**Stress Management Bot**



**Submitted by** Prince Kumar 12309518

In partial fulfilment for the requirements of the award of the degree of

**B. Tech CSE**

School of Computer Science **Lovely Professional University Phagwara, Punjab**

CODE USED:

MAIN CODE:

from flask import Flask, render\_template, request, jsonify

from chatbot import StressBot

import os

from dotenv import load\_dotenv

# Load environment variables

load\_dotenv()

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = os.getenv('SECRET\_KEY', 'your-secret-key-here')

# Initialize chatbot

chatbot = StressBot()

@app.route('/')

def index():

    return render\_template('chat.html')

@app.route('/chat', methods=['POST'])

def chat():

    user\_message = request.json.get('message')

    response = chatbot.process\_message(user\_message)

    return jsonify({'response': response})

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

CODE FOR CHATBOT:

import os

import google.generativeai as genai

from datetime import datetime

import json

import re

from dotenv import load\_dotenv

# Load environment variables

load\_dotenv()

# Configure Gemini API

GEMINI\_API\_KEY = os.getenv("GEMINI\_API\_KEY")

genai.configure(api\_key=GEMINI\_API\_KEY)

SECRET\_KEY = os.getenv("SECRET\_KEY")

class StressBot:

    def \_\_init\_\_(self):

        # Initialize the Gemini model

        self.model = genai.GenerativeModel('gemini-1.5-pro')

        self.user\_profile = {}

        self.conversation\_history = []

        self.stress\_info = {

            "stress\_level": None,

            "stress\_triggers": [],

            "coping\_mechanisms": [],

            "sleep\_quality": None,

            "physical\_symptoms": [],

            "emotional\_symptoms": []

        }

    def save\_profile(self, filename="user\_profile.json"):

        """Save the user profile to a file"""

        with open(filename, 'w') as f:

            json.dump({"profile": self.user\_profile, "stress\_info": self.stress\_info}, f)

        return "User profile saved successfully!"

    def load\_profile(self, filename="user\_profile.json"):

        """Load the user profile from a file"""

        try:

            with open(filename, 'r') as f:

                data = json.load(f)

                self.user\_profile = data.get("profile", {})

                self.stress\_info = data.get("stress\_info", {})

            return "User profile loaded successfully!"

        except FileNotFoundError:

            return "No profile found. Let's create a new one!"

    def update\_profile(self, name=None, age=None, occupation=None):

        """Update the user profile with basic information"""

        if name:

            self.user\_profile["name"] = name

        if age:

            self.user\_profile["age"] = age

        if occupation:

            self.user\_profile["occupation"] = occupation

        return "Profile updated successfully!"

    def set\_stress\_info(self, stress\_level=None, triggers=None, coping\_mechanisms=None,

                       sleep\_quality=None, physical\_symptoms=None, emotional\_symptoms=None):

        """Update the stress information"""

        if stress\_level:

            self.stress\_info["stress\_level"] = stress\_level

        if triggers:

            self.stress\_info["stress\_triggers"] = triggers if isinstance(triggers, list) else [triggers]

        if coping\_mechanisms:

            self.stress\_info["coping\_mechanisms"] = coping\_mechanisms if isinstance(coping\_mechanisms, list) else [coping\_mechanisms]

        if sleep\_quality:

            self.stress\_info["sleep\_quality"] = sleep\_quality

        if physical\_symptoms:

            self.stress\_info["physical\_symptoms"] = physical\_symptoms if isinstance(physical\_symptoms, list) else [physical\_symptoms]

        if emotional\_symptoms:

            self.stress\_info["emotional\_symptoms"] = emotional\_symptoms if isinstance(emotional\_symptoms, list) else [emotional\_symptoms]

        return "Stress information updated successfully!"

    def assess\_stress\_level(self):

        """Assess the user's stress level based on current information"""

        prompt = f"""

        Assess the stress level of a person with the following information:

        Stress triggers: {', '.join(self.stress\_info['stress\_triggers']) if self.stress\_info['stress\_triggers'] else 'None specified'}

        Physical symptoms: {', '.join(self.stress\_info['physical\_symptoms']) if self.stress\_info['physical\_symptoms'] else 'None specified'}

        Emotional symptoms: {', '.join(self.stress\_info['emotional\_symptoms']) if self.stress\_info['emotional\_symptoms'] else 'None specified'}

        Sleep quality: {self.stress\_info['sleep\_quality'] or 'Not specified'}

        Provide:

        1. An assessment of their current stress level (low, moderate, high, or severe)

        2. Potential health risks associated with their current stress level

        3. Recommendations for stress reduction

        4. Signs that indicate they should seek professional help

        """

        response = self.model.generate\_content(prompt)

        return response.text

    def suggest\_coping\_strategies(self):

        """Suggest coping strategies based on the user's situation"""

        prompt = f"""

        Suggest personalized stress management strategies based on:

        Stress level: {self.stress\_info['stress\_level'] or 'Not specified'}

        Stress triggers: {', '.join(self.stress\_info['stress\_triggers']) if self.stress\_info['stress\_triggers'] else 'None specified'}

        Current coping mechanisms: {', '.join(self.stress\_info['coping\_mechanisms']) if self.stress\_info['coping\_mechanisms'] else 'None specified'}

        Include:

        1. Immediate stress relief techniques

        2. Long-term stress management strategies

        3. Mindfulness and meditation exercises

        4. Physical activity recommendations

        5. Sleep improvement tips

        6. Time management strategies

        7. Boundary-setting techniques

        """

        response = self.model.generate\_content(prompt)

        return response.text

    def provide\_relaxation\_exercises(self):

        """Provide relaxation exercises based on the user's profile"""

        prompt = f"""

        Provide relaxation exercises based on:

        Stress level: {self.stress\_info['stress\_level'] or 'Not specified'}

        Physical symptoms: {', '.join(self.stress\_info['physical\_symptoms']) if self.stress\_info['physical\_symptoms'] else 'None specified'}

        Emotional symptoms: {', '.join(self.stress\_info['emotional\_symptoms']) if self.stress\_info['emotional\_symptoms'] else 'None specified'}

        Include:

        1. Breathing exercises with step-by-step instructions

        2. Progressive muscle relaxation techniques

        3. Guided imagery exercises

        4. Mindfulness meditation scripts

        5. Body scan exercises

        6. Quick relaxation techniques for busy situations

        """

        response = self.model.generate\_content(prompt)

        return response.text

    def suggest\_lifestyle\_changes(self):

        """Provide lifestyle change recommendations"""

        prompt = f"""

        Suggest lifestyle changes to reduce stress based on:

        Stress level: {self.stress\_info['stress\_level'] or 'Not specified'}

        Sleep quality: {self.stress\_info['sleep\_quality'] or 'Not specified'}

        Stress triggers: {', '.join(self.stress\_info['stress\_triggers']) if self.stress\_info['stress\_triggers'] else 'None specified'}

        Include:

        1. Sleep hygiene recommendations

        2. Dietary changes to support stress reduction

        3. Exercise recommendations

        4. Social connection strategies

        5. Work-life balance techniques

        6. Time management tips

        7. Digital wellness practices

        """

        response = self.model.generate\_content(prompt)

        return response.text

    def process\_message(self, user\_message):

        """Process user messages and generate appropriate responses"""

        # Add message to conversation history

        self.conversation\_history.append({"role": "user", "content": user\_message})

        # Extract information from user message

        if "stress level" in user\_message.lower():

            level\_match = re.search(r"(low|moderate|high|severe)", user\_message.lower())

            if level\_match:

                self.stress\_info["stress\_level"] = level\_match.group(1)

        # Generate response based on message content

        if "stress level" in user\_message.lower() or "assess" in user\_message.lower():

            response = self.assess\_stress\_level()

        elif "coping" in user\_message.lower() or "strategies" in user\_message.lower():

            response = self.suggest\_coping\_strategies()

        elif "relax" in user\_message.lower() or "exercise" in user\_message.lower():

            response = self.provide\_relaxation\_exercises()

        elif "lifestyle" in user\_message.lower() or "change" in user\_message.lower():

            response = self.suggest\_lifestyle\_changes()

        else:

            # Default response for general queries

            prompt = f"""

            You are a helpful stress management assistant. The user has provided the following information:

            Stress level: {self.stress\_info['stress\_level'] or 'Not specified'}

            Stress triggers: {', '.join(self.stress\_info['stress\_triggers']) if self.stress\_info['stress\_triggers'] else 'None specified'}

            Physical symptoms: {', '.join(self.stress\_info['physical\_symptoms']) if self.stress\_info['physical\_symptoms'] else 'None specified'}

            Emotional symptoms: {', '.join(self.stress\_info['emotional\_symptoms']) if self.stress\_info['emotional\_symptoms'] else 'None specified'}

            Please provide a helpful response to: {user\_message}

            Focus on:

            1. Stress assessment

            2. Coping strategies

            3. Relaxation techniques

            4. Lifestyle changes

            5. When to seek professional help

            """

            response = self.model.generate\_content(prompt)

            response = response.text

        # Add response to conversation history

        self.conversation\_history.append({"role": "assistant", "content": response})

        return response

def main():

    bot = StressBot()

    print("Stress Management Assistant initialized. Type 'quit' to exit.")

    while True:

        user\_input = input("You: ")

        if user\_input.lower() == 'quit':

            break

        response = bot.process\_message(user\_input)

        print("Assistant:", response)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

CODE TO GENERATE API:

import secrets

# Generate a secure random string

secret\_key = secrets.token\_hex(32)

print(f"Generated Secret Key: {secret\_key}")