Name :- **Nilesh Navnath Ugale**

Reg. no. :- **25BCE10316**

Project Report: Python Weather Forecasting Application

Course: Computer Science Engineering (CSE)

Subject: Python Programming / Application Development

1. Introduction

The **Weather Forecasting Application** is a desktop-based Graphical User Interface (GUI) tool developed using Python. It allows users to search for any city worldwide and retrieve real-time weather data, including current temperature, humidity, wind speed, and a 7-day forecast.

The application utilizes the **Open-Meteo API** to fetch weather data without requiring an API key, ensuring ease of use and accessibility.

2. System Requirements

To run this application, the following system configurations and libraries are required:

- **Operating System:** Windows, macOS, or Linux.
- **Language:** Python 3.x
- **Required Libraries:**
  - tkinter (Built-in standard GUI library)
  - requests (For HTTP API calls)
  - threading (For asynchronous operations)
  - datetime (For date manipulation)
  - Pillow (PIL) (For image processing)

Installation Command:

```
pip install requests pillow
```

3. Code Structure and Analysis

The project is structured using Object-Oriented Programming (OOP) principles, specifically using a class named WeatherApp. Below is the breakdown of the code modules.

3.1 Imports and Libraries

The top section of the code handles necessary imports.

```
import tkinter as tk
from tkinter import ttk, messagebox
import requests
import threading
from datetime import datetime
```

Explanation:

- **tkinter**: Used to create the window, frames, buttons, and labels.
- **requests**: Used to communicate with the Internet (Open-Meteo API) to get weather data.
- **threading**: Crucial for User Experience. It runs the API call in a background thread so the app doesn't "freeze" or "hang" while waiting for the internet connection.
- **datetime**: Used to convert the raw date string (e.g., "2023-10-25") into a readable format (e.g., "Wed, Oct 25").

3.2 UI Layout (setup_ui)

This method defines how the application looks. It uses the pack() geometry manager for responsive placement.

```
def setup_ui(self):
    # main container
    main = tk.Frame(self.root, bg="#1e3a8a")
    main.pack(fill="both", expand=True, padx=20, pady=20)

    # Search bar setup
    search_frame = tk.Frame(main, bg="#1e3a8a")
    # ... (Entry and Button configuration)
```

Explanation:

- Sets the background color to a deep blue (#1e3a8a) for a professional look.
- Creates input fields for the City Name.
- Binds the "Enter" key to the search function for better usability.

3.3 The Logic Core (go and fetch_stuff)

This is the "brain" of the application.

```
def go(self):
    city = self.city.get().strip()
    # Input validation
    if not city:
        messagebox.showwarning("oops", "gotta type something dude")
        return
    # Start threading
    threading.Thread(target=self.fetch_stuff, args=(city,), daemon=True).start()
```

Explanation:

1. **Input Validation:** Checks if the user actually typed a city name.
2. **Threading:** calls fetch_stuff in a separate thread to keep the interface smooth.

3.4 API Integration (fetch_stuff)

This function performs a two-step API process.

```
# Step 1: Geocoding (City Name -> Latitude/Longitude)
geo =
requests.get("[https://geocoding-api.open-meteo.com/v1/search](https://geocoding-api.open
-meteo.com/v1/search)", ...

# Step 2: Weather Data (Latitude/Longitude -> Weather Info)
url = "[https://api.open-meteo.com/v1/forecast](https://api.open-meteo.com/v1/forecast)"
data = requests.get(url, params=params).json()
```

Explanation:

- **Step 1:** The app asks the Geocoding API: "Where is London?" The API responds with coordinates (e.g., Lat: 51.5, Lon: -0.12).
- **Step 2:** The app sends those coordinates to the Weather API to get temperature, humidity, and forecast data.

3.5 Displaying Data (show_current & show_forecast)

These functions take the raw JSON data from the API and update the GUI labels.

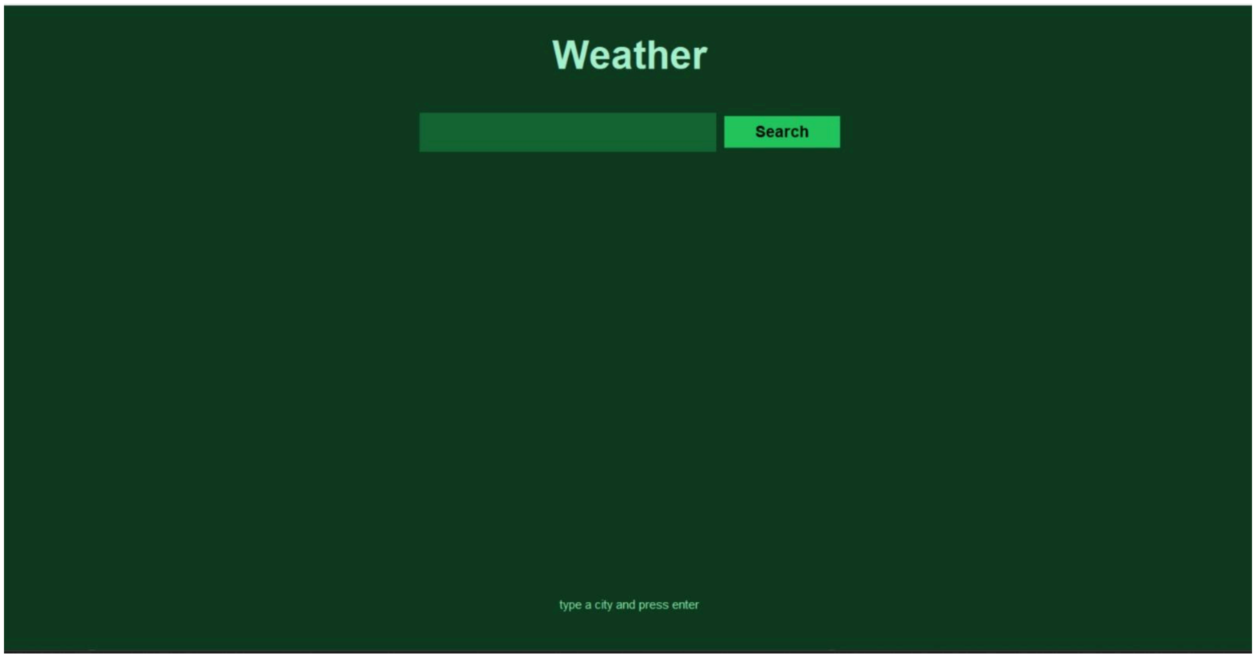
```
def show_forecast(self, daily):
    for i in range(7):
        # Create a card for each day
        day_box = tk.Frame(row, bg="#0f172a", ...)
        # Display Max/Min Temp and Precipitation
```

Explanation:

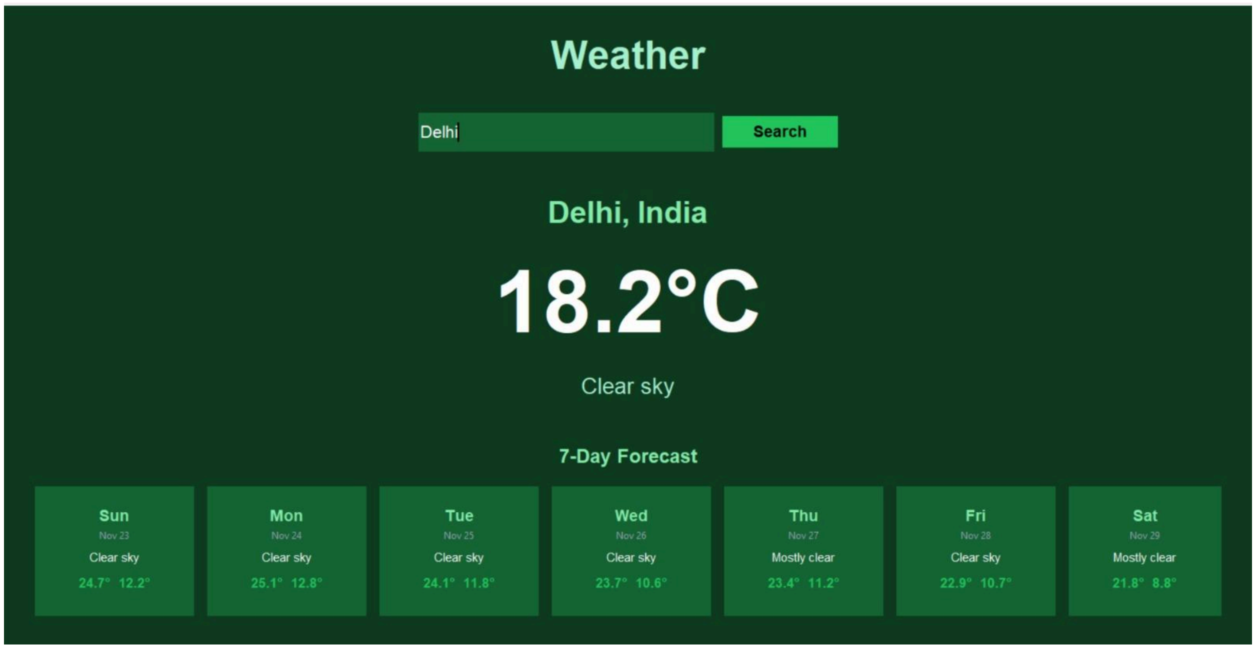
- Dynamically clears previous results using widget.destroy().
- Loops 7 times to create a forecast card for every day of the week.
- Displays high/low temperatures and rain probability.

4. Application Output

4.1 Home Screen



4.2 Search Result (Success)



Weather

Delhi, India

18.2°C

Clear sky

7-Day Forecast

Sun	Mon	Tue	Wed	Thu	Fri	Sat
Nov 23	Nov 24	Nov 25	Nov 26	Nov 27	Nov 28	Nov 29
Clear sky	Clear sky	Clear sky	Clear sky	Mostly clear	Clear sky	Mostly clear
24.7° 12.2°	25.1° 12.8°	24.1° 11.8°	23.7° 10.6°	23.4° 11.2°	22.9° 10.7°	21.8° 8.8°

5. Conclusion and Future Scope

Conclusion

This project successfully demonstrates the use of Python for creating practical desktop applications. It integrates REST APIs, manages JSON data, and handles multi-threading to ensure a smooth user experience.

Future Enhancements

- **Icon Support:** Adding weather icons (sun, cloud, rain) dynamically based on the weather code.
- **Graph Plotting:** Using matplotlib to show a temperature graph for the week.
- **Location Services:** Automatically detecting the user's location via IP address.