HealthAI: Intelligent Healthcare Assistant Using IBM Granite

INTRODUCTION

Project Overview

HealthAI is an AI-powered web application built using FastAPI, Transformers, and the IBM Granite 3.3-2B Instruct model. It assists users in predicting possible diseases based on symptom inputs and suggests home remedies. The system delivers accurate, accessible, and real-time health insights via a user-friendly interface.

REQUIREMENT ANALYSIS

Customer Journey Map

1. Start Application – User opens the HealthAI web app.
2. Input Symptoms – User enters symptoms via a form.
3. Predict Disease – AI model (Granite 3.3-2B) predicts the most likely disease.
4. Input Disease – User can enter the predicted disease to get a remedy.
5. Suggest Remedy – AI provides natural remedy suggestions.
6. View Results – User sees predictions and suggestions in a web interface.

Session Requirements (Revised):

- Text input form for symptoms and diseases
- FastAPI backend handling predictions
- Real-time response generation from IBM Granite model
- Frontend rendering using HTML (Jinja2) and CSS

TECHNOLOGY STACK

Layer	Tool/Technology
Frontend	HTML, CSS (Jinja2 Templates)
Backend	FastAPI (Python)
AI Model	IBM Granite 3.3-2B via HuggingFace
NLP Framework	HuggingFace Transformers
Hosting	Uvicorn or local server (for testing)

PROJECT DESIGN

Solution Architecture

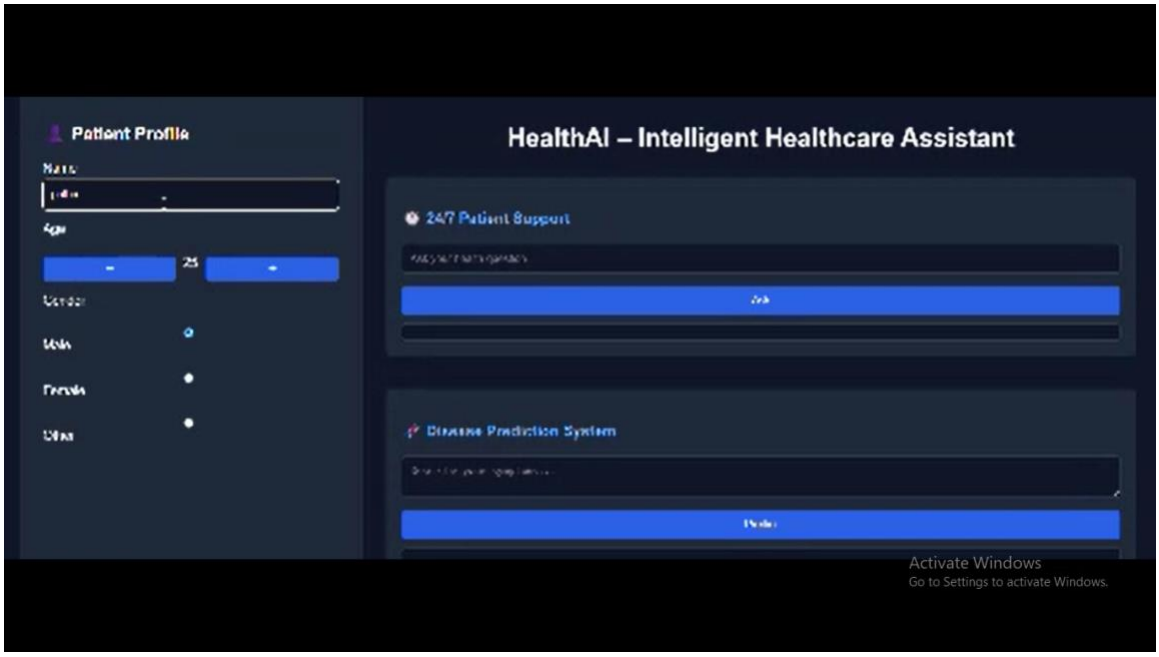
- UI Layer: HTML form (index.html) for symptom/disease entry.
- Backend Logic: FastAPI app with /predict and /remedy endpoints.
- AI Service: IBM Granite 3.3-2B accessed using HuggingFace transformers.
- Helper Logic: model_utils.py handles tokenization and AI responses.

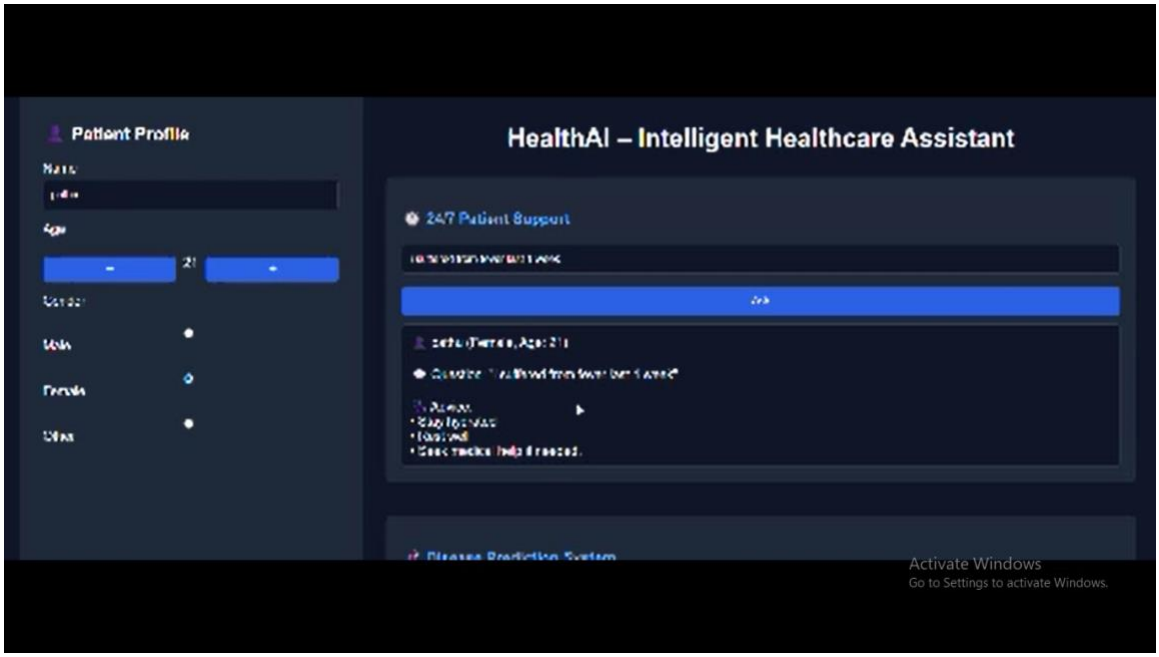
TESTING

Performance & Functional Testing

- Unit Testing: Tokenizer and model response test.
- Integration Testing: API endpoints (/predict, /remedy) with real inputs.
- Manual Testing: UI form interactions, output rendering.
- Error Handling: Basic validation for empty inputs; fallback messaging for invalid queries.

RESULTS





ADVANTAGES & DISADVANTAGES

Advantages:

- Lightweight, fast FastAPI app
- Uses a state-of-the-art IBM Granite model
- Natural language interaction
- User-friendly frontend

Disadvantages:

- No persistent storage or user accounts
- General AI model not trained on specific clinical datasets
- Limited to single-turn interactions (no memory)

FUTURE SCOPE

- Add user authentication (login/signup)
- Build database for saving user sessions or history
- Add multi-turn chatbot memory
- Integrate medical dataset fine-tuning
- Deploy with Docker for portability

APPENDIX

GitHub Link: [Update with correct URL if needed]
Source Code Files:

- main.py – FastAPI application
- model_utils.py – Model logic
- templates/index.html – HTML form
- static/style.css – Styling