AI-Powered Virtual Health Assistant

CODING

main.py

```
from flask import Flask, request, render_template, jsonify  # Import jsonify
import numpy as np
import pandas as pd
import pickle
# flask app initialization================================
app = Flask(__name__)
# load databasedataset===================================
sym_des = pd.read_csv(r"C:\Users\Latha\OneDrive\Desktop\AI\symtoms_df.csv")
precautions = pd.read_csv(r"C:\Users\Latha\OneDrive\Desktop\AI\precautions_df.csv")
workout = pd.read_csv(r"C:\Users\Latha\OneDrive\Desktop\AI\workout_df.csv")
description = pd.read_csv(r"C:\Users\Latha\OneDrive\Desktop\AI\description.csv")
medications = pd.read_csv(r"C:\Users\Latha\OneDrive\Desktop\AI\medications.csv")
diets = pd.read_csv(r"C:\Users\Latha\OneDrive\Desktop\AI\diets.csv")
# load model=============================================
svc = pickle.load(open(r"C:\Users\Latha\OneDrive\Desktop\AI\svc.pkl",'rb'))
# custome and helping functions==========================
def helper(dis):
    desc = description[description['Disease'] == dis]['Description']
    desc = " ".join([w for w in desc])
    pre = precautions[precautions['Disease'] == dis][['Precaution_1', 'Precaution_2', 'Precaution_3',
'Precaution_4']]
    pre = [col for col in pre.values]
    med = medications[medications['Disease'] == dis]['Medication']
```

```
    med = [med for med in med.values]
    die = diets[diets['Disease'] == dis]['Diet']
    die = [die for die in die.values]
    wrkout = workout[workout['disease'] == dis] ['workout']
    return desc,pre,med,die,wrkout
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3,
'shivering': 4, 'chills': 5, 'joint_pain': 6, 'stomach_pain': 7, 'acidity': 8, 'ulcers_on_tongue': 9,
'muscle_wasting': 10, 'vomiting': 11, 'burning_micturition': 12, 'spotting_ urination': 13, 'fatigue':
14, 'weight_gain': 15, 'anxiety': 16, 'cold_hands_and_feets': 17, 'mood_swings': 18, 'weight_loss':
19, 'restlessness': 20, 'lethargy': 21, 'patches_in_throat': 22, 'irregular_sugar_level': 23, 'cough':
24, 'high_fever': 25, 'sunken_eyes': 26, 'breathlessness': 27, 'sweating': 28, 'dehydration': 29,
'indigestion': 30, 'headache': 31, 'yellowish_skin': 32, 'dark_urine': 33, 'nausea': 34,
'loss_of_appetite': 35, 'pain_behind_the_eyes': 36, 'back_pain': 37, 'constipation': 38,
'abdominal_pain': 39, 'diarrhoea': 40, 'mild_fever': 41, 'yellow_urine': 42, 'yellowing_of_eyes':
43, 'acute_liver_failure': 44, 'fluid_overload': 45, 'swelling_of_stomach': 46,
'swelled_lymph_nodes': 47, 'malaise': 48, 'blurred_and_distorted_vision': 49, 'phlegm': 50,
'throat_irritation': 51, 'redness_of_eyes': 52, 'sinus_pressure': 53, 'runny_nose': 54, 'congestion':
55, 'chest_pain': 56, 'weakness_in_limbs': 57, 'fast_heart_rate': 58,
'pain_during_bowel_movements': 59, 'pain_in_anal_region': 60, 'bloody_stool': 61,
'irritation_in_anus': 62, 'neck_pain': 63, 'dizziness': 64, 'cramps': 65, 'bruising': 66, 'obesity': 67,
'swollen_legs': 68, 'swollen_blood_vessels': 69, 'puffy_face_and_eyes': 70, 'enlarged_thyroid':
71, 'brittle_nails': 72, 'swollen_extremeties': 73, 'excessive_hunger': 74, 'extra_marital_contacts':
75, 'drying_and_tingling_lips': 76, 'slurred_speech': 77, 'knee_pain': 78, 'hip_joint_pain': 79,
'muscle_weakness': 80, 'stiff_neck': 81, 'swelling_joints': 82, 'movement_stiffness': 83,
'spinning_movements': 84, 'loss_of_balance': 85, 'unsteadiness': 86,
'weakness_of_one_body_side': 87, 'loss_of_smell': 88, 'bladder_discomfort': 89, 'foul_smell_of
urine': 90, 'continuous_feel_of_urine': 91, 'passage_of_gases': 92, 'internal_itching': 93,
'toxic_look_(typhos)': 94, 'depression': 95, 'irritability': 96, 'muscle_pain': 97,
'altered_sensorium': 98, 'red_spots_over_body': 99, 'belly_pain': 100, 'abnormal_menstruation':
101, 'dischromic _patches': 102, 'watering_from_eyes': 103, 'increased_appetite': 104, 'polyuria':
105, 'family_history': 106, 'mucoid_sputum': 107, 'rusty_sputum': 108, 'lack_of_concentration':
109, 'visual_disturbances': 110, 'receiving_blood_transfusion': 111,
'receiving_unsterile_injections': 112, 'coma': 113, 'stomach_bleeding': 114,
'distention_of_abdomen': 115, 'history_of_alcohol_consumption': 116, 'fluid_overload.1': 117,
'blood_in_sputum': 118, 'prominent_veins_on_calf': 119, 'palpitations': 120, 'painful_walking':
121, 'pus_filled_pimples': 122, 'blackheads': 123, 'scurring': 124, 'skin_peeling': 125,
'silver_like_dusting': 126, 'small_dents_in_nails': 127, 'inflammatory_nails': 128, 'blister': 129,
'red_sore_around_nose': 130, 'yellow_crust_ooze': 131}
```

```
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic ulcer diseae', 1: 'AIDS', 12: 'Diabetes ', 17: 'Gastroenteritis', 6: 'Bronchial Asthma', 23: 'Hypertension ', 30: 'Migraine', 7: 'Cervical spondylosis', 32: 'Paralysis (brain hemorrhage)', 28: 'Jaundice', 29: 'Malaria', 8: 'Chicken pox', 11: 'Dengue', 37: 'Typhoid', 40: 'hepatitis A', 19: 'Hepatitis B', 20: 'Hepatitis C', 21: 'Hepatitis D', 22: 'Hepatitis E', 3: 'Alcoholic hepatitis', 36: 'Tuberculosis', 10: 'Common Cold', 34: Pneumonia', 13: 'Dimorphic hemmorhoids(piles)', 18: 'Heart attack', 39: 'Varicose veins', 26: 'Hypothyroidism', 24: 'Hyperthyroidism', 25: 'Hypoglycemia', 31: 'Osteoarthristis', 5: 'Arthritis', 0: '(vertigo) Paroymsal  Positional Vertigo', 2: 'Acne', 38: 'Urinary tract infection', 35: 'Psoriasis', 27: 'Impetigo'}

def get_top_predictions(patient_symptoms, severity_scores=None, top_n=3):
    input_vector = np.zeros(len(symptoms_dict))
    for symptom in patient_symptoms:
        if symptom in symptoms_dict:
            idx = symptoms_dict[symptom]
            input_vector[idx] = severity_scores.get(symptom, 1.0) if severity_scores else 1.0
    try:
        proba = svc.predict_proba([input_vector])[0]
        top_indices = np.argsort(proba)[::-1][:top_n]
        return [(diseases_list[i], round(proba[i]*100, 2)) for i in top_indices]
    except AttributeError:
        # If the model doesn't support predict_proba
        predicted_index = svc.predict([input_vector])[0]
        return [(diseases_list[predicted_index], None)]
# creating routes========================================
@app.route("/")
def index():
    return render_template("index.html")
@app.route('/predict', methods=['GET', 'POST'])
def home():
```

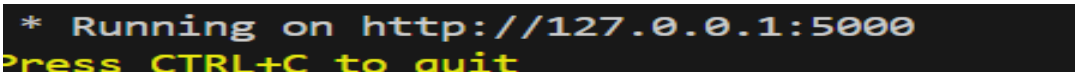
```
    if request.method == 'POST':
        symptoms = request.form.get('symptoms')
        if symptoms == "Symptoms" or not symptoms.strip():
            message = "Please either write symptoms or you have written misspelled symptoms."
            return render_template('index.html', message=message,
symptoms=list(symptoms_dict.keys()))
        # Clean and split user symptoms
        user_symptoms = [s.strip().lower().replace('-', '_').replace(' ', '_') for s in symptoms.split(',')]
        user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
        # 🛑 Check for invalid symptoms
        invalid_symptoms = [symptom for symptom in user_symptoms if symptom not in
symptoms_dict]
        if invalid_symptoms:
            message = f"These symptoms were not recognized: {', '.join(invalid_symptoms)}"
            return render_template('index.html', message=message,
symptoms=list(symptoms_dict.keys()))
        # ✅ Predict disease
        # In real app, you'd extract severity info from form
        # For now, assume all symptoms are 'severe' => score 1.0
        severity_scores = {sym: 1.0 for sym in user_symptoms}
        top_predictions = get_top_predictions(user_symptoms, severity_scores)
        predicted_disease = top_predictions[0][0]  # use top one for detailed info
        dis_des, precautions, medications, rec_diet, workout = helper(predicted_disease)
        my_precautions = []
        for i in precautions[0]:
            my_precautions.append(i)
        rec_data_count = {
            'Precautions': len(my_precautions),
```

```
            'Medications': len(medications),
            'Diet': len(rec_diet),
            'Workout': len(workout)
        }
        return render_template(
    'index.html',
    predicted_disease=predicted_disease,
    dis_des=dis_des,
    my_precautions=my_precautions,
    medications=medications,
    my_diet=rec_diet,
    workout=workout,
    symptoms=list(symptoms_dict.keys()),  # for autocomplete
    rec_data_count=rec_data_count,
    top_predictions=top_predictions  # 🔥 add this!
)
    return render_template('index.html', symptoms=list(symptoms_dict.keys()))
# about view funtion and path
@app.route('/about')
def about():
    return render_template("about.html")
# contact view funtion and path
@app.route('/contact')
def contact():
    return render_template("contact.html")
# developer view funtion and path
@app.route('/developer')
```

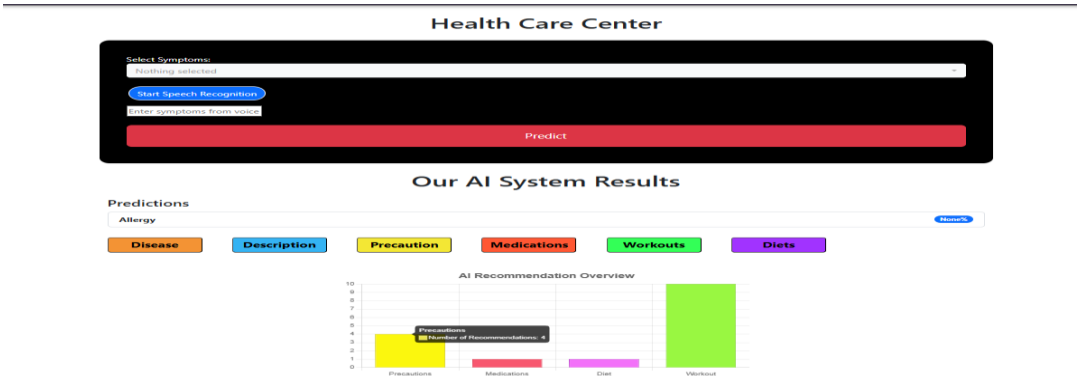
```
def developer():
    return render_template("developer.html")
# about view funtion and path
@app.route('/blog')
def blog():
    return render_template("blog.html")
@app.route('/get_symptoms', methods=['GET'])
def get_symptoms():
    return jsonify(list(symptoms_dict.keys()))
if __name__ == '__main__':
        app.run(debug=True)
```

OUTPUT

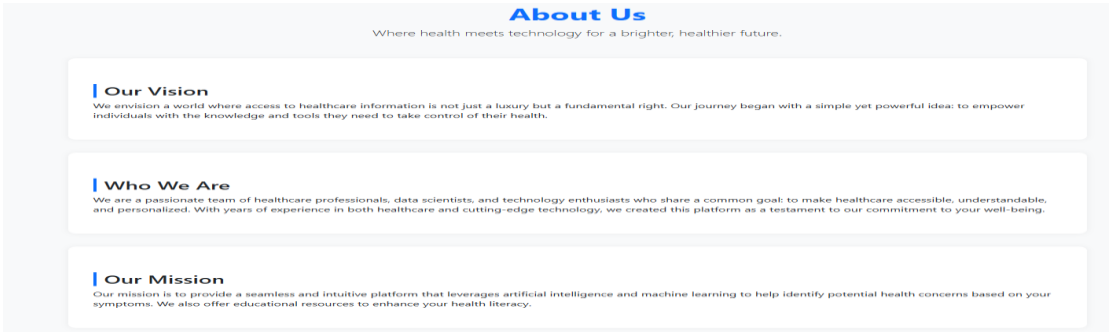
1. Link to Web Page



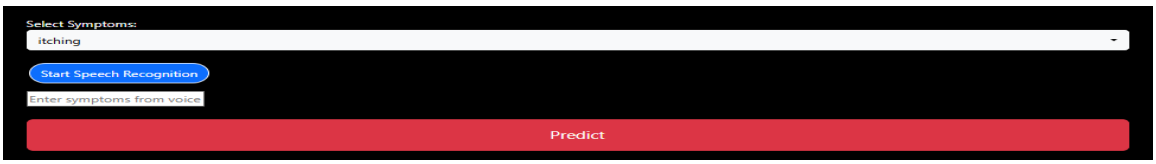
2. Home Page



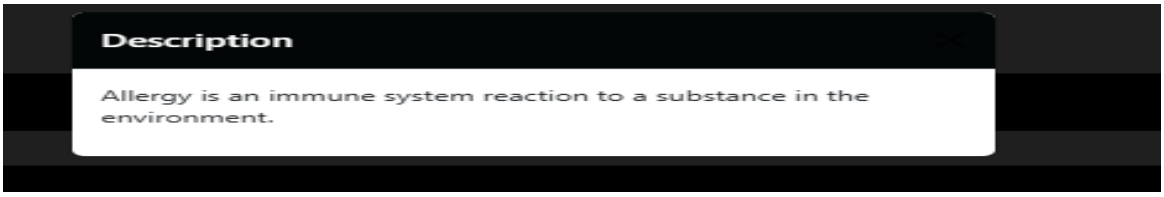
3. About Us



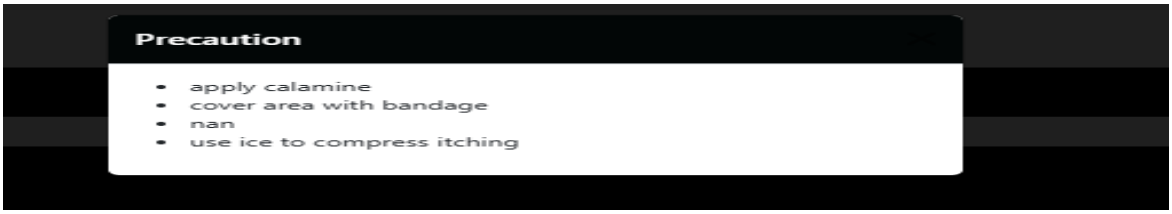
4. Symptoms Input



5. Description



6. Precaution



7. Medication

8. Diets

9. Workouts