NEXT ASSIGNMENT – ADVANCED JAVA + DJANGO

PART 1 – ADVANCED JAVA ASSIGNMENT

Distributed Alert & Monitoring Engine (Threads + Schedulers + Queues + File Logging)

Objective:

Build a Java system that monitors multiple data sources using threads, triggers alerts, manages concurrency, and generates reports.

Java Requirements:

1. Data Source Module

- Simulate 3–5 data sources (CPU, Temperature, Network, etc.)

- Each runs in its own Thread

- Generates random values every 2 seconds

- Pushes data into a BlockingQueue

2. Alert Rule Engine

- Class: AlertRule(ruleId, metricName, threshold, condition)

- Continuously reads queue

- Generates alerts when rule is violated

3. Alert System

- Class: Alert

- Stores alerts in list

- Writes to alerts.log

- Auto-clean expired alerts using ScheduledExecutorService

- Cache last N values in ConcurrentHashMap

4. Concurrency Requirements

- ExecutorService thread pool

- BlockingQueue

- synchronized / locks for alert list

- Sensors run in parallel

5. Reporting Module

- Alerts per metric
- Highest frequency metric
- Last 10 alerts
- Cache dump
- Save as alert_report.txt

Java Submission:

- Source code
- alerts.log
- alert_report.txt
- README

PART 2 – DJANGO ASSIGNMENT

Django App – Basic Setup + Model + REST CRUD Views (No DRF)

Django Requirements:

1. Create project:

```
django-admin startproject monitorapp
```

2. Create app:

```
python manage.py startapp api
```

3. Model: SensorRecord

- metric_name (CharField)
- value (FloatField)
- timestamp (DateTimeField auto_now_add)

4. CRUD Views using JsonResponse (NO DRF):

- Create

- List

- Retrieve

- Update

- Delete

5. URLs:

/api/records/ -> List + Create

/api/records// -> Retrieve + Update + Delete

6. Test using browser or Postman

Django Submission:

- Full project

- CRUD screenshots

- README