FEDF ASSIGNEMTN 3

write Reactjs code for sum of two matrices using props and state

SOLUTION:

Src-

MatrixSumCalculator.jsx
Main.jsx

MatrixSumCalculator.jsx:

```
import React, { useState } from 'react';
const MatrixSumCalculator = () => {
  const [rowsA, setRowsA] = useState(2);
  const [colsA, setColsA] = useState(2);
  const [rowsB, setRowsB] = useState(2);
  const [colsB, setColsB] = useState(2);
  const [matrixA, setMatrixA] = useState([]);
  const [matrixB, setMatrixB] = useState([]);
  const [resultMatrix, setResultMatrix] = useState([]);

  // Initialize Matrix A
  const handleMatrixASetup = () => {
    const emptyMatrix = Array.from({ length: rowsA }, () =>
      Array.from({ length: colsA }, () => 0)
    );
    setMatrixA(emptyMatrix);
  };

  // Initialize Matrix B
  const handleMatrixBSetup = () => {
    const emptyMatrix = Array.from({ length: rowsB }, () =>
      Array.from({ length: colsB }, () => 0)
    );
    setMatrixB(emptyMatrix);
  };

  // Handle Input Changes for Both Matrices
  const handleMatrixChange = (matrixSetter, rowIndex, colIndex, value) => {
    matrixSetter((prevMatrix) => {
      const updatedMatrix = [...prevMatrix];
      updatedMatrix[rowIndex][colIndex] = parseInt(value) || 0; // Ensure numeric input
      return updatedMatrix;
    });
  };

  // Calculate Sum of Matrices
  const calculateSum = () => {
    if (
```

```
    matrixA.length !== matrixB.length ||
    matrixA[0].length !== matrixB[0].length
  ) {
    alert("Matrix A and Matrix B must have the same dimensions for addition.");
    return;
  }

  const result = matrixA.map((row, i) =>
    row.map((val, j) => val + matrixB[i][j])
  );

  setResultMatrix(result);
};

// Render a Matrix with Editable Inputs
const renderMatrix = (matrix, setMatrix) => (
  <table>
    <tbody>
      {matrix.map((row, rowIndex) => (
        <tr key={rowIndex}>
          {row.map((col, colIndex) => (
            <td key={colIndex}>
              <input
                type="number"
                value={col}
                onChange={(e) =>
                  handleMatrixChange(setMatrix, rowIndex, colIndex, e.target.value)
                }
                style={{ width: "50px", textAlign: "center" }}
              />
            </td>
          ))}
        </tr>
      ))}
    </tbody>
  </table>
);

return (
  <div>
    <h2>Matrix Sum Calculator</h2>

    {/* Matrix A Setup */}
    <div>
      <h3>Matrix A Dimensions</h3>
      <label>
        Rows:
        <input
          type="number"
          value={rowsA}
          onChange={(e) => setRowsA(parseInt(e.target.value) || 0)}
```

```
      style={{ width: "50px", marginLeft: "5px", marginRight: "10px" }}
    />
  </label>
  <label>
    Columns:
    <input
      type="number"
      value={colsA}
      onChange={(e) => setColsA(parseInt(e.target.value) || 0)}
      style={{ width: "50px", marginLeft: "5px", marginRight: "10px" }}
    />
  </label>
  <button onClick={handleMatrixASetup}>Set Matrix A</button>
</div>

{matrixA.length > 0 && (
  <>
    <h3>Matrix A</h3>
    {renderMatrix(matrixA, setMatrixA)}
  </>
)}

{/* Matrix B Setup */}
<div>
  <h3>Matrix B Dimensions</h3>
  <label>
    Rows:
    <input
      type="number"
      value={rowsB}
      onChange={(e) => setRowsB(parseInt(e.target.value) || 0)}
      style={{ width: "50px", marginLeft: "5px", marginRight: "10px" }}
    />
  </label>
  <label>
    Columns:
    <input
      type="number"
      value={colsB}
      onChange={(e) => setColsB(parseInt(e.target.value) || 0)}
      style={{ width: "50px", marginLeft: "5px", marginRight: "10px" }}
    />
  </label>
  <button onClick={handleMatrixBSetup}>Set Matrix B</button>
</div>

{matrixB.length > 0 && (
  <>
    <h3>Matrix B</h3>
    {renderMatrix(matrixB, setMatrixB)}
  </>
```

```
      )}

      {/* Calculate Sum */}
      {matrixA.length > 0 &&
       matrixB.length > 0 &&
       matrixA.length === matrixB.length &&
       matrixA[0].length === matrixB[0].length && (
        <button onClick={calculateSum} style={{ marginTop: "10px" }}>
          Calculate Sum
        </button>
       )}

      {/* Result */}
      {resultMatrix.length > 0 && (
       <>
        <h3>Result Matrix</h3>
        <table>
          <tbody>
            {resultMatrix.map((row, rowIndex) => (
             <tr key={rowIndex}>
               {row.map((col, colIndex) => (
                 <td key={colIndex}>{col}</td>
               ))}
             </tr>
            ))}
          </tbody>
        </table>
       </>
      )}
    </div>
  );
};

export default MatrixSumCalculator;
```

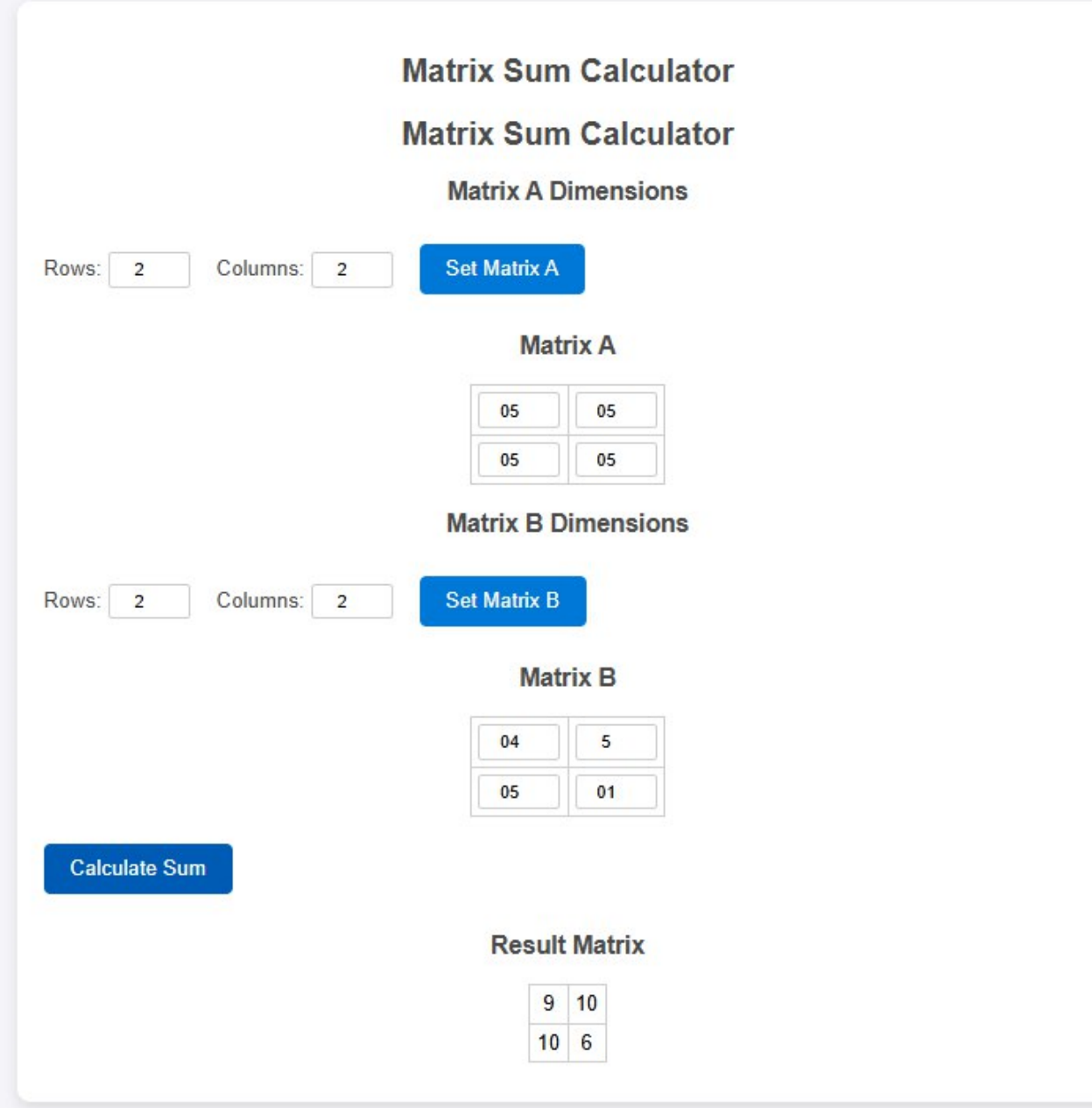
App.jsx

```
import React from "react";
import MatrixSumCalculator from "./MatrixSumCalculator"; // Ensure correct path
import "./Styles.css"; // Import the CSS file

const App = () => {
  return (
    <div className="container">
      <h2>Matrix Sum Calculator</h2>
      <MatrixSumCalculator />
    </div>
  );
};

export default App;
```

output-


Matrix Sum Calculator
Matrix Sum Calculator
Matrix A Dimensions
Rows: 2
Columns: 2
Set Matrix A
Matrix A
05
05
05
05
Matrix B Dimensions
Rows: 2
Columns: 2
Set Matrix B
Matrix B
04
5
05
01
Calculate Sum
Result Matrix
9
10
10
6