Coding Challenge

-Log Monitoring Application-

You are provided with a log file, jobs.log, containing entries of both scheduled tasks and background jobs. Each entry includes a timestamp, job type, job ID, whether it started or finished, and a PID. The goal of this challenge is to build a log monitoring application that reads the file, measures how long each job takes from start to finish, and generates warnings or errors if the processing time exceeds certain thresholds.

Your application should be able to:

1. Parse the jobs.log file.
2. Identify each job or task and track its start and finish times.
3. Calculate the duration of each job from the time it started to the time it finished.
4. Produce a report or output that:
    - Logs a warning if a job took longer than a specified warning threshold to complete (e.g. > 10 seconds).
    - Logs a error if a job took longer than a specified critical threshold to complete (e.g. > 30 seconds).

The threshold (warning/error) can be hard-coded or configurable via environment variables or command-line arguments.

Log Structure:

- **HH:MM:SSS** is a timestamp in hours, minutes, and milliseconds (e.g., 11:35:002 means 11 hours, 35 minutes, and 2 milliseconds).
- Job type is either **"scheduled task"** or **"background job"**.
- The job ID is either a 3-digit numeric ID for scheduled tasks (e.g., 002, 947) or a 3-letter alphabetic ID for background jobs (e.g., xyz, abc).
- Each job has a PID associated with it, e.g., 46578.
- A job "started" line appears before a corresponding "finished" line for the same job ID and PID.

Additional Requirements:

- **Code Quality**:
    - Write clean, readable, and well-structured code.
    - Include comments to explain your logic, especially any tricky parts.
    - Commit changes with clear and descriptive commit messages to illustrate your development process.
- **Testing**:
    - Include unit tests to verify key parts of your logic, particularly:
        - Parsing log lines
        - Matching start/finish entries
        - Computing durations correctly
        - Handling edge cases (e.g., job that doesn't finish, malformed input lines)
- **Documentation**:
    - Write a brief README or project description detailing:
        - How to run the application
        - How to configure thresholds
        - What the output looks like
        - How to run tests
- **Version Control and Submission**:
    - Use a source code management system (e.g., Git).
    - Create a **public repository** (e.g., on GitHub or GitLab) containing your source code, tests, and documentation.
    - Provide a link to your repository as proof of completion.

What We Are Looking For:

- Your ability to parse and interpret data from a structured log.
- Problem-solving and good coding practices.

- Clear unit tests and documentation that shows you understand how to verify correctness.
- Consistent use of version control with meaningful commits.