messages or warnings when disconnected.

12. FUTURE ENHANCEMENTS

1. 👤 **User Authentication System**
   - Add secure login and registration features.
   - Support role-based access (Admin, Doctor, Patient).
   - Enable password encryption and session management.

2. 🗄️ **Database Integration**
   - Replace static CSV files with a real-time database (e.g., Firebase, PostgreSQL).
   - Store user profiles, chat history, health records persistently.

3. 📱 **Mobile App Version**
   - Develop a cross-platform mobile app using Flutter or React Native.
   - Sync with the same backend API and AI models.

4. 🧠 **AI Model Fine-Tuning**
   - Train a custom disease prediction model using local healthcare datasets.
   - Fine-tune IBM Granite on medical-specific prompts for better accuracy.

5. 🌐 **Multi-language Support**
   - Enable multilingual interactions in Patient Chat.
   - Support voice input/output for accessibility.

6. 📥 **PDF/Email Reports**
   - Allow users to download or email their treatment plans and analytics summaries.

7. 🔔 **Health Alerts & Reminders**
   - Notify users when vitals are abnormal or symptoms need attention.
   - Schedule AI-based follow-ups and medication reminders.

8. 📊 Doctor Dashboard

- Provide doctors with a summary of patient data, trends, and AI suggestions.
- Add features for note-taking, alerts, and exportable reports.

9. 🛡️ HIPAA/GDPR Compliance

- Implement consent management and encrypted storage.
- Ensure user data privacy for healthcare-grade deployment.