

# Health AI: Intelligent Healthcare Assistant

*Generative AI with IBM*



Team ID: NM2025TMID11853

Team Size: 5

Team Leader: ULAGANATHAN E

Team member: MUTHAMIZHRAJA M

Team member: SURESH GOBI K

Team member: SUTHAKAR S

Team member: PANTHALADEVAN A

# HealthAI: Intelligent Healthcare Assistant Using IBM Granite

## Project Description:

HealthAI harnesses IBM Watson Machine Learning and Generative AI to provide intelligent healthcare assistance, offering users accurate medical insights. The platform includes a Patient Chat for answering health-related questions, Disease Prediction that evaluates user-reported symptoms to deliver potential condition details, Treatment Plans that provide personalized medical recommendations, and Health Analytics to visualize and monitor patient health metrics.

Utilizing IBM's Granite-13b-instruct-v2 model, HealthAI processes user inputs to deliver personalized and data-driven medical guidance, improving accessibility to healthcare information. Built with Streamlit and powered by IBM Watson, the platform ensures a seamless and user-friendly experience. With secure API key management and responsible data handling, HealthAI empowers users to make informed health decisions with confidence.

## Scenarios:

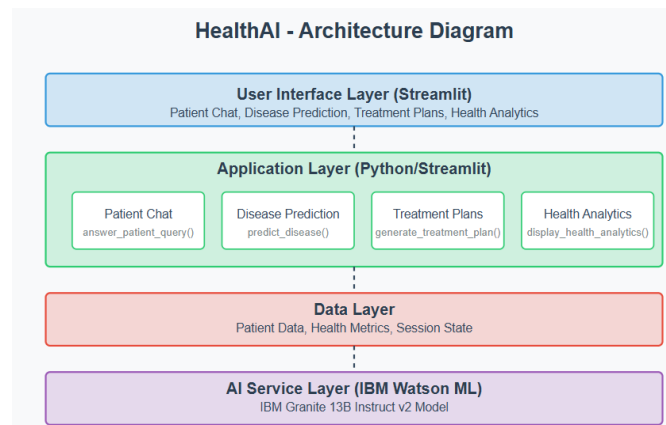
**Scenario 1:** A user inputs their symptoms into the Disease Prediction system, describing issues like persistent headache, fatigue, and mild fever. The system analyzes the symptoms along with the patient's profile and health data to provide potential condition predictions, including likelihood assessments and recommended next steps.

**Scenario 2:** A user needs personalized treatment recommendations for a diagnosed condition. By entering their condition in the Treatment Plans generator, the AI processes the information along with patient data to create a comprehensive, evidence-based treatment plan that includes medications, lifestyle modifications, and follow-up testing.

**Scenario 3:** A user wants insights about their health trends. Using the Health Analytics dashboard, they can visualize their vital signs over time (heart rate, blood pressure, blood glucose, etc.) and receive AI-generated insights about potential health concerns and improvement recommendations.

A

rchitecture



## 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. Version Control with Git: [Git Documentation](#)
6. Development Environment Setup: [Flask Installation Guide](#)

## Activity 1: Model Selection and Architecture

- Activity 1.1: Research and select the appropriate AI model from IBM Watson for medical assistance (IBM Granite 13B Instruct v2).
- Activity 1.2: Setup and Access your IBM WatsonX API key.
- Activity 1.3: Define the architecture of the application, detailing

interactions between the frontend, backend, and AI integration.

- Activity 1.4: Set up the development environment, installing necessary libraries and dependencies for Streamlit and IBM Watson ML.

## Activity 2: Core Functionalities Development

- Activity 2.1: Develop the core functionalities: Patient Chat, Disease Prediction, Treatment Plan Generation, and Health Analytics.
- Activity 2.2: Implement patient data utilities to manage and visualize health metrics.

## Activity 3: App.py Development

- Activity 3.1: Write the main application logic in app.py, establishing functions for each feature and integrating AI responses.
- Activity 3.2: Create prompting strategies for the IBM Granite model to generate high-quality medical content.

## Activity 4: Frontend Development

- Activity 4.1: Design and develop the user interface using Streamlit components, ensuring a responsive and intuitive layout.
- Activity 4.2: Create dynamic visualizations with Plotly to display health metrics and trends.

## Activity 5: Deployment

- Activity 5.1: Prepare the application for deployment by configuring environment variables for API credentials.
- Activity 5.2: Deploy the application on a suitable hosting platform to make it accessible to users.

Activity 1.2: Setup and Access your IBM WatsonX API key.

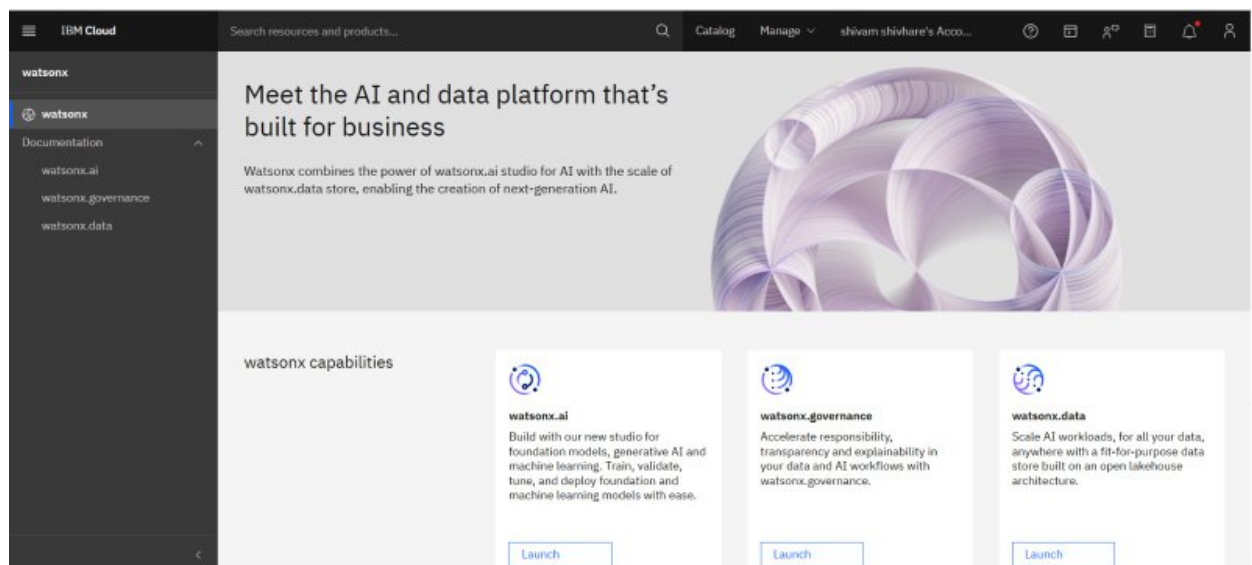
# Milestone 1: Model Selection and Architecture

In this milestone, we focus on selecting the appropriate AI model from IBM

Watson for our medical assistance needs. This involves researching the capabilities and performance of various models, ensuring that the chosen model aligns well with our application's objectives of creating a Patient Chat system, Disease Prediction, Treatment Plan Generation, and Health Analytics.

With IBM's Text2SQL functionality, that's exactly what happens. From business users and analysts to data stewards and engineers, anyone can go from question to execution in seconds. No IT bottlenecks. Less friction. More confidence. And because every rule is standardized, traceable and auditable, governance improves too.

You don't need to speak SQL to work with data anymore. Just explain what you're looking for, and the system takes care of the rest – accurately, securely and at scale. By combining natural language understanding with autonomous agent capabilities, the Text2SQL engine dynamically interprets user intent, explores relevant schemas, validates query logic, and even asks clarifying questions when needed—delivering accurate, governed results with minimal friction. It works across multiple SQL dialects—so whether your data lives in one engine or several, the logic stays consistent.



### Activity 1.3: Define the architecture of the application

1. Draft an Architectural Diagram: Create a visual representation of

the application architecture.

2. **Detail Frontend Functionality:** Outline how users will interact with the application through a Streamlit interface.
3. **Outline Backend Responsibilities:** Specify how the backend will process user input and communicate with the IBM Granite model.
4. **Describe AI Integration Points:** Define how the application will make API calls to IBM Watson ML.

#### Activity 1.3: Define the architecture of the application

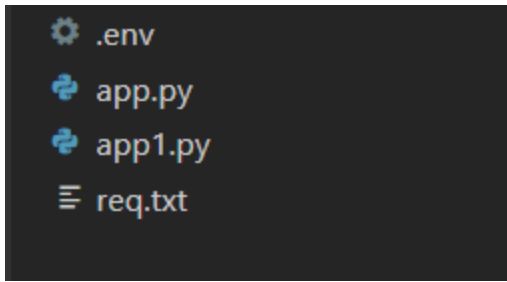
1. **Draft an Architectural Diagram:** Create a visual representation of the application architecture.
2. **Detail Frontend Functionality:** Outline how users will interact with the application through a Streamlit interface.
3. **Outline Backend Responsibilities:** Specify how the backend will process user input and communicate with the IBM Granite model.
4. **Describe AI Integration Points:** Define how the application will make API calls to IBM Watson ML.

#### Activity 1.4: Set up the development environment

1. **Install Python and Pip:** Ensure Python is installed along with pip for managing dependencies.
2. **Create a Virtual Environment:** Set up a virtual environment to isolate project dependencies.
3. **Install Required Libraries:**

```
pip install streamlit pandas numpy plotly ibm-watson-machine-learning python-dotenv
```

4. **Set Up Application Structure:** Create the initial directory structure for the HealthAI application.



## Milestone 2: Core Functionalities Development

2. Disease Prediction System:
  - o Create symptom input interface
  - o Develop prediction function using patient data and reported symptoms
  - o Structure output format to show potential conditions with likelihood and next steps
3. Treatment Plan Generator:
  - o Build input interface for condition and patient details
  - o Create prompting system for personalized treatment plans
  - o Structure output to include medications, lifestyle changes, and follow-up care
4. Health Analytics Dashboard:
  - o Implement patient data visualization with interactive charts
  - o Create metrics summary with trend indicators
  - o Develop AI-generated insights based on health trends

Disease prediction using machine learning is used in healthcare to provide accurate and early diagnosis based on patient symptoms. We can build predictive models that identify diseases efficiently. In this article, we will explore the end-to-end implementation of such a system.

```
import numpy as np
import pandas as pd
from scipy.stats import mode
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
```

```

from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from imblearn.over_sampling import RandomOverSampler

```

Reading the dataset

In this step we load the dataset and encode disease labels into numbers and visualize class distribution to check for imbalance. We then use [RandomOverSampler](#) to balance the dataset by duplicating minority classes and ensuring all diseases have equal samples for fair and effective model training.

```
data = pd.read_csv('/content/improved_disease_dataset.csv')
```

```

encoder = LabelEncoder()
data["disease"] = encoder.fit_transform(data["disease"])

```

```

X = data.iloc[:, :-1]
y = data.iloc[:, -1]

```

```

plt.figure(figsize=(18, 8))
sns.countplot(x=y)
plt.title("Disease Class Distribution Before Resampling")
plt.xticks(rotation=90)
plt.show()

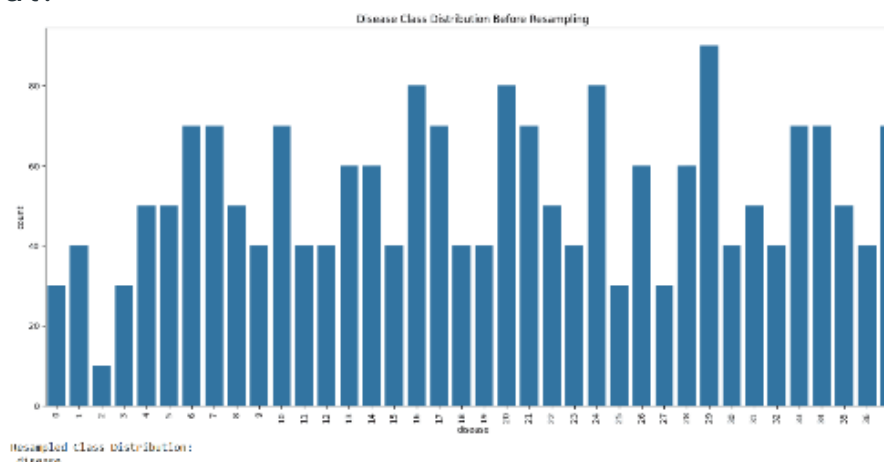
```

```

ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X, y)

```

Output:





## Activity 2.2: Implement data management utilities

1. Patient Data Generation:
  - o Create sample data with realistic health metrics
  - o Implement date-based trend generation for visualization
  - o Structure data for efficient analysis and display
2. Patient Profile Management:
  - o Develop interface for managing patient details
  - o Create session state handling for persistent data
  - o Implement profile update functionality
    - and return JSON responses.
2. Integrate the Gemini AI Responses in Each Function
  - o Implement function calls within each route to leverage Gemini's models based on user-provided details.
  - o Ensure each response from Gemini API is processed in a way that enhances readability and displays clearly to the user.
  - o Implement profile update functionality
1. Imports and Setup:
  - o Import necessary libraries (Streamlit, pandas, plotly, IBM Watson ML)
  - o Load environment variables for API keys
  - o Initialize IBM Granite model connection
2. Core Functions:
  - o `init_granite_model()`: Set up connection to IBM Watson ML
  - o `predict_disease()`: Analyze symptoms for potential diagnoses
  - o `generate_treatment_plan()`: Create personalized treatment recommendations
  - o `answer_patient_query()`: Process health questions with AI responses
3. UI Components:
  - o Main application layout with sidebar navigation
  - o Tab-based interface for different features
  - o Custom CSS styling for enhanced user experience
4. Feature Implementation:
  - o `display_patient_chat()`: Chatbot interface for health questions
  - o `display_disease_prediction()`: Symptom analysis system
  - o `display_treatment_plans()`: Treatment plan generator

## display\_health\_analytics(): Interactive health dashboard

```
def answer_patient_query(query):  
    """Use IBM Granite to answer patient health questions"""  
    model = init_granite_model()  
  
    # Create prompt for answering patient query  
    query_prompt = f"""  
    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:  
    """  
  
    answer = model.generate_text(prompt=query_prompt)  
    return answer
```

```
prediction_prompt = f"""  
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_bp_systolic}/{avg_bp_diastolic} mmHg  
- Average Blood Glucose: {avg_glucose} mg/dL  
- Recently Reported Symptoms: {recent_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.  
"""  
  
prediction = model.generate_text(prompt=prediction_prompt)  
return prediction
```

```

treatment_prompt = f"""
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.
"""

treatment_plan = model.generate_text(prompt=treatment_prompt)
return treatment_plan

```

```

treatment_prompt = f"""
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.
"""

treatment_plan = model.generate_text(prompt=treatment_prompt)
return treatment_plan

```

```

treatment_prompt = f"""
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.
"""

treatment_plan = model.generate_text(prompt=treatment_prompt)
return treatment_plan

```

# Milestone 4: Frontend Development

Design and develop the user interface

1. Main Application Layout:
  - o Configure page title, icon, and layout preferences
  - o Implement a sidebar for patient profiles and feature selection
  - o Create custom CSS for enhanced visual appearance
2. Feature- Specific Interfaces:
  - o Patient Chat: Chat- style interface with message history
  - o Disease Prediction: Symptom input form and prediction display
  - o Treatment Plans: Condition input and treatment plan output
  - o Health Analytics: Interactive charts and metrics summary

# Milestone 5: Deployment

Activity 5.1: Prepare for deployment

1. Environment Variable Configuration:

Create a `.env` file for IBM Watson API credentials:

`WATSONX_API_KEY=your_api_key_here`

- `WATSONX_PROJECT_ID=your_project_id_here`

- Implement secure loading of credentials

2. Dependency Management:
  - o Create `requirements.txt` file with all necessary packages
  - o Document installation process for deployment

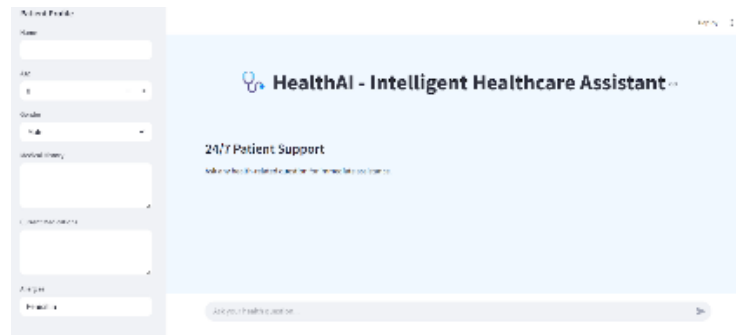
Activity 5.2: Deploy the application

1. Local Deployment Testing:
  - o Run the application using `streamlit run app.py`
  - o Test all features for functionality
  - o Verify responsive design and performance
2. Cloud Deployment Options:

- o Deploy on Streamlit Cloud for public access
- o Configure environment variables in the deployment platform
- o Set up monitoring and error logging

## Milestone 6: functional testing and verify

### Patient Chat Page



Description: Here user have access to responsive healthcare communication platform enabling seamless dialogue about wellness concerns, with chronological message tracking for context retention. The system delivers intelligent, algorithmically-generated wellness guidance while providing verifiable medical insights supported by authoritative healthcare sources, creating a comprehensive virtual consultation experience.

Patient Chat output:

Deploy

22

-

+

Male

▼

None

✕

None

✕

None

✕

HealthAI - Intelligent Healthcare Assistant

24/7 Patient Support

Ask any health-related question for immediate assistance.

I have been suffering from a fever for 2 days, my symptoms are running nose, cough, headache, and joint pain. So give me some medications

Running nose, cough, headache, and joint pain are symptoms of the common cold. For most people, the common cold will resolve on its own within 7 to 10 days. However, if your symptoms persist or get worse, you should consult your doctor. If you are taking any other medications, please check with your doctor to see if they are safe to take with other medications.

Ask your health question...

## Disease Prediction Page:

Deploy

22

-

+

Male

▼

None

✕

None

✕

None

✕

HealthAI - Intelligent Healthcare Assistant

Disease Prediction System

Enter symptoms and patient data to receive potential condition predictions.

Current Symptoms

Describe symptoms in detail (e.g., persistent headache for 3 days, fatigue, mild fever of 10.5°F)

Generate Prediction

Description: The webpage presents sophisticated diagnostic assessment tool featuring an intuitive symptom documentation section where users can detail their health concerns in depth. The interface includes a prominent analysis initiation control that, when activated, processes entered data to generate a comprehensive analysis of possible medical conditions. Results are presented with statistical probability indicators and detailed explanatory content for each potential diagnosis. The system incorporates essential medical advisories emphasizing the informational nature of the analysis and recommending professional healthcare consultation.

## Disease Prediction Output:

Patient Profile

Name

Rithvik

Age

22

Gender

Male

Medical History

None


Current Medications


None

Allergies

None

Deploy


 **HealthAI - Intelligent Healthcare Assistant**

 **Disease Prediction System**

Enter symptoms and patient data to receive potential condition predictions.

Current Symptoms

Dry cough, Shortness of breath, Loss of taste or smell, Extreme tiredness, called fatigue, Digestive symptoms such as upset stomach, vomiting or loose stools, called diarrhea, Pain, such as headaches and body or muscle aches, Fever or chills.



Potential Conditions

1. COVID-19 2. Bronchitis 3. Pneumonia

## Treatment Plans Page:

Patient Profile

Name

Rithvik

Age

22

Gender

Male

Medical History

None


Current Medications


None

Allergies

None

Deploy

 **HealthAI - Intelligent Healthcare Assistant**

 **Personalized Treatment Plan Generator**

Generate personalized treatment plans based on patient data and medical conditions.

Generate Plan

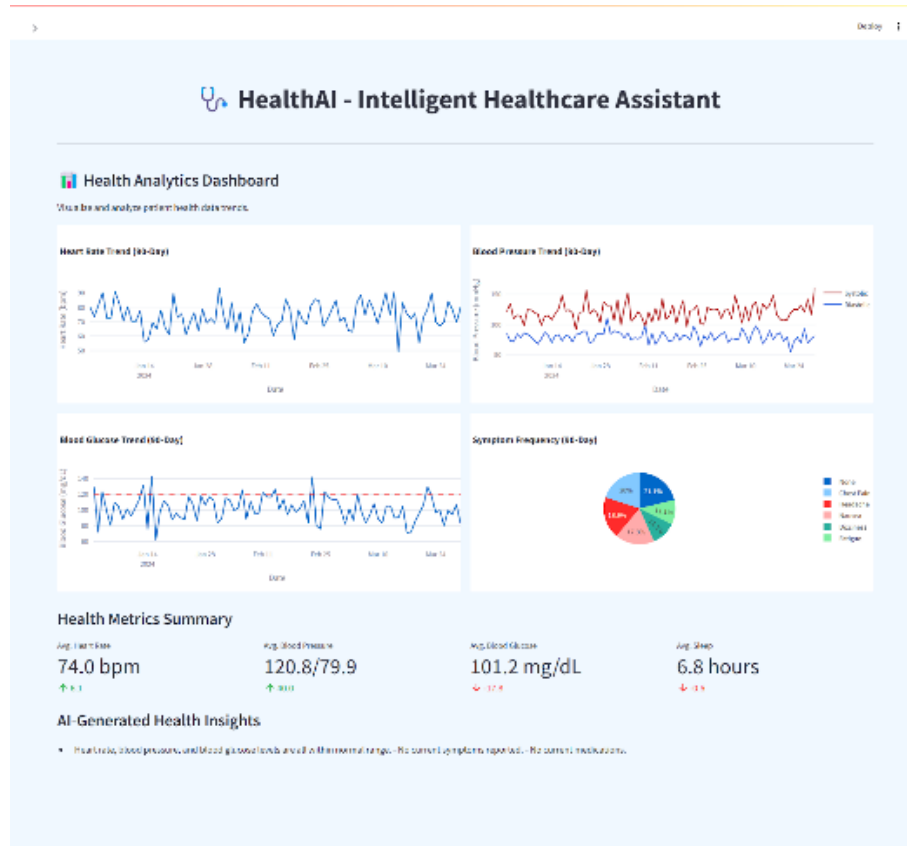
Personalized Treatment Plan

1. Recommended treatment plan for COVID-19, including self-isolation, hydration, and over-the-counter medications to manage symptoms.

## Description:

This page contains medical condition entry area where users can specify their health concerns, complemented by a prominent action button for plan creation. Upon activation, the system produces comprehensive, organized therapeutic recommendations tailored to the specified condition. The interface includes essential medical advisories clarifying the informational nature of the provided guidance.

## Health Analytics Dashboard:



Description: The interface features dynamic visualizations including heart rate trends, blood pressure patterns, and blood glucose fluctuations over time. A color-coded pie chart illustrates symptom occurrence frequency, while the metrics summary section provides key health indicators with directional trend markers. The dashboard is enhanced with AI-powered insights that analyze collected data to offer personalized health recommendations and observations.

## Conclusion

The HealthAI project effectively demonstrates the potential of AI in revolutionizing healthcare assistance. By integrating IBM's Granite language model, the platform enables users to receive personalized health insights through Patient Chat, Disease Prediction, Treatment Plan Generation, and Health Analytics, making healthcare information more accessible.

Utilizing IBM Watson Machine Learning, the application ensures



accurate health question answering, detailed disease prediction, personalized treatment recommendations, and insightful health trend analysis. The structured development process—spanning model selection, core feature implementation, backend and frontend development, and deployment—led to the creation of an interactive, user-friendly platform.

Built with Streamlit, HealthAI facilitates seamless visualization of health data and AI-generated insights, ensuring an efficient and responsive experience. This project highlights how targeted AI models and a well-structured framework can enhance healthcare accessibility. With future scalability in mind, HealthAI has the potential to expand its capabilities, incorporating more advanced diagnostics and broader medical applications.