# Project Documentation

1. **Introduction**

**Project Title:** Medical AI Assistant

## Team Members:

* + Team member Krishna.s
  + Team member Daniel Christopher.j
  + Team member king krister.s
  + Team member akash Christopher g

# Project Overview

## Purpose:

The purpose of the Medical AI Assistant is to provide users with preliminary, informational insights based on their symptoms. It leverages AI models to predict possible diseases and suggest treatment plans, helping users make informed decisions while reminding them to consult healthcare professionals.

## Features:

* + **Symptom-based Disease Prediction:** Takes input symptoms and predicts possible conditions.
  + **Treatment Plan Suggestions:** Provides general recommendations and care tips.
  + **Conversational Interface:** Easy-to-use text-based interface for interaction.
  + **AI-powered Predictions:** Uses ML models for analyzing symptoms and conditions.
  + **User-friendly UI:** Developed with Gradio for accessibility.

# Architecture

**Frontend (Gradio):** Provides an interactive UI for symptom entry and result visualization. **Backend (FastAPI/Flask):** Processes user input, runs AI models, and returns predictions. **ML Model:** Trained on medical datasets to provide accurate predictions.

**Database (Optional):** Stores user interactions and symptom-condition mappings.

# Setup Instructions

* + Python 3.9 or later
  + Install dependencies from requirements.txt
  + Run backend server (FastAPI/Flask)
  + Launch frontend using Gradio
  + Enter symptoms and view predictions

# Folder Structure

* + app/ – Contains backend logic and APIs
  + ui/ – Gradio-based frontend components
  + models/ – ML models for prediction
  + utils/ – Helper scripts and preprocessing functions
  + main.py – Entry script to launch application

# Running the Application

To start the project: run the backend API server, launch the Gradio frontend, and interact with the Medical AI Assistant by entering symptoms. The system will display possible conditions and

recommendations.

# API Documentation

* + POST /analyze-symptoms – Accepts symptoms and returns possible conditions.
  + GET /treatment-plan – Provides general care recommendations.
  + POST /feedback – Collects user feedback for improvements.

# Authentication

This demo project runs in an open environment. Secure deployments can include token-based authentication, OAuth2, and role-based access control.

# User Interface

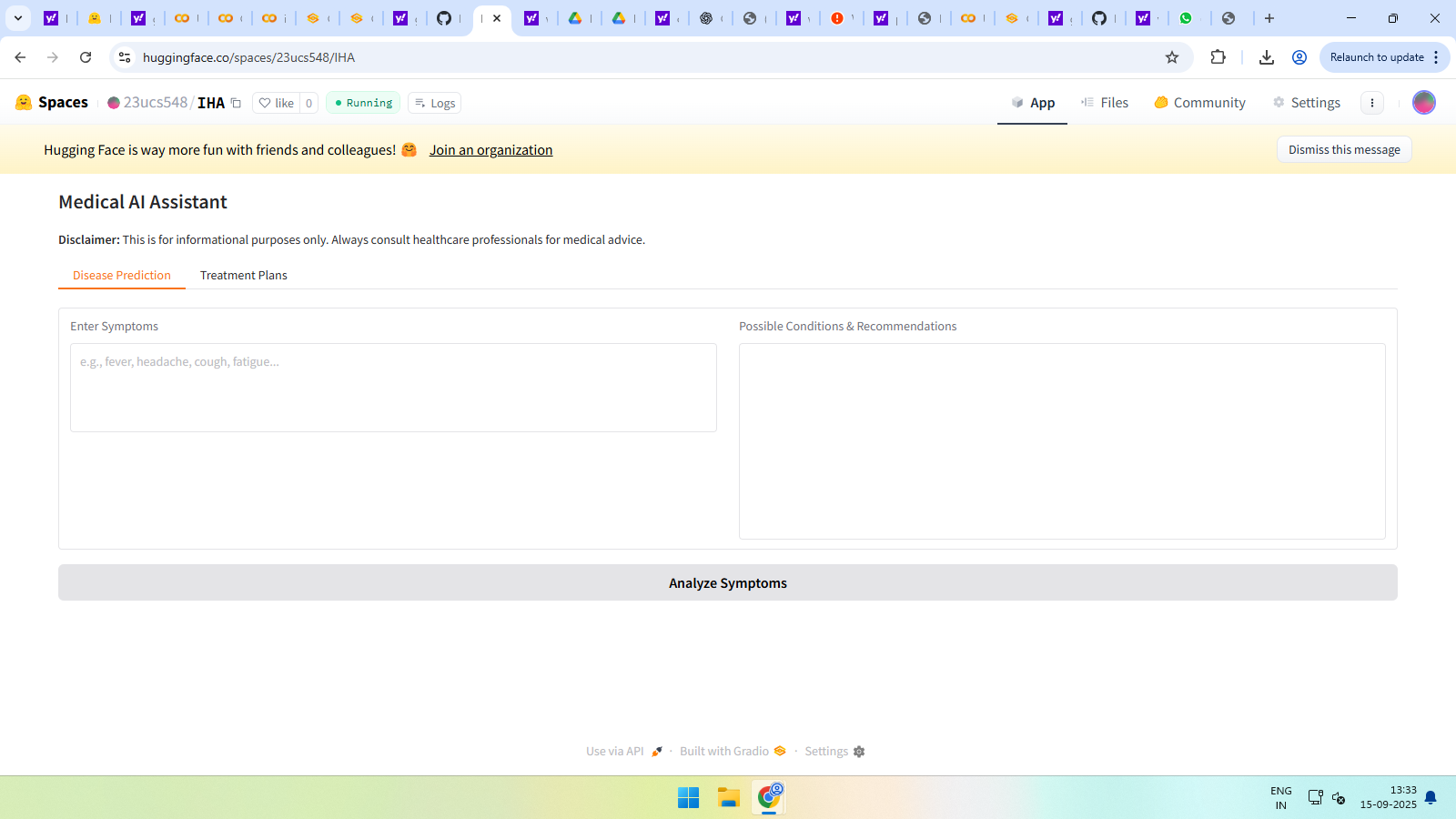
The interface is simple and user-friendly, with symptom input boxes, prediction outputs, and treatment plan tabs. It is built using Gradio for smooth accessibility.

# Testing

Testing includes unit tests for AI model predictions, API testing using Postman, and manual testing for symptom input/output consistency.

# Screenshots

Screenshot of Medical AI Assistant interface (refer to attached image).



# Known Issues

Predictions are not 100% accurate and should not replace medical consultation. Limited dataset coverage may affect results.

# Future Enhancements

* + Expand training dataset with more medical conditions.
  + Integrate with electronic health records (EHR).
  + Enable voice-based input for accessibility.
  + Add real-time chatbot support with healthcare professionals.
  + Mobile app version for broader access.