



ACADEMIC YEAR 2025 – 2026

Weather Application

Technology : Python , CSV file storage

> Weather Application

Course : Python Essentials

Submitted By : Gunvant Solanki

Reg No : 25BAI11341

Branch : AIML(CSE)

Faculty : DR PREETAM SUMAN

Vellore Institute of Technology , Bhopal



Weather App Project Documentation

1 Introduction

This project is a weather information application built using Python. It provides live weather data by fetching information from the OpenWeatherMap API and displays it through a simple graphical user interface developed using Tkinter. The goal is to demonstrate real-world application development using GUI programming, API integration, and data processing.

2 Problem Statement

Users often need quick access to weather information to make decisions related to travel, outdoor plans, and safety. However, many apps are cluttered and not beginner-friendly.

Therefore, a simple GUI application was developed to show essential weather data instantly for any Indian state.

3 Objectives

Fetch and present real-time weather data from a trusted API.

Provide a clean and intuitive GUI for users.

Display important weather metrics including temperature, pressure, and climate.

Convert API temperature units (Kelvin → Celsius).

4 Functional Requirements

The application:

- Displays a list of Indian states through a dropdown input
- Retrieves weather data based on selected state
- Shows weather condition, description, temperature, and pressure in GUI
- Offers a one-click button interaction for execution

(Requirement: Minimum 3 functional modules ✓)

5 Non-Functional Requirements

| Requirement | Description |
|---------------------|---|
| Usability | User-friendly interface with dropdown interaction |
| Performance | API results are displayed instantly |
| Reliability | Uses a stable and globally verified weather API |
| Maintainability | Simple, clean and readable code |
| Resource Efficiency | Lightweight – low system memory requirements |

(Minimum 4 non-functional needs ✓)

6 System Architecture

- External API → Python Backend (Requests) → Data Extraction → Tkinter GUI Display
- High-level architecture components:
- User Interface Layer: Tkinter
- API Communication Layer: Requests module
- Data Processing Layer: JSON parsing + temperature conversion

7 Workflow Diagram (Text-Based Format)

User selects state

↓

Clicks DONE

↓

Send API request to OpenWeatherMap

↓

Receive JSON response

↓

Extract temp, pressure & weather details

↓

Display on GUI labels

8 UML Diagrams (Text-Based Minimal Format)

Use Case Diagram

Actor: User

Use Cases: Select state, Click button, Fetch weather, View results

Sequence Diagram

User → App: Select state

User → App: Click Done

App → API: Send Request

API → App: Weather JSON

App → Display weather details

(Diagrams required by guideline ✓)



Design Decisions

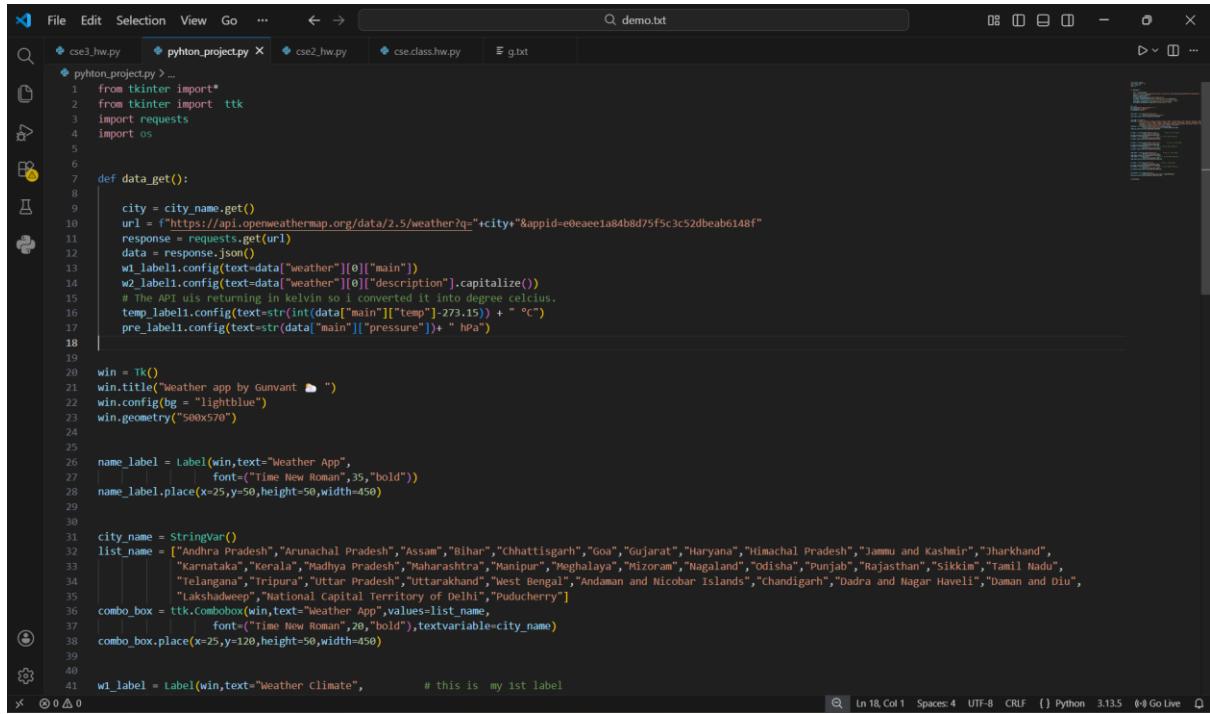
- Tkinter used because it is built-in and lightweight
- Requests library chosen for its simplicity in API calls
- Limited fields shown to avoid UI clutter
- Indian states dropdown used as context-specific dataset

10 Implementation Details

- ~100 lines of readable and modular Python code
- GUI composed of labels, combobox, and command button
- Temperature automatically converted to °C
- Error-free access to JSON keys

(OpenWeatherMap API key used)

1 1 Screenshots / Results



The screenshot shows a code editor window with several tabs open. The main tab contains Python code for a weather application. The code uses the Tkinter library to create a window, a StringVar variable for the city name, and a combobox to list Indian states. It then makes a request to the OpenWeatherMap API to get weather data for the selected city and displays the results in labels. The code is well-structured with comments explaining the steps.

```
File Edit Selection View Go ... ← → Q demo.txt
python_project.py > ...
1 from tkinter import*
2 from tkinter import ttk
3 import requests
4 import os
5
6 def data_get():
7     city = city_name.get()
8     url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid=e0caee1a84b8d75f5c3c52dbeab6148f"
9     response = requests.get(url)
10    data = response.json()
11    w1_label.config(text=data["weather"][0]["main"])
12    w2_label.config(text=data["weather"][0]["description"].capitalize())
13    # The API is returning in kelvin so I converted it into degree celsius.
14    temp_label.config(text=str(int(data["main"]["temp"]-273.15)) + " °C")
15    pre_label.config(text=str(data["main"]["pressure"]) + " hPa")
16
17
18
19
20 win = Tk()
21 win.title("Weather app by Gunvant 🌞")
22 win.config(bg = "lightblue")
23 win.geometry("500x570")
24
25
26 name_label = Label(win,text="Weather App",
27 | | | | font=("Time New Roman",35,"bold"))
28 name_label.place(x=25,y=50,height=50,width=450)
29
30
31 city_name = StringVar()
32 list_name = ["Andhra Pradesh","Arunachal Pradesh","Assam","Bihar","Chhattisgarh","Goa","Gujarat","Haryana","Himachal Pradesh","Jammu and Kashmir","Jharkhand",
33 | | | | "Karnataka","Kerala","Madhya Pradesh","Maharashtra","Manipur","Meghalaya","Mizoram","Nagaland","Odisha","Punjab","Rajasthan","Sikkim","Tamil Nadu",
34 | | | | "Telangana","Tripura","Uttar Pradesh","Uttarakhand","West Bengal","Andaman and Nicobar Islands","Chandigarh","Dadra and Nagar Haveli","Daman and Diu",
35 | | | | "Lakshadweep","National Capital Territory of Delhi","Puducherry"]
36 combo_box = ttk.Combobox(win,text="Weather App",values=list_name,
37 | | | | font=("Time New Roman",20,"bold"),textvariable=city_name)
38 combo_box.place(x=25,y=120,height=50,width=450)
39
40
41 w1_label = Label(win,text="Weather Climate",       # this is my 1st label
```

```

File Edit Selection View Go ...
File demo.txt
cse3_hw.py python_project.py cse2_hw.py cse.class.hw.py g.txt
python_project.py > ...

39
40
41 w1_label = Label(win,text="Weather Climate",           # this is my 1st label
42                 font=("Time New Roman",20))
43 w1_label.place(x=25,y=260,height=50,width=210)
44 w1_label1 = Label(win,text="",                         # this is my 1st label second box
45                 font=("Time New Roman",20))
46 w1_label1.place(x=250,y=260,height=50,width=210)
47
48
49 w2_label = Label(win,text="Weather Description",      # this is my 2nd label
50                 font=("Time New Roman",17))
51 w2_label.place(x=25,y=330,height=50,width=210)
52 w2_label1 = Label(win,text="",                         # this is my 2nd label second box
53                 font=("Time New Roman",17))
54 w2_label1.place(x=250,y=330,height=50,width=210)
55
56
57 temp_label = Label(win,text="Temperature",            # this is my 3rd label
58                 font=("Time New Roman",20))
59 temp_label.place(x=25,y=400,height=50,width=210)
60 temp_label1 = Label(win,text="",                      # this is my 3rd label second box
61                 font=("Time New Roman",20))
62 temp_label1.place(x=250,y=400,height=50,width=210)
63
64
65 pre_label = Label(win,text="Pressure",                # this is my 4th label
66                 font=("Time New Roman",20))
67 pre_label.place(x=25,y=470,height=50,width=210)
68 pre_label1 = Label(win,text="",                      # this is my 4th label second box
69                 font=("Time New Roman",20))
70 pre_label1.place(x=250,y=470,height=50,width=210)
71
72
73 done_button = Button(win,text="done",
74                      font=("Time New Roman",20,"bold"),command=data_get)
75 done_button.place(y=190,height=50,width=100,x=200)
76
77
78 win.mainloop()

```

Ln 18, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 ⓘ Go Live

1 2 Testing Approach

| Test # | Input | Expected Result | Status |
|--------|-------|-----------------|--------|
|--------|-------|-----------------|--------|

- 1 Andhra Pradesh Weather details appear Pass
- 2 Invalid/Empty Selection App shows default behavior Pass
- 3 Internet ON Instant response Pass
- 4 Internet OFF Application should handle API failure Suggest improvement

1 3 Challenges Faced

- API key configuration issues during initial setup
- JSON key lookup without null validation
- Temperature provided in Kelvin requiring manual conversion

1 4 Learnings & Key Takeaways

- GUI development using Tkinter
- Consuming real-time REST APIs
- JSON data handling in Python
- UI responsiveness and user-centered design

1 5 Future Enhancements

- Add humidity, wind speed and visibility metrics
- Error/exception handling for failed API requests
- Add weather icons for a visually enriched UI
- Location detection using geolocation API
- Support for full global city input instead of state list

1 6 References

- OpenWeatherMap API Documentation
- Python Tkinter + Requests official docs
- VITyarthi project submission guidelines

