

# **COMPUTER ENGINEERING WORKSHOP**

**S.E. (CIS) OEL REPORT**

**Project Group ID:**

OMAR RASHID ( CS-23136 )

AZEN SAJID ( CS-23103 )

UMAIMA ( CS-23111)

**BATCH: 2023**

**Department of Computer and Information Systems Engineering**

**NED University of Engg. & Tech.,  
Karachi-75270**

## **CONTENTS**

<b>S.No.</b>		<b>Page No.</b>
<b>1.</b>	<b>Problem Description</b>	<b>2</b>
<b>2.</b>	<b>Methodology</b>	<b>2 - 4</b>
<b>3.</b>	<b>Results</b>	<b>4 - 5</b>

## CHAPTER 1

### PROBLEM DESCRIPTION

Environmental monitoring has become increasingly important due to the rising impact of climate change and the need for sustainable development. This project focuses on developing an **Integrated Environmental Monitoring System** using **C programming** that interacts with a free API to retrieve real-time environmental data, including temperature and humidity.

The core objectives of this system are:

- To **retrieve, process, and store environmental data** efficiently.
- To provide **real-time alerts** for critical environmental conditions.
- To employ **C programming fundamentals**, such as pointers, dynamic memory allocation, and modularization, to ensure optimal functionality.
- To leverage **Linux shell scripting** for task automation.

The project aligns with **Course Learning Objective-1 (CLO-1)** by providing hands-on experience with contemporary technologies and **Program Learning Outcome-5 (PLO-5)** by focusing on the modern tools of computer engineering.

### METHODOLOGY

This section provides a description of the step-by-step procedures carried out in designing and implementing the weather monitoring system. The methodology includes data collection, processing, analysis, logging, and scalability, thus providing a systematic approach for the entire system.

#### Data Collection and Processing and Storage:

The first major step is real-time weather data obtained by using an API such as OpenWeatherMap. The system inquires and gets weather data at regular intervals, thus obtaining essential information about temperature, humidity, wind speed, and pressure. This data is then stored in a JSON format (**weather\_history.json**), the raw data archive.

Once the data is collected, the next step involves also its processing and cleansing. This is where the erroneous and missing values are treated, errors that may have arisen on account of network issues or differences in the data emanating from a specific source. For example, temperature values from the source might be in Fahrenheit, whereupon all temperate readings are converted to Celsius. The processed data will then be compiled and saved in `processed_data.txt` for use in subsequent analysis, ensuring its cleanliness, correctness, and proper formation.

### **Calculation and Averaging of Daily Temperature:**

After the data has been cleaned, the system calculates the average daily temperature, which aggregates temperature readings taken throughout the day. This involves summing together all the temperature readings taken at different times of the day and dividing that value by the total number of readings taken to produce the daily average. Average temperatures will be saved to a separate file (average\_temperatures.txt), which will help to track long-term trends in temperature fluctuation.

This temperature analysis not only informs on the general weather of the day but also provides a basis for anomaly detection. For instance, temperature readings thought to be unusually high or low would generate alerts. This is an important step in the recognition of extreme in climate events that is then useful in the building of early warning systems.

### **Alert Generation and Logging:**

After processing the data and performing the analysis, the system goes on to check for anomalies in temperature, such as extreme deviations from normal or sudden extreme changes that can indicate hazardous weather. When these anomalies are detected, alerts are generated in order to warn the user of possible extreme conditions, such as heat waves or cold spells.

These alerts, along with normal system operations, are logged in logfile.log. The system logs not only successful actions-such as data collection, processing, and alert generation-but also any errors or failures that are encountered. Alerts are distinctly marked in the logs with time stamps and severity levels to allow quick identification. For instance, a value exceeding temperature thresholds might generate a high-priority alert, whereas some minor systems-event would be logged as routine entries.

The main goal of this process is to ensure transparency and provide an audit trail for troubleshooting. Any errors, like failed data collection or processing, are recorded. The system either retries to fix the issue or handles errors smoothly to keep it running. Transparency is key, and the auditor can use the records to check the system's status and perform periodic cleanups.

### **System Reliability and Transparency:**

To ensure smooth operation, the system includes strong error handling and transparency features. It detects and logs any faults or missing data for quick correction. The system automatically retries failed API calls and notifies the user if the issue persists after retries. This design reduces the need for human intervention, improving the system's reliability during operation.

Logfiles log any and all activity, success or not. They serve as a complete record of every one of the system's activities, which will facilitate monitoring performance, trend detection, and regular maintenance of the system's well-being. Such alerts for extreme weather events are highlighted in the log; accordingly, remedial action from the user can come swiftly.

## Scalability and Future Enhancements:

The architecture of the system provides for future scalability, including the possibility to add new features easily. Think for example new data sources, such as satellite weather feeds, would provide a wealth of weather data that is richer and more context-sensitive.

Following this systematic approach, the weather monitoring system will give an accurate value for its weather data by monitoring and alerting in real-time, providing an architecture easily scalable in the future.

## RESULTS

### Weather History File (weather\_history.json):

This is a JSON file which stores historic data of all ungodly data from the API. This ensures data is available for a long time for performing years of retrospective analysis.

```
{ } weather_history.json > ...
1  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
2  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
3  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
4  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
5  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
6  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
7  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
8  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
9  [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
10 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
11 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
12 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
13 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
14 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
15 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
16 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
17 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
18 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
19 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
20 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
21 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
22 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
23 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
24 [{"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],
```

### Average Temperature Report (average\_temperatures.txt):

This report gives the average temperature after every 24 readings. It shows time trends and the way the system processes and, in summary, summarizes enormous amounts of data.

```
≡ average_temperatures.txt
1  Average Temperature: 10.42°C
2  Average Temperature: 12.21°C
3  Average Temperature: 13.72°C
4  Average Temperature: 15.02°C
5  Average Temperature: 16.15°C
6  Average Temperature: 16.99°C
7  Average Temperature: 17.58°C
8  Average Temperature: 26.17°C
9  Average Temperature: 29.55°C
10 Average Temperature: 31.25°C
```

### Processed Data File (processed\_data.txt):

This is data that has been processed into a simple format in which humans can read it lightly. It contains the temperature of the weather description. The report favors fast manual review of very important environmental factors.

```
388 26.90°C, few clouds
389 26.90°C, few clouds
390 26.90°C, few clouds
391 26.90°C, few clouds
392 26.90°C, few clouds
393 26.90°C, few clouds
394 26.90°C, few clouds
395 26.90°C, few clouds
396 26.90°C, few clouds
397 26.90°C, few clouds
398 26.90°C, few clouds
399 26.90°C, few clouds
400 26.90°C, few clouds
401 26.90°C, few clouds
402 26.90°C, few clouds
403 26.90°C, few clouds
404 26.90°C, few clouds
405 26.90°C, few clouds
406 26.90°C, few clouds
407 26.90°C, few clouds
408 26.90°C, few clouds
409 26.90°C, few clouds
410 26.90°C, few clouds
411 26.90°C, few clouds
```

### Log File with Alerts (logfile.log):

The alerts ensure timely operator notifications for serious environmental readings.

```
3186
3187 Current Weather for Karachi:
3188 -----
3189 Temperature: 31.90°C
3190 Condition: few clouds
3191 -----
3192 ALERT: High Temperature! 31.90°C
3193
3194 (zenity:3210): Gtk-WARNING **: 16:08:18.912
3195
3196 Current Weather for Karachi:
3197 -----
3198 Temperature: 31.90°C
3199 Condition: few clouds
3200 -----
3201 ALERT: High Temperature! 31.90°C
3202
3203 Average Temperature for the last 24 readings: 16.15°C
3204
3205 Current Weather for Karachi:
3206 -----
3207 Temperature: 29.90°C
3208 Condition: clear sky
3209 -----
```

### Log File without Alerts (logfile.log):

This log contains the usual operations of the system when no alerts have been triggered.

```
logfile.log
3427 Current Weather for Karachi:
3428 -----
3429 Temperature: 26.90°C
3430 Condition: few clouds
3431 -----
3432
3433 Current Weather for Karachi:
3434 -----
3435 Temperature: 26.90°C
3436 Condition: few clouds
3437 -----
3438
3439 Current Weather for Karachi:
3440 -----
3441 Temperature: 26.90°C
3442 Condition: few clouds
3443 -----
3444
3445 Current Weather for Karachi:
3446 -----
3447 Temperature: 26.90°C
3448 Condition: few clouds
3449 -----
3450 rm -f main.o functions.o weather_app
```