工创创新设计Ⅰ

Proposal for Engineering Innovation Design I

姓名：彭勇智

学号：1820202027

班级：13222001

项目类型：独立项目

**Weather Dashboard with API Integration**

1. Project Objective

This project aims to delve into the practical application of Qt software, with a specific focus on leveraging Qt Designer to create an intuitive and functional graphical user interface (GUI) for a weather dashboard integrated with APIs. The primary objective is to enhance understanding of Qt’s GUI development capabilities while mastering the design of user interfaces and their integration with backend functionality using C++. Upon project completion, a fully operational interface will be developed, capable of displaying real-time weather data such as temperature, humidity, and forecasts. This endeavour will not only deepen knowledge of Qt but also strengthen practical skills in GUI design and implementation.

2.0 Specification

A weather dashboard with API integration serves as a practical tool for staying informed about current weather conditions, aiding in planning for work and outdoor activities. By utilizing APIs like OpenWeatherMap, the dashboard can retrieve real-time data, including weather descriptions, temperature, humidity, and wind conditions. Looking ahead, I aim to enhance this system with an alert feature that provides warnings for severe weather, such as heavy rain, fog, snow, or strong winds, 30 minutes to an hour in advance. This functionality would allow users to prepare for adverse conditions by taking precautions like wearing raincoats or warm clothing and carrying umbrellas. Alerts could be delivered through short messages, ensuring timely notifications. Furthermore, this dashboard could be adapted for additional purposes, integrating APIs for air quality or fire weather information, offering broader utility and improving everyday convenience.

3.0 Development Setup

This project primarily focuses on leveraging Qt Designer and Qt Creator to develop an intuitive user interface (UI) and utilize the Qt Network module to fetch weather data using an API key. By specifying a country and its corresponding zip codes, the application will display detailed weather reports for multiple cities directly on the UI. Additionally, the project aims to enhance the user experience by designing a visually appealing interface, incorporating icons, and implementing animations to reflect weather changes dynamically. If feasible, further efforts will be directed toward refining the GUI for greater elegance and usability.

Software Tools:

Qt Designer: This tool serves as the primary platform for crafting the visual components of the user interface, including buttons, text fields, and labels, through an intuitive drag-and-drop mechanism. The focus will be on mastering layout design, configuring widget properties, and establishing connections between signals and slots for seamless interaction.

Qt Creator: Following the interface design phase, Qt Creator will be employed to develop the C++ backend, enabling the integration of the UI with real-time weather data retrieval using an API key. This step ensures the application is both interactive and fully functional, bridging the gap between design and operational capabilities.

Programming Language:

C++: While the user interface is primarily developed using Qt Designer, the backend logic will be implemented in C++. This includes managing user interactions, handling API requests and responses—such as retrieving data from the OpenWeatherMap API—and processing various events. C++ ensures efficient execution of these functionalities, creating a seamless integration between the user interface and the application's core operations.

Tools for sending requests and receiving data:

OpenWeatherMap API: OpenWeather offers a free API for accessing real-time weather data for any specified location. This API aggregates and processes data from various sources, including global and regional weather models, satellites, radars, and an extensive network of weather stations. Utilizing this API allows us to fetch accurate and comprehensive weather information seamlessly.

OpenSSL: To leverage the Qt Network module and enable TLS functionality, it is crucial to install a compatible version of OpenSSL. For instance, the OpenSSL 1.1.1g Win64 package is compatible with Qt version 5.15.2. OpenSSL is a widely used software library that facilitates secure communication over networks, protecting against eavesdropping and ensuring the authentication of the communicating parties. It implements open-source versions of the SSL and TLS protocols, making it a critical component for secure data transmission in applications such as HTTPS websites and network services.

4.0 UI design

Before starting the coding process, it is advisable to draft a preliminary design of the user interface (UI). This step helps in organizing the expected layout and defining the core functionalities of the interface in advance. The key features of the UI will include the following:

* Establishing a connection to the API's URL.
* Retrieving weather data using the provided API keys.
* Processing the fetched data for meaningful use.
* Displaying the processed information clearly on the interface.

Based on these features, we created a draft of the UI.

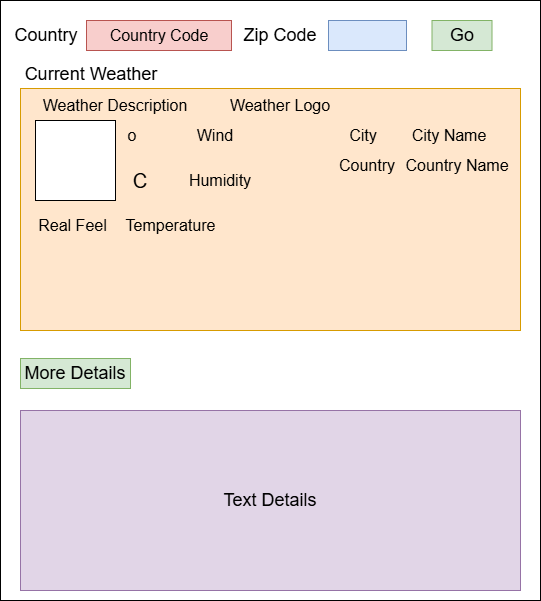


Figure 1 UI interface design draft

The design only involves a Main Window. Main Window is the weather dashboard. Users will interact with the dashboard, and the window will display all the received weather data in real time.

5.0 Implementation Details

5.1 Creating a New Project and adding the “WeatherApp” setup dialog

First, we create a new Qt Widgets project, named WeatherApp, using the qmake build system. We keep the class name as MainWindow, ensuring that a corresponding Qt Designer form is generated.

5.2 Designing the main window

Next, in Qt Designer, we proceed to design the main window based on the draft created earlier. The layout and widgets closely followed the blueprint. If possible, all widgets are renamed to reflect their functionality, making the code more readable and maintainable. At this point, the design is purely visual and no backend functionality has been implemented yet.

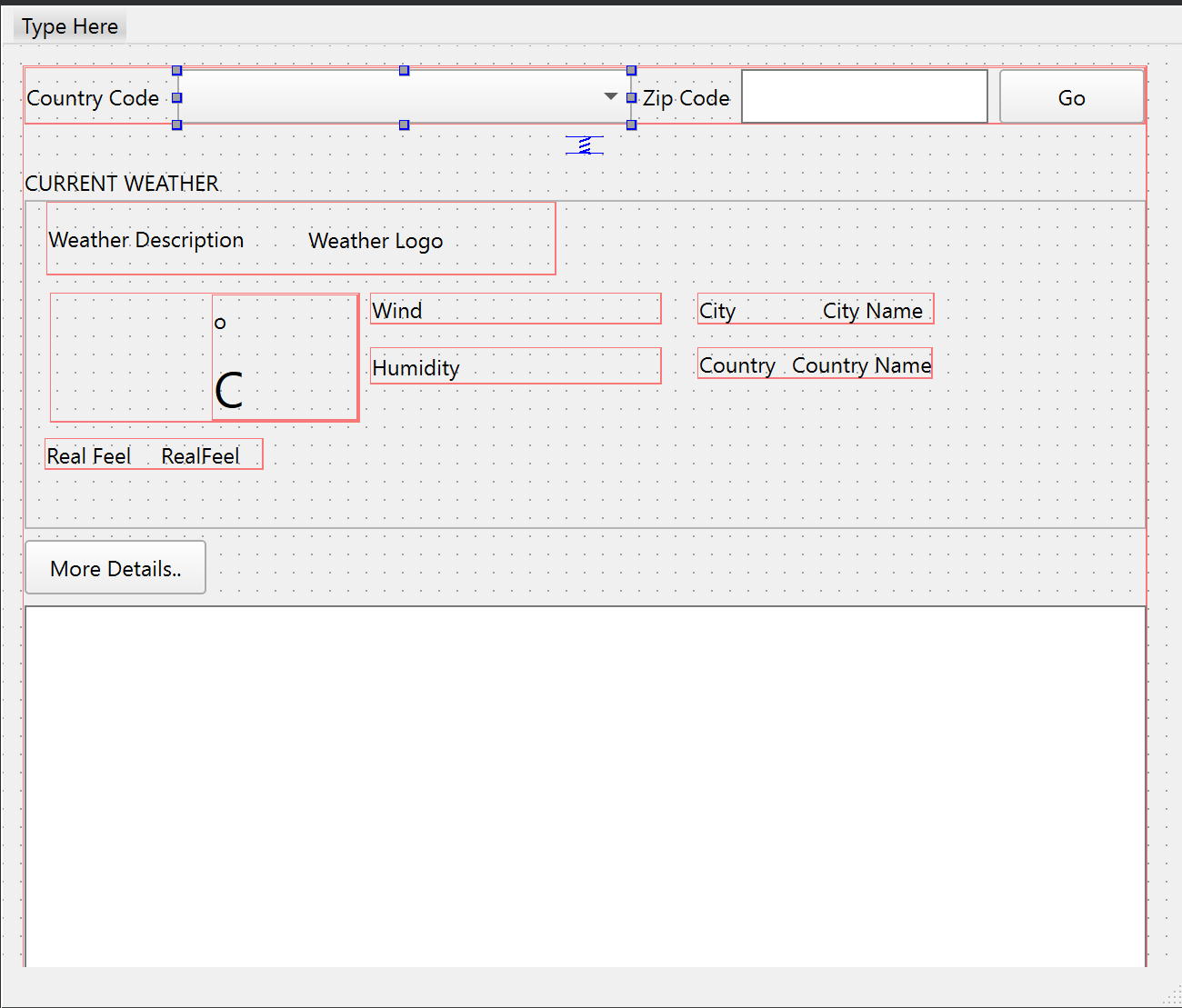


Figure 2 Main Window UI interface design

5.3 Implementing the functionalities of the Main Window

The main window is responsible for managing weather dashboard. Below are key implementations:

* Weather Data Handling: In getWeatherData section, we create the API URL with the inputs of zip code, country code, and API key. We send the requests by using the Qt Network module. Then, we connect the reply finished signal to a slot to process the result.
* Error Handling: If there is an error from the reply, this suggests that network error and sends the message “No Data Found!”.
* Receiving and Displaying Weather Data: In onWeatherDataReceived section, after obtaining the reply, we read the data and parse the JSON response. Next, we extract the data from the JSON response such as weather description, humidity and temperature. This information will then display in the UI interface successfully.

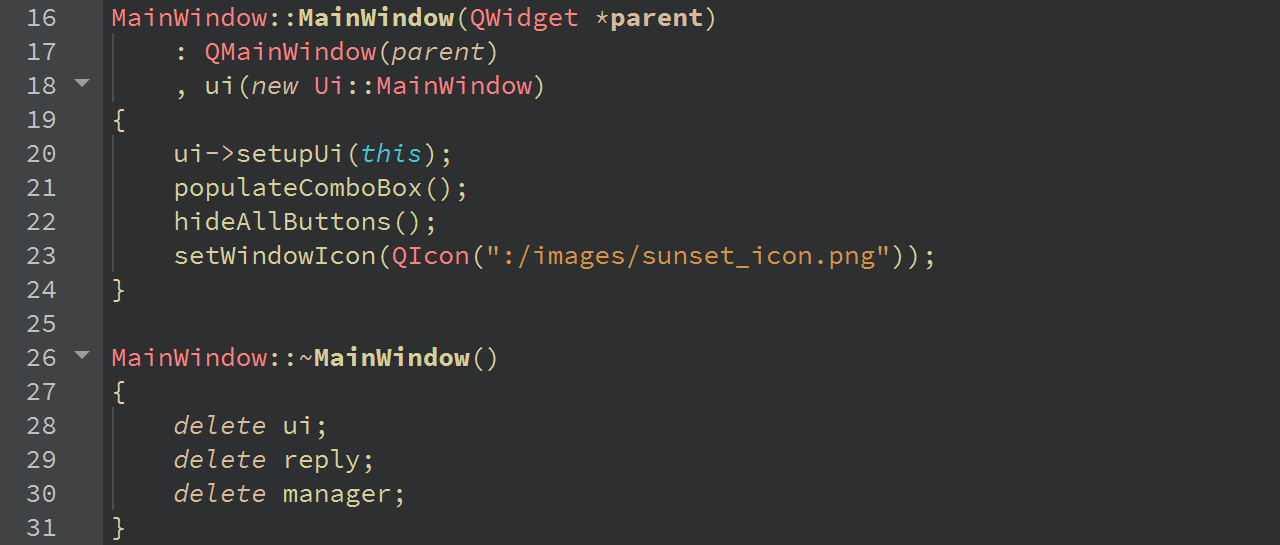


Figure 3-1 Main Window class constructor



Figure 3-2 Main Window class getWeatherData section

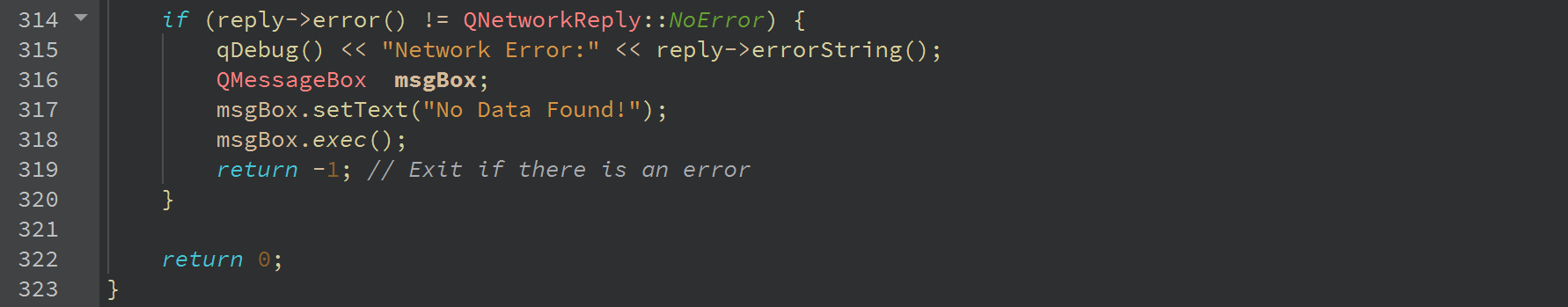


Figure 3-3 Main Window error handling



Figure 3-4 Main Window class onWeatherDataReceived section (Part 1)



Figure 3-5 Main Window class onWeatherDataReceived section (Part 2)

5.4 Additional details

* Modifications to the qmake File: We add network to the QT environment to enable the API URL for fetching the weather data.

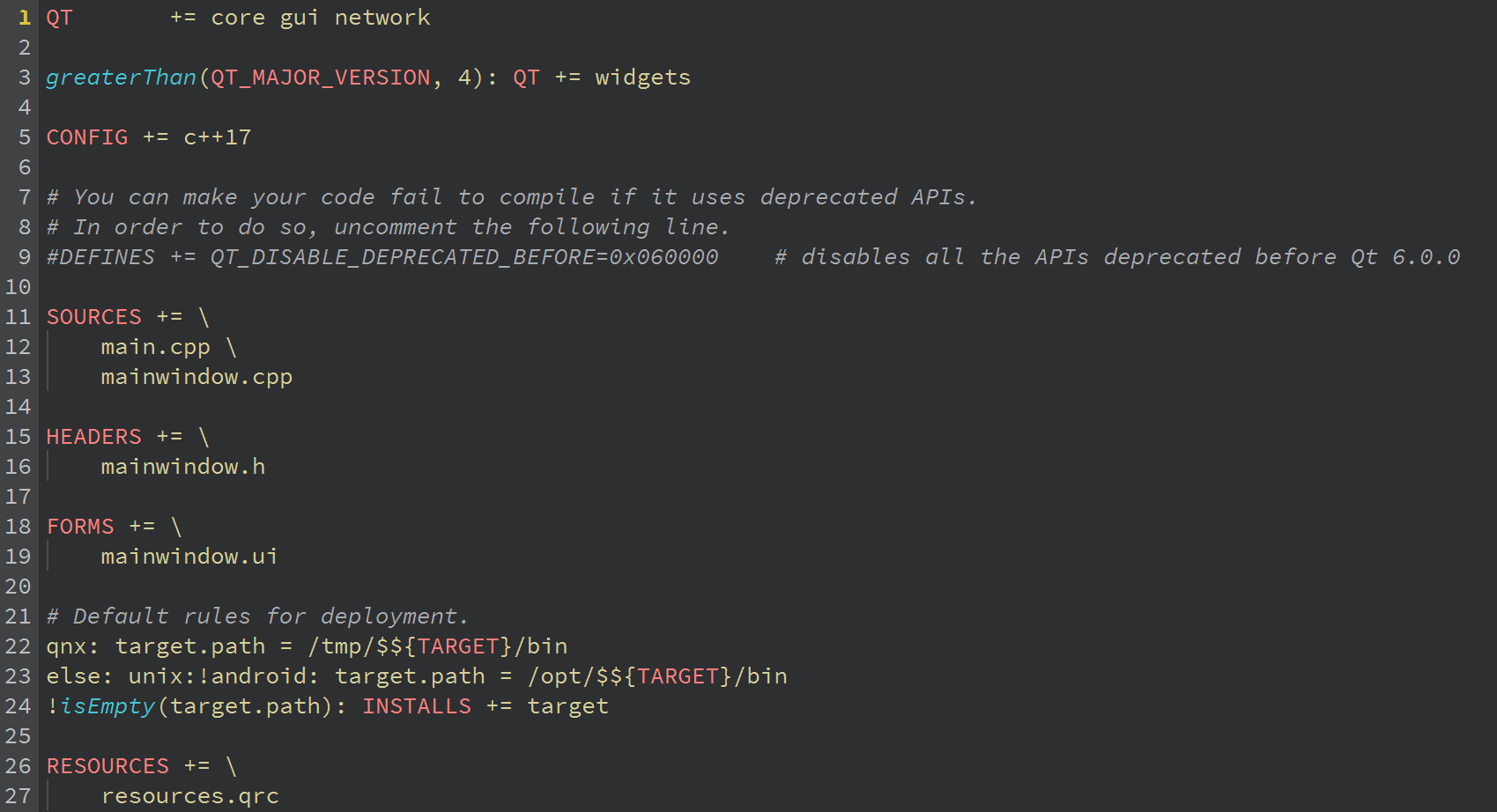


Figure 3-6 qmake

* Includes: The necessary includes for managing weather dashboard functionality are added to the main window.

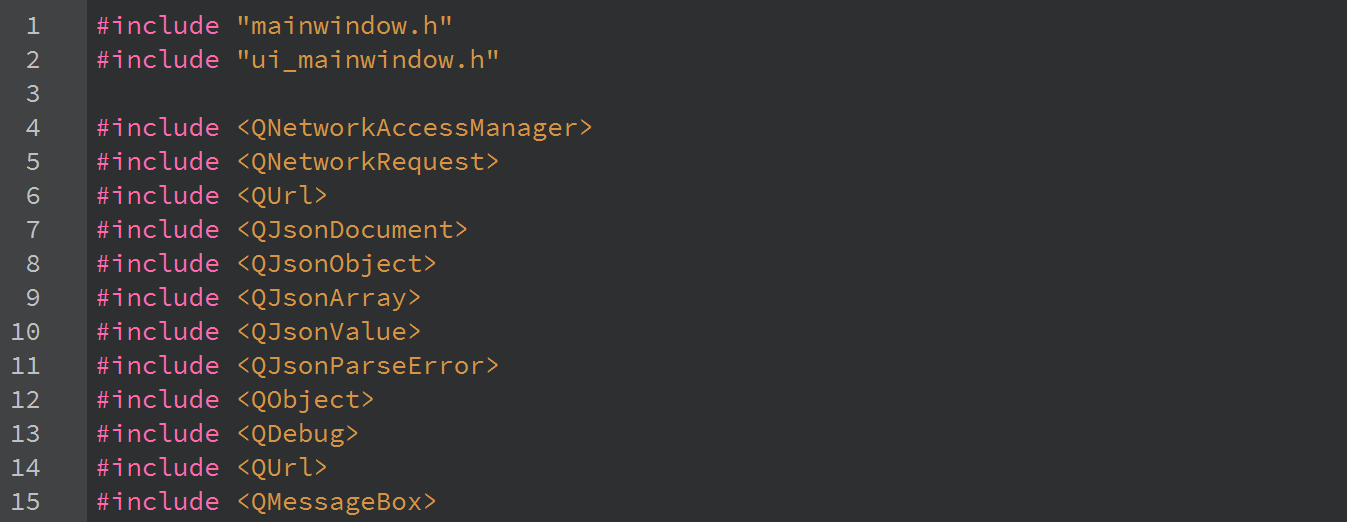


Figure 3-7 Main Window includes



Figure 3-8 mainwindow.h (Part 1)

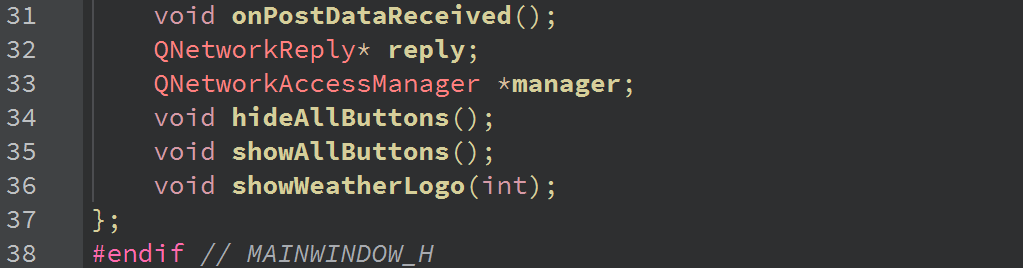


Figure 3-9 mainwindow.h (Part 2)

Last but not least, we added code to main.cpp to instantiate the main window and launch the UI.



Figure 3-10 main.cpp

6.0 Testing and Results

For testing, we first run the program, then the UI interface should appear. Select the country from the combo box and input the zip code for the related cities. After that, click the “Go” button to execute the program and the detailed weather data will be appeared in the UI interface. This confirms that the UI interface worked as expected.

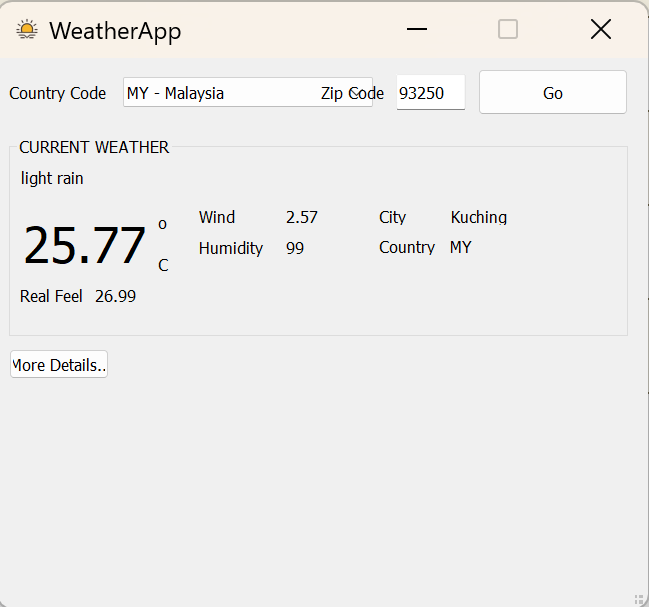


Figure 4 Test showing weather description, temperature and humidity in "WeatherApp" UI interface

7.0 Optimization

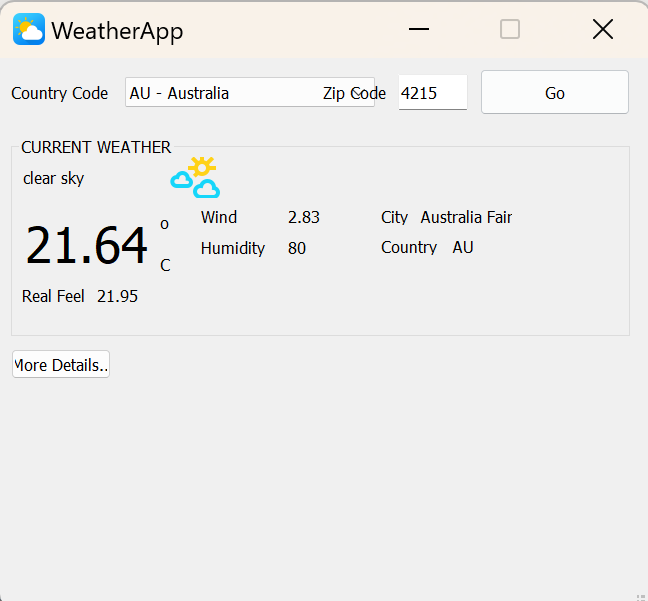
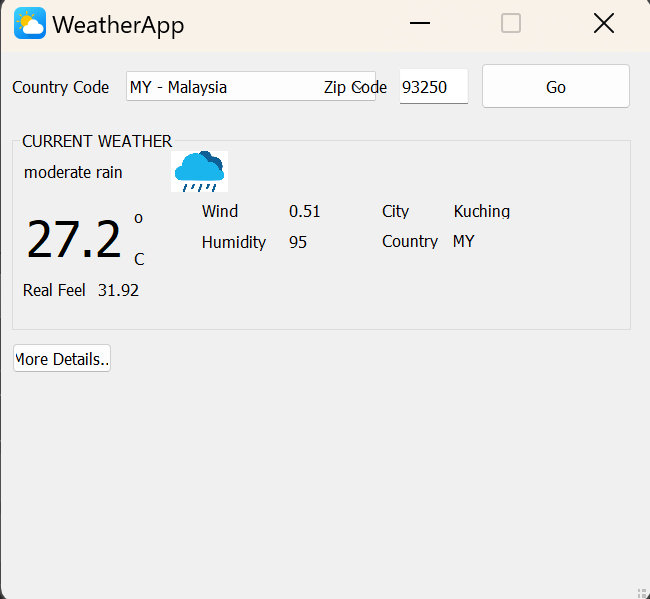
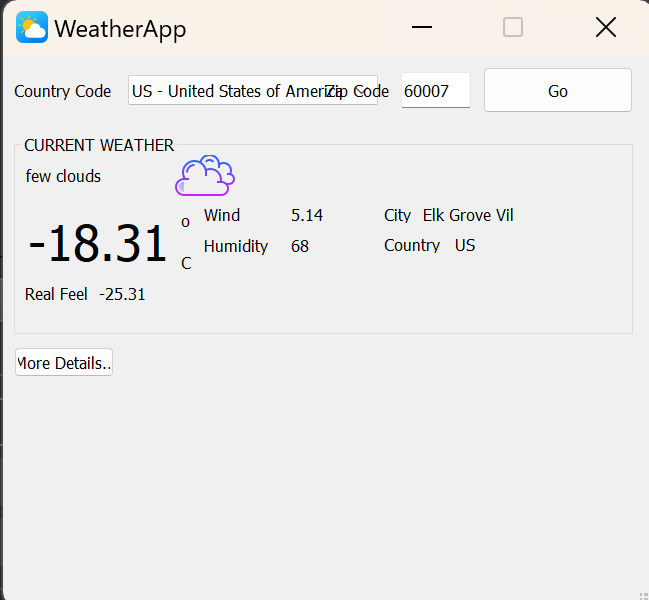


Figure 5 Weather images or logos are included besides the weather descirption in UI interface

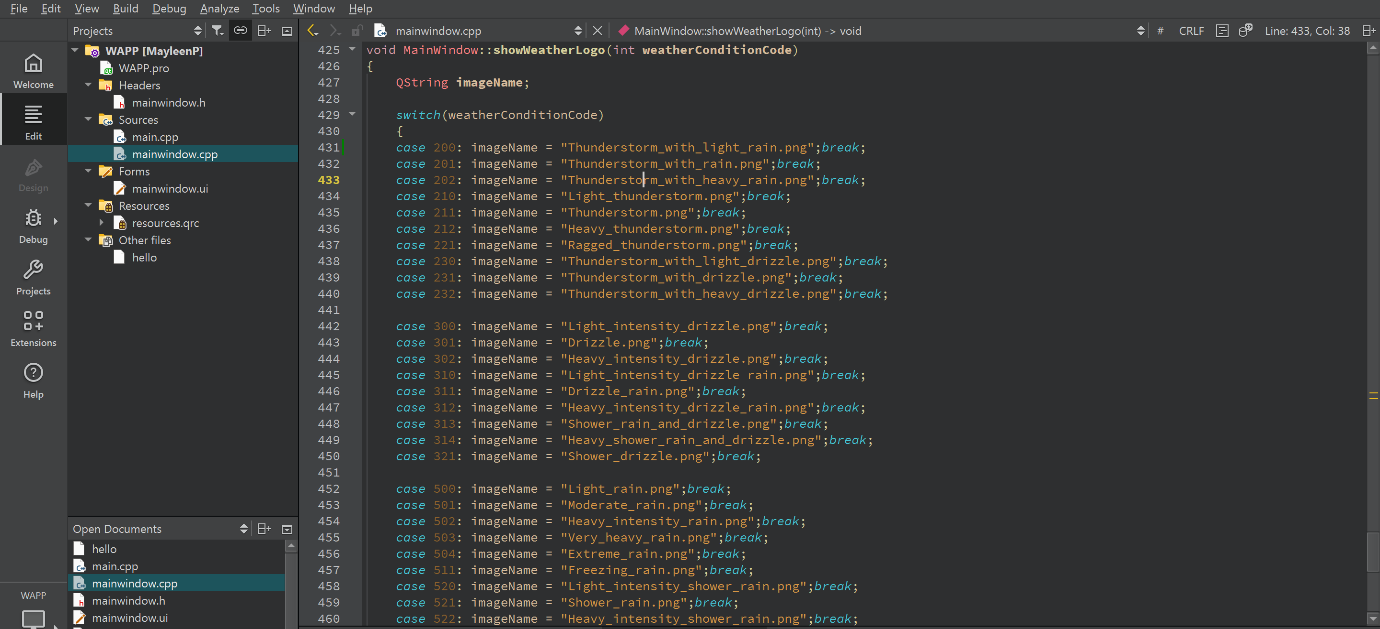
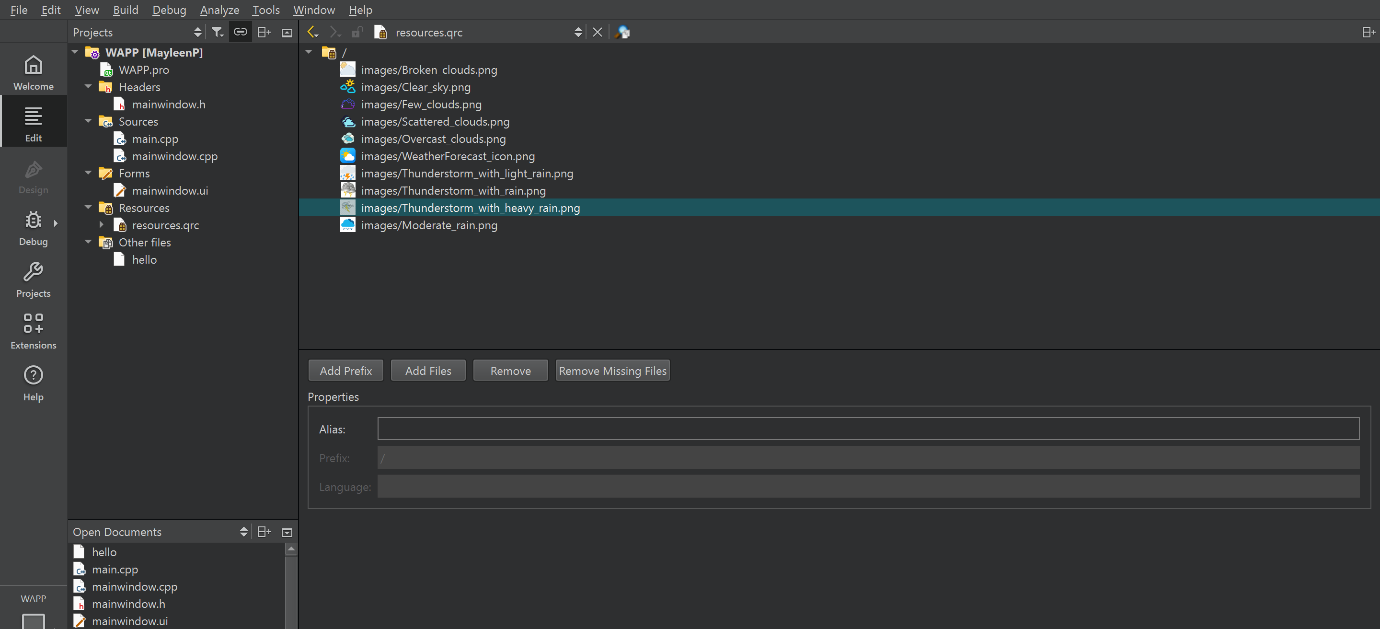


Figure 6 The location stored for those weather images or logos

In order to improve the UI interface of the weather dashboard, I not only add in icon for it, but also include weather images or logos. Those weather images or logos are stored in a Resources file. When a specific weather condition is displayed, the corresponding weather images or logos will also appear, making it easier for everyone to understand the current outdoor weather conditions. As I am still exploring the development of dynamic page displays in Qt application software, I was unable to present a more refined UI interface in this coursework. Therefore, I sincerely ask for your understanding, professor. However, I did learn a lot from this coursework, especially applying and developing a software using Qt.