Project Title

Project Documentation

1.Introduction

- Project title : Health AI: Intelligent Healthcare Assistant Using IBM Granite
- Team member : E.karthika
- Team member : G.Hemapriya
- Team member : S.janani
- Team member :B.jenifer

2.project overview

- Purpose :

  The purpose of this Health AI assistant is to improve healthcare delivery and empower both patients and medical professionals with intelligent, data-driven support. By leveraging AI and real-time health data, the assistant provides:

  Personalized health guidance for patients.

  Summaries of medical records and treatment guidelines for doctors.

  Predictive analytics to forecast patient outcomes and detect anomalies in vital signs.

  Decision-making support for healthcare staff with actionable insights.

  Ultimately, this assistant bridges technology, healthcare providers, and patient engagement to create a more accessible, efficient, and patient-centered healthcare ecosystem.

- Features:

Conversational Interface

Key Point: Natural language interaction

Functionality: Patients and doctors can ask health-related questions, receive symptom guidance, and access records in plain language.

Medical Record Summarization

Key Point: Simplified medical understanding

Functionality: Converts lengthy medical histories, prescriptions, or guidelines into concise, actionable summaries.

Health Forecasting

Key Point: Predictive analytics

Functionality: Predicts patient health risks, disease progression, or hospital resource needs based on historical and real-time health data.

Personalized Health Tips

Key Point: Patient-specific guidance

Functionality: Provides daily lifestyle and wellness advice (diet, exercise, reminders) tailored to patient conditions.

Patient Feedback Loop

Key Point: Continuous improvement

Functionality: Collects patient feedback on treatments and services to improve healthcare delivery.

KPI Forecasting for Hospitals

Key Point: Strategic planning support

Functionality: Projects hospital performance indicators such as patient inflow, recovery rates, and staff utilization.

Anomaly Detection

Key Point: Early health warning system

Functionality: Identifies abnormal trends in patient vitals (e.g., blood pressure spikes, irregular heart rates).

Multimodal Input Support

Key Point: Flexible data handling

Functionality: Accepts text, PDFs, and CSVs for medical reports, prescriptions, and datasets.

Streamlit or Gradio UI

Key Point: User-friendly interface

Functionality: Provides a simple dashboard for patients, doctors, and hospital administrators to interact with the assistant.

3. Architecture

Frontend (Streamlit):
Interactive web UI with modules for health dashboards, medical uploads, patient chat, feedback, and report viewing.

Backend (FastAPI):
Manages API endpoints for medical document processing, chat, health-tip generation, report creation, and vector embeddings.

LLM Integration (IBM Watsonx Granite):
Granite models handle medical language understanding, patient guidance, and report summarization.

Vector Search (Pinecone):
Stores embedded medical records for fast semantic search using natural language queries.

ML Modules (Forecasting & Anomaly Detection):
Forecasts patient outcomes and detects irregularities using scikit-learn with time-series health data.

4. Setup Instructure

Prerequisites:

Python 3.9+

Pip & virtual environment

IBM Watsonx & Pinecone API keys

Internet access

Installation:

1. Clone the repository

2. Install requirements (requirements.txt)

3. Configure .env with credentials

4. Start FastAPI backend

5. Launch Streamlit frontend

6. Upload patient records & interact with assistant

5. Folder Structure

app/ → FastAPI backend logic

app/api/ → API routes (chat, health tips, medical record processing)

ui/ → Streamlit frontend (dashboards, chat UI, forms)

health_dashboard.py → Main Streamlit entry script

granite_llm.py → Watsonx Granite integration for summaries & chat

document_embedder.py → Embeds medical records into Pinecone

kpi_forecaster.py → Forecasts hospital/patient KPIs

anomaly_checker.py → Detects anomalies in vitals or patient data

report_generator.py → Generates AI-based health reports

6. Running the Application

1. Start FastAPI backend

2. Run Streamlit frontend

3. Use sidebar to navigate

4. Upload patient data or medical files

5. Get real-time chat guidance, reports, and predictions

7. API Documentation

POST /chat/ask → AI health query response

POST /upload-doc → Upload & embed patient records

GET /search-docs → Search medical guidelines/records

GET /get-health-tips → Get daily patient wellness tips

POST /submit-feedback → Collect patient feedback

All APIs are tested with Swagger UI & Postman.

8. Authentication

Current version runs open for demo.

Secure deployments can use:

JWT tokens / API keys

OAuth2 with IBM Cloud credentials

Role-based access (doctor, patient, admin)

9. User Interface

Sidebar navigation

Patient health visualizations (graphs, KPIs)

Tabs for sechat, reports, and tips

Downloadable medical summaries

Accessibility-focused design

10. Testing

Unit Testing → Prompt engineering & scripts

API Testing → Swagger UI & Postman

Manual Testing → Uploads, chat & outputs

Edge Case Handling → Invalid medical inputs, large records

11.screen shots

To add

12. Known Issues

Limited offline support

Requires stable internet for IBM Granite API

13. Future enhancement

Integration with IoT health devices (wearables, sensors)

Multilingual support for local healthcare delivery

Advanced disease prediction using deep learning