Product Requirements and Specification Document

Project Name

DevSecOpsLab - Secure CI/CD Pipeline Simulator

---

1. Overview

Description:
DevSecOpsLab is an interactive simulator for secure DevOps practices in business environments. It provides hands-on modules for static code analysis, secrets management, container security, and CI/CD security best practices using Jenkins, Docker, and GitHub Actions. The platform targets education and entrepreneurship, enabling users to learn and apply security measures in automated pipelines.

Target Users:

- DevOps engineers
- Security professionals
- Technical educators
- Entrepreneurs/startups

Difficulty Level:
Hard

---

2. Objectives

- Simulate secure CI/CD pipelines with real-world tools
- Educate users on integrating security into DevOps workflows
- Provide practical modules for vulnerability scanning, secure deployment, and automated security testing

---

3. Core Features

Feature	Description	Technologies
Static Code Analysis	Integrate tools (e.g., Bandit, SonarQube) for code scanning in pipelines	Python, Jenkins
Secrets Management	Detect and manage secrets in code and pipelines	GitHub Actions, Python
Container Security	Scan Docker images for vulnerabilities and misconfigurations	Docker, Trivy
CI/CD Security Best Practices	Enforce secure pipeline configurations and access controls	Jenkins, GitHub Actions

Vulnerability Scanning	Automated scanning of code, dependencies, and containers	Python, Docker
Secure Deployment	Simulate secure deployment workflows with rollback and audit logging	Jenkins, Docker
Automated Security Testing	Integrate security tests into CI/CD (e.g., SAST, DAST)	Jenkins, Python
Reporting & Feedback	Generate actionable security reports and recommendations	Python, Web UI

---

4. Functional Requirements

ID	Requirement	Priority
FR1	Users can configure and run simulated CI/CD pipelines with security modules	High
FR2	System performs static code analysis on user-submitted code	High
FR3	System detects and manages secrets in code and environment variables	High
FR4	Docker images are scanned for vulnerabilities before deployment	High
FR5	Security best practices are enforced in pipeline configurations	High
FR6	Automated security tests are executed as part of the pipeline	High
FR7	Users receive detailed security reports after each pipeline run	Medium
FR8	Users can select between Jenkins and GitHub Actions for pipeline simulation	Medium
FR9	System provides educational content and guidance for each security module	Medium

---

5. Non-Functional Requirements

ID	Requirement	Priority
NFR1	System must be deployable via Docker Compose	High
NFR2	All user data must be isolated and ephemeral	High
NFR3	Response time for pipeline execution < 2 min	Medium
NFR4	Platform must support concurrent users	Medium
NFR5	Documentation must be clear and concise	High

---

6. Technical Specifications

Component	Specification
Frontend	Minimal web UI for pipeline configuration, status, and report viewing
Backend	Python (Flask/FastAPI) for orchestration and API endpoints

CI/CD Engines	Jenkins (Dockerized), GitHub Actions (simulated via local runners)
Security Tools	Bandit, Trivy, SonarQube, custom Python scripts
Containerization	All components run in isolated Docker containers
Secrets Detection	Use open-source tools (e.g., git-secrets, detect-secrets)
Reporting	JSON and HTML reports, downloadable by users
Deployment	Docker Compose for local and cloud deployment

---

7. User Flow

```
graph TD
A[User Login] --> B[Select Pipeline Engine]
B --> C[Configure Pipeline & Security Modules]
C --> D[Submit Code/Repo]
D --> E[Pipeline Execution]
E --> F[Security Scanning & Testing]
F --> G[View Reports & Recommendations]
```

---

8. Success Metrics

Metric	Target Value
Pipeline completion rate	> 95%
Average user session time	> 15 minutes
User-reported learning gain	> 80% positive
Security issue detection rate	> 90% (test cases)

---

9. Risks & Mitigations

Risk	Mitigation
Tool integration complexity	Use Dockerized, pre-configured images
Security tool false positives	Allow user override and feedback
Resource-intensive operations	Limit concurrent jobs, optimize containers
User data leakage	Enforce strict container isolation

---

10. Milestones

Milestone	Target Date

Requirements & Design Complete	Week 2
Core Pipeline Engine (Jenkins)	Week 4
Security Modules Integration	Week 6
GitHub Actions Simulation	Week 7
Reporting & UI	Week 8
Testing & Documentation	Week 9
MVP Release	Week 10

---

11. Appendix

Key Technologies:

- Jenkins, Docker, GitHub Actions, Python, Bandit, Trivy, SonarQube

References:

- OWASP DevSecOps Guidelines
- Jenkins Security Best Practices

End of Document