API Document

1.API url :

const apiUrl = 'https://services.nvd.nist.gov/rest/json/cves/2.0';

2.DOM elements used:

The application uses several DOM elements for user interaction and data display. These are selected and stored in constants at the beginning of the script.

```
const tableBody = document.getElementById('tableRows');
const pageSizeSelect = document.getElementById('pageSize');
const totalRecordsDisplay = document.getElementById('totalRecords');
const cveDetailsDiv = document.getElementById('cveDetails');
const mainContentDiv = document.getElementById('mainContent');
const prevPageBtn = document.getElementById('prevPage');
const nextPageBtn = document.getElementById('nextPage');
const currentPageSpan = document.getElementById('currentPage');
const backToListBtn = document.getElementById('backToList');
const clearFiltersBtn = document.getElementById('clearFilters');
const cveIdFilterInput = document.getElementById('cveIdFilter');
const yearFilterInput = document.getElementById('yearFilter');
const cvssMinScoreInput = document.getElementById('cvssMinScore');
const cvssMaxScoreInput = document.getElementById('cvssMaxScore');
const loadingDiv = document.getElementById('loading');
```

3. Pagination State

```
let currentPage = 1;
let pageSize = 10;
let totalRecords = 0;
```

- currentPage: Tracks the current page of results.
- pageSize: Number of results to display per page.
- totalRecords: Total number of records returned by the API.
- 'cveIdFilter`, `yearFilter`, `cvssMinScore`, `cvssMaxScore`: Store the current filter values.

```
let cveIdFilter = '';
let yearFilter = '';
let cvssMinScore = '';
let cvssMaxScore = '';
```

cveIdFilter: A string to filter CVEs by their ID.
yearFilter: A string to filter CVEs by the year of publication.

cvssMinScore: A string or number to represent the minimum CVSS score for filtering.
cvssMaxScore: A string or number for the maximum CVSS score for filtering

-Debounce Function

If the debounced function is triggered multiple times (like during user input), it will only execute the original function once the user has stopped typing for the specified delay

```
function debounce(func, delay) {
    let timeoutId;
    return function (...args) {
        clearTimeout(timeoutId);
        timeoutId = setTimeout(() => func.apply(this, args), delay);
    };
}
```
- `func`: The function to debounce.
- `delay`: The number of milliseconds to delay.

-FetchCVEData

Fetches CVE data from the API based on the current page, limit, and filters.

- `page`: The page number to fetch.
- `limit`: The number of results per page.

This function constructs the API URL, sends the request, and processes the response. It then calls `populateTable()`, updates the total records count, and refreshes the pagination.

-PopulateTable(vulnerabilities) :

Populates the table with the fetched CVE data.

- `vulnerabilities`: An array of vulnerability objects from the API response.

This function clears the existing table rows and creates new ones for each vulnerability, adding a click event listener to show details for each CVE.

4. **Applying Filters**

```
let params = [];
```

This initialises an empty array to hold the query parameters that will be appended to the URL for filtering results.

5. CVE ID Filter

```
if (cveIdFilter) {
    params.push(`cveId=${cveIdFilter}`);
}
```

Fetches and displays detailed information for a specific CVE.

- `cveId`: The ID of the CVE to fetch details for.

This function makes an API call for the specific CVE, then populates various elements in the `cveDetails` with the fetched data.

6. **Year Filter**

```
if (yearFilter) {
    const startDate = `${yearFilter}-01-01T00:00:00.000`;
    const endDate = `${yearFilter}-12-31T23:59:59.999`;
    params.push(`pubStartDate=${startDate}&pubEndDate=${endDate}`);
}
```

yearFilter is set, it constructs the start and end dates for the specified year. The dates are formatted to include the full year (from January 1 to December 31) and are added to the params array for filtering by publish date

7. **CVSS Score Filter**

```
if (cvssMinScore || cvssMaxScore) {
    let cvssFilter = 'cvssV2Metrics.baseScore:';

    if (cvssMinScore && cvssMaxScore) {
        cvssFilter += `[${cvssMinScore} TO ${cvssMaxScore}]`;
    } else if (cvssMinScore) {
        cvssFilter += `[${cvssMinScore} TO *]`;

    } else if (cvssMaxScore) {
```

```
            cvssFilter += `[* TO ${cvssMaxScore}]`;
        }
        params.push(cvssFilter);
    }
```

Using the if else condition to display the range of the Cvss min score and Cvss maxx score

8. Fetching Data:

```
    console.log('Fetching URL:', url);

    const response = await fetch(url);
    const data = await response.json();
    populateTable(data.vulnerabilities);
    totalRecords = data.totalResults;
    totalRecordsDisplay.textContent = `Total Records: ${totalRecords}`;
    updatePagination();
```

9.Date from the API response:

```
function formatDate(dateString) {
    const date = new Date(dateString);
    return date.toLocaleDateString('en-US', { day: '2-digit', month: 'short', year: 'numeric' });
}
```

Formats a date string into a more readable format.

dateString: The date string to format.

10.updatePagination():

Updates the pagination controls based on the current page and total number of records.

11.applyFilters():

A debounced function that applies the current filter values and fetches new data.

12.clearFilters():

Clears all filter inputs and resets the data fetch.