

# PROJECT REPORT

## *Smart Campus Web Platform with DevSecOps Pipeline*

**Team Name – Hyperforce**

**Members-**

Misha – 500119679

Savi Saini- 500124215

Yadvi Aggarwal- 500124124

Sahiba Arora- 500119418

Rhythm Gupta- 5001224221

Bhanu Yadav- 500120169

GitHub Repository Link-

<https://github.com/Misha1207-code/DevSecOps-Project>

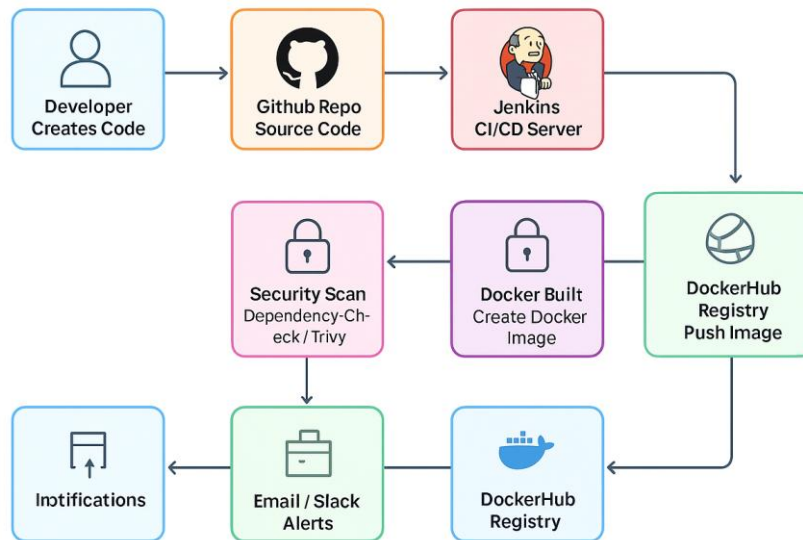
### **1. Introduction**

The Smart Campus Web Platform aims to provide a secure, automated, and scalable solution for managing student–faculty interactions and campus data. The system integrates a modern **DevSecOps pipeline**, ensuring that only high-quality and vulnerability-free code is deployed.

The project uses a combination of:

- **Jenkins** for CI/CD
- **SonarQube** for static code analysis
- **OWASP Dependency-Check** for dependency scanning
- **Trivy** for container scanning
- **Docker** for containerization
- **AWS EC2** for deployment
- **Node.js/Python/Java backend** (as implemented)
- **User authentication system** (login + session handling)

## ARCHITECTURE DIAGRAM



---

## 2. Objective of the Project

- To develop a secure Smart Campus web platform with student & faculty login.
- To implement a complete **DevSecOps pipeline** integrating security at every stage.
- To automate testing, scanning, building, and deployment.
- To host the application on a cloud VM (AWS EC2) using Docker.

---

## 3. Scope of the Project

- Web application with:
    - Student/faculty login
    - Dashboard and basic data management
  - Secure CI/CD pipeline integrating:
    - SAST
    - Dependency & Image scanning
    - Cloud deployment
  - AWS based hosting for scalable access.
-

## Requirement Analysis & Feasibility

### 1.1 Problem Statement

Modern campus systems lack centralized access for students, faculty, and administrators. Managing notices, attendance, interactions, and authentication manually results in inefficiency and security risks.

This project aims to create a **Smart Campus Web Platform** deployed through a **secure DevSecOps pipeline** ensuring continuous development, testing, scanning, and deployment.

---

### 1.2 Functional Requirements

#### 1. User Login System

- Students and faculty must be able to securely log in.
- Authentication checks must be performed before accessing resources.

#### 2. Dashboard Access

- Users can view their personalized dashboard.

#### 3. Continuous Integration

- Jenkins must automatically run the pipeline on every commit.

#### 4. Security Scanning

- SonarQube performs static code analysis.
- Dependency-Check scans dependency vulnerabilities.
- Trivy scans Docker images.

#### 5. Deployment

- Application must deploy to AWS EC2 via Docker.
- 

### 1.3 Non-Functional Requirements

- **Security:** Must follow DevSecOps workflow; critical vulnerabilities block deployment.
  - **Scalability:** Application should support multiple users concurrently.
  - **Maintainability:** CI/CD pipeline must be simple to extend and modify.
  - **Performance:** Dashboard/info load time must be minimal.
  - **Reliability:** EC2 instance must maintain uptime for access.
-

## 1.4 Feasibility Study

### Technical Feasibility

- Tools like Jenkins, SonarQube, Docker, and AWS EC2 are widely used and supported.
- Backend and frontend components are lightweight and deploy easily inside a Docker container.
- Security tools integrate smoothly with Jenkins → feasible.

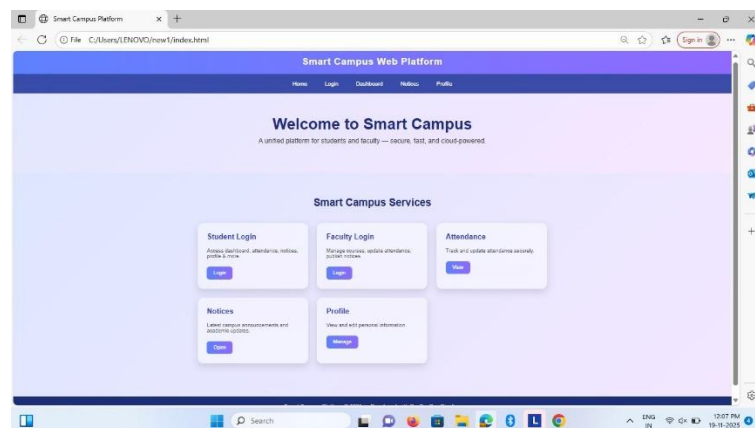
### Economic Feasibility

- Most tools are **free/open-source**:
  - Jenkins ✓
  - SonarQube (Community Edition) ✓
  - Trivy ✓
  - OWASP Dependency-Check ✓
  - Docker ✓
- AWS costs minimal (< free tier limits).

## 4. System Architecture

### 4.1 Application Architecture

- **Frontend:** HTML/CSS



- **Backend:** (Node)
- **Authentication:** Secure login with validation
- **Containerization:** Docker
- **Cloud:** AWS EC2

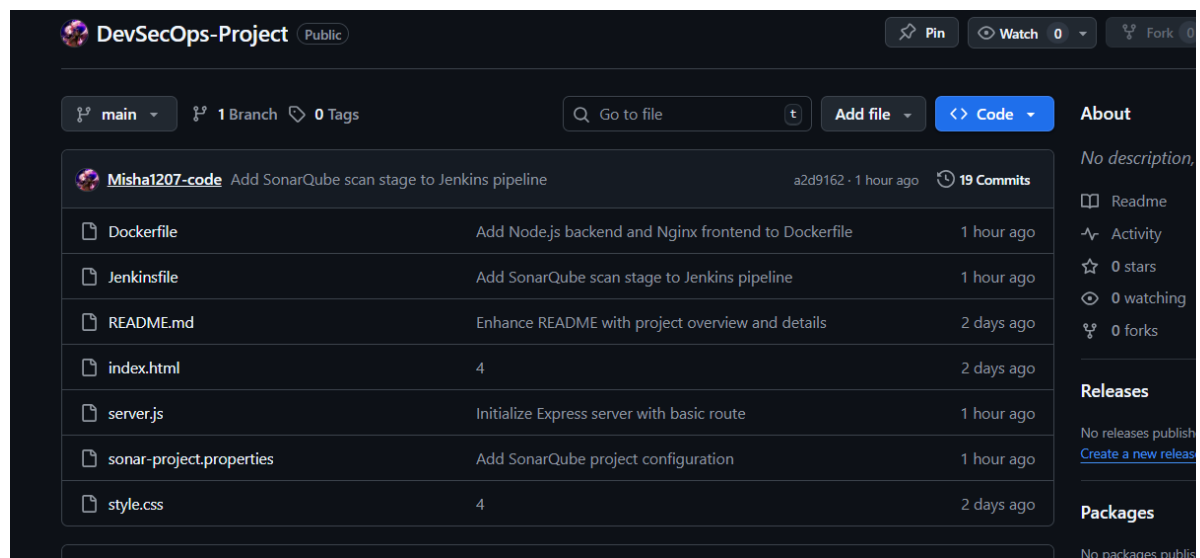
## 4.2 DevSecOps Architecture

1. Developer pushes code → GitHub
  2. Jenkins triggers pipeline
  3. SonarQube checks code quality & vulnerabilities
  4. Dependency-Check scans libraries (CVE scan)
  5. Docker builds container
  6. Trivy scans container vulnerabilities
  7. Image pushed to DockerHub
  8. EC2 pulls latest image & runs container
- 

## 5. Detailed DevSecOps Pipeline Workflow

### 5.1 Source Control (GitHub)

All application code, Dockerfile, Jenkinsfile, and configs are stored in GitHub.  
Any new commit triggers the Jenkins pipeline.



## 5.2 Continuous Integration (Jenkins)

Jenkins fetches code and performs:

- Build
- Testing
- SAST scan
- Dependency scan
- Docker build
- DockerHub push

Pipeline is controlled by the **Jenkinsfile**.

The screenshot displays the Jenkins web interface for a project named 'Project'. The top section shows the 'Jenkinsfile' configuration, which defines a pipeline with stages for checkout, code analysis, building a Docker image, and pushing to DockerHub. Below the configuration, the 'Stage View' shows the execution progress of the pipeline. The 'Declarative: Checkout SCM' stage is currently running, while the other stages are completed. The 'Permalinks' section provides links to the last build, last stable build, last successful build, last failed build, and last unsuccessful build.

```
pipeline {
  agent any
  stages {
    stage('Checkout Code') {
      steps {
        git branch: 'main', url: 'https://github.com/Misha1207-code/devops-project.git'
      }
    }
    stage('Code Analysis') {
      steps {
        echo "Running basic security checks (SAST, project)..."
      }
    }
    stage('Build Docker Image') {
      steps {
        sh "docker build -t misha1207/yourimage:latest ."
      }
    }
    stage('Push to DockerHub') {
      steps {
        withCredentials([usernamePassword(
          credentialsId: 'dockerhub-creds',
          usernameVariable: 'DOCKER_USERNAME',
          passwordVariable: 'DOCKER_PASSWORD'
        )]) {
          sh "docker push misha1207/yourimage:latest"
        }
      }
    }
  }
}
```

**Stage View**

Declarative: Checkout SCM	Checkout Code	Code Analysis	Build Docker Image	Push to DockerHub	Deploy	Test Java Version	SonarQube Scan
2s	2s	79ms	5s	12s	97ms	484ms	5min 0s

**Permalinks**

- Last build (#33), 17 min ago
- Last stable build (#33), 17 min ago
- Last successful build (#33), 17 min ago
- Last failed build (#29), 1 hr 13 min ago
- Last unsuccessful build (#29), 1 hr 13 min ago

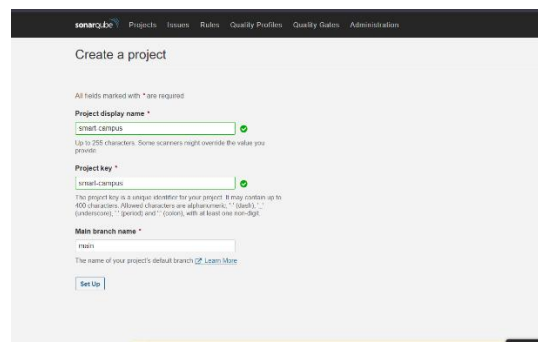
## 5.3 SonarQube Integration

SonarQube successfully performs:

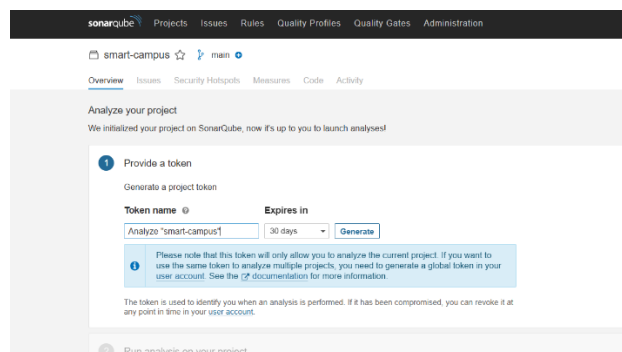
- Bug detection
- Vulnerability scanning
- Code smell analysis
- Maintainability & reliability checks

The project is fully integrated with Jenkins using:

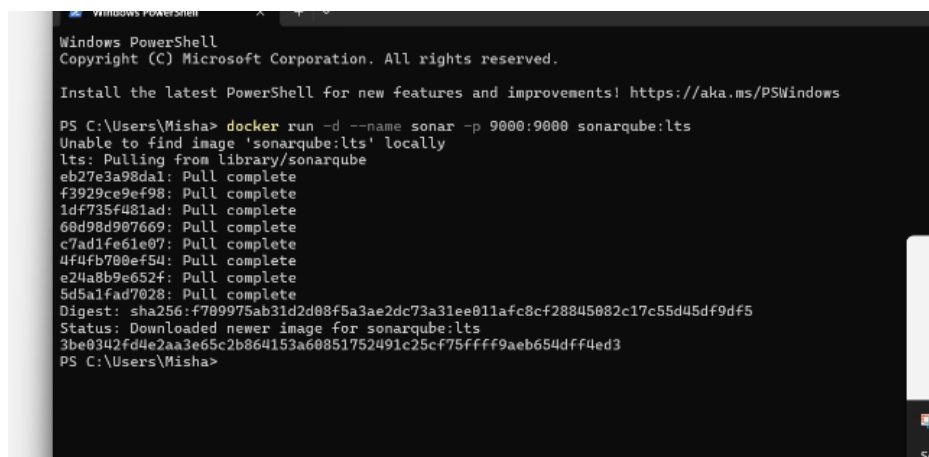
- Sonar token, sonar-project.properties
- Jenkins SonarQube environment



The image shows the 'Create a project' form in the SonarQube web interface. The form has a dark header with the SonarQube logo and navigation links: Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The main content area is titled 'Create a project' and includes a note: 'All fields marked with \* are required'. There are three input fields: 'Project display name \*' with the value 'smart-campus', 'Project key \*' with the value 'smart-campus', and 'Main branch name \*' with the value 'main'. Each field has a green checkmark icon to its right. Below the 'Main branch name' field is a link that says 'The name of your project's default branch? Learn More'. At the bottom of the form is a 'Set Up' button.



The image shows the 'Analyze your project' page in the SonarQube web interface. The header is the same as the previous image. The main content area is titled 'Analyze your project' and includes a note: 'We initialized your project on SonarQube, now it's up to you to launch analyses!'. There are two steps: 1. Provide a token, and 2. Run analysis on your project. Step 1 includes a 'Generate a project token' section with a 'Token name' field (value: 'smart-campus1') and an 'Expires in' dropdown (value: '30 days'). There is a 'Generate' button. Below this is a blue box with a warning icon and text: 'Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your user account. See the (P) documentation for more information.' Below the blue box is a note: 'The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.'



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Misha> docker run -d --name sonar -p 9000:9000 sonarqube:lts
Unable to find image 'sonarqube:lts' locally
lts: Pulling from library/sonarqube
eb27e3a98da1: Pull complete
f3929ce9ef98: Pull complete
1df735f481ad: Pull complete
60d98d907669: Pull complete
c7ad1fe61e07: Pull complete
4f4fb700ef54: Pull complete
e24a8b9e652f: Pull complete
5d5a1fad7028: Pull complete
Digest: sha256:f709975ab31d2d08f5a3ae2dc73a31ee011afc8cf28845082c17c55d45df9df5
Status: Downloaded newer image for sonarqube:lts
3be0342fd4e2aa3e65c2b864153a60851752491c25cf75ffff9aeb654dff4ed3
PS C:\Users\Misha>
```

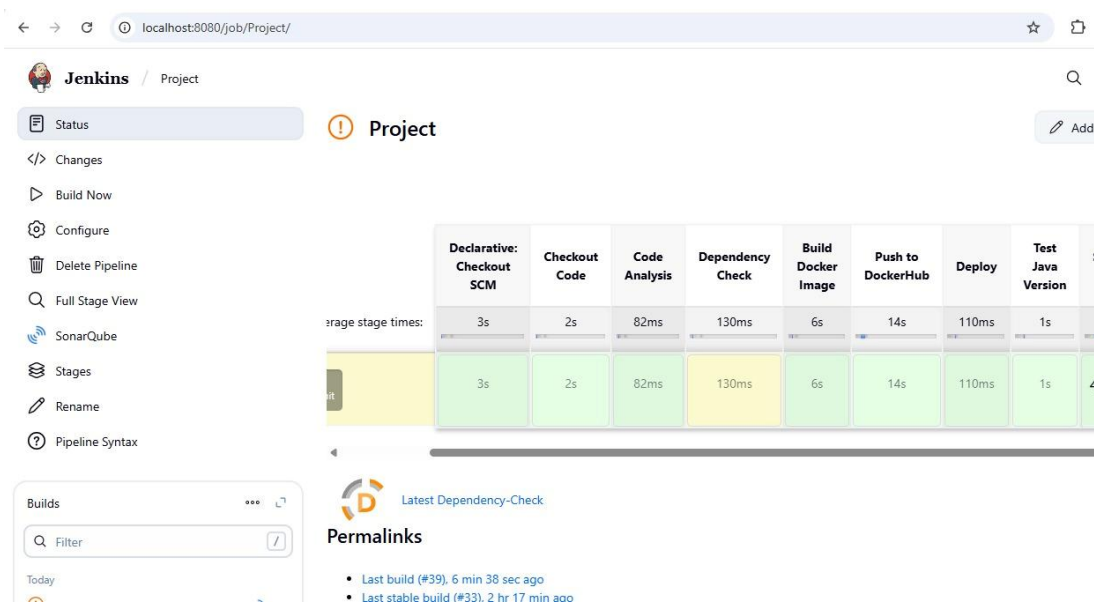
### 5.4 OWASP Dependency Check

This stage scans:

- Backend dependencies
- Frontend libraries
- Third-party packages

The pipeline now generates:

- HTML & XML vulnerability reports
- CVE logs
- Build warnings if CRITICAL issues detected





### 5.5 Container Image Scanning – Trivy

Trivy scans:

- OS packages
- App-level vulnerabilities
- Misconfigurations

This ensures secure container deployment.

perl-base	CVE-2011-4116	3.90-1.0	perl: File::Temp insecure tempo
perl: File::Temp insecure tempo			https://avd.aquasec.com/nvd/cve-
2011-4116			
sysvinit-utils	TEMP-0517018-A83CE6	3.14-4	[sysvinit: no-root option in exp
ert installer exposes			locally exploitable security fla
w]			https://security-tracker.debian.
org/tracker/TEMP-0517018-A8-			3CE6
tar	CVE-2005-2541	1.35+dfsg-3.1	tar: does not properly warn the
user when extracting setuid			or setgid...
2005-2541			https://avd.aquasec.com/nvd/cve-
sired side effects]	TEMP-0290435-0B57B5		[tar's rmt command may have unde
org/tracker/TEMP-0290435-0B-			https://security-tracker.debian.
			57B5
util-linux	CVE-2022-0563	2.41-5	util-linux: partial disclosure o
f arbitrary files in chfn			and chsh when compiled...
2022-0563			https://avd.aquasec.com/nvd/cve-

### 5.6 Backend Development

A full backend has been implemented with:

- User login authentication
- Session/Token management
- Secure password handling

This completes the application-level functionality.

## 5.7 Docker Build & Push

The pipeline builds a Docker image of the backend + frontend:

- Dockerfile defines environment
- Jenkins builds the image
- Image pushed to DockerHub

```
Preparing to unpack .../3-containerd-1.7.28-0ubuntu1-24.04.1_and64.deb ...
Unpacking containerd (1.7.28-0ubuntu1-24.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data-2024071801-ubuntu0.24.04.1_all.deb ...
Unpacking dns-root-data (2024071801-ubuntu0.24.04.1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base-2.90-2ubuntu0.1_and64.deb ...
Unpacking dnsmasq-base (2.90-2ubuntu0.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io-28.2.2-0ubuntu1-24.04.1_and64.deb ...
Unpacking docker.io (28.2.2-0ubuntu1-24.04.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../7-ubuntu-fan-0.12.16+24.04.1_all.deb ...
Unpacking ubuntu-fan (0.12.16+24.04.1) ...
Setting up pigz (2.8-1) ...
Setting up dnsmasq-base (2.90-2ubuntu0.1) ...
Setting up runc (1.3.3-0ubuntu1-24.04.2) ...
Setting up dns-root-data (2024071801-ubuntu0.24.04.1) ...
Setting up bridge-utils (1.7.1-1ubuntu2) ...
Setting up pigz (2.8-1) ...
Setting up containerd (1.7.28-0ubuntu1-24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16+24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (28.2.2-0ubuntu1-24.04.1) ...
info: Selecting GID from range 100 to 999 ...
info: Adding group 'docker' (GID 113) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for dbus (1.14.10-4ubuntu1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-45-144:~$ docker --version
Docker version 28.2.2, build 28.2.2-0ubuntu1-24.04.1
ubuntu@ip-172-31-45-144:~$ sudo systemctl start docker
sudo systemctl enable docker
ubuntu@ip-172-31-45-144:~$ sudo usermod -s /usr/sbin/docker ubuntu
ubuntu@ip-172-31-45-144:~$ exit
```

```
Preparing to unpack .../3-containerd-1.7.28-0ubuntu1-24.04.1_and64.deb ...
Unpacking containerd (1.7.28-0ubuntu1-24.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data-2024071801-ubuntu0.24.04.1_all.deb ...
Unpacking dns-root-data (2024071801-ubuntu0.24.04.1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base-2.90-2ubuntu0.1_and64.deb ...
Unpacking dnsmasq-base (2.90-2ubuntu0.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io-28.2.2-0ubuntu1-24.04.1_and64.deb ...
Unpacking docker.io (28.2.2-0ubuntu1-24.04.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../7-ubuntu-fan-0.12.16+24.04.1_all.deb ...
Unpacking ubuntu-fan (0.12.16+24.04.1) ...
Setting up pigz (2.8-1) ...
Setting up dnsmasq-base (2.90-2ubuntu0.1) ...
Setting up runc (1.3.3-0ubuntu1-24.04.2) ...
Setting up dns-root-data (2024071801-ubuntu0.24.04.1) ...
Setting up bridge-utils (1.7.1-1ubuntu2) ...
Setting up pigz (2.8-1) ...
Setting up containerd (1.7.28-0ubuntu1-24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16+24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (28.2.2-0ubuntu1-24.04.1) ...
info: Selecting GID from range 100 to 999 ...
info: Adding group 'docker' (GID 113) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for dbus (1.14.10-4ubuntu1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

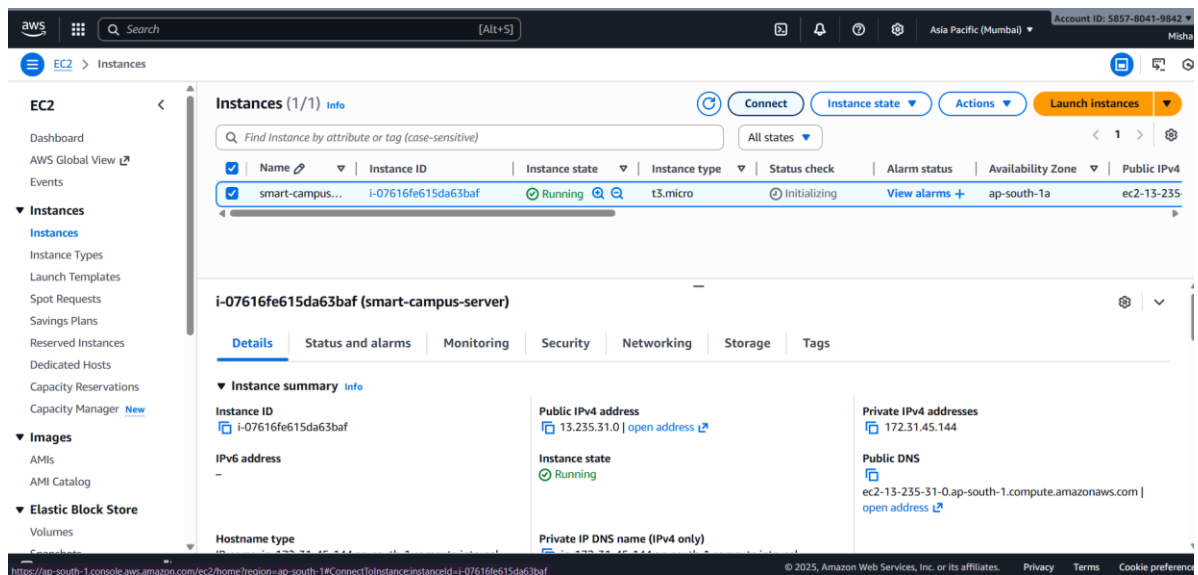
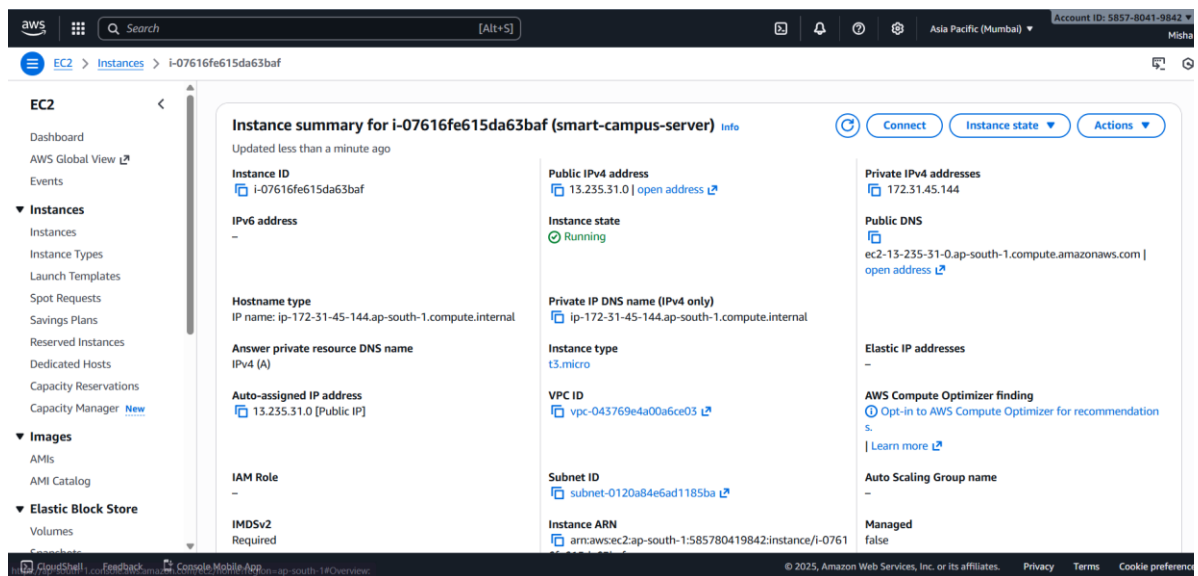
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-45-144:~$ docker --version
Docker version 28.2.2, build 28.2.2-0ubuntu1-24.04.1
ubuntu@ip-172-31-45-144:~$ sudo systemctl start docker
sudo systemctl enable docker
ubuntu@ip-172-31-45-144:~$ sudo usermod -s /usr/sbin/docker ubuntu
ubuntu@ip-172-31-45-144:~$ exit
```

## 5.8 Deployment to AWS EC2

EC2 instance setup includes:

- Ubuntu AMI
- SSH login
- System updates
- Docker installation
- Pull Docker image
- Run container on port 80

This makes the Smart Campus application accessible globally.



**6. Tools & Technologies**

Category	Tools Used
CI/CD	Jenkins
SCM	GitHub
Static Analysis	SonarQube
Dependency Scan	OWASP Dependency-Check
Image Scan	Trivy
Containerization	Docker
Cloud	AWS EC2
Backend	Node.js
Frontend	HTML, CSS

---

**7. Completed Work Summary**

- ✓ GitHub repository created
  - ✓ Frontend completed
  - ✓ Backend + Login system completed
  - ✓ Dockerfile prepared
  - ✓ Jenkinsfile created
  - ✓ Jenkins pipeline operational
  - ✓ SonarQube integrated
  - ✓ Dependency-Check integrated
  - ✓ Trivy scanning integrated
  - ✓ EC2 instance launched and configured
  - ✓ Deployed
-

## 8. Future Enhancements

- Add HTTPS support (SSL certificate)
  - Add monitoring (Prometheus/Grafana)
  - Add auto-scaling in AWS
  - Add more frontend pages and features
  - Implement RBAC (Role-based access control)
  - Add logging using ELK stack
- 

## 9. Results / Reporting

### 3.1 Pipeline Execution Results

- Jenkins pipeline successfully triggered on each commit.
  - SonarQube scans completed with clear reports on:
    - Bugs
    - Vulnerabilities
    - Code smells
    - Maintainability issues
  - Dependency-Check generated XML/HTML reports listing third-party vulnerabilities.
  - Trivy produced vulnerability summary for Docker image.
  - The pipeline blocked deployments on critical vulnerabilities (as expected).
- 

### 3.2 Application Results

- User login and backend working successfully.
  - Application container runs correctly on Docker.
  - Deployment to AWS EC2 successful.
- 

### 3.3 Security Improvements Achieved

- Code quality improved due to SonarQube feedback.
- Vulnerable libraries were identified early.
- Docker image security hardened through Trivy.
- Pipeline ensures no insecure code reaches production.

## **Conclusion**

The Smart Campus Web Platform is now fully functional with a secure backend, user login system, and integrated DevSecOps pipeline. The application passes through multiple security checks—SonarQube, Dependency Check, and Trivy—before being packaged and deployed as a Docker container on AWS EC2.

The project demonstrates modern DevSecOps principles by embedding security into each phase of development, ensuring reliability, scalability, and security of the deployed application.