COMPUTER ENGINEERING WORKSHOP

S.E. (CIS) OEL REPORT

Project Group ID:

MUHAMMAD ARHAM KHAN	CS-047
MUHAMMAD OWAIS MADNI	CS-070
MUHAMMAD OWAIS QADIR	CS-071

BATCH: 2023

Department of Computer and Information Systems Engineering

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

CHAPTER:1

Problem Description

Objective:

Develop an Integrated Environmental Monitoring System in C that interacts with a free API to retrieve, process, and report real-time environmental data.

Key Features:

Core Components:

1. Live Data Collection:

- Fetch real-time metrics like temperature and humidity from a free API.

2. Data Management:

- Save raw data in files and process it for meaningful insights.

3. Automated Workflows:

- Implement shell scripts to streamline data collection and processing.

4. Critical Alerts:

- Generate real-time notifications for hazardous environmental conditions using Linux system calls.

5. Optimized Resource Handling:

- Use pointers and dynamic memory allocation for efficient data operations.

6. Clean and Structured Code:

- Utilize header files to organize functionalities for clarity and ease of maintenance.

PURPOSE:

This project highlights hands-on programming skills and problem-solving abilities by tackling real-world challenges in computer engineering. It applies cutting-edge technologies to deliver an effective solution for environmental monitoring.

CHAPTER:2

METHODOLOGY:

API Interaction

- We selected and understood a suitable free API to fetch environmental data.
- We wrote C code to retrieve and parse the data.

Data Storage

- We designed a file structure to store both raw and processed data.
- Functions were implemented to save this data into files.

Shell Script Automation

- We created shell scripts to automate the retrieval and processing of data.
- Cron jobs were set up to schedule these tasks.

Pointers and Dynamic Memory Allocation

- We used pointers to handle data more efficiently.
- Dynamic memory allocation was implemented for better data management.

Real-Time Alerts

- Linux system calls were utilized to continuously monitor data.
- Alerts were set up to notify us about any critical readings.

Code Modularity

- Our code was organized into header files to enhance readability.
- Functions and data structures were clearly defined in these headers.

Testing and Debugging

- Each component and the entire system were thoroughly tested.
- Debugging tools were employed to fix any issues that arose.

CHAPTER:3

RESULTS:

1. API Data Retrieval Results:

- Successfully accessed and retrieved real-time environmental data through a free API.

2. Efficient Data Storage:

- Raw and processed data were effectively stored in files according to the designed structure.

3. Automation and Scheduling:

- Automated the tasks of data retrieval and processing using shell scripts, scheduled with corn jobs.

4. Advanced Memory Management:

- Employed pointers and dynamic memory allocation to enhance data manipulation efficiency.

5. Real-Time Environmental Alerts:

- Implemented real-time alerts to notify relevant personnel of critical environmental conditions.

6. Modular Code Implementation:

- Structured code into header files to improve readability and maintainability.

7. Comprehensive Testing and Debugging:

- Conducted thorough testing of individual components and the entire system, addressing all issues discovered during debugging.