HealthAI: Intelligent Healthcare Assistant Using IBM Granite

HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Team ID: NM2025TMID11863

Team Size: 4

Team Leader: PRIYANKA P

Team member: HABIBUR RAHMAN S

Team member: MADHAN A

Team member: MOULISHWARAN B

**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

### Streamlit Framework Knowledge: [Streamlit Documentation](https://docs.streamlit.io/)

### IBM Watson Machine Learning: [IBM Watson ML Documentation](https://www.ibm.com/docs/en/watson-studio/1.2.0?topic=models-watson-machine-learning)

### Python Programming Proficiency: [Python Documentation](https://docs.python.org/3/)

### Data Visualization Libraries: [Plotly Documentation](https://plotly.com/python/)

### Version Control with Git: [Git Documentation](https://git-scm.com/doc)

### Development Environment Setup: [Flask Installation Guide](https://www.google.com/search?q=Development+Environment+Setup%3A+Flask+Installation+Guide&oq=Development+Environment+Setup%3A+Flask+Installation+Guide&gs_lcrp=EgZjaHJvbWUyBggAEEUYOdIBBzM0MmowajSoAgCwAgA&sourceid=chrome&ie=UTF-8)

#### 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.

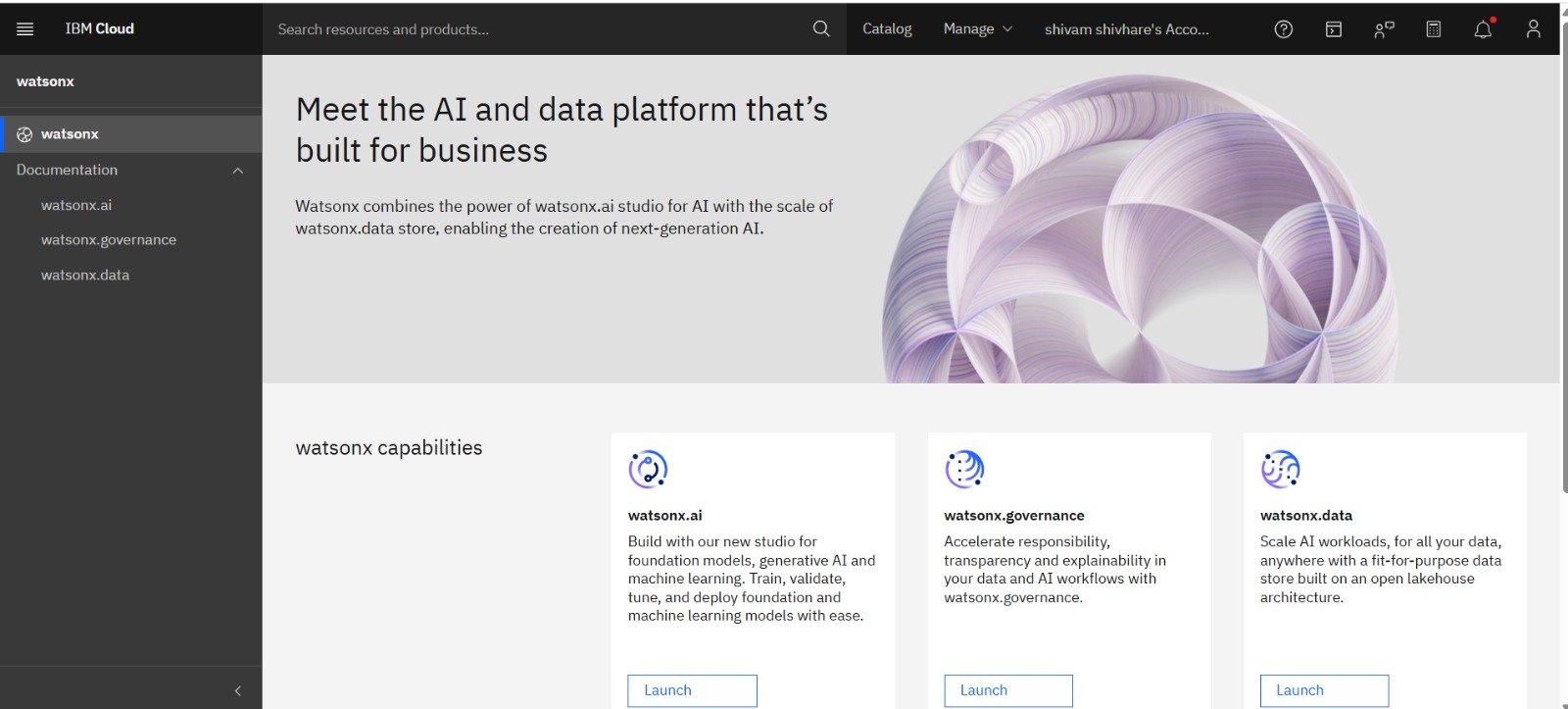
### Activity 1.2: Setup and Access your IBM WatsonX API key.

### 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

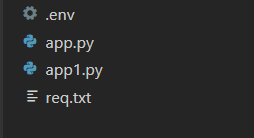
1. Draft an Architectural Diagram: Create a visual representation of the application architecture.
2. Detail Frontend Functionality: Outline how users will interact with the application through a Streamlit interface.
3. Outline Backend Responsibilities: Specify how the backend will process user input 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

1. **Disease Prediction System:**
   * Create symptom input interface
   * Develop prediction function using patient data and reported symptoms
   * Structure output format to show potential conditions with likelihood and next steps
2. **Treatment Plan Generator:**
   * Build input interface for condition and patient details
   * Create pro
   * mpting system for personalized treatment plans
   * Structure output to include medications, lifestyle changes, and follow-up care
3. **Health Analytics Dashboard:**
   * Implement patient data visualization with interactive charts
   * Create metrics summary with trend indicators
   * 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](https://www.geeksforgeeks.org/machine-learning/imbalanced-learn-module-in-python/" \t "_blank)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")

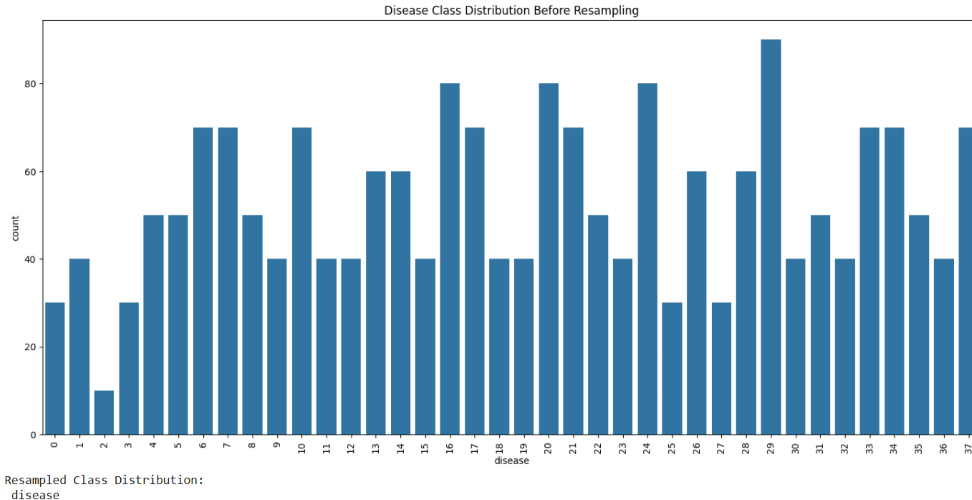
plt.xticks(rotation=90)

plt.show()

ros = RandomOverSampler(random\_state=42)

X\_resampled, y\_resampled = ros.fit\_resample(X, y)

**Output:**



### Activity 2.2: Implement data management utilities

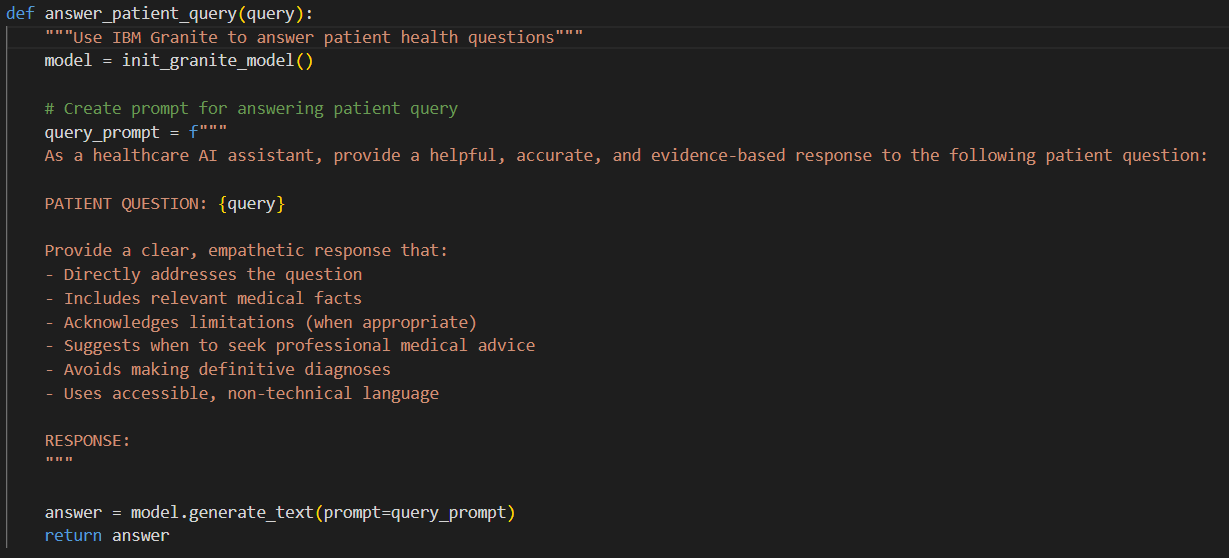
1. **Patient Data Generation:**
   * Create sample data with realistic health metrics
   * Implement date-based trend generation for visualization
   * Structure data for efficient analysis and display
2. **Patient Profile Management:**
   * Develop interface for managing patient details
   * Create session state handling for persistent data
   * Implement profile update functionality

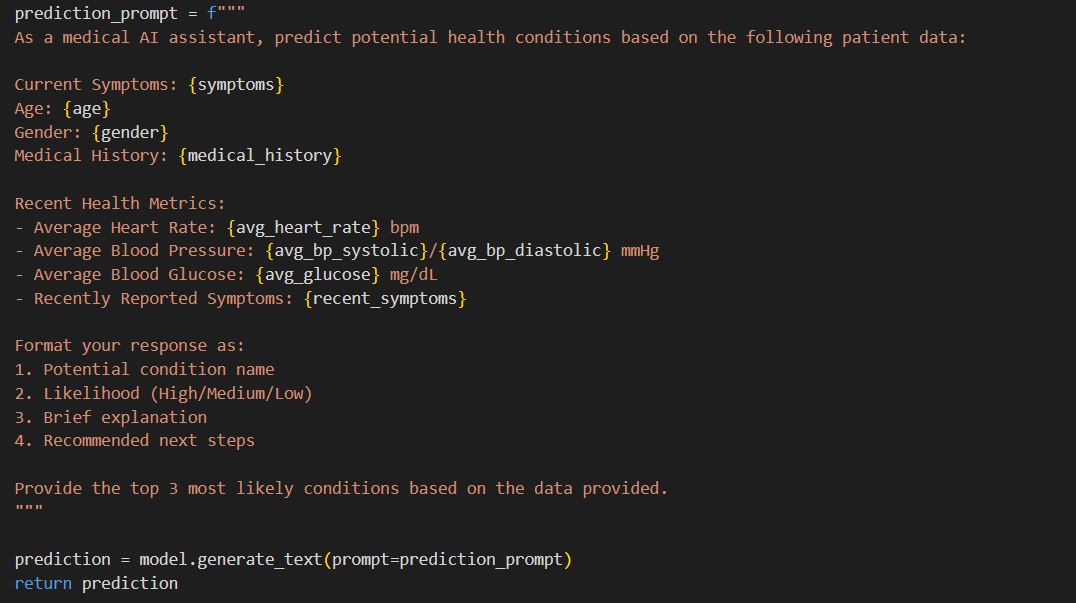
* and return JSON responses.

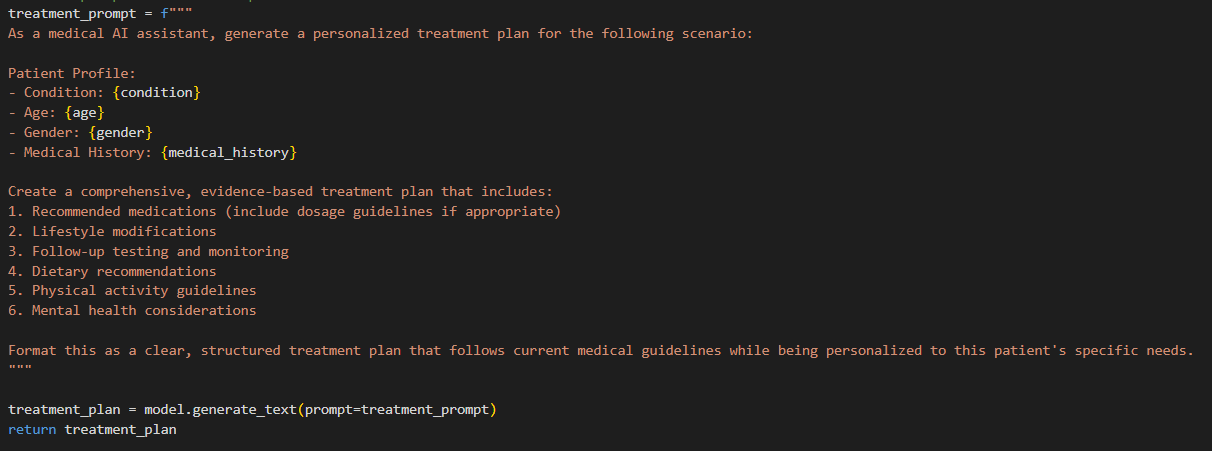
1. **Integrate the Gemini AI Responses in Each Function**
   * Implement function calls within each route to leverage Gemini's models based on user-provided details.
   * Ensure each response from Gemini API is processed in a way that enhances readability and displays clearly to the use.
   * Implement profile update functionality
2. **Imports and Setup:**
   * Import necessary libraries (Streamlit, pandas, plotly, IBM Watson ML)
   * Load environment variables for API keys
   * Initialize IBM Granite model connection
3. **Core Functions:**
   * init\_granite\_model(): Set up connection to IBM Watson ML
   * predict\_disease(): Analyze symptoms for potential diagnoses
   * generate\_treatment\_plan(): Create personalized treatment recommendations
   * answer\_patient\_query(): Process health questions with AI responses

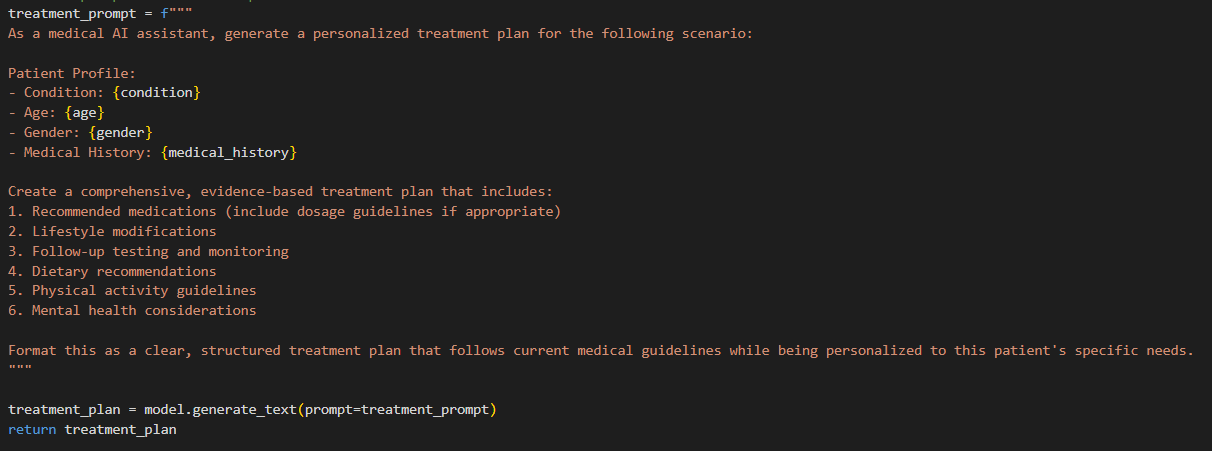
1. **UI Components:**
   * Main application layout with sidebar navigation
   * Tab-based interface for different features
   * Custom CSS styling for enhanced user experience
2. **Feature Implementation:**
   * display\_patient\_chat(): Chatbot interface for health questions
   * display\_disease\_prediction(): Symptom analysis system
   * display\_treatment\_plans(): Treatment plan generator

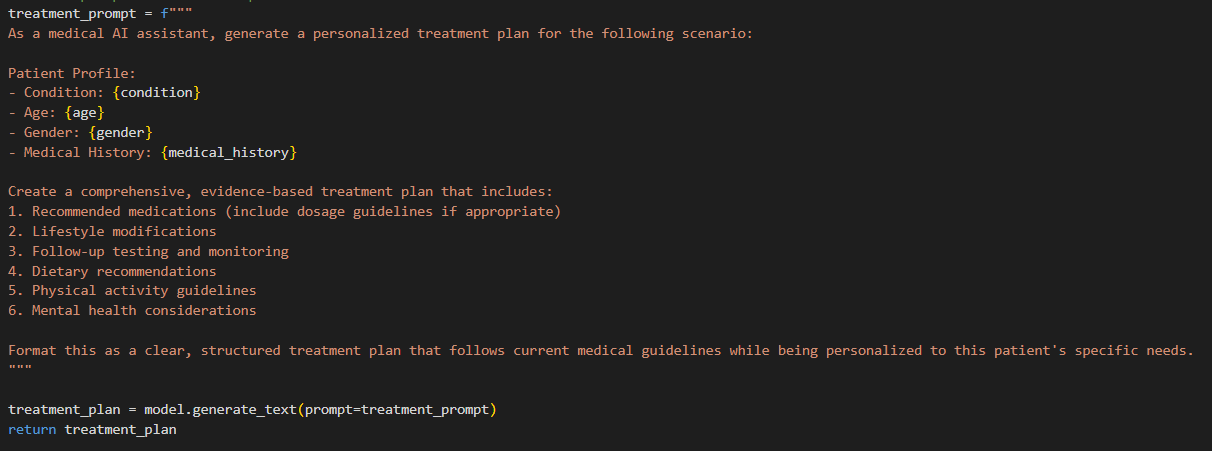
display\_health\_analytics(): Interactive health dashboard











### Milestone 4: Frontend Development

### Design and develop the user interface

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

1. **Dependency Management:**
   * Create requirements.txt file with all necessary packages
   * Document installation process for deployment

### Activity 5.2: Deploy the application

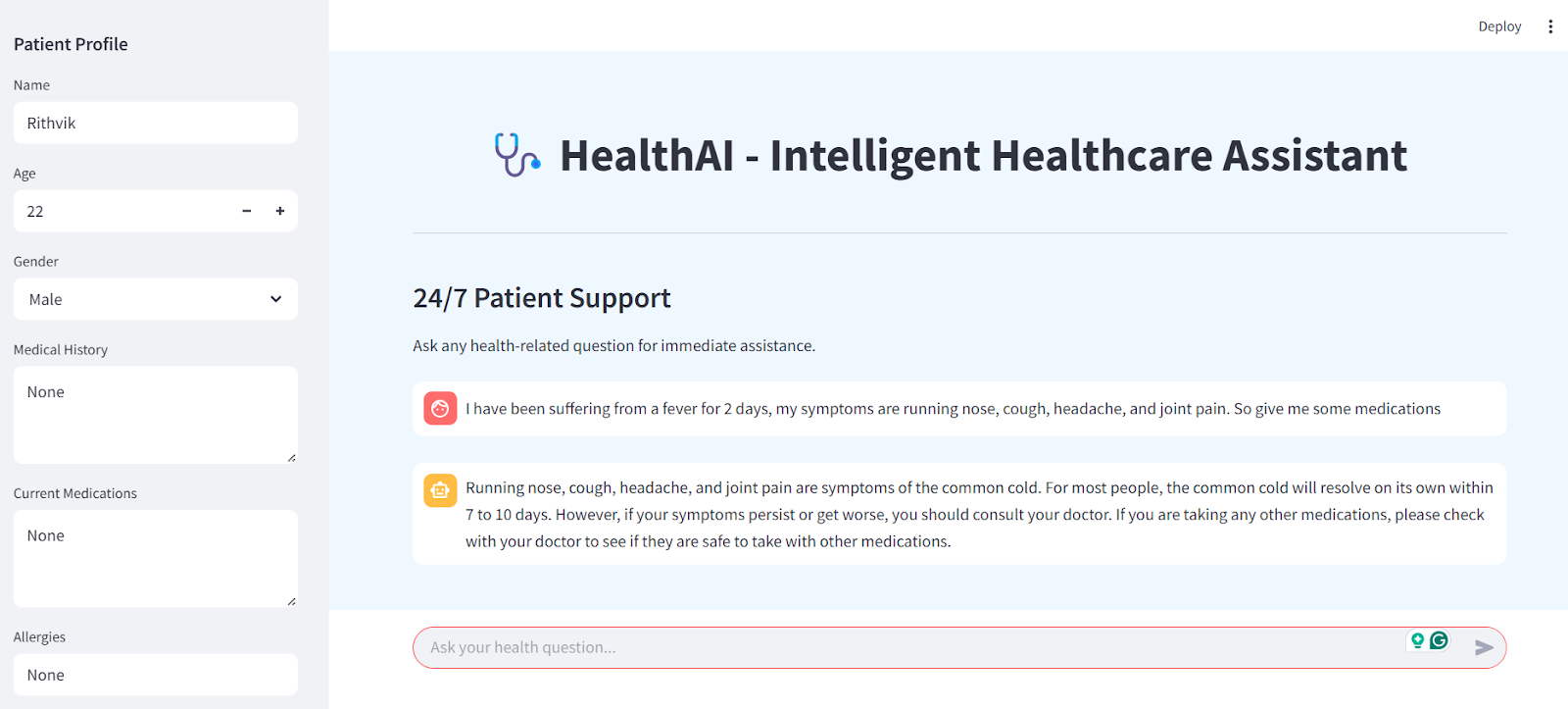
1. **Local Deployment Testing:**
   * Run the application using streamlit run app.py
   * Test all features for functionality
   * Verify responsive design and performance
2. **Cloud Deployment Options:**
   * Deploy on Streamlit Cloud for public access
   * Configure environment variables in the deployment platform
   * Set up monitoring and error logging

### Milestone 6: functional testing and verify

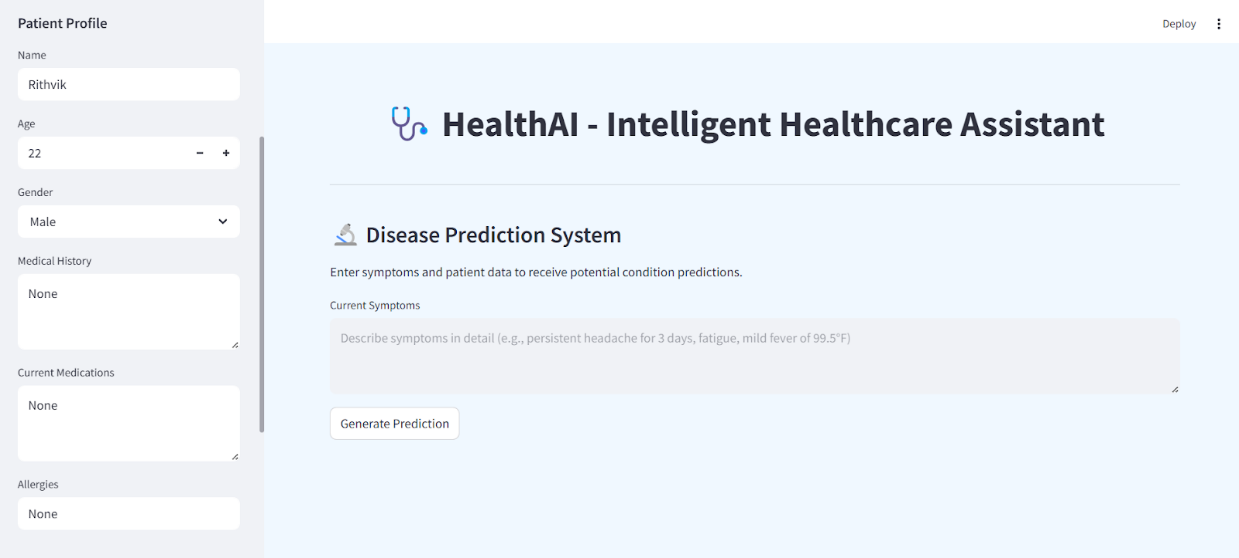
**Patient Chat Page**



**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.

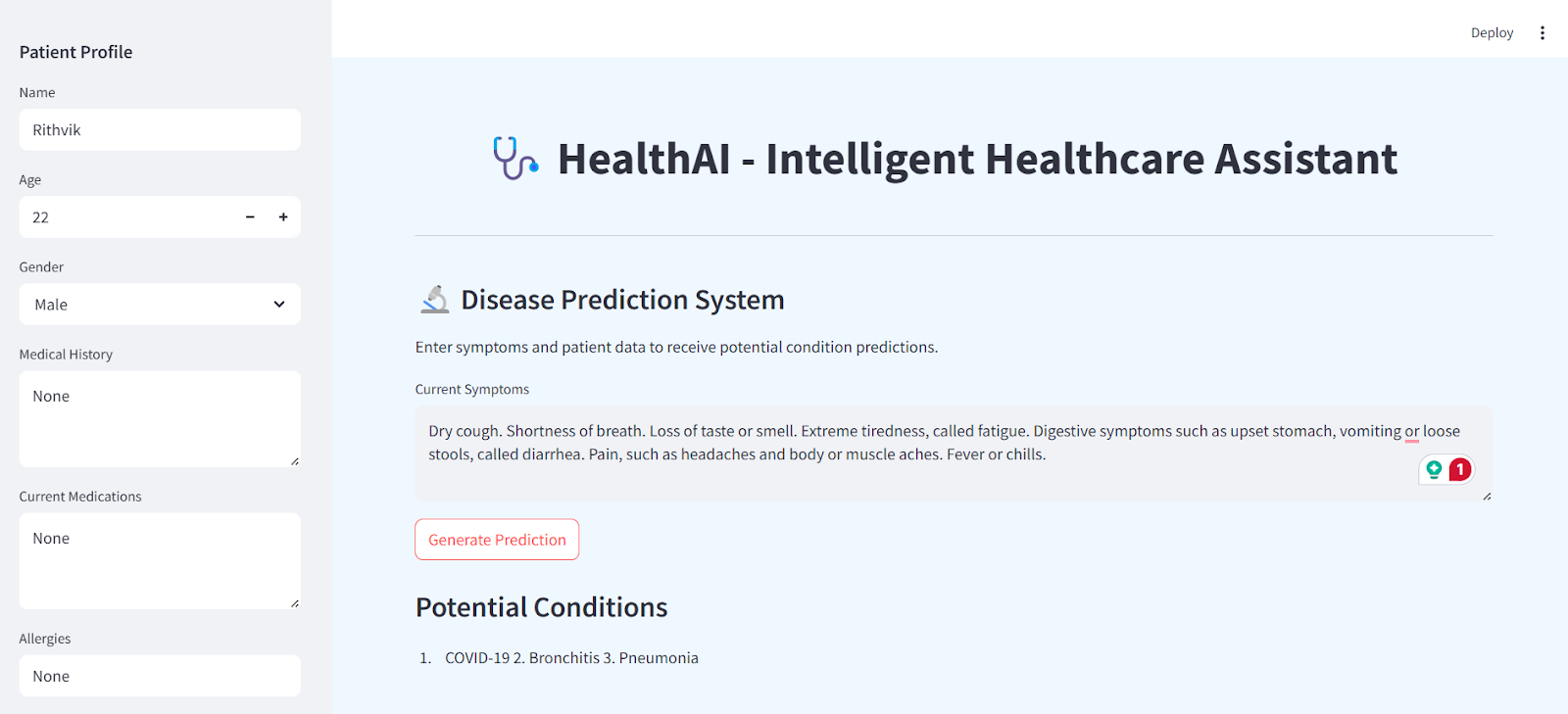
**Patient Chat output:  
**

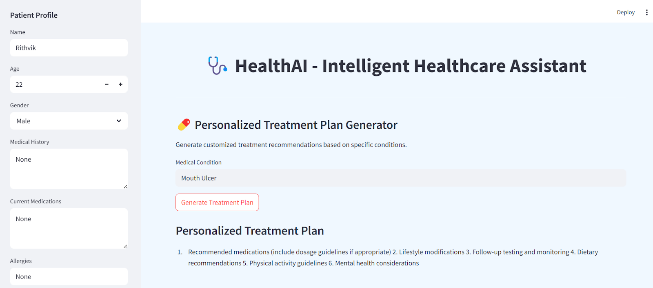
**Disease Prediction Page:**



**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:**

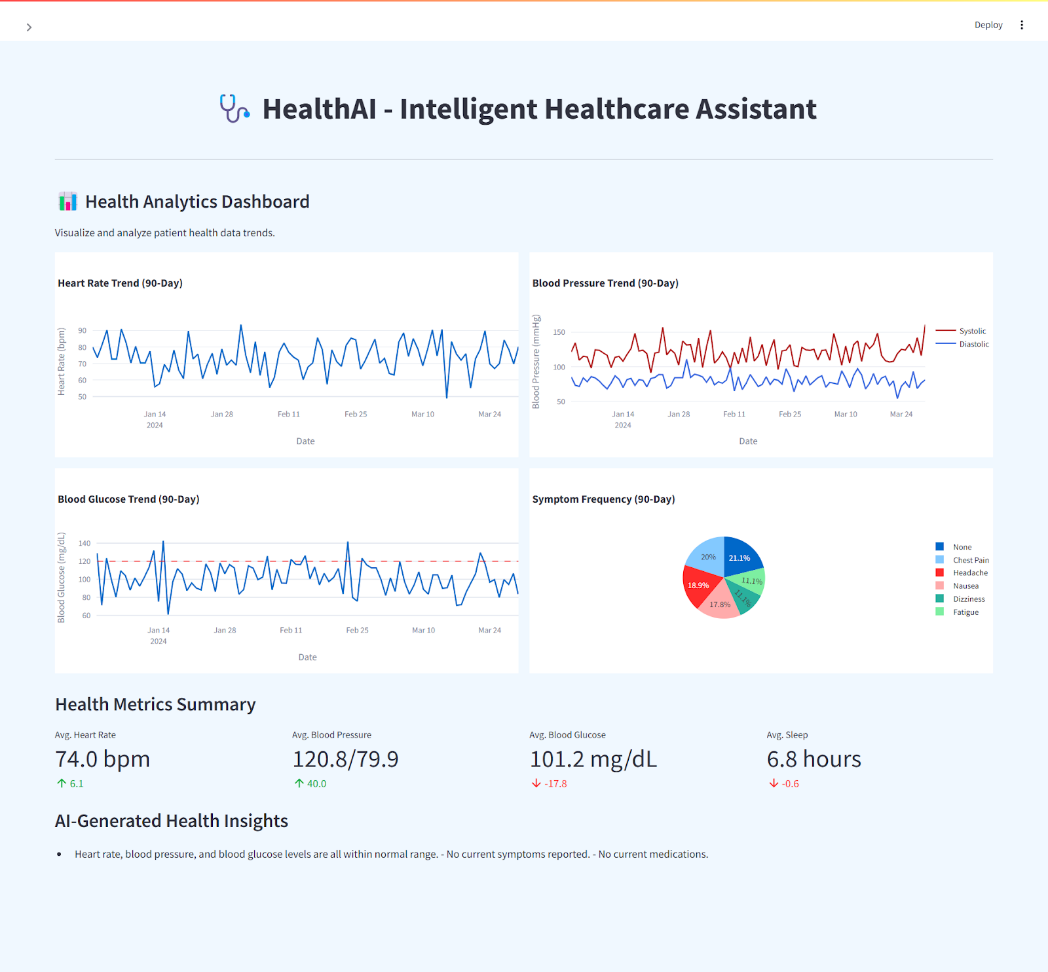


**Treatment Plans Page:  
**

**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.

### 