YASH SRIVASTAVA
25BAI10932

Python Weather Checker

A GUI Application for Real-Time Weather Data

Python • Tkinter • OpenWeatherMap API

Goal & Design Philosophy

Core Objectives

- Establish reliable connection to a third-party API (OpenWeatherMap).
- Develop a responsive Graphical User Interface (GUI) using Tkinter.
- Ensure robust error handling for API and network failures.
- Provide accurate, localized weather data on demand.

The "Pink Sky" Aesthetic

```
PINK_SKY_BG = "#fff0f5"
ROSE_ACCENT = "#e91e63"
DARK_MAGENTA = "#880e4f"
```

The design uses a high-contrast, calming palette for visual distinctiveness.

System Architecture: Data Flow

[Image of a weather API data flow diagram]

1. User Input

City name entered via Tkinter Entry widget.

2. API Request

Python's `requests` module fetches JSON data from OpenWeatherMap.

3. GUI Update

Extracted data updates Tkinter Labels in the output frame.

Robust API Interaction

The `get_weather(city)` Function

- **Metric Units:** Requests temperature in Celsius (`units=metric`).
- **Timeout:** Uses a 10-second timeout to prevent application hanging.
- **Key Retrieval:** Safely extracts nested data points (e.g., `data["sys"]["country"]`).

Comprehensive Error Handling

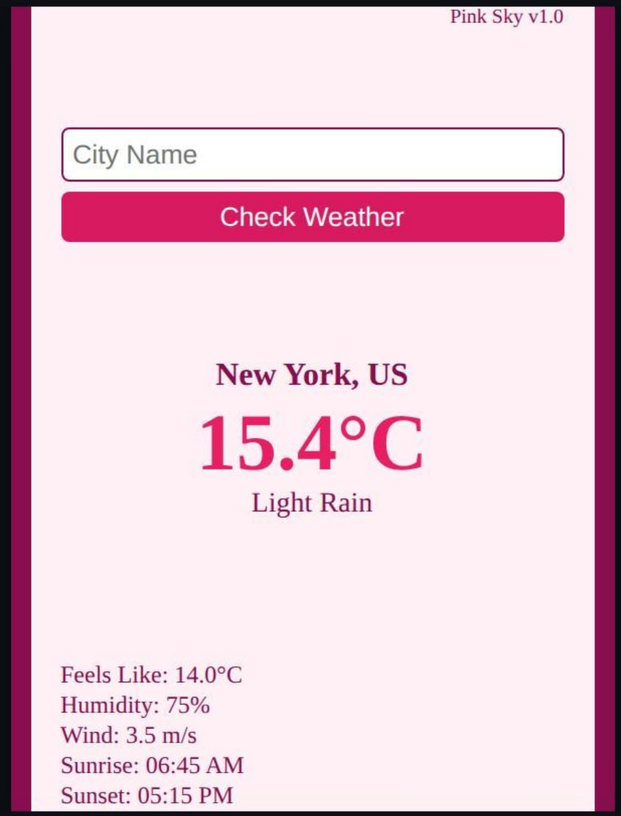
Uses Python's `try...except` structure to manage:

- `requests.exceptions.Timeout` (Network issues)
- `response.raise_for_status()` (4xx/5xx HTTP errors)
- `API-specific 404` (City not found)
- `KeyError` (Unexpected API response format)

All errors are displayed via Tkinter's `messagebox.showerror`.

Tkinter Layout and Responsiveness

- **Layout:** Uses `tk.Frame` containers for modular organization (Input Frame, Output Frame).
- **Packing:** Primarily uses `pack()` for vertical alignment and `grid()` within the input frame for horizontal control.
- **Input Bind:** The ` ` key is bound to the search function for faster user interaction.
- **Visual Identity:** All widgets utilize the pre-defined Pink Sky color variables for a consistent look.



Detailed Weather Output

The application extracts and formats several data points beyond the basic temperature.

Feels Like Temp

Contextual temperature index for user comfort.

Wind & Humidity

Essential supplementary data for daily planning.

Sunrise/Sunset

Timestamps converted from Unix time to local clock format.

Future Development Roadmap

- **Unit Conversion:** Add a toggle button to switch between Celsius and Fahrenheit/Kelvin.
- **Icon Mapping:** Implement custom weather icons (PNG/SVG) based on the API's weather condition codes instead of only text descriptions.
- **Five-Day Forecast:** Integrate the `data/2.5/forecast` API endpoint to display future weather projections.
- **Configuration File:** Move the API key and color palette definitions to a separate configuration file (e.g., .ini or YAML) for easier management.

Summary & Q&A

The project successfully integrates Python GUI development with external API data for a functional, aesthetically unique application.

Thank you.

What questions do you have?