542 – Hackathon (12 – 24) Weather App Proposal

Group 4 : Navaneeth kumar, Ramya, Eslin, Sindhu, Venkat

The Weather App is a user-friendly application designed to provide real-time weather information for multiple cities. With a clean and intuitive interface, users can easily input a list of cities to retrieve current weather data, including temperature, weather conditions, humidity, and wind speed.

Features:

1. User-Friendly Interface:
- Clean and intuitive design for easy user interaction.
- Simple input of cities separated by commas.

2. Real-Time Weather Data:
- Fetches real-time weather data from the OpenWeatherMap API.
- Displays temperature, weather conditions, humidity, and wind speed.

3. Error Handling:
- Gracefully handles API errors and connection issues.
- Provides informative messages to users in case of problems.

4. Project Information:
Users can access project information, including contributor names, via the "Project Info" button.

Technical Details:

1. Programming Language:
- Developed using Python with the Tkinter library for the user interface.

2. API Integration:
- Utilizes the OpenWeatherMap API with a unique API key for weather data retrieval.

3. Customization:
- Users can customize the app's appearance, including background color, text color, and font styles.

4. Popup Dialogs:

- Implements popup dialogs for displaying project information and error messages.

Usage:

1. Input Cities:
- Users input a list of cities in the provided entry field, separated by commas.

2. Fetch Weather:
- Clicking "Get Weather" initiates the process of fetching weather data for specified cities.

3. View Results:
- Weather data for each city is displayed in the result area, showing temperature, weather conditions, humidity, and wind speed.

4. Project Info:
- Clicking "Project Info" opens a popup dialog displaying names of project contributors.

Conclusion:

The Weather App offers a straightforward solution for users to quickly access weather information for multiple cities. Its simplicity, user-friendly design, and real-time data retrieval make it a valuable tool for staying informed about current weather conditions.

Python code:

Installing required : "**requests"** module is used to make HTTP requests to the Weatherstack API. This module is a popular Python library for sending HTTP requests and receiving responses.

```
In [1]: pip install requests
        Requirement already satisfied: requests in ./anaconda3/lib/python3.11/site-packages (2.31.0)
        Requirement already satisfied: charset-normalizer<4,>=2 in ./anaconda3/lib/python3.11/site-packages (from requests)
        (2.0.4)
        Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python3.11/site-packages (from requests) (3.4)
        Requirement already satisfied: urllib3<3,>=1.21.1 in ./anaconda3/lib/python3.11/site-packages (from requests) (1.2
        6.16)
        Requirement already satisfied: certifi>=2017.4.17 in ./anaconda3/lib/python3.11/site-packages (from requests) (202
        3.7.22)
        Note: you may need to restart the kernel to use updated packages.
```

"**Tkinter**," it is a standard GUI (Graphical User Interface) toolkit that comes with Python. Tkinter provides a set of tools to create graphical user interfaces for desktop applications. It is commonly used to build windows, dialogs, buttons, text widgets, and other GUI elements.

The **get_weather** function in the provided Python code is designed to retrieve weather information for a specific city using the Weatherstack API.

1. **API Endpoint:** The function constructs the API endpoint by combining the base URL (**http://api.weatherstack.com/current**) with the specified parameters.
2. **HTTP GET Request:** It uses the **requests.get** method to make an HTTP GET request to the Weatherstack API.
3. **Response Handling:** The function checks if the response status code is 200 (indicating a successful request) and then returns the parsed JSON data. If there's an error, it displays an error message using Tkinter's **messagebox**.
4. **Exception Handling:** The function includes exception handling to deal with connection errors, such as when there is no internet connection.

```
In [*]: import tkinter as tk
        from tkinter import messagebox
        import requests

        def get_weather(api_key, city):
            base_url = "http://api.weatherstack.com/current"
            params = {"query": city, "access_key": api_key, "units": "m"}

            try:
                response = requests.get(base_url, params=params)
                data = response.json()

                if response.status_code == 200:
                    return data
                else:
                    messagebox.showerror("Error", f"Error for {city}: {data['error']['info']}")
                    return None
            except requests.ConnectionError:
                messagebox.showerror("Error", "Failed to connect to the weather API. Please check your internet connection.")
                return None
```

The **display_weather** function is responsible for updating the user interface with the weather information obtained from the Weatherstack API.

1. **Updating the UI:** The function uses the **result_text** variable, which is a Tkinter **StringVar**. This variable is associated with a Tkinter Label, and changing its value will automatically update the label in the user interface.
2. **Displaying Weather Data:** If the **weather_data** is not **None** (indicating that weather data was successfully obtained), the function constructs a string containing various weather details such as city name, temperature, weather description, humidity, and wind speed.
3. **Updating StringVar:** The **result_text.set(...)** method is used to update the **result_text** variable with the new information. This, in turn, updates the associated Tkinter Label in the user interface.
4. **Handling Failure:** If there's an issue fetching weather data (e.g., if **weather_data** is **None**), it appends an error message to the existing text in the label, indicating that weather data retrieval failed for a city.

```
def display_weather(weather_data, city):
    if 'location' in weather_data and 'current' in weather_data:
        result_text.set(
            result_text.get() +
            f"\nCity: {weather_data['location']['name']}, {weather_data['location']['country']}\n"
            f"Temperature: {weather_data['current']['temperature']}°C\n"
            f"Weather: {weather_data['current']['weather_descriptions'][0]}\n"
            f"Humidity: {weather_data['current']['humidity']}%\n"
            f"Wind Speed: {weather_data['current']['wind_speed']} m/s\n"
        )
    else:
        result_text.set(result_text.get() + f"\nFailed to fetch weather data for {city}. Please check the location n
```

The **get_weather_button_clicked** function is called when the "Get Weather" button is clicked in the GUI.

1. It retrieves the text entered in the entry field for cities (**city_entry.get()**).
2. Splits the entered text into a list of cities using commas (**split(",")**).
3. Clears the previous results displayed in the result area (**result_text.set("")**).
4. Iterates through each city in the list.
5. Strips any leading or trailing whitespaces from the city name (**city.strip()**).
6. Calls the **get_weather** function with the API key and the current city.
7. Passes the obtained weather data and the current city to the **display_weather** function.
8. The weather information for each city is displayed in the result area.

```
def get_weather_button_clicked():
    cities = city_entry.get().split(",")  # Split cities by comma
    api_key = "[redacted]"  # Replace with your Weatherstack API key

    result_text.set("")  # Clear previous results

    for city in cities:
        city = city.strip()  # Remove leading/trailing whitespaces
        weather_data = get_weather(api_key, city)
        display_weather(weather_data, city)
```

The **show_names_popup** function is called when the "Project Info" button is clicked in the GUI.

1. It creates a list of names belonging to "Group 4."
2. Constructs a formatted string with information about the project, including contributor names and a brief description of the Weather App.
3. Displays the constructed string in a pop-up dialog using the **messagebox.showinfo** method.

```
def show_names_popup():
    names = [
        "Group 4: Navaneeth Kumar 😎, Ramya 🐵, Eslin, Sindhu 🙈, Venkat 🤦",
        "\n\nThe Weather App is a user-friendly application designed to provide real-time weather information for mu
        "\n\nFeatures:"
        "\n1. User-Friendly Interface:"
        "\n   • Clean and intuitive design for easy user interaction."
        "\n   • Simple input of cities separated by commas."
        "\n2. Real-Time Weather Data:"
        "\n   • Fetches real-time weather data from the OpenWeatherMap API."
        "\n   • Displays temperature, weather conditions, humidity, and wind speed."
        "\n3. Error Handling:"
        "\n   • Gracefully handles API errors and connection issues."
        "\n   • Provides informative messages to users in case of problems."
        "\n4. Project Information:"
        "\n   Users can access project information, including contributor names, via the 'Project Info' button."
        "\n\nTechnical Details:"
        "\n1. Programming Language:"
        "\n   Developed using Python with the Tkinter library for the user interface."
        "\n2. API Integration:"
        "\n   Utilizes the OpenWeatherMap API with a unique API key for weather data retrieval."
        "\n3. Customization:"
        "\n   Users can customize the app's appearance, including background color, text color, and font styles."
        "\n4. Popup Dialogs:"
        "\n   Implements popup dialogs for displaying project information and error messages."
        "\n\nUsage:"
        "\n1. Input Cities:"
        "\n   Users input a list of cities in the provided entry field, separated by commas."
        "\n2. Fetch Weather:"
        "\n   Clicking 'Get Weather' initiates the process of fetching weather data for specified cities."
        "\n3. View Results:"
        "\n   Weather data for each city is displayed in the result area, showing temperature, weather conditions, h
        "\n4. Project Info:"
        "\n   Clicking 'Project Info' opens a popup dialog displaying names of project contributors."
        "\n\nConclusion:"
        "\n   The Weather App offers a straightforward solution for users to quickly access weather information for
        "\n   Its simplicity, user-friendly design, and real-time data retrieval make it a valuable tool for staying
    ]

    names_str = "\n\n".join(names)
    messagebox.showinfo("Project Information", names_str)
```

GUI setup

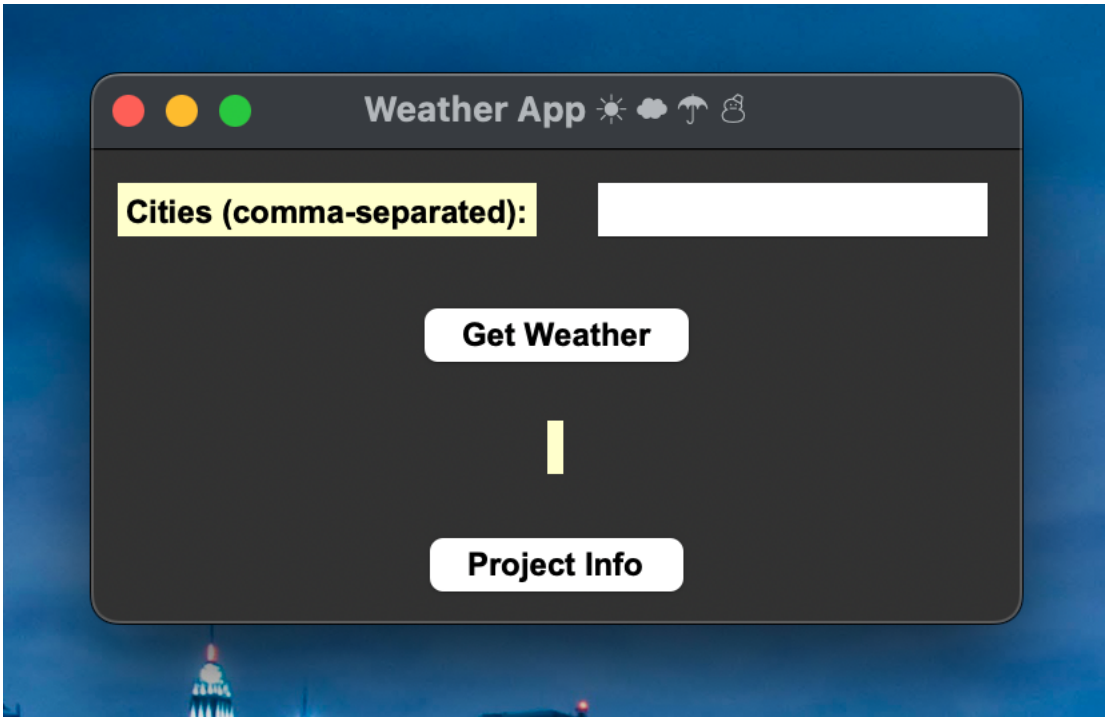
1. **Tkinter Application Setup:**
   The code initializes the Tkinter application using **tk.Tk()**.
2. **Setting Application Title:**
   The title of the application window is set to "Weather App ☀️☁️☂️☃️".
3. **Style Definitions:**
   - Background color (**bg_color**): Light yellow (**#ffffcc**).
   - Text color (**text_color**): Black (**#000000**).
   - Font style (**font_style**): Arial, 12-point, bold.
4. **Creating Labels and Entry Field:**
   A label ("Cities (comma-separated):") and an entry field for city input are created.
5. **Setting Cursor Color:**
   The cursor color of the entry field is set to black.
6. **Creating Buttons:**
   "Get Weather" button with a command to call **get_weather_button_clicked** function.
   "Project Info" button with a command to call **show_names_popup** function.
7. **Grid Layout Management:**

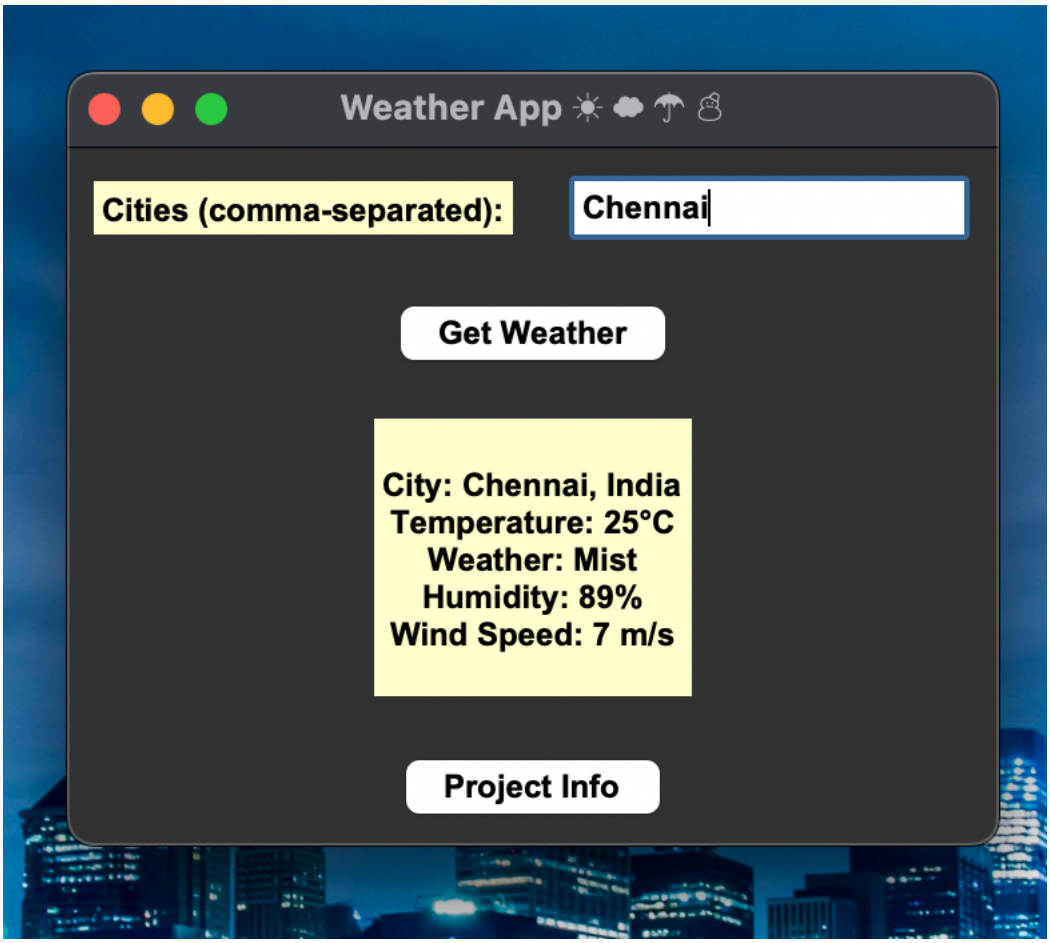
- Widgets are organized in a grid layout for proper placement within the application window.
- The **grid** method is used to specify row, column, padding, and sticky properties.

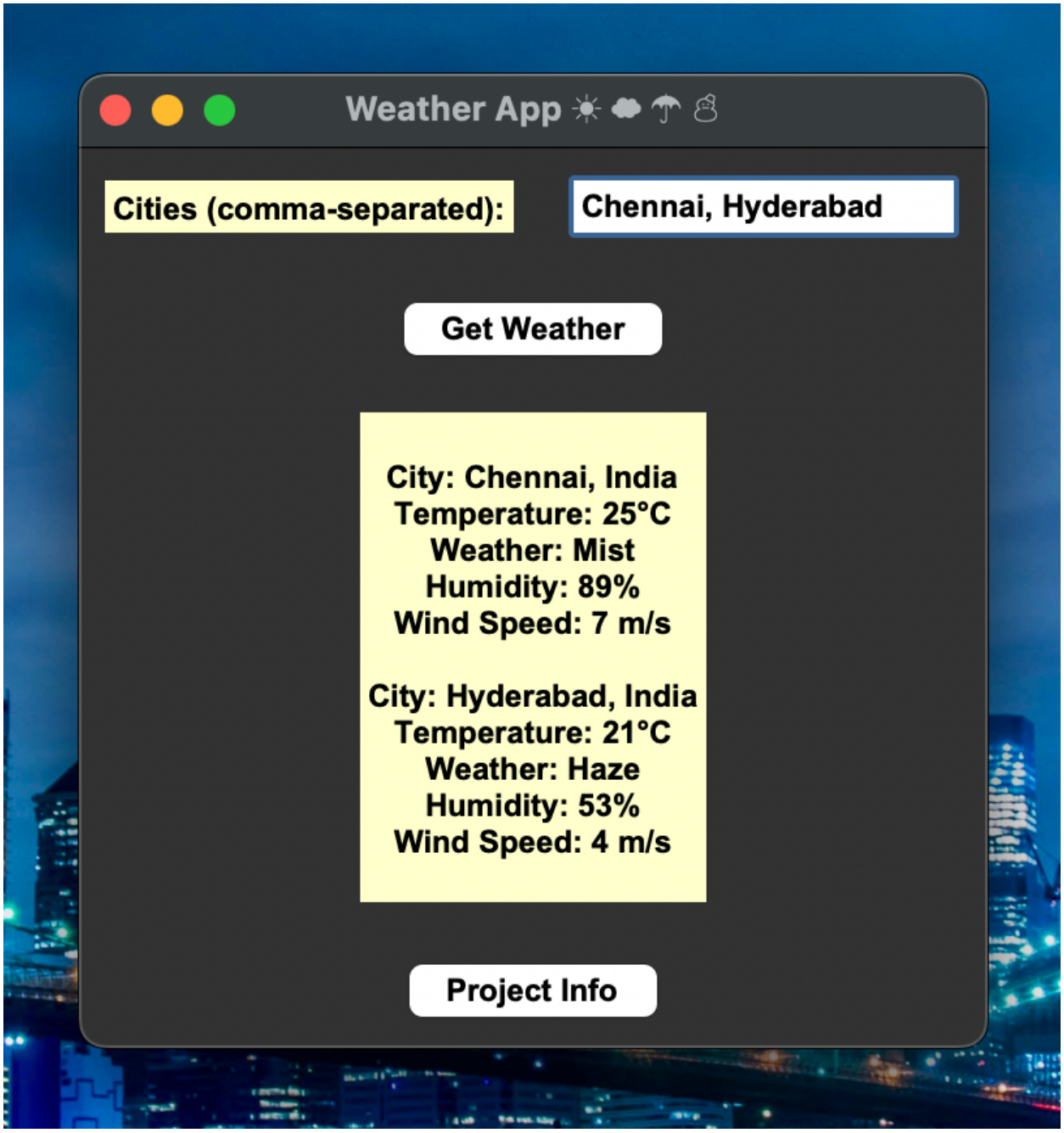
8. **Result Display Label:**
   A label (**result_label**) to display weather information is created.
9. **Running Tkinter Main Loop:**
   app.mainloop() starts the Tkinter event loop, allowing the application to respond to user interactions.

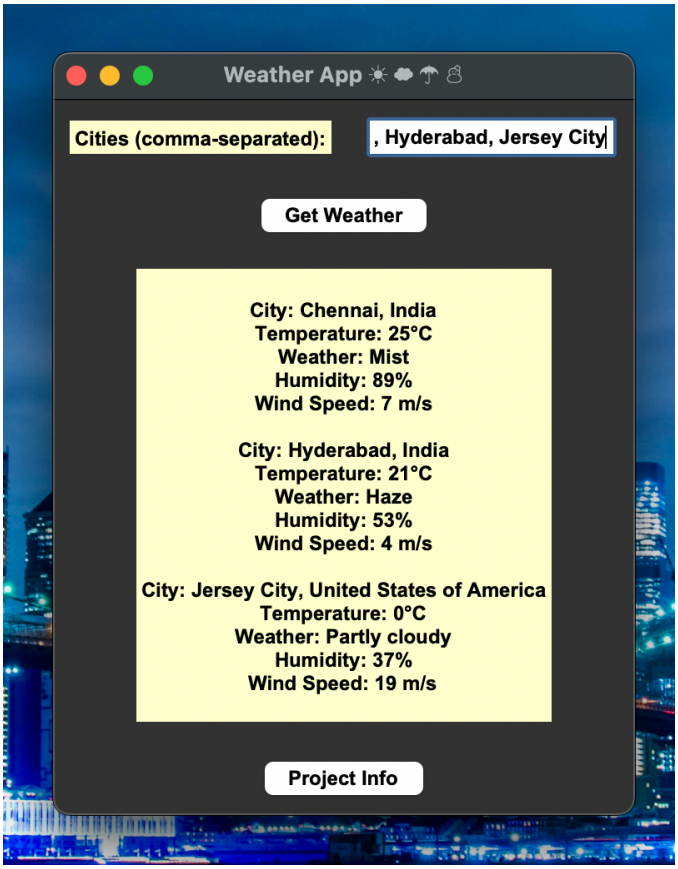
```
# GUI setup
app = tk.Tk()
app.title("Weather App ☀ ☁ ☂ ☃")

# Define styles for yellow background and black text
bg_color = "#ffffcc"  # Light yellow background color
text_color = "#000000"  # Black text color
font_style = ("Arial", 12, "bold")

# Create a label that looks like a title bar

city_label = tk.Label(app, text="Cities (comma-separated):", bg=bg_color, fg=text_color, font=font_style)
city_label.grid(row=1, column=0, padx=10, pady=10, sticky="W")

city_entry = tk.Entry(app, bg="white", fg=text_color, font=font_style)

city_entry.grid(row=1, column=1, padx=10, pady=10)

# Set cursor color to black
city_entry["insertbackground"] = text_color

get_weather_button = tk.Button(app, text="Get Weather", command=get_weather_button_clicked, bg=bg_color, fg=text_col
get_weather_button.grid(row=2, column=0, columnspan=2, pady=10)

result_text = tk.StringVar()
result_label = tk.Label(app, textvariable=result_text, bg=bg_color, fg=text_color, font=font_style)
result_label.grid(row=3, column=0, columnspan=2, padx=10, pady=10)

# Button to show names popup
show_names_button = tk.Button(app, text="Project Info", command=show_names_popup, bg=bg_color, fg=text_color, font=f
show_names_button.grid(row=4, column=0, columnspan=2, pady=10)

app.mainloop()
```

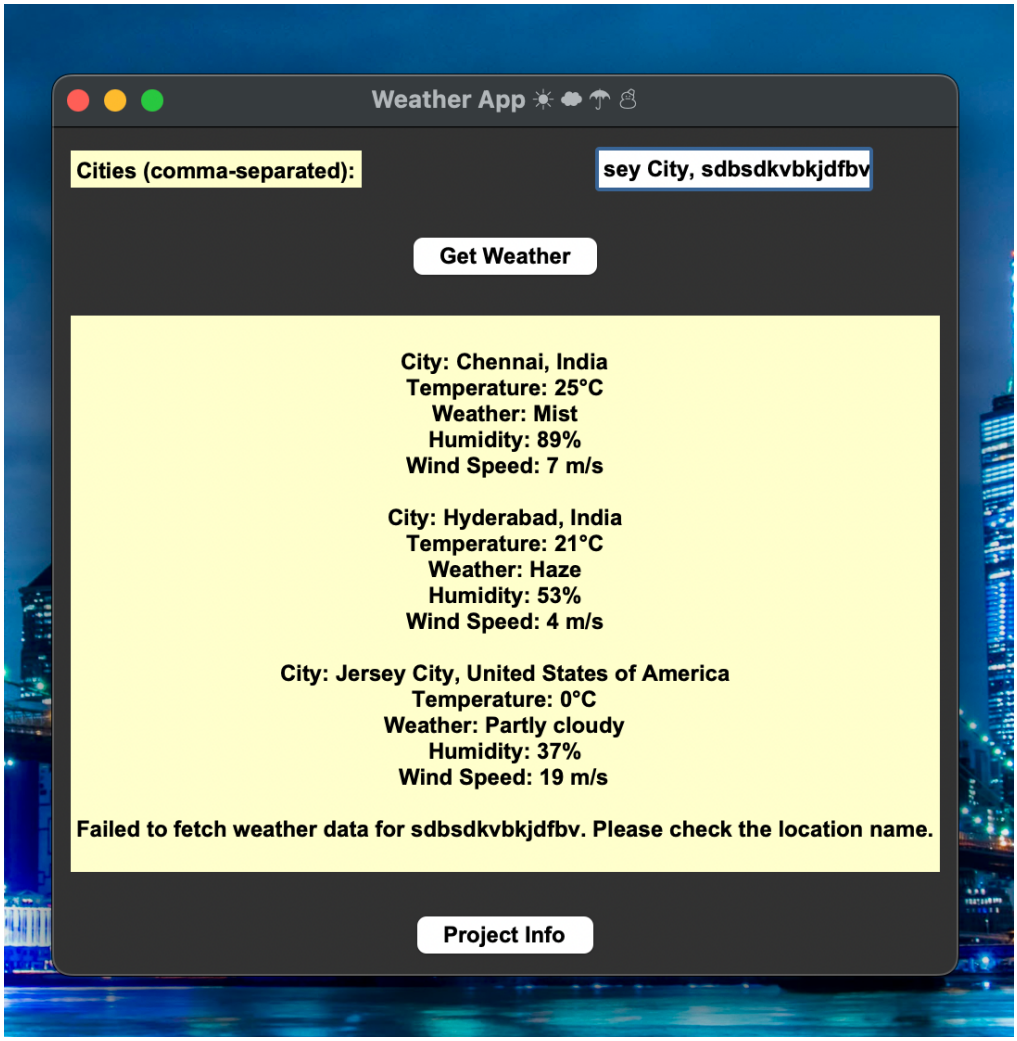
Output:


Weather App
Cities (comma-separated):
Get Weather
Project Info


Weather App
Cities (comma-separated):
Chennai
Get Weather
City: Chennai, India
Temperature: 25°C
Weather: Mist
Humidity: 89%
Wind Speed: 7 m/s
Project Info


Weather App
Cities (comma-separated):
Chennai, Hyderabad
Get Weather
City: Chennai, India
Temperature: 25°C
Weather: Mist
Humidity: 89%
Wind Speed: 7 m/s
City: Hyderabad, India
Temperature: 21°C
Weather: Haze
Humidity: 53%
Wind Speed: 4 m/s
Project Info


Weather App
Cities (comma-separated):
, Hyderabad, Jersey City
Get Weather
City: Chennai, India
Temperature: 25°C
Weather: Mist
Humidity: 89%
Wind Speed: 7 m/s
City: Hyderabad, India
Temperature: 21°C
Weather: Haze
Humidity: 53%
Wind Speed: 4 m/s
City: Jersey City, United States of America
Temperature: 0°C
Weather: Partly cloudy
Humidity: 37%
Wind Speed: 19 m/s
Project Info


Weather App
Cities (comma-separated):
sey City, sdbsdkvbkjdfbv
Get Weather
City: Chennai, India
Temperature: 25°C
Weather: Mist
Humidity: 89%
Wind Speed: 7 m/s
City: Hyderabad, India
Temperature: 21°C
Weather: Haze
Humidity: 53%
Wind Speed: 4 m/s
City: Jersey City, United States of America
Temperature: 0°C
Weather: Partly cloudy
Humidity: 37%
Wind Speed: 19 m/s
Failed to fetch weather data for sdbsdkvbkjdfbv. Please check the location name.
Project Info

Group 4: Navaneeth Kumar 😎, Ramya 🐶, Eslin, Sindhu 👧, Venkat 👨‍🦲

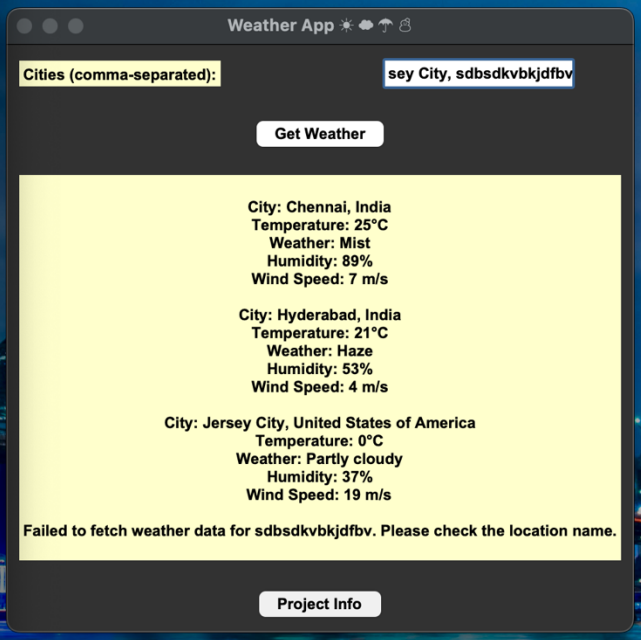
The Weather App is a user-friendly application designed to provide real-time weather information for multiple cities. With a clean and intuitive interface, users can easily input a list of cities to retrieve current weather data, including temperature, weather conditions, humidity, and wind speed.

Features:
1. User-Friendly Interface:
• Clean and intuitive design for easy user interaction.
• Simple input of cities separated by commas.
2. Real-Time Weather Data:
• Fetches real-time weather data from the OpenWeatherMap API.
• Displays temperature, weather conditions, humidity, and wind speed.
3. Error Handling:
• Gracefully handles API errors and connection issues.
• Provides informative messages to users in case of problems.
4. Project Information:
Users can access project information, including contributor names, via the 'Project Info' button.

Technical Details:
1. Programming Language:
Developed using Python with the Tkinter library for the user interface.
2. API Integration:
Utilizes the OpenWeatherMap API with a unique API key for weather data retrieval.
3. Customization:
Users can customize the app's appearance, including background color, text color, and font styles.
4. Popup Dialogs:
Implements popup dialogs for displaying project information and error messages.

Usage:
1. Input Cities:
Users input a list of cities in the provided entry field, separated by commas.
2. Fetch Weather:
Clicking 'Get Weather' initiates the process of fetching weather data for specified cities.
3. View Results:
Weather data for each city is displayed in the result area, showing temperature, weather conditions, humidity, and wind speed.
4. Project Info:
Clicking 'Project Info' opens a popup dialog displaying names of project contributors.

Conclusion:
The Weather App offers a straightforward solution for users to quickly access weather information for multiple cities. Its simplicity, user-friendly design, and real-time data retrieval make it a valuable tool for staying informed about current weather conditions.



```
pip install requests
import tkinter as tk
from tkinter import messagebox
import requests

def get_weather(api_key, city):
    base_url = "http://api.weatherstack.com/current"
    params = {"query": city, "access_key": api_key, "units": "m"}

    try:
        response = requests.get(base_url, params=params)
        data = response.json()

        if response.status_code == 200:
            return data
        else:
            messagebox.showerror("Error", f"Error for {city}: {data['error']['info']}")
            return None
    except requests.ConnectionError:
        messagebox.showerror("Error", "Failed to connect to the weather API. Please check your
internet connection.")
        return None

def display_weather(weather_data, city):
    if 'location' in weather_data and 'current' in weather_data:
        result_text.set(
            result_text.get() +
            f"\nCity: {weather_data['location']['name']}, {weather_data['location']['country']}\n"
            f"Temperature: {weather_data['current']['temperature']}°C\n"
            f"Weather: {weather_data['current']['weather_descriptions'][0]}\n"
            f"Humidity: {weather_data['current']['humidity']}%\n"
            f"Wind Speed: {weather_data['current']['wind_speed']} m/s\n"
        )
    else:
        result_text.set(result_text.get() + f"\nFailed to fetch weather data for {city}. Please check
the location name.\n")

def get_weather_button_clicked():
    cities = city_entry.get().split(",")  # Split cities by comma
    api_key = "4321a02377dc10474c4ad119e7f9e5ff"  # Replace with your Weatherstack API key

    result_text.set("")  # Clear previous results
```

```
    for city in cities:
        city = city.strip()  # Remove leading/trailing whitespaces
        weather_data = get_weather(api_key, city)
        display_weather(weather_data, city)

def show_names_popup():
    names = [
        "Group 4: Navaneeth Kumar 😎, Ramya 🐵, Eslin, Sindhu 🙈, Venkat 👨‍🦲",
        "\n\nThe Weather App is a user-friendly application designed to provide real-time
weather information for multiple cities. With a clean and intuitive interface, users can easily
input a list of cities to retrieve current weather data, including temperature, weather
conditions, humidity, and wind speed."
        "\n\nFeatures:"
        "\n1. User-Friendly Interface:"
        "\n   • Clean and intuitive design for easy user interaction."
        "\n   • Simple input of cities separated by commas."
        "\n2. Real-Time Weather Data:"
        "\n   • Fetches real-time weather data from the OpenWeatherMap API."
        "\n   • Displays temperature, weather conditions, humidity, and wind speed."
        "\n3. Error Handling:"
        "\n   • Gracefully handles API errors and connection issues."
        "\n   • Provides informative messages to users in case of problems."
        "\n4. Project Information:"
        "\n   Users can access project information, including contributor names, via the 'Project
Info' button."
        "\n\nTechnical Details:"
        "\n1. Programming Language:"
        "\n   Developed using Python with the Tkinter library for the user interface."
        "\n2. API Integration:"
        "\n   Utilizes the OpenWeatherMap API with a unique API key for weather data retrieval."
        "\n3. Customization:"
        "\n   Users can customize the app's appearance, including background color, text color, and
font styles."
        "\n4. Popup Dialogs:"
        "\n   Implements popup dialogs for displaying project information and error messages."
        "\n\nUsage:"
        "\n1. Input Cities:"
        "\n   Users input a list of cities in the provided entry field, separated by commas."
        "\n2. Fetch Weather:"
        "\n   Clicking 'Get Weather' initiates the process of fetching weather data for specified
cities."
```

```
        "\n3. View Results:"
        "\n   Weather data for each city is displayed in the result area, showing temperature, weather conditions, humidity, and wind speed."
        "\n4. Project Info:"
        "\n   Clicking 'Project Info' opens a popup dialog displaying names of project contributors."
        "\n\nConclusion:"
        "\n   The Weather App offers a straightforward solution for users to quickly access weather information for multiple cities."
        "\n   Its simplicity, user-friendly design, and real-time data retrieval make it a valuable tool for staying informed about current weather conditions."
    ]

    names_str = "\n\n".join(names)
    messagebox.showinfo("Project Information", names_str)

# GUI setup
app = tk.Tk()
app.title("Weather App ☀️☁️☂️☃️")

# Define styles for yellow background and black text
bg_color = "#ffffcc"  # Light yellow background color
text_color = "#000000"  # Black text color
font_style = ("Arial", 12, "bold")

# Create a label that looks like a title bar

city_label = tk.Label(app, text="Cities (comma-separated):", bg=bg_color, fg=text_color, font=font_style)
city_label.grid(row=1, column=0, padx=10, pady=10, sticky="W")

city_entry = tk.Entry(app, bg="white", fg=text_color, font=font_style)

city_entry.grid(row=1, column=1, padx=10, pady=10)

# Set cursor color to black
city_entry["insertbackground"] = text_color

get_weather_button = tk.Button(app, text="Get Weather", command=get_weather_button_clicked, bg=bg_color, fg=text_color, font=font_style)
get_weather_button.grid(row=2, column=0, columnspan=2, pady=10)

result_text = tk.StringVar()
```

```
result_label = tk.Label(app, textvariable=result_text, bg=bg_color, fg=text_color,
font=font_style)
result_label.grid(row=3, column=0, columnspan=2, padx=10, pady=10)

# Button to show names popup
show_names_button = tk.Button(app, text="Project Info", command=show_names_popup,
bg=bg_color, fg=text_color, font=font_style)
show_names_button.grid(row=4, column=0, columnspan=2, pady=10)

app.mainloop()
```