Sensor Data Logger (app.js)

```
document.addEventListener('DOMContentLoaded', () => {
    const form = document.getElementById('sensor-form');
    const sortBy = document.getElementById('sort-by');

    // Fetch and display all sensors
    function fetchAndDisplaySensors(sort = 'timestamp') {
      fetch('http://localhost:3000/sensorReadings')
        .then(response => response.json())
        .then(sensors => {
          // Sort sensors based on the selected criteria
          sensors.sort((a, b) => {
            if (sort === 'timestamp') {
              return new Date(b.timestamp) - new Date(a.timestamp);
            } else if (sort === 'value') {
              return b.value - a.value;
            }
          });
          displaySensors(sensors);
        });
    }

    // Function to display sensors
    function displaySensors(sensors) {
      const sensorList = document.getElementById('sensor-list');
      sensorList.innerHTML = '';  // Clear list before rendering new data

      sensors.forEach(sensor => {
        const sensorDiv = document.createElement('div');
        sensorDiv.innerHTML = `
          <p>${sensor.type}: ${sensor.value} (${sensor.timestamp})</p>
          <button class="edit-btn" data-id="${sensor.id}">Edit</button>
          <button class="delete-btn" data-id="${sensor.id}">Delete</button>
        `;
        sensorList.appendChild(sensorDiv);
      });
    }

    // Handle edit button click
    document.getElementById('sensor-list').addEventListener('click',
(event) => {
      if (event.target.classList.contains('edit-btn')) {
        const sensorId = event.target.getAttribute('data-id');

        // Fetch specific sensor data
        fetch(`http://localhost:3000/sensorReadings/${sensorId}`)
          .then(response => response.json())
          .then(sensor => {
            // Prefill form with sensor data
```

```
            document.getElementById('sensor-type').value = sensor.type;
            document.getElementById('sensor-value').value = sensor.value;
            // Set the sensor id to form's data-edit-id
            form.setAttribute('data-edit-id', sensor.id);
          });
      }

      // Handle delete button click
      if (event.target.classList.contains('delete-btn')) {
        const sensorId = event.target.getAttribute('data-id');

        // Send DELETE request to delete sensor
        fetch(`http://localhost:3000/sensorReadings/${sensorId}`, {
          method: 'DELETE'
        })
        .then(() => {
          fetchAndDisplaySensors(sortBy.value);  // Refresh the sensor list
after deletion
        });
      }
    });

    // Handle form submit (Create/Update sensor)
    form.addEventListener('submit', (event) => {
      event.preventDefault();

      const sensorId = form.getAttribute('data-edit-id');
      const updatedSensor = {
        type: document.getElementById('sensor-type').value,
        value: parseFloat(document.getElementById('sensor-value').value),
        timestamp: new Date().toISOString()
      };

      if (sensorId) {
        // If sensorId exists, it's an edit
        fetch(`http://localhost:3000/sensorReadings/${sensorId}`, {
          method: 'PATCH',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify(updatedSensor)
        })
        .then(response => response.json())
        .then(() => {
          form.removeAttribute('data-edit-id');  // Clear the edit flag
          form.reset();  // Reset the form
          fetchAndDisplaySensors(sortBy.value);  // Refresh the sensor list
        });
      } else {
        // Create new sensor data
        fetch('http://localhost:3000/sensorReadings', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
```

```
            },
            body: JSON.stringify(updatedSensor)
          })
          .then(response => response.json())
          .then(sensor => {
            displaySensors([sensor]);  // Append the new sensor to the UI
            form.reset();
          });
        }
      });

      // Handle sort option change
      sortBy.addEventListener('change', () => {
        fetchAndDisplaySensors(sortBy.value);
      });

      // Initial fetch to display sensors on page load, default sorting by
timestamp
      fetchAndDisplaySensors();
    });
```