Health AI

Project Documentation

Introduction:

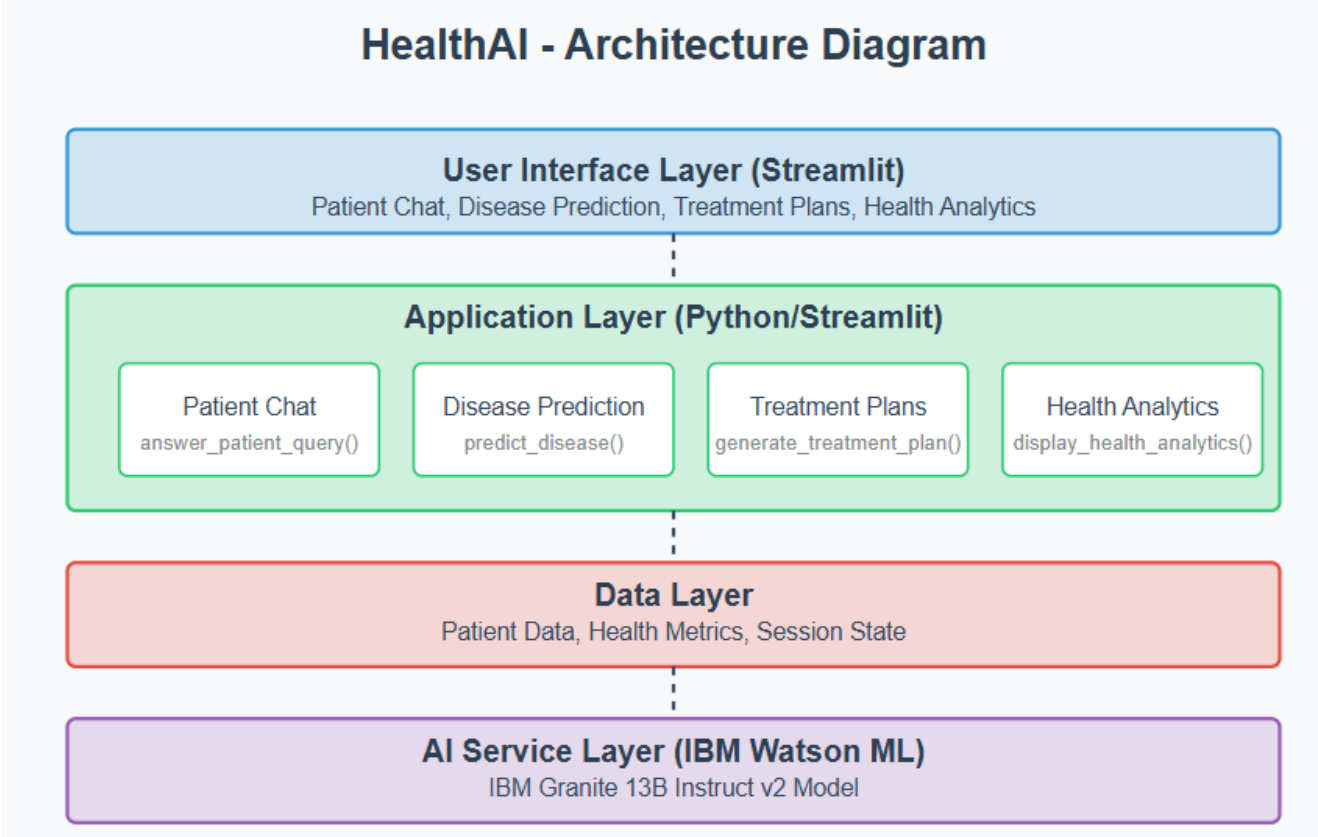
- Project title : Health AI
- Team member : Dharshini.N
- Team member : Monisha.A
- Team member :Gugapriya.R
- Team member :Pavunkumar.L

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.

HEALTH AI – Architecture Diagram:



Define the architecture of the application

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

Set up the development environment

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

  pip install streamlit pandas numpy plotly ibm-watson-machine-learning python-dotenv
- Set Up Application Structure: Create the initial directory structure for the HealthAI application.

Develop core functionalities

1. **Patient Chat System:**
   - Implement conversational interface for answering health questions
   - Create prompting system for the IBM Granite model to provide medical advice
   - Develop session-based chat history management
2. **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
3. **Treatment Plan Generator:**
   - Build input interface for condition and patient details

- Create prompting system for personalized treatment plans
- Structure output to include medications, lifestyle changes, and follow-up care

4. Health Analytics Dashboard:

- Implement patient data visualization with interactive charts
- Create metrics summary with trend indicators
- Develop AI-generated insights based on health trends

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.

3. 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

Write the main application logic:

The app.py file is organized into several key sections:

1. Imports and Setup:

- Import necessary libraries (Streamlit, pandas, plotly, IBM Watson ML)
- Load environment variables for API keys
- Initialize IBM Granite model connection

2. 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

3. UI Components:

- Main application layout with sidebar navigation
- Tab-based interface for different features
- Custom CSS styling for enhanced user experience

4. 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

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

Create dynamic visualizations:

1. Health Metric Charts:

- Heart rate trend line chart
- Blood pressure dual-line chart
- Blood glucose trend line chart with reference line
- Symptom frequency pie chart

2. Metrics Summary:

- Key health indicators with trend deltas
- Color-coded metrics to indicate normal/abnormal ranges
- Interactive tooltip information

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:

- Create requirements.txt file with all necessary packages
- Document installation process for deployment

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

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.