Weather Dashboard Application

Report file

Name = Sahil Singh

Roll no= 2300290110151

Branch=CSIT (3C)

Submitted To: Nikita Sinha Mam

Course: B.Tech

Institution: Kiet group of institution

Date: 11-12-2024

Index

1. Introduction

The Weather Dashboard Application is a Python-based program designed to fetch and display real-time weather data for a user-specified location. It utilizes the OpenWeatherMap API for weather data retrieval and employs Tkinter for building the graphical user interface (GUI).

---

2. Objectives

- Develop a user-friendly GUI application.
- Provide real-time weather data for cities worldwide.
- Display weather information such as temperature, weather conditions, humidity, and wind speed.
- Integrate external APIs into a Python application.

---

3. Tools and Technologies

1. **Programming Language**: Python 3.x
2. **Libraries Used**:
   - requests: To interact with the OpenWeatherMap API and retrieve weather data.
   - tkinter: To create the graphical user interface.
   - messagebox (from Tkinter): To display error or alert messages.
3. **API**: OpenWeatherMap API

---

4. Features

1. **City Input**: Users can enter the name of a city to fetch weather details.
2. **Weather Display**: Information displayed includes:
   - City and country. ○ Current temperature (in Celsius).
   - Weather condition (e.g., "Cloudy", "Rainy"). ○ Humidity level (in percentage).
   - Wind speed (in meters per second).
3. **Error Handling**: The application:
   - Alerts the user if the input is invalid or empty. ○ Displays an error if the city is not found.

---

5. Implementation Details

a. API Integration

The application fetches data from the OpenWeatherMap API. The endpoint used is: bash Copy code

http://api.openweathermap.org/data/2.5/weather

Parameters:

- q: City name (entered by the user).
- appid: API key (required to authenticate requests).

- units: Set to "metric" for temperature in Celsius.

b. GUI Design

- **Tkinter** is used to create the GUI.
- Input field, labels, and buttons are arranged vertically.
- Labels dynamically update to display the weather data fetched from the API.

---

6. Code Structure

a. Main Functions

1. get_weather(city):
   - Sends a GET request to the API with the city name. ○ Parses and extracts weather details if the request is successful.
   - Returns None if the API response is invalid.
2. search_weather():
   - Reads the city name from the input field. ○ Calls get_weather() to fetch weather details. ○ Updates GUI labels with the fetched data or displays an error message.

b. GUI Components

1. **Entry Widget**: Allows the user to input a city name.
2. **Button Widget**: Triggers the search_weather function.
3. **Label Widgets**: Display the weather information dynamically.

c. Error Handling

- Uses messagebox.showerror() to display error alerts for invalid inputs or failed API requests.

---

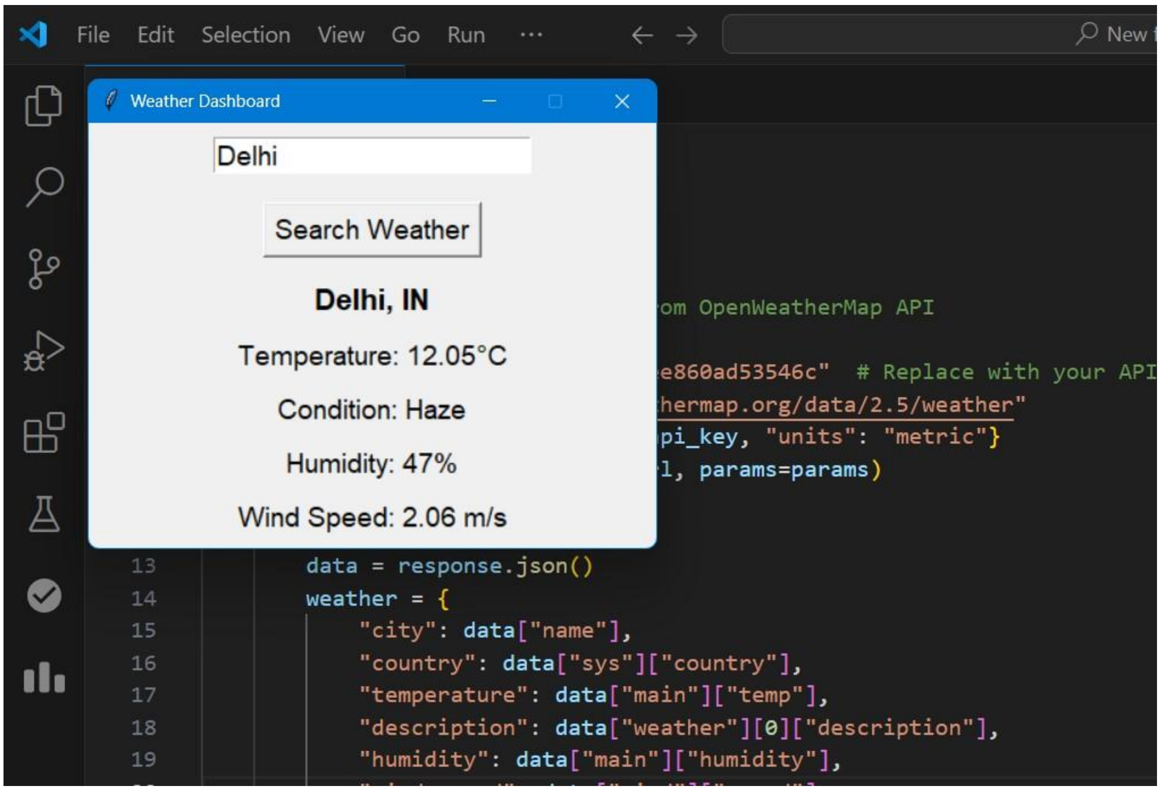
7. User Instructions

1. Open the application.
2. Enter the desired city name in the input field.
3. Click the "Search Weather" button.
4. View the weather details displayed on the screen.
5. If an error occurs, follow the error message instructions.

Code -------

```
import requests from tkinter

import * from tkinter import

messagebox


# Function to fetch weather data from OpenWeatherMap API def

get_weather(city):

   api_key = "e047e900e2145cfef0cee860ad53546c"  # Replace with your API key

base_url = "http://api.openweathermap.org/data/2.5/weather"

   params = {"q": city, "appid": api_key, "units": "metric"}

response = requests.get(base_url, params=params)


   if response.status_code == 200:

      data = response.json()

weather = {

         "city": data["name"],
            "country": data["sys"]["country"],
            "temperature": data["main"]["temp"],
            "description": data["weather"][0]["description"],
            "humidity": data["main"]["humidity"],
            "wind_speed": data["wind"]["speed"],
        }
        return weather
    else:
        return None

# Function to display weather data in the GUI
def search_weather():     city =
city_entry.get()

    if not city:
        messagebox.showerror("Input Error", "Please enter a city name")
        return

    weather = get_weather(city)
    if weather:
        location_label["text"] = f"{weather['city']}, {weather['country']}"
temperature_label["text"] = f"Temperature: {weather['temperature']}°C"
description_label["text"] = f"Condition: {weather['description'].capitalize()}"
humidity_label["text"] = f"Humidity: {weather['humidity']}%"        wind_label["text"]
= f"Wind Speed: {weather['wind_speed']} m/s"
    else:
        messagebox.showerror("Error", f"Could not find weather for '{city}'")

# Create main Tkinter app window
app = Tk() app.title("Weather
Dashboard")

app.geometry("400x300")

app.resizable(False, False)


# Input field for city name city_entry =

Entry(app, font=("Helvetica", 14))

city_entry.pack(pady=10) city_entry.insert(0,

"Enter city name")


# Search button

search_button = Button(app, text="Search Weather", font=("Helvetica", 14),
command=search_weather) search_button.pack(pady=10)

# Weather display labels location_label = Label(app, text="Location",

font=("Helvetica", 16, "bold")) location_label.pack(pady=5)


temperature_label = Label(app, text="Temperature", font=("Helvetica", 14))

temperature_label.pack(pady=5)


description_label = Label(app, text="Condition", font=("Helvetica", 14))

description_label.pack(pady=5)


humidity_label = Label(app, text="Humidity", font=("Helvetica", 14))

humidity_label.pack(pady=5)


wind_label = Label(app, text="Wind Speed", font=("Helvetica", 14))

wind_label.pack(pady=5)


# Run the Tkinter app app.mainloop()
```

8. Output -----



Delhi

Delhi, IN

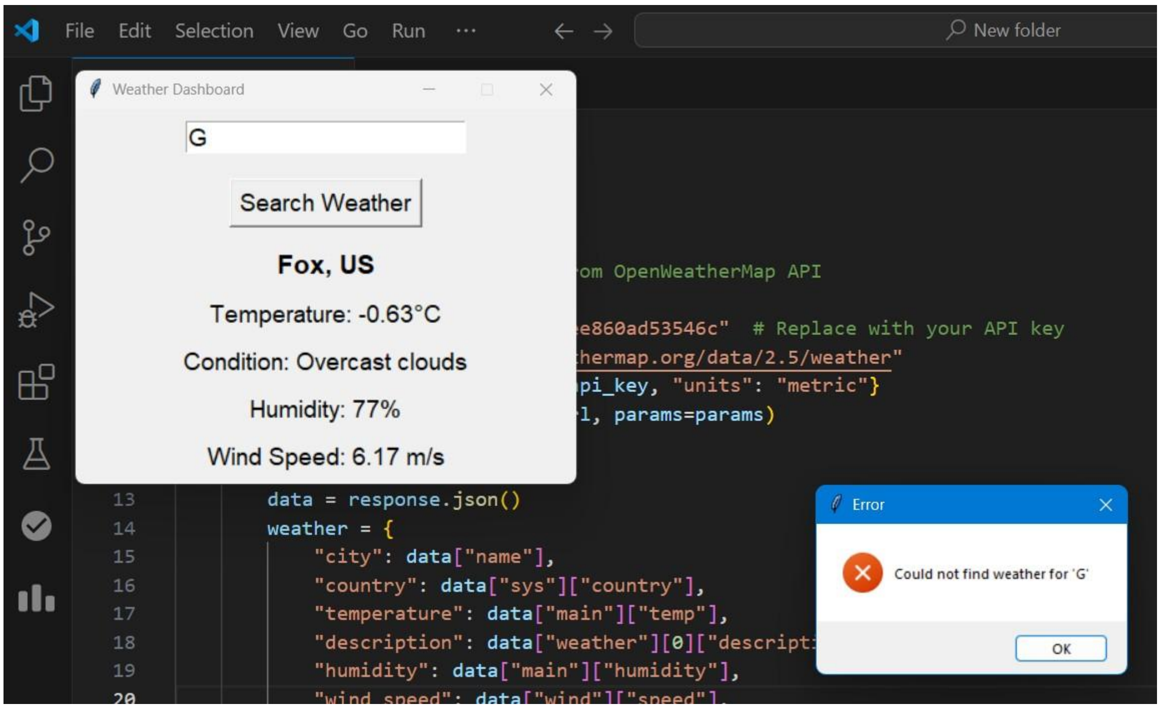
Temperature: 12.05°C

Condition: Haze

Humidity: 47%

Wind Speed: 2.06 m/s

```
om OpenWeatherMap API

e860ad53546c"  # Replace with your API
hermap.org/data/2.5/weather"
pi_key, "units": "metric"}
l, params=params)

        data = response.json()
        weather = {
            "city": data["name"],
            "country": data["sys"]["country"],
            "temperature": data["main"]["temp"],
            "description": data["weather"][0]["description"],
            "humidity": data["main"]["humidity"],
```



G

Fox, US

Temperature: -0.63°C

Condition: Overcast clouds

Humidity: 77%

Wind Speed: 6.17 m/s

Could not find weather for 'G'

```
om OpenWeatherMap API

e860ad53546c"  # Replace with your API key
hermap.org/data/2.5/weather"
pi_key, "units": "metric"}
l, params=params)

        data = response.json()
        weather = {
            "city": data["name"],
            "country": data["sys"]["country"],
            "temperature": data["main"]["temp"],
            "description": data["weather"][0]["descript
            "humidity": data["main"]["humidity"],
            "wind_speed": data["wind"]["speed"],
```

9. Challenges

- Properly handling API errors, such as invalid city names or network issues.
- Designing a clean and intuitive user interface.

---

10. Future Enhancements

- Allow users to toggle between Celsius and Fahrenheit.
- Add a 5-day weather forecast feature.
- Display additional weather parameters (e.g., sunrise/sunset times, atmospheric pressure).
- Improve GUI with custom themes or graphics.

---

11. Conclusion

The Weather Dashboard Application is a simple yet powerful tool for retrieving real-time weather data. It demonstrates how to integrate external APIs with Python and provides a practical example of building user-friendly applications with Tkinter.

---

12. References

1. Python Requests Library Documentation
2. Tkinter Documentation
3. OpenWeatherMap API Documentation

Thank you!