

Weather Application in PyQt5

Szabó Róbert, August 2025

1. Introduction.....	2
2. System requirements	2
3. Application flowchart	3
4. Implementation Details.....	3
5. API Integration	4
6. Testing.....	4
7. Possible conclusion	5
8. Conclusion	5

1. Introduction

The weather application in PyQt5 is a desktop program to provide real-time weather information for a city, which the user needs to input via the application. The application communicates with OpenWeatherApp API(application programming interference) to fetch weather conditions, including temperature, a descriptive text of the weather, and a corresponding emoji representation for a more intuitive and visually appealing experience.

This project was developed in Python using the PyQt5 framework for the graphical user interface and the Requests library for handling HTTP requests to the weather API. Its primary purpose is to demonstrate the integration of GUI(*graphical user interface*) development, API consumption, and error handling in a beginner-friendly yet practical software project.

The application's main features include:

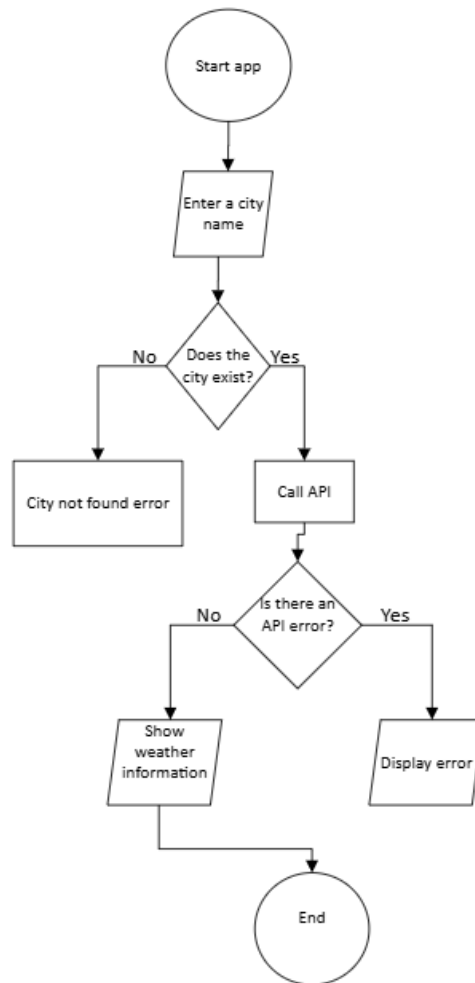
- User-friendly interface with input validation.
- Instant weather retrieval based on city name.
- Display of temperature in Celsius, textual weather description, and visual weather emoji.
- Comprehensive error handling for invalid inputs, network issues, or API-related problems.

Beyond its functionality, this project serves as an example of how Python-based applications can interact with third-party APIs to create real-time, data-driven desktop tools. It is suitable as a portfolio project, an educational example for learning PyQt5, or a base for expanding into more complex weather-related applications.

2. System requirements

The application doesn't particularly have high requirements. The application requires Python 3.10 or newer with the PyQt5 and Requests libraries installed, as well as an active internet connection to access the OpenWeatherMap API. It can run on any modern PC or laptop with at least 2 GB of RAM.

3. Application flowchart



Picture 1. *Flowchart of the application*

The flowchart illustrates the operational logic of the weather application. The process begins by launching the application, after the launch the user is prompted to enter a city name. The system first checks whether the entered city exists. If it does not, an error message is displayed, and the process ends. However if the city does exist then the application proceeds to call the OpenWeatherMap API to retrieve weather data. The system then verifies if the API request returned any errors. If an error occurs, such as a network issue or invalid API response, an error message is displayed. If there are no errors, the retrieved weather information is shown to the user. Finally, the process concludes.

4. Implementation Details

The application is developed in Python using the PyQt5 library for the graphical user interface and the Requests library to handle HTTP requests. When we start the application the program initializes the main window containing

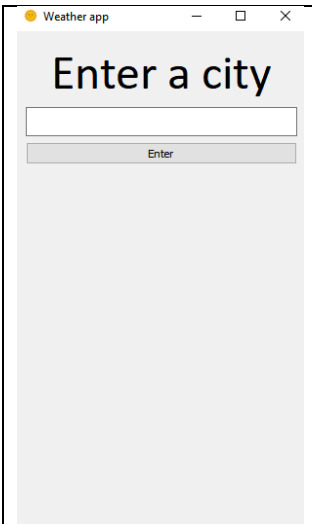
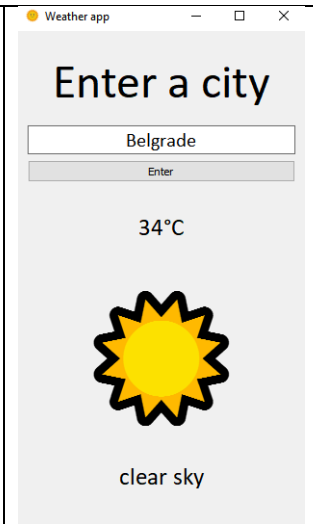
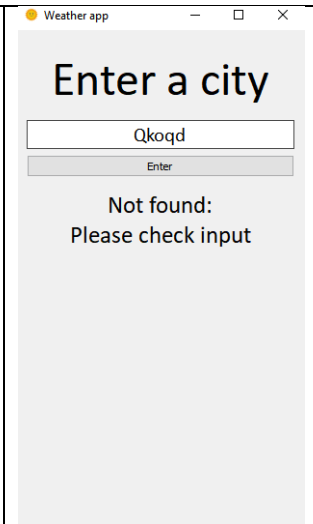

input fields, labels, and styling defined via Qt stylesheets. When the user enters a city name, the program validates the input and sends a request to the OpenWeatherMap API to retrieve current weather data in JSON format. As a valid response we should get extracted information about the temperature, weather description as well as corresponding weather emoji. If there is an invalid input, missing data or connection error then we have a error-handling routines that display appropriate messages to the user. The logic flow ensures a smooth user experience by guiding the user from data entry to the final weather display or error feedback.

5. API Integration

The application communicates with the OpenWeather API to retrieve real-time weather data. Communication is established via HTTP GET requests using the Python requests library. The API endpoint constructed based on the city name entered by the user and includes the required API key for authentication. The server responds with a JSON object containing weather details such as temperature, weather conditions, and icon codes. This JSON data is parsed and mapped to the application’s display elements, including text labels and emojis. Error handling is implemented to detect issues such as invalid city names, incorrect API keys, and network connectivity failures, ensuring the program responds with clear and informative error messages.

6. Testing

The application was tested in four scenarios: launching the app in its default state, entering a valid city name, entering an invalid city name, and running the app without an internet connection. Each case produced the expected behavior, including correct weather display for valid input and appropriate error messages for invalid input or connection issues.

			
Picture 2. <i>Default state of the application</i>	Picture 3. <i>State of the application when we enter a valid input</i>	Picture 4. <i>State of the application following a invalid input</i>	Picture 5. <i>State of the application when there is no internet connection</i>

7. Discussion and Future Enhancements

The project successfully demonstrates the integration of a graphical user interface with real-time weather data using the OpenWeatherMap API. The application performs reliably under various input scenarios, providing accurate results for valid cities and clear error messages for invalid inputs or connection issues. With minor enhancements such as improved UI adaptability and extended error handling, the tool could be further optimized for broader use. Potential enhancements include adding support for multiple languages, integrating forecast data for upcoming days, and displaying additional weather details such as humidity, wind speed, and sunrise/sunset times. Implementing caching could improve performance by reducing API calls, while refining the interface with adaptive layouts would ensure better usability on various screen sizes. Finally, packaging the application into an executable file would allow easier distribution and use without requiring a Python environment.

8. Conclusion

The weather application successfully retrieves and displays real-time weather information using the OpenWeatherMap API within a simple and user-friendly PyQt5 interface. Testing confirmed that the program responds correctly to valid and invalid inputs, as well as network connectivity issues. The project demonstrates effective integration of API data into a desktop application and provides a solid foundation for future feature expansion.

