**Documentation for CVE Data Flask Application**

**Overview**

This documentation provides an overview and detailed explanation of a Flask web application designed to fetch and display CVE (Common Vulnerabilities and Exposures) data from the NVD (National Vulnerability Database) API. The application supports pagination and allows users to select the number of entries displayed per page.

**Application Structure**

The application consists of a Python script (app.py) that sets up the Flask server and an HTML template (indexx.html) for rendering the data.

**app.py**

This is the main file of the Flask application.

* Flask, render\_template, request: Flask modules for creating the web application, rendering templates, and handling HTTP requests.
* requests: For making HTTP requests to external APIs.
* json: For parsing JSON responses.
* datetime: For date and time manipulation.

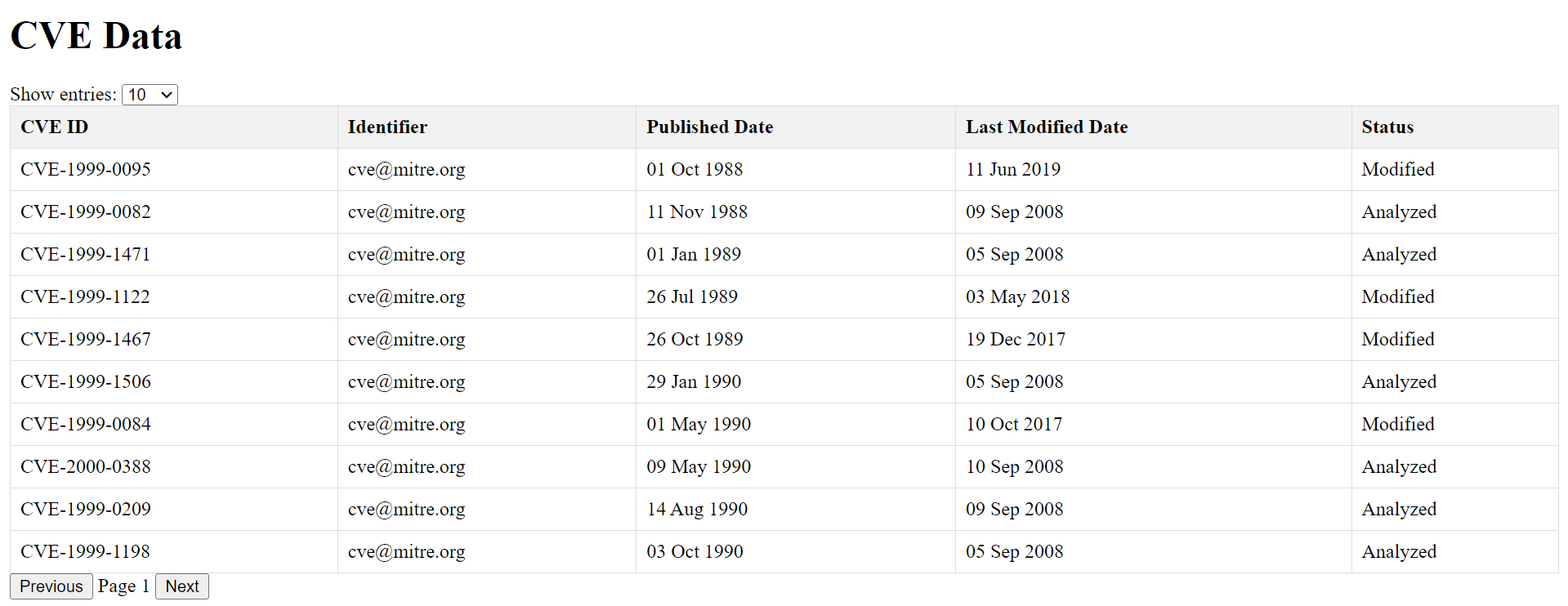
**Route Definition**

Defines the / route which handles the main page. Retrieves page and per\_page parameters from the URL query string. Defaults are set to 1 and 10, respectively. Calculates start\_index and end\_index for pagination. Makes a GET request to the NVD API to fetch CVE data. Checks if the response status code is 200 (OK). Parses the JSON response and reformats the published and lastModified dates. Paginates the data based on the page and per\_page parameters.

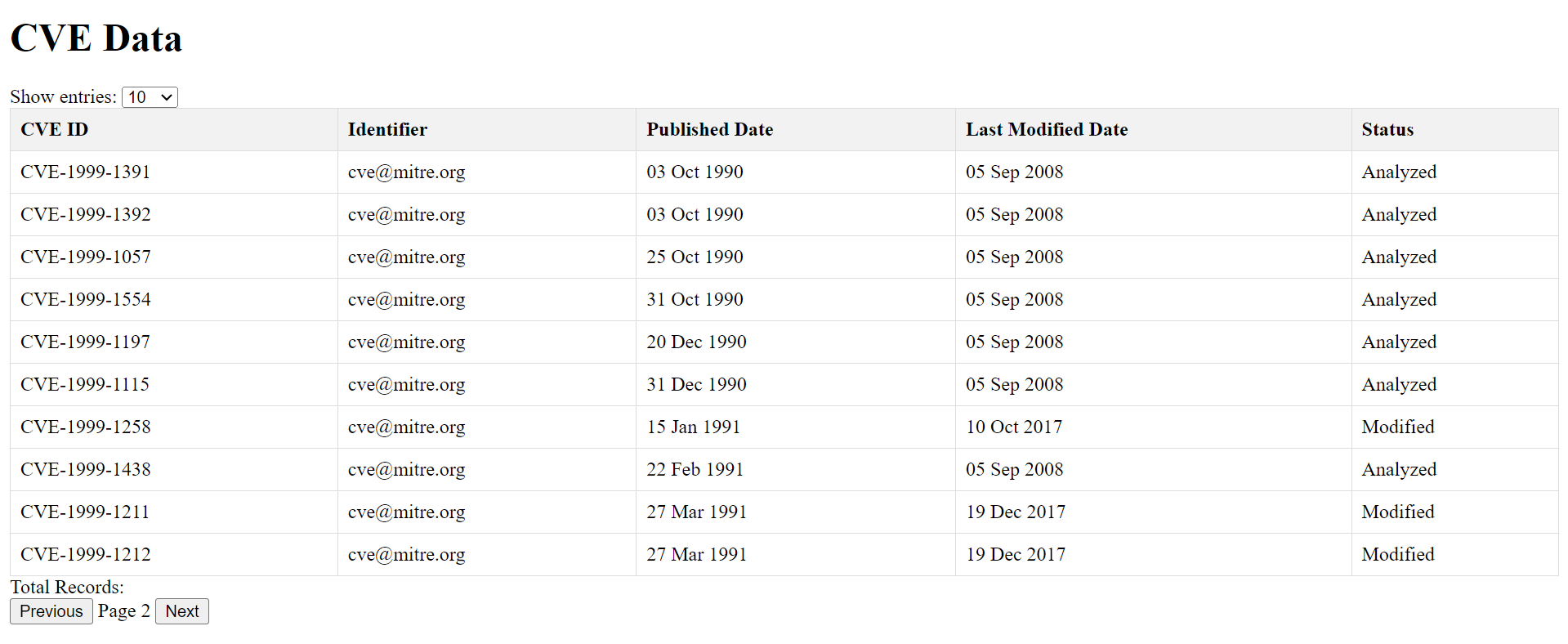
**indexx.html**

This is the HTML template for displaying the CVE data.

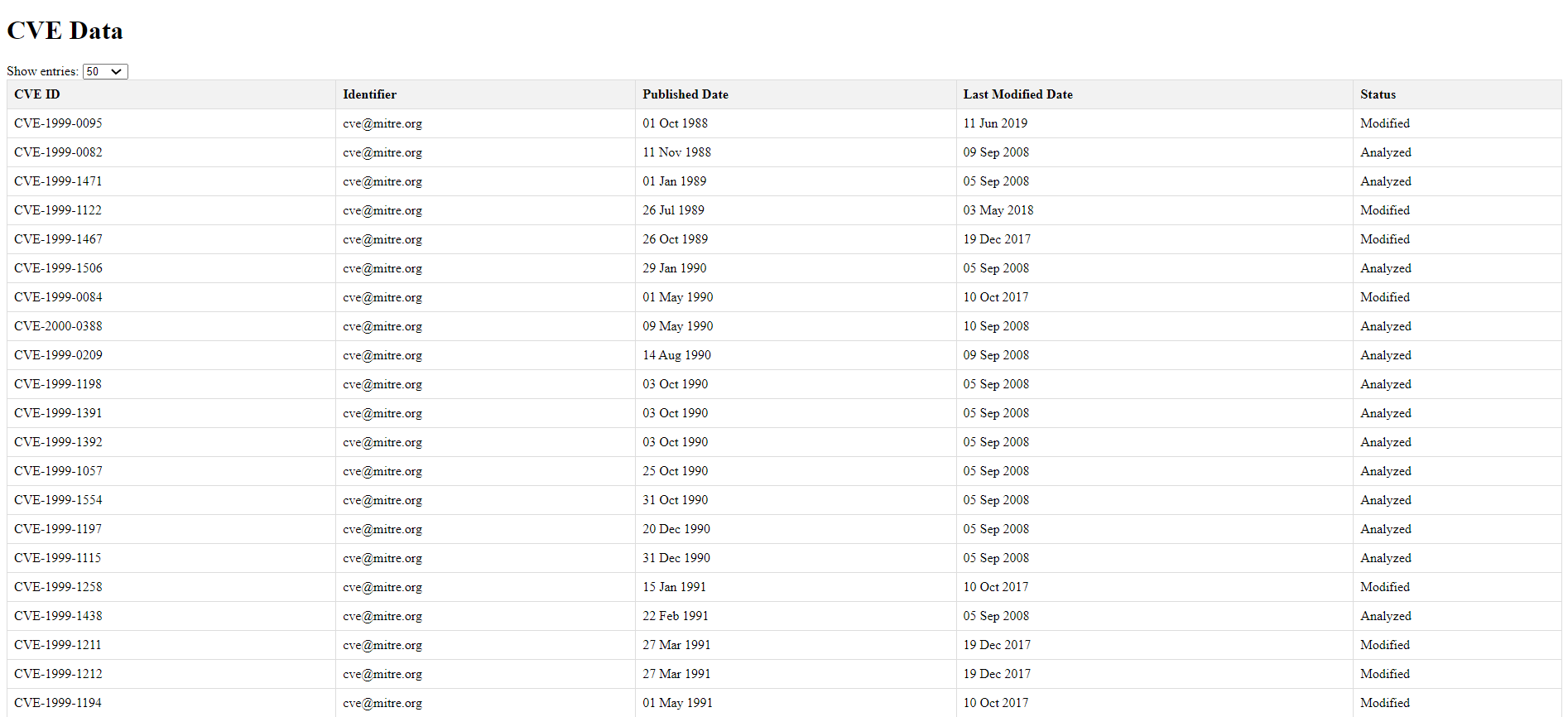
The HTML template defines the document type and includes basic meta information in the DOCTYPE and HTML Head sections. CSS styles are used to make the table visually appealing. The body of the document includes several key components: a header that displays the title "CVE Data"; an entries selection dropdown to select the number of entries per page, which calls the `changePerPage()` function when changed; a table that displays the CVE data with columns for CVE ID, Identifier, Published Date, Last Modified Date, and Status, using Jinja2 templating syntax to loop through the data passed from the Flask app; and pagination controls with "Previous" and "Next" buttons to navigate between pages, calling the `previousPage()` and `nextPage()` functions respectively. The JavaScript section includes variables `currentPage` and `perPage` to track the current page and entries per page, the `changePerPage()` function to reload the page with the selected `per\_page` value, and the `previousPage()` and `nextPage()` functions to navigate between pages by adjusting the `page` parameter in the URL.



The above image shows the first page.



When next is clicked , it goes to next page.



The above is the figure that displays 50 records when the show entry is set to 50. A total of 50 records are displayed in the table.

EXPLANATION:

The provided Flask application is a web-based tool for displaying and exploring Common Vulnerabilities and Exposures (CVE) data. It consists of two main routes: one for displaying a paginated list of CVE entries and another for showing detailed information about a specific CVE entry.

In the index route (/), the application sends a GET request to the NVD API to fetch the list of CVE entries. It then processes the data, including formatting the date fields, and paginates the results based on the requested page and number of entries per page. The paginated data is then passed to the indexx.html template for rendering, along with pagination information.

The indexx.html template contains a table displaying the CVE entries, with columns for the CVE ID, identifier, published date, last modified date, and status. It also includes controls for changing the number of entries per page and navigating between pages.

In the cve\_detail route (/cves/<cve\_id>), the application fetches detailed information about a specific CVE entry using the given CVE ID. It then renders the detail.html template, which displays the CVE ID, description, CVSS V2 metrics, scores, and CPE information in a structured format.