import streamlit as st  
import pandas as pd  
import matplotlib.pyplot as plt  
import requests  
from datetime import datetime, timedelta  
import pytz  
  
# Define the Jerusalem timezone  
jerusalem\_tz = pytz.timezone('Asia/Jerusalem')  
  
# API Key and Base URL  
# API\_KEY = "f935fae84b5f4044931182757250501"  
  
API\_KEY = st.secrets["API\_KEY"]  
  
BASE\_URL = "http://api.weatherapi.com/v1/current.json"  
BASE\_URL\_FORECAST = "http://api.weatherapi.com/v1/forecast.json"  
BASE\_URL\_HISTORY = "http://api.weatherapi.com/v1/history.json"  
  
  
  
def fetch\_weather(city):  
 *"""  
 Fetch weather data for the given city using WeatherAPI.  
 """* url = f"{BASE\_URL}?key={API\_KEY}&q={city}"  
 response = requests.get(url)  
  
 if response.status\_code == 200:  
 return response.json()  
 else:  
 return None  
  
  
def fetch\_forecast(city, days=4):  
 *"""  
 Fetch forecast data for the given city using WeatherAPI.  
 """* url = f"{BASE\_URL\_FORECAST}?key={API\_KEY}&q={city}&days={days}"  
 response = requests.get(url)  
 if response.status\_code == 200:  
 return response.json()  
 return None  
  
  
def fetch\_historical(city, date):  
 *"""  
 Fetch historical weather data for a given city and date using WeatherAPI.  
 """* url = f"{BASE\_URL\_HISTORY}?key={API\_KEY}&q={city}&dt={date}"  
 response = requests.get(url)  
 if response.status\_code == 200:  
 return response.json()  
 return None  
  
  
# Streamlit UI  
st.title("Welcome to the best weather App \n"  
 "Check the weather in any city worldwide")  
  
# Make the input prompt larger  
st.markdown("### What is the city's name you would like to check?")  
  
# Adjust the input field width  
city = st.text\_input("", "", max\_chars=25)  
  
# Auto-correct for Holon to default to Israel  
if city.lower() == "holon":  
 city = "Holon, Israel"  
  
if city:  
 weather\_data = fetch\_weather(city)  
 if weather\_data:  
 # Current Weather  
 location = f"{weather\_data['location']['name']}, {weather\_data['location']['country']}"  
 temp\_c = weather\_data['current']['temp\_c']  
 feels\_like = weather\_data['current']['feelslike\_c']  
 condition = weather\_data['current']['condition']['text']  
 icon\_url = f"https:{weather\_data['current']['condition']['icon']}"  
  
 st.subheader(f"Weather in {location}")  
 st.image(icon\_url, width=100)  
 st.write(f"\*\*Current temperature:\*\* {temp\_c}°C")  
 st.write(f"\*\*Feels Like:\*\* {feels\_like}°C")  
 st.write(f"\*\*Condition:\*\* {condition}")  
  
 # Forecast  
 forecast\_data = fetch\_forecast(city, days=4) # Fetch 4 days to ensure 3 future days after excluding today  
 if forecast\_data:  
  
 forecast\_days = forecast\_data['forecast']['forecastday']  
  
 # Exclude today's forecast  
 today\_date = datetime.now().strftime('%Y-%m-%d')  
 future\_days = [day for day in forecast\_days if day['date'] != today\_date][:3] # Keep only 3 future days  
  
  
 # Determine the most frequent weather condition  
 conditions = [day['day']['condition']['text'] for day in future\_days]  
 most\_frequent\_condition = max(set(conditions), key=conditions.count).lower()  
  
  
 # Prepare data for the bar chart excluding today's forecast  
 forecast\_df = pd.DataFrame([  
 {  
 "Date": datetime.strptime(day['date'], "%Y-%m-%d").strftime("%d/%m/%Y"),  
 "Max Temp (°C)": day['day']['maxtemp\_c'],  
 "Min Temp (°C)": day['day']['mintemp\_c']  
 }  
 for day in future\_days # Use future\_days to exclude today  
 ])  
  
 # 3-Day Forecast Bar Chart and summary for the condition  
  
 # Calculate average chance of rain  
 avg\_rain\_chance = sum(day['day']['daily\_chance\_of\_rain'] for day in future\_days) / len(future\_days)  
  
 st.subheader("3-Day temperature forecast")  
  
 # Generate summary sentence  
 if avg\_rain\_chance > 50:  
 st.write(  
 f"The next 3 days are expected to be mostly {most\_frequent\_condition} with a high chance of rain.")  
 else:  
 st.write(  
 f"The next 3 days are expected to be mostly {most\_frequent\_condition} with no rain expectancy.")  
  
  
 forecast\_df[["Max Temp (°C)", "Min Temp (°C)"]] = forecast\_df[["Max Temp (°C)", "Min Temp (°C)"]].round(1)  
 forecast\_df.set\_index("Date", inplace=True)  
 forecast\_df.plot(kind='bar', figsize=(8, 5))  
 plt.title("Temperature Forecast (°C)")  
 plt.ylabel("Temperature (°C)")  
 plt.xlabel("Date")  
 plt.xticks(rotation=0) # Horizontal x-axis labels  
 st.pyplot(plt)  
  
 # Historical Data  
 st.subheader("Historical data comparison")  
 today = datetime.now()  
 past\_date = (datetime.now() - timedelta(days=365)).strftime('%Y-%m-%d') # Same day last year  
 historical\_data = fetch\_historical(city, past\_date)  
 if historical\_data:  
 historical\_temp = historical\_data['forecast']['forecastday'][0]['day']['avgtemp\_c']  
 temp\_difference = temp\_c - historical\_temp  
 if temp\_difference > 0:  
 st.write(f"\*\*Today is warmer by {temp\_difference:.1f}°C compared to the previous year in the same week (was {historical\_temp:.1f}°C).\*\*")  
 else:  
 st.write(f"\*\*Today is colder by {abs(temp\_difference):.1f}°C compared to the previous year in the same week (was {historical\_temp:.1f}°C).\*\*")  
 else:  
 st.error("Failed to fetch weather data. Please check the city name and try again.")