Product Overview

Name: Container Image Vulnerability Dashboard
Purpose: To help users identify, assess, and prioritize vulnerabilities in container images stored in their repositories, especially those with critical/high severity.

2. Goals & Objectives

- Provide a centralized view of scanned container images and associated vulnerabilities.
- Help users quickly identify and prioritize remediation for critical/high vulnerabilities.
- Enable filtering, sorting, and searching across thousands of images.
- Provide actionable insights like links to remediation steps or upgrade paths.

3. User Stories

◇ Primary User: DevSecOps Engineer

1. **As a user**, I want to see a list of container images and their vulnerability status so that I can focus on the most affected ones.
2. **As a user**, I want to filter images by severity (critical/high/medium/low/none) so that I can prioritize remediation efforts.
3. **As a user**, I want to drill down into a specific image to see which vulnerabilities are present, their severity, and the affected packages.
4. **As a user**, I want to search for a specific image by name or tag.
5. **As a user**, I want to export a report or share findings with other teams.

4. Key Features & Requirements

4.1. Image Overview Page

Feature	Description
Table of container images	Displays image name, tag, last scanned date, total vulnerabilities, severity breakdown (Critical, High, etc.)

Filtering	By severity, image name, last scanned date
Sorting	By number of vulnerabilities, severity, scan date
Search	Free-text search by image name or tag

4.2. Image Details Page

Feature	Description
Vulnerability list	Shows all vulnerabilities with severity, package, version, and fix availability
Group by severity	Auto-group vulnerabilities
Suggested fixes	Recommended package upgrade or workaround
Metadata	Image size, base OS, scan timestamp

4.3. Dashboard Summary (optional MVP+)

Metric	Example
Total images scanned	10,000+
Images with critical vulns	135
Most vulnerable image	`backend-app:v1.4`
Severity distribution	Pie chart / bar graph

5. User Flow

```
Dashboard → Image List → Image Details → Suggested Fix
```

6. Non-Functional Requirements

- Handle high-scale datasets (10k+ images)
- Secure API integration with container registry and scanner
- Responsive UI

- Scans triggered daily or on-demand

7. Out of Scope (for MVP)

- Automatic patching
- Deep integration with CI/CD tools
- Real-time alerts/notifications

Low-Fidelity Wireframes

1. Image Overview Page

```
+------------------------------------------------------------+
| Search: [backend-app]        [Filter: Severity ▼]         |
+------------------------------------------------------------+
| Image Name       | Tag    | Last Scanned | Critical | High | Total |
|-----------------|-------|--------------|----------|------|-------|
| backend-app      | v1.4  | 2025-04-10   |   5      | 12   | 25    |
| redis-container  | v2.0  | 2025-04-11   |   0      | 2    | 2     |
| nginx            | latest| 2025-04-09   |   3      | 7    | 12    |
+------------------------------------------------------------+
```

Clicking a row leads to → Image Details Page

2. Image Details Page

```
Image: backend-app:v1.4
Last Scanned: 2025-04-10
Base OS: Debian

+-----------------------------------------------------------+
| Severity: [All ▼]   Group by: [Severity ▼]               |
+----------------------------------------------------------+
| CVE ID      | Severity | Package   | Version | Fix         |
|------------ |----------|----------|---------|------------|
| CVE-1234-567| Critical | openssl   | 1.1.1k  | 1.1.1u      |
| CVE-9999-888| High     | libcurl   | 7.70.0  | 7.78.0      |
| CVE-2222-333| Medium   | bash      | 5.0     | Not Available |
+----------------------------------------------------------+
```

3. Dashboard Summary (Optional)

```
+---------------------------------------------+
| Total Images: 10,124   | Scanned Today: 512 |
| Images w/ Critical Vulns: 135               |
| Most Vulnerable Image: backend-app:v1.4     |
| Severity Distribution: [Bar Graph]          |
+---------------------------------------------+
```

Development Action Items

1. **API Integrations**
    a. Connect to container registry (e.g., Docker Hub, ECR)
    b. Connect to scanning engine (e.g., Trivy, Clair, Snyk)
2. **Backend Services**
    a. Database schema for storing image metadata + vulnerabilities
    b. Scheduled scan jobs / webhook support
    c. API endpoints for:
        i. Get all images with summaries
        ii. Get image details (vuln list)
        iii. Filtering/sorting support
3. **Frontend**
    a. Build dashboard UI (React or similar)
    b. Table components for overview + detail views
    c. Charts (Optional) using Chart.js or D3.js

4. **Security**
    a. Auth (OAuth/JWT)
    b. RBAC (Role-based access for users)
5. **Scalability**
    a. Pagination for large datasets
    b. Lazy loading of data
6. **Testing**
    a. Unit tests for backend services
    b. UI/UX usability tests