Coding Challenge

-Log Monitoring Application-

You are provided with a log file, application.log. The goal of this challenge is to build a log monitoring application that creates a comprehensive report on the system's behavior, focusing on performance, reliability, and stability indicators derived from the logs. This task should take at most 90 minutes of your time, focus on problem solving and good coding practices.

Application Logs (application.log):

- These logs use ISO 8601 timestamps.
- Each log line records events from various applications, identified by application names.
- Each event has a PID (process ID), a log level (Informational, Warn, or Critical), and a descriptive message.
- Messages often contain dynamic and contextual information such as filenames, task IDs, measured times, and codes. Some events may also include user IDs.

Reporting:

- Produce a detailed report summarizing both sets of logs. This report could be in a structured format (e.g., CSV, JSON, or a formatted text table).
- Include:
    - **From Complex Application Logs**:
        - The count of events by log level and application.
        - Top 5 applications by number of warnings and critical errors.
        - Any notable metrics extracted from messages (e.g., highest memory usage, longest reported time_spent, etc.).
        - Summaries of critical errors: how many, which codes occur most frequently, and any patterns with user IDs.

Code Quality and Best Practices:

- Write clean, maintainable, and well-structured code.

- Include comments, docstrings, and clear naming conventions.
- Use meaningful commit messages to show incremental progress.
- Include unit tests to validate:
    - Correct parsing of log lines.
    - Accurate computation of durations.
    - Proper handling of thresholds and categorization of events.
    - Edge cases (e.g., tasks that never finish, malformed lines, unexpected log levels).

2. **Documentation**:
    - Provide a README or instructions that explain:
        - How to run your application.
        - How to configure thresholds, input files, and output formats.
        - A sample of the expected output.
    - Explain how one might extend the code for new log formats or additional metrics.
3. **Version Control**:
    - Use a public repository (e.g., GitHub or GitLab).
    - Commit regularly with clear messages.
    - Share the public repository link as proof of completion.

Tips for Good Code:

- Keep functions small and single-purpose.
- Separate parsing logic from aggregation and reporting logic.
- Write clear tests that focus on both typical and edge-case scenarios.
- Consider performance and scalability—assume large logs.
- Document how to run tests and where results are stored.