CodeXray Intern Evaluation Project

Theme:

Build a simplified **Observability & Security Microservice** that collects system metrics, generates alerts, and exposes secure APIs for reporting.

Project Objectives

Evaluate the intern's ability to:

1. Use **data structures and algorithms** effectively.
2. Write **clean, modular, and secure code**.
3. Apply **problem-solving and engineering principles** to a realistic problem.
4. Demonstrate **understanding of observability basics** — metrics, alerts, and reporting.
5. Optionally design a **web dashboard** for visualization.

Project Overview

You will build a small program or microservice (Python/Node.js/Go/Java) that:

1. **Collects system resource metrics** (like CPU and memory usage).
2. **Generates alerts** when thresholds are breached.
3. **Stores logs and alerts efficiently**.
4. **Exposes a secure REST API** to generate and retrieve a summary report.
5. *(Bonus)* Provides a **web dashboard** to visualize data and configure alerts dynamically.

Evaluation Phases

Phase 1: Fundamentals & Data Structures (20 pts)

Task 1: Log Analyzer Utility

- Parse a provided sample log file.
- Output count of each log level (INFO, WARN, ERROR).
- Show top 5 most frequent errors.
  Skills Tested: Hash maps, string parsing, sorting, efficiency.

Scoring:

- Correct parsing and counts – 5 pts
- Efficient use of DS (maps/lists) – 5 pts
- Clean, modular, readable code – 5 pts
- Comments and testability – 5 pts

CODE X RAY

Phase 2: Secure Coding & Encoding (20 pts)

Task 2: Password & Session Management

- Implement password hashing (SHA-256 or bcrypt).
- Implement simple session storage and validation using a hash map.
- API endpoints for `/register`, `/login`, `/validate-session`.
  Skills Tested: Security fundamentals, API design, modularity.

Scoring:

- Secure password storage (hashing, no plaintext) – 7 pts
- Correct session validation logic – 7 pts
- Code clarity and structure – 3 pts
- Documentation – 3 pts

Phase 3: Observability Core (30 pts)

Task 3: Metric Collection & Alerting

- Collect **any two system resource metrics**, e.g.:
  - CPU usage (%)
  - Memory usage (%)
    (Tip: Use OS libraries like `psutil` in Python or `os`/`systeminformation` in Node.js)
- Define thresholds for each metric (e.g., CPU > 80%, Memory > 75%).
- Generate alerts and store them with timestamps.
- Implement in-memory or lightweight DB (e.g., SQLite) storage.
  Skills Tested: Real-time data handling, logic, efficiency, observability basics.

Scoring:

- Metric collection works correctly – 10 pts
- Alert generation logic accurate – 10 pts
- Data handling (structure, efficiency) – 5 pts
- Clean modular code & error handling – 5 pts

Phase 4: Reporting API (20 pts)

Task 4: External API for Summary Reports

- Create a REST API endpoint `/summary` that returns:
  - Total alerts generated.
  - Breakdown by type (CPU/Memory).
  - Last N alert timestamps.
  - Average metric values for the last 10 readings.
- Secure the endpoint with token/session-based access.

Scoring:

- Functional API and correct output – 10 pts
- Secure access (token/session) – 5 pts
- Code structure & documentation – 5 pts

Bonus Phase: Web Dashboard (10 pts)

Task 5: Visualization & Configuration UI

- Create a simple **web dashboard** (HTML/JS or any framework like Vue.js/React) to:
  - Visualize metric trends over time.
  - Show active and historical alerts.
  - Allow user to **configure alert thresholds** (saved locally or via API).
    Skills Tested: Frontend design, API integration, data visualization.

Scoring:

- Data visualization (charts or tables) – 4 pts
- Configurable thresholds (API or UI) – 4 pts
- Overall polish (design + usability) – 2 pts

Overall Scoring Summary

Category	Phase	Points
Phase 1	Fundamentals (Log Analyzer)	20
Phase 2	Security & Session Management	20
Phase 3	Metrics & Alerting	30
Phase 4	Reporting API	20
Bonus	Dashboard	10
Total		**100 + 10 Bonus**

Submission Requirements

1. **Code Repository (GitHub/GitLab)**:
   - Proper folder structure.
   - Clear **README.md** with setup and API usage.
   - **Sample outputs or screenshots** for metrics and alerts.
2. **Commit Quality**:
   - Logical commits with meaningful messages.
3. **Documentation**:
   - Inline comments + short architecture summary.