Advanced DevOps Practices for Enterprise-Scale Micro SaaS Development

Course: System Administration & Maintenance | Code: IT31023 | Term: Intake 11 Term 1

Overview

This project-based assessment challenges you to synthesize advanced DevOps practices with real-world SaaS development. You'll build a production-ready Micro SaaS application while implementing enterprise-grade development workflows, security practices, and operational excellence.

Learning Objectives

Upon completion, students will demonstrate mastery in:

1. Architecting scalable, cloud-native Micro SaaS solutions using modern DevOps practices.
2. Implementing enterprise-grade CI/CD pipelines with comprehensive testing and security controls.
3. Applying data-driven product development methodologies aligned with market demands.
4. Managing complex distributed systems using observability and reliability engineering principles.
5. Documenting technical decisions and processes following industry best practices.

Project Phases & Evaluation Criteria

Phase 1: Strategic Planning & Market Research (20%)

Market Analysis & Product Definition

1. Conduct comprehensive competitor analysis using industry tools (G2, Crunchbase, etc.)
2. Develop detailed user personas and journey maps
3. Create a compelling value proposition supported by market research
4. Design pricing strategy with detailed unit economics

Technical Architecture

1. Design system architecture using the C4 model (Context, Containers, Components, Code)
2. Evaluate and justify technology choices considering scalability, maintainability, and cost

3. Create infrastructure-as-code templates for cloud resources
4. Document security and compliance considerations

Phase 2: Development Environment & Workflow (25%)

Git Workflow Implementation

1. Configure branch protection rules and code review requirements
2. Implement Conventional Commits with automated changelog generation
3. Set up automated dependency updates and security patches
4. Create pull request templates and contribution guidelines

Project Management Integration

1. Configure GitHub Projects with automated workflows
2. Integrate issue templates for bugs, features, and technical debt
3. Establish SLA monitoring for issue resolution
4. Document team collaboration protocols

Phase 3: CI/CD & Quality Assurance (30%)

Automated Pipeline Configuration

1. Implement matrix testing across multiple Node.js versions (if applicable)
2. Configure parallel test execution for unit, integration, and E2E tests
3. Set up dependency caching and artifact management
4. Integrate security scanning (SAST, DAST, SCA) with defined quality gates

Monitoring & Observability

1. Configure distributed tracing and logging infrastructure
2. Implement custom metrics and SLO monitoring
3. Create alerting rules with incident response procedures
4. Set up synthetic monitoring for critical user journeys

Phase 4: Production Deployment & Documentation (25%)

Infrastructure & Deployment

1. Configure multi-environment deployment pipeline (dev, staging, prod)
2. Implement a blue-green deployment strategy
3. Set up CDN and edge caching configuration
4. Create disaster recovery and backup procedures

Documentation & Knowledge Base

1. Develop comprehensive technical documentation following the Divio framework
2. Create API documentation using OpenAPI 3.0 specification
3. Document incident response playbooks
4. Prepare system architecture decision records (ADRs)

Deliverables

1. GitHub Repository

1. Complete source code with documented commit history
2. Configured CI/CD pipelines and workflow configurations
3. Comprehensive documentation in the wiki section

2. Technical Implementation

1. Deployed application with a monitoring dashboard
2. Working CI/CD pipeline with quality gates
3. Infrastructure-as-code templates
4. API documentation and test suite

3. Project Documentation

1. Product requirements document (PRD)
2. Technical design document (TDD)
3. System architecture diagrams
4. Operations runbook

4. Final Presentation

1. Live demo of the application
2. Overview of architecture and technical decisions
3. Discussion of challenges and lessons learned
4. Future improvement roadmap

Submission Guidelines

1. All code and documentation must be submitted via GitHub
2. The GitHub link must be included in the report.

Academic Integrity

Students must work individually on this project. While you may discuss concepts and approaches with classmates, all submitted work must be your own. The use of AI coding assistants is permitted but must be documented in your submission.