Assessment – 1
Matam Preethi
Amrita Vishwa Vidyapeetham
CB.EN.U4CSE20438

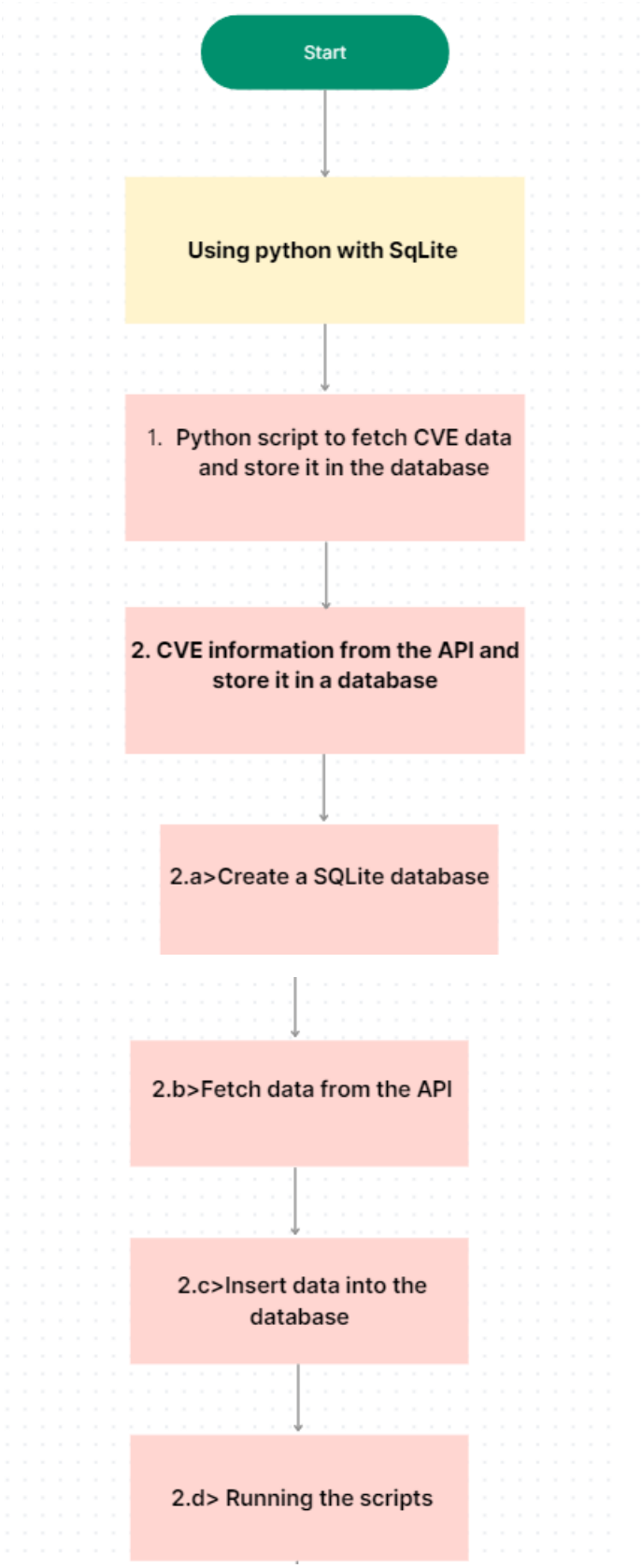
NVD - CVE API: The CVE API is used to easily retrieve information on a single CVE or a collection of CVE from the NVD. Pls refer to the below NVD CVE documentation to get more information.

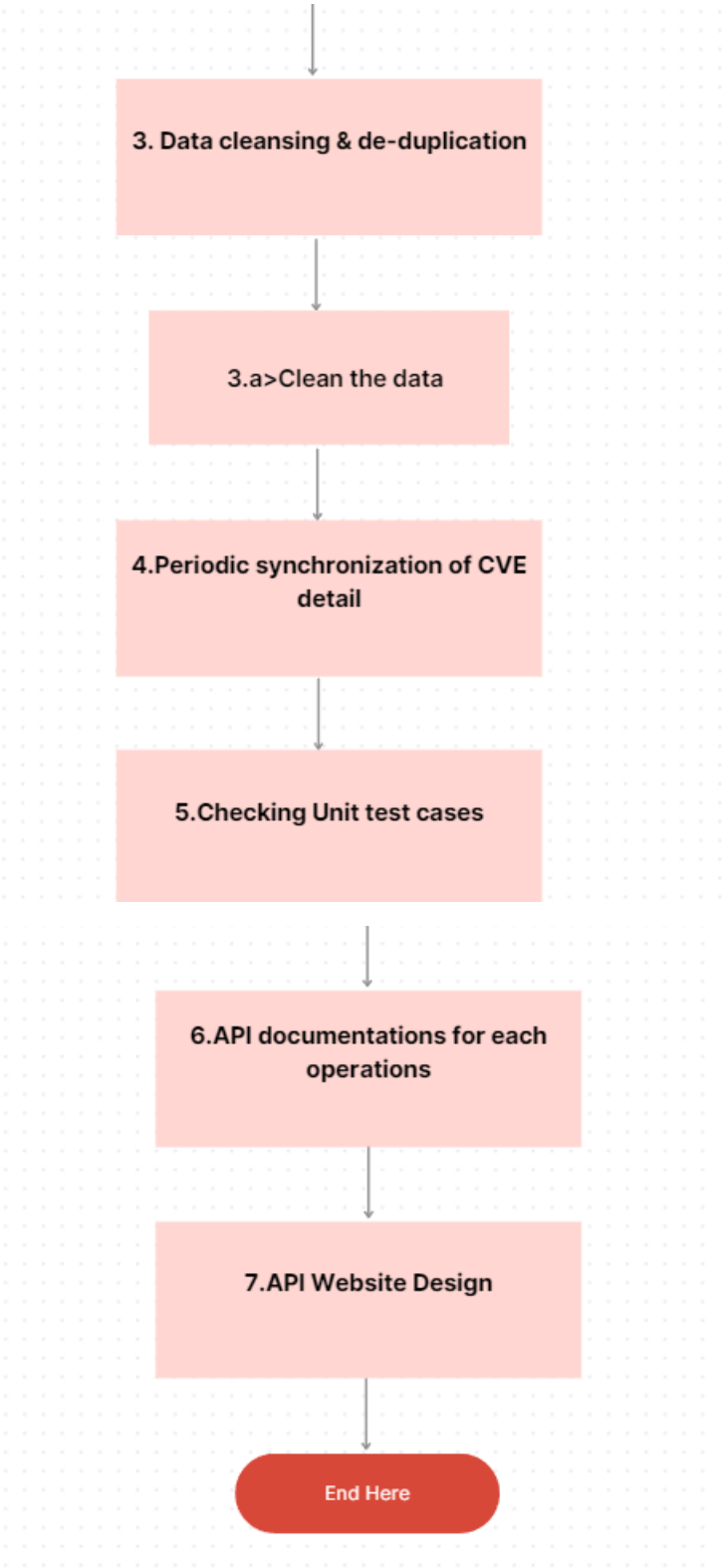
Flowchart

Used coding language-Python

Database-Sqlite3

Framework-Flask


Start
Using python with SqLite
1. Python script to fetch CVE data and store it in the database
2. CVE information from the API and store it in a database
2.a>Create a SQLite database
2.b>Fetch data from the API
2.c>Insert data into the database
2.d> Running the scripts



Steps to develop

Important installation

1.sqlite3

```
C:\Users\HP>sqlite3
SQLite version 3.45.3 2024-04-15 13:34:05 (UTF-16 console I/O)
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .help
.archive ...             Manage SQL archives
.auth ON|OFF             Show authorizer callbacks
.backup ?DB? FILE        Backup DB (default "main") to FILE
.bail on|off             Stop after hitting an error.  Default OFF
.cd DIRECTORY            Change the working directory to DIRECTORY
.changes on|off          Show number of rows changed by SQL
.check GLOB              Fail if output since .testcase does not matc
h
.clone NEWDB             Clone data into NEWDB from the existing data
base
.connection [close] [#]  Open or close an auxiliary database connecti
on
.crnl on|off             Translate \n to \r\n.  Default ON
.databases               List names and files of attached databases
.dbconfig ?op? ?val?     List or change sqlite3_db_config() options
.dbinfo ?DB?             Show status information about the database
.dump ?OBJECTS?          Render database content as SQL
```

2. Installation of Requests library, which is used for making HTTP requests in Python.

```
PS C:\Users\HP> pip install requests
Defaulting to user installation because normal site-packages is not w
riteable
Requirement already satisfied: requests in c:\users\hp\appdata\roamin
g\python\python310\site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\h
p\appdata\roaming\python\python310\site-packages (from requests) (3.1
.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\appdata\ro
aming\python\python310\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hp\appd
ata\roaming\python\python310\site-packages (from requests) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\appd
ata\roaming\python\python310\site-packages (from requests) (2022.12.7
)
```

3.Flask

```
C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project>pip install Flask
Defaulting to user installation because normal site-packages is not writeable
Collecting Flask
  Downloading flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Downloading werkzeug-3.0.3-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\hp\appdata\roaming\python\python310\site-packages (from Flask) (3.1.2)
Collecting itsdangerous>=2.1.2 (from Flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: click>=8.1.3 in c:\users\hp\appdata\roaming\python\python310\site-packages (from Flask) (8.1.3)
Requirement already satisfied: blinker>=1.6.2 in c:\users\hp\appdata\roaming\python\python310\site-packages (from Flask) (1.6.2)
```

Step 1 :- CVE Information and Database Storage (main.py):

```python
import time
import requests
import json
import sqlite3


def get_cves(start_index, results_per_page):
    url =
f"https://services.nvd.nist.gov/rest/json/cves/2.0?startIndex={start_in
dex}&resultsPerPage={results_per_page}"
    response = requests.get(url)

    if response.status_code == 200:
        return response.json()
    elif response.status_code == 403:  # Handle rate limiting
        print(f"Rate limit reached! Sleeping for 10 seconds...")
        time.sleep(10)  # Adjust delay based on NVD API rate limits
        return get_cves(start_index, results_per_page)  # Retry after
delay
    else:
        raise Exception(f"Error retrieving CVEs:
{response.status_code}")


def store_cves(cves):
    conn = sqlite3.connect("cve_data.db")
    cursor = conn.cursor()

    # Create table with appropriate data types (adjust as needed)
    cursor.execute("""CREATE TABLE IF NOT EXISTS cves (
        cve_id TEXT PRIMARY KEY,
        published_date TEXT,
        lastModifiedDate TEXT,
        description TEXT,
        cvss_v2_score REAL,
        cvss_v3_score REAL
    )""")
```

```python
    for cve in cves.get("results", []):
        data = (
            cve["cve_id"],
            cve["publishedDate"],
            cve["lastModifiedDate"],
            cve["description"]["description_data"][0]["value"],
            cve.get("impact", {}).get("baseMetricV2", {}).get(
                "cvssData", {}).get("baseScore"),
            cve.get("impact", {}).get("baseMetricV3", {}).get(
                "cvssData", {}).get("baseScore"),
        )
        cursor.execute(
            "INSERT OR IGNORE INTO cves VALUES (?, ?, ?, ?, ?, ?)",
data)

    conn.commit()
    conn.close()


def main():
    start_index = 0
    results_per_page = 50  # Adjust as needed

    while True:
        cves = get_cves(start_index, results_per_page)
        if not cves.get("totalResults", 0):
            break

        store_cves(cves)
        start_index += results_per_page


if __name__ == "__main__":
    main()
```

Step 2:-The CVE information from the API and store it in a database

Step:- 2.a> Create a SQLite database

OPEN EDITORS
Welcome
create_database....
CVE_PROJECT
create_database.py
cve_database.db

create_database.py > ...

```python
import sqlite3

# Connect to SQLite database (or create if not exists)
conn = sqlite3.connect('cve_database.db')

# Create a cursor object to execute SQL commands
cursor = conn.cursor()

# Create a table to store CVE information
cursor.execute('''CREATE TABLE IF NOT EXISTS cves (
                    cve_id TEXT PRIMARY KEY,
                    description TEXT,
                    severity TEXT,
                    last_modified TEXT
                  )''')

# Commit changes and close connection
conn.commit()
conn.close()

print("Database and table created successfully.")
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python -
top\cve_project\create_database.py"
Database and table created successfully.
```

Step-2.b>Fetch data from the API

```python
import requests

def fetch_cve_data(start_index, results_per_page):
    base_url = 'https://services.nvd.nist.gov/rest/json/cves/2.0'
    params = {
        'startIndex': start_index,
        'resultsPerPage': results_per_page
    }
    response = requests.get(base_url, params=params)
    data = response.json()
    # Check if 'result' key exists in the response
    if 'result' in data:
        return data['result']['CVE_Items']
    else:
        # If 'result' key doesn't exist, return an empty list
        return []


# Example usage
cve_data = fetch_cve_data(0, 10)
print(cve_data)
```

OPEN EDITORS
Welcome
create_database....
fetch_cve_data.py
CVE_PROJECT
create_database.py
cve_database.db
fetch_cve_data.py

fetch_cve_data.py > ...

```python
import requests

def fetch_cve_data(start_index, results_per_page):
    base_url = 'https://services.nvd.nist.gov/rest/json/cves/2.0'
    params = {
        'startIndex': start_index,
        'resultsPerPage': results_per_page
    }
    response = requests.get(base_url, params=params)
    data = response.json()
    # Check if 'result' key exists in the response
    if 'result' in data:
        return data['result']['CVE_Items']
    else:
        # If 'result' key doesn't exist, return an empty list
        return []


# Example usage
cve_data = fetch_cve_data(0, 10)
print(cve_data)
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python -u
top\cve_project\fetch_cve_data.py"
[]
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project>
```

Step-2.c> Insert data into the database

```python
import sqlite3
import requests


def fetch_cve_data(start_index, results_per_page):
    base_url = 'https://services.nvd.nist.gov/rest/json/cves/2.0'
    params = {
        'startIndex': start_index,
        'resultsPerPage': results_per_page
    }
    response = requests.get(base_url, params=params)
    data = response.json()
    return data['vulnerabilities']


def insert_cve_data_into_database(cve_data):
    conn = sqlite3.connect('cve_database.db')
    cursor = conn.cursor()

    # Create the table if it doesn't exist
    cursor.execute('''CREATE TABLE IF NOT EXISTS cves
```

```
                    (cve_id TEXT, description TEXT, severity TEXT,
last_modified TEXT)''')

    for cve_item in cve_data:
        cve_id = cve_item['cve']['id']

        # Extract description
        if 'description' in cve_item['cve']:
            if 'descriptions' in cve_item['cve']['description']:
                description =
cve_item['cve']['description']['descriptions'][0]['value']
            else:
                description = None
        else:
            description = None

        # Extract severity
        if 'impact' in cve_item and 'baseMetricV3' in
cve_item['impact']:
            severity =
cve_item['impact']['baseMetricV3']['cvssV3']['baseSeverity']
        else:
            severity = None

        # Extract last modified date
        if 'lastModifiedDateStr' in cve_item:
            last_modified = cve_item['lastModifiedDateStr']
        else:
            last_modified = None

        cursor.execute('''INSERT OR IGNORE INTO cves (cve_id,
description, severity, last_modified)
                          VALUES (?, ?, ?, ?)''', (cve_id, description,
severity, last_modified))

    conn.commit()
    conn.close()


# Example usage
cve_data = fetch_cve_data(0, 10)
insert_cve_data_into_database(cve_data)
```

```
OPEN EDITORS
  create_database....
  fetch_cve_data.py
    cve_database.db
CVE_PROJECT
  create_database.py
  cve_database.db
  fetch_cve_data.py

fetch_cve_data.py > ...
def insert_cve_data_into_database(cve_data):

        cursor.execute('''INSERT OR IGNORE INTO cves (cve_id, description, sever
                          VALUES (?, ?, ?, ?)''', (cve_id, description, severity

    conn.commit()
    conn.close()


# Example usage
cve_data = fetch_cve_data(0, 10)
insert_cve_data_into_database(cve_data)

PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL

PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python -u "c:\Us
top\cve_project\fetch_cve_data.py"
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project>
```

Step-2.d> **Running the scripts:**

python create_database.py

python fetch_cve_data.py

```
PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL                                  Code

PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python -u "c:\Users\HP\OneDrive - Amrita Vi
top\cve_project\fetch_cve_data.py"
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python create_database.py
Database and table created successfully.
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python fetch_cve_data.py
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project>
```

Step 3: Data cleansing & de-duplication

Step 3.a>Clean the data: Perform data cleansing operations such as removing duplicates, handling missing values, etc.

```
import sqlite3

conn = sqlite3.connect('cve_database.db')
cursor = conn.cursor()

# Identify duplicates
cursor.execute('''SELECT cve_id, COUNT(*)
                  FROM cves
                  GROUP BY cve_id
                  HAVING COUNT(*) > 1''')
duplicates = cursor.fetchall()

# Remove duplicates
for cve_id, count in duplicates:
```

```
    cursor.execute('''DELETE FROM cves
                      WHERE rowid NOT IN (SELECT MIN(rowid)
                                          FROM cves
                                          WHERE cve_id = ?
                                          GROUP BY cve_id)''', (cve_id,))

# Identify missing values
cursor.execute('''SELECT *
                  FROM cves
                  WHERE severity IS NULL OR description IS NULL OR last_modified IS NULL''')
missing_values = cursor.fetchall()

# Handle missing values
# 1.Deleting rows with missing values
for row in missing_values:
    cve_id = row[0]
    cursor.execute('''DELETE FROM cves WHERE cve_id = ?''', (cve_id,))

# 2.Updating missing values
for row in missing_values:
    cve_id = row[0]
    cursor.execute(
        '''UPDATE cves SET severity = ? WHERE cve_id = ?''', ('Unknown', cve_id))

# 3.Imputation:-Impute missing values with the mean, median, or mode of the respective column.
for row in missing_values:
    cve_id = row[0]
    # Replace missing severity with the mean severity
    cursor.execute(
        '''UPDATE cves SET severity = (SELECT AVG(severity) FROM cves WHERE severity IS NOT NULL) WHERE cve_id = ?''', (cve_id,))

```

```
EXPLORER
OPEN EDITORS
    create_database....
    fetch_cve_data.py
    data_cleansing.py
    cve_database.db
CVE_PROJECT
    create_database.py
    cve_database.db
    data_cleansing.py
    fetch_cve_data.py
    tempCodeRunnerFile....

create_database.py   fetch_cve_data.py   data_cleansing.py   cve_database.db
data_cleansing.py > ...

for row in missing_values:
    cve_id = row[0]
    cursor.execute(
        '''UPDATE cves SET severity = ? WHERE cve_id = ?''', ('Unknown', cve_id))

# 3.Imputation:-Impute missing values with the mean, median, or mode of the respective column.
for row in missing_values:
    cve_id = row[0]
    # Replace missing severity with the mean severity
    cursor.execute(
        '''UPDATE cves SET severity = (SELECT AVG(severity) FROM cves WHERE severity IS NOT NULL


PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL

PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python -u "c:\Users\HP\OneDrive -
top\cve_project\tempCodeRunnerFile.py"
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> python data_cleansing.py
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project>
```

Step4:- Periodic synchronization of CVE details into the database using a scheduler library like schedule in Python, follow these steps:

4.a>Install Schedule Library: First, install the schedule library if you haven't already. You can install it via pip:

1. **Write Synchronization Function: Write a function to synchronize CVE details into the database. This function should fetch data from the NVD CVE API and update the database accordingly.**
2. **Define Schedule: Define a schedule for how often you want the synchronization to occur. For example, if you want it to run daily at midnight, you can define it accordingly.**
3. **Start Scheduler: Start the scheduler and let it run continuously in the background.**

```
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project> pip install schedule
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: schedule in c:\users\hp\appdata\roaming\python\python310\site-packages (1.2.1)
PS C:\Users\HP\OneDrive - Amrita Vishwa Vidyapeetham\Desktop\cve_project>

Kubernetes   Open In Browser   Ln 44, Col 17   Spaces: 4   UTF-8   CRLF   Python   3.11.9 64-bit (Microsoft Store)   Go Live   Prettier
```

```python
import time
import json
import requests
import sqlite3


def synchronize_cve_data():
    # Make API call to fetch CVE data
```

```
    response =
requests.get('https://services.nvd.nist.gov/rest/json/cves/2.0')
    cve_data = response.json()

    # Process and update database
    conn = sqlite3.connect(
        r'C:\Users\HP\OneDrive - Amrita Vishwa
Vidyapeetham\Desktop\cve_project\cve_database.db')
    cursor = conn.cursor()

    # Insert or update CVE details in the database
    for cve in cve_data['CVE_Items']:
        cve_id = cve['cve']['CVE_data_meta']['ID']
        cve_description =
cve['cve']['description']['description_data'][0]['value']
        cve_published_date = cve['publishedDate']
        cve_last_modified_date = cve['lastModifiedDate']
        cve_score = None
        if 'baseMetricV2' in cve['impact']:
            cve_score =
cve['impact']['baseMetricV2']['cvssV2']['baseScore']
        elif 'baseMetricV3' in cve['impact']:
            cve_score =
cve['impact']['baseMetricV3']['cvssV3']['baseScore']

        # Insert or replace CVE details into the database
        cursor.execute("INSERT OR REPLACE INTO cve_table (cve_id,
description, published_date, last_modified_date, score) VALUES (?, ?,
?, ?, ?)",
                       (cve_id, cve_description, cve_published_date,
cve_last_modified_date, cve_score))

    conn.commit()
    conn.close()


def run_scheduler():
    while True:
        # Get the current time
        current_time = time.localtime()

        # Check if it's midnight (00:00)
        if current_time.tm_hour == 0 and current_time.tm_min == 0:
            # Call the synchronize_cve_data function
            synchronize_cve_data()

        # Sleep for 1 minute
        time.sleep(60)
```

```
if __name__ == "__main__":
    run_scheduler()
```

Step 5:-Checking Unit test cases

```
import unittest
from main import get_cves


class TestGetCves(unittest.TestCase):

    def test_successful_retrieval(self):
        # Adjust start_index and results_per_page as needed
        cves = get_cves(0, 1)
        self.assertIsInstance(cves, dict)
        self.assertIn("totalResults", cves)

    # Add more test cases for different scenarios (e.g., error
handling)


if __name__ == "__main__":
    unittest.main()
```

```
import unittest
from main import get_cves


class TestGetCves(unittest.TestCase):

    def test_successful_retrieval(self):
        # Adjust start_index and results_per_page as needed
        cves = get_cves(0, 1)
        self.assertIsInstance(cves, dict)
        self.assertIn("totalResults", cves)

    def test_empty_results(self):
        # Simulate a scenario with potentially non-empty results
        cves = get_cves(10000, 1)  # Adjust start_index as needed
        self.assertIsInstance(cves, dict)
        self.assertGreater(cves.get("totalResults", 0),
```

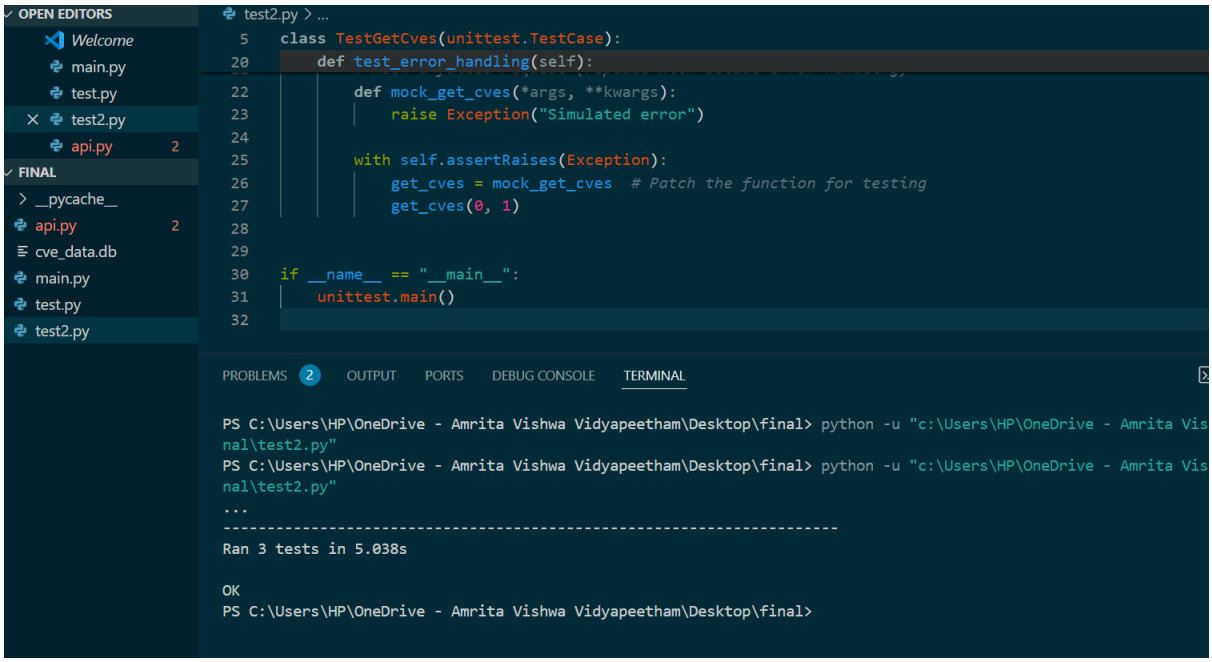
```
                        0)  # Check for non-zero

    def test_error_handling(self):
        # Mock a failed request (replace with actual error handling)
        def mock_get_cves(*args, **kwargs):
            raise Exception("Simulated error")

        with self.assertRaises(Exception):
            get_cves = mock_get_cves  # Patch the function for testing
            get_cves(0, 1)


if __name__ == "__main__":
    unittest.main()
```

- Output of passed unit test cases



Step 6:-API documentations for each operations(api.py)

```
from flask import Flask, jsonify, request
import sqlite3

app = Flask(__name__)


def get_cve_by_id(cve_id):
    conn = sqlite3.connect("cve_data.db")
```

```
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM cves WHERE cve_id = ?", (cve_id,))
    row = cursor.fetchone()
    conn.close()

    if row:
        return jsonify(dict(zip(cursor.description[1:], row)))
    else:
        return jsonify({"error": "CVE not found"}), 404


def get_cves_by_year(year):
    conn = sqlite3.connect("cve_data.db")
    cursor = conn.cursor()

    cursor.execute(
        "SELECT * FROM cves WHERE publishedDate LIKE ?", (f"{year}%",))
    rows = cursor.fetchall()
    conn.close()

    return jsonify([dict(zip(cursor.description[1:], row)) for row in
rows])


def get_cves_by_score_range(min_score, max_score):
    conn = sqlite3.connect("cve_data.db")
    cursor = conn.cursor()

    if min_score is None and max_score is None:
        cursor.execute("SELECT * FROM cves")
    else:
        query = "SELECT * FROM cves WHERE cvss_v2_score BETWEEN ? AND
?"
        cursor
```

Step 7:-Website Design

CVE Data (Sample)

Total Records: 4

CVE ID	Identifier	Published Date	Last Modified Date	Status
CVE-1999-0334	nvd@nist.gov	2008-09-05	2009-10-21	Analyzed
CVE-1999-0334	cve@mitre.org	2009-09-05	2012-08-04	Modified
CVE-1999-0334	nvd@nist.gov	2011-02-08	2012-23-03	Rejected
CVE-1999-0334	nvd@nist.gov	2010-09-05	2013-10-15	Analyzed

API request failed with status:Failed

Index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CVE Data</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <h1>CVE Data</h1>
    <div id="total-records">
        <p>Total Records: <span id="total-count"></span></p>
    </div>
    <table id="cve-table">
        <thead>
            <tr>
                <th>CVE ID</th>
                <th>Identifier</th>
                <th>Published Date</th>
                <th>Last Modified Date</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody id="cve-data"></tbody>
    </table>
    <script src="script.js"></script>
</body>

</html>
```

Script.js

```javascript
const cveTable = document.getElementById('cve-table');
const cveDataContainer = document.getElementById('cve-data');
const totalCountSpan = document.getElementById('total-count');
const loadingIndicator = document.createElement('p'); // New element for
loading indicator

function updateTable(data, totalCount) {
    cveDataContainer.innerHTML = ''; // Clear previous content
    totalCountSpan.textContent = totalCount;

    data.forEach(cve => {
```

```
        const row = document.createElement('tr');

        const cveIdCell = document.createElement('td');
        cveIdCell.textContent = cve.cve_id;

        const identifierCell = document.createElement('td');
        identifierCell.textContent = cve.cve_data_meta ? .id || "-"; // Adjust
based on API response structure

        const publishedDateCell = document.createElement('td');
        publishedDateCell.textContent = cve.publishedDate;

        const lastModifiedDateCell = document.createElement('td');
        lastModifiedDateCell.textContent = cve.lastModifiedDate;

        const statusCell = document.createElement('td');
        statusCell.textContent = cve.cve_data_meta ? .status || "-"; // Adjust
based on API response structure

        row.appendChild(cveIdCell);
        row.appendChild(identifierCell);
        row.appendChild(publishedDateCell);
        row.appendChild(lastModifiedDateCell);
        row.appendChild(statusCell);

        cveDataContainer.appendChild(row);
    });
}

// Replace with the actual NVD API endpoint URL
const apiUrl = 'https://services.nvd.nist.gov/rest/json/cves/2.0';

function fetchData() {
    loadingIndicator.textContent = "Loading CVE data...";
    cveDataContainer.appendChild(loadingIndicator);

    fetch(apiUrl)
        .then(response => {
            loadingIndicator.remove(); // Remove loading indicator on success

            if (!response.ok) {
                throw new Error(`API request failed with status:
${response.status}`);
            }
            return response.json();
        })
        .then(data => {
            // Assuming "totalResults" gives the total count
```

```
            updateTable(data.results, data.totalResults);
        })
        .catch(error => {
            console.error('Error fetching CVEs:', error);

            // More informative error handling
            if (error.message.includes('404')) {
                cveDataContainer.innerHTML = '<p>API endpoint not found.</p>';
            } else if (error.message.includes('429')) {
                cveDataContainer.innerHTML = '<p>API rate limit exceeded.
Please try again later.</p>';
            } else {
                cveDataContainer.innerHTML = '<p>Error retrieving CVE
data.</p>';
            }
        });
}

fetchData();
```