Project Report

1. Introduction

This project demonstrates the implementation of a **Continuous Integration and Continuous Deployment (CI/CD)** pipeline using **GitHub Actions**. The pipeline automates the build, test, containerization, and deployment of a **React-based web application** into a **Kubernetes cluster**.

The main goal of this project is to showcase how modern DevOps practices streamline software delivery, reduce manual intervention, and ensure reliable deployments.

2. Objectives

- Automate build and deployment using GitHub Actions.
- Containerize the React application with Docker.
- Deploy the application in Kubernetes using manifests.
- Provide a simple, reusable template for CI/CD pipelines.

3. Technology Stack

- **Frontend:** React (JavaScript)
- **Version Control:** GitHub
- **CI/CD:** GitHub Actions
- **Containerization:** Docker
- **Orchestration:** Kubernetes (Minikube/Cluster)
- **Tools:** Node.js, kubectl, npm

4. System Architecture

Workflow Overview:

1. Developer pushes code to GitHub repository.
2. GitHub Actions triggers CI/CD pipeline.
3. Pipeline builds & tests React application.
4. Docker image is built and pushed.
5. Kubernetes manifests are applied to deploy the app.

5. Repository Structure

- src/ – React application source code
- Dockerfile – Build instructions for Docker image

- .github/workflows/ci-cd.yml – GitHub Actions pipeline definition
- k8s-manifests/ – Deployment & service files for Kubernetes
- package.json – Project metadata and dependencies
- README.md – Documentation and usage guide

6. CI/CD Pipeline Stages

- **Build:** Install dependencies and build React app.
- **Test:** Run unit tests (if implemented).
- **Dockerize:** Build Docker image and push to registry.
- **Deploy:** Apply Kubernetes manifests to deploy application.

7. Features & Highlights

- Fully automated CI/CD pipeline.
- Demonstrates containerized deployment.
- Portable and reusable for similar React apps.
- Easy integration with cloud-native environments.

8. How to Use

1. Clone the repository.
2. Install dependencies using npm install.
3. Run locally with npm start.
4. Push changes to GitHub → CI/CD pipeline runs automatically.
5. Deploy application on Kubernetes cluster using kubectl apply.

9. Applications

- Learning **CI/CD concepts** with GitHub Actions.
- Demonstrating DevOps practices for academic or professional purposes.
- Serving as a **starter template** for React projects with Kubernetes deployment.

10. Conclusion

This project provides a practical demonstration of how to implement CI/CD pipelines for modern applications. By combining GitHub Actions, Docker, and Kubernetes, it delivers a **scalable, automated, and production-ready workflow**. The template can be easily extended with additional features such as testing frameworks, quality checks, and multi-environment deployments.