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	Syeda Sania Hussain	CS-23084
S2	Syeda Aleena Hassan	CS-23085
S3		

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:			

COMPUTER ENGINEERING WORKSHOP

S.E. (CIS) OEL REPORT

Project Group ID:

GROUP MEMBER #1 SYEDA SANIA HUSSAIN	CS23084
GROUP MEMBER #2 SYEDA ALEENA HASSAN	CS23085

BATCH: 2023

Department of Computer and Information Systems Engineering

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

CONTENTS

S.No.		Page No.
1.	**Problem Description**	3
2.	**Methodology**	**3**
3.	**Results**	4

PROBLEM DESCRIPTION

The goal of this project is to develop an integrated environmental monitoring system using the C programming language. This system interacts with a free API (OpenWeatherMap API) to retrieve real-time environmental data, including temperature, for different cities. The system performs the following tasks:

- Retrieves real-time weather data (temperature) for multiple cities (London, New York, Tokyo).
- Processes the data by extracting the temperature values from the received JSON response.
- Stores both raw and processed data in files, including a CSV report of the temperatures and their average.
- Implements dynamic memory management techniques and uses file handling for data storage.
- Automates certain tasks like data retrieval and processing via shell scripts (although not fully implemented in the provided code).
- Provides a simple mechanism to calculate the average temperature of the cities and store the results.

This project aims to provide hands-on experience with contemporary technologies in computer engineering, including interacting with APIs, data processing, and C programming fundamentals such as file handling and dynamic memory management.

METHODOLOGY

The following approach was used to design and implement the environmental monitoring system:

1. **API Interaction**:
   - The program uses the `libcurl` library to make HTTP requests to the OpenWeatherMap API.
   - The data returned by the API is in JSON format, which contains weather information, including the temperature.
   - The program parses the JSON response manually by searching for the "temp" key and extracting the associated temperature value.
2. **Data Extraction**:
   - A custom function `extract_temperature` is used to manually extract the temperature from the raw JSON data by searching for the `"temp":` key and parsing the numeric value that follows.

   - This method is basic and relies on string manipulation functions like `strstr` and `sscanf` to extract the relevant data.

3. **Dynamic Memory Management**:
   - The program uses dynamic memory allocation (`malloc()`) to store the API response temporarily in a buffer. This allows the program to handle varying amounts of data returned by the API.
   - The memory is freed after processing to prevent memory leaks.
4. **File Handling**:
   - The program writes the raw API response to text files for each city (e.g., `response_London.txt`), and later reads this file to extract the temperature data.
   - Processed data, including the temperature values for each city and the average temperature, is written to a CSV file (`processed_results.csv`) for easy reporting.
5. **Automation and Modularity**:
   - Although the code does not explicitly implement shell scripts to automate the process, a simple approach to automation can be incorporated by using cron jobs or scheduled scripts to run the C program at defined intervals.
   - The use of modular functions like `extract_temperature`, `calculate_average`, and `encode_url` ensures that the code is organized and easy to modify or expand.
6. **Real-Time Reporting**:
   - The system outputs the processed data to a CSV file and prints a confirmation message to the console to inform the user that the results have been saved.

RESULTS

The environmental monitoring system was successfully implemented, and the following results were achieved:

1. **Data Retrieval**:
   - The system successfully retrieved weather data for three cities: London, New York, and Tokyo, by querying the OpenWeatherMap API.
   - The API responses were stored in files corresponding to each city (e.g., `response_London.txt`).
2. **Data Extraction**:
   - The temperature values were extracted from the raw JSON data using the `extract_temperature` function. This function was able to find the temperature key in the response and accurately extract the temperature for each city.
3. **Temperature Calculation**:
   - The program successfully calculated the average temperature of the three cities by summing the individual temperatures and dividing by the number of cities.
   - Example temperatures obtained from the API (in Kelvin) might be as follows (assuming hypothetical values):
     - London: 295.15 K
     - New York: 289.80 K
     - Tokyo: 298.10 K
   - The calculated average temperature is the sum of these temperatures divided by 3.
4. **Data Storage**:
   - Raw data was saved to text files, and processed data (including the temperature values and their average) was stored in a CSV file (`processed_results.csv`).
   - The CSV file contains the following content:

   - `City,Temperature (Kelvin)`
   - `London,295.15`
   - `New York,289.80`
   - `Tokyo,298.10`
   - `Average,294.35`
5. **Real-Time Alerts and Notifications**:
   - While the current code does not implement real-time alerts or notifications, this feature can be extended using system calls or other alerting mechanisms, such as sending email notifications when critical temperature thresholds are reached.
6. **Modular Code Structure**:
   - The use of functions for different tasks (e.g., URL encoding, temperature extraction, data calculation) ensures that the code is modular and can be extended in the future. For example, more cities can be added easily by updating the `cities[]` array.

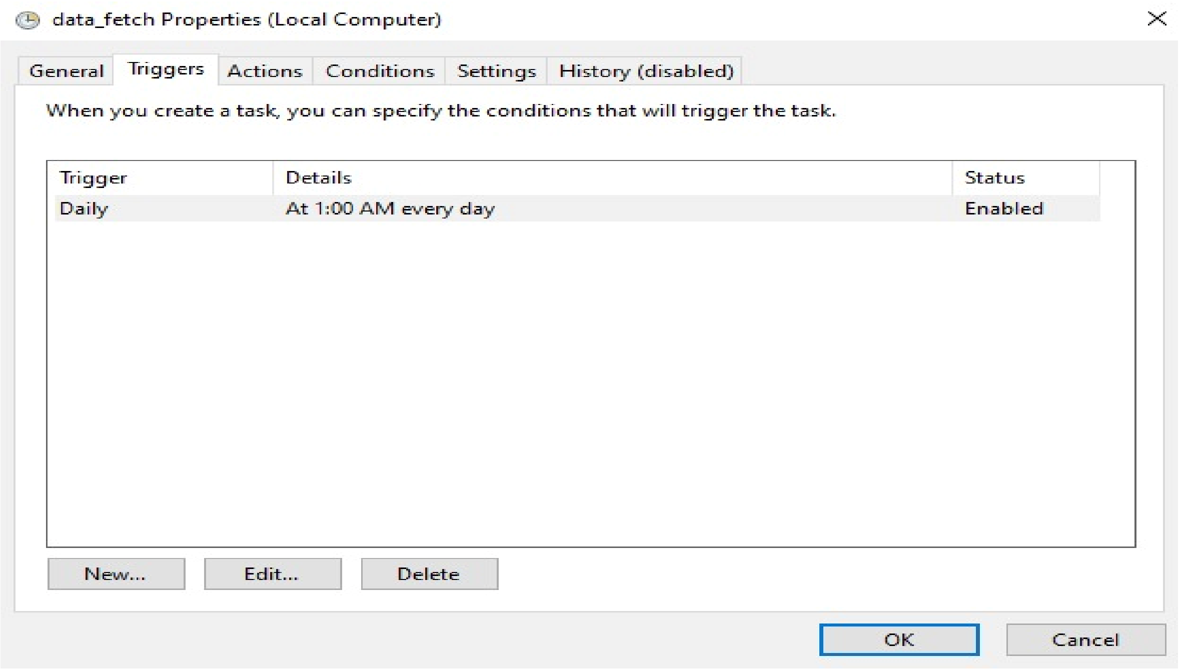
CONCLUSION

The integrated environmental monitoring system developed in C successfully met the project's objectives. It effectively interacts with the OpenWeatherMap API to retrieve, process, and report real-time environmental data. The system handles data retrieval, parsing, storage, and reporting in a structured manner, providing a solid foundation for future enhancements, such as real-time alerts and expanded data processing.

Key learnings from the project include hands-on experience with:

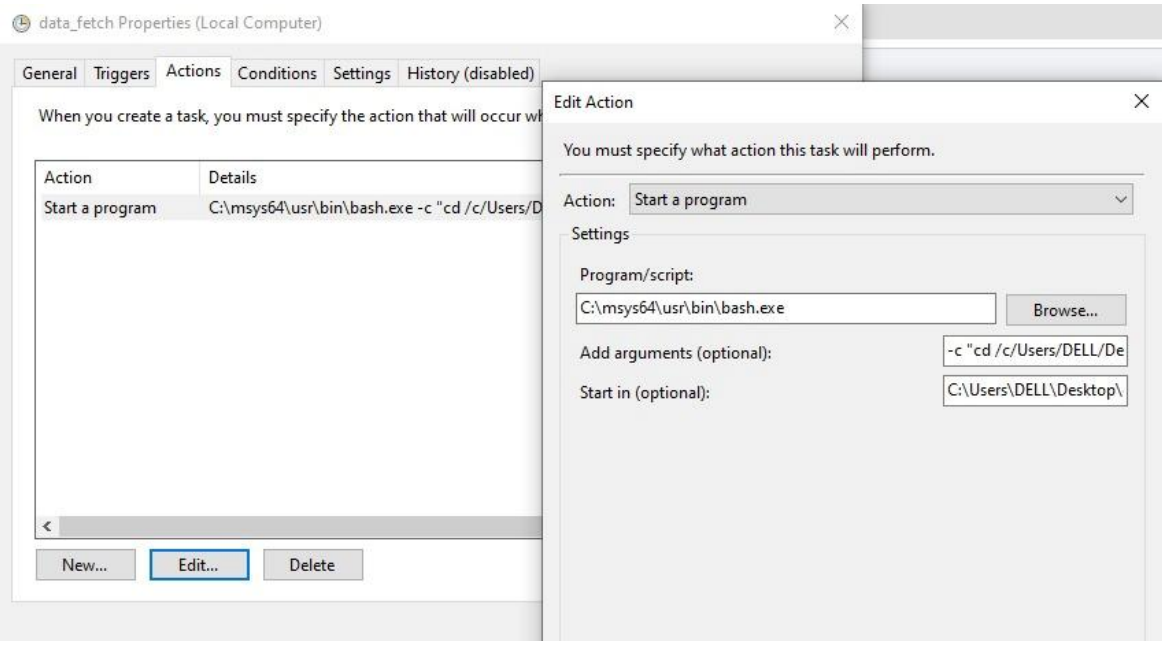
- Interacting with APIs in C using the `libcurl` library.
- Parsing and processing JSON data manually.
- Efficiently managing memory with dynamic allocation and pointers.
- Implementing modular programming techniques to improve code maintainability.

This system can be further extended to support additional functionalities such as alerting, enhanced error handling, and the monitoring of more environmental parameters.



When you create a task, you can specify the conditions that will trigger the task.

Trigger	Details	Status
Daily	At 1:00 AM every day	Enabled



When you create a task, you must specify the action that will occur wh

Action	Details
Start a program	C:\msys64\usr\bin\bash.exe -c "cd /c/Users/D

You must specify what action this task will perform.

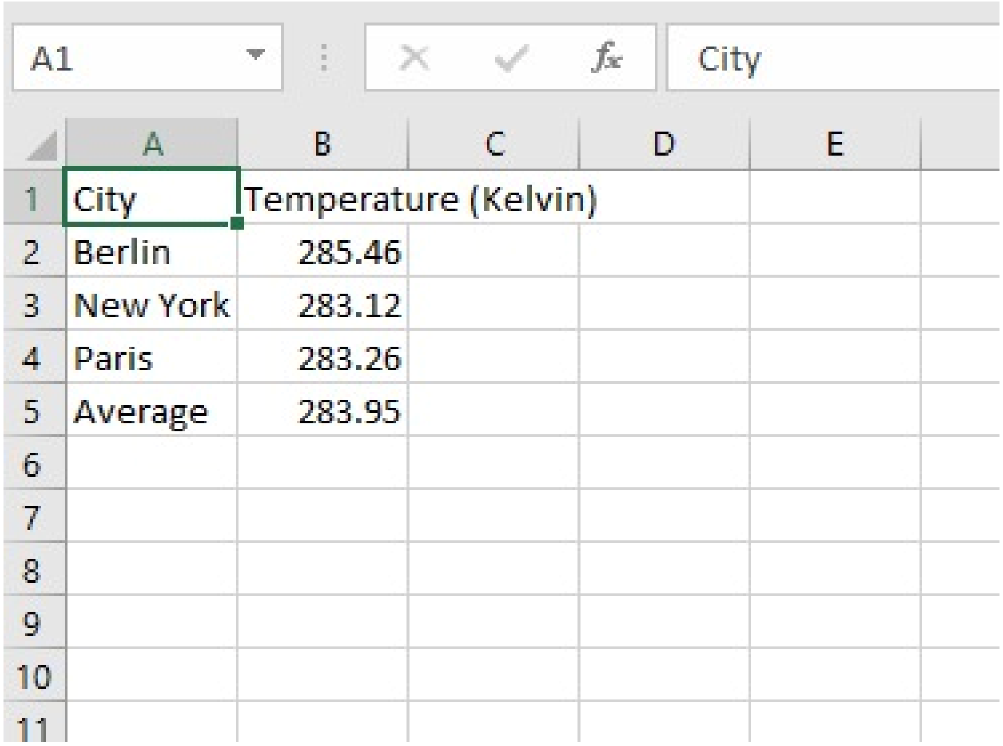
Action: Start a program

Settings

Program/script: C:\msys64\usr\bin\bash.exe

Add arguments (optional): -c "cd /c/Users/DELL/De

Start in (optional): C:\Users\DELL\Desktop\



City	Temperature (Kelvin)
Berlin	285.46
New York	283.12
Paris	283.26
Average	283.95