**Log Analyzer Home Assignment**

**Overview**

Implement a command-line tool that analyzes log files to extract and report structured event data.   
Please use your preferred coding language. I recommend Python.

**1. Input**

The tool accepts the following **command-line arguments**:

* **--log-dir <path>**: Path to a folder containing log files.
* **--events-file <path>**: Path to a configuration file (events.txt) that defines how to filter log events.
* **--from <timestamp>**: (Optional) Filter logs with timestamp starting from this value (inclusive).
* **--to <timestamp>**: (Optional) Filter logs with timestamp up to this value (inclusive).

Timestamp format: YYYY-MM-DDTHH:MM:SS  
Example: 2025-06-01T14:00:00

**Example Input Line**

LogAnalyzer –log-dir /temp/logs –events-file events.txt –from 2025-06-01T10:00:00 –to 2025-06-01T15:00:00

**2. Log File Format**

Each log line is expected to follow this format:

<TIMESTAMP> <LEVEL> <EVENT\_TYPE> <MESSAGE>

The event rows have ascending time order.

Example:

2025-06-01T14:03:05 INFO TELEMETRY Iteration time: 1793.845 sec

2025-06-01T14:05:22 WARNING DEVICE detected high temperature of device c95fe73e-14db-4ef4-909d-0c1db67c41ee: 40C

2025-06-01T14:10:00 ERROR GNMI unresponsive telemetry at endpoint http://SWX1:9001/ low\_freq\_debug

**3. Events Configuration File**

The events.txt file contains event definitions, one per line, with optional filters:

**Format:**

EVENT\_TYPE [--count] [--level LEVEL] [--pattern REGEX]

* EVENT\_TYPE: A string representing the type of event to match (e.g. TELEMETRY, DEVICE, GNMI).
* **--count: Optional**. Instructs the tool to **count** matching log entries.  
  If omitted, all matching log lines will be printed.
* **--level LEVEL: Optional**. Only match log entries with this log level (e.g., INFO, WARNING, ERROR).
* **--pattern REGEX: Optional**. A regular expression to apply to the event’s message field.

Multiple filters for the same event type are allowed.  
If multiple filters are applied for the same event, apply **each filter independently** and display the output in separate lines. The count flag is applied to each filter separately.

**Example events.txt**

TELEMETRY --count --pattern ^Iteration time:\s\d+\.\d+\ssec$

DEVICE --count --level WARNING

GNMI --level ERROR

**4. Example Output**

The output format should match the following example:

Event: TELEMETRY pattern [^Iteration time:\s\d+\.\d+\ssec$] count — matches: 1 entries

Event: DEVICE level [WARNING] count — matches: 1 entries

Event: GNMI level [ERROR] — matching log lines:

2025-06-01T14:10:00 ERROR GNMI unresponsive telemetry at endpoint http://SWX1:9001/ low\_freq\_debug

**5. Required Submission Outcome**

1. Submit all related code files or project files that are required for the code to run.
2. Add all the test inputs and configuration files that were used.
3. Add all the corresponding output files.
4. Add a description of your solution and design considerations.

**6. General points to pay attention to**

1. **Good CLI UX (--help, proper error messages)**
2. **Modular code** with clean interfaces
3. **Readability and maintainability**

**7. BONUS**

1. Add support for compressed logs. Allow .log, .log.gz in the log folder.
2. Add performance considerations to improve speed.

Good Luck!