HEALTH AI : Intelligent Healthcare Assistant

Project Documentation

1. INTRODUCTION

●Project Title : HEALTH AI

●Project Leader : PARKAVI.S

●Project Member: SRIKRISHNA .M

●Project Member: ASHOK.S

●Project Member: AJAYKRISHNA.R

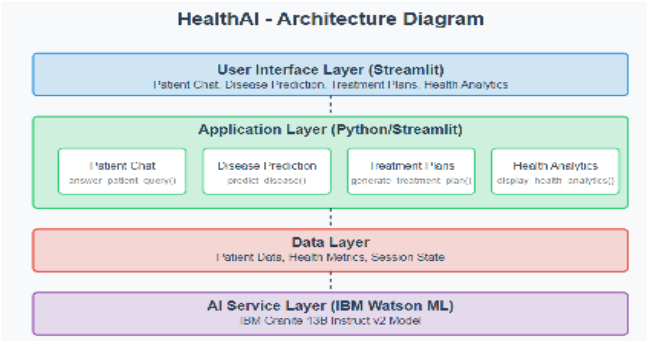
2. PROJECT OVERVIEW

●PURPOSE:

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.

3. ARCHITECTURE



4. SETUP INSTRUCTIONS

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

5. FOLDER STRUCTURE

- Streamlit Framework Knowledge: Streamlit Documentation
- IBM Watson Machine Learning: IBM Watson ML Documentation
- Python Programming Proficiency: Python Documentation
- Data Visualization Libraries: Plotly Documentation
- Version Control with Git: Git Documentation
- Development Environment Setup: Flask Installation Guide

6. SETUP INSTRUCTION

- Research and select the appropriate AI model from IBM Watson for medical assistance (IBM Granite 13B Instruct v2).
- Setup and Access your IBM Watson X API key.
- Define the architecture of the application, detailing interactions between the frontend, backend, and AI integration.
- Set up the development environment, installing necessary libraries and dependencies for Streamlit and IBM Watson ML.

7. Folder Structure

app/ – Contains all Fast API backend logic including routers, models, and

integration modules.

app/api/–Sub directory for modular API routeslike chat,feedback,report,and document vectorization.

ui/–Containsfrontend componentsfor Streamlit pages,cardlayouts,and form UIs.

smart_dashboard.py–Entry script for launchingthem a in Streamlit dashboard.

granite_llm.py–Handlesall communication with IBMWatson xGranite model including summarization and chat.

document_embedder.py–Convertsdocumentsto embeddingsand stores in Pinecone.

8. RunningtheApplication

To start the project:

Launch the Fast API server to expose backend end points.
Run the Streamlit dashboard to accessthe web interface.
Navigate through pagesvia the side bar.
Upload documentsor CSVs,inter act with the chat assistant,and view outputslike reports, summaries, and predictions.
All inter actionsarereal-time and use backend APIsto dynamically update the frontend.

Frontend(Streamlit):

The frontend isbuilt with Stream lit, offering an interactive web UI with multiple pagesincluding dashboards, file uploads, chat interface, feedback forms,and reportviewers.Navigation ishandled through aidebar usingthe stream lit-option-menu library. Each page is modularized for scalability.

Backend(FastAPI):

1. Fast API servesasthe backend REST frame work that powers APIendpointsfor document processing, chat interactions, eco tip generation, report creation, and vector embedding.It isoptimized for asynchronousperformance an deasy Swagger integration.

9. TESTING

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

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

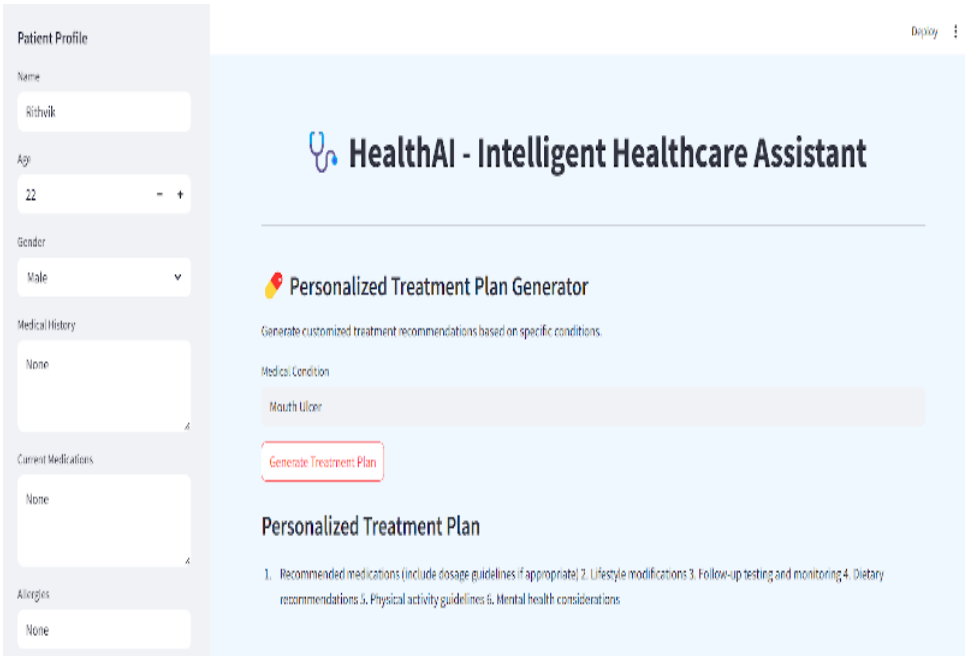
Treatment Plan Generator:

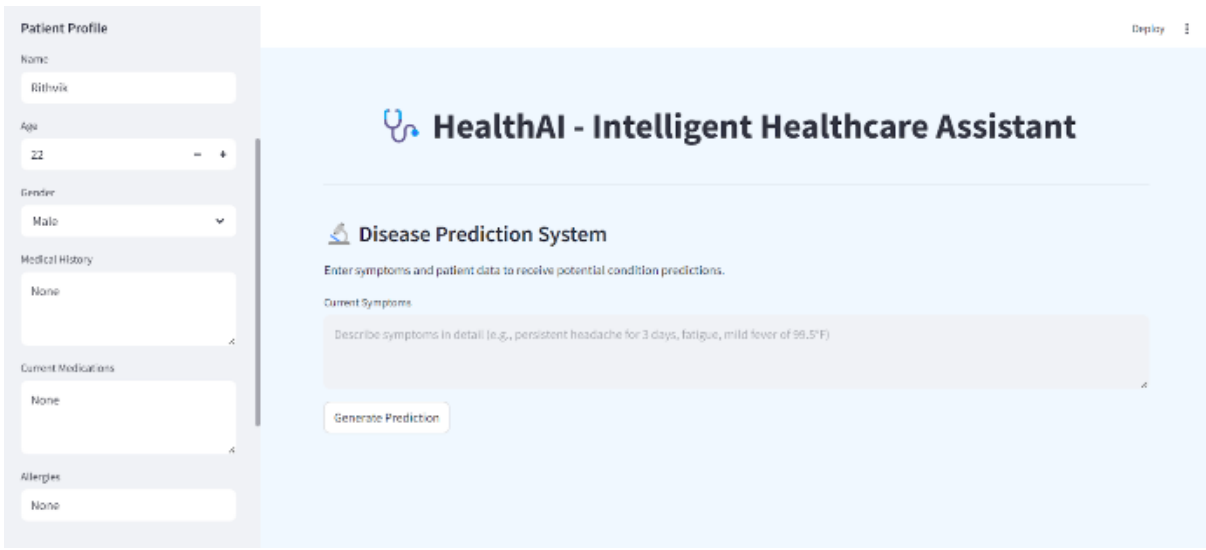
- Build input interface for condition and patient details
- Create prompting system for personalized treatment plans
- Structure output to include medications, lifestyle changes, and follow-up care

Health Analytics Dashboard:

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

10. SCREENSHOT





11. FUTURE EHANCEMENT

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.

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.

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.