CS-219

COMPUTER ENGINEERING WORKSHOP

Prepared By:

SYEDA ZAINAB FATIMA (CS 22051)

SYEDA SAMANA ZEHRA (CS 22052)

ANUM AZAM (CS 22053)

OPEN ENDED LAB

PROBLEM DEFINITION:

Construct an integrated environmental monitoring system in C, covering a range of fundamental concepts and practical applications. The project involves interacting with a free API that provides real-time environmental data. The system's core functionalities include data retrieval, processing, analysis, and reporting.

INTRODUCTION:

This C program fetches weather information from the OpenWeatherMap API for a specific city (in this case, Karachi) using libcurl, processes the JSON response, performs analysis, and generates reports.

INSTALLATION:

- **CURL (Command-Line URL)** is a command-line utility and library for transmitting data via URLs. In the Weather App: Made HTTP calls to a weather API. It facilitates contact with a remote server in order to receive weather data.
- **JSON Library**: JSON (JavaScript Object Notation) libraries help in encoding and decoding JSON data. In the Weather App: Used to process and handle the weather API's JSON-formatted response. The purpose of this function is to allow the extraction and use of particular meteorological information from the JSON response.

FLOW OF THE PROGRAM:

We created two main files, main1.c and main2.c, each serving distinct purposes in our project. main1.c provides an interactive interface with options for the user to choose, generating results based on their selection. On the other hand, main2.c is designed to automate the retrieval and display of weather information at regular intervals.

The automation.sh script has been enhanced to seamlessly integrate the C programs. It compiles and executes main2.c, which includes linking other necessary C files. This automation ensures that the weather information is printed every minute.

In addition, the compile.sh file compiles and executes the process for main1.c, presenting a user-friendly interface with selectable options.

Here's a breakdown of the program:

1. **Libraries:** The program includes standard C libraries (stdio.h, stdlib.h, string.h, time.h) and additional libraries for handling HTTP requests (curl/curl.h) and JSON data (json-c/json.h).
2. **Configuration:**
   - Set the OpenWeatherMap API key in the apiUrl variable.
   - Configure email settings in "email.c".
3. **MemoryStruct Structure:** Defines a structure to store the API response in memory (struct MemoryStruct).
4. **Global Variables:** Temperature and humidity are global variables to store weather information for analysis.
5. **getCurrentTime Function:** Gets the current time and date in a specified format.
6. **WriteCallback Function:** Callback function for libcurl to handle writing data received from the API.
7. **processData Function:** Parses the JSON response and extracts relevant weather information, performs analysis for anomalies, and prints the results. It also writes the processed data to a file named processed_data.txt.
8. **generateReport Function:** Generates a report based on the analyzed data and writes it to a file named report.txt.
9. **getProcessedData:** This function retrieves and prints the processed data from the "processed_data.txt" file.
10. **printReport:** The printReport function is responsible for retrieving and displaying the content of the "report.txt" file.
11. **main Function:** The main program flow includes:
    - Initializing libcurl.
    - Setting the API URL and making a request to OpenWeatherMap.
    - Writing the raw data to a file named raw_data.txt.
    - Calling processData to analyze and print the weather information.
    - Calling generateReport to create a report based on the analysis.

12. **Cleanup:** Freeing allocated memory and cleaning up libcurl resources.
13. **File Handling**
    - Raw API data is stored in "raw_data.txt".
    - Processed weather information is stored in "processed_data.txt".
    - Reports are generated in "report.txt".
14. **Error Handling:** Error messages are printed to stderr for failed operations.

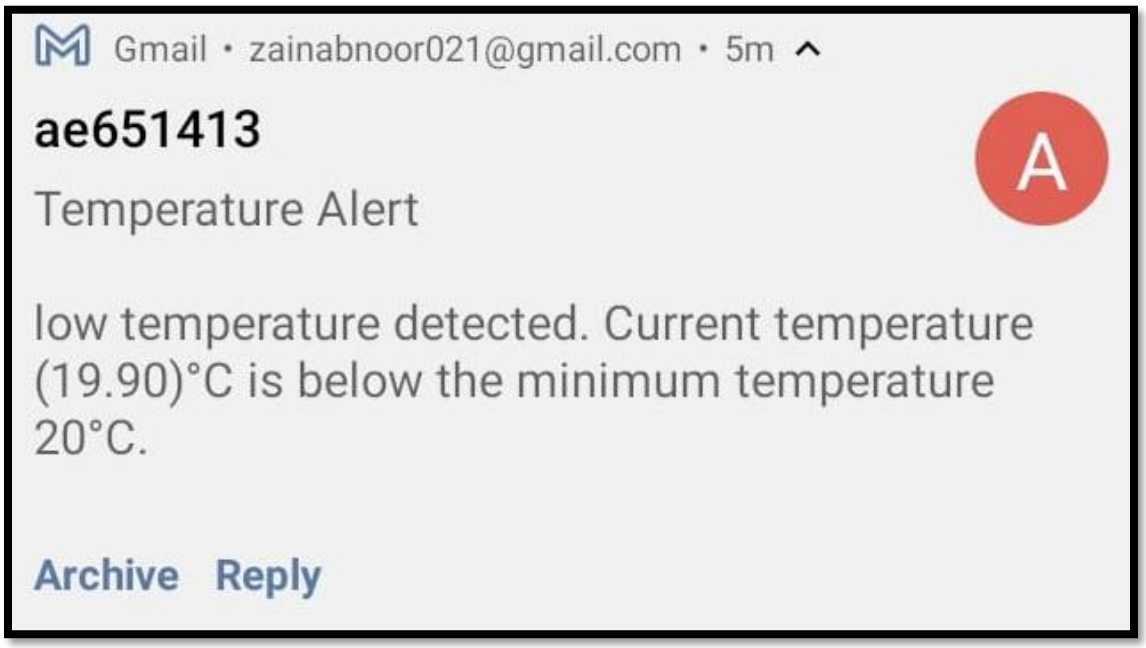
EMAIL NOTIFICATIONS:

The Weather Monitoring System includes an email notification feature to alert users when anomalies or specific trends are detected in the weather data. This functionality is implemented in the send_email function located in the email.c source file.

The send_email function is created to accept the recipient's email address and a predefined email body as parameters. The email body is dynamically generated based on the detected anomalies or trends in the weather data.

To configure the email settings in the email.c source file, the SMTP server, port, sender's email address, and authentication credentials like the app password are provided due to which a fake gmail ID has to be created. This ensures that the email notifications are sent successfully.

Here is the Snapshot of the email received:



Snapshot of the Output: