Project - High Level Design

on

CI/CD for Vue.js Technology App

Course Name: DevOps FOUNDATION

Institution Name: Medicaps University – Datagami Skill Based Course

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

Sr no	Student Name	Enrolment Number
1.	ANIKET JAT	EN22CS304010
2.	SHRIYANSH GUHA	EN22CS303054
3.	HIMANSHU SAHRAWAT	EN22CS304029
4.	YASHASVI BAKORE	EN22CS303059
5.	ROUNAK PATHEKAR	EN22CS303046

Group Name: Group 03D11

Project Number: DO-29

Industry Mentor Name:

University Mentor Name: Prof. Shyam Patel

Academic Year:2022-2026

MEDICAPS UNIVERSITY

Datagami
"Lead Digital Technology"

Table of Contents

1. Introduction

1.1 Scope of the Document

This document describes the design and implementation of a CI/CD pipeline for a Vue.js technology application using GitHub Actions and Docker. It includes the system architecture, workflow automation, testing process, containerization, and deployment readiness. The document provides detailed technical information about system components, data flow, interfaces, and design considerations.

The scope includes:

- Vue.js application development
- Automated testing using Vitest
- Continuous Integration using GitHub Actions
- Containerization using Docker
- Automated build and validation process

1.2 Intended Audience

This document is intended for:

- Developers working on the project
- DevOps engineers managing CI/CD pipelines
- Project reviewers and evaluators
- System administrators
- Students and technical learners studying CI/CD concepts

It helps readers understand system structure, workflow automation, and deployment mechanisms.

1.3 System Overview

The system is a Vue.js web application integrated with a CI/CD pipeline. When the developer pushes code to GitHub, the pipeline automatically:

- Installs dependencies
- Runs unit tests
- Builds the application
- Prepares the application for containerization

Docker is used to package the application into a container, ensuring consistent deployment across environments.

System Flow:

Developer → GitHub → GitHub Actions → Test → Build → Docker Container

2. System Design

2.1 Application Design

The application follows a modular frontend design using Vue.js.

Main components include:

- App.vue → Main UI component
- main.js → Application entry point
- public folder → Static files
- tests folder → Unit test files

The application runs locally on a browser and renders the user interface.

2.2 Process Flow

The process flow follows CI/CD automation steps:

Step 1: Developer writes code
Step 2: Code is pushed to GitHub
Step 3: GitHub Actions pipeline is triggered
Step 4: Dependencies are installed
Step 5: Unit tests are executed

Step 6: Application build is generated
Step 7: Docker container is prepared

2.3 Information Flow

Information flow moves through different system layers:

Developer → Git Repository → CI Pipeline → Test Module → Build System → Docker Container

Each component processes data and passes it to the next stage.

2.4 Components Design

System components include:

1. Vue.js Application

Responsible for frontend UI rendering.

2. GitHub Repository

Stores source code and version history.

3. GitHub Actions CI/CD Pipeline

Automates testing and build process.

4. Testing Module (Vitest)

Verifies application functionality.

5. Docker Container

Packages application into portable runtime environment.

2.5 Key Design Considerations

Key considerations include:

- Automation of testing and build
- Easy deployment using containers
- Scalability and maintainability
- Reliable and fast build process
- Secure code management
- Platform independence using Docker

2.6 API Catalogue

This project currently does not use external APIs.

However, internally used system tools include:

Tool	**Purpose**
Git	Version control
GitHub Actions	CI/CD automation
Docker	Containerization
Node.js	Runtime environment
Vitest	Testing framework

3. Data Design

3.1 Data Model

The system uses a file-based data model.

Main data includes:

- Source code files
- Test files
- Configuration files
- Dockerfile
- CI/CD workflow files

Example:

```
src/
tests/
package.json
Dockerfile
ci.yml
```

3.2 Data Access Mechanism

Data is accessed using:

- Git commands for version control
- Node.js runtime for application execution
- GitHub Actions for CI/CD automation
- Docker engine for container execution

3.3 Data Retention Policies

Data retention includes:

- Source code stored permanently in GitHub
- Commit history maintained in Git repository
- Docker images stored until manually deleted
- CI logs stored in GitHub Actions

3.4 Data Migration

Data migration occurs through:

- Git push and pull operations
- Docker image transfer between environments
- Repository cloning on different systems

4. Interfaces

System includes following interfaces:

1. User Interface

Browser interface displaying Vue.js application.

Example:

http://localhost:8080

2. GitHub Interface

Web interface for repository and CI/CD monitoring.

3. Docker Interface

Docker Desktop for container management.

4. Command Line Interface

Used for executing Git, Docker, and Node commands

5. State and Session Management

The application currently does not use session management.

State is managed internally by Vue.js framework.

State includes:

- Application UI state
- Component rendering state

Session is temporary and browser-based.

6. Caching

Caching is handled by:

- Browser caching for static files
- Node.js dependency caching
- Docker image layer caching
- GitHub Actions dependency caching

Caching improves performance and reduces build time.

7. Non-Functional Requirements

7.1 Security Aspects

Security measures include:

- Secure GitHub repository access
- Authentication using GitHub credentials
- Controlled access to Docker containers
- Secure code versioning

No sensitive data is stored in the system.

7.2 Performance Aspects

Performance requirements include:

- Fast build execution
- Automated test execution
- Efficient Docker container startup
- Minimal resource consumption

Expected performance:

- Build time: 10-30 seconds
- Test execution: <5 seconds
- Container startup: <10 seconds

8. References

References used:

- **Vue.js Official Documentation**
  https://vuejs.org
- **GitHub Actions Documentation**
  https://docs.github.com/actions
- **Docker Documentation**
  https://docs.docker.com
- **Node.js Documentation**
  https://nodejs.org
- **Vitest Documentation**
  https://vitest.dev