**Weather App Proposal**

The Weather App is a user-friendly applica5on designed to provide real-5me weather informa5on for mul5ple ci5es. With a clean and intui5ve interface, users can easily input a list of ci5es to retrieve current weather data, including temperature, weather condi5ons, humidity, and wind speed.

**Features:**

1. **User-Friendly Interface:** 
   * Clean and intui5ve design for easy user interac5on.
   * Simple input of ci5es separated by commas.

1. **Real-Time Weather Data:** 
   * Fetches real-5me weather data from the OpenWeatherMap API.
   * Displays temperature, weather condi5ons, humidity, and wind speed.

1. **Error Handling:** 
   * Gracefully handles API errors and connec5on issues.
   * Provides informa5ve messages to users in case of problems.

1. **Project InformaMon:**

Users can access project informa5on, including contributor names, via the "Project Info" buJon.

**Technical Details:**

**1. Programming Language:**

- Developed using Python with the Tkinter library for the user interface.

**2. API IntegraMon:**

- U5lizes the OpenWeatherMap API with a unique API key for weather data retrieval.

**3. CustomizaMon:**

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

**4. Popup Dialogs:**

- Implements popup dialogs for displaying project informa5on and error messages.

**Usage:**

**1. Input CiMes:**

- Users input a list of ci5es in the provided entry field, separated by commas.

**2. Fetch Weather:**

- Clicking "Get Weather" ini5ates the process of fetching weather data for specified ci5es.

**3. View Results:**

- Weather data for each city is displayed in the result area, showing temperature, weather condi5ons, humidity, and wind speed.

**4. Project Info:**

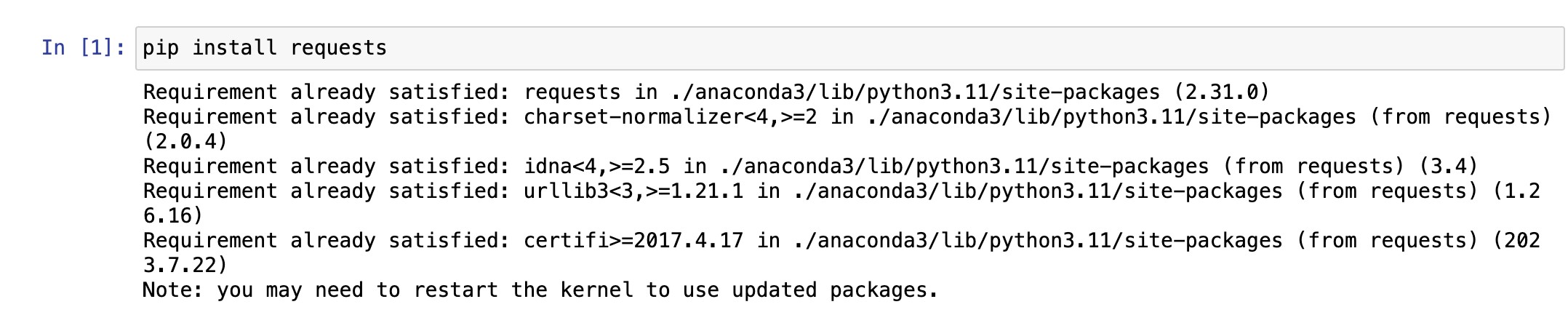
- Clicking "Project Info" opens a popup dialog displaying names of project contributors.

**Conclusion:**

The Weather App offers a straighRorward solu5on for users to quickly access weather informa5on for mul5ple ci5es. Its simplicity, user-friendly design, and real-5me data retrieval make it a valuable tool for staying informed about current weather condi5ons.

**Python code:**

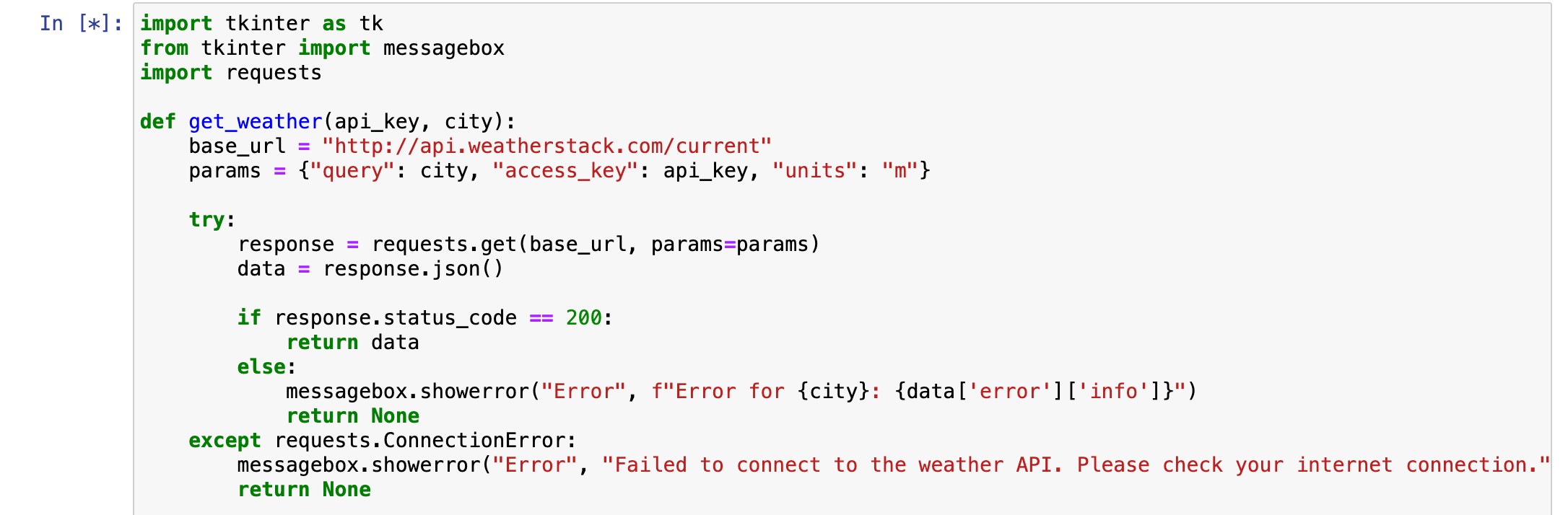
Installing required : “**requests”** module is used to make HTTP requests to the Weatherstack API. This module is a popular Python library for sending HTTP requests and receiving responses.



"**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 applica5ons. It is commonly used to build windows, dialogs, buJons, text widgets, and other GUI elements.

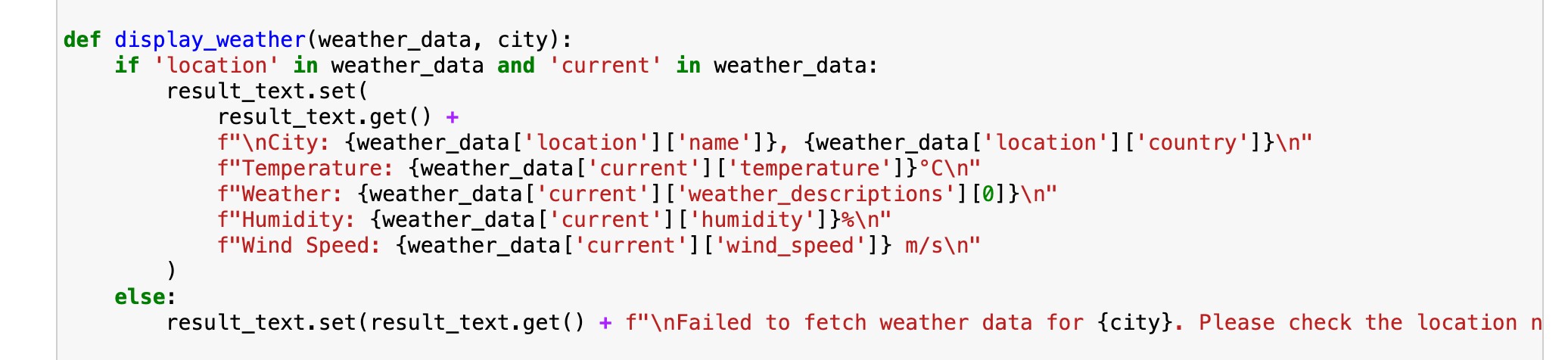
The **get\_weather** func5on in the provided Python code is designed to retrieve weather informa5on for a specific city using the Weatherstack API.

1. **API Endpoint:** The func5on constructs the API endpoint by combining the base URL (**hVp://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 func5on checks if the response status code is 200 (indica5ng 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. **ExcepMon Handling:** The func5on includes excep5on handling to deal with connec5on errors, such as when there is no internet connec5on.



The **display\_weather** func5on is responsible for upda5ng the user interface with the weather informa5on obtained from the Weatherstack API.

1. **UpdaMng the UI:** The func5on uses the **result\_text** variable, which is a Tkinter **StringVar**. This variable is associated with a Tkinter Label, and changing its value will automa5cally update the label in the user interface.
2. **Displaying Weather Data:** If the **weather\_data** is not **None** (indica5ng that weather data was successfully obtained), the func5on constructs a string containing various weather details such as city name, temperature, weather descrip5on, humidity, and wind speed.
3. **UpdaMng StringVar:** The **result\_text.set(...)** method is used to update the **result\_text** variable with the new informa5on. 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 exis5ng text in the label, indica5ng that weather data retrieval failed for a city.



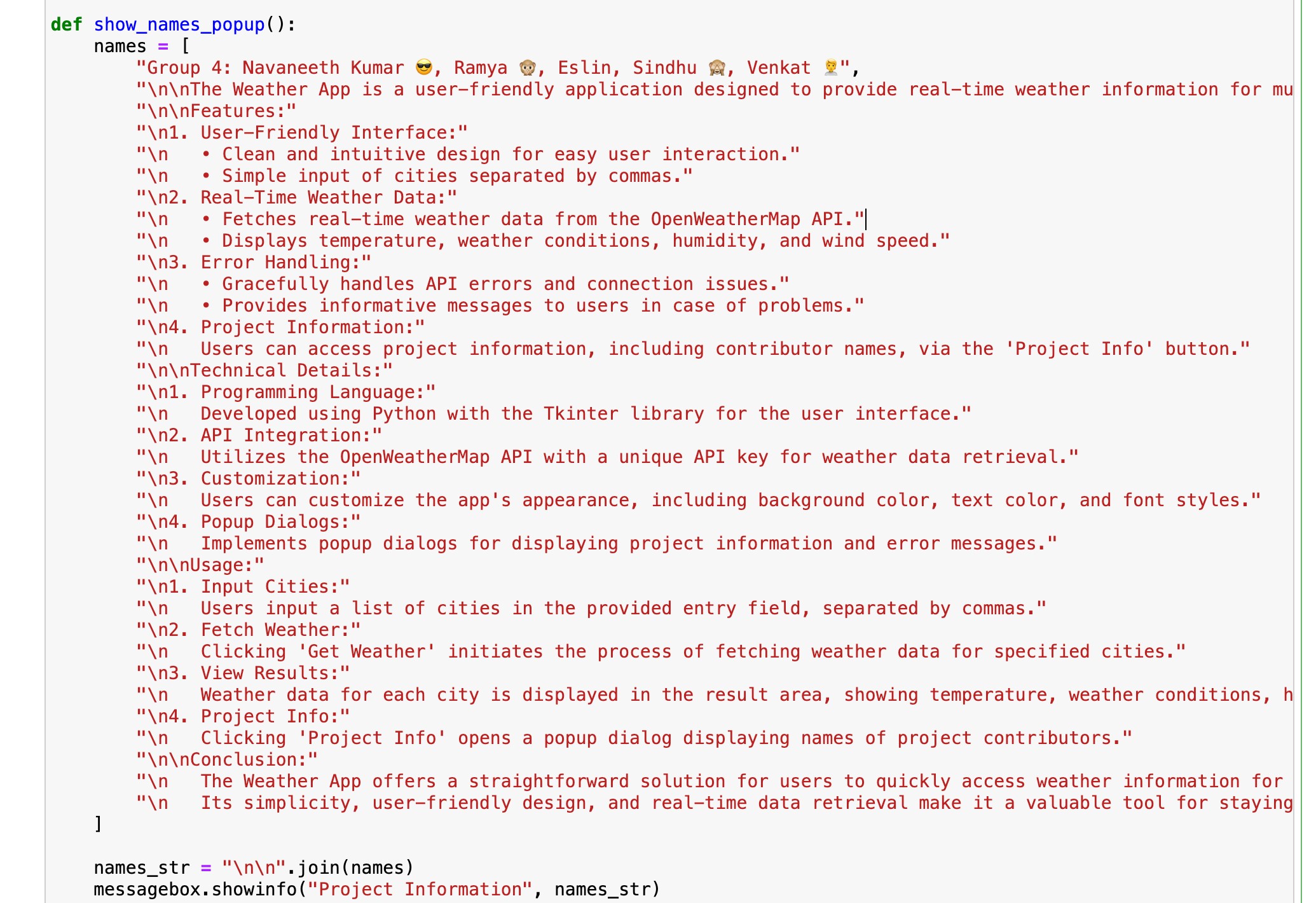
The **get\_weather\_buVon\_clicked** func5on is called when the "Get Weather" buJon is clicked in the GUI.

1. It retrieves the text entered in the entry field for ci5es (**city\_entry.get()**).
2. Splits the entered text into a list of ci5es 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** func5on with the API key and the current city.
7. Passes the obtained weather data and the current city to the **display\_weather** func5on.
8. The weather informa5on for each city is displayed in the result area.



The **show\_names\_popup** func5on is called when the "Project Info" buJon is clicked in the GUI.

1. It creates a list of names belonging to "Group 4."
2. Constructs a formaJed string with informa5on about the project, including contributor names and a brief descrip5on of the Weather App.
3. Displays the constructed string in a pop-up dialog using the **messagebox.showinfo** method.



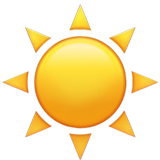
GUI setup

1. **Tkinter ApplicaMon Setup:**

The code ini5alizes the Tkinter applica5on using **tk.Tk()**.

1. **Se]ng ApplicaMon Title:**

The 5tle of the applica5on window is set to "Weather App ☀ ☁ ☂ ☃".



1. **Style DefiniMons:**
   * Background color (**bg\_color**): Light yellow (**#ffffcc**).
   * Text color (**text\_color**): Black (**#000000**).
   * Font style (**font\_style**): Arial, 12-point, bold.
2. **CreaMng Labels and Entry Field:**

A label ("Ci5es (comma-separated):") and an entry field for city input are created.

1. **Se]ng Cursor Color:**

The cursor color of the entry field is set to black.

1. **CreaMng BuVons:**

"Get Weather" buJon with a command to call **get\_weather\_buVon\_clicked** func5on.

"Project Info" buJon with a command to call **show\_names\_popup** func5on.

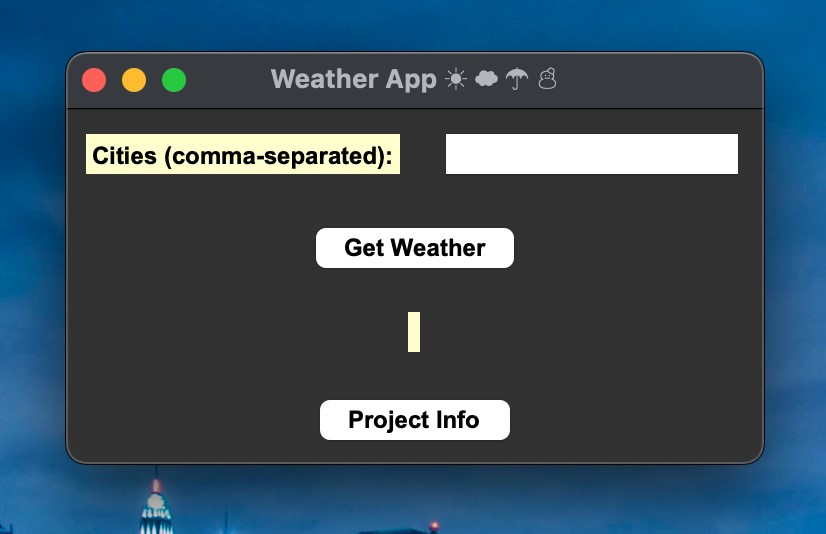
1. **Grid Layout Management:**
   * Widgets are organized in a grid layout for proper placement within the applica5on window.
   * The **grid** method is used to specify row, column, padding, and s5cky proper5es.
2. **Result Display Label:**

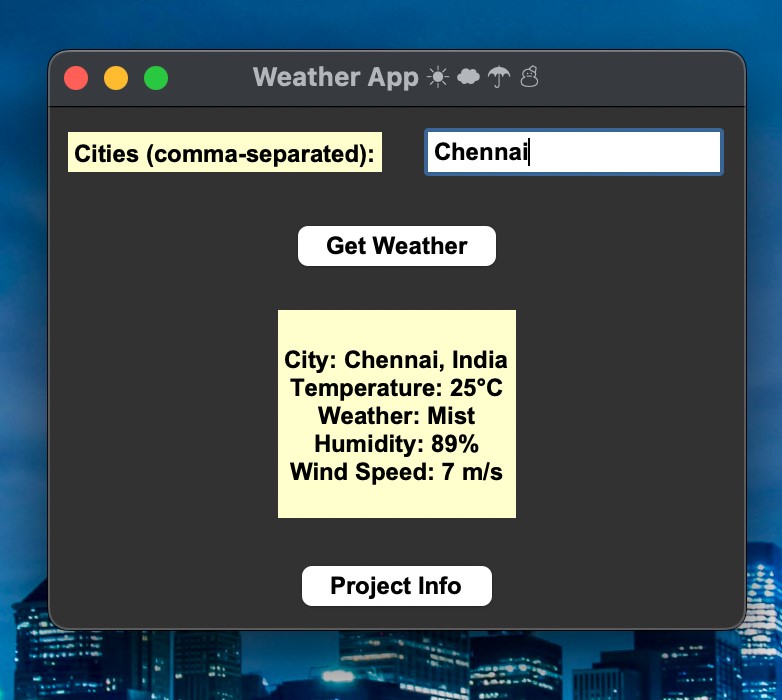
A label (**result\_label**) to display weather informa5on is created.

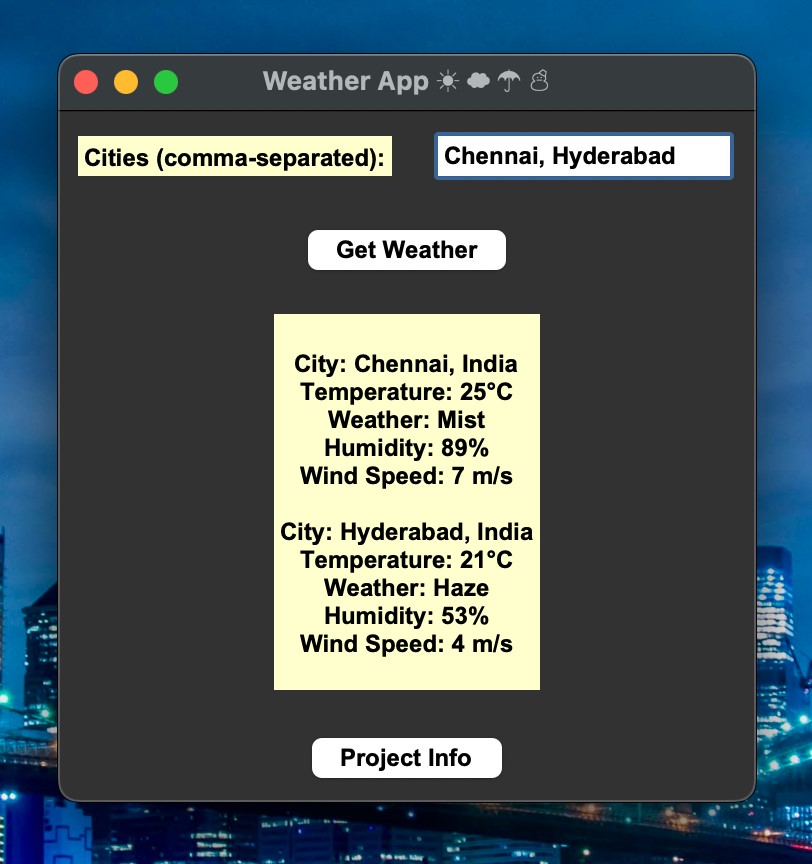
1. **Running Tkinter Main Loop:** **app.mainloop()** starts the Tkinter event loop, allowing the applica5on to respond to user interac5ons.

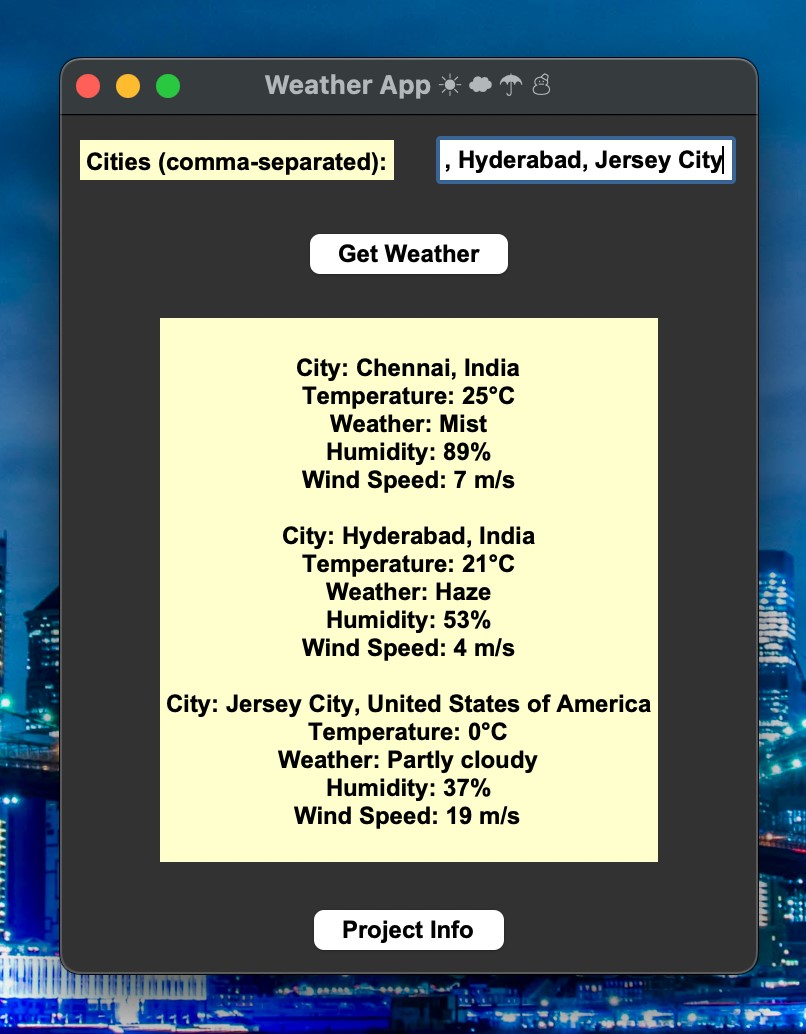


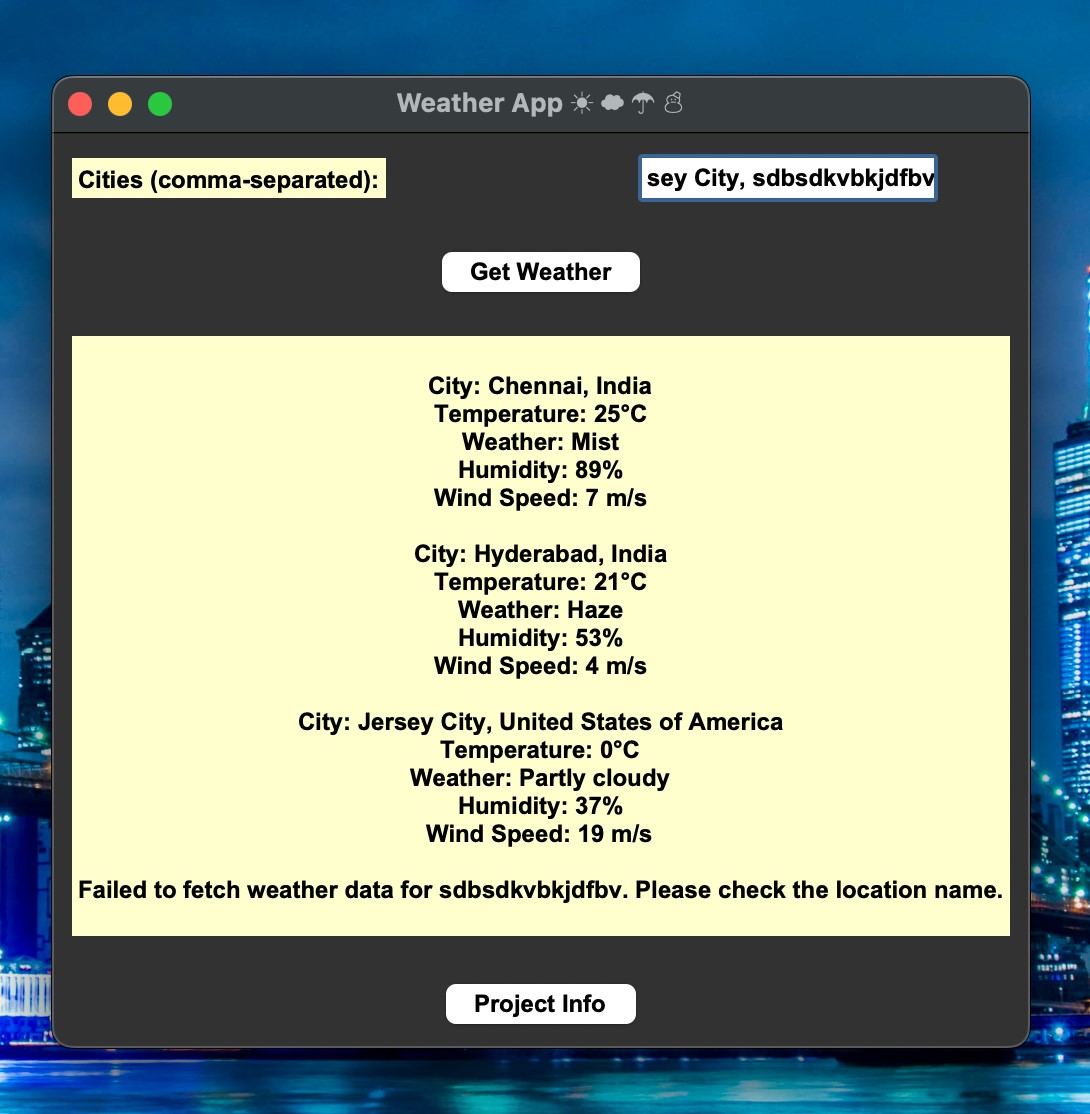
**Output:**

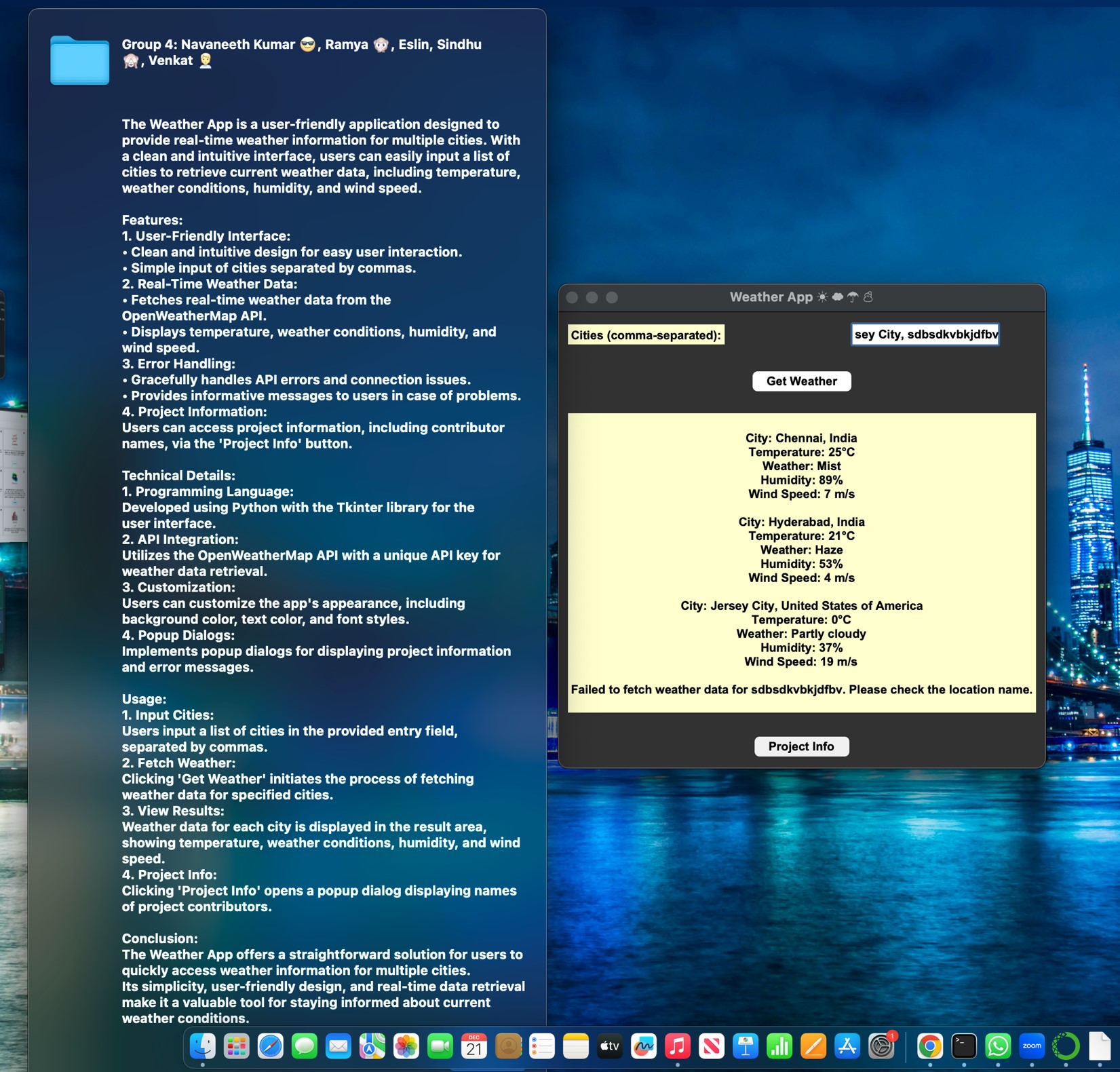












pip install requests import tkinter as tk

from tkinter import messagebox import requests

def get\_weather(api\_key, city): base\_url = "hJp://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.Connec5onError: messagebox.showerror("Error", "Failed to connect to the weather API. Please check your internet connec5on.") return None

def display\_weather(weather\_data, city): if 'loca5on' in weather\_data and 'current' in weather\_data:

result\_text.set( result\_text.get() +

f"\nCity: {weather\_data['loca5on']['name']}, {weather\_data['loca5on']['country']}\n" f"Temperature: {weather\_data['current']['temperature']}°C\n" f"Weather: {weather\_data['current']['weather\_descrip5ons'][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 loca5on name.\n")

def get\_weather\_buJon\_clicked():

ci5es = city\_entry.get().split(",") # Split ci5es by comma

api\_key = "4321a02377dc10474c4ad119e7f9e5ff" # Replace with your Weatherstack API key

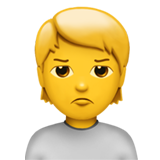
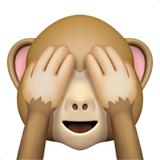
result\_text.set("") # Clear previous results

for city in ci5es:

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 applica5on designed to provide real-5me weather informa5on for mul5ple ci5es. With a clean and intui5ve interface, users can easily input a list of ci5es to retrieve current weather data, including temperature, weather condi5ons, humidity, and wind speed."

"\n\nFeatures:"

"\n1. User-Friendly Interface:"

"\n • Clean and intui5ve design for easy user interac5on."

"\n • Simple input of ci5es separated by commas."

"\n2. Real-Time Weather Data:"

"\n • Fetches real-5me weather data from the OpenWeatherMap API."

"\n • Displays temperature, weather condi5ons, humidity, and wind speed." "\n3. Error Handling:"

"\n • Gracefully handles API errors and connec5on issues."

"\n • Provides informa5ve messages to users in case of problems."

"\n4. Project Informa5on:"

"\n Users can access project informa5on, including contributor names, via the 'Project

Info' buJon."

"\n\nTechnical Details:"

"\n1. Programming Language:"

"\n Developed using Python with the Tkinter library for the user interface." "\n2. API Integra5on:"

"\n U5lizes the OpenWeatherMap API with a unique API key for weather data retrieval." "\n3. Customiza5on:"

"\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 informa5on and error messages." "\n\nUsage:"

"\n1. Input Ci5es:"

"\n Users input a list of ci5es in the provided entry field, separated by commas." "\n2. Fetch Weather:"

"\n Clicking 'Get Weather' ini5ates the process of fetching weather data for specified ci5es."

"\n3. View Results:"

"\n Weather data for each city is displayed in the result area, showing temperature, weather condi5ons, 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 straighRorward solu5on for users to quickly access weather informa5on for mul5ple ci5es."

"\n Its simplicity, user-friendly design, and real-5me data retrieval make it a valuable tool for staying informed about current weather condi5ons."

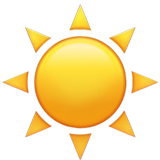
]

names\_str = "\n\n".join(names)

messagebox.showinfo("Project Informa5on", names\_str)

# GUI setup app = tk.Tk()

app.5tle("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 5tle bar

city\_label = tk.Label(app, text="Ci5es (comma-separated):", bg=bg\_color, fg=text\_color, font=font\_style)

city\_label.grid(row=1, column=0, padx=10, pady=10, s5cky="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\_buJon = tk.BuJon(app, text="Get Weather",

command=get\_weather\_buJon\_clicked, bg=bg\_color, fg=text\_color, font=font\_style) get\_weather\_buJon.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)

# BuJon to show names popup

show\_names\_buJon = tk.BuJon(app, text="Project Info", command=show\_names\_popup, bg=bg\_color, fg=text\_color, font=font\_style)

show\_names\_buJon.grid(row=4, column=0, columnspan=2, pady=10)

app.mainloop()