# COMPUTER ENGINEERING WORKSHOP

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

**Project Group ID:**

|  |  |
| --- | --- |
| Rida Batool | CS-015 |
| Abdul Wahab | CS-035 |
|  |  |
|  |  |

**BATCH:** 2023

**Department of Computer and Information Systems Engineering**

**NED University of Engg. & Tech.,**

**Karachi-75270**

# CONTENTS

|  |  |  |
| --- | --- | --- |
| S.No. | Description | Page No. |
| 1 | Problem Description | 3 |
| 2 | Methodology | 3 |
| 3 | Results | 5 |
| 4 | Evaluation Criteria | 6 |
| 5 | Conclusion | 6 |

# CHAPTER 1: PROBLEM DESCRIPTION

Project Title:  
Integrated Environmental Monitoring System using Linux and C  
  
Objective:  
To design and develop a robust environmental monitoring system capable of real-time data retrieval, processing, and reporting using Linux and C programming. This project integrates modern programming concepts, including API interaction, dynamic memory allocation, file management, modularization, and real-time notifications.

# CHAPTER 2: METHODOLOGY

The development of the Integrated Environmental Monitoring System followed a structured and modular approach to ensure efficiency, reliability, and scalability. Below is a detailed explanation of the steps:

1. **Data Retrieval**:
   * Leveraged the OpenWeatherMap API to fetch real-time environmental data such as temperature, humidity, and wind speed.
   * The HTTP requests were implemented using the libcurl library in C, which provides robust mechanisms for API communication.
   * Configured API keys for secure and authorized data access.
2. **Data Processing**:
   * Extracted relevant environmental parameters from JSON responses using parsing techniques.
   * A Bash script was developed to process the retrieved data, which included:
     + Calculating averages for key parameters (temperature, humidity, and wind speed) every 12 readings.
     + Handling errors during data extraction and ensuring proper data formatting.
3. **File Management**:
   * Organized retrieved data systematically into text files for raw and processed information:
     + weather\_data.txt: Stored raw weather data fetched from the API.
     + weather\_averages.txt: Recorded averages calculated for every 12 readings.
     + counter.txt: Maintained a running count of completed readings for batch processing.
   * File-based storage ensured data persistence and easy retrieval for future analysis.
4. **Automation with Shell Scripts**:
   * Developed a shell script (run\_fetch.sh) to automate periodic execution of the C program responsible for API calls.
   * Implemented an infinite loop with timed intervals to execute tasks seamlessly.
   * Designed mechanisms to reset data and logs after each cycle of 12 readings.
5. **Dynamic Memory Allocation & Modularity**:
   * Efficiently utilized dynamic memory allocation through malloc and realloc functions to handle varying API response sizes.
   * Adopted a modular approach by dividing the code into separate header and implementation files for better readability and debugging.
6. **Real-Time Alerts**:
   * Configured alerts for extreme weather conditions based on predefined thresholds (e.g., high temperature >30 °C, low temperature <11 °C).
   * Alerts were sent via email using the msmtp tool, integrated with shell scripting for automated notifications.
   * This feature ensured timely updates for critical environmental changes.

# CHAPTER 3: RESULTS

The Integrated Environmental Monitoring System successfully achieved its objectives. Below is a breakdown of the system's outputs and their significance:

1. **Data Logs and Storage**:

Example entries in the data files demonstrate the accuracy and structure of the collected information:

* + - **Raw Data (weather\_data.txt)**:

Weather Data:

Temperature: 15.32 °C

Humidity: 78%

Wind Speed: 3.45 m/s

* + - **Processed Data (weather\_averages.txt)**:

Averages for Last 12 entries:

Average Temperature: 8.80 °C

Average Humidity: 84.00%

Average Wind Speed: 4.12 m/s

1. **Automation Efficiency**:
   * The shell script ensured consistent data fetching at five-second intervals without manual intervention.
   * Average calculations and storage were automated after every 12 readings, demonstrating the system's reliability in periodic data management.
2. **Real-Time Alerts**:
   * Alert notifications proved effective in monitoring critical conditions:
     + **Example Alert**:
       - *Subject*: "Temperature Alert: Very Hot"
       - *Body*: "Alert: It is very hot! Average Temperature: 31.23 °C"
   * This feature is particularly useful for stakeholders who require immediate updates, such as agricultural managers or climate researchers.
3. **System Robustness**:
   * Extensive error handling mechanisms were implemented, including:
     + Detection and logging of HTTP request failures.
     + Graceful memory handling to avoid program crashes.
4. **Evaluation Metrics**:
   * The system demonstrated high accuracy in recording environmental data and generating averages.
   * Alert thresholds were customizable, providing flexibility for various use cases.
5. **Practical Applications**:
   * The project highlighted the potential of integrating Linux tools and C programming for real-world applications such as:
     + Weather monitoring stations.
     + Agricultural climate analysis.
     + Industrial environmental compliance systems.

# CHAPTER 4: EVALUATION CRITERIA

1. CLO-1: Practical experience with Linux and C for real-world systems.  
2. PLO-5: Modern tool usage demonstrated through API integration and automation.

# CHAPTER 5: CONCLUSION

The project demonstrates the successful development of a scalable environmental monitoring system. It meets the objectives and offers future scalability through the following enhancements:  
  
- Include air quality monitoring in API data.  
- Develop a GUI-based interface.  
- Improve error handling with retry mechanisms.