1. Objective

Referring to "Kubernetes Project Assignment_ FlaskApp with Database task", Build a complete CI/CD workflow for a Flask application backed by MySQL. Containerize it, test it, and deploy it to Kubernetes using Kustomize with separate overlays for testing and production environments.

2. Part I: CI/CD and Testing

CI Pipeline

- Use **GitHub** Actions or **GitLab** CI.
- Trigger on push to **main** and **pull requests.**
- **Pipeline Stages:**
    1. Run unit tests , generate and upload coverage report.
    2. Lint with flake8.
    3. Build and push Docker images (FlaskApp & MySQL).
    4. CD to k8s.
- Use **pytest** to write unit tests (routes, DB, logic).
- Generate a coverage report using **coverage.py** (target: ≥ **80%**).
- Push Docker images to dockerhub or any docker registries.

3. Part II: Kubernetes Deployment with Kustomize

Cluster Setup

- Use kubeadm, k3s, microk8s or a cloud-based cluster provider.
- Minikube is not allowed.
- Minimum nodes: 1 master and 1 or more worker nodes.

Kustomize Base and Overlays

Kubernetes directory structure:

k8s/

```
├── base/
│   ├── deployment.yaml
│   ├── service.yaml
│   ├── configmap.yaml
│   ├── pvc.yaml
│   ├── ingress.yaml
│   └── kustomization.yaml
└── overlays/
    ├── testing/
    └── production/
```

Environment Configuration Differences:

Feature	Testing	Production
Replicas	1	3
CPU Limit	200m	500m
Memory	256Mi	1024Mi

Secrets and ConfigMaps

- Use Kustomize secret and config generator.
- Store DB credentials and DockerHub credentials securely using secrets.
- Do not commit plaintext secrets to version control.

Ingress Configuration

- Use ingress-nginx or an equivalent ingress controller.
- Route path `/flask` to the FlaskApp service.
- Ensure correct annotations for path rewrite and access.

Kubernetes Enhancements

- Implement liveness and readiness probes for FlaskApp.
- Use PersistentVolumeClaim (PVC) for MySQL to enable data persistence.
- Define resource limits and requests.
- Use image pull secrets for private DockerHub access.
- Optional Features:
    - NetworkPolicy.
    - StorageClass (e.g., NFS, local-path).

4. Part III: Validation & Testing

- Validate FlaskApp ↔ MySQL database connectivity
- Confirm ingress path `/flask` is accessible

- Verify resource limits are correctly applied per environment
- Ensure secrets are injected and used properly
- Verify data persists in MySQL after restarts

5. Part IV: Submission Checklist

- GitHub/Gitlab repository should include:
    - Dockerfiles
    - CI configuration file (.github/workflows/ci.yml)
    - tests/ directory with pytest and coverage
    - Kubernetes manifests in k8s/base and k8s/overlays/
- README.md must include documentation for all steps.