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

from flask\_sqlalchemy import SQLAlchemy

from textblob import TextBlob

import google.generativeai as genai

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

app.secret\_key = ''

genai.configure(api\_key="AIzaSyAiDJtsZ-hI9RdNpFbCvDNtG2DpJM6yJfY")

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///chatbot.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# ========== User Model ==========

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

email = db.Column(db.String(100), unique=True, nullable=False)

password = db.Column(db.String(100), nullable=False)

with app.app\_context():

db.create\_all()

@app.route('/')

def home():

return render\_template("chat.html")

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

def register():

email = request.form['email']

password = request.form['password']

if not User.query.filter\_by(email=email).first():

user = User(email=email, password=password)

db.session.add(user)

db.session.commit()

session['user'] = email

return redirect('/')

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

def login():

email = request.form['email']

password = request.form['password']

user = User.query.filter\_by(email=email, password=password).first()

if user:

session['user'] = email

return redirect('/')

@app.route('/logout')

def logout():

session.pop('user', None)

return redirect('/')

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

def analyze\_sentiment(text):

blob = TextBlob(text)

sentiment\_score = blob.sentiment.polarity

if sentiment\_score > 0:

sentiment\_label = "Positive"

elif sentiment\_score < 0:

sentiment\_label = "Negative"

else:

sentiment\_label = "Neutral"

return sentiment\_label, sentiment\_score

# @app.route("/chatbot", methods=["POST"])

# def chatbot():

# """Handles user messages and returns chatbot responses with sentiment analysis."""

# data = request.get\_json()

# user\_message = data.get("message", "")

# # Analyze User Sentiment

# sentiment\_label, sentiment\_score = analyze\_sentiment(user\_message)

# # Generate AI Response using Gemini

# model = genai.GenerativeModel("gemini-1.5-flash")

# response = model.generate\_content(user\_message)

# bot\_reply = response.text if response.text else "I'm here to listen. Tell me more."

# return jsonify({

# "reply": bot\_reply,

# "sentiment": sentiment\_label,

# "sentiment\_score": sentiment\_score

# })

@app.route("/chatbot", methods=["POST"])

def chatbot():

"""Handles user messages and returns chatbot responses with sentiment and mood analysis."""

data = request.get\_json()

user\_message = data.get("message", "")

sentiment\_label, sentiment\_score = analyze\_sentiment(user\_message)

lowered = user\_message.lower()

if "happy" in lowered or sentiment\_score > 0.5:

mood = "Joyful"

elif "sad" in lowered or sentiment\_score < -0.5:

mood = "Sad"

elif "angry" in lowered:

mood = "Angry"

elif any(word in lowered for word in ["anxious", "worried", "nervous"]):

mood = "Anxious"

elif "excited" in lowered:

mood = "Excited"

else:

mood = "Calm"

model = genai.GenerativeModel("gemini-1.5-flash")

response = model.generate\_content(user\_message)

bot\_reply = response.text if response.text else "I'm here to listen. Tell me more."

return jsonify({

"reply": bot\_reply,

"sentiment": sentiment\_label,

"sentiment\_score": sentiment\_score,

"mood": mood

})

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

app.run(debug=True)