February 17th

Define the required functionalities of the weather module:

1. Get the current weather
2. Show the weather forecast for the next three days.
3. Update data regularly: refresh the current weather every 30 minutes and the weather forecast every 12 hours.
4. When showing the result, print it on the terminal and also return a struct for others to use.
5. The project should be able to run on a Raspberry Pi in the end.

February 19th to 26th

Task: Initially build the main program weather.cpp to get weather information.

Content: After comparing several weather websites, we decided to use the OpenWeather API. Next, we will use libcurl to get data from the API and jsoncpp to parse the returned data.

Result: Successfully printed the current weather information and a simple weather forecast for the coming days.

Bug: The weather forecasts for today and the next two days are accurate, but the forecast for the third day has some noticeable errors.

February 27th to March 4th

Task: Adjust the method of retrieving the weather forecast and handle the timestamps to ensure they are displayed in Glasgow local time.

Content: Switched to using a combination of the Weather API and Forecast API for weather forecasting. Then used localtime\_r() and strftime() to convert timestamps into the format YYYY-MM-DD.

Result: The accuracy of the weather forecast improved, confirming the effectiveness of the combined API approach. During daytime tests, the output time was displayed correctly.

Bug: However, during nighttime tests, the displayed time tends to shift to the next day. To ensure accurate three-day forecasts, it’s necessary to exclude data from the current day.

March 4th to 8th

Task: Make the program automatically fetch data at regular intervals without blocking the main program.

Content: Used std::thread to create two separate threads and controlled the time interval with this\_thread::sleep\_for (), ensuring that the main thread remains unblocked.

Test: The two threads can run in parallel. One updates the current weather every 30 minutes, and the other updates the weather forecast every 12 hours (with shorter intervals used during testing).

Result: The program runs stably.

March 12th

Task: Modularize the structure by creating weather.h to make it easier for others to use the module.

Content: Encapsulated the returned data into a struct called WeatherData, and provided two interfaces: getCurrentWeather () and getWeatherForecast ().

Result: Successfully tested. Other programs can now retrieve data by including and calling functions from the weather.h file.

March 17th

Task: Upload the code to GitHub and push it to the group repository.

Content: Created a WeatherProject folder containing .cpp, .h, and main.cpp files. Used the Git branch named weather and pushed the project to the group repository.

Result: Successfully uploaded. Team members can directly clone the repository or switch to the weather branch to view the code.