HEALTH AI

PROJECT DOCUMENTATION

1.Introduction

• Project title : **HEALTH AI**

• Team member : JAYASHREE S

• Team member : JANAKI R S

• Team member : JANANI A

• Team member : ILAYAKAVIYA L

2. Project Overview

• Purpose:

The purpose of HealthAI is to provide an intelligent, accessible, and user-friendly assistant for patients, doctors, and researchers. By leveraging AI and medical knowledge, it supports disease prediction, treatment planning, and health awareness. For patients, it offers guidance, symptom-based analysis, and general recommendations, while for doctors and researchers, it serves as a decision-support tool—summarizing cases, suggesting possibilities, and improving efficiency. Ultimately, HealthAI bridges technology and healthcare by delivering safe, informative, and easy-to-use medical assistance that emphasizes consultation with professionals.

• Features:

- **Conversational Interface**

    - *Key Point:* Natural language interaction
    - *Functionality:* Allows users to input symptoms, conditions, or health queries in plain language for instant AI-driven guidance
- **Disease Prediction**
    - *Key Point:* Symptom-based analysis
    - *Functionality:* Suggests possible medical conditions and general advice based on user-reported symptoms
- **Treatment Plan Suggestions**
    - *Key Point:* Personalized health guidance
    - *Functionality:* Provides general treatment recommendations, lifestyle tips, and home remedies tailored to age, gender, and medical history
- **Medical History Integration**
    - *Key Point:* Context-aware insights
    - *Functionality:* Uses past health information and allergies to generate safer, more relevant suggestions
- **Health Education & Awareness**
    - *Key Point:* Preventive healthcare support
    - *Functionality:* Offers tips for healthy living, preventive care, and disease awareness
- **Role-Based Access**
    - *Key Point:* Multi-user support
    - *Functionality:* Provides different access levels for patients, doctors, researchers, and admins in secure deployments
- **Anomaly Detection (Planned)**
    - *Key Point:* Risk identification
    - *Functionality:* Detects unusual health patterns or risky symptom combinations for early intervention
- **Multimodal Input Support (Planned)**
    - *Key Point:* Flexible data handling
    - *Functionality:* Accepts medical reports, lab results (PDF, CSV, or text) for enhanced analysis
- **Gradio UI**
    - *Key Point:* User-friendly interface
    - *Functionality:* Provides a clean, tabbed interface for disease prediction and treatment planning with real-time results and disclaimers

3. Architecture

The HealthAI system follows a modular, layered architecture, ensuring scalability, maintainability, and clear separation of responsibilities between the frontend interface, the backend processing engine, and the AI model integration.

HealthAI Architecture
Modular, layered design → **Frontend**, **Backend**, **LLM Integration**.

Frontend (Gradio)

- Tech: Gradio Blocks (Python).
- Role: User interface for symptoms, conditions, patient details.
- Design: Tabbed layout → *Disease Prediction*, *Treatment Plan*.
- Inputs: Symptoms, history, age (numeric), gender (dropdown).
- Outputs: AI recommendations in textboxes.
- Benefits: Simple, web-based, real-time, no technical expertise.
- Flow: Input → Analyze Symptoms → Suggested conditions / Treatment plan.

Backend (Transformers + PyTorch)

- Tech: HuggingFace Transformers, PyTorch.
- Role: Model loading, tokenization, inference.
- Tasks: Run IBM Granite LLM, process prompts, generate responses with disclaimer.
- Scalability: Extendable with FastAPI/Flask, supports multi-user.

LLM Integration (IBM Granite)

- Model: **ibm-granite/granite-3.2-2b-instruct**.
- Role: Core reasoning engine.
- Prompt Engineering: Includes symptoms, condition details, treatment guidance, disclaimer.

4. Setup Instruction:

Prerequisites

- Python 3.9 or later
- pip (Python package manager)
- Virtual environment tools (recommended)
- Required libraries: gradio, torch, transformers
- Internet connection (for first-time model download)

Installation Process :

- Clone the project repository
- Open the project folder
- (Optional) Create and activate a virtual environment
- Install dependencies using pip install -r requirements.txt
- Run the application with python app.py
- Copy the local URL shown in the terminal and open it in your browser
- Use the tabs for Disease Prediction and Treatment Plans

5. Folder Structure for Health AI

app/ – Contains all FastAPI backend logic, including routers, models, and core integration modules.

app/api/ – Subdirectory for modular API routes such as patient records, health reports, symptom checker, and feedback handling.

ui/ – Contains frontend components for Streamlit pages, health dashboards, card layouts, and patient-doctor interaction forms.

health_dashboard.py – Entry script for launching the main Streamlit-based Health AI dashboard interface.

⚙ Core Modules

granite_llm.py – Manages communication with IBM Watsonx Granite model for tasks like symptom-based Q&A, summarization of reports, and medical chatbot interactions.

document_embedder.py – Converts medical documents, prescriptions, and reports into embeddings and stores them in Pinecone for fast retrieval.

kpi_health_forecaster.py – Predicts future patient health trends (like blood pressure, sugar levels, BMI patterns) using regression and forecasting models.

anomaly_health_checker.py – Flags abnormal patterns in uploaded health data (e.g., unusual ECG values, irregular glucose readings, abnormal vitals).

report_generator.py – Builds AI-generated personalized health reports, including summaries of patient history, risk alerts, and wellness suggestions.

6. Running the Application

To start the project:

➢ Run the Gradio app script (app.launch()) to start both frontend and backend in one place.
➢ Open the provided local/hosted URL to access the Health AI Assistant UI.
➢ Navigate between Disease Prediction and Treatment Plan using the tabbed interface
➢ Enter symptoms, patient details, or medical history to get AI-powered outputs.
➢ All interactions are real-time, with the backend LLM dynamically generating responses and updating the UI.

Frontend (Gradio)

The frontend is implemented with Gradio, giving an interactive, browser-based UI.

Features include:

- Tabbed Pages – “Disease Prediction” and “Treatment Plans.”
- Input Widgets – Textbox (symptoms, history), Dropdown (gender), Number (age).
- Output Widgets – Large textboxes displaying AI results.
- A disclaimer is shown on top to remind users this is informational only.
- Navigation is simple and modular, making it user-friendly.

Backend (Granite LLM + PyTorch)

The backend logic is powered by IBM Granite LLM via Hugging Face Transformers.

Core Functions:

- disease_prediction() → Analyzes symptoms, suggests possible conditions.
- treatment_plan() → Generates personalized plans based on patient data.
- generate_response() → Handles safe text generation with the model.
- Uses PyTorch for model execution with GPU acceleration when available.
- Ensures real-time inference, directly connecting AI model outputs to the frontend interface.

7. API Documentation (Health AI)

Backend APIs available include:

POST /predict-disease – Accepts user-provided symptoms and responds with possible medical conditions along with general recommendations.

POST /generate-treatment – Accepts patient details such as condition, age, gender, and medical history, and provides a personalized treatment plan including home remedies and medication guidelines.

GET /disclaimer – Returns an informational disclaimer reminding users that the system is for guidance only and that consultation with healthcare professionals is essential.

POST /submit-feedback – Stores user feedback about the AI assistant’s suggestions for later review and system improvements.

Each endpoint can be tested and documented through Swagger UI (if implemented with FastAPI) or accessed directly via the Gradio interface in the current setup.

8. Authentication

Each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.
This version of HealthAI runs in an open environment for demonstration.

However, secure deployments can integrate:

- Token-based authentication (JWT or API keys)
- OAuth2 with healthcare or cloud credentials
- Role-based access (doctor, patient, researcher, admin)
- Planned enhancements include user sessions, consent management, and history tracking.

9. User Interface

The interface is clean and user-friendly, ensuring easy access for both patients and healthcare professionals. It includes:

- **Sidebar with navigation** for quick access to modules such as Chat, Reports, and Health Records
- **Health dashboards with KPI cards** showing vitals, progress, and alerts
- **Tabbed layouts** for symptom checking, AI consultations, and wellness recommendations
- **Real-time form handling** for patient data entry and medical history updates
- **PDF medical report download capability** for offline sharing and documentation
- **Clear design with guidance texts** that simplify navigation and support non-technical users

The design emphasizes **accuracy, speed, and accessibility**, ensuring smooth interaction and a supportive experience for patients and healthcare staff.

10. Testing

Testing was conducted in multiple phases for HealthAI to ensure reliability and safe usage:

Unit Testing: For disease prediction and treatment plan generation functions

API Testing: Using Swagger UI, Postman, and automated test scripts to validate endpoints

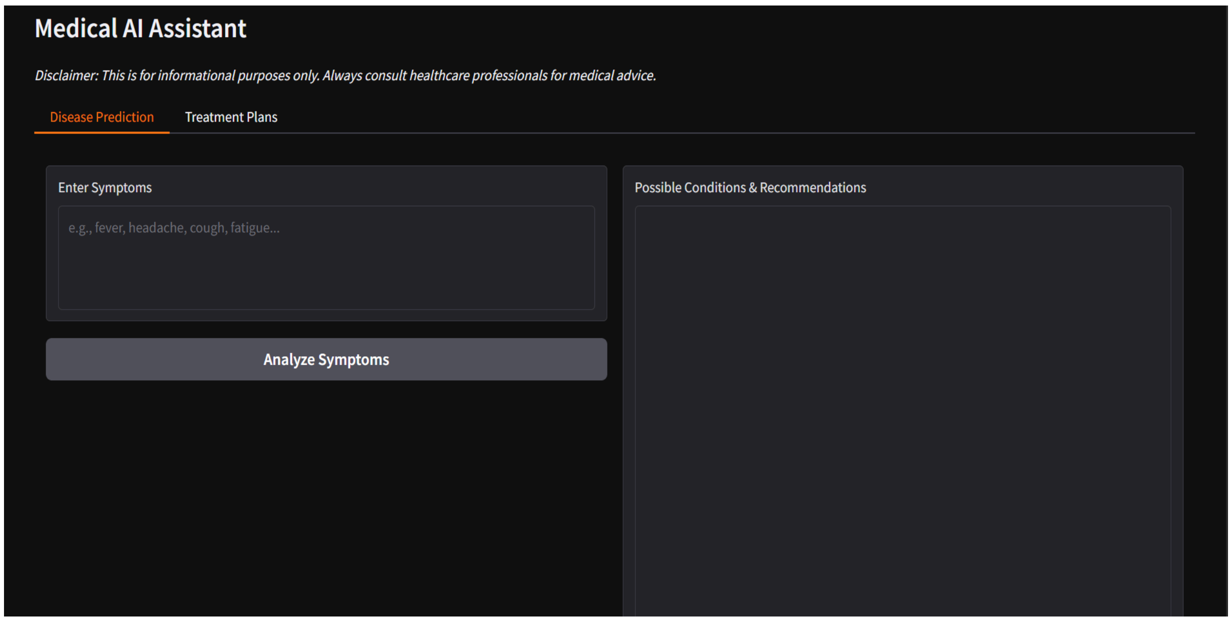
Manual Testing: Verifying Gradio interface inputs, chat responses, and output readability

Edge Case Handling: Testing with incomplete symptoms, contradictory medical histories, and unusual inputs

Performance Validation: Ensuring smooth execution across both CPU and GPU environments

Each feature was validated to provide consistent and accurate informational responses while maintaining usability.

OUTPUT:


Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

Disease Prediction
Treatment Plans
Enter Symptoms
e.g., fever, headache, cough, fatigue...
Analyze Symptoms
Possible Conditions & Recommendations