TABLE OF CONTENTS

CODES

Backend Integration :-

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import pandas as pd
import os
import ast
import difflib
import requests
app = Flask(__name__)
CORS(app)
print("📁 Working directory:", os.getcwd())
# Load datasets
try:
    def clean_df(df, rename=None):
        df.columns = df.columns.str.strip()
        if rename:
            df.rename(columns=rename, inplace=True)
        return df
    symptom_df = pd.read_csv("dataset/symtoms_df.csv")
    symptom_df = clean_df(symptom_df)
    symptom_df['all_symptoms'] = symptom_df[['Symptom_1', 'Symptom_2', 'Symptom_3', 'Symptom_4']] \
                                 .fillna('').agg(' '.join, axis=1).str.lower()
    medications_df = pd.read_csv("dataset/medications.csv")
    medications_df = clean_df(medications_df, rename={"Medication": "Medicine"})
    description_df = pd.read_csv("dataset/description.csv")
    description_df = clean_df(description_df)
```

```
    precautions_df = pd.read_csv("dataset/precautions_df.csv")
    precautions_df = clean_df(precautions_df)
    diets_df = pd.read_csv("dataset/diets.csv")
    diets_df = clean_df(diets_df)
    workout_df = pd.read_csv("dataset/workout_df.csv")
    workout_df = clean_df(workout_df, rename={"disease": "Disease", "workout": "Workout"})
    all_symptom_values = pd.unique(symptom_df[['Symptom_1', 'Symptom_2', 'Symptom_3', 'Symptom_4']].values.ravel())
    known_symptoms = [s.strip().lower() for s in all_symptom_values if isinstance(s, str)]
    print("✅ All datasets loaded successfully.")
except Exception as e:
    print("❌ Error loading datasets:", e)
    raise
# Doctor mapping
doctor_map = {
    "diabetes": "Endocrinologist",
    "migraine": "Neurologist",
    "asthma": "Pulmonologist",
    "gerd": "Gastroenterologist",
    "hypertension": "Cardiologist",
    "depression": "Psychiatrist",
    "arthritis": "Rheumatologist",
    "eczema": "Dermatologist",
    "dengue": "General Physician",
    "covid": "Infectious Disease Specialist",
    "flu": "General Physician",
    "anxiety": "Psychiatrist",
```

```
    "back pain": "Orthopedic",
}
def match_similar_symptoms(user_inputs, known_symptoms):
    matched = []
    for symptom in user_inputs:
        close_matches = difflib.get_close_matches(symptom, known_symptoms, n=1,
cutoff=0.6)
        if close_matches:
            matched.append(close_matches[0])
    return matched
@app.route("/api/ml_predict", methods=["POST"])
def predict_disease():
    try:
        data = request.get_json()
        user_symptoms = data.get("symptoms", "").strip().lower()

        if not user_symptoms:
            return jsonify({"error": "No symptoms provided"}), 400
        raw_symptom_list = [s.strip() for s in user_symptoms.split(',') if s.strip()]
        matched_symptom_list = match_similar_symptoms(raw_symptom_list,
known_symptoms)
        print("🧠 Matched symptoms:", matched_symptom_list)
        matched = symptom_df[symptom_df['all_symptoms'].apply(
            lambda s: any(symptom in s for symptom in matched_symptom_list)
        )]
        if matched.empty:
            return jsonify({"error": "No disease matched for given symptoms."}), 404
        def get_confidence(row):
```

```
            symptoms = [row['Symptom_1'], row['Symptom_2'], row['Symptom_3'],
row['Symptom_4']]
            symptoms = [s.lower().strip() for s in symptoms if isinstance(s, str)]
            matched_count = sum(1 for s in matched_symptom_list if s in symptoms)
            return matched_count / len(symptoms) if symptoms else 0
        matched = matched.copy()
        matched['confidence'] = matched.apply(get_confidence, axis=1)
        top_match = matched.sort_values(by='confidence', ascending=False).iloc[0]
        disease = top_match['Disease']
        confidence_percent = round(top_match['confidence'] * 100, 2)


        return jsonify({
            "predicted_disease": disease,
            "confidence": f"{confidence_percent}%",
            "matched_symptoms": matched_symptom_list
        })
    except Exception as e:
        print("❌ Error in /api/ml_predict:", e)
        return jsonify({"error": "Internal server error"}), 500
@app.route("/api/details", methods=["GET"])
def get_disease_details():
    try:
        disease = request.args.get("disease", "").strip().lower()
        if not disease:
            return jsonify({"error": "No disease provided"}), 400
        description = description_df[description_df['Disease'].str.lower() ==
disease]['Description'].values
        description = description[0] if len(description) > 0 else "No description
available."
```

```
        medicine = medications_df[medications_df['Disease'].str.lower() ==
disease]['Medicine'].values
        medicine = ast.literal_eval(medicine[0]) if len(medicine) > 0 and
medicine[0].startswith("[") else \
                [medicine[0]] if len(medicine) > 0 else ["No medicine found."]
        precautions = precautions_df[precautions_df['Disease'].str.lower() == disease] \
                .drop(columns=['Disease'], errors='ignore').values.flatten().tolist()
        precautions = [p for p in precautions if isinstance(p, str) and p.strip()]
        diet = diets_df[diets_df['Disease'].str.lower() == disease]['Diet'].values
        diet = ast.literal_eval(diet[0]) if len(diet) > 0 and diet[0].startswith("[") else \
            [diet[0]] if len(diet) > 0 else ["No diet found."]
        workout = workout_df[workout_df['Disease'].str.lower() ==
disease]['Workout'].values
        workout = workout[0] if len(workout) > 0 else "No workout recommendation."
        specialist = doctor_map.get(disease, "General Physician")
        return jsonify({
            "description": description,
            "medicine": medicine,
            "precautions": precautions,
            "diet": diet,
            "workout": workout,
            "specialist": specialist
        })
    except Exception as e:
        print("❌ Error in /api/details:", e)
        return jsonify({"error": "Failed to retrieve details."}), 500
@app.route("/api/suggestions", methods=["GET"])
def suggest_symptoms():
    try:
```

```
        query = request.args.get("q", "").strip().lower()
        if not query:
            return jsonify([])
        matches = [s for s in known_symptoms if query in s]
        return jsonify(matches[:10])
    except Exception as e:
        print("❌ Suggestion error:", e)
        return jsonify([]), 500

# ✅ Ollama chat integration
@app.route("/api/chat", methods=["POST"])
def chat_with_ollama():
    try:
        data = request.get_json()
        user_message = data.get("message", "").strip()
        if not user_message:
            return jsonify({"error": "Empty message"}), 400
        print("💬 User Message:", user_message)
        ollama_res = requests.post(
            "http://localhost:11434/api/generate",
            json={
                "model": "llama3",
                "prompt": user_message,
                "stream": False
            }
        )
        if ollama_res.status_code != 200:
            print("❌ Ollama Error:", ollama_res.text)
            return jsonify({"error": "Ollama failed"}), 500
```

```
        reply = ollama_res.json().get("response", "").strip()
        print("🤖 LLaMA Response:", reply)
        return jsonify({"response": reply})
    except Exception as e:
        print("❌ Chat error:", e)
        return jsonify({"error": "Internal server error"}), 500
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```