Weather App Using OpenWeatherMap API - Project Report

1. Introduction

This report documents the design and implementation of a Weather Application built using Python. The application utilizes the OpenWeatherMap API to retrieve and display current weather information based on user input (city name). It is developed with the `tkinter` library to provide a graphical user interface (GUI) and `requests` to manage HTTP communication with the external API.

2. Objectives

- To develop a simple, user-friendly weather forecasting application.
- To integrate third-party APIs (OpenWeatherMap) with Python.
- To learn GUI development using `tkinter`.

3. Tools and Technologies Used

- Programming Language: Python
- Libraries:
  - Tkinter: For creating the graphical user interface.
  - `requests`: For sending HTTP requests to the API.
- API Used: OpenWeatherMap API

4. Prerequisites

- Python must be installed on the system.
- The `requests` library should be installed (`pip install requests`).
- A valid API key from OpenWeatherMap is needed to access the weather data.

5. System Design

The application consists of the following major components:

- User Interface: Built with Tkinter, it includes an entry field for the city name, a button to trigger the weather fetch, and a label to display the results.

- Backend Logic: Communicates with the OpenWeatherMap API using HTTP GET requests and parses the JSON response.

6. Workflow

1. The user inputs the city name into the entry field.

2. On clicking the "Get Weather" button, a function is triggered.

3. This function sends a request to the OpenWeatherMap API with the city name and the API key.

4. The response JSON is parsed to extract relevant weather information.

5. The extracted information is displayed in the GUI.

7. Code Explanation

```
import tkinter as tk
```

```
from tkinter import messagebox
```

```
import requests
```

- Importing required libraries. Tkinter is used for the GUI, requests for API calls, and messagebox for displaying error/warning popups.

```
API_KEY = "6a548b062158c655f488c867754c3045"
```

- Replace this with your own API key from OpenWeatherMap.

```
def get_weather(city):
    base_url = "http://api.openweathermap.org/data/2.5/weather"
    params = {
        "q": city,
        "appid":6a548b062158c655f488c867754c3045 ,
        "units": "metric"
```

- This function constructs the API request with the city name, API key, and unit (Celsius).

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

- It sends the request, checks for errors, and parses the JSON response.

```
        weather = {
            "city": data["name"],
            "temperature": data["main"]["temp"],
            "description": data["weather"][0]["description"],
            "humidity": data["main"]["humidity"],
            "wind": data["wind"]["speed"]
        }
        return weather
```

- Extracts and returns relevant information as a dictionary.

```
    except requests.exceptions.HTTPError as errh:
        messagebox.showerror("HTTP Error", str(errh))
    except requests.exceptions.RequestException as err:
        messagebox.showerror("Error", str(err))
def show_weather():
    city = city_entry.get()
    if not city:
    weather = get_weather(city)
    if weather:
        result_label.config(
            text=f"City: {weather['city']}\n"
                 f"Temperature: {weather['temperature']} °C\n"
                 f"Weather: {weather['description'].title()}\n"
```

```
            f"Humidity: {weather['humidity']}%\n"
            f"Wind Speed: {weather['wind']} m/s"
```

- Retrieves city name, fetches weather data, and displays it in the GUI.

GUI Setup

```
root = tk.Tk()
root.title("Weather App")
root.geometry("300x300")
root.resizable(False, False)
```

- Initializes the main window.

```
tk.Label(root, text="Enter City:", font=("Arial", 12)).pack(pady=10)
city_entry = tk.Entry(root, width=25, font=("Arial", 12))
city_entry.pack()
```

- Creates label and entry widget for city input.

```
tk.Button(root, text="Get Weather", command=show_weather, font=("Arial", 12)).pack(pady=10)
result_label = tk.Label(root, text="", font=("Arial", 10), justify="left")
result_label.pack(pady=10)
```

- Creates a button to fetch weather and a label to display results.

```
root.mainloop()
```

- Starts the GUI event loop.

8. Testing and Validation

The app has been tested for various city names and successfully displays real-time data. Error messages are shown when an invalid city is entered or if there's a network problem.

9. Limitations

- No autocomplete or dropdown for cities.

- Doesn't support forecast, only current weather.

- Requires an internet connection.

10. Future Improvements

- Add forecast functionality.

- Improve UI with modern libraries like `PyQt` or `Kivy`.

- Add geolocation support.

- Show weather icons.

11. Conclusion

This weather application demonstrates how Python can interact with web APIs and create GUI applications. It provides a practical example of combining frontend and backend skills in a single project, making it an excellent learning experience for beginners and intermediates alike.