Calculator.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Manoj SJ | Advanced Calculator</title>
  <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
rel="stylesheet"
onerror="this.href='https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2.2.19/tailwind.
min.css';">
  <link
href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swa
p" rel="stylesheet"
onerror="this.href='https://fonts.googleapis.com/css2?family=Arial&display=swap';">
  <style>
    body {
      background: linear-gradient(135deg, #2d3748, #4a5568);
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      margin: 0;
    }

    .calculator {
      background: linear-gradient(135deg, #e5e7eb, #d1d5db);
      padding: 20px;
    border-radius: 12px;
    width: 400px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.display-container {
    background-color: #4a5568;
    padding: 10px;
    border-radius: 8px;
    margin-bottom: 10px;
}

.display {
    background-color: #2d3748;
    color: #48bb78;
    padding: 15px;
    text-align: right;
    font-size: 2.2em;
    border-radius: 4px;
}

.buttons {
    display: grid;
    grid-template-columns: repeat(5, 1fr);
    gap: 6px;
}

.btn {
```

```
            padding: 12px;
            background-color: #4a5568;
            color: #e5e7eb;
            border-radius: 6px;
            border: none;
        }

        .error-message {
            color: #f56565;
            font-size: 0.9em;
            text-align: right;
            padding: 5px;
            display: none;
        }

        /* Hidden by default */
    </style>
</head>

<body>
    <div class="calculator">
        <div class="flex justify-between items-center mb-4">
            <h4 class="text-lg text-white font-semibold">Advanced Calculator</h4>
        </div>
        <div class="display-container">
            <div class="memory-indicator text-sm text-yellow-400 text-left font-mono hidden"
id="memoryIndicator">M</div>
            <div class="history-display text-sm text-gray-300 text-right font-mono"
id="history"></div>
```

```
        <div class="display" id="display">0</div>

        <div class="error-message" id="error"></div> <!-- No longer hidden by default, but
errors are avoided -->

      </div>

      <div class="buttons grid grid-cols-5 gap-2">

        <button class="btn clear bg-red-600 text-white p-3 rounded-lg"
onclick="clearDisplay()">C</button>

        <button class="btn function bg-gray-600 text-white p-3 rounded-lg"
onclick="backspace()">⌫</button>

        <button class="btn function bg-gray-600 text-white p-3 rounded-lg"
onclick="memoryClear()">MC</button>

        <button class="btn function bg-gray-600 text-white p-3 rounded-lg"
onclick="memoryRecall()">MR</button>

        <button class="btn function bg-gray-600 text-white p-3 rounded-lg"
onclick="memoryAdd()">M+</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('7')">7</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('8')">8</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('9')">9</button>

        <button class="btn operator bg-yellow-600 text-white p-3 rounded-lg"
onclick="appendOperator('/')">/</button>

        <button class="btn function bg-gray-600 text-white p-3 rounded-lg"
onclick="memorySubtract()">M-</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('4')">4</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('5')">5</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('6')">6</button>

        <button class="btn operator bg-yellow-600 text-white p-3 rounded-lg"
onclick="appendOperator('*')">×</button>
```

```
        <button class="btn function bg-green-600 text-white p-3 rounded-lg"
onclick="calculatePercentage()">%</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('1')">1</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('2')">2</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('3')">3</button>

        <button class="btn operator bg-yellow-600 text-white p-3 rounded-lg"
onclick="appendOperator('-')">-</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg col-span-2"
onclick="appendNumber('0')">0</button>

        <button class="btn number bg-gray-700 text-white p-3 rounded-lg"
onclick="appendNumber('.')">.</button>

        <button class="btn operator bg-yellow-600 text-white p-3 rounded-lg"
onclick="appendOperator('+')">+</button>

        <button class="btn function bg-green-600 text-white p-3 rounded-lg"
onclick="calculateSquareRoot()">√</button>

        <button class="btn equals bg-blue-600 text-white p-3 rounded-lg"
onclick="calculate()">=</button>

      </div>

    </div>

    <script>

      // Initialize elements

      const display = document.getElementById('display');

      const historyDisplay = document.getElementById('history');

      const errorMessage = document.getElementById('error');

      const memoryIndicator = document.getElementById('memoryIndicator');

      let currentInput = '0';

      let previousInput = '';

      let operator = '';
      let shouldResetDisplay = false;

      let memory = 0;


      // Update memory indicator

      function updateMemoryIndicator() {

        memoryIndicator.classList.toggle('hidden', memory === 0);

      }


      // Validate numeric input

      function isValidNumber(str) {

        return /^-?\d*\.?\d+$/.test(str) && !isNaN(parseFloat(str));

      }


      // Append number or decimal with silent correction

      function appendNumber(number) {

        if (shouldResetDisplay) {

          currentInput = number;

          shouldResetDisplay = false;

        } else if (number === '.' && currentInput.includes('.')) {

          return; // Silently ignore duplicate decimal

        } else if (number === '.' && !currentInput) {

          currentInput = '0.';

        } else {

          currentInput = currentInput === '0' && number !== '.' ? number : currentInput +
number;

        }

        if (!isValidNumber(currentInput)) {

          currentInput = '0'; // Reset to 0 without error
          }
          updateDisplay();
      }

      // Append operator with silent validation
      function appendOperator(op) {
          if (!currentInput || !isValidNumber(currentInput)) {
              currentInput = '0'; // Reset invalid input
              return;
          }
          if (currentInput && (previousInput || operator)) calculate();
          previousInput = currentInput;
          operator = op;
          historyDisplay.textContent = `${previousInput} ${op}`;
          shouldResetDisplay = true;
          updateDisplay();
      }

      // Update display
      function updateDisplay() {
          display.textContent = currentInput.length > 12 ? currentInput.slice(0, 12) + '...' :
currentInput;
          if (!operator) historyDisplay.textContent = '';
      }

      // Clear all
      function clearDisplay() {
          currentInput = '0';
      previousInput = '';
      operator = '';
      shouldResetDisplay = false;
      historyDisplay.textContent = '';
      updateDisplay();
}

// Backspace
function backspace() {
      currentInput = currentInput.length > 1 ? currentInput.slice(0, -1) : '0';
      if (!isValidNumber(currentInput)) {
          currentInput = '0'; // Reset without error
      }
      updateDisplay();
}

// Perform calculation with infinity for division by zero
function calculate() {
      if (!previousInput || !currentInput || !operator) return;

      const num1 = parseFloat(previousInput);
      const num2 = parseFloat(currentInput);

      if (isNaN(num1) || isNaN(num2)) {
          currentInput = '0'; // Reset without error
          return;
      }
```

```
let result;
switch (operator) {
  case '+':
    result = num1 + num2;
    break;
  case '-':
    result = num1 - num2;
    break;
  case '*':
    result = num1 * num2;
    break;
  case '/':
    if (num2 === 0) {
      result = '∞';
    } else {
      result = num1 / num2;
    }
    break;
  default:
    currentInput = '0'; // Reset for unknown operator
    return;
}

if (typeof result === 'number' && !isFinite(result)) {
  currentInput = '0'; // Reset without error
  return;
}
if (typeof result === 'number' && result.toString().length > 12) {
        result = result.toPrecision(10); // Adjust precision silently
    }

    currentInput = result.toString();
    previousInput = '';
    operator = '';
    shouldResetDisplay = true;
    historyDisplay.textContent = '';
    updateDisplay();
}

// Calculate percentage with silent correction
function calculatePercentage() {
    if (!currentInput || !isValidNumber(currentInput)) {
        currentInput = '0'; // Reset invalid input
        return;
    }
    const num = parseFloat(currentInput);
    let result = num / 100;

    if (previousInput && operator) {
        const prevNum = parseFloat(previousInput);
        if (isNaN(prevNum)) {
            currentInput = '0'; // Reset invalid previous number
            return;
        }
        result = (operator === '+' || operator === '-') ? prevNum * (num / 100) : num / 100;
    }
```

```
        currentInput = result.toString();
        historyDisplay.textContent = `${previousInput || ''} ${operator || ''} ${num}%`;
        shouldResetDisplay = true;
        updateDisplay();
    }

    // Memory functions with silent correction
    function memoryAdd() {
        if (currentInput && isValidNumber(currentInput)) memory +=
parseFloat(currentInput);
        else currentInput = '0'; // Reset invalid input
        updateMemoryIndicator();
    }

    function memorySubtract() {
        if (currentInput && isValidNumber(currentInput)) memory -=
parseFloat(currentInput);
        else currentInput = '0'; // Reset invalid input
        updateMemoryIndicator();
    }

    function memoryRecall() {
        currentInput = memory.toString();
        shouldResetDisplay = true;
        updateDisplay();
    }

    function memoryClear() {
```

```
    memory = 0;
    updateMemoryIndicator();
}

// Calculate square root with silent correction
function calculateSquareRoot() {
    if (!currentInput || !isValidNumber(currentInput)) {
        currentInput = '0'; // Reset invalid input
        return;
    }
    const num = parseFloat(currentInput);
    if (num < 0) {
        currentInput = '0'; // Reset for negative number
        return;
    }
    currentInput = Math.sqrt(num).toString();
    historyDisplay.textContent = `√(${num})`;
    shouldResetDisplay = true;
    updateDisplay();
}

// Keyboard support
document.addEventListener('keydown', (e) => {
    e.preventDefault();
    const key = e.key;
    if (/[0-9]/.test(key)) appendNumber(key);
    else if (key === '.') appendNumber('.');
    else if (['+', '-', '*', '/'].includes(key)) appendOperator(key);
```

```
            else if (key === 'Enter' || key === '=') calculate();
            else if (key === 'Escape') clearDisplay();
            else if (key === 'Backspace') backspace();
            else if (key.toLowerCase() === 'p') calculatePercentage();
            else if (key.toLowerCase() === 'm') memoryRecall();
        });

        // Initialize
        updateMemoryIndicator();
    </script>
</body>

</html>
```

Script.jss

```
/* Global Styles */
body {
    font-family: 'Roboto', sans-serif;
    margin: 0;
    padding: 0;
}

.calculator-container {
    perspective: 1000px;
}

.calculator {
    width: 400px;
    background: linear-gradient(135deg, #d1d5db, #9ca3af);
    border-radius: 12px;
    padding: 20px;
    box-shadow: 10px 10px 20px rgba(0, 0, 0, 0.3), -5px -5px 10px rgba(255,
255, 255, 0.2);
    border: 2px solid #4b5e6a;
    transform: rotateX(10deg) rotateY(5deg);
    transition: transform 0.3s ease;
}

.calculator:hover {
    transform: rotateX(10deg) rotateY(5deg) scale(1.02);
}

.display-container {
    background: #1a202c;
    padding: 10px;
    border-radius: 8px;
    box-shadow: inset 2px 2px 5px rgba(0, 0, 0, 0.5);
    position: relative;
    margin-bottom: 10px;
}

.memory-indicator {
    position: absolute;

    top: 10px;

    left: 10px;

    font-size: 0.8em;

    font-weight: bold;

    color: #facc15;

}


.history-display {

    min-height: 1.5em;

    padding: 5px;

    color: #d1d5db;

    opacity: 0.7;

    text-align: right;

}


.display {

    background: #2d3748;

    padding: 15px;

    font-size: 2.2em;

    text-align: right;

    border-radius: 5px;

    color: #10b981;

    font-family: 'Orbitron', sans-serif;

    text-shadow: 0 0 5px rgba(16, 185, 129, 0.5);
    overflow: hidden;
    white-space: nowrap;
    text-overflow: ellipsis;
    animation: flicker 0.1s infinite alternate;
}

@keyframes flicker {
    0% {
      opacity: 1;
    }

    100% {
      opacity: 0.95;
    }
}

.error-message {
    padding: 5px;
    font-size: 0.9em;
    color: #ef4444;
}

.buttons {
    display: grid;
    grid-template-columns: repeat(5, 1fr);
    gap: 6px;
}

.btn {
    padding: 12px;
    font-size: 1.2em;
    border: none;
    border-radius: 6px;
    cursor: pointer;
    background: linear-gradient(145deg, #4b5e6a, #2d3748);
    box-shadow: 3px 3px 6px rgba(0, 0, 0, 0.4), -2px -2px 4px rgba(255, 255,
255, 0.1);
    color: #e5e7eb;
    transition: transform 0.1s, box-shadow 0.1s, background 0.2s;
}

.btn:hover {
    transform: translateY(-2px);
    box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5), -3px -3px 6px rgba(255, 255,
255, 0.15);
    background: linear-gradient(145deg, #5a6b88, #3d4a60);
}

.btn:active {
    transform: translateY(2px);
    box-shadow: inset 2px 2px 5px rgba(0, 0, 0, 0.5);
```

```css
}

.btn.number {
   background: linear-gradient(145deg, #6b7280, #4b5563);
   color: #f9fafb;
}

.btn.operator {
   background: linear-gradient(145deg, #f97316, #ea580c);
   color: #ffffff;
}

.btn.function {
   background: linear-gradient(145deg, #10b981, #065f46);
   color: #ffffff;
}

.btn.clear {
   background: linear-gradient(145deg, #ef4444, #dc2626);
   color: #ffffff;
}

.btn.equals {
   background: linear-gradient(145deg, #3b82f6, #2563eb);
   color: #ffffff;
```

```css
  grid-column: span 2;
}

/* Responsive Design */
@media (max-width: 640px) {
  .calculator {
    width: 90%;
    max-width: 360px;
    transform: rotateX(0deg) rotateY(0deg);
  }

  .btn {
    padding: 10px;
    font-size: 1em;
  }

  .display {
    font-size: 1.8em;
    padding: 10px;
  }

  .history-display {
    font-size: 0.9em;
  }
}
```

Styles.css

```
/* Global Styles */
body {
   font-family: 'Roboto', sans-serif;
   margin: 0;
   padding: 0;
}

.calculator-container {
   perspective: 1000px;
}

.calculator {
   width: 400px;
   background: linear-gradient(135deg, #d1d5db, #9ca3af);
   border-radius: 12px;
   padding: 20px;
   box-shadow: 10px 10px 20px rgba(0, 0, 0, 0.3), -5px -5px 10px rgba(255,
255, 255, 0.2);
   border: 2px solid #4b5e6a;
   transform: rotateX(10deg) rotateY(5deg);
   transition: transform 0.3s ease;
}

.calculator:hover {
   transform: rotateX(10deg) rotateY(5deg) scale(1.02);
}

.display-container {
   background: #1a202c;
   padding: 10px;
   border-radius: 8px;
   box-shadow: inset 2px 2px 5px rgba(0, 0, 0, 0.5);
   position: relative;
   margin-bottom: 10px;
}

.memory-indicator {
   position: absolute;
   top: 10px;
   left: 10px;
   font-size: 0.8em;
   font-weight: bold;
   color: #facc15;
}

.history-display {
   min-height: 1.5em;
   padding: 5px;
   color: #d1d5db;
```

```
  opacity: 0.7;
  text-align: right;
}

.display {
  background: #2d3748;
  padding: 15px;
  font-size: 2.2em;
  text-align: right;
  border-radius: 5px;
  color: #10b981;
  font-family: 'Orbitron', sans-serif;
  text-shadow: 0 0 5px rgba(16, 185, 129, 0.5);
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
  animation: flicker 0.1s infinite alternate;
}

@keyframes flicker {
  0% {
    opacity: 1;
  }

  100% {
        opacity: 0.95;
    }
}

.error-message {
    padding: 5px;
    font-size: 0.9em;
    color: #ef4444;
}

.buttons {
    display: grid;
    grid-template-columns: repeat(5, 1fr);
    gap: 6px;
}

.btn {
    padding: 12px;
    font-size: 1.2em;
    border: none;
    border-radius: 6px;
    cursor: pointer;
    background: linear-gradient(145deg, #4b5e6a, #2d3748);
    box-shadow: 3px 3px 6px rgba(0, 0, 0, 0.4), -2px -2px 4px rgba(255, 255,
255, 0.1);
```

```
    color: #e5e7eb;

    transition: transform 0.1s, box-shadow 0.1s, background 0.2s;

}


.btn:hover {

    transform: translateY(-2px);

    box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5), -3px -3px 6px rgba(255, 255,
255, 0.15);

    background: linear-gradient(145deg, #5a6b88, #3d4a60);

}


.btn:active {

    transform: translateY(2px);

    box-shadow: inset 2px 2px 5px rgba(0, 0, 0, 0.5);

}


.btn.number {

    background: linear-gradient(145deg, #6b7280, #4b5563);

    color: #f9fafb;

}


.btn.operator {

    background: linear-gradient(145deg, #f97316, #ea580c);

    color: #ffffff;

}
```

```
.btn.function {
  background: linear-gradient(145deg, #10b981, #065f46);
  color: #ffffff;
}

.btn.clear {
  background: linear-gradient(145deg, #ef4444, #dc2626);
  color: #ffffff;
}

.btn.equals {
  background: linear-gradient(145deg, #3b82f6, #2563eb);
  color: #ffffff;
  grid-column: span 2;
}

/* Responsive Design */
@media (max-width: 640px) {
  .calculator {
    width: 90%;
    max-width: 360px;
    transform: rotateX(0deg) rotateY(0deg);
  }
  .btn {
    padding: 10px;
    font-size: 1em;
  }

  .display {
    font-size: 1.8em;
    padding: 10px;
  }

  .history-display {
    font-size: 0.9em;
  }
}
```

Project link: http://calculatormanoj.ccbp.tech