Technical Round - 1
Coding Assignment (Backend + Reliability Focus)

Languages Allowed: Java, Python, Go, Node.js

Problem Statement: Build a Resilient "User Metadata Service"

Create a backend microservice with two APIs:

API 1 — POST /user

Create a new user with fields:

- user_id
- name
- email
- phone
- created_at

API 2 — GET /user/{id}

Return the stored user metadata.

Mandatory Reliability Requirements

1. Implement **idempotency** (same request should not create duplicates).
2. Add **retry with exponential backoff and jitter when DB writes fail**.
3. Add a **circuit breaker** for the database layer.
4. Record metrics:
    - total_requests
    - success_count
    - failure_count
    - request_latency_ms
5. Log:

    - Request ID
    - Latency
    - Error summary
6. Containerize the service using Docker.

Technical Round - 2
Platform Engineering / Automation Assignment (IDP + CI/CD)

Deliverables: Git repo with Terraform, Jenkinsfile/GitLab CI file, API spec

Problem Statement: Build a Self-Service "Deployment Portal Backend"

Create a small backend service + Terraform automation that allows a developer to:

Feature 1 — Register a New Microservice

- Inputs: service_name, team_name, repo_url
- Automatically create:
    - ECR repo
    - IAM role for service
    - Basic K8s deployment manifest (template)
    - Jenkins/GitLab pipeline YAML

Feature 2 — Trigger a Deployment Job (Simulation Only)

- REST API to trigger a CI job
- Return:
    - build_id
    - status: QUEUED, RUNNING, SUCCESS/FAILED

Feature 3 — Health Dashboard (Simple)

API endpoint returning:

- service_name
- last_deployment_time
- deployment_status
- pod_count
- CPU/Memory usage (static mocked values allowed)

Round - 3 Kubernetes + Observability + SLO Assignment

Deliverables:

- Kubernetes manifests (YAML)
- Helm chart (bonus)
- Dashboards (Datadog/Grafana JSON or screenshots)
- SLO document

Problem Statement: Deploy a Sample App with Production-Grade Standards

Given a simple containerized app (you may use any public app):

Kubernetes Requirements

- Deploy to **EKS-like structure**:
    - Deployment
    - Service
    - HPA (CPU + memory)
    - PodDisruptionBudget
    - Liveness/readiness probes
    - Resource requests/limits

Observability Requirements

Create:

1. A **Metrics Dashboard** showing:
    - Latency (p50/p95/p99)

        - Errors
        - Request volume
        - CPU/Memory usage

2. A **Log pipeline** (sidecar, FluentBit, or any approach)
3. A **Trace example** (Datadog/OpenTelemetry)

SLO Requirements

Prepare a 2–3 page document defining:

- SLO (e.g., 99.9% availability)
- SLIs (latency < 200ms, error rate < 0.1%)
- Error budget
- Alerts →
    - slow burn
    - fast burn
    - high latency (p99)
    - traffic anomalies

Round - 3

Security & Compliance Assignment

Problem Statement: Build a Secure Deployment Design for a FinTech Microservice

Create a 2–3 page design covering:

Requirements

- Secure CI/CD flow (SAST, DAST, secrets scanning)
- IAM design for microservice
- Service-to-service mTLS
- PCI-DSS logging requirements
- Audit log structure (as per PMLA)
- Secrets rotation plan
- SBOM generation