Technical Design Documentation

1. API Service – Source Code

The API Service is a RESTful Spring Boot service providing CRUD operations on the `**student_results**` table with the following schema:

Column	Data Type	Notes
roll_number	BIGINT	Primary key, unique student ID
marks	INT	Student marks
exam_year	INT	Partitioning column (exam year)

Key Features

- **Write Replica:** Handles Create, Update, and Delete operations.
- **Read Replica:** Handles Read operations (10:1 ratio).
- Built with Java 17 (Spring Boot 3.5.4).
- Designed with separate read/write replicas routed via **pgbouncer.**

Image Build & push to dockerhub

```
docker build -t aabidrahman/api-service:latest .

docker push aabidrahman/api-service:latest
```

Project Metadata

- **Build Tool:** Maven
- **Group:** com.examserver
- **Artifact:** api-service
- **Packaging:** JAR
- **Dependencies:** Spring Web, Spring Data JPA, PostgreSQL Driver

Local Testing

Forward ports for API access:

Write API:
minikube kubectl -- port-forward svc/api-service-write 8080:8080 -n exam-server

Read API:
minikube kubectl -- port-forward svc/api-service-read 8081:8080 -n exam-server

2. Infrastructure as Code (IaC)

Two approaches were designed:

● **Local Setup (Used in this project):** Provisioned Kubernetes with Minikube. Scripts & Ansible in iac/ and k8s/

● **Cloud Setup (Remote VM):** Ansible scripts available but not fully implemented due to constraints.

3. Helm Charts

Helm charts located in helm-charts/api-service/.

Includes Deployments for **read** and **write** replicas, Services (-read, -write), and configurable values.yaml.

Deployable with:

helm upgrade --install api-service ./helm-charts/api-service --namespace exam-server --create-namespace --set image.repository=aabidrahman/api-service --set image.tag=latest

4. CI/CD Pipeline

Implemented with **GitHub Actions(CI):**

● **Trigger:** Push to main branch.

● **Steps:** Checkout repository, Set up JDK, Build API, Set up Minikube, Set up Helm, Build Docker image, Load Docker Image into Minikube, Deploy via Helm to Minikube.

● **Notes:** CD was limited due to GitHub secrets restriction.

Zero-downtime Deployment strategy:

● Uses **RollingUpdate** istrategy in Kubernetes Deployments.

● Ensures new pods become ready before old pods terminate.

5. Database Deployment & Partitioning

PostgreSQL Master-worker setup

● Deployed as a StatefulSet with persistent volumes.

● Deployed the Postgres cluster according to the master-worker architecture.

● Configured for replication (replica role + access seperated for read and write).

● Deployed Pgbouncer as a HA connection pooler.

● 1M rows inserted with random marks & unique roll_numbers via script, available in /sql along with row counter scripts.

● Partitioning: student_results table partitioned by exam_year (script in /sql).

6. Scaling Configuration

● Read replicas **autoscaled** using **HPA**.

● Write replicas kept smaller due to lower volume.

● **Pgbouncer** replicas deployed for **HA**.

7. Observability Setup

● **Prometheus + Grafana** deployed with Helm.

● Grafana dashboards exported to docs/observability/.

● Access Grafana via: minikube service grafana -n monitoring.

8. Deliverables

● **Source Code:** api-service/ (Spring Boot + Dockerfile).

● **IaC:** iac/ scripts + k8s/ manifests.

● **Helm Charts:** helm-charts/api-service/.

● **CI/CD Config:** .github/workflows/github-actions.yaml, ci-cd/.

● **Database SQL:** sql/student_results.sql (table create,partition,insert)

● **Observability:** docs/observability/ with Grafana JSON