HEALTH AI - Project Report

Project Heading: Health AI

Team Members:

1. Michael Raja C (Leader)
2. Samuvel Joshwa P
3. Samvel H
4. Praveen V

1. Introduction

The Health AI project leverages IBM Granite AI models to assist in medical support tasks such as disease prediction and treatment plan suggestions. It demonstrates the application of artificial intelligence in healthcare through a user-friendly Gradio interface. Disclaimer: This tool is for informational purposes only and not a replacement for professional medical advice.

2. Project - Purpose

The purpose of Health AI is to:
- Provide quick insights into possible conditions from reported symptoms.
- Suggest treatment approaches considering patient age, gender, and medical history.
- Explore the safe use of AI in healthcare support.

Features

- Conversational Interface – Interactive natural language support.
- Policy Summarization – Simplifies lengthy healthcare documentation.
- Resource Forecasting – Predictive use of patient/health data.

- Eco-Tip Generator – Wellness suggestions for healthy living.
- Multimodal Input Support – Accepts text and structured inputs.
- Streamlit or Gradio UI – Simple, interactive dashboards.

3. Architecture

Frontend (Streamlit/Gradio): Provides user interface with tabs for symptoms analysis and treatment plans.
Backend (FastAPI): Handles API endpoints, connects to IBM Granite model, processes user inputs, and returns predictions.

4. Setup Instructions

Prerequisites:
- Python 3.9 or later
- pip package manager
- IBM Granite model access
- Internet connectivity

Installation Process:
1. Clone repository.
2. Install dependencies (transformers, torch, gradio).
3. Configure environment with credentials if required.
4. Run FastAPI backend server.
5. Launch frontend via Streamlit or Gradio.

5. Folder Structure

- app/ – Backend logic
- api/ – API routes
- ui/ – Gradio/Streamlit UI components
- model/ – Granite model loading
- notebooks/ – Jupyter/Colab files
- scripts/ – Utility scripts for processing

6. Running the Application

1. Start the backend server with FastAPI.
2. Run the Gradio app for UI.
3. Enter symptoms for disease prediction.
4. Provide patient details for treatment plans.
5. View real-time AI responses and recommendations.

7. API Documentation

- POST /predict-disease – Accepts symptoms and returns possible conditions.
- POST /treatment-plan – Generates treatment suggestions from patient data.
- GET /status – Health check of API.

8. Authentication

This demo runs in an open environment. For production, security can include:
- Token-based authentication (JWT)
- OAuth2 with IBM credentials
- Role-based access controls

9. User Interface

The interface uses Gradio tabs for Disease Prediction and Treatment Plans.

It allows text inputs for symptoms, condition, and history, plus dropdowns for gender and numeric fields for age.

10. Testing

Testing Phases:
- Unit Testing: Functions for prompt generation.
- API Testing: Tested with Postman and direct calls.

- Manual Testing: Symptom and condition input validation.
- Edge Case Handling: Empty inputs, long text, invalid values.

11. Screenshots

Screenshots of the Gradio UI and outputs can be added here.

12. Known Issues

- Limited accuracy due to model generalization.
- Requires GPU for fast responses.
- No integration with medical databases yet.

13. Future Enhancements

- Multi-language support for global accessibility.
- Integration with real medical datasets and EHR.
- Mobile app deployment.
- Advanced analytics and visualization.