DevOps CI/CD Project Proposal

Name:	Jiya Singhal
Roll Number:	10043
Date:	January 2026

1. Project Title

CI/CD Pipeline Implementation using GitHub Actions with Security Integration

2. GitHub Repository URL

https://github.com/jiya-singhal/devops-ci-cd

3. Application Description

I am building a simple Java Spring Boot web application that provides REST API endpoints. The application has three main endpoints: a health check endpoint that returns "OK" to verify the application is running, a hello endpoint that greets users, and a version endpoint that shows the current version.

The application also includes a Calculator service with basic math operations (add, subtract, multiply, divide) which I use to demonstrate unit testing. I chose Spring Boot because it's widely used in the industry and the assignment recommends Java.

The application will be containerized using Docker with a multi-stage build to keep the image small and secure.

4. CI/CD Problem Statement

In traditional software development, many problems occur because of manual processes:

- Developers forget to run tests before pushing code, leading to bugs in production
- Security vulnerabilities in code go unnoticed until it's too late
- Dependencies (external libraries) may have known security issues that nobody checks
- "It works on my machine" problem - code works locally but fails elsewhere
- Manual deployments are slow and error-prone

My project solves these problems by creating an automated pipeline that runs on every code push. The pipeline automatically builds the code, runs tests, checks for security issues, creates a Docker container, and pushes it to Docker Hub - all without any manual intervention.

5. Chosen CI/CD Stages and Justification

Stage	Tool	Why I'm Including This
Checkout	actions/checkout	To get the source code from GitHub
Setup Java	actions/setup-java	To install Java 17 for building the app
Linting	Checkstyle	To enforce coding standards and catch style issues early
Unit Tests	JUnit 5	To verify my code works correctly and catch bugs
Build	Maven	To compile code and create the JAR file
SAST	CodeQL	To scan my code for security vulnerabilities like SQL injection
SCA	OWASP Dependency Check	To check if my libraries have known vulnerabilities
Docker Build	Docker	To package my app in a container for consistent deployment
Image Scan	Trivy	To scan the container for OS-level vulnerabilities
Runtime Test	curl	To verify the container actually starts and works
Push	Docker Hub	To publish the verified image for deployment

I ordered the stages this way because faster checks (like linting) should run first. If linting fails in 5 seconds, there's no point running security scans that take 5 minutes. This is called the "fail-fast" principle.

6. Expected Outcomes

By the end of this project, I expect to achieve:

- A fully automated pipeline that runs on every code push without manual intervention
- Automatic detection of code quality issues through Checkstyle linting
- Security vulnerabilities caught early through SAST (CodeQL) and SCA (OWASP)
- A Docker image that is scanned for vulnerabilities before being published
- Verified container that is tested at runtime before pushing to Docker Hub
- Clear understanding of DevSecOps principles and shift-left security

This project will help me understand how real-world CI/CD pipelines work and why each stage is important for delivering secure, quality software.