Practical Exam: Full-Stack Developer (Intern)

Problem: Develop a "Demo Log Management System" that supports multiple data sources and can be deployed both as a **Hardware Appliance** (installed on a single machine/VM) and as **SaaS/Cloud** (with a public URL for external testing).

Deadline + Demo Interview: Within 10 days of seeing this assignment. The student must schedule the appointment.

Work Format: Individual work.

Technology: Candidates can choose their own Tech Stack (Open Source/Cloud Native recommended).

1) Objectives

1. **Evaluate Full-Stack Capabilities:** Architecture design, Backend/API coding, Frontend/UI, Data Pipeline, DevOps/Deployment.
2. **Evaluate Event/Log Management:** Handling data from multiple sources, Normalization, Indexing, Search, Visualization, Alerting.
3. **Evaluate Security Understanding:** AuthN/AuthZ, TLS, Multi-tenant (separating customer data).

2) Scope & Must-have

2.1 Supported Data Sources (At least 4 real sources + others can use sample/simulator)

- **Firewall / Network** (e.g., Syslog UDP/TCP 514 or HTTP Ingest)
- **API** (Receive JSON via REST/HTTP POST)
- **CrowdStrike** (Synthetic sample JSON/CSV allowed)
- **AWS** (e.g., CloudTrail/ALB/NLB - sample files allowed)
- **Microsoft 365** (Unified Audit Log - sample JSON allowed)
- **Microsoft AD/Windows Security** (EventID 4624/4625 etc. - sample allowed)

Note: Connecting to all real sources is not mandatory. At least 4 sources must be capable of real ingestion (e.g., Syslog + HTTP API + JSON batch file + script to simulate sending) and centralized Schema Normalization before storage.

2.2 Minimum Features (Functional)

- **Ingestion:** Accept logs in multiple formats (Syslog / HTTP JSON / File batch) and support at least 2 protocols.
- **Normalization:** Convert all log types into a Central Schema (see section 3 for schema example).
- **Storage & Query:** Store data in a searchable format (e.g., OpenSearch/ClickHouse/Postgres+GIN/etc.).
- **Dashboard:** UI showing a summary (Top IP/User/EventType, Timeline, Filter by time range/tenant).
- **Alert (Minimum 1 rule):** Set simple conditions (e.g., repeated failed logins from the same IP within 5 minutes) and display in an Alert page or send via Webhook/Email.

- **AuthN/AuthZ:** At least 2 user roles (Admin/Viewer) and separate data by tenant (parameter or header/claim).
- **Deployment (2 Modes):**
  - **Appliance:** Runs on a single machine/VM (Docker Compose recommended).
  - **SaaS/Cloud:** Runs on public cloud (e.g., VM/Container with a URL for the committee to access).
- **TLS:** Enable HTTPS at least in SaaS mode (Self-signed is acceptable if steps are clearly explained).
- **Retention:** Set data retention for at least 7 days (delete/rollover/partition).

2.3 Nice-to-have (Not mandatory but recommended)

- **True Multi-tenant** (Separate index/table), RBAC per field/tenant.
- **Enrichment** (e.g., reverse DNS, GeoIP) during ingestion.
- **CI/CD** (GitHub Actions/etc.), **IaC** (Terraform/Helm).
- **Unit/Integration tests** (Minimum coverage).

3) Central Schema (Recommended)

Adaptable, but should cover key fields for searching:

- `@timestamp` (RFC3339)
- `tenant`
- `source` (firewall | crowdstrike | aws | m365 | ad | api | network)
- `vendor` , `product`
- `event_type` , `event_subtype`
- `severity` (0-10)
- `action` (allow | deny | create | delete | login | logout | alert)
- `src_ip` , `src_port` , `dst_ip` , `dst_port` , `protocol`
- `user` , `host` , `process` , `url` , `http_method` , `status_code`
- `rule_name` , `rule_id`
- `cloud.account_id` , `cloud.region` , `cloud.service`
- `raw` (store raw text)
- `_tags` (array)

4) Sample Logs (Short for testing)

Use as seed data or send via API.

4.1 Firewall/Syslog

```
<134>Aug 20 12:44:56 fw01 vendor=demo product=ngfw action=deny src=10.0.1.10 dst=8.8.8.
```

4.2 Network (Router Syslog)

```
<190>Aug 20 13:01:02 r1 if=ge-0/0/1 event=link-down mac=aa:bb:cc:dd:ee:ff reason=carrie
```

4.3 HTTP API (POST /ingest)

```
{
  "tenant": "demo",
  "source": "api",
  "event_type": "app_login_failed",
  "user": "alice",
  "ip": "203.0.113.7",
  "reason": "wrong password",
  "@timestamp": "2025-08-20T07:20:00Z"
}
```

4.4 CrowdStrike (sample JSON)

```
{
  "tenant": "demoA",
  "source": "crowdstrike",
  "event_type": "malware_detected",
  "host": "WIN10-01",
  "process": "powershell.exe",
  "severity": 8,
  "sha256": "abc...",
  "action": "quarantine",
  "@timestamp": "2025-08-20T08:00:00Z"
}
```

4.5 AWS CloudTrail (abbreviated)

```
{
  "tenant": "demoB",
  "source": "aws",
  "cloud": { "service": "iam", "account_id": "123456789012", "region": "ap-southeast-1"
  "event_type": "CreateUser",
  "user": "admin",
  "@timestamp": "2025-08-20T09:10:00Z",
  "raw": { "eventName": "CreateUser", "requestParameters": { "userName": "temp-user" }
}
```

4.6 Microsoft 365 Audit (abbreviated)

```
{
  "tenant": "demoB",
  "source": "m365",
  "event_type": "UserLoggedIn",
  "user": "bob@demo.local",
  "ip": "198.51.100.23",
```

```
  "status": "Success",
  "workload": "Exchange",
  "@timestamp": "2025-08-20T10:05:00Z"
}
```

4.7 Microsoft AD/Windows Security (abbreviated, 4625)

```
{
  "tenant": "demoA",
  "source": "ad",
  "event_id": 4625,
  "event_type": "LogonFailed",
  "user": "demo\\eve",
  "host": "DC01",
  "ip": "203.0.113.77",
  "logon_type": 3,
  "@timestamp": "2025-08-20T11:11:11Z"
}
```

5) Reference Architecture (Optional/Adjustable)

- **Collector/Ingest:** Vector / Fluent Bit / Logstash / Custom (Node/Go/Python)
- **Storage/Index:** OpenSearch / ClickHouse / PostgreSQL(+JSONB/GIN) / Elasticsearch
- **Backend API:** FastAPI / Express / Go Fiber / NestJS (Auth + Ingest endpoint)
- **UI/Dashboard:** React / Vue + Chart library OR OpenSearch Dashboards / Grafana
- **Packaging:** Docker Compose (Appliance) + Cloud VM/Container (SaaS)

Appliance Requirements (Minimum): Ubuntu 22.04+, 4 vCPU, 8 GB RAM, 40 GB Disk, expose necessary ports (e.g., 80/443/514).

6) Deliverables

1. **Git Repository containing:**
   - `/docs/architecture.md` (Diagram + Data flow/Tenant model explanation)
   - `/docs/setup_appliance.md` and `/docs/setup_saas.md` (Detailed installation steps)
   - `docker-compose.yml` (and/or Helm chart) + init/seed scripts
   - `env.example` (Config values), `Makefile` / `run.sh` scripts
   - `/samples/` (Sample log files + sending scripts e.g., `send_syslog.sh` , `post_logs.py`)
   - `/backend/` , `/frontend/` , `/ingest/` (Code with README)
   - `/tests/` (At least 2-3 example cases)
2. **Demo Video (30 mins):** Explain architecture + Demo ingestion, search, dashboard, alert.
3. **Demo URL (SaaS)** and/or **OVA File/Spin-up method for Appliance**.
4. **Postman/Insomnia Collection** for API ingest/search (if applicable).

7) Acceptance Checklist (Steps the examiner will test)

- [ ] Open the system in Appliance mode according to docs (1 command or very few steps).
- [ ] Send sample Syslog (e.g., via logger/nc) and see it in UI within 1 minute.
- [ ] Call `POST /ingest` with sample JSON and be able to search for it.
- [ ] Upload/Point to sample AWS/M365/AD files and system normalizes them.
- [ ] Dashboard displays Top N, Timeline, Filter by tenant/source/time correctly.
- [ ] Create a sample Alert rule and witness the notification (UI/Email/Webhook).
- [ ] Test RBAC: Viewer user sees only their own tenant.
- [ ] SaaS mode is accessible via HTTPS.

8) Grading Criteria (100 Points)

Category	Details	Points
Architecture & Docs	Clarity, Patterns, Reason for technology choice	15
Ingestion	Supports multiple sources/protocols, Stability, Simulation	20
Normalization/Schema	Proper Schema design and mapping	10
Storage & Query	Fast/Correct search, Good indexing/partitioning	10
Dashboard/UI	Easy to use, Necessary graphs/tables/filters	10
Alerting	At least 1 rule type + Successful notification	10
Security	AuthN/AuthZ, RBAC, Minimum TLS	10
Deployment	Appliance + SaaS actually works	10
Tests & DX	Test examples, Scripts/Makefile, .env.example	5
Total		100

Bonus Points (Max +10): Real Multi-tenant (separate index/table), Enrichment, CI/CD, IaC, Observability (metrics/trace), Hardening.

Passing Score: 60 points. **Strong Hire Consideration:** ≥ 85 points with clear design explanation.