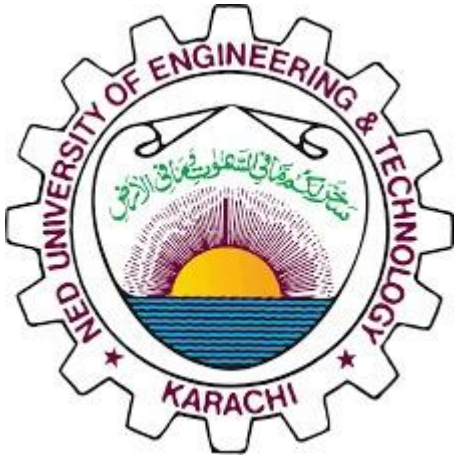


Computer Engineering Workshop (CS-219)

Department of Computer Science and Information Technology

OPEN ENDED LAB

Name: Mariam Fatima , Sitwat Samara , Marium Shad

Roll No: CS-23003 , CS-23105 , CS-23116

Section: C

Submitted to: Miss Mahnoor

NED University of Engineering & Technology, Karachi

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-219
Course Title: Computer Engineering Workshop
Open Ended Lab
SE Batch 2023, Fall Semester 2024
Grading Rubric
TERM PROJECT Group Members:

Student No.	Name	Roll No.
S1	**Mariam Fatima**	**CS-23003**
S2	**Sitwat Samara**	**CS-23105**
S3	**Marium Shad**	**Cs-23116**

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion1: Has the student implemented an efficient and scalable solution for data retrieval, processing, and reporting?						
0	1	2	3			
The student has not even implemented a basic solution that meets the project's requirements.	The student has implemented a basic solution that meets the project's requirements but may lack optimization in certain aspects.	The student has implemented a proficient and well-optimized solution.	The student has implemented an exceptionally efficient and scalable solution.			
Criterion 2: Has student demonstrated a strong understanding of C programming fundamentals?						
0	1	2	3			
The student doesn't have basic understanding of C programming fundamentals.	The student exhibits a basic understanding of C programming fundamentals.	The student demonstrates a strong understanding of C programming fundamentals.	The student demonstrates an exceptional understanding of C programming fundamentals.			
Criterion 3: How well written is the report?						
0	1	2	3			
The submitted report is unfit to be graded.	The report is partially acceptable.	The report is complete and concise.	The report is exceptionally written.			
			Total Marks:			

PROBLEM DESCRIPTION

The goal of this project is to design and implement an Environmental Monitoring System using the C programming language. This system retrieves real-time environmental data (e.g., temperature, humidity) from a free API, processes the data, and provides notifications when predefined thresholds are exceeded. Additionally, it incorporates automation via shell scripts for periodic data retrieval and integrates modular programming practices through header files for enhanced readability and maintainability.

OBJECTIVES

- Interact with a free API to fetch real-time environmental data.
- Process and store the data efficiently using pointers and dynamic memory allocation.
- Automate data retrieval and reporting using shell scripts.
- Implement a notification system for critical environmental conditions.
- Apply modular programming principles with header files

TECHNICAL DETAILS

Language: C

Libraries Used:

- libcurl: For HTTP requests and data fetching.
- cJSON: For parsing JSON responses.
- libnotify: For sending desktop notifications.

Threshold: 35°C (configurable in main.c).

File Outputs:

- raw_data.json: Contains raw API responses.
- processed_data.txt: Stores the processed temperature data in Celsius

PROJECT IMPLEMENTATION

Core Functionalities:

Data Retrieval:

The system uses the retrieve Data function to fetch environmental data and store it in a structured format (Environmental Data). This function is intended to simulate API interaction for real-time monitoring.

Alert Mechanism:

When temperature values exceed a specified threshold (e.g., TEMP_THRESHOLD= 20.00), the system triggers the send Alert function. The function uses Linux system calls or prints notifications to the console as a placeholder.

Automation With Shell Scripts:

A shell script automates the periodic execution of the data retrieval process, logging outputs for debugging and tracking.

Code Structures:

- The main.c file contains the main logic, including data retrieval and alert handling.
- The alerts.h header file defines the interface for the notification mechanism.
- External files (api_handler.h) modularize API interactions, improving code clarity.

Dynamic Memory Management:

Pointers and dynamic memory allocation techniques are utilized to ensure the system efficiently handles variable data sizes, minimizing memory overhead.

Shell Script Integration:

A cron tab is configured to execute the retrieve_data.sh script every minute, ensuring continuous monitoring and average.sh to execute after 24 hours to determine the average at the end of the day.

CODE AND FUNCTIONALITY

Threshold Check:

The program compares the retrieved temperature against the threshold.

Notification System:

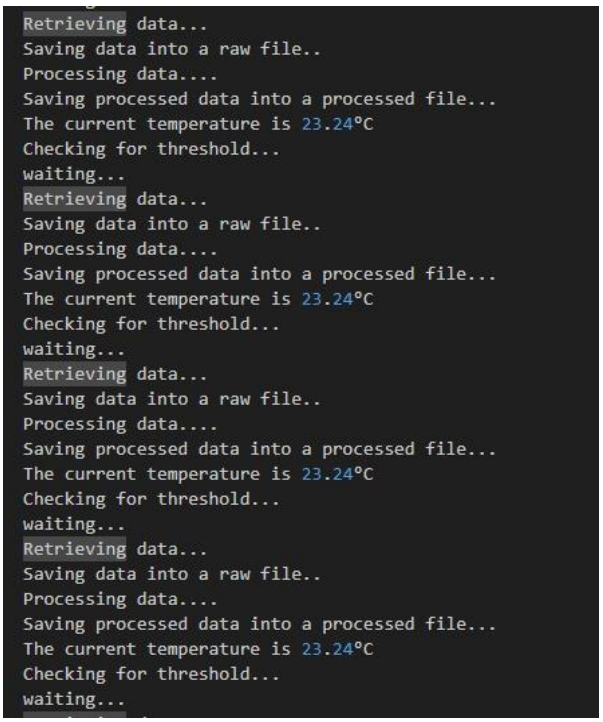
Alerts are handled via the send Alert function defined in alerts.h.

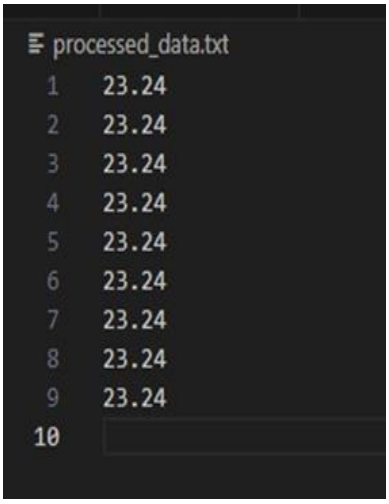
Automated Workflow:

The workflow is managed by a while-loop in main.c (currently commented out), which ensures periodic data retrieval and condition checks.

TESTING AND RESULTS.

-The result of the Temperature Monitoring and Alert System is a functional application that successfully retrieves real-time weather data, processes it, and triggers notifications when the temperature exceeds a specified threshold. The system effectively communicates with the Open WeatherMap API, parses the data, and converts it to Celsius. It then compares the temperature to a set threshold (35°C) and sends a desktop notification if the threshold is exceeded. Additionally, the system logs both raw and processed data for reference.





FUTURE WORK

1. Expand the system to monitor additional parameters (e.g., humidity, air quality).

2. Implement a database for long-term data storage and analysis.

3. Integrate visualization tools to provide real-time dashboards.

CHALLENGES FACED

API Integration: Handling API communication and response structure issues.

Error Handling: Addressing network errors, malformed responses, and memory allocation failures.

CONCLUSION

This project demonstrates an integrated approach to environmental monitoring, utilizing C programming fundamentals, shell scripting, and modular design principles. While functional, further improvements can include:

- Real API integration for actual data retrieval.

- More robust alert systems (e.g., email or SMS notifications).

- Enhanced user interaction for dynamic threshold adjustments.