Product Requirements

D	roh	lom	Statement:
\boldsymbol{r}	1 ()()	16111	Sialemeni

Modern containerized environments use thousands of container images that bundle applications with dependencies. These images may carry known vulnerabilities that pose critical security risks.

Security and DevOps teams need a simple yet powerful interface to:

- Detect vulnerabilities in container images
- Prioritize fixes based on severity
- Take actionable steps to remediate issues

Goals:

- 1. Identify all container images and list their vulnerability status.
- 2. Enable sorting/filtering by vulnerability severity (Critical/High/Medium/Low).
- 3. Highlight images requiring immediate remediation.
- 4. Support vulnerability scan history and trends.
- 5. Provide integration options with CI/CD or ticketing systems (bonus).

Target Users:

- Security Engineers
- DevOps Teams
- Compliance Auditors

Features:

1. Dashboard Overview

- Total images scanned
- Number of images with vulnerabilities
- Breakdown by severity
- Scan history graph

2. Image Inventory View

- Table with image name, last scanned time, total vulnerabilities, critical/high counts
- Sortable by severity, name, or scan date
- Search functionality

3. Image Detail Page

- Vulnerability summary for a specific image
- List of CVEs (ID, severity, fix version, description)
- Option to export or generate remediation ticket

4. Scan Management

- Manual or scheduled scans
- Integration with GitHub/DockerHub
- Status of last scan (success/failure)

5. Notifications (Optional)

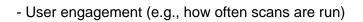
- Alert system for newly discovered critical vulnerabilities

Developer Action Items

Developer Action Items:

- Average time to detect and report vulnerabilities

Back-End:			
- Implement container scanning using open tools (e.g., Trivy, Clair)			
- Create APIs to fetch image/vulnerability data			
- Schedule and run scans in background jobs			
Front-End:			
- Build dashboard, table views, and detail pages			
- Implement filtering/sorting/searching logic			
DevOps:			
- Automate scan triggers on image push			
- Store scan history (e.g., in PostgreSQL)			
Integrations (Optional but Strong Plus):			
- GitHub Actions, Jenkins plugins			
- Jira or Slack alerts for new critical vulnerabilities			
Success Metrics:			
- % of images scanned successfully			



- Reduction in number of unresolved critical vulns over time