FINAL CODE of our project

import streamlit as st

import random

import pandas as pd

from textblob import TextBlob

from googletrans import Translator

from datetime import datetime

# Set Page Configuration

st.set\_page\_config(page\_title="Your Personal Companion", layout="wide")

# Custom Black & Yellow Theme (Full UI)

st.markdown(

"""

<style>

body {

background-color: black;

color: yellow;

}

.stButton>button {

background-color: yellow;

color: black;

font-weight: bold;

}

.stTextInput>div>input {

background-color: black;

color: yellow;

border: 1px solid yellow;

}

.stTextArea>div>textarea {

background-color: black;

color: yellow;

border: 1px solid yellow;

}

.stRadio>div>label {

color: yellow;

}

.stSlider>div>label {

color: yellow;

}

.stMarkdown {

color: yellow;

}

.stVideo {

border: 2px solid yellow;

}

.stCheckbox>div>label {

color: yellow;

}

.title {

text-align: center;

font-size: 36px;

font-weight: bold;

color: black;

background-color: yellow;

padding: 10px;

margin-bottom: 20px;

}

.chat-message {

margin-bottom: 10px;

}

.user-message {

background-color: yellow;

color: black;

padding: 10px;

border-radius: 5px;

}

.bot-message {

background-color: gray;

color: black;

padding: 10px;

border-radius: 5px;

}

</style>

""", unsafe\_allow\_html=True

)

# Display Title

st.markdown('<div class="title">Zenalyse - Your Personal Companion</div>', unsafe\_allow\_html=True)

# Load dataset

@st.cache\_data

def load\_data():

return pd.read\_csv("/content/FinalDataset.csv", encoding="ISO-8859-1")

df = load\_data()

translator = Translator()

def get\_response(user\_input):

# Convert to lowercase for easier matching

clean\_input = user\_input.strip().lower()

# Custom responses

if clean\_input in {"hi", "hello", "hey"}:

return "👋 Hello! What would you like help with today?"

if clean\_input in {"how are you", "how are you?"}:

return "🤖 I'm a program without human feelings, but fully ready to assist you!"

if any(word in clean\_input for word in {"who are you", "what are you"}):

return "💻 I'm your stress management assistant, here to help you feel better!"

# Existing dataset lookup

responses = df[df["Context"].str.lower() == clean\_input]["Response"].tolist()

if responses:

response = random.choice(responses)

return add\_emoji(response)

return "🤖 I'm here for you! Want to talk more about something specific?"

def add\_emoji(response):

"""Add relevant emojis based on response content"""

response\_lower = response.lower()

emoji\_map = {

'relax': '🧘',

'stress': '😥',

'happy': '😊',

'sad': '😞',

'exercise': '🏃',

'sleep': '💤',

'food': '🍎',

'advice': '💡',

'help': '🆘'

}

for keyword, emoji in emoji\_map.items():

if keyword in response\_lower:

return f"{emoji} {response}"

return f"🤖 {response}"

# Mood Rating

st.subheader( "Mood Rating")

mood = st.radio("How are you feeling today?", ["😊", "😐", "😞"])

if mood == "😊":

st.write(" Great to hear! Keep spreading positivity!")

st.write(" You're doing awesome, and your happiness is contagious!")

st.video("https://www.youtube.com/watch?v=d-diB65scQU")

elif mood == "😐":

st.write(" Stay hopeful! A little motivation goes a long way.")

st.video("https://www.youtube.com/watch?v=1ZYbU82GVz4")

elif mood == "😞":

st.write(" Take a deep breath. You're doing great!")

st.video("https://www.youtube.com/watch?v=CA\_JjqvEGhw&t=32s") # Calm music

# Stress Assessment

st.subheader("🧠 Stress Assessment")

questions = [

("How often do you feel overwhelmed with tasks?", ["Rarely", "Sometimes", "Often", "Always"]),

("Do you experience difficulty in concentrating?", ["No", "Occasionally", "Frequently", "Almost always"]),

("How often do you feel anxious or worried?", ["Never", "Sometimes", "Often", "Always"]),

("Do you have trouble sleeping due to stress?", ["No", "Occasionally", "Frequently", "Almost always"]),

("How often do you feel tired or exhausted?", ["Rarely", "Sometimes", "Often", "Always"]),

("Do you find yourself procrastinating important tasks?", ["Never", "Sometimes", "Often", "Always"]),

("How often do you experience headaches or body pain?", ["Rarely", "Sometimes", "Often", "Always"]),

("Do you feel socially withdrawn or disconnected?", ["No", "Sometimes", "Frequently", "Always"]),

("How often do you engage in activities to relieve stress?", ["Daily", "A few times a week", "Rarely", "Never"]),

("Do you feel like you lack control over your time and responsibilities?", ["Never", "Sometimes", "Often", "Always"]),

]

stress\_score = 0

for question, choices in questions:

answer = st.radio(question, choices, key=question)

stress\_score += choices.index(answer) # Higher index means higher stress

# Calculate stress level

if stress\_score <= 10:

st.write("✅ \*You're doing great! Keep maintaining a balanced lifestyle.\*")

elif stress\_score <= 20:

st.write("😌 \*You're experiencing some stress, but it's manageable. Try relaxation techniques.\*")

else:

st.write("⚠ \*You're experiencing high stress! Consider practicing mindfulness or talking to someone.\*")

# Journaling with Close Button

st.subheader(" Journaling")

entry = st.text\_area("Write about your day")

close\_journal = st.button("Close Journal Section")

if close\_journal:

st.write(" Journal section closed!")

entry = "" # Clear journal content if closed

else:

if entry:

blob = TextBlob(entry)

sad\_words = ["stress", "frustrated", "sad", "overwhelmed", "tired", "anxious", "depressed", "lonely"]

if any(word in blob.lower() for word in sad\_words):

st.write(" Take a deep breath. Here are some calming tips to help you feel better:")

st.write("1. Try practicing deep breathing or mindfulness exercises.")

st.write("2. Take a walk outside and enjoy nature.")

st.write("3. Listen to calming music or your favorite relaxing playlist.")

st.write("4. Reach out to a friend or family member to talk.")

st.write("5. Write down your thoughts and feelings to release them.")

st.write("6. Take a moment to meditate or do some light stretching.")

# Checklist with date

st.subheader(" Your To-Do List")

tasks = st.text\_area("Enter your tasks (separate by commas)")

if tasks:

task\_list = tasks.split(",")

task\_dates = []

for task in task\_list:

task = task.strip()

task\_date = st.date\_input(f"Select due date for: {task}", min\_value=datetime.today())

task\_dates.append({"task": task, "due\_date": task\_date})

if st.button("Check Progress"):

st.write(" Stay motivated! Try the Pomodoro technique for better time management!")

for task in task\_dates:

st.write(f"Task: {task['task']} - Due Date: {task['due\_date'].strftime('%Y-%m-%d')}")

# Sleep Tracker

st.subheader("Sleep & Food Tracker")

sleep\_hours = st.slider("How many hours did you sleep?", 1, 12, 7)

if sleep\_hours <= 3:

st.write("😴 You got very little sleep! Try to rest more for better focus and energy. 🌟")

elif sleep\_hours <= 6:

st.write("😌 You got some sleep, but a little more can help you feel even better! Try to get at least 7 hours. 💤")

elif sleep\_hours <= 9:

st.write("✅ Great! You're getting enough sleep. A well-rested mind leads to better focus and productivity! 🚀")

else:

st.write("🌟 Amazing! Proper sleep fuels a fresh mind. Stay focused and make the most of your day! 💡🔥")

# food

diet = st.radio("Are you following a healthy diet?", ["Yes", "No"])

if diet == "No":

st.write(" A balanced diet can improve your mood. Try adding more greens and proteins!")

else:

st.write(" You're doing amazing! A healthy diet can work wonders for your mood and energy!")

# Food Lists

healthy\_foods = [

"Fruits", "Vegetables", "Whole Grains", "Proteins", "Dairy", "Water",

"Nuts", "Seeds", "Olive Oil", "Fish", "Eggs", "Yogurt", "Legumes", "Oats", "Avocado"

]

unhealthy\_foods = [

"Pizza", "Shawarma", "Noodles", "Pani Puri", "Packed Food", "Soft Drinks",

"French Fries", "Sugary Cereals", "Processed Meat", "Ice Cream", "Candy", "Fried Food"

]

st.subheader("🥗 Healthy & Unhealthy Food Checklist")

selected\_foods = st.multiselect("Select the foods you've eaten today", healthy\_foods + unhealthy\_foods)

selected\_healthy = [food for food in selected\_foods if food in healthy\_foods]

selected\_unhealthy = [food for food in selected\_foods if food in unhealthy\_foods]

# Provide Feedback Based on Selection

if selected\_foods:

if selected\_unhealthy:

st.write("⚠ Eat a balanced diet with plenty of fruits, vegetables, and water!")

else:

st.write("✅ Great job! You're making healthy choices!")

st.write(f"### Your Choices: {', '.join(selected\_foods)}")

else:

st.write("🍎 Remember to include more healthy foods in your diet!")

# Personal Chatbot

st.subheader(" Personal Assistant Chatbot")

if "messages" not in st.session\_state:

st.session\_state["messages"] = []

user\_input = st.text\_input("Type a message...")

if user\_input:

response = get\_response(user\_input)

st.session\_state["messages"].append({"role": "user", "content": f"👤 {user\_input}"})

st.session\_state["messages"].append({"role": "assistant", "content": response})

for message in st.session\_state["messages"]:

div\_class = "user-message" if message["role"] == "user" else "bot-message"

st.markdown(

f"<div class='chat-message {div\_class}'>"

f"{message['content']}"

"</div>",

unsafe\_allow\_html=True

)