1. Continuous Mean Value Calculation
Approach – 1

Date: 17/01/2024

Explanation

Index.html

The provided HTML code creates a web page for a "Continuous Mean Calculator." The page features an input field where users can enter numeric values, along with two buttons: "Calculate Mean" and "End Input." The layout is aesthetically styled using inline CSS, ensuring a clean and readable design. The JavaScript section of the code is responsible for handling user interactions and asynchronous communication with the server.

The `calculateMean()` function is triggered by the "Calculate Mean" button and asynchronously sends a POST request to the server (`/calculate`). It retrieves the user-entered value, updates the result display area with the calculated mean received from the server, and logs any errors to the console.

The `endInput()` function, activated by the "End Input" button, resets the display to the initial message prompting users to enter a value. Simultaneously, it sends a POST request to the server (`/reset`) to reset ongoing calculations and displays a response alert. Similar to `calculateMean()`, error handling is in place to log any issues.

The user interface encourages a straightforward experience: users input values, calculate the mean asynchronously with the click of a button, and have the option to reset calculations. The use of asynchronous communication with the server enhances responsiveness, providing an interactive and user-friendly environment for continuous mean calculation.

Steps

1. **HTML Structure:**
   - The HTML document starts with the usual structure, specifying the document type, language, and containing a `<head>` and `<body>` section.
   - The `<head>` section includes meta tags for character set and viewport settings. It also sets the title of the page to "Continuous Mean Calculator."
   - Inside the `<head>` section, there's an inline CSS style block defining styles for the body, input, button, and result display area.

2. **Body Content:**
   - The `<body>` section contains a heading (`<h1>`) indicating the title of the calculator: "Continuous Mean Calculator."
   - Following the heading, there is an input field (`<input>`) labeled "Enter a value" and two buttons (`<button>`):
   - "Calculate Mean" button: Triggers the `calculateMean()` JavaScript function.
   - "End Input" button: Triggers the `endInput()` JavaScript function.
   - A `<div>` element with the id "result" is present to display the calculated mean or instructions to enter a value.

3. **Inline CSS Styling:**
   - The inline CSS styling provides a clean and simple layout for the webpage. It adjusts margins, padding, and text alignment for a visually appealing user interface.

4. **JavaScript Code:**
   - The JavaScript code is embedded within a `<script>` block.
   - The `calculateMean()` function is an asynchronous function that retrieves the input value from the user, sends a POST request to the server using the Fetch API, and updates the result display area with the calculated mean received from the server. Any errors during this process are caught and logged to the console.
   - The `endInput()` function is also asynchronous and is triggered by the "End Input" button. It resets the result display to the initial message, indicating that the user should enter a value to calculate the mean. Additionally, it sends a POST request to the server to reset calculations, displaying an alert with the server's response. Like `calculateMean()`, errors are caught and logged.

5. **Server Communication:**
   - The Fetch API is used for asynchronous communication with the server. Two endpoints are utilized:
   - `/calculate`: For calculating the mean based on the user-provided value.
   - `/reset`: For resetting the calculation, triggered by the "End Input" button.

6. **User Interaction:**
   - Users can input values in the provided text field and trigger calculations using the "Calculate Mean" button. The "End Input" button resets the ongoing calculation, providing a user-friendly interface for continuous mean calculation.

App.py

The `app.py` code represents a Flask web application designed to serve as the backend for a Continuous Mean Calculator. The code utilizes the Flask framework to handle HTTP requests and responses. It includes global variables, `total_sum` and `count`, which maintain the running sum and count of entered values. The `is_valid_float` function checks whether an input value is a valid floating-point number by looking for the presence of a decimal point.

The application defines three routes: the main page ("/"), a route for calculating the mean ("/calculate"), and a route to reset calculations ("/reset"). The main page renders an HTML template, and the "/calculate" route handles POST requests with JSON payloads containing numerical values. It calculates the mean, updates global variables, and returns the current mean. The "/reset" route resets the global variables to zero, providing a way to restart the calculation process.

The script includes an execution block to run the Flask application when executed directly, and it runs in debug mode to facilitate development and debugging. It's important to note that this backend code is intended to work in conjunction with a corresponding HTML template (such as "index.html") that handles user interaction and asynchronous communication with the backend using the Fetch API. The overall structure allows for a dynamic and interactive continuous mean calculation experience on the frontend.

Index.html code

```html
<!-- index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Continuous Mean Calculator</title>
    <!-- Inline CSS for styling -->
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 50px;
        }

        footer {
            position: fixed;
            bottom: 0;
            left: 0;
            width: 100%;
            padding: 10px;
            background-color: rgba(255, 255, 255, 0.8);
            z-index: 1000;
            display: flex;
            justify-content: center; /* Center the content horizontally */
            align-items: center; /* Center the content vertically */
        }

        footer img {
            height: 100px; /* Adjust the height as needed */
            width: 900px;
            margin-right: 20px;
        }

        input {
            padding: 8px;
            margin-right: 10px;
        }

        button {
            padding: 8px;
        }

        #result {
            margin-top: 20px;
```

```html
        }
    </style>
</head>
<body>

    <h1>Continuous Mean Calculator</h1>

    <!-- Input field and two buttons for user interaction -->
    <label for="value">Enter a value:</label>
    <input type="text" id="value" placeholder="Enter a number">
    <button onclick="calculateMean()">Calculate Mean</button>
    <button onclick="endInput()">End Input</button>

    <!-- Display area for the result -->
    <div id="result">Enter a value to calculate the mean.</div>

    <footer>
        <img src="https://www.itgroup-
europe.com/fileadmin/_processed_/a/a/csm_logo_cab83c2c3a.jpg" alt="Footer
Logo">
    </footer>

<!-- JavaScript code using Fetch API for asynchronous communication -->
    <script>
        // Asynchronous function to calculate mean
        async function calculateMean() {
            // Get the input value from the user
            var inputValue = document.getElementById('value').value;

            try {
                // Send a POST request to the server using Fetch API
                var response = await fetch('/calculate', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ value: inputValue })
                });

                // Get the result from the response and display it
                var result = await response.text();
                document.getElementById('result').innerHTML = result;
            } catch (error) {
                // Handle errors, if any
                console.error('Error:', error);
            }
        }
```

```
        // Function to end input (triggered by the "End Input" button)
        async function endInput() {
            // Clear and reset the current mean value to 0
            document.getElementById('result').innerHTML = "Enter a value to
calculate the mean.";

            // Send a request to the server to reset calculations
            try {
                var response = await fetch('/reset', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    }
                });

                // Get the result from the response and display it
                var result = await response.text();
                alert(result);
            } catch (error) {
                // Handle errors, if any
                console.error('Error:', error);
            }
        }
    </script>



</body>
</html>
```

App.py code

```python
# app.py (Python backend)
from flask import Flask, request, render_template

app = Flask(__name__)

# Variables to store the sum and count
total_sum = 0
count = 0

# Function to check if the input value is a valid floating-point number
def is_valid_float(value):
    return '.' in value

# Route for the main page
@app.route("/")
def index():
    return render_template("index.html")

# Route for calculating the mean
@app.route("/calculate", methods=["POST"])
def calculate_mean():
    global total_sum, count

    try:
        input_value = request.json["value"]

        # Check if the entered value is a valid floating-point number
        if not is_valid_float(input_value):
            raise ValueError("Please enter a valid floating-point number.")

        new_value = float(input_value)

    except ValueError:
        return "Invalid input. Please enter a valid floating-point number."

    # Update variables
    total_sum += new_value
    count += 1

    # Calculate the current mean
    current_mean = total_sum / count

    return f"Current mean: {current_mean}"

# Route to reset calculations (triggered by the "End Input" button)
```

```python
@app.route("/reset", methods=["POST"])
def reset_calculations():
    global total_sum, count

    # Reset total sum and count to 0
    total_sum = 0
    count = 0

    return "Calculations reset. You can start again."

if __name__ == "__main__":
    app.run(debug=True)
```