Health AI – Project Documentation

1. Introduction

Project Title: Health AI – AI-powered Medical Assistant
Team Members: (List your team members here)

2. Project Overview

Purpose: Health AI is designed to help users understand their symptoms, get general health information, and receive personalized treatment suggestions. By leveraging advanced generative AI models, Health AI provides users with helpful, conversational, and personalized insights.
Disclaimer: The system is for informational purposes only and should not replace professional medical advice.

Feature	Key Point	Functionality
Disease Prediction	Symptom-based health guidance	Users enter their symptoms, and Health AI provides possible conditions
Personalized Treatment Plans	Tailored recommendations	Users input condition, age, gender, and medical history to get general tr
Conversational Interface	Natural language interaction	Built using Gradio for a simple, tabbed UI with clear outputs.
Generative AI Integration	IBM Granite model usage	Uses the ibm-granite/granite-3.2-2b-instruct model for natural language

3. Architecture

Frontend (Gradio): The frontend is built with Gradio’s Blocks API, offering an interactive interface with tabs for “Disease Prediction” and “Treatment Plans”. It displays input forms and output areas for AI-generated responses.
Backend (Python): The backend runs Python code integrating Hugging Face Transformers to load the IBM Granite model and tokenizer, process prompts, and generate responses.
LLM Integration (IBM Granite): The system uses IBM Granite large language models to interpret symptoms and conditions and generate natural language responses and treatment plans.

4. Setup Instructions

Prerequisites: Python 3.9 or later, pip and virtual environment tools, GPU (optional), Internet access to download model weights.
Installation Process:
1. Clone the repository.
2. Install dependencies: pip install -r requirements.txt
3. Run the application: python healthai.py
4. Open the provided Gradio link in your browser to access the interface.

5. Folder Structure

healthai.py – Main application script with model loading, prompt functions, and Gradio UI
requirements.txt – Python dependencies (Gradio, Transformers, Torch)
README.md – Project overview and quickstart instructions (to be created)

6. Running the Application

Launch the Python script. Gradio provides a local or public URL. Navigate to the URL to use the app. Disease Prediction: Enter symptoms and click 'Analyze Symptoms'. Treatment Plans: Enter condition, age, gender, and history, then click 'Generate Treatment Plan'.

7. API Documentation

Currently built as a GUI-only application, but the main functions can be exposed as APIs:

Function	Input	Output	Description
disease_prediction(symptoms)	Text (symptoms)	Text (conditions & recommendations)	Generates AI-based health analysis for given symp
treatment_plan(condition, age, gender, medical_history)	Condition, Age, Gender, Medical history	Text (detailed plan)	Produces treatment suggestions based on inputs

8. Authentication

This version runs openly for demonstration purposes. For production, you can add API keys for model access, user authentication with tokens or OAuth, and role-based access for different types of users.

9. User Interface

Two main tabs: Disease Prediction and Treatment Plans. Input Fields: Text boxes, dropdowns, and number fields for structured data. Output Fields: Large text boxes for clear, readable AI responses.

10. Testing

Testing recommendations: Unit Testing for prompt functions, Manual Testing with various symptoms and conditions, Edge Case Handling for malformed inputs.

11. Known Issues

Outputs depend on model accuracy and may vary. Responses may occasionally be incomplete due to token length limits.

12. Future Enhancements

Add multilingual support. Integrate secure API endpoints. Include symptom tracking and health history. Export treatment plans as downloadable PDF reports.