```

//**************************Homework 8************************************ */
// Function to handle the AJAX process for loading CD data
// Function to load CD data using the Fetch API - Failed to work!
/*function loadCDData() {
    // Making a GET request to the specified URL
    fetch("AjaxAssignment/cd_catalog.xml")
    .then(response => {
        // Checking if the response is successful
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        // Parsing the response as text
        return response.text();
    })
    .then(str => {
        // Parsing the string response to an XML Document
        const parser = new DOMParser();
        const xmlDoc = parser.parseFromString(str, "application/xml");
        // Updating the table with data from the XML
        updateTable(xmlDoc);
    })
    .catch(error => {
        console.error('There has been a problem with your fetch operation:', error);

        // Selecting the element where the error message will be displayed
        const errorMessageElement = document.getElementById("demo");

        // Setting the inner HTML to the error message
        errorMessageElement.innerHTML = 'Failed to load CD data.';

        // Changing the style of the element to have white text
        errorMessageElement.style.color = 'white';
        // Additional styling can be added here if needed
    });
}

Function to update the table with CD data
function updateTable(xmlDoc) {
    // Getting all CD elements from the XML document
    const cds = xmlDoc.getElementsByTagName("CD");
    // Starting the table HTML with headers
    let table = "<tr><th>Artist</th><th>Year</th></tr>";

    // Looping through each CD element and adding its data to the table
    for (let cd of cds) {
        // Getting the artist and year from the CD element
        const artist = cd.getElementsByTagName("ARTIST")[0].textContent;
        const year = cd.getElementsByTagName("YEAR")[0].textContent;
        // Appending a table row for each CD
        table += `<tr><td>${artist}</td><td>${year}</td></tr>`;
    }

    // Setting the inner HTML of the table element to the constructed table
    document.getElementById("demo").innerHTML = table;
}
*/

//Following Code Copied from given ajaxassignment code  making changes in year from title
function loadDoc() {
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
    myFunction(this);
    }
    xhttp.open("GET", "AjaxAssignment/cd_catalog.xml");
    xhttp.send();
}
function myFunction(xml) {
    const xmlDoc = xml.responseXML;
```

```
    const x = xmlDoc.getElementsByTagName("CD");
    let table="<tr><th>Artist</th><th>Year</th></tr>";
    for (let i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "</td><td>" +
    x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue +
    "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}


// Function to create and display an ordered list of process steps
document.getElementById('showProcess').addEventListener('click', function() {
    // Define the process steps as an array of strings
    const processSteps = [
        "Making the Order (Sending the AJAX Request): <br>You decide you're hungry for some data. This is like picking up your phone and dialing the pizza place to order pizza.",
        "Waiting for the Pizza (Waiting for the Response): <br>After placing your order, you go back to your activities. Meanwhile, the pizza place prepares your order.",
        "Pizza is Ready! (Receiving the Response): <br>The pizza place lets you know that your pizza is ready. You check to make sure it's the right order.",
        "Unboxing the Pizza (Processing the Data): <br>You open the pizza box and check out each slice. This is like processing the XML data.",
        "Enjoying Your Meal (Displaying the Data): <br>You arrange the pizza slices on plates, similar to how the data is arranged in a readable format on the webpage.",
        "Sharing the Pizza (Updating the Webpage): <br>You invite everyone to eat. The webpage is updated with the new data.",
        "Handling Pizza Mishaps (Error Handling): <br>If something goes wrong, like receiving the wrong pizza, you know how to handle it. This is like error checking in your code."
    ];

    // Create an ordered list element
    const processList = document.createElement('ol');

    // For each step, create a list item, set its innerHTML, and append it to the list
    processSteps.forEach(step => {
        const listItem = document.createElement('li');
        listItem.innerHTML = step;
        processList.appendChild(listItem);
    });

    // Get the container element and set its content to the process list
    const container = document.getElementById('processContainer');
    container.innerHTML = ''; // Clear any previous content
    container.appendChild(processList); // Add the process list to the container
});


//************************************REST API**************************************
*/

// Adding an event listener to the button with the ID 'showReport'
document.getElementById('showReport').addEventListener('click', function() {
    // Defining the content of the report as a string literal
    const reportContent = `
    <p><strong>Enhancing Heart Pizza's Online System with CalorieNinjas API</p>
    <p>Purpose:</strong><br>We are poised to take Heart Pizza's online ordering system to the next level by integrating the CalorieNinjas API. This tool provides essential nutritional information, including calorie counts, for all our pizza ingredients, enriching our customers' experience with valuable health insights.</p>
    <p><strong>JSON Response Format:</strong><br>The API offers data in JSON format, ensuring compatibility with modern web technologies and facilitating easy data handling in our web application.</p>
    <p><strong>API Key Requirement:</strong><br>Access to the API is secured with an API
```

```
        key, providing a layer of protection and ensuring that our data transactions are
        safe and authenticated.</p>
        <p><strong>Cost-Effectiveness:</strong><br>The API's flexible pricing structure
        allows us to start with basic, free features, with the option to scale up as our
        needs grow and evolve.</p>
        <p><strong>Comprehensive Documentation:</strong><br>Detailed <a
        href='https://calorieninjas.com' target='_blank'>documentation</a> is available to
        guide our developers through seamless integration and effective utilization of the
        API.</p>
        <p>By adopting the CalorieNinjas API, we are not just improving our service
        offerings but also moving towards a more informed and health-conscious business
        model.</p>
    `;

        // Inserting the report content into the element with the ID 'reportContainer'
        document.getElementById('reportContainer').innerHTML = reportContent;
    });

    // Adding an event listener to the button with the ID 'calculateCalories'
    document.getElementById('calculateCalories').addEventListener('click', function() {
        // Retrieving the value of the input field with the ID 'ingredients'
        const ingredients = document.getElementById('ingredients').value;

        // Fetching data from the CalorieNinjas API using the ingredients provided by the
        user

        fetch(`https://api.calorieninjas.com/v1/nutrition?query=${encodeURIComponent(ingredie
        nts)}`, {
            headers: { 'X-Api-Key': '<hidden API Key>' } // CalorieNinjas API key
        })
        .then(response => response.json()) // Parsing the response to JSON format
        .then(data => {
            // Calculating the total calories by summing up the calories of each ingredient
            const calories = data.items.reduce((total, item) => total + item.calories, 0);

            // Displaying the total calories in the element with the ID 'calorieResult',
            dynamically showing the pizza type
            document.getElementById('calorieResult').innerText = `Total Calories for
            ${ingredients}: ${calories.toFixed(2)}`;
        })
        .catch(error => {
            // Logging any errors to the console and displaying an error message to the user
            console.error('Error:', error);
            document.getElementById('calorieResult').innerText = 'Failed to calculate
            calories.';
        });
    });

```