**NAME:A.KARTHIKEYAN**

**REG NO:192325143**

**SUB:CSA0814-PYTHON PROGRAM FOR LIST IMPLEMENTATION**

**Title 1: Real-Time Weather Monitoring System**

**Scenario:**

You are developing a real-time weather monitoring system for a weather forecasting company. The system needs to fetch and display weather data for a specified location.

**Tasks:**

1. **Model the data flow for fetching weather information from an external API and displaying it to the user.**
2. **Implement a Python application that integrates with a weather API (e.g., OpenWeatherMap) to fetch real-time weather data.**
3. **Display the current weather information, including temperature, weather conditions, humidity, and wind speed.**
4. **Allow users to input the location (city name or coordinates) and display the corresponding weather data.**

**Deliverables:**

* Data flow diagram illustrating the interaction between the application and the API.
* Pseudocode and implementation of the weather monitoring system.
* Documentation of the API integration and the methods used to fetch and display weather data.
* Explanation of any assumptions made and potential improvements.

**Approach:**

The goal of this real-time weather monitoring system is to fetch and display up-to-date weather information for a specified location using a third-party API. The system should be user-friendly, allowing the user to input a city name and retrieve relevant weather data such as temperature, weather conditions, humidity, and wind speed. The application also aims to represent this data visually using a Tkinter-based GUI, which is enhanced by graphs and images to convey seasonal changes.

**Pseudocode:**

 Initialize GUI components and variables.

 Define a function to fetch weather data using the API.

 Define a function to update the plot with new data:

* Fetch weather data.
* Update data lists (temperature, humidity, etc.).
* Clear and redraw the plot.

 Set **up** the GUI with dropdowns, labels, and plot area.

 Start periodic updates to fetch and display weather data.

**Detailed explanation of the actual code:**

* **Importing Libraries: The code begins by importing essential libraries. requests is used for making HTTP requests to the weather API. Tkinter is used for creating the graphical user interface, while Matplotlib is used to create and update the weather data graph. The PIL library is used to manage images, which are displayed based on the season inferred from the temperature.**
* **API Configuration: The API URL and required headers (including the API key) are defined. This configuration allows the application to authenticate and fetch data from the weather API.**
* **Data Storage Initialization: The program initializes several lists to store data like time, temperature, humidity, rainfall, and corresponding graph colors. These lists will be updated in real-time as new data is fetched from the API.**
* **Handling Season Images: The application includes a set of images representing different seasons (summer, winter, monsoon, and autumn). These images are stored in a dictionary, with each season mapped to a corresponding image path.**
* **Fetching Weather Data: The fetch\_weather function takes a city name as input, sends a request to the API, and retrieves the current weather data, including temperature, humidity, rainfall, and a descriptive weather condition. Depending on the condition (e.g., rain, cloud), the function assigns a color to be used in the graph.**
* **Season Determination: The get\_season function determines the current season based on the temperature. For example, if the temperature is above 30°C, it’s considered summer, while temperatures below 10°C indicate winter.**
* **Updating the Plot and Information: The update\_plot function is central to the application. It first fetches the weather data for the selected city, then updates the graph with the new data, and finally adjusts the displayed season image based on the temperature. The function runs at regular intervals, ensuring the data stays up to date.**
* **Graphical User Interface (GUI): The GUI is created using Tkinter. It includes a dropdown menu for selecting cities, a refresh button, and labels to display the current temperature, humidity, and rainfall. The graph and season image are also embedded in the GUI, providing a comprehensive view of the weather data.**
* **Embedding Semi-Opaque Text in the Graph: The application also features a watermark, "Naash Weather Tracking," embedded in the graph, providing a professional touch.**
* **Real-Time Updates: The application is designed to fetch and display updated weather data every minute. The GUI remains responsive, with the displayed data and images reflecting the most current information.**

**Assumptions made (if any):**

 **API Availability:** The system assumes that the WeatherAPI service is always available and that the API key remains valid for use.

 **Temperature as a Season Indicator:** The code assumes that temperature alone is sufficient to determine the season, which may not account for regional climatic nuances.

**Limitations:**

**API: obtaining API is the hardest task is ever felt during this project**

**API Limits: The application is subject to the rate limits imposed by the WeatherAPI, which could restrict the frequency of data updates.**

**Code:**

import requests  
import matplotlib.pyplot as plt  
from tkinter import Tk, Label, OptionMenu, StringVar, Button, Frame  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from datetime import datetime  
from PIL import Image, ImageTk  
  
# Define the API endpoint and your API key  
url = "https://weatherapi-com.p.rapidapi.com/current.json"  
headers = {  
 "x-rapidapi-key": "634ef24f07mshc0598f2465d3939p16058cjsn92985c8a46ea",  
 "x-rapidapi-host": "weatherapi-com.p.rapidapi.com"  
}  
  
# Initialize lists to store weather data  
times = []  
temperatures = []  
humidities = []  
rainfalls = []  
colors = []  
  
# Season images paths  
season\_images = {  
 'summer': "C:\\Users\\Avinaash.A\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093204.png",  
 'winter': "C:\\Users\\Avinaash.A\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093226.png",  
 'monsoon': "C:\\Users\\Avinaash.A\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093822.png",  
 'autumn': "C:\\Users\\Avinaash.A\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093213.png"  
}  
  
# Function to fetch weather data  
def fetch\_weather(city):  
 querystring = {"q": city}  
 try:  
 response = requests.get(url, headers=headers, params=querystring)  
 response.raise\_for\_status()  
 data = response.json()  
  
 current = data['current']  
 temperature = current['temp\_c']  
 humidity = current['humidity']  
 rainfall = current.get('precip\_mm', 0)  
 condition = current['condition']['text'].lower()  
  
 # Determine graph color based on weather condition  
 if 'rain' in condition:  
 color = 'blue'  
 elif 'cloud' in condition:  
 color = 'gray'  
 else:  
 color = 'orange'  
  
 return temperature, humidity, rainfall, color  
  
 except requests.exceptions.RequestException as e:  
 print(f"API request error: {e}")  
 return None, None, None, 'black'  
 except ValueError as e:  
 print(f"Value error: {e}")  
 return None, None, None, 'black'  
  
# Function to determine the season  
def get\_season(temperature):  
 if temperature >= 30:  
 return 'summer'  
 elif temperature <= 10:  
 return 'winter'  
 elif 10 < temperature < 30:  
 if temperature < 20:  
 return 'autumn'  
 else:  
 return 'monsoon'  
 return 'autumn'  
  
# Function to update the plot and weather information  
def update\_plot():  
 city = city\_var.get()  
 temperature, humidity, rainfall, color = fetch\_weather(city)  
  
 if temperature is not None:  
 times.append(datetime.now().strftime('%H:%M:%S'))  
 temperatures.append(temperature)  
 humidities.append(humidity)  
 rainfalls.append(rainfall)  
 colors.append(color)  
  
 ax.clear()  
 ax.plot(times, temperatures, color='red', label='Temperature (°C)')  
 ax.plot(times, humidities, color='green', label='Humidity (%)')  
 ax.plot(times, rainfalls, color='blue', label='Rainfall (mm)')  
  
 ax.set\_xlabel('Time')  
 ax.set\_ylabel('Value')  
 ax.legend(loc='upper left')  
 ax.set\_title(f'Weather Data for {city}')  
 plt.tight\_layout()  
  
 # Add semi-opaque watermark  
 plt.text(0.5, 0.5, 'Naash Weather Tracking', fontsize=40, color='gray', alpha=0.3,  
 ha='center', va='center', transform=ax.transAxes)  
  
 # Update the text labels with the latest values  
 current\_temperature.set(f"Temperature: {temperature} °C")  
 current\_humidity.set(f"Humidity: {humidity} %")  
 current\_rainfall.set(f"Rainfall: {rainfall} mm")  
  
 # Determine the season  
 season = get\_season(temperature)  
 season\_image\_path = season\_images.get(season, "C:\\Users\\Avinaash.A\\Downloads\\Design.jpeg")  
  
 # Load the image and display it  
 image = Image.open(season\_image\_path)  
 season\_photo = ImageTk.PhotoImage(image)  
 season\_label.config(image=season\_photo)  
 season\_label.image = season\_photo # Keep a reference to avoid garbage collection  
  
 # Redraw the canvas  
 canvas.draw()  
  
 root.after(60000, update\_plot)  
  
# Set up the GUI  
def setup\_gui():  
 global city\_var, root, canvas, fig, ax  
 global current\_temperature, current\_humidity, current\_rainfall, season\_label  
  
 root = Tk()  
 root.title("Naash Weather Tracker")  
  
 control\_frame = Frame(root, bg='white')  
 control\_frame.pack(side='top', fill='x')  
  
 Label(control\_frame, text="Select City:", bg='white').pack(side='left')  
  
 city\_var = StringVar(root)  
 city\_var.set("Mumbai")  
  
 city\_dropdown = OptionMenu(control\_frame, city\_var, "Mumbai", "Delhi", "Bangalore", "Kolkata", "Chennai",  
 "Hyderabad", "Lucknow", "Amritsar", "Jaipur", "Shimla")  
 city\_dropdown.pack(side='left')  
  
 refresh\_button = Button(control\_frame, text="Refresh", command=update\_plot)  
 refresh\_button.pack(side='left')  
  
 info\_frame = Frame(root, bg='white')  
 info\_frame.pack(side='top', fill='x')  
  
 current\_temperature = StringVar()  
 current\_humidity = StringVar()  
 current\_rainfall = StringVar()  
  
 Label(info\_frame, textvariable=current\_temperature, bg='white', font=('Helvetica', 16)).pack(side='left')  
 Label(info\_frame, textvariable=current\_humidity, bg='white', font=('Helvetica', 16)).pack(side='left')  
 Label(info\_frame, textvariable=current\_rainfall, bg='white', font=('Helvetica', 16)).pack(side='left')  
  
 season\_label = Label(root)  
 season\_label.pack(side='bottom', fill='x')  
  
 fig, ax = plt.subplots(figsize=(10, 6))  
  
 canvas = FigureCanvasTkAgg(fig, master=root)  
 canvas.draw()  
 canvas.get\_tk\_widget().pack(side='bottom', fill='both', expand=True)  
  
 update\_plot()  
 root.mainloop()  
  
# Run the GUI setup  
setup\_gui()

**Sample Output / Screen shots:**