```
import React, { useState, useEffect } from 'react';

// Tailwind CSS is assumed to be available

const App = () => {
  const [initialAmount, setInitialAmount] = useState('');
  const [reductionPercent, setReductionPercent] = useState('12.13');
  const [suggestedReductionPercent, setSuggestedReductionPercent] =
useState(null); // New state for suggested percentage
  const [reducedAmount, setReducedAmount] = useState(null);
  const [portion30, setPortion30] = useState(null);
  const [final30, setFinal30] = useState(null);
  const [portion70, setPortion70] = useState(null);
  const [final70, setFinal70] = useState(null);
  const [totalFinalAmount, setTotalFinalAmount] = useState(null);
  const [errorMessage, setErrorMessage] = useState('');
  const [copySuccess, setCopySuccess] = useState('');

  // Effect to calculate suggested reduction percentage whenever
initialAmount changes
  useEffect(() => {
    const X = parseFloat(initialAmount);
    if (!isNaN(X) && X > 0) {
      // Derived from: X = X * (1 - reductionPercent / 100) * 1.138
      // So, 1 = (1 - reductionPercent / 100) * 1.138
      // (1 - reductionPercent / 100) = 1 / 1.138
      // reductionPercent / 100 = 1 - (1 / 1.138)
      // reductionPercent = (1 - (1 / 1.138)) * 100
      const calculatedSuggestion = (1 - (1 / 1.138)) * 100;
      setSuggestedReductionPercent(calculatedSuggestion);
    } else {
      setSuggestedReductionPercent(null);
    }
    resetResults(); // Clear results when initial amount changes
  }, [initialAmount]);

  const handleCalculate = () => {
    setErrorMessage(''); // Clear previous errors
    setCopySuccess(''); // Clear previous copy success message
    const X = parseFloat(initialAmount);
    const reduceBy = parseFloat(reductionPercent);

    if (isNaN(X) || X <= 0) {
      setErrorMessage("Please enter a valid positive number for the
initial amount.");
      resetResults();
      return;
    }
```

```
    if (isNaN(reduceBy) || reduceBy < 0 || reduceBy > 100) {
      setErrorMessage("Please enter a valid percentage (0-100) for the
reduction.");
      resetResults();
      return;
    }

    // Step 1: Minus the variable percentage
    const reductionFactor = reduceBy / 100;
    const calculatedReducedAmount = X * (1 - reductionFactor);
    setReducedAmount(calculatedReducedAmount);

    // Step 2: Split reduced amount in 30/70
    const calculatedPortion30 = calculatedReducedAmount * 0.30;
    const calculatedPortion70 = calculatedReducedAmount * 0.70;
    setPortion30(calculatedPortion30);
    setPortion70(calculatedPortion70);

    // Step 3: Add 18% to 30% portion
    const gstRate30 = 0.18;
    const calculatedFinal30 = calculatedPortion30 * (1 + gstRate30);
    setFinal30(calculatedFinal30);

    // Step 4: Add 12% to 70% portion
    const gstRate70 = 0.12;
    const calculatedFinal70 = calculatedPortion70 * (1 + gstRate70);
    setFinal70(calculatedFinal70);

    // Total final amount
    setTotalFinalAmount(calculatedFinal30 + calculatedFinal70);
  };

  const resetResults = () => {
    setReducedAmount(null);
    setPortion30(null);
    setFinal30(null);
    setPortion70(null);
    setFinal70(null);
    setTotalFinalAmount(null);
  };

  const copyResultsToClipboard = () => {
    if (reducedAmount === null) {
      setCopySuccess("No results to copy. Please calculate first.");
      return;
    }
```

```
    const resultsText = `
Financial Split Calculation Results:

Initial Amount (X): ₹${parseFloat(initialAmount).toFixed(2)}
First Reduction Percentage:
${parseFloat(reductionPercent).toFixed(4)}%

Amount after -${parseFloat(reductionPercent).toFixed(4)}%:
₹${reducedAmount.toFixed(2)}

30% Portion:
  Initial 30% Portion: ₹${portion30.toFixed(2)}
  Final 30% Portion (+18%): ₹${final30.toFixed(2)}

70% Portion:
  Initial 70% Portion: ₹${portion70.toFixed(2)}
  Final 70% Portion (+12%): ₹${final70.toFixed(2)}

Grand Total (Final 30% + Final 70%): ₹${totalFinalAmount.toFixed(2)}
    `.trim();

    const textarea = document.createElement('textarea');
    textarea.value = resultsText;
    document.body.appendChild(textarea);
    textarea.select();
    try {
      document.execCommand('copy');
      setCopySuccess('Results copied to clipboard!');
    } catch (err) {
      console.error('Failed to copy text:', err);
      setCopySuccess('Failed to copy results.');
    }
    document.body.removeChild(textarea);

    setTimeout(() => setCopySuccess(''), 3000);
  };

  return (
    <div className="min-h-screen bg-gradient-to-br from-purple-50
to-pink-100 p-4 sm:p-6 md:p-8 font-inter text-gray-800 flex flex-col
items-center justify-center">
      <h1 className="text-3xl sm:text-4xl font-extrabold text-center
text-purple-800 mb-8 mt-4 rounded-lg p-3 bg-white shadow-lg">
        Financial Split Calculator
      </h1>

      <div className="w-full max-w-md bg-white p-6 rounded-xl
shadow-lg mb-8">
```

```
        <h2 className="text-2xl font-bold text-purple-700 mb-6">Enter
Amounts & Percentages</h2>

        <div className="mb-4">
          <label htmlFor="initialAmount" className="block
text-gray-700 text-sm font-semibold mb-2">
            Initial Amount (X):
          </label>
          <input
            type="number"
            id="initialAmount"
            placeholder="Enter initial amount (X)"
            value={initialAmount}
            onChange={(e) => setInitialAmount(e.target.value)}
            className="w-full p-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-purple-500 outline-none transition
duration-200 text-lg"
          />
        </div>

        {suggestedReductionPercent !== null && (
          <div className="mb-4 p-3 bg-yellow-50 border
border-yellow-200 rounded-lg flex items-center justify-between
shadow-sm">
            <div>
              <p className="text-sm font-semibold text-yellow-800">
                Suggested Reduction % (for Total = X):
              </p>
              <p className="text-xl font-bold text-yellow-900">
                {suggestedReductionPercent.toFixed(8)}% {/* Display
with high precision */}
              </p>
            </div>
            <button
              onClick={() => {

setReductionPercent(suggestedReductionPercent.toFixed(8)); // Apply
suggested %
                setErrorMessage(''); // Clear any percentage error
              }}
              className="ml-4 bg-yellow-600 text-white text-sm px-4
py-2 rounded-md hover:bg-yellow-700 transition duration-200 shadow-md"
            >
              Apply
            </button>
          </div>
        )}
```

```
        <div className="mb-6">
          <label htmlFor="reductionPercent" className="block
text-gray-700 text-sm font-semibold mb-2">
            First Reduction Percentage (%):
          </label>
          <input
            type="number"
            id="reductionPercent"
            placeholder="e.g., 12.13"
            value={reductionPercent}
            onChange={(e) => setReductionPercent(e.target.value)}
            className="w-full p-3 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-purple-500 outline-none transition
duration-200 text-lg"
          />
        </div>

        {errorMessage && (
          <p className="text-red-600 text-sm mb-4">{errorMessage}</p>
        )}
        <button
          onClick={handleCalculate}
          className="w-full bg-purple-600 text-white font-semibold
py-3 rounded-lg hover:bg-purple-700 transition duration-300
ease-in-out transform hover:scale-105 shadow-md"
        >
          Calculate
        </button>
      </div>

      {reducedAmount !== null && (
        <div className="w-full max-w-md bg-white p-6 rounded-xl
shadow-lg">
          <h2 className="text-2xl font-bold text-purple-700
mb-6">Calculation Results</h2>

          <div className="space-y-4 mb-6">
            <div className="flex justify-between items-center
bg-blue-50 p-3 rounded-md shadow-sm">
              <p className="font-medium text-blue-700">Amount after
-{parseFloat(reductionPercent).toFixed(4)}%:</p>
              <p className="font-bold text-lg
text-blue-900">₹{reducedAmount.toFixed(2)}</p>
            </div>

            <div className="border-t border-gray-200 pt-4">
              <h3 className="text-xl font-semibold text-gray-700
mb-2">30% Portion</h3>
```

```
              <div className="flex justify-between items-center pl-4
pr-2 py-2">
                <p className="text-gray-600">Initial 30% Portion:</p>
                <p className="font-semibold
text-gray-800">₹{portion30.toFixed(2)}</p>
              </div>
              <div className="flex justify-between items-center
bg-green-50 p-3 rounded-md shadow-sm">
                <p className="font-medium text-green-700">Final 30%
Portion (+18%):</p>
                <p className="font-bold text-lg
text-green-900">₹{final30.toFixed(2)}</p>
              </div>
            </div>

            <div className="border-t border-gray-200 pt-4">
              <h3 className="text-xl font-semibold text-gray-700
mb-2">70% Portion</h3>
              <div className="flex justify-between items-center pl-4
pr-2 py-2">
                <p className="text-gray-600">Initial 70% Portion:</p>
                <p className="font-semibold
text-gray-800">₹{portion70.toFixed(2)}</p>
              </div>
              <div className="flex justify-between items-center
bg-red-50 p-3 rounded-md shadow-sm">
                <p className="font-medium text-red-700">Final 70%
Portion (+12%):</p>
                <p className="font-bold text-lg
text-red-900">₹{final70.toFixed(2)}</p>
              </div>
            </div>

            <div className="border-t-2 border-purple-500 pt-4 mt-6">
              <div className="flex justify-between items-center
bg-purple-100 p-4 rounded-md shadow-md">
                <p className="text-xl font-bold text-purple-800">Grand
Total (Final 30% + Final 70%):</p>
                <p className="text-2xl font-extrabold
text-purple-900">₹{totalFinalAmount.toFixed(2)}</p>
              </div>
            </div>
          </div>

          <button
            onClick={copyResultsToClipboard}
            className="w-full bg-indigo-600 text-white font-semibold
py-3 rounded-lg hover:bg-indigo-700 transition duration-300
```

```
ease-in-out transform hover:scale-105 shadow-md mt-4"
          >
            Copy Results to Clipboard
          </button>
          {copySuccess && (
            <p className="text-center text-green-600 text-sm
mt-2">{copySuccess}</p>
          )}
        </div>
      )}
    </div>
  );
};

export default App;
```