ADVANCED REGEX CHALLENGE

Enterprise System Log Intelligence Engine

Background Scenario

A **distributed microservices platform** deployed on Kubernetes generates **heterogeneous logs** from:

- API Gateways
- Authentication services
- Container runtime
- Database access layers
- CI/CD pipelines

The logs are:

- Mixed format (JSON-like, key-value, free text)
- Case inconsistent
- Sometimes quoted
- Sometimes masked
- Generated by multiple services simultaneously

Your task is to **design REGEX-ONLY solutions** to validate, extract, redact, and analyze these logs.

No string splitting, no parsing libraries, no JSON parsers — **regex only**.

Sample Log Stream (Input)

```
[INFO] 2025-03-21T14:22:19Z service=auth userId=USR_1023 action=LOGIN_SUCCESS ip=192.168.1.10
[WARN] 2025-03-21T14:22:22Z service=auth userId=USR_2045 passwordTemp123 LOGIN_FAILED
```

```
[ERROR] 2025-03-21T14:22:30Z service=payment txnId=TXN998877 amount=₹45,000.50
status=FAILED
[DEBUG] <***> service=payment <===> txnId=TXN112233 amount=$1200 status=SUCCESS
[INFO] "user passwordReset456 completed successfully"
[CRITICAL] service=db query="SELECT * FROM users WHERE password='abc123'"
[KUBE] pod=api-gateway-7f9d8 container=nginx restartCount=3
```

TASK SET (COMPLEXITY: HIGH → VERY HIGH)

Task 1: Validate Standard Log Header

Requirement

Write a regex that validates:

- Severity inside `[ ]`: `INFO`, `WARN`, `ERROR`, `DEBUG`, `CRITICAL`
- ISO-8601 timestamp (`YYYY-MM-DDTHH:MM:SSZ`)
- Exactly one space between sections

Regex Output

- Match the **entire header**
- Reject malformed timestamps

Example Match

```
[INFO] 2025-03-21T14:22:19Z
```

Task 2: Extract Service Name and User ID (Conditional Presence)

Requirement

Using **named capturing groups**, extract:

- `service` value
- `userId` value **only if present**

Constraints

- `userId` format: `USR_` followed by digits
- `service` must be lowercase letters only

Expected Groups

service → auth
userId → USR_1023

Task 3: Detect and Extract Weak Password References

Requirement

Write a regex that:

- Detects `password` followed by alphanumeric characters
- Works in:
    - Plain text
    - Quoted strings
    - SQL queries
- Case-insensitive

Must Match

passwordTemp123
passwordReset456
password='abc123'

Must NOT Match

pass_word
pwd123

Task 4: Extract Transaction Data with Multi-Currency Support

Requirement

Capture:

- Transaction ID: `TXN` + digits
- Amount:
  - `₹` with commas and decimals
  - `$` without commas

Must Extract

txnId → TXN998877
amount → ₹45,000.50

Task 5: Ignore Masked or Redacted Secrets

Requirement

Write a regex that **matches secrets only if NOT masked**.

Masked patterns:

password=****
password=XXXXX
password=####

Must Match

password=abc123
passwordTemp456

Must NOT Match

password=****

Hint: Negative lookahead required.

Task 6: Identify SQL Injection Risk Queries

Requirement

Detect SQL queries that:

- Contain `SELECT`
- Reference `password`
- Use `WHERE`

Order does not matter.

Must Match

SELECT * FROM users WHERE password='abc123'

Hint: Multiple lookaheads.

Task 7: Kubernetes Restart Detection

Requirement

Extract:

- Pod name
- Container name
- Restart count > 0

Expected Extraction

pod → api-gateway-7f9d8
container → nginx
restartCount → 3

Task 8: Flag High-Risk Log Lines

Requirement

Match a log line if **ANY** of the following occur:

- Severity = `ERROR` or `CRITICAL`
- Contains `password`
- Contains `FAILED`
- Kubernetes restartCount ≥ 3

Single regex allowed.

Task 9: Validate ISO-8601 Timestamp Strictly

Requirement

Validate timestamps:

- UTC only (`Z`)
- Correct date and time ranges
- No milliseconds

Valid

2025-03-21T14:22:19Z

Invalid

2025-13-40T99:99:99Z

Task 10: Redact Sensitive Data Using Regex Replace

Requirement

Write **regex replace rules** to:

- Replace passwords with `***REDACTED***`
- Replace credit card numbers with `XXXX-XXXX-XXXX-XXXX`
- Preserve log structure