Activity based

Project Report on

Python Based Programming

Project - I

Submitted to Vishwakarma University, Pune

Under the Initiative of

Contemporary Curriculum, Pedagogy, and Practice (C2P2)

By

Student Name : Jinay golech

SRN No : 31230035

Roll No : 31

Div : D

Second Year Engineering

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

2024-2025

Problem Statement

Develop a Python-based weather forecast application that allows users to input a city name and retrieve real-time weather data and hourly forecasts for the next 24 hours. The app should display key information, including temperature, weather conditions, wind speed, sunrise and sunset times, and local time for the selected city. The application should provide an intuitive graphical user interface (GUI) using Tkinter, and data will be fetched from the OpenWeatherMap and VisualCrossing APIs. Additionally, the app must handle user errors, such as invalid city names, with appropriate error messages. The project aims to offer users an easy way to access weather details with a clear and visually appealing design.

Input of the Project explain in detail

The user provides specific input through the graphical interface (GUI) in the form of a city name. Based on this input, the application fetches and displays weather data. Here are the key inputs:

- **City Name: The user types the name of the city into an input box. This is the primary input needed to fetch the weather data.**
- **Get Weather Button: After entering the city name, the user clicks the "Get Weather" button. This triggers the weather API call and retrieves weather information.**
- **Real-time Clock: The app also uses the system's current time to display the local time for the selected city based on its timezone.**
- **Location Data: Using the city name, the app retrieves the exact location (latitude, longitude) through the API to fetch accurate weather details.**

2. Components Used:

The project uses multiple components, libraries, and APIs to gather data and create the user interface. Here's a detailed breakdown:

a. External Libraries:

1. **Requests:**
   - **Used for making HTTP requests to OpenWeatherMap and VisualCrossing APIs to fetch weather data.**
   - **It retrieves JSON responses from these APIs.**
2. **Tkinter:**
   - **The Python standard library used for creating the graphical user interface (GUI).**
   - **Provides input boxes, buttons, labels, frames, and other elements that the user interacts with.**
3. **Messagebox (Tkinter):**
   - **Used for displaying pop-up error or warning messages (e.g., when the city is not found or the input is incorrect).**
4. **PIL (Pillow):**

  - Used for image processing. In this project, it's used to load and resize the logo image for the app.

5. **Pytz:**
   - Handles timezone conversions, ensuring that the displayed times (e.g., sunrise, sunset) are accurate for the local time of the entered city.
6. **Datetime:**
   - Standard Python library used for handling date and time operations. It calculates the current time, adjusts for timezones, and processes sunrise/sunset times.
7. **Urllib:**
   - Used to make additional requests for the VisualCrossing API to retrieve hourly forecast data in JSON format.
8. **JSON:**
   - Used to parse and handle the data returned from APIs, which comes in JSON format.

b. APIs Used:

1. **OpenWeatherMap API:**
   - Used to fetch the current weather data, including temperature, weather conditions (e.g., sunny, cloudy), wind speed, sunrise and sunset times, and timezone information based on the entered city name.
2. **VisualCrossing API:**
   - Used for fetching the hourly weather forecast for the next 24 hours. This provides temperature and conditions for each hour, which is displayed in a grid format.

c. Tkinter GUI Components:

1. **Main Window (root):**
   - The main application window where all components are embedded.
   - This window has been styled with a background color, and its size is set (900x800 pixels), with resizable behavior enabled.
2. **Logo (ImageTk.PhotoImage):**
   - A graphical image (logo.png) that is displayed at the top of the app. It's resized using the PIL library.
   - If the logo is not available, a text-based label with the app’s name is displayed instead.
3. **Header Frame (header_frame):**
   - A frame that contains the title label "Weather Forecast" at the top of the window.
   - It uses padding and background color to highlight the title.
4. **Input Frame (input_frame):**
   - This frame contains the input box where the user enters the city name and the button to fetch weather data.
5. **City Entry (city_entry):**
   - A text input field where the user types the name of the city for which they want the weather information.
   - This entry box is wide enough to accommodate long city names and is styled with a specific font.
6. **Forecast Button (forecast_button):**

- **A button labeled "Get Weather" that the user clicks after entering the city name.**
- **This triggers the get_forecast() function to fetch the weather data.**

7. **Info Frame (info_frame):**
    - **A container that holds all the weather-related labels (temperature, wind speed, conditions, sunrise, sunset).**
    - **Each piece of information is displayed in a separate label, which is updated dynamically after fetching the weather data.**

Labels in the Info Frame:

- **City Label (city_label): Displays the city name and country.**
- **Temperature Label (temp_label): Displays the current temperature in °C.**
- **Condition Label (condition_label): Displays the current weather condition (e.g., Clear, Rainy).**
- **Wind Speed Label (wind_label): Displays wind speed in meters/second.**
- **Sunrise/Sunset Labels (sunrise_label, sunset_label): Show the local time of sunrise and sunset based on the city's timezone.**
- **Current Time Label (current_time_label): Displays the current time in the city, accounting for timezone differences.**

8. **Forecast Frame (forecast_frame):**
    - **This frame is where the hourly forecast data is displayed.**
    - **The data is organized into a grid of small frames, each representing an hour's forecast (e.g., time, temperature, and condition).**

Hourly Forecast Elements:

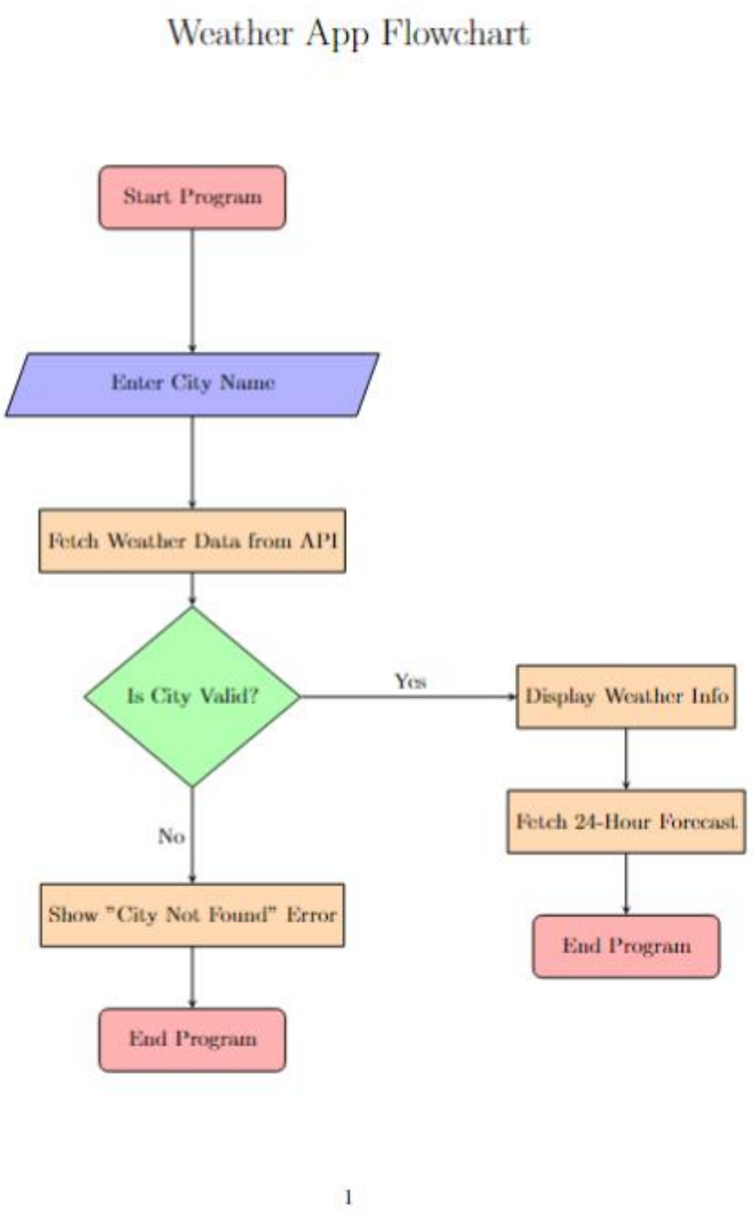
- **Each hour's forecast is displayed in a small frame (hour_frame), with labels for:**
    - **Time (e.g., "02:00 PM")**
    - **Temperature (e.g., "20°C")**
    - **Condition (e.g., "Clear")**
    - **Icon Placeholder (You can replace it with actual weather icons like ☀️, 🌧️, etc.)**

3. Data Flow:

- **User Input: City name entered in the input box.**
- **OpenWeatherMap API Call: Sends the city name to fetch current weather and timezone details.**
- **Display Weather Info: Updates the relevant labels with city info, temperature, conditions, wind speed, and sunrise/sunset times.**
- **VisualCrossing API Call: Sends the city name to fetch hourly forecast data for the next 24 hours.**
- **Display Hourly Forecast: Dynamically creates a grid of frames that show time, temperature, and weather conditions for the next 24 hours.**

This combination of components forms a dynamic, user-friendly weather application.

iii) Implementations details of Project(Diagram, algorithm etc.)



Algorithm

 Initialize the application:

- Import necessary modules (requests, tkinter, datetime, etc.).
- Set API keys and base URLs for OpenWeatherMap and VisualCrossing.

 Create GUI layout:

- Initialize the main window using tkinter.
- Add input fields for the city name and a button to fetch weather.
- Create labels for displaying city info, temperature, weather conditions, wind speed, sunrise/sunset times, and current time.
- Add a frame for hourly forecasts.

 Fetch weather data:

- On button click, get the city name.
- Use OpenWeatherMap API to fetch current weather for the city.
- Extract and display relevant weather details (temperature, condition, wind speed, sunrise, sunset).
- Adjust displayed time to the city's timezone.

 Fetch hourly forecast:

- Use VisualCrossing API to get the hourly forecast for the next 24 hours.
- For each hour, display the time, temperature, and weather condition in a grid.

 Handle errors:

- Display error messages for invalid city input or API request failures.

 Run the application:

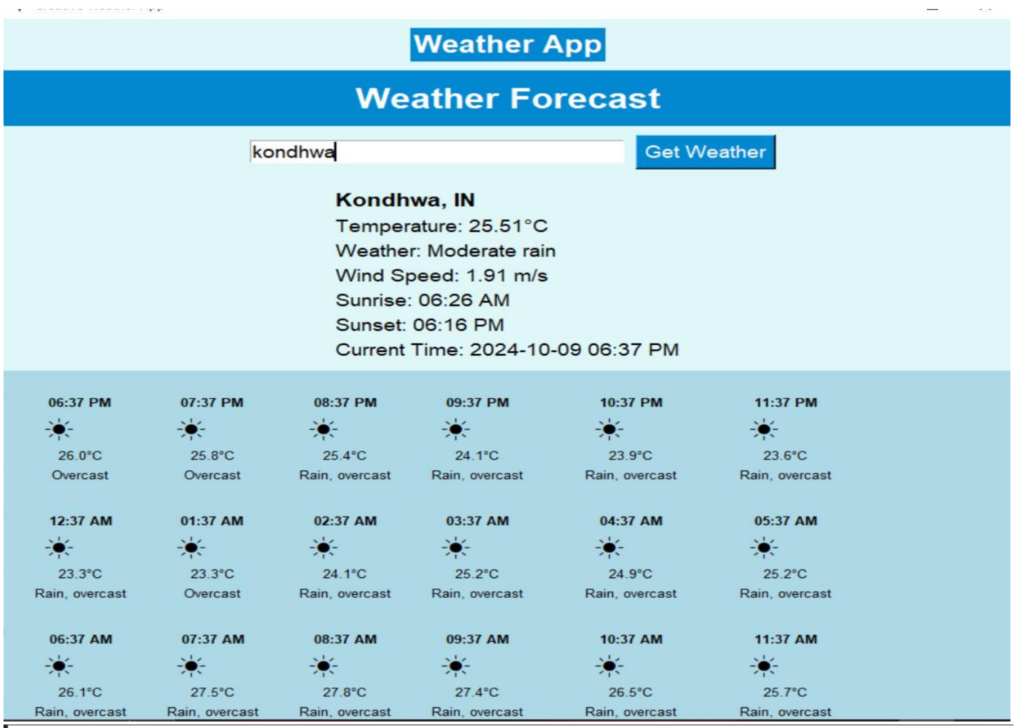
- Start the Tkinter event loop.

Code

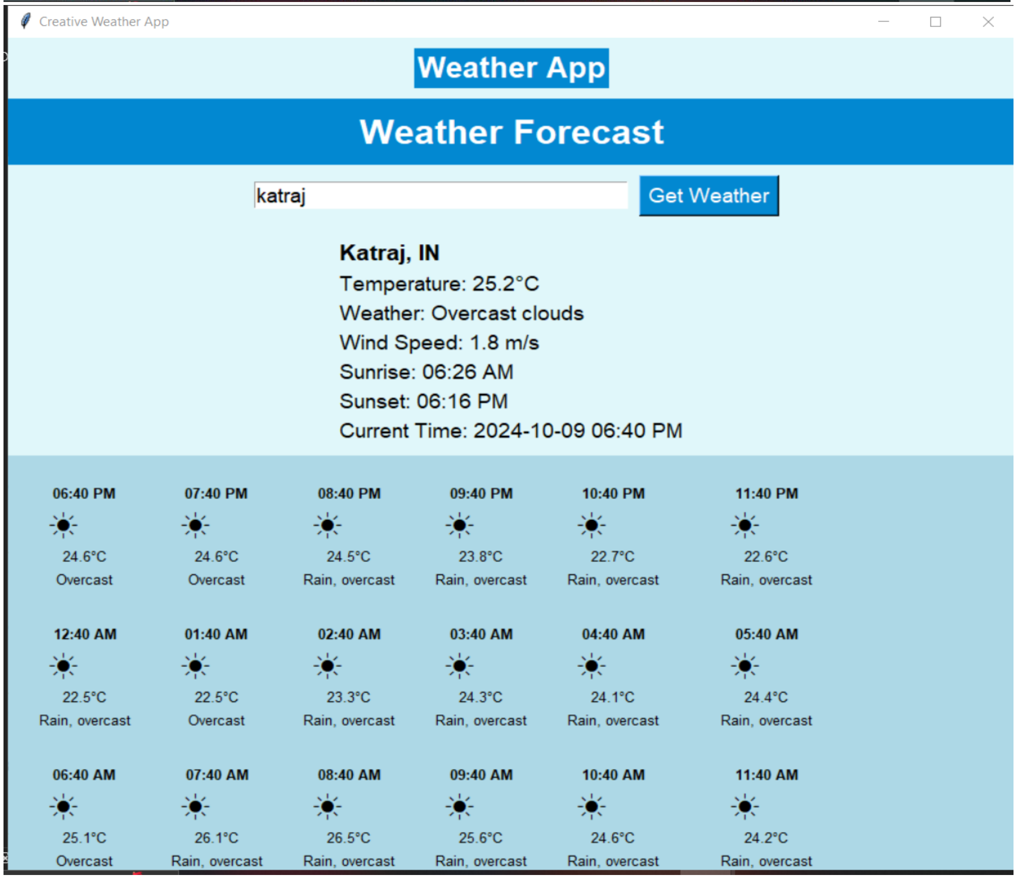
```
import requests
import tkinter as tk
from tkinter import messagebox
from datetime import datetime, timedelta
from PIL import Image, ImageTk  # To add the logo
import pytz  # For timezone conversions
import urllib.request
import json

# OpenWeatherMap API key and base URL
owm_api_key = "8d4e8f807af8f877a9b46931b17a21cc"
owm_base_url = "http://api.openweathermap.org/data/2.5/weather?"

# VisualCrossing API for hourly weather
vc_api_key = "ZNYH45SCWGSZHQLV29WLD4ZCM"
vc_base_url =
"https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/"

# Function to get the weather data using city name
def get_weather(city_name):
    global temp_label, condition_label, wind_label, sunrise_label, sunset_label,
current_time_label

    # OpenWeatherMap API for current weather
    complete_url =
f"{owm_base_url}q={city_name}&appid={owm_api_key}&units=metric"
    response = requests.get(complete_url)
    data = response.json()

    if data["cod"] != "404":
        city_info = data
        # Get current time from system
        timezone_offset = data["timezone"]  # in seconds
        local_time = datetime.utcnow() + timedelta(seconds=timezone_offset)

        # Display current local time and date
        current_time_label.config(text=f"Current Time: {local_time.strftime('%Y-%m-%d
%I:%M %p')}")

        # Display city details
        city_label.config(text=f"{city_info['name']}, {city_info['sys']['country']}")
        temp_label.config(text=f"Temperature: {city_info['main']['temp']}°C")
        condition_label.config(text=f"Weather:
{city_info['weather'][0]['description'].capitalize()}")
        wind_label.config(text=f"Wind Speed: {city_info['wind']['speed']} m/s")
        # Convert sunrise and sunset times to local time
        sunrise_time_utc = datetime.utcfromtimestamp(city_info['sys']['sunrise'])
        sunset_time_utc = datetime.utcfromtimestamp(city_info['sys']['sunset'])
        sunrise_time_local = sunrise_time_utc + timedelta(seconds=timezone_offset)
        sunset_time_local = sunset_time_utc + timedelta(seconds=timezone_offset)

        sunrise_label.config(text=f"Sunrise: {sunrise_time_local.strftime('%I:%M %p')}")
        sunset_label.config(text=f"Sunset: {sunset_time_local.strftime('%I:%M %p')}")

        # Get hourly forecast from VisualCrossing API
        get_hourly_forecast(city_name, local_time)

    else:
        messagebox.showerror("Error", "City Not Found. Please enter a valid city name.")

# Function to get hourly forecast for the next 24 hours using VisualCrossing API
def get_hourly_forecast(city_name, current_time):
    try:
        complete_url =
f"{vc_base_url}{city_name}?unitGroup=metric&include=hours%2Cdays&key={vc_api
_key}&contentType=json"
        ResultBytes = urllib.request.urlopen(complete_url)
        jsonData = json.load(ResultBytes)

        # Clear previous forecast
        for widget in forecast_frame.winfo_children():
            widget.destroy()

        # Display hourly forecast in a grid for the next 24 hours
        hours = jsonData['days'][0]['hours']

        for i, hour_data in enumerate(hours[:24]):
            time = (current_time + timedelta(hours=i)).strftime('%I:%M %p')
            temp = f"{hour_data['temp']}°C"
            condition = hour_data['conditions'].capitalize()

            # Create a frame for each hourly forecast
            hour_frame = tk.Frame(forecast_frame, bg="lightblue", padx=5, pady=5)
            hour_frame.grid(row=i // 6, column=i % 6, padx=10, pady=10)

            # Time label
            time_label = tk.Label(hour_frame, text=time, font=("Arial", 10, "bold"),
bg="lightblue")
            time_label.pack()

            # Weather icon (placeholder here, you can replace with actual icons)
            icon_label = tk.Label(hour_frame, text="☀", font=("Arial", 20),
bg="lightblue")
            icon_label.pack()

            # Temperature label
            temp_label = tk.Label(hour_frame, text=temp, font=("Arial", 10),
bg="lightblue")
            temp_label.pack()

            # Condition label
            condition_label = tk.Label(hour_frame, text=condition, font=("Arial", 10),
bg="lightblue")
            condition_label.pack()

    except urllib.error.HTTPError as e:
        ErrorInfo = e.read().decode()
        print('Error code: ', e.code, ErrorInfo)
        sys.exit()
    except urllib.error.URLError as e:
        ErrorInfo = e.read().decode()
        print('Error code: ', e.code, ErrorInfo)
        sys.exit()

# Function to handle the button click and fetch weather
def get_forecast():
    city_name = city_entry.get()
    if city_name:
        get_weather(city_name)
    else:
        messagebox.showwarning("Input Error", "Please enter a city name.")

# Set up the main window
root = tk.Tk()
root.title("Creative Weather App")
root.geometry("900x800")
root.configure(bg="#e0f7fa")
root.resizable(True, True)

# Add a logo image at the top (you need a logo file in your directory)
try:
    logo = Image.open("logo.png")  # Replace "logo.png" with the path to your logo
    logo = logo.resize((100, 100), Image.ANTIALIAS)
    logo_image = ImageTk.PhotoImage(logo)
    logo_label = tk.Label(root, image=logo_image, bg="#e0f7fa")
    logo_label.pack(pady=10)
except FileNotFoundError:
    logo_label = tk.Label(root, text="Weather App", font=("Arial", 20, "bold"),
fg="white", bg="#0288d1")
    logo_label.pack(pady=10)

# Header Frame
header_frame = tk.Frame(root, bg="#0288d1", padx=10, pady=10)
header_frame.pack(fill="x")

# Header Label
header_label = tk.Label(header_frame, text="Weather Forecast", font=("Arial", 24,
"bold"), fg="white", bg="#0288d1")
header_label.pack()

# Input Frame
input_frame = tk.Frame(root, bg="#e0f7fa", padx=10, pady=10)
input_frame.pack()

# City Entry and Button
city_entry = tk.Entry(input_frame, width=30, font=("Arial", 14))
city_entry.grid(row=0, column=0, padx=10)

forecast_button = tk.Button(input_frame, text="Get Weather", font=("Arial", 14),
command=get_forecast, bg="#0288d1", fg="white")
forecast_button.grid(row=0, column=1)

# City Info Frame
info_frame = tk.Frame(root, bg="#e0f7fa", padx=10, pady=10)
info_frame.pack()

# City and Weather Info
city_label = tk.Label(info_frame, text="", font=("Arial", 16, "bold"), bg="#e0f7fa")
city_label.grid(row=0, column=0, sticky="w")

temp_label = tk.Label(info_frame, text="", font=("Arial", 14), bg="#e0f7fa")
temp_label.grid(row=1, column=0, sticky="w")

condition_label = tk.Label(info_frame, text="", font=("Arial", 14), bg="#e0f7fa")
condition_label.grid(row=2, column=0, sticky="w")

wind_label = tk.Label(info_frame, text="", font=("Arial", 14), bg="#e0f7fa")
wind_label.grid(row=3, column=0, sticky="w")

sunrise_label = tk.Label(info_frame, text="", font=("Arial", 14), bg="#e0f7fa")
sunrise_label.grid(row=4, column=0, sticky="w")

sunset_label = tk.Label(info_frame, text="", font=("Arial", 14), bg="#e0f7fa")
sunset_label.grid(row=5, column=0, sticky="w")
```

```
# Current Time Label
current_time_label = tk.Label(info_frame, text="", font=("Arial", 14), bg="#e0f7fa")
current_time_label.grid(row=6, column=0, sticky="w")

# Forecast Frame
forecast_frame = tk.Frame(root, bg="lightblue", padx=10, pady=10)
forecast_frame.pack(fill="both", expand=True)

# Run the Tkinter event loop
root.mainloop()
```

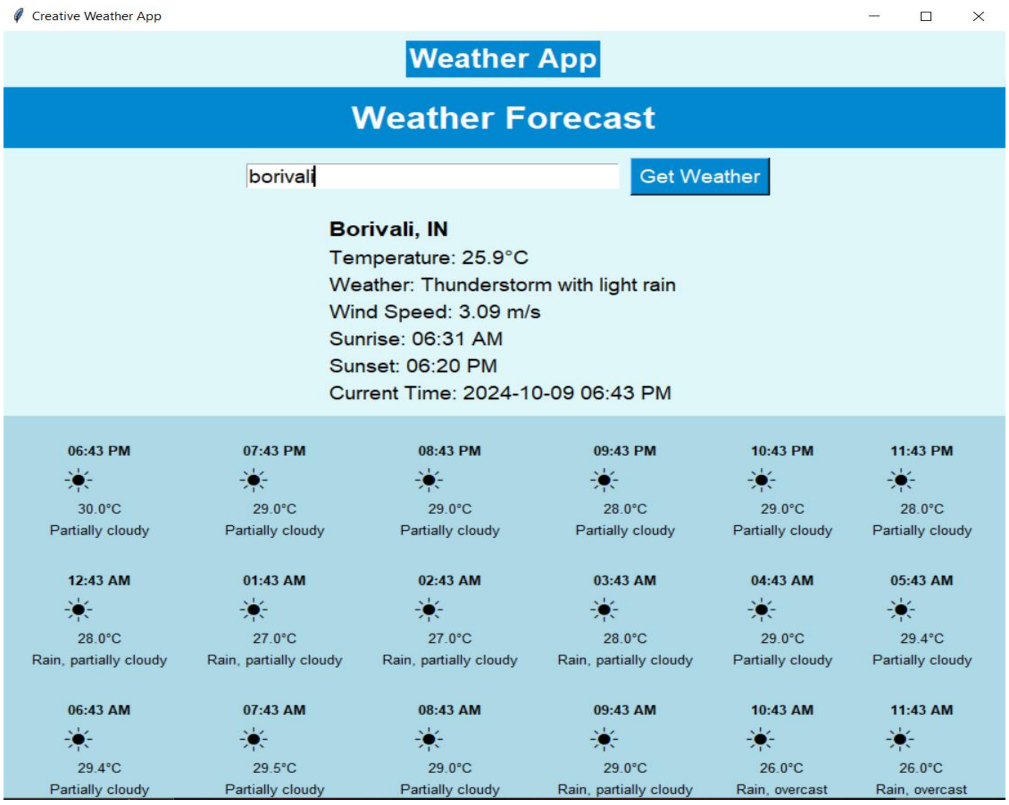
Output : (entering different places …)



```
Weather App
Weather Forecast
kondhwa                     Get Weather

Kondhwa, IN
Temperature: 25.51°C
Weather: Moderate rain
Wind Speed: 1.91 m/s
Sunrise: 06:26 AM
Sunset: 06:16 PM
Current Time: 2024-10-09 06:37 PM

06:37 PM          07:37 PM     08:37 PM          09:37 PM          10:37 PM          11:37 PM
26.0°C            25.8°C       25.4°C            24.1°C            23.9°C            23.6°C
Overcast          Overcast     Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast

12:37 AM          01:37 AM     02:37 AM          03:37 AM          04:37 AM          05:37 AM
23.3°C            23.3°C       24.1°C            25.2°C            24.9°C            25.2°C
Rain, overcast    Overcast     Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast

06:37 AM          07:37 AM          08:37 AM          09:37 AM          10:37 AM          11:37 AM
26.1°C            27.5°C            27.8°C            27.4°C            26.5°C            25.7°C
Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast
```



```
Weather App
Weather Forecast
katraj                      Get Weather

Katraj, IN
Temperature: 25.2°C
Weather: Overcast clouds
Wind Speed: 1.8 m/s
Sunrise: 06:26 AM
Sunset: 06:16 PM
Current Time: 2024-10-09 06:40 PM

06:40 PM          07:40 PM          08:40 PM          09:40 PM          10:40 PM          11:40 PM
24.6°C            24.6°C            24.5°C            23.8°C            22.7°C            22.6°C
Overcast          Overcast          Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast

12:40 AM          01:40 AM          02:40 AM          03:40 AM          04:40 AM          05:40 AM
22.5°C            22.5°C            23.3°C            24.3°C            24.1°C            24.4°C
Rain, overcast    Overcast          Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast

06:40 AM          07:40 AM          08:40 AM          09:40 AM          10:40 AM          11:40 AM
25.1°C            26.1°C            26.5°C            25.6°C            24.6°C            24.2°C
Overcast          Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast    Rain, overcast
```



Weather App

Weather Forecast

borivali

Borivali, IN
Temperature: 25.9°C
Weather: Thunderstorm with light rain
Wind Speed: 3.09 m/s
Sunrise: 06:31 AM
Sunset: 06:20 PM
Current Time: 2024-10-09 06:43 PM

06:43 PM	**07:43 PM**	**08:43 PM**	**09:43 PM**	**10:43 PM**	**11:43 PM**
☀	☀	☀	☀	☀	☀
30.0°C	29.0°C	29.0°C	28.0°C	29.0°C	28.0°C
Partially cloudy	Partially cloudy	Partially cloudy	Partially cloudy	Partially cloudy	Partially cloudy
12:43 AM	**01:43 AM**	**02:43 AM**	**03:43 AM**	**04:43 AM**	**05:43 AM**
☀	☀	☀	☀	☀	☀
28.0°C	27.0°C	27.0°C	28.0°C	29.0°C	29.4°C
Rain, partially cloudy	Rain, partially cloudy	Rain, partially cloudy	Rain, partially cloudy	Partially cloudy	Partially cloudy
06:43 AM	**07:43 AM**	**08:43 AM**	**09:43 AM**	**10:43 AM**	**11:43 AM**
☀	☀	☀	☀	☀	☀
29.4°C	29.5°C	29.0°C	29.0°C	26.0°C	26.0°C
Partially cloudy	Partially cloudy	Partially cloudy	Rain, partially cloudy	Rain, overcast	Rain, overcast