# Heart Disease Prediction: MLOps End-to-End Implementation Report

**Submitted By:**

* **Gaurav Sharma** (2024AB05112)
* **Harshitha S** (2024AA05282)
* **NA.R. Yangyeswar** (2024AA05285)
* **Pratima Chauhan** (2024AB05324)
* **Hathi Shivani Niren** (2024AB05014)

## 1. Project Overview

This project implements a scalable, reproducible, and production-ready Machine Learning solution to predict the risk of heart disease based on patient health data. It utilizes the **Heart Disease UCI Dataset** and adheres to modern **MLOps** best practices for seamless deployment and monitoring.

### 1.1 Project Deliverables

The following resources and links are essential to the project's implementation:

* **GitHub Repository:** [heart-disease-app](https://github.com/2024ab05112/heart-disease-app.git)
* **Live Web Application:** [Health Prediction Portal](http://heart-disease-2024ab05112.centralindia.cloudapp.azure.com/)
* **Project Demo & Description Video:** [Watch Video on Google Drive](https://drive.google.com/file/d/1EAkUQg3R94hodZxZxqRHMX2v1R3LgmU4/view?usp=sharing)
* **Interactive API Documentation:** [FastAPI Swagger Docs](http://heart-disease-2024ab05112.centralindia.cloudapp.azure.com/api/docs)
* **System Monitoring (Grafana):** [Grafana Dashboard](http://heart-disease-2024ab05112.centralindia.cloudapp.azure.com/grafana/)
* **Metrics & Alerts (Prometheus):** [Prometheus Server](http://heart-disease-2024ab05112.centralindia.cloudapp.azure.com/prometheus/)

## 2. System Architecture

The system is deployed on **Azure Kubernetes Service (AKS)** using a microservices-based architecture, optimized for high availability, scalability, and observability.

### 2.1 Architecture Diagram

graph TD  
 User((User)) -->|HTTPS| Proxy[Django Reverse Proxy]  
 Proxy -->|Routing| Backend[Backend API Pods]  
 Proxy -->|Static Serving| Frontend[Frontend Pods]  
 Backend -->|Log Experiments| MLflow[(MLflow Server)]  
 Backend -->|Expose Metrics| Prometheus[Prometheus]  
 Prometheus -->|Visualize| Grafana[Grafana]  
   
 subgraph "Production Environment (AKS Cluster)"  
 Proxy  
 Backend  
 Frontend  
 Prometheus  
 Grafana  
 end

## 3. Data Science & Experimentation

### 3.1 Exploratory Data Analysis (EDA)

* **Visualizations:** Histograms were generated for feature distributions, along with a correlation heatmap to identify key predictors.
* **Insights:** Significant correlations were observed between the target variable and features such as chest pain type (cp) and maximum heart rate (thalach).
* **Data Quality:** Verified zero missing values and implemented categorical variable encoding.

### 3.2 Modelling Strategy

* **Baseline Model:** Used **Logistic Regression** for its interpretability and ability to capture linear relationships.
* **Ensemble Model:** Trained a **Random Forest Classifier** to capture complex, non-linear patterns in the data.
* **Evaluation:** Employed 5-fold cross-validation to ensure model robustness, focusing primarily on **F1-Score** and **ROC-AUC** metrics.

### 3.3 Experiment Tracking (MLflow)

Every training iteration was tracked using **MLflow**:

* **Parameters:** Logged hyperparameters such as max\_depth and n\_estimators.
* **Metrics:** Monitored Accuracy and AUC-ROC curves in real-time.
* **Artifacts:** Serialized model files (.pkl) and evaluation plots were versioned and stored securely.

## 4. MLOps: CI/CD & Deployment

### 4.1 CI/CD Pipeline Design

The project lifecycle is fully automated via **GitHub Actions**:

1. **Continuous Integration (CI):** Every Pull Request triggers code linting (flake8) and unit testing (pytest).
2. **Containerization:** Upon successful testing, Docker builds for the API and Frontend images are created and tagged with the unique GitHub SHA.
3. **Continuous Deployment (CD):** Images are pushed to Docker Hub, and Kubernetes manifests are dynamically updated before being applied to the AKS cluster.

### 4.2 Verification Workflow

Deployment success can be verified through the following:

* **Pipeline Logs:** "Green" status in GitHub Actions.
* **API Health:** Accessing the live FastAPI Swagger Documentation endpoint.
* **Observability:** Monitoring pod health, resource usage, and traffic via the Grafana dashboard.

## 5. Setup & Repository Access

### 5.1 SSH Repository Access

Personal SSH keys are provided in the .ssh/ directory for secure authenticated operations.

[!IMPORTANT]

Security Note: These SSH keys are included in this project package specifically for evaluation purposes and are NOT available on the public GitHub repository.

**Instructions:**

chmod 600 .ssh/id\_rsa  
eval "$(ssh-agent -s)"  
ssh-add .ssh/id\_rsa  
ssh -T git@github.com # Verify the connection

### 5.2 Local Execution

1. **Clone & Setup:**  
   git clone git@github.com:2024ab05112/heart-disease-app.git  
   cd heart-disease-app/backend  
   python -m venv venv && source venv/bin/activate  
   pip install -r requirements.txt
2. **Docker Deployment:**  
   docker-compose up --build