**Introduction**

Continuous Integration and Continuous Deployment (CI/CD) are essential practices in modern DevOps workflows, ensuring rapid and reliable delivery of software. This project demonstrates a complete CI/CD pipeline using **GitHub Actions** and **Docker**, deploying the application locally without any cloud services. The pipeline automates code testing, container image building, and deployment to a local environment using **Minikube** or a local VM.

**Abstract**

The goal of this project is to build a seamless automated pipeline that integrates code changes, builds Docker images, pushes them to Docker Hub, and deploys the updated application locally. By leveraging GitHub Actions, every code commit triggers an automated workflow that ensures the application is always in a deployable state. This eliminates manual intervention, reduces errors, and improves delivery speed.

**Tools Used**

1. **GitHub Actions** – For automating build, test, and deployment workflows.
2. **Docker** – For containerizing the application.
3. **Docker Hub** – To store and share Docker images.
4. **Minikube / Local VM** – To simulate deployment on a local Kubernetes cluster or virtual environment.
5. **Git & GitHub** – For version control and CI/CD workflow management.

**Steps Involved in Building the Project**

1. **Dockerfile Creation**  
   A Dockerfile was written to define the environment, dependencies, and startup commands for the application.
2. **docker-compose.yml Setup**  
   Used to simplify local multi-container deployment and testing.
3. **GitHub Actions Workflow Configuration**
   * Added a .github/workflows/ci-cd.yml file.
   * Steps included:
     + Checkout repository
     + Build Docker image
     + Run tests
     + Push image to Docker Hub
4. **Docker Hub Integration**  
   Configured Docker Hub credentials as GitHub Secrets (DOCKER\_USERNAME, DOCKER\_PASSWORD).
5. **Deployment on Local Environment**
   * Pulled the Docker image using:
   * docker pull <username>/<image-name>:latest
   * Deployed it using Minikube or Docker Compose.
   * Verified deployment via kubectl get pods or docker ps.
6. **Validation and Testing**  
   Ensured successful image build, test pass, and local deployment by monitoring logs and GitHub Action status.

**Conclusion**

This project successfully established a full **CI/CD pipeline** without relying on cloud platforms. It automated code integration, image building, and local deployment using **GitHub Actions** and **Docker**. The setup demonstrates how DevOps automation can be achieved efficiently with open-source tools, improving reliability and developer productivity.