Solution Approach

Problem Statement 1: Product Requirement and Low-Fidelity Wireframes

1. **Product Requirements Document:**
    - **Introduction:**
        - Purpose of the document
        - Scope of the product
    - **User Stories:**
        - As a user, I need to see a list of container images with vulnerabilities.
        - As a user, I need to filter images based on vulnerability severity.
        - As a user, I need to get detailed information about each vulnerability.
        - As a user, I need to prioritize fixing critical and high vulnerabilities.
    - **Functional Requirements:**
        - Scanning container images for vulnerabilities.
        - Displaying a list of images with vulnerabilities.
        - Filtering and sorting images based on severity.
        - Detailed view of each vulnerability.
    - **Non-Functional Requirements:**
        - Performance requirements
        - Security requirements
        - Usability requirements
    - **Assumptions and Constraints:**
        - Assumptions about the user environment
        - Constraints related to technology and resources
2. **Low-Fidelity Wireframes:**
    - **Dashboard:**

        - Overview of total images scanned, number of images with vulnerabilities, and severity distribution.
    - **Image List View:**
        - List of container images with columns for image name, number of vulnerabilities, and severity.
    - **Image Detail View:**
        - Detailed information about vulnerabilities in a selected image, including severity, description, and remediation steps.
3. **Development Action Items:**
    - Define the API endpoints for scanning and retrieving vulnerability data.
    - Implement the backend logic for scanning container images.
    - Develop the frontend components for displaying the dashboard, image list, and detail views.
    - Integrate the frontend with the backend API.

Problem Statement 2: Kubernetes Security Scan

1. **Install Local K8s Cluster:**
    - Choose a tool (Minikube, K3s, Kind, etc.) and set up the local cluster.
2. **Scan for Findings:**
    - Use a tool like Kubescape to scan the cluster for security findings.
3. **Generate JSON File:**
    - Export the findings to a JSON file.

Problem Statement 3: GoLang Program and Kubernetes Deployment

1. **Create GoLang Program:**
    - Develop a simple GoLang web application that displays the current date and time.
    - Host the code on GitHub.
    - Create a Dockerfile and push the image to DockerHub.
2. **Deploy to Kubernetes:**

- Write a Kubernetes deployment manifest to deploy the container with 2 replicas.
- Apply the manifest to the Kubernetes cluster.

3. **Expose the App:**
    - Create a Kubernetes service to expose the application to the Internet.