**NVD API DOCUMENTATION:**

**DATABASE:**

1. **CVE\_DATA:**

This Flask application connects to a MongoDB database and fetches CVE (Common Vulnerabilities and Exposures) data from the NVD (National Vulnerability Database) API. It transforms the fetched data into a structured format and stores it in the database. Key functionalities include formatting dates, transforming CVSS metrics, and handling configurations. The /store-cves endpoint triggers the process to fetch, process, and store CVEs. The app avoids duplicate entries by checking the database for existing records. It uses MongoEngine for database interactions and processes data in batches for efficient storage.

1. **CVE\_DATA\_FORMAT:**

This code defines the schema for storing CVE (Common Vulnerabilities and Exposures) data in a MongoDB database using MongoEngine. The CvssData and CvssMetricV2 classes describe the structure of CVSS metrics, including severity, exploitability, and impact scores. The Description class stores multilingual descriptions of vulnerabilities. The Node and Configuration classes represent the configurations and associated CPE (Common Platform Enumeration) matches. The CVE class acts as the main document to store detailed information about a CVE, including its ID, metadata, descriptions, metrics, and configurations. These schemas allow seamless storage and querying of structured CVE data within the MongoDB database.

**BACKEND:**

1. **SERVER:**

This Flask application provides a RESTful API for accessing and retrieving CVE (Common Vulnerabilities and Exposures) data from a MongoDB database. The /api/get-cves endpoint supports pagination, allowing users to fetch a specified number of CVE records per page along with metadata like total records and total pages. The /api/get-cve endpoint retrieves detailed information about a specific CVE by its ID, including descriptions, CVSS metrics, and configuration details such as vulnerable software criteria (CPE matches). The application uses Flask-CORS to handle cross-origin requests and integrates with MongoDB via PyMongo for data storage and retrieval. Both endpoints handle errors gracefully, ensuring proper user feedback in case of invalid requests or server issues.

**FRONTEND:**

1. **MAIN\_PAGE\_JAVASCRIPT:**

This JavaScript script dynamically fetches CVE (Common Vulnerabilities and Exposures) data from a RESTful API and displays it in an HTML table. The fetchCVEs function retrieves paginated data from the API using the currentPage and perPage variables. The table is updated with the CVE details like id, sourceIdentifier, and vulnerability status, and allows navigation to a details page for individual CVEs. Pagination controls (prevPage and nextPage) are dynamically enabled or disabled based on the current page and total pages. A changePerPage function allows users to adjust the number of records displayed per page. The showDetails function provides a detailed view of a selected CVE, including descriptions and CVSS V2 metrics, displaying them in a formatted section. Error handling and loading indicators ensure a seamless user experience during data fetching.

1. **CVE\_DETAILS\_JAVASCRIPT:**

The fetchCveDetails function retrieves and displays CVE details from a local API based on a CVE ID obtained from the URL query parameter. First, it checks if the CVE ID is provided; if not, it displays an error message. It then makes an asynchronous fetch request to the API using the CVE ID. If the request is successful, it parses the response and dynamically updates the HTML to show the CVE description, CVSS metrics (severity, score, vector string), and CPE configuration data. If the fetch fails, it shows an error message. The function is invoked when the page loads.