📘 WEATHER MONITORING APPLICATION

---

1. Cover Page

Project Title:
Weather Monitoring (API) Using Python

Submitted By:
Your Name : ANSHU KUMAR
Reg NO. - 25BHI10040

Submitted To:Dr. G Ganesan

Academic Year: 2025-2026

2. Introduction

Weather conditions affect various aspects of human life, such as travel planning, agriculture, aviation, outdoor events, and everyday decision-making. Accurate and timely weather updates are therefore crucial.

This project aims to develop a desktop-based Weather Monitoring Application using Python, Tkinter (GUI framework), and the OpenWeatherMap API to fetch real-time weather data for any city in the world.

The project is user-friendly, lightweight, fast, and requires minimal system resources. By simply entering a city name, the user receives:

- Current temperature
- Weather condition
- Minimum and maximum temperature

- Humidity and pressure
- Wind speed
- Sunrise and sunset timings

This project demonstrates concepts of:

- API integration
- JSON data handling
- GUI application development
- Event-driven programming
- Data formatting and presentation

3. Problem Statement

Weather information is often scattered across different websites and apps, requiring multiple steps to obtain basic details. Many official weather apps include advertisements, heavy interface elements, or require account creation.

There is a need for:

- A simple desktop tool
- Zero advertisements
- Fast loading
- Easy interface
- Accurate real-time data
- Thus, the project solves the problem by creating a one-window weather app that provides real-time weather conditions quickly and clearly.

4. Functional Requirements

4.1 User Functions

- Users can enter any city name into the input field.
- Users can press Enter to fetch its weather details.

4.2 System Functions

The system will:

1. Validate user input.
2. Form a request URL using the provided city name and API key.
3. Send an HTTP request to the OpenWeatherMap API.
4. Receive weather information in JSON format.
5. Extract the required fields from JSON.
6. Convert temperature from Kelvin to Celsius.
7. Convert sunrise/sunset UNIX timestamps to readable time.
8. Display the results on the GUI.

4.3 Display Requirements

The app displays:

- Weather condition (e.g., Clear, Clouds, Rain)
- Current temperature (°C)
- Minimum & maximum temperature
- Pressure (hPa)
- Humidity (%)

- Wind speed (m/s)
- Sunrise time
- Sunset time

5. Non-Functional Requirements

5.1 Performance

- The weather data should load within 1–3 seconds depending on internet speed.

5.2 Reliability

- The app must handle wrong input gracefully.
- Should not crash even if API fails.

5.3 Usability

- Simple GUI for beginners.
- Single-click or Enter key functionality.

5.4 Portability

- Works on any system with Python installed (Windows, Linux, Mac).

5.5 Maintainability

- Code is modular.
- Easy to change fonts, colors, or add more features.

5.6 Availability

- The app requires an active internet connection to fetch data.

5.7 Security

- API key is used securely inside code.
- No login or sensitive data is used.

6. System Architecture

The architecture consists of three main components:

1. User Interface Layer (Tkinter GUI)

- Accepts user input (city name)
- Displays processed weather data

2. Application Logic Layer

- Handles events (Enter key)
- Sends API request
- Processes JSON response
- Formats output for display

3. External Service Layer

- OpenWeatherMap API
  Provides real-time weather data in JSON format.

Workflow Diagram

```
Start
  |
  v
User enters city name
  |
  v
Form API request URL
  |
  v
Send request to Weather Server
  |
  v
Receive JSON data
  |
  v
Extract required fields
  |
  v
Convert data (Kelvin->Celsius)
  |
  v
Display weather information
  |
  v
End
```

7. Design Decisions & Rationale

1. Python Tkinter Chosen

- Comes built into Python
- Lightweight
- Ideal for small GUI applications

2. OpenWeatherMap API Used

- Free
- Easy-to-use endpoint
- Accurate global weather coverage

3. JSON Response Handling

- JSON is lightweight
- Easy to parse using Python’s built-in functions

4. Time Conversion

- Required converting UNIX timestamp to local readable time
- Python’s `time` module used

5. Entry Box + Enter Key Handling

- Makes application fast and intuitive

9. Implementation Details

Programming Language: Python 3.x

GUI Library: Tkinter

API Used: https://api.openweathermap.org

Data Format: JSON

Step-by-step Implementation:

```
import tkinter as tk
```

import requests

import time

2. Create Main Window

- Set title
- Set size
- Apply fonts

3. Create Entry Field

User types city here.

4. Bind Enter Key

The app reacts instantly when the user presses Enter.

5. API Request & JSON Parsing

Use:

```
requests.get(api).json()
```

6. Extract Required Data

- temperature
- humidity
- pressure
- wind
- sunrise
- sunset

7. Convert Temperature

Kelvin → Celsius.

8. Display Using Labels

Two labels show main info and details.

---

10. Screenshots / Results

(Add your screenshots here)

- Screenshot of main window
- Screenshot after entering a city
- Error-handling screen (optional)

---

11. Testing Approach

Test Cases

Test No	Input	Expected Output	Actual Result	Status
1	Mumbai	Weather data shown	Pass	✓
2	Empty string	No crash	Pass	✓
3	Random text	Error handled	Pass	✓

4	Slow internet	Delayed result	Pass	✓
5	Repeated searches	Updated correctly	Pass	✓

Testing Methods Used

- Functional testing
- GUI testing
- Boundary value testing
- Error-handling test

---

12. Challenges Faced

1. Understanding how to use API keys
2. JSON structure was initially confusing
3. Formatting GUI properly
4. Converting UNIX timestamp
5. Handling invalid city inputs
6. Ensuring the app doesn’t freeze during API calls

---

13. Learnings & Key Takeaways

- How to integrate an external API in Python
- Basics of GUI development with Tkinter
- Understanding real-time data parsing
- Error handling and user input validation
- Practical experience with JSON
- Improved Python coding skills
- Understanding event-driven programming

---

14. Future Enhancement

- Add 7-day weather forecast
- Add weather graphs
- Add weather icons dynamically
- Add search history
- Add voice-based city input
- Convert into Android app using Kivy
- Notify user about weather changes
- Add map integration