SecOps® Solution

SecOps Solution
Frontend Engineering Internship Test

Problem Statement:

Your task is to develop a responsive web-based CVE database application that allows users to view and manage CVE records. The application should provide a user-friendly interface for viewing detailed information about CVE entries. Additionally, users should be able to add, edit, and delete CVE records as needed. **You can refer to the attached CVE record's CSV file for demo entries.**

Stepwise Requirements:

Implement a responsive design to ensure compatibility across various devices and screen sizes. Use anything from Angular 2+, ReactJS, VueJS, HTML, CSS, Bootstrap, Tailwind, etc to build interactive components and enhance user experience.

The application should include the following features:

1. **View CVE Records:**
   a. Display a table layout showing CVE records.
   b. Include columns for CVE-ID, Severity, CVSS, Affected Packages and CWE-ID.
   c. Each row should represent a single CVE record with its corresponding details displayed in the respective columns.
   d. Add "Edit" and "Delete" options in each row to allow users to edit or delete the corresponding CVE record.

2. **Add CVE Record:**
   a. Provide an "Add CVE" button.
   b. When the "Add CVE" button is clicked, open a modal dialog box.
   c. Inside the modal, include input fields for CVE-ID, Severity, CVSS, Affected Packages, and CWE-ID.
   d. Add "Save" and "Cancel" buttons to confirm or cancel the addition of the new CVE record.
   e. Validate input data to ensure all required fields are filled before saving the new CVE record.

3. **Edit CVE Record:**
    a. Include an "Edit" option in each row of the CVE table.
    b. When the "Edit" option is clicked, open a modal dialog pre-filled with the existing CVE details for editing.
    c. Allow users to modify the details of the CVE record within the modal dialog.
    d. Add "Save" and "Cancel" buttons to confirm or cancel the changes made to the CVE record.
    e. Validate input data to ensure all required fields are filled before saving the new CVE record.

4. **Delete CVE Record:**
    a. Include a "Delete" option in each row of the CVE table.
    b. When the "Delete" option is clicked, prompt the user with a confirmation message asking if they're sure they want to delete the CVE record.
    c. Provide options to confirm or cancel the deletion of the CVE record.
    d. If confirmed, permanently remove the CVE record from the data entries.

5. **Bonus Enhancements (Optional):**
    a. Include client-side data sorting and filtering for the CVE table:
        i. Enable users to sort CVE records based on different criteria such as CVE-ID, Severity, CVSS, Affected Packages, and CWE-ID.
        ii. Provide filtering options to allow users to view CVE records based on specific criteria (e.g., CVE-ID, Severity, CVSS, Affected Packages, and CWE-ID).

Ensure HTML, CSS, and JavaScript are utilized to:

1. Create an HTML structure for the CVE database application, including tables for displaying CVE records and modal dialogs for adding/editing CVE records.
2. Style the tables, modals, and buttons to make the application visually appealing and user-friendly.
3. Implement functionality to open and close the modal dialogs.
4. Create proper functions to handle adding, editing, and deleting CVE records.
5. Ensure data validation in the modals (e.g., all the required fields are filled.)

****Note: The assignment is only front-end focused so NO BACKEND development is needed. You can maintain demo records on the client side only.**

Evaluation Criteria:

1. **Functionality Assessment:** Evaluate the correctness and functionality of frontend features such as viewing, adding, editing, and deleting CVE records.
2. **User Interface Design:** Assess the design, layout, and user interface for visual appeal, consistency, intuitiveness, and ease of navigation.
3. **Responsiveness and Compatibility:** Test the responsiveness across various devices and browsers to ensure a consistent user experience.
4. **Data Handling and Validation:** Review data input and validation, including proper handling of user inputs and display of error messages.
5. **Code Quality and Structure:** Evaluate code clarity, organization, adherence to best practices, readability, and maintainability.

Submission Instructions:

1. Please submit your code along with a README.md file explaining the steps necessary to run the web app, including its dependencies.
2. Ensure that the README.md file provides clear and concise instructions for setting up and running the application locally.
3. Once you have prepared the application and README file, zip the entire codebase and send it as an attachment to “recruitment@secopsolution.com”.
4. We will review the provided instructions and run the application accordingly for evaluation purposes.

Feel free to use any available resources, such as online tutorials, documentation, forums, or even AI tools like ChatGPT, to help solve the problem effectively. However, simply copying and pasting code without understanding it, or submitting projects directly generated by AI tools or copied from repositories like GitHub, will result in automatic disqualification.