Health AI Assistant – Project Documentation

This document outlines the Health AI Assistant project, detailing its purpose, features, architecture, and future enhancements. It serves as a comprehensive guide for understanding and deploying the AI assistant.

1. Introduction

Project Title:

Health AI Assistant

Team Members:

- Mukilavani
- Nandhini
- Nishitha
- Nivedha

2. Project Overview

Purpose:

The purpose of the Health AI Assistant is to provide users with an accessible tool for symptom analysis and treatment suggestions. By leveraging AI-driven natural language processing models, the assistant helps generate possible conditions and treatment guidelines, while strongly emphasizing the importance of consulting healthcare professionals for accurate diagnosis.

Features:

Disease Prediction

Key Point: AI-generated condition insights

Functionality: Suggests possible medical conditions based on symptoms.

Treatment Plan Generation

Key Point: Personalized recommendations

Functionality: Creates treatment plans based on condition, age, gender, and history.

Conversational AI

Key Point: Natural interaction

Functionality: Users interact in plain text with AI.

Gradio Web UI

Key Point: User-friendly interface

Functionality: Provides tab-based interaction for symptoms and treatment.

Disclaimer Handling

Key Point: Safety emphasis

Functionality: Displays warnings to avoid misuse as medical advice.

3. Architecture

1

Frontend (Gradio)

Provides an intuitive tabbed interface with textboxes, buttons, and outputs.

2

Backend (PyTorch + Hugging Face)

Loads and serves IBM Granite 3.2 Instruct model for generating predictions and responses.

3

Model Integration

Uses AutoTokenizer and AutoModelForCausalLM with optimized GPU/CPU inference (FP16/FP32).

4. Setup Instructions

Prerequisites:

- Python 3.9+
- Packages: torch, transformers, gradio

Installation Process:

1. Clone repository / upload script.
2. Install dependencies (pip install -r requirements.txt).
3. Run the script:

```
python healthai.py
```

1. Access the Gradio app (local URL or share link).

5. Folder Structure & 6. Running the Application

Folder Structure:

```
healthai.py # Main application script
requirements.txt # Dependencies (to be created if needed)
```

Running the Application:

- Launch the script: `python healthai.py`
- Open the Gradio interface in the browser.
- Use Disease Prediction tab for symptom analysis.
- Use Treatment Plans tab for patient-specific treatment suggestions.

7. API Documentation & 8. Authentication

API Documentation:

(Currently no REST APIs exposed; app runs entirely on Gradio UI.)

Authentication:

- **Demo version:** Open use with no authentication.
- **For production:** could integrate token-based authentication or user sessions.

9. User Interface

Two Tabs:

- Disease Prediction
- Treatment Plans

Input Fields:

For text (symptoms, condition, history) and numeric values (age).

Disclaimer:

Disclaimer text shown at top of interface.

10. Testing

Unit Testing

Prompt-response checked for validity.

Manual Testing

Verified with various symptom and condition inputs.

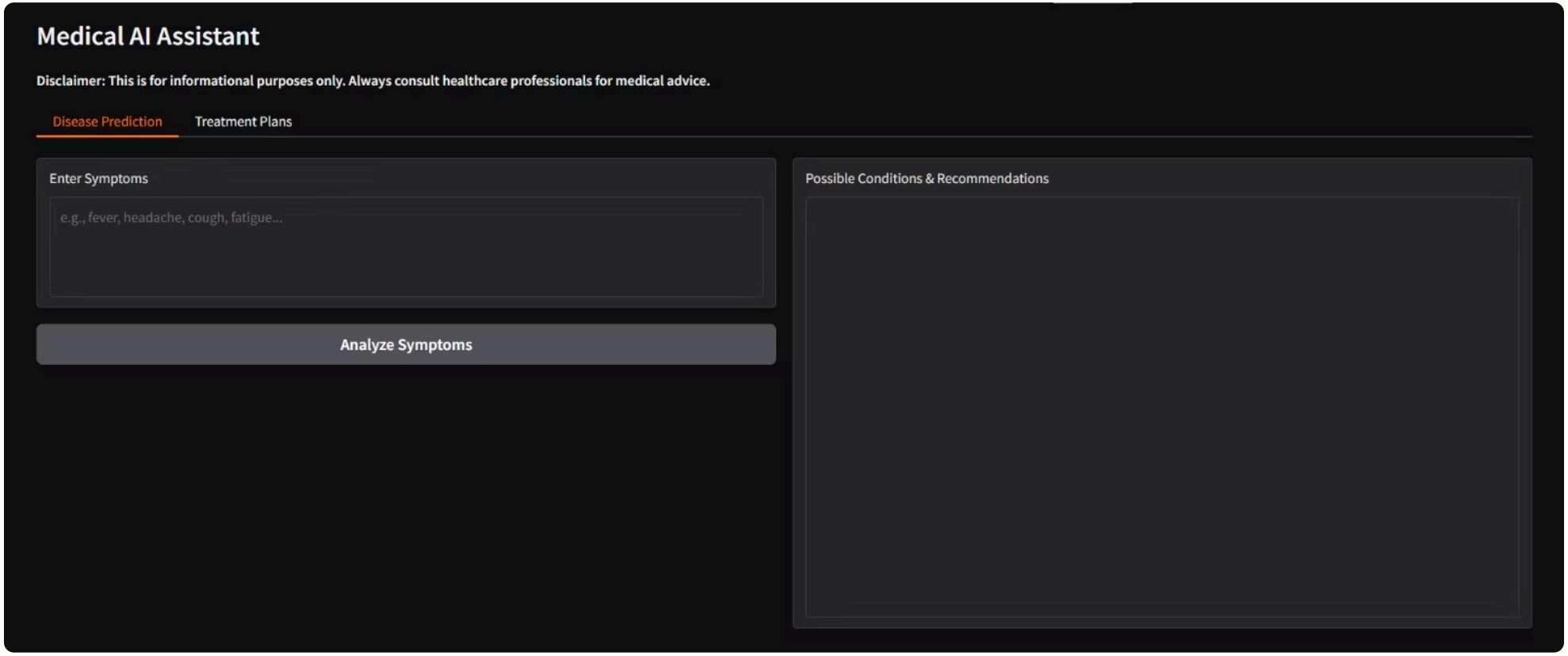
Performance Testing

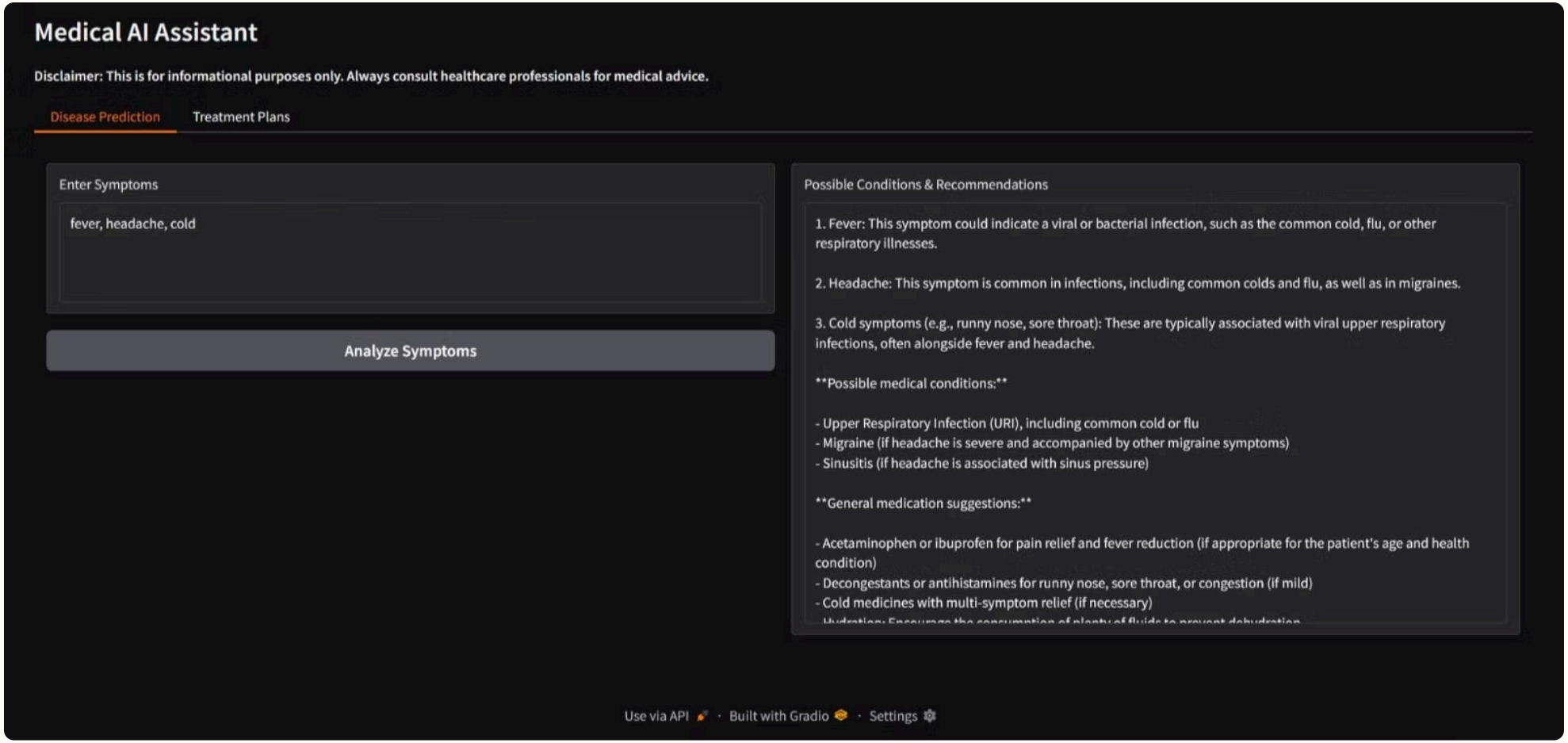
- GPU: 2–4 seconds response
- CPU: 30–45 seconds response

Edge Cases

Empty input, invalid age accepted (needs fix).

Screenshot:





12. Known Issues & 13. Future Enhancements

Known Issues:

- No input validation (negative age, empty symptoms allowed).
- May generate unsafe advice (e.g., medication dosage).
- Slow performance on CPU.
- No error handling for out-of-memory failures.
- UI is plain (difficult to read long text).

Future Enhancements:

- Add safety filters for outputs.
- Include streaming responses for better UX.
- Improve UI formatting with Markdown/sections.
- Add authentication for secure usage.
- Deploy as Hugging Face Space or Dockerized app.