Product Requirements Document

1. Overview

- **Product Name:** Container Image Vulnerability Scanner
- **Objective:** A security product that scans container images, detects vulnerabilities, categorizes their severity, and provides actionable insights for remediation.
- **Target Users:** DevOps engineers, security teams, and developers managing containerized applications.

2. Problem Statement

1. **Visibility of vulnerabilities** – Users need a **clear, centralized dashboard** to view vulnerabilities in container images.
2. **Severity Classification** – Users need a way to prioritize vulnerabilities based on severity (**Critical, High, Medium, Low**).
3. **Scalability** – Users manage thousands of images, so filtering and sorting are essential.
4. **Remediation Actions** – Users need **guidance** on how to fix vulnerabilities (patching, upgrading, or removing affected dependencies).

3. User Stories & Functional Requirements

User Story	**Priority**	**Feature**
As a user, I want to see a list of scanned images with their vulnerability status.	High	Dashboard with list of container images, showing number of vulnerabilities per image.
As a user, I want to filter images by severity (Critical, High, Medium, Low).	High	Filtering option in UI with severity levels.

As a user, I want to search for a specific image by name.	Medium	Search bar in UI.
As a user, I want to drill down into a specific image and see details of vulnerabilities.	High	Clicking an image opens a details page with vulnerability information.
As a user, I want to receive recommendations on how to fix vulnerabilities.	High	Show upgrade/patch suggestions for affected packages.
As a user, I want an automated report of vulnerabilities.	Medium	Export scan results in CSV/PDF.
As a security lead, I want to get a weekly summary of vulnerabilities.	Low	Email notification system for summary reports.

4. Feature Scope

MVP (Minimum Viable Product) Features

- **Container Image Scanning** – Pull image metadata and scan for vulnerabilities.
- **Severity-Based Sorting & Filtering** – Allow users to focus on critical vulnerabilities.
- **Detailed Vulnerability Insights** – Show CVE (Common Vulnerabilities and Exposures) details.
- **Fix Recommendations** – Suggest patching or upgrading affected dependencies.
- **Basic Reporting** – Export scan results.

Future Enhancements

- **Automated Remediation** – Directly apply patches via integration with CI/CD pipelines.
- **Role-Based Access Control (RBAC)** – Restrict access based on user roles.

- **Integration with DevOps Tools** – Compatibility with Jenkins, GitHub Actions, etc.

5. UX Flow (How Users Will Interact with the Product)

- **User logs in and lands on the Dashboard** – Sees a list of scanned container images.
- **User selects an image** – Opens the detailed vulnerability view.
- **User filters vulnerabilities** – Focuses on Critical & High vulnerabilities.
- **User takes action** – Fixes the issue by following the recommendations.
- **User exports report** – Downloads a CSV or PDF for auditing.

6. Success Metrics

The success of the product will be measured by:

1. **Reduction in Critical Vulnerabilities**: Percentage decrease in critical/high vulnerabilities over time.
2. **User Satisfaction**: Feedback from users on the product's usability and effectiveness.
3. **Time Saved**: Reduction in time spent manually identifying and fixing vulnerabilities.
4. **Adoption Rate**: Number of users and repositories integrated with the product.

Low-Fidelity Wireframes

The wireframes for the **Container Image Vulnerability Scanner** can be accessed on Figma.
Click the link below to view the wireframe:
https://www.figma.com/design/ewvvh9MKpfwrufxUFMbcTP/Accuknox-problem-statement-1?node-id=0-1&t=CaSBlgPlKDe5n13s-1

Development Action Items

1. Backend Development

- Use **Trivy** or **Clair** for vulnerability scanning.
- Store scan results in a **PostgreSQL/Elasticsearch database**.
- Expose **REST APIs** for fetching vulnerabilities.

2. Frontend Development

- Build UI using **React/Next.js**.
- Use **Chart.js** for visualizing vulnerability trends.
- Implement **filtering, sorting, and bulk action functionalities**.

3. CI/CD & Automation

- Automate scanning in **GitHub Actions/Jenkins**.
- Enable **weekly email reports** via **AWS SES** or **SendGrid**.

4. Database Design

- Design a database to store **container image metadata** and **vulnerability data**.

5. Testing

- Perform **unit testing, integration testing, and user acceptance testing (UAT)**.