Individual Project - Log Parser and Aggregator CLI Tool

Name: Rushabh Runwal
SJSU ID# 018176834

Question #1: Describe what problem you're solving.

We aim to streamline the processing and analysis of application-generated log files, which often contain a mix of performance metrics, system events, and user requests. While these logs are essential for assessing application health, troubleshooting, and performance monitoring, their unstructured format can make them hard to interpret. Our solution classifies logs into distinct categories—such as APM logs, application logs, and request logs—and performs key aggregations like minimum, median, average, and count. This transforms raw log data into clear, actionable insights. The processed output is saved into structured JSON files, ensuring easy integration with other tools and systems. The solution is built with scalability and flexibility in mind, allowing support for new log types and formats in the future.

Question #2: What design pattern(s) will be used to solve this?

To address this requirement, the **Chain of Responsibility (CoR) design pattern** is applied. It provides a clean and modular way to process different types of logs—such as APM, Application, and Request logs. In this approach, each handler is dedicated to processing a specific type of log attribute, such as performance metrics, severity levels, or response times. If a handler cannot process a given log, it forwards it to the next handler in the chain. This design offers several benefits:

1. Dynamically processes logs based on their type.
2. Allows easy integration of new log types without modifying existing code.
3. Keeps each handler focused on a single responsibility, improving maintainability.
4. Ensures invalid or unprocessable logs are either flagged or passed along safely.
5. Enables the system to adapt to evolving requirements and log formats.

Overall, the CoR pattern enhances efficiency, flexibility, and scalability in log processing.

Question #3: Describe the consequences of using this/these pattern(s).

- **Unmatched Logs Risk Being Ignored:**

Logs that don't fit any handler criteria may go unprocessed unless explicitly identified and flagged.

- **Challenging Debugging:**

Tracing the flow of a log through a lengthy chain can be difficult, making it harder to diagnose processing issues.

- **Potential Performance Overhead:**

As the number of handlers increases, logs may experience delays due to passing through multiple processing stages.

Question #4: Class Diagram: