🚀 MLOps Assignment - Complete Execution Guide

✅ Project Validation Against Requirements

Task Completion Status

Task	Requirement	Files/Evidence

1. Data Acquisition & EDA	✅ Download script
	✅ Data cleaning
	✅ Preprocessing
	✅ Professional visualizations
		`src/download_data.py`
		`notebooks/01_EDA.ipynb`
		`src/preprocessing.py`

2. Feature Engineering & Models	✅ Scaling/encoding ✅ 2+ models (LR, RF, GB) ✅ Cross-validation ✅ Metrics evaluation	`src/train.py` `src/preprocessing.py`
3. Experiment Tracking	✅ MLflow integration ✅ Log params/metrics ✅ Artifacts storage	`src/train.py` (lines 50-200)

4. Model Packaging	✅ Saved models
	✅ requirements.txt
	✅ Preprocessing pipeline
		`models/best_model.pkl`
		`models/preprocessor.pkl`
		`requirements.txt`

5. CI/CD Pipeline	✅ Unit tests ✅ GitHub Actions ✅ Linting/testing ✅ Artifacts/logging	`tests/` folder `.github/workflows/ci-cd.yml`

6. Containerization	✅ Docker container
	✅ FastAPI with /predict
	✅ JSON input/output
		`Dockerfile`
		`src/app.py`
		`sample_input.json`

7. Production Deployment	✅ K8s manifests ✅ Load Balancer/Ingress ✅ Deployment instructions	`deployment/kubernetes/` `CLOUD_DEPLOYMENT_GUIDE.md`

8. Monitoring & Logging	✅ API logging
	✅ Prometheus + Grafana
		`src/app.py` (logging)
		`deployment/kubernetes/monitoring.yaml`
		`docker-compose.yml`

Task	Requirement	Files/Evidence

9. Documentation	✅ Setup instructions
	✅ Architecture diagram
	✅ Screenshots folder
		`README.md`
		`docs/`
		`screenshots/`

✅ **All requirements implemented!**

📋 Step-by-Step Execution Guide

Phase 1: Environment Setup

Step 1.1: Navigate to Project Directory

```
# Navigate to the heart-disease-mlops directory
cd heart-disease-mlops
```

Step 1.2: Run Setup Script

```
chmod +x setup.sh
./setup.sh
```

This will:

- Create Python virtual environment
- Install all dependencies from requirements.txt
- Create necessary directories
- Set up the environment

Step 1.3: Activate Virtual Environment

```
source venv/bin/activate
```

Phase 2: Data Acquisition & EDA

Step 2.1: Download Dataset

```
python src/download_data.py
```

Expected Output:

```
✓ Data downloaded successfully!
✓ Dataset saved to data/processed/heart_disease.csv
✓ Raw data copied to data/raw/
Dataset shape: (303, 14)
```

Step 2.2: Execute EDA Notebook

1. Open `notebooks/01_EDA.ipynb` in VS Code
2. **Run All Cells** (Shift + Enter on each cell or Run All)

Phase 3: Model Training & Experiment Tracking

Step 3.1: Train Models with MLflow

```
python src/train.py
```

Expected Output:

```
✓ Data loaded successfully
✓ Preprocessing pipeline created
Training Logistic Regression...
✓ Logistic Regression - Accuracy: 0.85, ROC-AUC: 0.90
Training Random Forest...
✓ Random Forest - Accuracy: 0.88, ROC-AUC: 0.92
Training Gradient Boosting...
✓ Gradient Boosting - Accuracy: 0.87, ROC-AUC: 0.91
✓ Best model saved to models/best_model.pkl
✓ Preprocessor saved to models/preprocessor.pkl
```

Step 3.2: View MLflow Experiments

```
mlflow ui
```

Then open browser: `http://localhost:5000`

Phase 4: Testing

Step 4.1: Run Unit Tests

```
pytest tests/ -v --cov=src --cov-report=html
```

Expected Output:

```
tests/test_preprocessing.py ✓✓✓✓✓✓✓✓✓✓ (10 passed)
tests/test_model.py ✓✓✓✓✓✓✓✓ (8 passed)
tests/test_api.py ✓✓✓✓✓✓✓✓✓✓✓✓ (12 passed)

Total: 30 tests passed
Coverage: 85%
```

Step 4.2: View Coverage Report

```
open htmlcov/index.html
```

Phase 5: API Testing

Step 5.1: Start API Server

```
uvicorn src.app:app --reload --host 0.0.0.0 --port 8000
```

Expected Output:

```
INFO: Uvicorn running on http://0.0.0.0:8000
INFO: Application startup complete
```

Step 5.2: Test API Endpoints

Terminal 2 - Health Check:

```
curl http://localhost:8000/health
```

Expected: `{"status":"healthy","model_loaded":true}`

Terminal 2 - Single Prediction:

```
curl -X POST http://localhost:8000/predict \
  -H "Content-Type: application/json" \
```

```
  -d @sample_input.json
```

Expected Response:

```
{
  "prediction": 1,
  "prediction_label": "Heart Disease",
  "probability": 0.78,
  "model_version": "1.0.0"
}
```

Step 5.3: Access API Documentation

Open browser: `http://localhost:8000/docs`

Screenshots:

- Swagger UI
- /predict endpoint test
- Response JSON

Phase 6: Docker Containerization

Step 6.1: Build Docker Image

```
docker build -t heart-disease-mlops:latest .
```

Step 6.2: Run Docker Container

```
docker run -d -p 8000:8000 --name heart-disease-api heart-disease-
mlops:latest
```

Step 6.3: Test Containerized API

```
# Wait 10 seconds for startup
sleep 10

# Test health endpoint
curl http://localhost:8000/health

# Test prediction
curl -X POST http://localhost:8000/predict \
```

```
  -H "Content-Type: application/json" \
  -d @sample_input.json
```

Step 6.4: View Container Logs

```
docker logs heart-disease-api
```

Step 6.5: Stop Container

```
docker stop heart-disease-api
docker rm heart-disease-api
```

Phase 7: Monitoring Stack

Step 7.1: Start Prometheus + Grafana

```
docker-compose up -d
```

Step 7.2: Access Dashboards

- **Prometheus:** http://localhost:9090
- **Grafana:** http://localhost:3000 (admin/admin)

Step 7.3: Generate API Traffic

```
# Run this script to generate requests
for i in {1..100}; do
  curl -X POST http://localhost:8000/predict \
    -H "Content-Type: application/json" \
    -d @sample_input.json
  sleep 0.1
done
```

Step 7.4: View Metrics

In Grafana:

1. Add Prometheus data source: http://prometheus:9090
2. Import dashboard or create custom panels
3. View metrics: request_count, request_duration, prediction_count

Phase 8: Kubernetes Deployment

Option A: Local Minikube

Step 8.1: Start Minikube

```
minikube start --driver=docker
```

Step 8.2: Load Docker Image

```
minikube image load heart-disease-mlops:latest
```

Step 8.3: Deploy to Kubernetes

```
kubectl apply -f deployment/kubernetes/deployment.yaml
kubectl apply -f deployment/kubernetes/ingress.yaml
```

Step 8.4: Check Deployment

```
kubectl get deployments
kubectl get pods
kubectl get services
```

Step 8.5: Access Service

```
minikube service heart-disease-api
```

Option B: Cloud Deployment (GKE/EKS/AKS)

See detailed instructions in `CLOUD_DEPLOYMENT_GUIDE.md`

Phase 9: CI/CD Validation (5 minutes)

Step 9.1: Review GitHub Actions Workflow

```
cat .github/workflows/ci-cd.yml
```

Step 9.2: Push to GitHub

```
git init
git add .
git commit -m "Complete MLOps pipeline implementation"
git branch -M main
git remote add origin <your-repo-url>
git push -u origin main
```

Step 9.3: Monitor Workflow

Go to GitHub → Actions tab → View workflow runs
