High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **DevSecOpsLab - Secure CI/CD Pipeline Simulator**. The project aims to provide a hands-on educational platform simulating secure DevOps practices, focusing on static code analysis, secrets management, container security, and CI/CD security best practices using Jenkins, Docker, and GitHub Actions.

---

1. System Architecture Overview

Architecture Description:
The system is composed of modular components simulating a secure CI/CD pipeline. Users interact via a web interface to trigger code commits, which flow through automated security and deployment stages orchestrated by Jenkins and GitHub Actions, leveraging Docker for containerization and various security tools for auditing.

Module/Component	Role/Function
User Interface	Web-based portal for user interaction, scenario selection, and results review
Code Repository (GitHub)	Stores user code, triggers pipeline events
CI/CD Orchestrators	Jenkins and GitHub Actions automate build, test, and deployment workflows
Security Modules	Static code analysis, secrets detection, vulnerability scanning, container security checks
Containerization Layer	Docker-based environment for builds and deployments
Results Dashboard	Aggregates and displays security findings and pipeline status

---

2. Component Interactions

Sequence Step	Interaction Description
1	User submits code or selects scenario via UI
2	Code pushed to GitHub triggers CI/CD pipeline (Jenkins/GitHub Actions)
3	Pipeline executes security modules: static analysis, secrets scan, vulnerability scan, container checks
4	Build and deployment steps run in Docker containers
5	Security and pipeline results sent to Results Dashboard for user review

---

3. Data Flow Overview

Data Flow Source	Destination	Data Type/Content
User Interface	GitHub Repository	Code submissions, scenario selections
GitHub Repository	CI/CD Orchestrators	Codebase, pipeline triggers
CI/CD Orchestrators	Security Modules	Build artifacts, code, container images
Security Modules	Results Dashboard	Security findings, scan reports
CI/CD Orchestrators	Containerization Layer	Build/deployment instructions, images
Results Dashboard	User Interface	Aggregated results, feedback

---

4. Technology Stack

Layer/Function	Technology/Frameworks
UI & Dashboard	Python (Flask/Django), HTML/CSS/JS
Code Repository	GitHub
CI/CD Orchestration	Jenkins, GitHub Actions
Containerization	Docker
Security Tools	Bandit, TruffleHog, Clair, custom scripts
Scripting/Automation	Python, Shell scripts

---

5. Scalability & Reliability

- **Isolation:** Each simulation runs in isolated Docker containers to ensure security and reproducibility.
- **Extensibility:** Modular design allows easy addition of new security tools or pipeline stages.
- **Security:** All secrets and sensitive data are managed using best practices; no persistent sensitive data stored.
- **Reliability:** Automated error handling and logging in all pipeline stages; stateless design for easy scaling.
