CVE Management System Documentation

This project is a web-based CVE (Common Vulnerabilities and Exposures) management system. It fetches CVE data from the NVD (National Vulnerability Database) API, stores it in a MySQL database, and exposes a set of API endpoints to query and display the CVE data. The system also includes a web interface for viewing the CVE details, filtering by specific parameters like CVE year, CVSS score, and modified date.

Requirements

- Python 3.x
- Flask
- Requests
- MySQL Server
- MySQL Python connector (`mysql-connector-python`)
- Jinja2 templates for rendering HTML

Setup Instructions

1. Install Dependencies

First, install the required Python libraries:

```
bash
CopyEdit
pip install flask requests mysql-connector-python
```

2. Set Up MySQL Database

- Install MySQL server if not already installed.
- Create a database named `cve_database` using the following command:

  ```
  sql
  CopyEdit
  CREATE DATABASE cve_database;
  ```

3. Configure MySQL Connection

Ensure that the MySQL connection details (host, user, password, and database) in the code match your environment. By default, the connection uses:

- `host`: localhost
- `user`: root
- `password`: Aravind@2003
- `database`: cve_database

4. Create the Table

The table `cve_details` will be created automatically on the first run of the application. It stores the CVE data fetched from the NVD API.

5. Run the Application

Run the Flask app:

```
bash
CopyEdit
python app.py
```

The application will run on `http://localhost:5000` by default.

6. Fetching CVE Data

The application periodically fetches CVE data from the NVD API, processes it, and stores it in the MySQL database. The `sync_cves` function runs in a separate background thread and periodically calls the NVD API to fetch new data.

7. Web Interface

The web interface consists of two main views:

- **Home Page**: The landing page redirects to the list of CVEs.
- **CVE List**: Displays a table of CVEs, which can be sorted and paginated.
- **CVE Details**: Displays detailed information about a specific CVE when clicked from the list.

8. API Endpoints

The following API endpoints are available:

GET `/api/cves/year/<int:year>`

- Fetches all CVEs for a given year.
- **Parameters**: `year` (integer)
- **Example**: `/api/cves/year/2022`

GET `/api/cves/score/<float:score>`

- Fetches all CVEs with a CVSS score greater than or equal to the specified score.
- **Parameters**: `score` (float)
- **Example**: `/api/cves/score/7.5`

GET `/api/cves/modified/<int:days>`

- Fetches all CVEs that were modified in the last N days.
- **Parameters**: `days` (integer)
- **Example**: `/api/cves/modified/30`

9. MySQL Schema

The `cve_details` table is created with the following schema:

```
sql
CopyEdit
CREATE TABLE IF NOT EXISTS cve_details (
    cve_id VARCHAR(255) PRIMARY KEY,
    description TEXT NOT NULL,
    published_date DATETIME NOT NULL,
    modified_date DATETIME NOT NULL,
    cvss_score FLOAT NOT NULL,
    cvss_v2_score FLOAT,
    year INT NOT NULL,
    status VARCHAR(50) NOT NULL DEFAULT 'new'
);
```

10. Data Flow

1. **Data Fetching**: The `fetch_cves` function fetches CVE data from the NVD API.
2. **Data Cleaning**: The `clean_data` function processes and extracts relevant information from the raw CVE data.
3. **Data Insertion**: The `store_cve` function stores the cleaned data in the MySQL database. If the CVE already exists, the function updates the existing record.
4. **Periodic Fetching**: The `sync_cves` function runs periodically in the background, fetching new data and updating the database.

11. Error Handling

- If a CVE entry has missing or malformed data, it is skipped.
- If the CVE ID already exists in the database, the entry is updated.
- API calls to the NVD API are retried up to 5 times if there is an error.

12. Example Usage

- **Home Page**: `http://localhost:5000/` - Displays a list of CVEs.
- **CVE List**: `http://localhost:5000/cves/list?page=1&resultsPerPage=10&sort=published_date&direction=ASC`
    - Displays a paginated list of CVEs sorted by the specified field and direction.
- **CVE Details**: `http://localhost:5000/cves/CVE-2022-1234`
    - Displays detailed information about a specific CVE.
- **API Endpoint**: `http://localhost:5000/api/cves/year/2022`
    - Returns all CVEs from the year 2022 in JSON format.

13. Enhancements and Future Work

- **Authentication**: Add authentication for restricted access to CVE data.
- **Search**: Implement search functionality for CVE description or ID.
- **Advanced Filtering**: Add more advanced filtering options based on CVE attributes.
- **Web Scraping**: If the NVD API is unavailable, consider adding a web scraping mechanism to fetch data directly from the website.