Code Documentation

Name : Tinesh Sakthi R

College : Dr.N.G.P.Institute of Technology

Project Name : NVD-CVE Databae Synchronization

Github : https://github.com/tineshsakthir/Securin-CVE-NVD-cybersecurity.git

Overview of the Document :

1.Tech Stack

2.Problem Statements:

- ✗ Consuming CVE data from the NVD's Database
- ✗ Controlling chunked responses
- ✗ Data cleaning and de-duplication
- ✗ CVE data synchronization (handling newly added CVEs and updated CVEs)
- ✗ API's for reading and filtering CVE details
- ✗ Reading API and visualizing in UI
- ✗ Controlling results per page
- ✗ Pagination functionality and total records
- ✗ Sorting

3.Logic and Description:

Detailed explanations of the logic and approaches used in the code and also about how i came up with the idea.

4.Code Flow for Synchronization Process:

Step-by-step explanation of code flow for initial loading, saving CVE data to the database, periodic synchronization, and CVE change API integration.

5.Backend API Documentation:

Detailed documentation of the various API endpoints, their URLs, methods, descriptions, query parameters, and responses.

6.Frontend and Backend Outputs:

Screenshots of the application's user interface and backend logs, demonstrating the synchronization and creation process, updation logs, and request logs.

7.How I come up with an Idea

Tech Stack :

Front End : React,React-Router-Dom,tailwind

Back End : Node.js,Express.js

Database : MongoDB, Mongoose(ORM)

Logic and Description to the Problem Statements :

1.Consume CVE from the NVD’s Database :

- NVD basically provides two APIs for the developers
  1.CVE API
  2.CVE change history API
- Initially, when the server starts, it checks the Db for data and fetches the CVE Api’s data if the Database is empty or just starts the periodic updation.
- For this i have choosen mongoDB, because it very easy to store a json in the mongoDB.
- But to just store the only the required details, i have designed a mongoose schema

2.Controlling Chuncked Responses :

- NVD’s Api by default limit the resultsPerPage to 2000,so i haven’t modified that.
- The recursive to the syncing funciton just sets the query parameter to the NVD’s API with different startIndex.
- The server makes the Api call Recursively with current startIndex+2000.
- These 2000 CVEs then are looped and stored in the local database seperately.

3.Data Cleaning & De-Duplication :

- The data duplication is avoided by checking whether there is already a CVE available with that particular cveId.
- Data cleaning is done by updating the fields as “Not Available”, which doesn’t have necessay details.

4.CVE Data Synchronization :

basically two parts.....

a) Handling newly added CVE to the API:

- Initially, when the Database is empty, all the CVEs are fetched from the Api and stored in the Database.
- But, after that all the CVEs are **not fetched** from from the Api to Synchronize.
- The Index at which the last Synchronization ended in stored in the Database, so when the **next Synchronization time** comes or when the **server restarts,** the server will start from that point and synchronizes the newly added CVEs to the local Database

b) Handling updated CVE in the API:

- This updated CVE data can be fetched from the **another API** called **CVE Change history Api** provided by NVD.
- The server will make an api call to this API in a p**eriodic manner** and gets the changes made to the CVE in that **Time Interval** and updates those particular CVEs

5. API's Read and Filter CVE details :

Three types of filtering can be done,

1.Filter by Base score.

2.Filter by CveId.

3.Filter by year.

6. Read the Api and visualize it in UI :

- During the initial loading of the front page, an api call will be made to the server to get the cve details.
- The server will provide the CVE details to the frontend as json response and it is used in the UI for visualisation.
- Similary when the user clicks a particular cve in the list, an another API call will be made to the server to fetch its details.

7. Controlling Results Per page :

- A select button is provided to the user to choose the Results Per Page as per their requirement.
- When the user changes the current Option, an api call will be made to the backend server to fetch those records..

8. Pagination Functionality and Total Records :

- Pagination is added as a feature in the app,it is editable thorugh the UI.
- For each Page, a separate api call will be made with the offset value and resultsPerPage as a query parameter to the backend server.
- A Api call is made to find the total number of CVEs in the Database

9. Sorting :

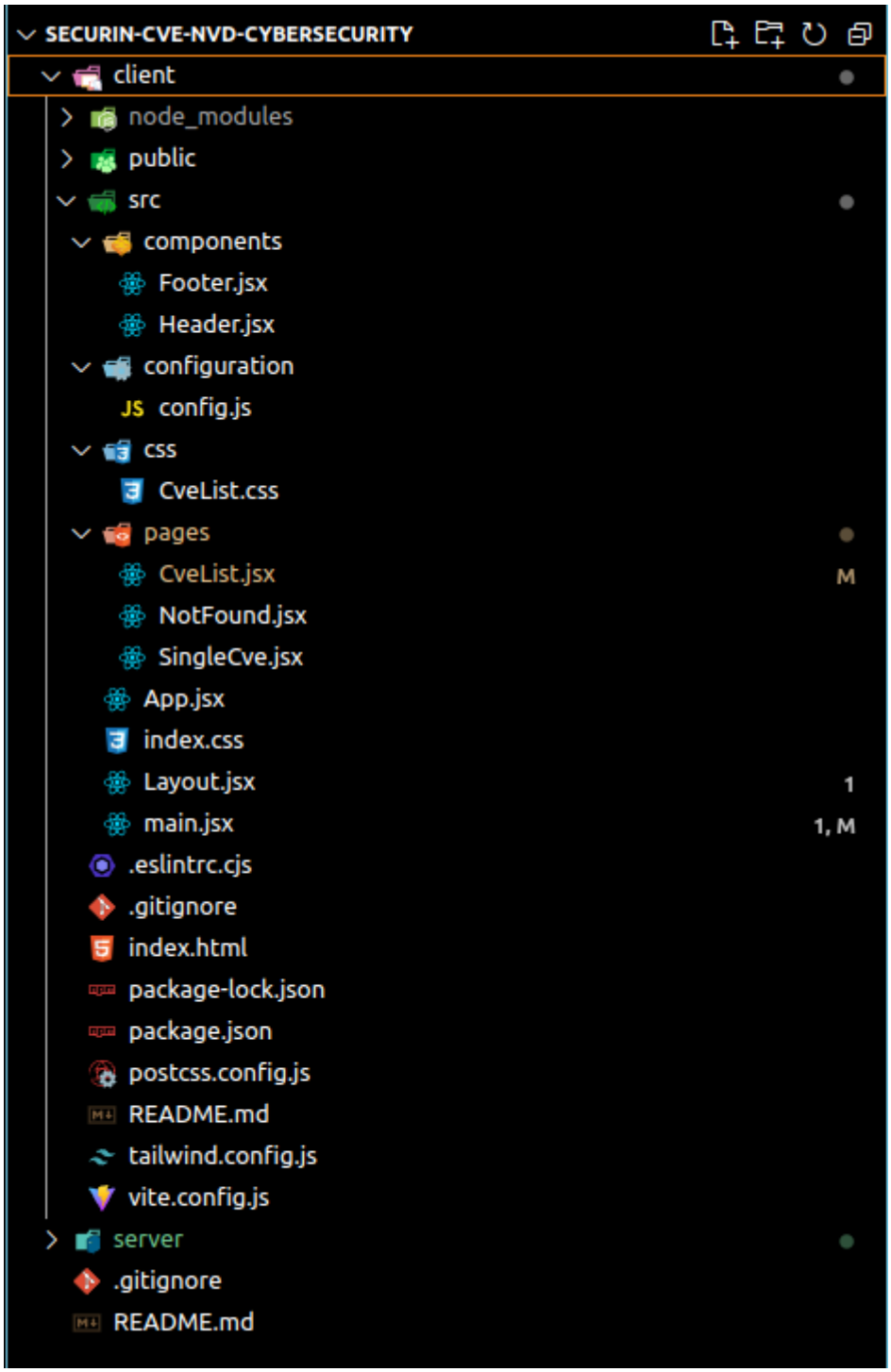
Sorting is done in the backend with the help of mongoose, which sorts the records according to the Published date and returns to the UI.

Code And Output Screenshots :

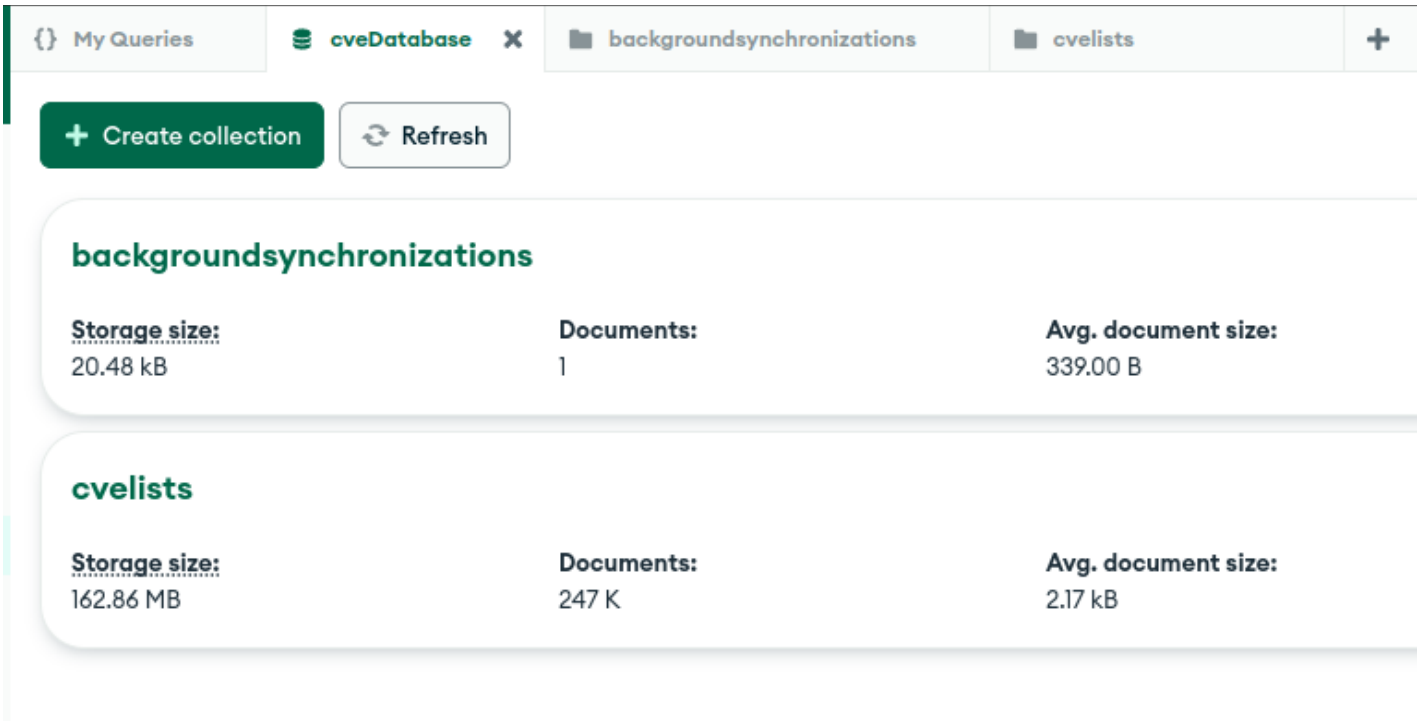
File Structure :

Server :

Client :


SECURIN-CVE-NVD-CYBERSECURITY
client
node_modules
public
src
components
Footer.jsx
Header.jsx
configuration
config.js
css
CveList.css
pages
CveList.jsx
M
NotFound.jsx
SingleCve.jsx
App.jsx
index.css
Layout.jsx
1
main.jsx
1, M
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
postcss.config.js
README.md
tailwind.config.js
vite.config.js
server
.gitignore
README.md

Totally two schema's are defined in the code for two collections :



DataBase Schema Design :

BackgroundSynchronizationSchema :

```
import mongoose from "mongoose";

const BackgroundSynchronizationSchema = new mongoose.Schema({
    synchronizedId : {type : Number , required : true} ,
    //The date in the cve api is in the format of ISO format,
    // here i stored in normal Date() format
    lastSynchronizedDateTime : {type : String , required : true} ,
    nextSynchronizationDateTime:{type : String , required : true},
    lastLeftIndex : {type : String , required : true} ,
    changeApilastSynchronized : {type : String , required : true}
})

export const BackgroundSynchronizationModel = mongoose
    .model("backgroundSynchronization" ,
        BackgroundSynchronizationSchema) ;      You, 1 second ago
```

Schema Definition :

1.lastSynchronizedDateTime : This parameter is mainly used to fix the next Synchrounization Time when the server starts.

2.nextSynchronizationDateTime : This parameter is used to determine whether the current time have crossed the next Synchrounization Time, if so then start the synchronization process.

3.lastLeftIndex : This parameter is used to determine the index where the server left the last synchronization process.

4.changeApiLastSynchronized : This parameter is only used by the ChangeApiDealer() function, to update the last Change Api Synchronized time .

Why two synchronization times ?

Consider a situation like, when the synchronization process starts, first the server calls both the handlers responsible for adding new Cves to the Database and updating the old cves in the database respectively

So, in that case, if the servers fails at the middle by just “adding new Cves to the database and by updating the **lastSynchronizedDateTime**”.

After that when the server starts, it will miss the changes made to the already existing CVEs in that previos time interval.

So here by we need to introduce another synchronization time only for the ChangeApiDealer,i.e.**changeApiLastSynchronized**

CVE Schema :

```
import mongoose from 'mongoose'

const CveListSchema = new mongoose.Schema({
    cveId: { type: String, required: true },
    sourceIdentifier: { type: String, required: true },
    published: { type: String, required: true },
    lastModified: { type: String, required: true },
    vulnStatus: { type: String, required: true },
    descriptions: [
        {
            lang: { type: String, required: true },
            value: { type: String, required: true }
        }
    ],

    // ===cvssMetricV2 ===
    baseSeverity: { type: String, required: true },
    baseScore: { type: String, required: true },
    vectorString: { type: String, required: true },
    accessVector: { type: String, required: true },
    accessComplexity: { type: String, required: true },
    authentication: { type: String, required: true },
    confidentialityImpact: { type: String, required: true },
    integrityImpact: { type: String, required: true },
    availabilityImpact: { type: String, required: true },

    // ===Score===
    exploitabilityScore: { type: String, required: true },
    impactScore: { type: String, required: true },

    // ===CPE====
    cpeMatch: [
        {
            vulnerable: { type: String, required: true },
            criteria: { type: String, required: true },
            matchCriteriaId: { type: String, required: true }
        }
    ]
})

export const CveListModel = mongoose.model('cveList', CveListSchema)
```

Schema Definition :

All the above fields are necessary as per the sample UI

Code Flow for Synchronization process By the Server:

Step 1 : Initial Loading and Storing Api Data

```
setTimeout(async () => {
    try {
        syncProcessLogger(`\n<<<---Synchronization Started At the Server--->>>\n`)

        //Gets the count from the controller
        const count = await getBgSyncModelCount();

        //If  count is 0, then first sync the data and then call the periodic sync
        if (count === 0) {
            syncProcessLogger(`Calling initialSynchronizeDataFromAPI()...`)
            await initialSynchronizeDataFromAPI();
            await periodicUpdateToSynchronizeDataFromAPI();
        }
        //If there is base data, then just call the periodic udpate handler....
        else {
            syncProcessLogger(`\nData already exists in the database.
         So just calling periodicUpdateToSynchronizeDataFromAPI`)

            await periodicUpdateToSynchronizeDataFromAPI();
        }
    } catch (error) {
        errorLogger(`Error occured while starting the setTimeOut - ${error}`);
    }
}, 5000); // This time out is needed to bear the start of the server,
//  without using this time out the server can't call an async call back function.
```

The above code first checks whether there is already the CVEs are present in the db and then starts the synchronization process according to it

Step 2 : Saving CVE data to the Db

```
const initialSynchronizeDataFromAPI = async () => {
    try {
        syncProcessLogger("\n<<<---At initialSynchronizeDataFromAPI()--->>>\n");

        const backgroundSynchronizationModel = await getBackgroundSynchronizationModel();
        let startIndex = 0;
        if (backgroundSynchronizationModel) {
            startIndex = parseInt(backgroundSynchronizationModel.lastLeftIndex);
        }
        createLogger(`At the offset : ${startIndex}`)
        // fetching the api and putting with the query parameter as startIndex
        const response = await axios.get(
            `https://services.nvd.nist.gov/rest/json/cves/2.0/?startIndex=${startIndex}`
        );
        const totalCve = response.data.vulnerabilities;
        // Process and save data to the database
        for (const singleCve of totalCve) { await saveNewCve(singleCve);  }
```

- ➔ This Synchronization function is reponsible for saving the **chunked responses** from the Api.
- ➔ This funciton is designed in a way that, it can be used for both **initial loading** of the data and **periodic adding** of the **data** to the local database.
- ➔ This **two functionality** is achieved in a single function with the help of the **BackgroundSynchronizationModel's lasteLeftIndex**.
- ➔ This Function will call it self **recursively** untill there is data in the **API to feed**.
- ➔ Finally, the **offset** at which the process stopped will be **stored in the Database** for the future synchronization.

Purpose of BackgroundSyncModel in initialSync function :

- if there is backgroundSync model, then we **get the startIndex** from it, and **update** the BackgroundSync model in the db

- orElse we **initialize** the startIndex as 0 and **create** the BackgroundSync model in the db

Step 3 : Periodic Synchronization

```
const periodicUpdateToSynchronizeDataFromAPI = async () => {      You, 2 seconds ago • Uncommitted changes
    try {
        syncProcessLogger("\n<<<---At periodicUpdateToSynchronizeDataFromAPI()--->>>\n");
        const backgroundSynchronization = await getBackgroundSynchronizationModel();
        const currentDateTime = new Date();
        const nextSynchronizationDateTime = new Date(
            backgroundSynchronization.nextSynchronizationDateTime
        );
        const timeDifference = nextSynchronizationDateTime - currentDateTime;
        //If the server starts after the next Sync time, then the next sync should happen immediately....
        const delayForSetTimeout = timeDifference >= 0 ? timeDifference : 0;

        syncProcessLogger(`\nNext Synchronization in ${delayForSetTimeout} ms `);
        // When the server starts next Synchronizing time will be determinded as above
        setTimeout(async () => {
            await changeAPiDealer();
            await initialSynchronizeDataFromAPI();
            syncProcessLogger(`\nNext Synchronization in ${regularTimeInterval} ms`);
            //After the initial sync, then the periodic sync is started with a regular time Interval
            setInterval(async () => {
                await changeAPiDealer();
                await initialSynchronizeDataFromAPI();
                syncProcessLogger(`\nNext Synchronization in ${regularTimeInterval} ms`);
            }, regularTimeInterval);
        }, delayForSetTimeout);
    } catch (error) {
        errorLogger(`Error occurred while updating synchronization from the API - ${error}`);
    }
};
```

➔ In this synchronization function, it periodically call the **changeApiDealer()** and **initialSync()** function.

➔ Here the regular Time is setted at 2 minutes so the sync happens at end of every 2 minutes.

Step 4 : CVE Change API Integration

```
const changeAPiDealer = async () => {      You, 4 seconds ago • Uncommitted chan
  try {
    syncProcessLogger("\n<<<---At changeAPiDealer()--->>>\n");
    const backgroundSynchronization = await getBackgroundSynchronizationModel();
    //With out this toISOString(), the NVD Api is not accepting the api call
    const changeStartDate = new Date(
      backgroundSynchronization.changeApilastSynchronized
    ).toISOString();
    const changeEndDate = new Date().toISOString();
    //Encode the DateTIme offset into %2B, because NVD Api is expecting it.....
    const encodedChangeStartDate = encodeURIComponent(changeStartDate);
    const encodedChangeEndDate = encodeURIComponent(changeEndDate);
    // API endpoint URL
    const apiUrl = `https://services.nvd.nist.gov/rest/json
          /cvehistory/2.0/?changeStartDate=${encodedChangeStartDate}
          &changeEndDate=${encodedChangeEndDate}`;

    syncProcessLogger(`The api url for change Api :${apiUrl} `);

    const response = await axios.get(apiUrl);
    const data = response.data;

    if (data.resultsPerPage == 0) {
      updateLogger(`Not any updation done in this time Interval`);
      return;
    }
    let semophore = true;
    let i = 0;
    const size = data.cveChanges.length;

    let intervalId = setInterval(async () => {
      if (semophore) {
        semophore = false;
        if (i >= size) {
          clearInterval(intervalId);
          //THis mainly helps even when the server got shut down at the middle
          await updateBgSynchByChangeApi(backgroundSynchronization);
          return;
        }
        await updateExistingCve(data.cveChanges[i].change.cveId);
        i++;
        semophore = true;
      }
    }, 10000);
  } catch (error) {
    errorLogger(`Error Dealing while Dealing with Change API: ${error}`);
```

- ➔ The function is **mainly** responsible for the talking with the **another Api** to just get **all the CVEs**, in which the **changes are made** in that time interval
- ➔ **CVE Change History API** requires both the **changeStartDate** and **changeEndDate** as query paramerters.
- ➔ It will respond with the **list of cves** in which the **changes** have been made.
- ➔ We can use that to just **update our local CVE**, with the help of the **CVE API**

[Notes : The usage of setInterval is just to slow down the API calls, because NVD is blocking our request, when we exceed their API call rate limits. That's the reason for using the setInterval in all the API calls]

Backend API DOCUMENTATION :

Base Url :

The base Url for all endpoints is just : '/'

Endpoints :

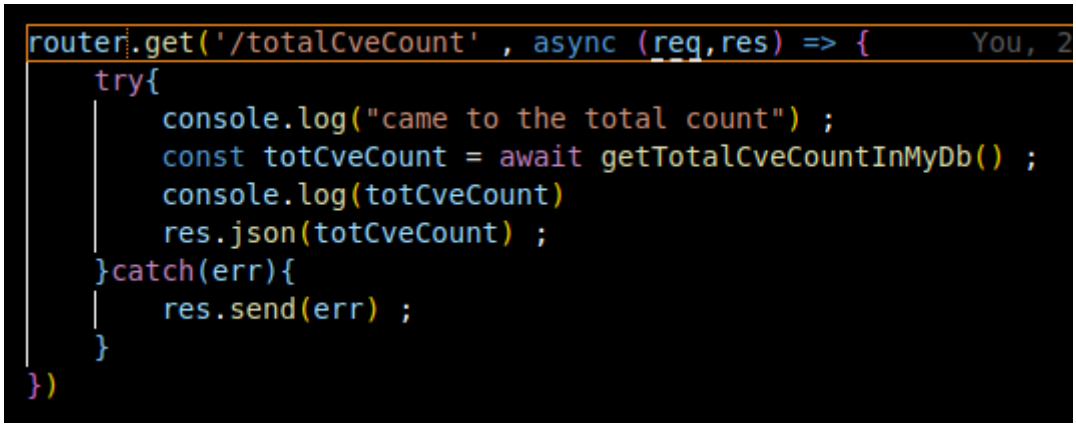
1.Get Total CVE counts :

URL: `/totalCveCounts`

Method : GET

Description : Retrives the total count of CVEs stored in the database.

Response : `count`(Number): Total coutn of the Cves in the databases.



```
router.get('/totalCveCount' , async (req,res) => {      You, 2
    try{
        console.log("came to the total count") ;
        const totCveCount = await getTotalCveCountInMyDb() ;
        console.log(totCveCount)
        res.json(totCveCount) ;
    }catch(err){
        res.send(err) ;
    }
})
```

2. Get CVE List

URL: `/cveList`

Method: GET

Description: Retrieves a list of CVEs from the database.

Query Parameters:

count(Number): Number of CVEs to retrieve.

offset(Number): Offset for pagination.

Response:

cveList (Array): Array of CVE objects.

```
router.get('/cveList' , async (req,res)=>{
    try{
        const count = 10 ;
        if(req.query.count){
            count =  req.query.count ;
        }
        const offset = 0 ;
        if(req.query.offset){
            offset = req.query.offset ;
        }
    const cveList = await getCveListFromMyDb(count,offset) ;
    res.json(cveList) ;
    }catch(err){
        res.send(err) ;
    }
})
```

3. Get Single CVE

URL: `/cve`

Method: GET

Description: Retrieves details of a single CVE from the database.

Query Parameters:

cveId (String): Identifier of the CVE.

Response:

cve (Object): Details of the requested CVE.

```
router.get('/cve' , async (req,res) => {
    try{
        const cveId = req.query.cveId ;
        const cve = await getSingleCveFromMyDb(cveId) ;
        res.json(cve) ;
    }catch(err){
        res.send(err) ;
    }
})
```

4. Get CVE List(filter by score)

URL: `/cveListByScore`

Method: GET

Description: Retrieves a list of CVEs from the database by filtering them according to the provided Base score.

Query Parameters:

score(Number):baseScore of the CVEs.

count(Number): Number of CVEs to retrieve.

offset(Number): Offset for pagination.

Response:

cveList (Array): Array of CVE objects.

```
router.get('/cveListByScore', async (req, res) => {
    try {
        const score = req.query.score;
        const offset = req.query.offset;
        const count = req.query.count;
        const cveList = await getCveListByScore(score, offset, count);
        res.json(cveList);
    } catch (err) {
        res.send(err);
    }
})
```

5. Get CVE List(filter by year)

URL: `/cveListByYear`

Method: GET

Description: Retrieves a list of CVEs from the database by filtering them according to the provided publised Year.

Query Parameters:

year(Number):Publised Year of the CVEs.

count(Number):Number of CVEs to retrieve.

offset(Number): Offset for pagination.

Response:

cveList (Array): Array of CVE objects.

```
router.get('/cveListByYear', async (req, res) => {      You, 43 minutes ag
    try {
        const year = req.query.year;
        const offset = req.query.offset;
        const count = req.query.count;
        const cveList = await getCveListByYear(year, offset, count);
        res.json(cveList);
    } catch (err) {
        res.send(err)
    }
})
```

Code Modularization :

Sample Scenario :

During the front page load, the client makes a request to the server for the list of CVE from the **CveList.jsx**

```
React.useEffect(() => {
    fetch(
        `http://localhost:${config.backendPort}/cveList?count=${resultsPerPage}&offset=${skip}`
    )
        .then((response) => response.json())
        .then((data) => {
            console.log(data);
            setData(data);
        })
        .catch((error) => {
            console.error("Error fetching CVE data:", error);
        });
}, [skip, resultsPerPage]);
```

The request is recieved to **server.js** file and applied some middlewares and after that the request is routed to a another file **cve.js**(routes file) for the seperation of concerns.

```
//Middlewares
//Cors configuration for allowing the React App to seek the endPoint with a request
import cors from "cors";
app.use(cors());


//Adding log for all the requests that are comming from the client
app.use(requestLogger) ;


//Routes for cve Api
app.use("/", cveRouter);
```

Now the request and response objects are recieved and the query parameters are retrieved from the request in the **cve.js**

```
router.get('/cveList', async (req, res) => {      You, 2 hours ago • U
    try {
        const count = req.query.count;
        const offset = req.query.offset;
        const cveList = await getCveListFromMyDb(count, offset);
        res.json(cveList);
    } catch (err) {
        res.send(err);
    }
})
```

But to get that list of Cve from the database we need to talk to a controller file **controller.js**, now the request is made to a **function** to get the list from the database.

```
const getCveListFromMyDb = async (count, offset) => {
    try {
        const cveList = await CveListModel.find({})
        .sort({ published: 1, lastModified: -1 })
        .skip(offset).limit(count);
        return cveList;
    } catch (err) {
        errorLogger(`Error occured while fetching CVE list from the Db - ${err}`);
    }
}
```

The controller is the only place where all database operations occurs.

Now the response will go in the same route as it comes.....

Frontend Output :

CVE list page :

National Vulnerability Database

Base Score: Search by Base Score

Base Year: Search by Base Year

CVE ID: Search by CVE ID

Total records:247237

CVE ID	Source Identifier	Published	Last Modified	Vulnerability Status
CVE-1999-0095	cve@mitre.org	Oct 01, 1988	Jun 11, 2019	Modified
CVE-1999-0082	cve@mitre.org	Nov 11, 1988	Sep 09, 2008	Analyzed
CVE-1999-1471	cve@mitre.org	Jan 01, 1989	Sep 05, 2008	Analyzed
CVE-1999-1122	cve@mitre.org	Jul 26, 1989	May 03, 2018	Modified
CVE-1999-1467	cve@mitre.org	Oct 26, 1989	Dec 19, 2017	Modified
CVE-1999-1506	cve@mitre.org	Jan 29, 1990	Sep 05, 2008	Analyzed
CVE-1999-0084	cve@mitre.org	May 01, 1990	Oct 10, 2017	Modified
CVE-2000-0388	cve@mitre.org	May 09, 1990	Sep 10, 2008	Analyzed
CVE-1999-0209	cve@mitre.org	Aug 14, 1990	Sep 09, 2008	Analyzed
CVE-1999-1392	cve@mitre.org	Oct 03, 1990	Sep 05, 2008	Analyzed

Results Per Page : 10

prev page 1 next page

Made by Tinesh Sakthi R

Filter By Score :

National Vulnerability Database

Base Score: 10 Search by Base Score

Base Year: Search by Base Year

CVE ID: Search by CVE ID

Total records:247237

CVE ID	Source Identifier	Published	Last Modified	Vulnerability Status
CVE-1999-0095	cve@mitre.org	Oct 01, 1988	Jun 11, 2019	Modified
CVE-1999-0082	cve@mitre.org	Nov 11, 1988	Sep 09, 2008	Analyzed
CVE-1999-1467	cve@mitre.org	Oct 26, 1989	Dec 19, 2017	Modified
CVE-1999-1193	cve@mitre.org	May 14, 1991	Oct 10, 2017	Modified
CVE-1999-0498	cve@mitre.org	Sep 27, 1991	Aug 17, 2022	Modified
CVE-1999-1493	cve@mitre.org	Dec 18, 1991	Dec 19, 2017	Modified
CVE-1999-1032	cve@mitre.org	Dec 31, 1991	Oct 10, 2017	Modified
CVE-1999-1059	cve@mitre.org	Feb 25, 1992	Sep 05, 2008	Analyzed
CVE-1999-1119	cve@mitre.org	Apr 27, 1992	Oct 10, 2017	Modified
CVE-1999-0214	cve@mitre.org	Jul 21, 1992	Aug 17, 2022	Modified

Results Per Page : 10

prev page 1 next page

Made by Tinesh Sakthi R

Filter by Year :

National Vulnerability Database

Base Score: Search by Base Score

Base Year: 1991 Search by Base Year

CVE ID: Search by CVE ID

Total records:247237

CVE ID	Source Identifier	Published	Last Modified	Vulnerability Status
CVE-1999-1194	cve@mitre.org	May 01, 1991	Oct 10, 2017	Modified
CVE-1999-1193	cve@mitre.org	May 14, 1991	Oct 10, 2017	Modified
CVE-1999-1123	cve@mitre.org	May 20, 1991	Dec 19, 2017	Modified
CVE-1999-1034	cve@mitre.org	May 23, 1991	Oct 10, 2017	Modified
CVE-1999-1415	cve@mitre.org	Aug 23, 1991	Sep 05, 2008	Analyzed
CVE-1999-1090	cve@mitre.org	Sep 10, 1991	Oct 10, 2017	Modified
CVE-1999-0498	cve@mitre.org	Sep 27, 1991	Aug 17, 2022	Modified
CVE-1999-1468	cve@mitre.org	Oct 22, 1991	Sep 10, 2008	Analyzed
CVE-1999-0167	cve@mitre.org	Dec 06, 1991	Aug 17, 2022	Modified
CVE-1999-1493	cve@mitre.org	Dec 18, 1991	Dec 19, 2017	Modified

Results Per Page : 10

prev page 3 next page

Made by Tinesh Sakthi R

Search By CVE ID :

National Vulnerability Database

Base Score: Search by Base Score

Base Year:1991 Search by Base Year

CVE ID:CVE-1999-1194 Search by CVE ID

National Vulnerability Database

CVE-1999-1194

Description:

chroot in Digital Ultrix 4.1 and 4.0 is insecurely installed, which allows local users to gain privileges. (en)
chroot en Digital Ultrix 4.1 y 4.0 es instalado de forma insegura, lo que permite a usuarios locales para ganar privilegios. (es)

CVSS V2 Metrics:

Access Vector	Access Complexity	Authentication	Confidentiality Impact	Integrity Impact	Availability Impact
LOCAL	LOW	NONE	COMPLETE	COMPLETE	COMPLETE

Score:

Exploitability Score: 3.9 Impact Score: 10

CPE:

Criteria	Match Criteria ID	Vulnerable
cpe:2.3:o:digital:ultrix:4.0:*:*:*:*:*:*:*	18EE0235-30CD-4104-ADB8-45AA5B3CBC3A	true
cpe:2.3:o:digital:ultrix:4.1:*:*:*:*:*:*:*	0699893A-AE83-4605-BF2B-FC0C17BF2A69	true

Made by Tinesh Sakthi R

Click an CVE row to view it:

National Vulnerability Database

Base Score: Search by Base Score

Base Year: Search by Base Year

CVE ID: Search by CVE ID

Total records:247237

CVE ID	Source Identifier	Published	Last Modified	Vulnerability Status
CVE-1999-0095	cve@mitre.org	Oct 01, 1988	Jun 11, 2019	Modified
CVE-1999-0082	cve@mitre.org	Nov 11, 1988	Sep 09, 2008	Analyzed
CVE-1999-1471	cve@mitre.org	Jan 01, 1989	Sep 05, 2008	Analyzed
CVE-1999-1122	cve@mitre.org	Jul 26, 1989	May 03, 2018	Modified
CVE-1999-1467	cve@mitre.org	Oct 26, 1989	Dec 19, 2017	Modified
CVE-1999-1506	cve@mitre.org	Jan 29, 1990	Sep 05, 2008	Analyzed
CVE-1999-0084	cve@mitre.org	May 01, 1990	Oct 10, 2017	Modified
CVE-2000-0388	cve@mitre.org	May 09, 1990	Sep 10, 2008	Analyzed
CVE-1999-0209	cve@mitre.org	Aug 14, 1990	Sep 09, 2008	Analyzed
CVE-1999-1392	cve@mitre.org	Oct 03, 1990	Sep 05, 2008	Analyzed

Results Per Page : 10

prev page 1 next page

National Vulnerability Database

CVE-1999-0082

Description:

CWD ~root command in ftpd allows root access. (en)

CVSS V2 Metrics:

Access Vector	Access Complexity	Authentication	Confidentiality Impact	Integrity Impact	Availability Impact
NETWORK	LOW	NONE	COMPLETE	COMPLETE	COMPLETE

Score:

Exploitability Score: 10 Impact Score: 10

CPE:

Criteria	Match Criteria ID	Vulnerable
cpe:2.3:a:ftp:ftp:*:*:*:*:*:*:*:*	30D7F58F-4C55-4D19-984C-79B6C9525BEB	true
cpe:2.3:a:ftpcd:ftpcd:*:*:*:*:*:*:*:*	1D85A7F5-C187-4707-8681-F96A91F58318	true

Made by Tinesh Sakthi R

Pagination and Results Per page:

Results Per Page : 10
10
20
30
3000

Made by Tinesh Sakthi R

prev page 1 next page

Backend Outputs :

Synchronization and Creation Log :

```
****************************************************************************************************
20240501    17:48:47    dff0847f-db2e-4456-81ce-1ff8bea4fd17
<<<---Connected to MongoDB--->>>

20240501    17:48:52    ff18257f-f5e6-4492-91b2-1a781a5c7af1
<<<---Synchronization Started At the Server--->>>

20240501    17:48:52    ab0f7c7f-55b1-47a7-96dc-e5d4639e791e
Data already exists in the database. So just calling periodicUpdateToSynchronizeDataFromAPI
20240501    17:48:52    8846099d-c882-4acc-aa22-c3bbbcd8a658
<<<---At periodicUpdateToSynchronizeDataFromAPI()--->>>

20240501    17:48:52    3fcccc62-c4c1-43bd-8829-00f8e8895029
Next Synchronization in 0 ms
20240501    17:48:52    222a3080-f340-452d-a2cd-fb9551b2f9bf
<<<---At changeAPiDealer()--->>>

20240501    17:48:52    eb152d0f-1b96-456d-b767-9353bc6d9a48
The api url for change Api :https://services.nvd.nist.gov/rest/json/cvehistory/2.0
/?changeStartDate=2024-05-01T11%3A20%3A49.000Z&changeEndDate=2024-05-01T12%3A18%3A52.236Z
20240501    17:48:52    686cb804-e5de-4e0c-ae7b-569788947f4f
<<<---At initialSynchronizeDataFromAPI()--->>>

20240501    17:48:52    f3551866-2b0d-4b46-b093-197299cc5f2b
Next Synchronization in 120000 ms
20240501    17:50:52    085c8f3d-706e-4ef2-83c9-3f653e0db0a8
<<<---At changeAPiDealer()--->>>

20240501    17:50:52    2cbf48fc-d921-43a8-9edc-1ec75920945a
The api url for change Api :https://services.nvd.nist.gov/rest/json/cvehistory/2.0
/?changeStartDate=2024-05-01T11%3A20%3A49.000Z&changeEndDate=2024-05-01T12%3A20%3A52.281Z
20240501    17:50:52    4bc4394c-1304-4684-88f5-70c3ba01a136
<<<---At initialSynchronizeDataFromAPI()--->>>

20240501    17:50:52    37c10a39-9b1b-42d9-9286-64f8d6234cd9
Next Synchronization in 120000 ms
```

Updation Log : (during periodic updation cycle-every 2 minutes)

Updates with the help of CVE change history API

```
851  ********************************************************************************************************
852  20240501    16:50:09    539b602b-543c-49f6-93bd-4697321c74b8    Updating Existing CVE, cveId : CVE-2024-32973
853  20240501    16:50:11    1c076a1d-6ca2-4ae9-90de-1b9ddef12129    cveId : CVE-2024-32973 Updated
854  20240501    16:50:19    e95f248d-ed36-45a0-9b22-26579a2e0546    Updating Existing CVE, cveId : CVE-2024-32979
855  20240501    16:50:20    75d7a8c2-14a8-4923-8e0c-481b31e498c4    cveId : CVE-2024-32979 Updated
856  20240501    16:50:29    9ad37d01-2679-4796-b085-092a8e5866d2    Updating Existing CVE, cveId : CVE-2024-32984
857  20240501    16:50:31    382de462-1f64-4409-975f-914d207fe2e8    cveId : CVE-2024-32984 Updated
858  20240501    16:50:39    1f0e8151-b57b-4929-9df4-17f67473f163    Updating Existing CVE, cveId : CVE-2024-33835
859  20240501    16:50:41    a8b7ed50-c435-4bce-b492-0a74d6eff40a    cveId : CVE-2024-33835 Updated
860  20240501    16:52:02    0f4e5c83-32c7-4daa-9300-29be2b3d62c7    Not any updation done in this time Interval
861  20240501    16:54:03    fc4ebd2f-86f3-43a0-8b09-b2d260accd3d    Not any updation done in this time Interval
862  20240501    16:54:16    75d84ae9-9634-4d94-a3fe-b327f77e9613
863  ********************************************************************************************************
864  20240501    16:56:05    7673f93b-ec4d-43c5-92ce-ce8d9dd87248    Not any updation done in this time Interval
865  20240501    16:58:09    c4f59f61-a39c-44ec-b93c-faec142935c3    Not any updation done in this time Interval
866  20240501    17:00:11    e03ab09a-d789-4c15-96d6-2038d2921bc2    Not any updation done in this time Interval
867  20240501    17:02:08    c6c9aa4b-7d04-4606-a68b-f3e5624c1a84    Not any updation done in this time Interval
868  20240501    17:04:09    c9069a7e-0cd0-4b34-93b9-8c9c317be080    Not any updation done in this time Interval
869  20240501    17:05:19    41460e66-4f38-4863-8fce-59cb1ed179bd
```

Request Log :

```
470  20240501    18:05:09    cf1d5302-a7e9-4919-943f-95c2905fbcb9    GET http://localhost:5173   /cveList?count=10&offset=0
471  20240501    18:05:10    6bf86b63-a453-4a1d-8c2a-3406f712fcf1    GET http://localhost:5173   /totalCveCount
472  20240501    18:05:10    39274d67-0205-46ae-afff-0e9d480cc70b    GET http://localhost:5173   /cveList?count=10&offset=0
473  20240501    18:05:12    75790fa3-c185-4e6b-8ab2-4719e0fae44a    GET http://localhost:5173   /cve?cveId=CVE-1999-0095
474  20240501    18:05:12    6f4ba088-bdfe-4dd0-8ac2-4bbdabbca3a0    GET http://localhost:5173   /cve?cveId=CVE-1999-0095
475  20240501    18:05:14    135a22c2-da89-40f0-828f-486c57a40e3a    GET http://localhost:5173   /totalCveCount
476  20240501    18:05:14    ce9b055e-dc72-4cc7-9bd1-b17270f6d772    GET http://localhost:5173   /cveList?count=10&offset=0
477  20240501    18:05:14    25aec5b0-f969-4d33-9488-9728d988f695    GET http://localhost:5173   /totalCveCount
478  20240501    18:05:14    44a659a5-3a00-4c0d-9d60-e4200d0bd70e    GET http://localhost:5173   /cveList?count=10&offset=0
479  20240501    18:05:27    95ac8111-8929-457b-9604-6db6fdfc09e8    GET http://localhost:5173   /cve?cveId=CVE-1999-0095
480  20240501    18:05:27    b601e51f-8e29-4096-b247-d4e297bde1bc    GET http://localhost:5173   /cve?cveId=CVE-1999-0095
481  20240501    18:05:30    16b473f6-eb99-4fce-92dd-8866b040dc24    GET http://localhost:5173   /totalCveCount
482  20240501    18:05:30    e42e22b4-dce5-4971-a183-bd9d28d74b67    GET http://localhost:5173   /cveList?count=10&offset=0
483  20240501    18:05:30    38e632f4-899f-4288-a495-d58c6077f823    GET http://localhost:5173   /totalCveCount
484  20240501    18:05:30    232dbcba-d3a6-41a1-9354-4743aea6489c    GET http://localhost:5173   /cveList?count=10&offset=0
485  20240501    18:14:29    898baea7-96c1-432a-95ac-0552d85fc0ed    GET http://localhost:5173   /cveListByScore?score=10&offset=0&count=10
486  20240501    18:14:57    7a505c7c-dd15-4bb6-aa42-1d8612830b13    GET http://localhost:5173   /cveList?count=10&offset=10
487  20240501    18:14:57    d2f105d9-7411-4496-bf27-b322a03461ba    GET http://localhost:5173   /cveList?count=10&offset=20
488  20240501    18:16:12    91365165-4480-4055-90f6-4434ed67e845    GET http://localhost:5173   /cve?cveId=CVE-1999-1194
489  20240501    18:16:12    850233bb-7cc9-4af0-8999-2f38cf5e0714    GET http://localhost:5173   /cve?cveId=CVE-1999-1194
490  20240501    18:16:35    d98afcb7-b530-497b-b806-d294cf19e8b2    GET http://localhost:5173   /totalCveCount
491  20240501    18:16:35    58cb1a68-5795-4489-beb3-7d789ebea090    GET http://localhost:5173   /cveList?count=10&offset=0
492  20240501    18:16:36    5b6994f2-cd8f-4ad2-aa85-3d11421cb5e8    GET http://localhost:5173   /totalCveCount
493  20240501    18:16:36    800ac544-f3f2-49c5-b83d-8f73570589ba    GET http://localhost:5173   /cveList?count=10&offset=0
494  20240501    18:17:58    51c27bbc-7e8b-4044-a888-480409bb55bb    GET http://localhost:5173   /cve?cveId=CVE-1999-0082
495  20240501    18:17:58    2bee6ba3-cc6e-4ee2-b566-31a5c5fbd4ac    GET http://localhost:5173   /cve?cveId=CVE-1999-0082
496  20240501    18:20:29    0da4cd1a-2ea3-44b6-aa5c-9dcf180e4331    GET http://localhost:5173   /totalCveCount
497  20240501    18:20:29    5910a655-b773-4044-a503-cabea23c9c3f    GET http://localhost:5173   /cveList?count=10&offset=0
498  20240501    18:20:29    9beb4164-243a-4a4f-b2dd-e067623a982f    GET http://localhost:5173   /totalCveCount
499  20240501    18:20:29    a7cdec67-077d-49a7-b303-085f291dfd80    GET http://localhost:5173   /cveList?count=10&offset=0
```

HOW I COME UP WITH THE IDEA :

Storing part :

- Initially i started designing the backend, where i first just setted up the storing of the data to the database.
- First i stored the entire 2000 results per page in a sinle document of a collections.
- Similarly i stored all the 2000 results chunks as a seperate document.
- But i several issues while fetching the data for the ui, so i decided to store all the cves seperately with a schema.
- Here we don't need a schema, but i used it to do the clensing and de-duplication process in the server itself.

Periodic Updation :

- This part is very crucial part, because, in the initial phase of this part, i just decided to loop around all the 2.5lakhs data again and again from the Api in a regular time interval.
- But that is a very bad idea. Because in some cases, for just updating one record, we need to loop around the entire Api data.
- Only after two days, i found that there is an another API called CVE change history Api.
- With the help of that, i updated the existing cves seamlessly.
- For handling newly added cves, i saved the last left index in the db, so that when ever the server start, it starts from that index.

Code Documentation Completed.Have a great Day.