**Problem Statement 1**

**Title: Vulnerability Management for Container Images**

**1. Background**

Modern applications are increasingly containerized. However, container images often bundle multiple dependencies, many of which can have known vulnerabilities (CVEs).  
Users (security engineers, developers, DevOps) need an efficient way to:

* Identify vulnerable container images.
* Prioritize fixes based on severity.
* Manage thousands of images efficiently.

**2. Goals**

* List all container images and show their vulnerability status.
* Allow users to filter by severity (Critical/High/Medium/Low).
* Highlight images that require immediate action.
* Offer an easy way to search, sort, and prioritize vulnerabilities.
* Optionally, link to remediation guidance.

**3. User Stories**

| **As a...** | **I want to...** | **So that...** |
| --- | --- | --- |
| Security Engineer | View a list of images and their vulnerabilities | I can identify which images need fixing |
| DevOps Engineer | Filter images by severity | I can prioritize critical vulnerabilities |
| Developer | See vulnerability details for a particular image | I can fix them faster |
| Manager | See a high-level overview (dashboard) | I can report risk to leadership |

**4. Functional Requirements**

* **Image List View**:
  + Image Name
  + Vulnerability Count
  + Highest Severity Present
  + Last Scan Timestamp
* **Filters**:
  + Severity (Critical/High/Medium/Low)
  + Image Name (Search)
  + Date Range (e.g., scanned within the last week)
* **Vulnerability Detail View**:
  + CVE ID
  + Severity
  + Component (library, package)
  + Version Affected
  + Fixed Version (if available)
  + Links to advisories
* **Dashboards/Reports**:
  + % of images with Critical vulnerabilities
  + Trends over time
  + Exportable reports (CSV, PDF)

**5. Non-Functional Requirements**

* **Scalable** to support 10,000+ container images.
* **Responsive UI** (Desktop and Tablet friendly).
* **Secure** access (authentication/authorization).
* **Real-time updates** or scheduled scans.

**6. Wireframe (Low-Fidelity Sketch)**

1. **Homepage/Dashboard**
   * Stats at the top (Total Images | Critical | High | Medium | Low).
   * Graph: Vulnerability trend over time.
   * Button: "View Images"
2. **Image List Page**
   * Table with columns:
     + **[Checkbox] | Image Name | Vulnerability Count | Highest Severity | Last Scan | Actions (View Details)**
   * Filters at the top (Severity dropdown, Search bar)
3. **Image Detail Page**
   * Breadcrumb: Home > Image List > Image Details
   * Image Summary (Name, Last Scanned, Total Vulnerabilities)
   * Vulnerability Table:
     + **CVE ID | Severity | Component | Version Affected | Fixed Version | Links**

**Bonus: Development Action Items**

1. **Backend API to fetch:**
   * List of images with vulnerability metadata
   * Detailed vulnerabilities per image
2. **Frontend:**
   * List page with pagination and filters
   * Detail page with vulnerability breakdown
   * Dashboard with charts (criticality trends, vulnerability counts)
3. **Security:**
   * Authentication/Authorization (role-based access)
4. **Scanning Infrastructure:**
   * Integrate with vulnerability scanners (e.g., Trivy, Clair, Grype)
   * Schedule periodic scans
   * Store scan results
5. **DevOps:**
   * Setup CI/CD for backend and frontend deployments
   * Cloud storage (e.g., AWS S3, GCP buckets) for scan reports