Manufacturing Operational Insights Report

LogiTrack

Course Name: DevOps

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrollment Number(s):

Sr no	Student Name	Enrolment Number
1.	Vasu Joshi	EN22EL301063
2.	Yash Kag	EN22CS3011103
3.	Yash Parashar	EN22CS3011106
4.	Yashee Verma	EN22CS3011115
5.	Yashraj Singh Sisodiya	EN22CS3011120

Group Name: Group 12D10

Project Number: DO-25

Industry Mentor Name:

University Mentor Name: Prof. Avnesh Joshi

Academic Year: 2025-26

1. Problem Statement & Objectives:

1.1 Problem Statement

CI/CD for Python Logistics App

In modern software development, deploying applications manually is time-consuming and error-prone. Logistics applications require continuous updates, fast delivery, and stable deployment because they handle important operations such as order processing, shipment tracking, and inventory management.

The problem is that traditional deployment methods do not support:

- Automated testing
- Continuous integration
- Fast and reliable delivery
- Error-free deployments

Therefore, there is a need to build a **CI/CD pipeline** for a Python-based Logistics Application that can automate the process of building, testing, containerizing, and deploying the application efficiently on cloud infrastructure.This project focuses on implementing a complete **CI/CD pipeline using GitHub Actions, Docker, and AWS EC2 deployment** for a Logistics application.

1.2 Project Objectives:

The main objectives of this project are:

- **To develop a Python-based logistics application** that simulates logistics operations such as order management and inventory flow.
- **To implement Continuous Integration (CI)** so that whenever code is pushed to GitHub, automated build and testing processes run.
- **To implement Continuous Deployment (CD)** so that the application can be deployed automatically without manual intervention.
- **To containerize the application using Docker** for portability and consistency across environments.
- **To deploy the application on AWS EC2 instance**, making it accessible through the cloud.
- **To reduce deployment time and human errors** by automating the software delivery lifecycle.

1.3 Scope of the Project:

The scope of this project includes:

- Building a logistics management backend application using Python
- Automating the CI/CD workflow using GitHub Actions
- Creating Docker images for consistent deployment
- Deploying the application on AWS EC2 infrastructure

- Ensuring smooth integration between development and deployment environments
- The project is limited to:
- Basic logistics operations simulation
- Deployment through Docker containers
- Single EC2 instance deployment (not multi-cloud or large-scale production)

2. Proposed Solution:

2.1 Key Features:

The key features of this CI/CD Logistics project are:

- **Automated Build and Testing** using GitHub Actions
- **Docker Containerization** of the Python application
- **Continuous Deployment** on AWS EC2 server
- **Version Control Integration** through GitHub
- **Reliable Application Delivery Pipeline**
- **Scalable Deployment Approach** using cloud infrastructure

2.2 Overall Architecture/Workflow:

The project follows a complete DevOps-based workflow:

1. **Developer writes code** for the logistics application.
2. Code is pushed to the GitHub repository
3. GitHub Actions triggers the CI/CD pipeline automatically
4. The pipeline performs:
    - Code Checkout
    - Automated Testing
    - Docker Image Build
5. The Docker container is deployed on an **AWS EC2 instance**
6. The application becomes available to users through EC2 public IP
7. Monitoring and verification ensure successful deployment

This workflow improves deployment speed, reliability, and consistency.

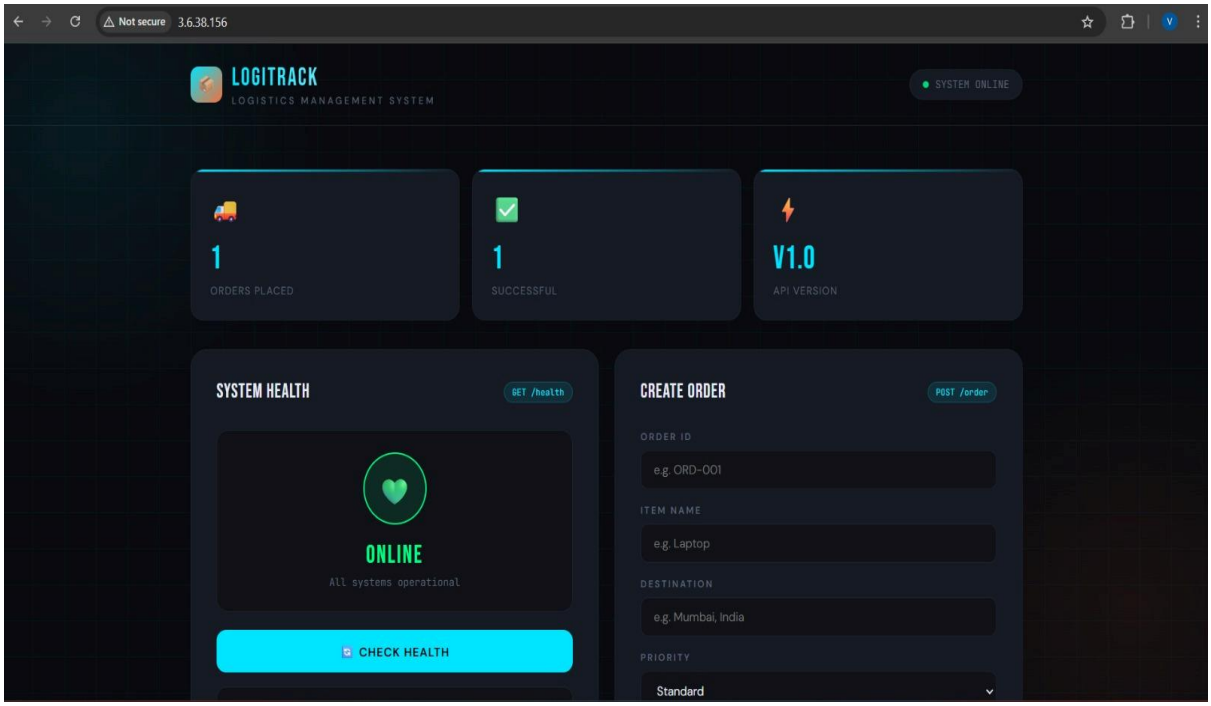
2.3 Tools & Technologies Used:

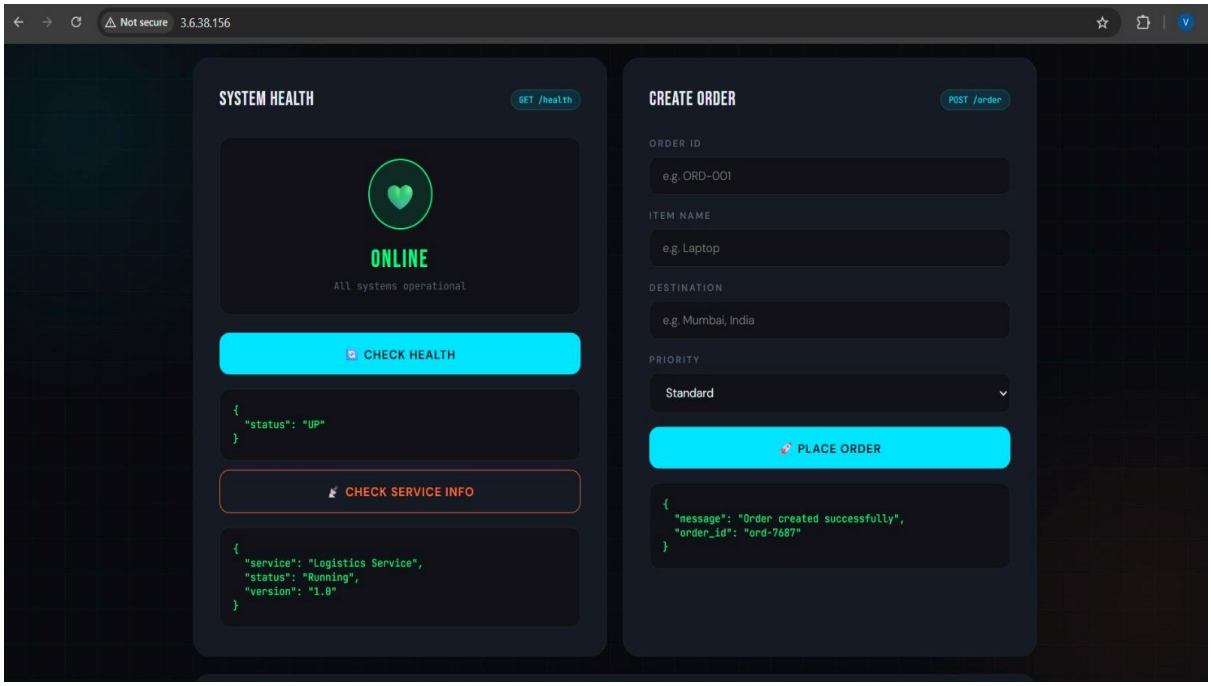
Tool/Technology	Purpose
Python	Application development
GitHub	Source code management
GitHub Actions	CI/CD automation
Docker	Containerization of application

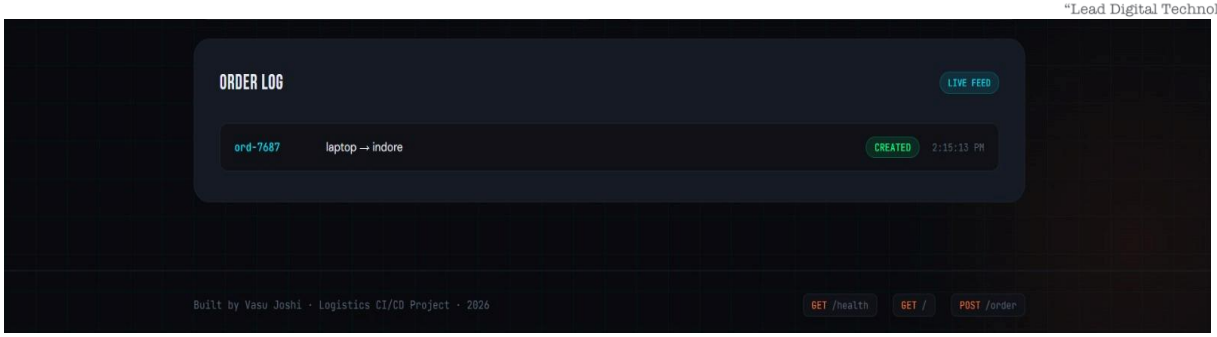
Tool/Technology	Purpose
AWS EC2	Cloud deployment server
Linux (Ubuntu)	EC2 operating environment
YAML	Workflow configuration for pipeline

3. Result & Output:

3.1 Screenshots/Outputs:







3.2 Reports /Dashboards /Models:

This project provides:
GitHub Actions execution logs as CI/CD reports
Docker container status reports using:
docker ps
Deployment monitoring through EC2 instance logs**:**
docker logs <container-id>

3.3 Key Outcomes:

The major outcomes achieved are:

- Successful automation of CI/CD pipeline
- Docker-based deployment ensures consistent runtime environment
- Reduced manual deployment effort
- Application successfully deployed on AWS EC2
- Improved reliability and faster software delivery
- Practical understanding of DevOps tools and cloud deployment

4. Conclusion:

This project successfully demonstrates the implementation of a CI/CD pipeline for a Python-based Logistics Application using GitHub Actions, Docker, and AWS EC2 deployment.
The project helped in understanding real-world DevOps practices such as:

- Continuous Integration
- Automated Testing
- Containerization
- Cloud Deployment

Overall, the solution improves deployment efficiency, reduces errors, and provides a scalable foundation for future logistics application development

5. Future Scope & Enhancements:

The project can be enhanced further by adding:

- **Advanced Monitoring Tools:** Prometheus, Grafana, CloudWatch
- **Load Balancing & Auto Scaling:** Deploying across multiple EC2 instances

- **Kubernetes Deployment:** Using EKS for container orchestration
- **Database Integration:** Adding RDS or DynamoDB for real logistics data storage
- **Security Improvements:** IAM roles, HTTPS, secrets management
- **Production-Grade CI/CD:** Blue-green deployment and rollback support