Log Parser - Design Document

1. Problem Being Solved

We are building a CLI tool that parses a mixed-format log file, identifies three types of logs (APM, Application, Request), and performs specific aggregations for each. The results are saved into separate JSON files (apm.json, application.json, request.json). The tool is designed to be fault-tolerant, extensible, and should generate output files even when no valid logs are found.

2. Design Pattern Used

We use the Chain of Responsibility design pattern. This pattern allows us to build a chain of log handlers — each responsible for one type of log (APM, Application, or Request). Each log type is handled by a separate class implementing the LogHandler interface. Each handler either processes the line or passes it to the next handler in the chain.

3. Consequences of Using This Pattern

- The logic for each log type is clean and isolated. Each handler uses its own aggregators to collect values, calculate summaries or calculations needed and write to JSON files.
- The invalid lines are safely ignored.
- Supports more log types in the future - To add a new log type, create a new LogHandler and aggregator, then link it into the chain. Existing code does

not need to change. The pattern supports extension by design, allowing new output files and parsing logic with minimal effort.