



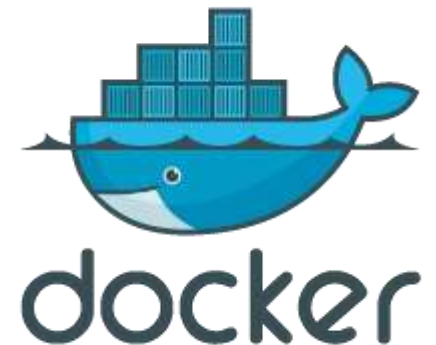
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
LOVELY PROFESSIONAL UNIVERSITY
JALANDHAR

DevOps Virtualization and Configuration Management
(INT-332)

Readers Garden – Library Management System

Containerized Infrastructure with CI/CD Pipeline

Submitted By:
Name – Ayan Yadav
Section- KO129
Roll no.- 22
Reg no.-12222524



PROJECT OVERVIEW

- Developed a full-stack **Library Management System** to streamline book lending and inventory tracking.
- Frontend built using **React** with **Vite** for enhanced speed and development experience.
- Backend developed with **Node.js** and **Express**, managing API requests and server logic.
- Includes a user interface for **browsing books, borrowing/returning, and managing user accounts**, with **MongoDB** as the database.
- Features an **admin panel** for managing book inventory, tracking borrow history, and overseeing user activities in real-time..
- Containerized the application using **Docker** to run frontend and backend as separate containers.
- Implemented a **CI/CD** pipeline using **GitHub Actions** for automation.
- Deployed on **Vercel** for fast, reliable delivery and version control.
- Demonstrates practical skills in full-stack development, containerization, and DevOps.

PROBLEM STATEMENT AND SOLUTION

PROBLEMS:-

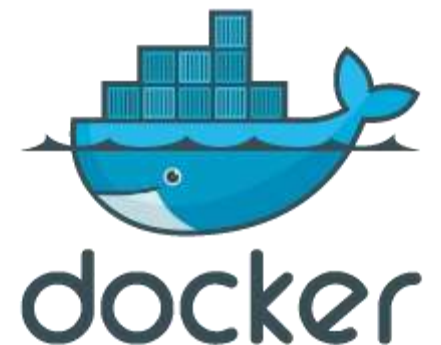
- Outdated Interface that result in poor user experience.
- Manual deployment process that slow down the rollout of new features.
- Inconsistence environment between deployment and production causing unexpected errors.
- Lack of automation testing and Integration.

SOLUTIONS:-

- Developing a user friendly and modern website using MERN stack.
- Containerizing the application using Docker to ensure consistence environment across the deployment.
- Implementing CI/CD pipeline for automated testing.
- The Combination of containerization and CI/CD will streamline the development without disrupting user experience.

TECHNOLOGIES STACK

CATEGORY	TECH USED
FRONTEND	REACT with VITE
BACKEND	NODE JS + EXPRESS
DATABASE	MONGO-DB
CONTAINERIZATION	DOCKER
CI/CD	GITHUB ACTIONS





PROJECT ARCHITECTURE

CONTAINERIZATION AND CI/CD PIPELINES

Docker Containerization

- Containerized both frontend and backend as separate services.
- Used separate Dockerfile for each and one docker-compose.yml in root directory.
- Ensures environment consistency across all stages.

CI/CD Pipeline

GitHub Actions:

- Triggers on every push and pull request on the master branch.
- Runs tests and builds the source code and other required libraries.

CHALLENGES FACED AND LEARNING OUTCOMES

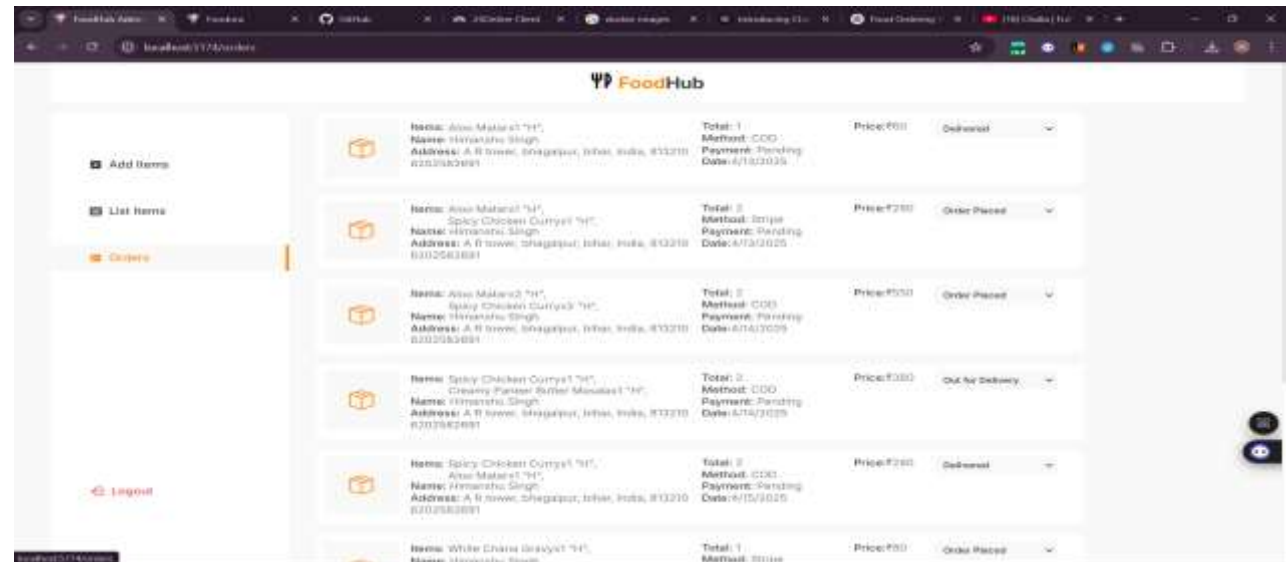
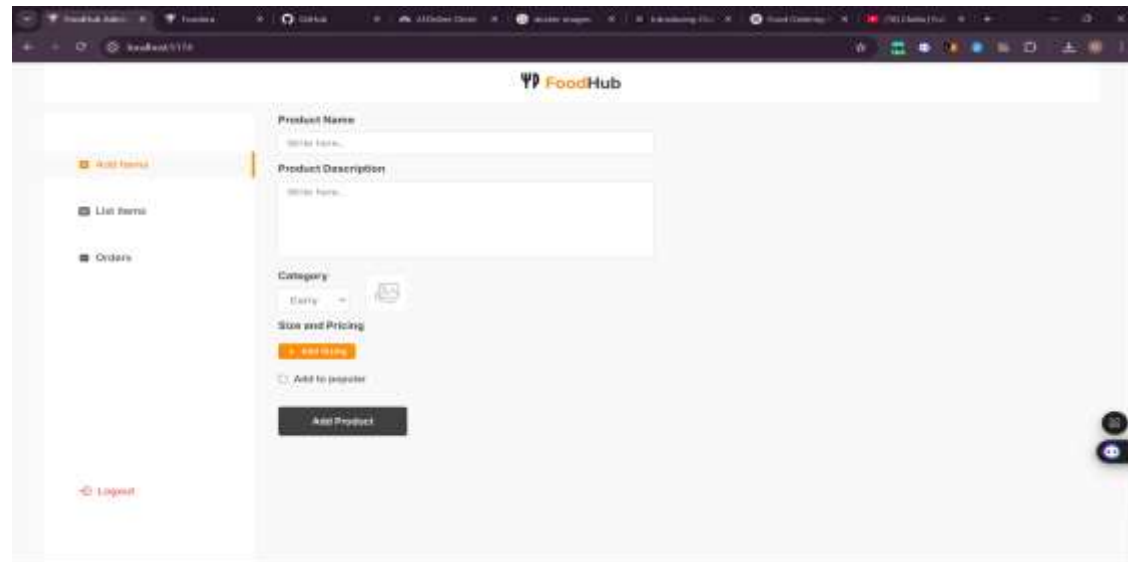
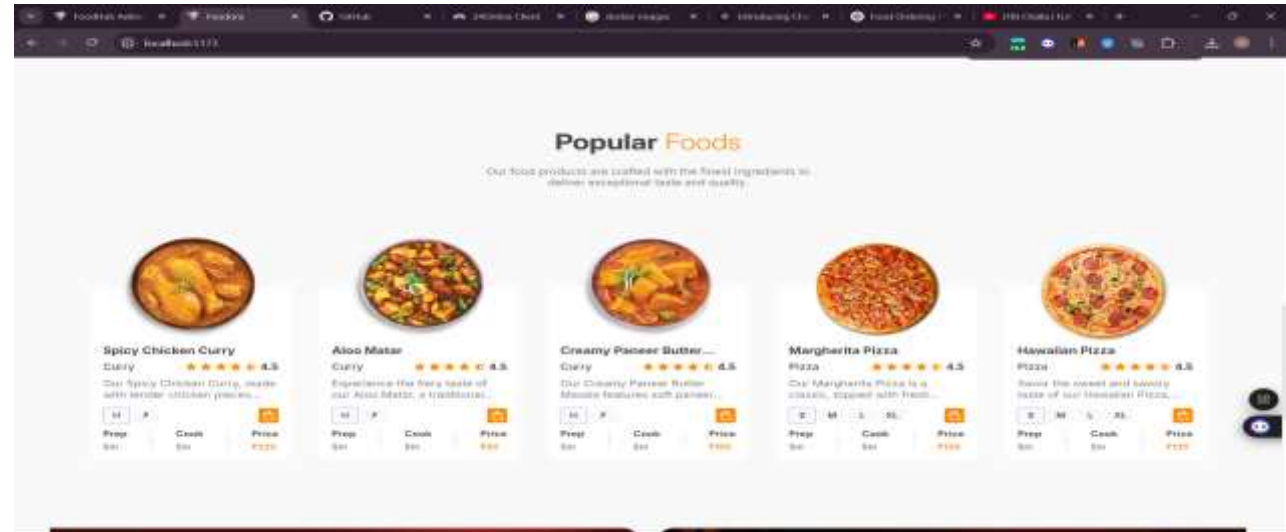
Challenges Faced:-

- Integrating Docker with Vite & Node.js.
- Managing secrets during deployment.
- Implementing CI/CD via GitHub Actions.

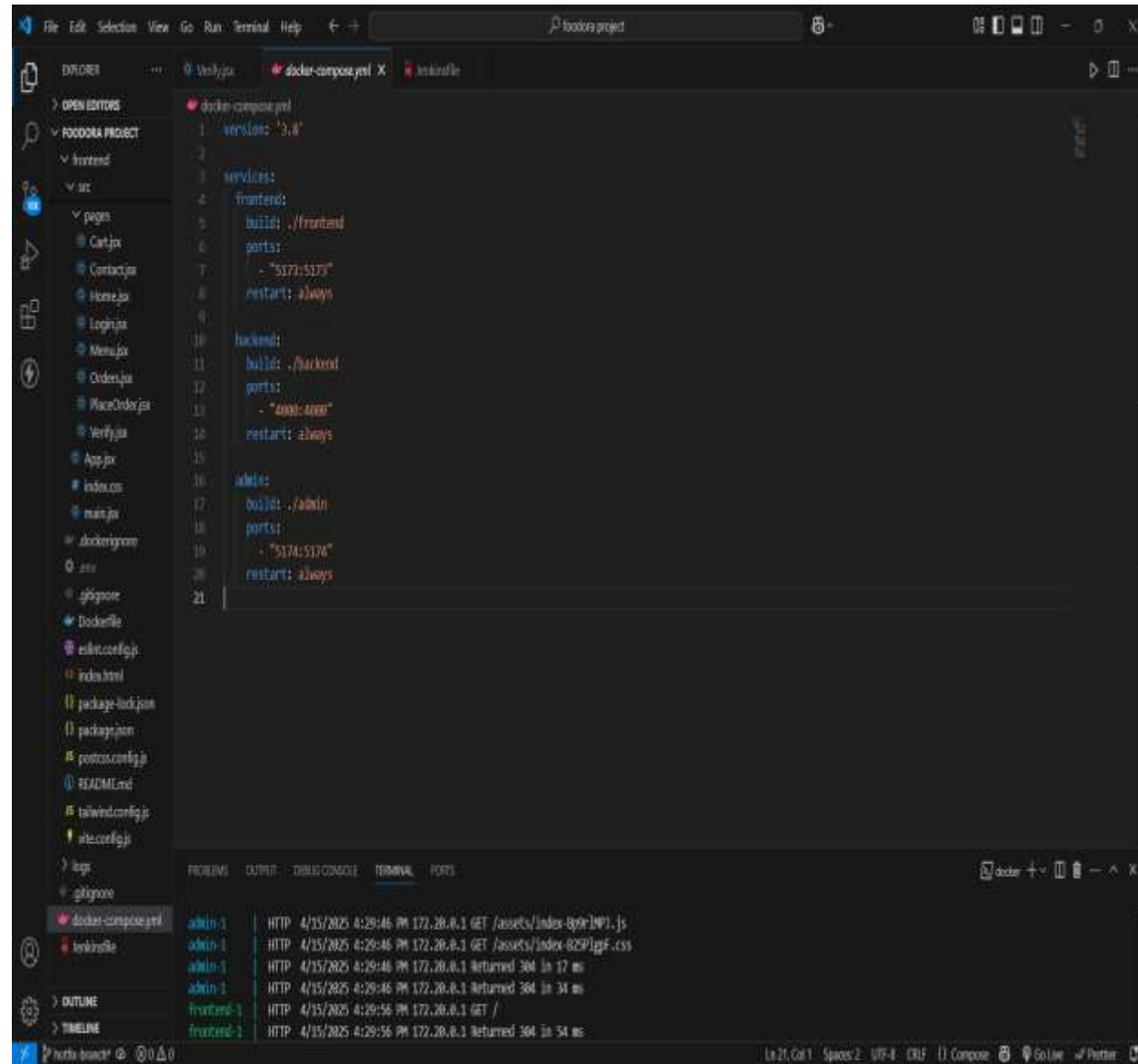
Learning Outcomes:-

- Hands-on experience with CI/CD.
- Real-world DevOps concepts: Docker, GitHub Actions.
- Improved full-stack development and deployment understanding.

SNAPSHOTS OF PROJECT AND CODE



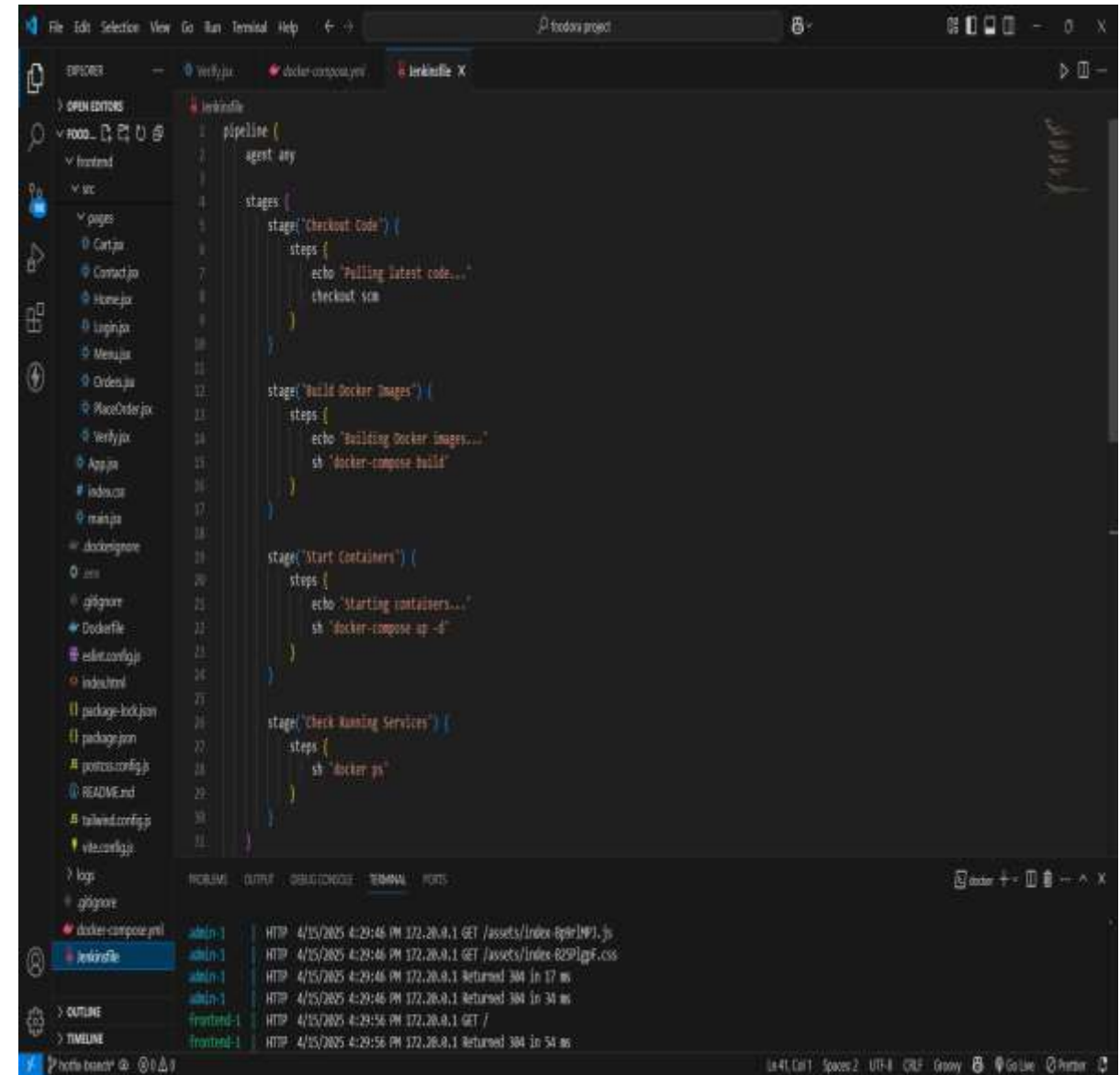
SNAPSHOT OF DOCKER-COMPOSE.YML FILE



```

1 version: '3.8'
2
3 services:
4   frontend:
5     build: ./frontend
6     ports:
7       - "5174:5174"
8     restart: always
9
10  backend:
11    build: ./backend
12    ports:
13      - "3000:3000"
14    restart: always
15
16  admin:
17    build: ./admin
18    ports:
19      - "5174:5174"
20    restart: always
  
```

SNAPSHOT OF JENKINSFILE



```





1 pipeline {
2   agent any
3
4   stages {
5     stage('Checkout Code') {
6       steps {
7         echo "Pulling latest code..."
8         checkout scm
9       }
10    }
11
12    stage('Build Docker Images') {
13      steps {
14        echo "Building Docker Images..."
15        sh "docker-compose build"
16      }
17    }
18
19    stage('Start Containers') {
20      steps {
21        echo "Starting containers..."
22        sh "docker-compose up -d"
23      }
24    }
25
26    stage('Check Running Services') {
27      steps {
28        echo "Checking services..."
29        sh "docker ps"
30      }
31    }
32  }
  
```

CONCLUSION AND FUTURE SCOPE

Conclusion:-

- Successfully built and deployed a scalable food ordering platform.
- Integrated modern DevOps practices like Docker and CI/CD to ensure smooth and automated deployments.
- The project is optimized for real-world use, with features for both users and admins, and a clean deployment pipeline for continuous updates.

Future Scope:-

-  Payment Gateway Integration
- Enables secure online transactions for seamless order processing.
-  Delivery Tracking
- Tracks order status in real-time, improving transparency and customer experience.
-  SMS/Email Notifications
- Sends instant updates to customers about order confirmation, dispatch, and delivery.
-  AI-Powered Suggestions (If implemented or planned)
- Suggests popular items or upsells based on user behavior (can be basic rule-based or ML-based).