```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Green-Builder Carbon Savings Widget</title>
  <style>
    /* Widget Styles */
    .carbon-widget {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      max-width: 800px;
      margin: 20px auto;
      background: linear-gradient(135deg, #f8fdf8 0%, #e8f5e8 100%);
      border: 2px solid #4a7c59;
      border-radius: 12px;
      box-shadow: 0 8px 32px rgba(74, 124, 89, 0.15);
      overflow: hidden;
    }

    .widget-header {
      background: linear-gradient(135deg, #4a7c59 0%, #6b8e23 100%);
      color: white;
      padding: 20px;
      text-align: center;
    }

    .widget-header h2 {
      margin: 0;
      font-size: 24px;
      font-weight: 600;
    }
```

```
.widget-header p {
    margin: 8px 0 0 0;
    opacity: 0.9;
    font-size: 14px;
}

.widget-content {
    padding: 30px;
}

.step {
    display: none;
    animation: fadeIn 0.3s ease-in;
}

.step.active {
    display: block;
}

@keyframes fadeIn {
    from { opacity: 0; transform: translateY(10px); }
    to { opacity: 1; transform: translateY(0); }
}

.step-header {
    text-align: center;
    margin-bottom: 25px;
}

.step-title {
```

```
    color: #4a7c59;
    font-size: 20px;
    font-weight: 600;
    margin: 0;
}

.step-description {
    color: #6b8e23;
    margin: 8px 0 0 0;
    font-size: 14px;
}

.form-group {
    margin-bottom: 20px;
}

.form-row {
    display: flex;
    gap: 15px;
    flex-wrap: wrap;
}

.form-row .form-group {
    flex: 1;
    min-width: 200px;
}

label {
    display: block;
    color: #4a7c59;
    font-weight: 500;
```

```
    margin-bottom: 6px;
    font-size: 14px;
}

input[type="number"], input[type="text"], select {
    width: 100%;
    padding: 12px;
    border: 2px solid #d4d4aa;
    border-radius: 8px;
    font-size: 14px;
    background: white;
    transition: border-color 0.3s ease;
    box-sizing: border-box;
}

input[type="number"]:focus, input[type="text"]:focus, select:focus {
    outline: none;
    border-color: #6b8e23;
    box-shadow: 0 0 0 3px rgba(107, 142, 35, 0.1);
}

.module-selection {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(180px, 1fr));
    gap: 15px;
    margin-bottom: 25px;
}

.module-card {
    background: white;
    border: 2px solid #d4d4aa;
```

```
    border-radius: 8px;
    padding: 20px;
    text-align: center;
    cursor: pointer;
    transition: all 0.3s ease;
}

.module-card:hover {
    border-color: #6b8e23;
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(107, 142, 35, 0.15);
}

.module-card.selected {
    border-color: #4a7c59;
    background: #f0f8f0;
}

.module-card input[type="checkbox"] {
    display: none;
}

.module-icon {
    font-size: 32px;
    margin-bottom: 10px;
    display: block;
}

.module-name {
    color: #4a7c59;
    font-weight: 600;
```

```
    margin: 0;
}

.dynamic-inputs {
    background: white;
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 20px;
    border: 1px solid #d4d4aa;
}

.dynamic-inputs h4 {
    color: #4a7c59;
    margin: 0 0 15px 0;
    font-size: 16px;
}

.navigation {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-top: 30px;
    padding-top: 20px;
    border-top: 1px solid #d4d4aa;
}

.btn {
    padding: 12px 24px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
```

```
    font-size: 14px;
    font-weight: 500;
    transition: all 0.3s ease;
    text-decoration: none;
    display: inline-block;
}

.btn-primary {
    background: linear-gradient(135deg, #6b8e23 0%, #4a7c59 100%);
    color: white;
}

.btn-primary:hover {
    transform: translateY(-1px);
    box-shadow: 0 4px 12px rgba(107, 142, 35, 0.3);
}

.btn-secondary {
    background: #f0f8f0;
    color: #4a7c59;
    border: 1px solid #d4d4aa;
}

.btn-secondary:hover {
    background: #e8f5e8;
}

.results-container {
    background: white;
    border-radius: 12px;
    padding: 25px;
```

```
    margin-top: 20px;
}

.results-header {
    text-align: center;
    margin-bottom: 30px;
}

.results-title {
    color: #4a7c59;
    font-size: 24px;
    font-weight: 600;
    margin: 0;
}

.summary-cards {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.summary-card {
    background: linear-gradient(135deg, #f8fdf8 0%, #e8f5e8 100%);
    border: 1px solid #d4d4aa;
    border-radius: 8px;
    padding: 20px;
    text-align: center;
}

.card-value {
```

```
    font-size: 24px;
    font-weight: 700;
    color: #4a7c59;
    margin: 0;
}

.card-label {
    color: #6b8e23;
    font-size: 12px;
    margin: 5px 0 0 0;
    text-transform: uppercase;
    letter-spacing: 0.5px;
}

.chart-container {
    background: white;
    border: 1px solid #d4d4aa;
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 20px;
}

.chart-title {
    color: #4a7c59;
    font-size: 16px;
    font-weight: 600;
    margin: 0 0 15px 0;
    text-align: center;
}

.step-indicator {
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    margin-bottom: 20px;
}

.step-dot {
    width: 12px;
    height: 12px;
    border-radius: 50%;
    background: #d4d4aa;
    margin: 0 8px;
    transition: background-color 0.3s ease;
}

.step-dot.active {
    background: #6b8e23;
}

.step-dot.completed {
    background: #4a7c59;
}

.error-message {
    color: #d32f2f;
    font-size: 12px;
    margin-top: 5px;
}

.comparison-table {
    width: 100%;
```

```
    border-collapse: collapse;
    margin-top: 20px;
}

.comparison-table th,
.comparison-table td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #d4d4aa;
}

.comparison-table th {
    background: #f0f8f0;
    color: #4a7c59;
    font-weight: 600;
}

@media (max-width: 768px) {
    .carbon-widget {
        margin: 10px;
        border-radius: 8px;
    }

    .widget-content {
        padding: 20px;
    }

    .form-row {
        flex-direction: column;
    }
```

```
            .module-selection {
                grid-template-columns: 1fr;
            }

            .summary-cards {
                grid-template-columns: 1fr;
            }

            .navigation {
                flex-direction: column;
                gap: 10px;
            }
        }
    </style>
</head>
<body>
    <div class="carbon-widget" id="carbonWidget">
        <div class="widget-header">
            <h2>🌱 Green-Builder Carbon Savings Calculator</h2>
            <p>Calculate your environmental impact and cost savings from sustainable building features</p>
        </div>

        <div class="widget-content">
            <div class="step-indicator">
                <div class="step-dot active" data-step="1"></div>
                <div class="step-dot" data-step="2"></div>
                <div class="step-dot" data-step="3"></div>
                <div class="step-dot" data-step="4"></div>
            </div>
```

```
        <!-- Step 1: Profile -->
        <div class="step active" id="step1">
          <div class="step-header">
            <h3 class="step-title">Building Profile</h3>
            <p class="step-description">Tell us about your building to get started</p>
          </div>

          <div class="form-row">
            <div class="form-group">
              <label for="buildingArea">Building Area (sq ft)</label>
              <input type="number" id="buildingArea" min="0" step="1" placeholder="e.g., 2000"
required>
              <div class="error-message" id="areaError"></div>
            </div>
            <div class="form-group">
              <label for="buildingType">Building Type</label>
              <select id="buildingType" required>
                <option value="">Select type</option>
                <option value="residential">Residential</option>
                <option value="commercial">Commercial</option>
                <option value="industrial">Industrial</option>
                <option value="mixed">Mixed Use</option>
              </select>
            </div>
          </div>

          <div class="form-group">
            <label for="occupants">Number of Occupants</label>
            <input type="number" id="occupants" min="1" step="1" placeholder="e.g., 4" required>
            <div class="error-message" id="occupantsError"></div>
          </div>
```

```
</div>

<!-- Step 2: Module Selection -->
<div class="step" id="step2">
  <div class="step-header">
    <h3 class="step-title">Select Green Features</h3>
    <p class="step-description">Choose which sustainable features you want to analyze</p>
  </div>

  <div class="module-selection">
    <div class="module-card" data-module="electricity">
      <span class="module-icon">⚡</span>
      <h4 class="module-name">Solar Electricity</h4>
      <input type="checkbox" id="electricityModule" value="electricity">
    </div>
    <div class="module-card" data-module="water">
      <span class="module-icon">💧</span>
      <h4 class="module-name">Water Conservation</h4>
      <input type="checkbox" id="waterModule" value="water">
    </div>
    <div class="module-card" data-module="building">
      <span class="module-icon">🏗️</span>
      <h4 class="module-name">Green Building</h4>
      <input type="checkbox" id="buildingModule" value="building">
    </div>
    <div class="module-card" data-module="iot">
      <span class="module-icon">🤖</span>
      <h4 class="module-name">IoT Controls</h4>
      <input type="checkbox" id="iotModule" value="iot">
    </div>
  </div>
```

```
</div>

<!-- Step 3: Dynamic Inputs -->
<div class="step" id="step3">
  <div class="step-header">
    <h3 class="step-title">Feature Details</h3>
    <p class="step-description">Provide specific information for your selected features</p>
  </div>

  <div id="dynamicInputs"></div>
</div>

<!-- Step 4: Results -->
<div class="step" id="step4">
  <div class="results-container">
    <div class="results-header">
      <h3 class="results-title">Your Carbon Savings Report</h3>
    </div>

    <div class="summary-cards" id="summaryCards"></div>

    <div class="chart-container">
      <h4 class="chart-title">Monthly Carbon Savings by Feature</h4>
      <canvas id="savingsChart" width="400" height="200"></canvas>
    </div>

    <table class="comparison-table" id="comparisonTable">
      <thead>
        <tr>
          <th>Metric</th>
          <th>Conventional</th>
```

```
                    <th>Sustainable</th>
                    <th>Savings</th>
                  </tr>
                </thead>
                <tbody id="comparisonTableBody"></tbody>
              </table>

              <div style="text-align: center; margin-top: 30px;">
                <button class="btn btn-primary" onclick="downloadReport()">📄 Download Report</button>
                <button class="btn btn-secondary" onclick="resetWidget()" style="margin-left: 10px;">🔄 Start Over</button>
              </div>
            </div>
          </div>

          <div class="navigation">
            <button class="btn btn-secondary" id="prevBtn" onclick="previousStep()" style="display: none;">← Previous</button>
            <div></div>
            <button class="btn btn-primary" id="nextBtn" onclick="nextStep()">Next →</button>
          </div>
        </div>
      </div>

      <script>
        // Constants and Configuration
        const EMISSION_FACTORS = {
          electricity: 0.92, // kg CO2e per kWh (US average)
          water: 0.298, // kg CO2e per gallon
          gas: 5.3 // kg CO2e per therm
        };
```

```
const COSTS = {
    electricity: 0.13, // $ per kWh
    water: 0.004, // $ per gallon
    gas: 1.02 // $ per therm
};

const BUILDING_MULTIPLIERS = {
    residential: 1.0,
    commercial: 1.3,
    industrial: 1.8,
    mixed: 1.15
};

// Widget State
let currentStep = 1;
let selectedModules = [];
let calculationResults = {};
let userInputs = {};

// Initialize Widget
function initializeWidget() {
    updateStepIndicator();
    setupModuleSelection();
}

// Step Navigation
function nextStep() {
    if (validateCurrentStep()) {
        if (currentStep === 2) {
            generateDynamicInputs();
```

```
        } else if (currentStep === 3) {
            calculateAndDisplayResults();
        }

        if (currentStep < 4) {
            currentStep++;
            showStep(currentStep);
            updateNavigation();
            updateStepIndicator();
        }
    }
}

function previousStep() {
    if (currentStep > 1) {
        currentStep--;
        showStep(currentStep);
        updateNavigation();
        updateStepIndicator();
    }
}

function showStep(step) {
    document.querySelectorAll('.step').forEach(s => s.classList.remove('active'));
    document.getElementById(`step${step}`).classList.add('active');
}

function updateNavigation() {
    const prevBtn = document.getElementById('prevBtn');
    const nextBtn = document.getElementById('nextBtn');
```

```
    prevBtn.style.display = currentStep > 1 ? 'block' : 'none';

    if (currentStep === 4) {
        nextBtn.style.display = 'none';
    } else {
        nextBtn.style.display = 'block';
        nextBtn.textContent = currentStep === 3 ? 'Calculate Results →' : 'Next →';
    }
}

function updateStepIndicator() {
    document.querySelectorAll('.step-dot').forEach((dot, index) => {
        const stepNumber = index + 1;
        dot.classList.remove('active', 'completed');

        if (stepNumber < currentStep) {
            dot.classList.add('completed');
        } else if (stepNumber === currentStep) {
            dot.classList.add('active');
        }
    });
}

// Module Selection
function setupModuleSelection() {
    document.querySelectorAll('.module-card').forEach(card => {
        card.addEventListener('click', function() {
            const moduleType = this.dataset.module;
            const checkbox = this.querySelector('input[type="checkbox"]');

            checkbox.checked = !checkbox.checked;
```

```
            if (checkbox.checked) {
                this.classList.add('selected');
                if (!selectedModules.includes(moduleType)) {
                    selectedModules.push(moduleType);
                }
            } else {
                this.classList.remove('selected');
                selectedModules = selectedModules.filter(m => m !== moduleType);
            }
        });
    });
}

// Validation
function validateCurrentStep() {
    clearErrors();

    if (currentStep === 1) {
        return validateStep1();
    } else if (currentStep === 2) {
        return validateStep2();
    } else if (currentStep === 3) {
        return validateStep3();
    }
    return true;
}

function validateStep1() {
    let isValid = true;
```

```
    const area = document.getElementById('buildingArea').value;
    const occupants = document.getElementById('occupants').value;
    const buildingType = document.getElementById('buildingType').value;

    if (!area || area <= 0) {
        showError('areaError', 'Please enter a valid building area');
        isValid = false;
    }

    if (!occupants || occupants <= 0) {
        showError('occupantsError', 'Please enter a valid number of occupants');
        isValid = false;
    }

    if (!buildingType) {
        isValid = false;
    }

    if (isValid) {
        userInputs.area = parseFloat(area);
        userInputs.occupants = parseInt(occupants);
        userInputs.buildingType = buildingType;
    }

    return isValid;
}

function validateStep2() {
    if (selectedModules.length === 0) {
        alert('Please select at least one green feature to analyze.');
        return false;
```

```
    }
    return true;
}

function validateStep3() {
    const inputs = document.querySelectorAll('#dynamicInputs input, #dynamicInputs select');
    let isValid = true;

    inputs.forEach(input => {
        if (input.hasAttribute('required') && !input.value) {
            isValid = false;
        }
    });

    if (!isValid) {
        alert('Please fill in all required fields.');
    }

    return isValid;
}

function showError(elementId, message) {
    document.getElementById(elementId).textContent = message;
}

function clearErrors() {
    document.querySelectorAll('.error-message').forEach(elem => {
        elem.textContent = '';
    });
}
```

```
// Dynamic Input Generation
function generateDynamicInputs() {
    const container = document.getElementById('dynamicInputs');
    container.innerHTML = '';

    selectedModules.forEach(module => {
        const moduleDiv = document.createElement('div');
        moduleDiv.className = 'dynamic-inputs';
        moduleDiv.innerHTML = generateModuleInputs(module);
        container.appendChild(moduleDiv);
    });
}

function generateModuleInputs(module) {
    switch (module) {
        case 'electricity':
            return `
                <h4> Solar Electricity</h4>
                <div class="form-row">
                    <div class="form-group">
                        <label for="avgUnits">Average Monthly Units (kWh)</label>
                        <input type="number" id="avgUnits" min="0" step="0.1" placeholder="e.g., 800"
required>
                    </div>
                    <div class="form-group">
                        <label for="billAmount">Monthly Bill Amount ($)</label>
                        <input type="number" id="billAmount" min="0" step="0.01" placeholder="e.g.,
120" required>
                    </div>
                </div>
                <div class="form-group">
                    <label for="solarCapacity">Planned Solar Capacity (kW)</label>
```

```
                    <input type="number" id="solarCapacity" min="0" step="0.1" placeholder="e.g., 5" required>
                </div>
            `;

        case 'water':
            return `
                <h4>💧 Water Conservation</h4>
                <div class="form-row">
                    <div class="form-group">
                        <label for="waterUsage">Monthly Water Usage</label>
                        <select id="waterUsageType" onchange="toggleWaterInputs()">
                            <option value="default">Use default (135 LPCD)</option>
                            <option value="custom">Enter custom usage</option>
                        </select>
                    </div>
                </div>
                <div id="customWaterInputs" style="display: none;">
                    <div class="form-row">
                        <div class="form-group">
                            <label for="monthlyVolume">Monthly Volume (gallons)</label>
                            <input type="number" id="monthlyVolume" min="0" step="1" placeholder="e.g., 3000">
                        </div>
                        <div class="form-group">
                            <label for="waterCost">Monthly Water Cost ($)</label>
                            <input type="number" id="waterCost" min="0" step="0.01" placeholder="e.g., 45">
                        </div>
                    </div>
                </div>
                <div class="form-group">
```

```
                <label for="waterEfficiency">Water Efficiency Improvement (%)</label>
                <input type="number" id="waterEfficiency" min="0" max="100" step="1" placeholder="e.g., 30" required>
              </div>
            `;

        case 'building':
          return `
            <h4>🏗️ Green Building Features</h4>
            <div class="form-row">
              <div class="form-group">
                <label for="aqi">Air Quality Index</label>
                <select id="aqi" required>
                  <option value="">Select AQI level</option>
                  <option value="good">Good (0-50)</option>
                  <option value="moderate">Moderate (51-100)</option>
                  <option value="poor">Poor (101+)</option>
                </select>
              </div>
              <div class="form-group">
                <label for="insulation">Insulation Quality</label>
                <select id="insulation" required>
                  <option value="">Select insulation</option>
                  <option value="excellent">Excellent</option>
                  <option value="good">Good</option>
                  <option value="average">Average</option>
                  <option value="poor">Poor</option>
                </select>
              </div>
            </div>
          `;
```

```
            case 'iot':
                return `
                    <h4>🤖 IoT Smart Controls</h4>
                    <div class="form-group">
                        <label for="energyReduction">Expected Energy Reduction (%)</label>
                        <input type="number" id="energyReduction" min="0" max="100" step="1"
placeholder="e.g., 15" required>
                    </div>
                    <div class="form-group">
                        <label for="iotSystems">IoT Systems to Install</label>
                        <select id="iotSystems" required>
                            <option value="">Select system type</option>
                            <option value="basic">Basic (Thermostat + Lighting)</option>
                            <option value="advanced">Advanced (HVAC + Security + Appliances)</option>
                            <option value="comprehensive">Comprehensive (Full Building
Automation)</option>
                        </select>
                    </div>
                `;

            default:
                return '';
        }
    }

    function toggleWaterInputs() {
        const type = document.getElementById('waterUsageType').value;
        const customInputs = document.getElementById('customWaterInputs');

        if (type === 'custom') {
            customInputs.style.display = 'block';
```

```
        customInputs.querySelectorAll('input').forEach(input => {
            input.required = true;
        });
    } else {
        customInputs.style.display = 'none';
        customInputs.querySelectorAll('input').forEach(input => {
            input.required = false;
            input.value = '';
        });
    }
}

// Calculations
function calculateAndDisplayResults() {
    calculationResults = {};

    selectedModules.forEach(module => {
        calculationResults[module] = calculateModuleSavings(module);
    });

    displayResults();
}

function calculateModuleSavings(module) {
    const buildingMultiplier = BUILDING_MULTIPLIERS[userInputs.buildingType] || 1.0;

    switch (module) {
        case 'electricity':
            return calculateElectricitySavings(buildingMultiplier);
        case 'water':
            return calculateWaterSavings(buildingMultiplier);
```

```
        case 'building':
            return calculateBuildingSavings(buildingMultiplier);
        case 'iot':
            return calculateIoTSavings(buildingMultiplier);
        default:
            return { monthlyCO2: 0, yearlyCO2: 0, monthlyCost: 0, yearlyCost: 0 };
    }
}

function calculateElectricitySavings(multiplier) {
    const avgUnits = parseFloat(document.getElementById('avgUnits').value);
    const solarCapacity = parseFloat(document.getElementById('solarCapacity').value);

    // Estimate solar generation (kWh per month): capacity * 30 days * 5 hours average sun
    const solarGeneration = solarCapacity * 30 * 5;
    const offsetUnits = Math.min(avgUnits, solarGeneration);

    const monthlyCO2 = offsetUnits * EMISSION_FACTORS.electricity * multiplier;
    const monthlyCost = offsetUnits * COSTS.electricity;

    return {
        monthlyCO2: monthlyCO2,
        yearlyCO2: monthlyCO2 * 12,
        monthlyCost: monthlyCost,
        yearlyCost: monthlyCost * 12
    };
}

function calculateWaterSavings(multiplier) {
    let monthlyGallons;
```

```
    if (document.getElementById('waterUsageType').value === 'custom') {
        monthlyGallons = parseFloat(document.getElementById('monthlyVolume').value);
    } else {
        // Default: 135 LPCD * occupants * 30 days, convert to gallons
        monthlyGallons = (135 * userInputs.occupants * 30) / 3.78541;
    }

    const efficiency = parseFloat(document.getElementById('waterEfficiency').value) / 100;
    const savedGallons = monthlyGallons * efficiency;

    const monthlyCO2 = savedGallons * EMISSION_FACTORS.water * multiplier;
    const monthlyCost = savedGallons * COSTS.water;

    return {
        monthlyCO2: monthlyCO2,
        yearlyCO2: monthlyCO2 * 12,
        monthlyCost: monthlyCost,
        yearlyCost: monthlyCost * 12
    };
}

function calculateBuildingSavings(multiplier) {
    const aqi = document.getElementById('aqi').value;
    const insulation = document.getElementById('insulation').value;

    // Baseline energy consumption estimate (kWh per sq ft per month)
    const baselineConsumption = userInputs.area * 1.2;

    // AQI multipliers for air filtration energy savings
    const aqiMultipliers = { good: 0.05, moderate: 0.10, poor: 0.15 };
```

```
    // Insulation multipliers for heating/cooling savings
    const insulationMultipliers = {
        excellent: 0.25,
        good: 0.20,
        average: 0.15,
        poor: 0.10
    };

    const aqiSavings = baselineConsumption * (aqiMultipliers[aqi] || 0);
    const insulationSavings = baselineConsumption * (insulationMultipliers[insulation] || 0);
    const totalSavings = (aqiSavings + insulationSavings) * multiplier;

    const monthlyCO2 = totalSavings * EMISSION_FACTORS.electricity;
    const monthlyCost = totalSavings * COSTS.electricity;

    return {
        monthlyCO2: monthlyCO2,
        yearlyCO2: monthlyCO2 * 12,
        monthlyCost: monthlyCost,
        yearlyCost: monthlyCost * 12
    };
}

function calculateIoTSavings(multiplier) {
    const reductionPercent = parseFloat(document.getElementById('energyReduction').value) /
100;
    const baselineConsumption = userInputs.area * 1.5; // kWh per sq ft per month

    const savedEnergy = baselineConsumption * reductionPercent * multiplier;

    const monthlyCO2 = savedEnergy * EMISSION_FACTORS.electricity;
```

```
    const monthlyCost = savedEnergy * COSTS.electricity;

    return {
        monthlyCO2: monthlyCO2,
        yearlyCO2: monthlyCO2 * 12,
        monthlyCost: monthlyCost,
        yearlyCost: monthlyCost * 12
    };
}

// Results Display
function displayResults() {
    displaySummaryCards();
    displayChart();
    displayComparisonTable();
}

function displaySummaryCards() {
    const container = document.getElementById('summaryCards');
    container.innerHTML = '';

    let totalMonthlyCO2 = 0;
    let totalYearlyCO2 = 0;
    let totalMonthlyCost = 0;
    let totalYearlyCost = 0;

    // Calculate totals
    Object.values(calculationResults).forEach(result => {
        totalMonthlyCO2 += result.monthlyCO2;
        totalYearlyCO2 += result.yearlyCO2;
        totalMonthlyCost += result.monthlyCost;
```

```
    totalYearlyCost += result.yearlyCost;
});

// Create summary cards
const cards = [
    {
        value: `${totalMonthlyCO2.toFixed(1)} kg`,
        label: 'Monthly CO₂ Saved'
    },
    {
        value: `${(totalYearlyCO2 / 1000).toFixed(1)} tons`,
        label: 'Annual CO₂ Saved'
    },
    {
        value: `${totalMonthlyCost.toFixed(0)}`,
        label: 'Monthly Cost Savings'
    },
    {
        value: `${totalYearlyCost.toFixed(0)}`,
        label: 'Annual Cost Savings'
    }
];

cards.forEach(card => {
    const cardElement = document.createElement('div');
    cardElement.className = 'summary-card';
    cardElement.innerHTML = `
        <div class="card-value">${card.value}</div>
        <div class="card-label">${card.label}</div>
    `;
    container.appendChild(cardElement);
```

```
    });
}

function displayChart() {
    const canvas = document.getElementById('savingsChart');
    const ctx = canvas.getContext('2d');

    // Clear canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Prepare data
    const moduleNames = {
        electricity: 'Solar',
        water: 'Water',
        building: 'Building',
        iot: 'IoT'
    };

    const data = selectedModules.map(module => ({
        name: moduleNames[module],
        value: calculationResults[module].monthlyCO2
    }));

    // Draw bar chart
    drawBarChart(ctx, data, canvas.width, canvas.height);
}

function drawBarChart(ctx, data, width, height) {
    const margin = { top: 20, right: 20, bottom: 60, left: 60 };
    const chartWidth = width - margin.left - margin.right;
    const chartHeight = height - margin.top - margin.bottom;
```

```
// Find max value
const maxValue = Math.max(...data.map(d => d.value));

// Colors
const colors = ['#6b8e23', '#4a7c59', '#8fbc8f', '#9acd32'];

// Draw bars
const barWidth = chartWidth / data.length * 0.7;
const barSpacing = chartWidth / data.length * 0.3;

data.forEach((item, index) => {
    const barHeight = (item.value / maxValue) * chartHeight;
    const x = margin.left + index * (barWidth + barSpacing) + barSpacing / 2;
    const y = margin.top + chartHeight - barHeight;

    // Draw bar
    ctx.fillStyle = colors[index % colors.length];
    ctx.fillRect(x, y, barWidth, barHeight);

    // Draw value on top of bar
    ctx.fillStyle = '#4a7c59';
    ctx.font = '12px sans-serif';
    ctx.textAlign = 'center';
    ctx.fillText(`${item.value.toFixed(1)}kg`, x + barWidth / 2, y - 5);

    // Draw label
    ctx.save();
    ctx.translate(x + barWidth / 2, height - 20);
    ctx.fillText(item.name, 0, 0);
    ctx.restore();
```

```
});

// Draw axes
ctx.strokeStyle = '#4a7c59';
ctx.lineWidth = 1;

// Y-axis
ctx.beginPath();
ctx.moveTo(margin.left, margin.top);
ctx.lineTo(margin.left, margin.top + chartHeight);
ctx.stroke();

// X-axis
ctx.beginPath();
ctx.moveTo(margin.left, margin.top + chartHeight);
ctx.lineTo(margin.left + chartWidth, margin.top + chartHeight);
ctx.stroke();

// Y-axis labels
ctx.fillStyle = '#4a7c59';
ctx.font = '10px sans-serif';
ctx.textAlign = 'right';

for (let i = 0; i <= 5; i++) {
    const value = (maxValue / 5) * i;
    const y = margin.top + chartHeight - (i / 5) * chartHeight;
    ctx.fillText(`${value.toFixed(0)}`, margin.left - 10, y + 3);

    // Grid lines
    if (i > 0) {
        ctx.strokeStyle = '#e0e0e0';
```

```
            ctx.lineWidth = 0.5;
            ctx.beginPath();
            ctx.moveTo(margin.left, y);
            ctx.lineTo(margin.left + chartWidth, y);
            ctx.stroke();
        }
    }
}

function displayComparisonTable() {
    const tbody = document.getElementById('comparisonTableBody');
    tbody.innerHTML = '';

    // Calculate totals
    let totalCO2Savings = 0;
    let totalCostSavings = 0;

    Object.values(calculationResults).forEach(result => {
        totalCO2Savings += result.yearlyCO2;
        totalCostSavings += result.yearlyCost;
    });

    // Estimate conventional values
    const conventionalCO2 = totalCO2Savings * 2; // Assume sustainable is 50% improvement
    const conventionalCost = totalCostSavings * 1.5; // Assume 50% cost savings

    const rows = [
        {
            metric: 'Annual CO₂ Emissions',
            conventional: `${(conventionalCO2 / 1000).toFixed(1)} tons`,
            sustainable: `${((conventionalCO2 - totalCO2Savings) / 1000).toFixed(1)} tons`,
```

```
            savings: `${(totalCO2Savings / 1000).toFixed(1)} tons`
        },
        {
            metric: 'Annual Energy Costs',
            conventional: `${(conventionalCost + totalCostSavings).toFixed(0)}`,
            sustainable: `${conventionalCost.toFixed(0)}`,
            savings: `${totalCostSavings.toFixed(0)}`
        }
    ];

    rows.forEach(row => {
        const tr = document.createElement('tr');
        tr.innerHTML = `
            <td>${row.metric}</td>
            <td>${row.conventional}</td>
            <td>${row.sustainable}</td>
            <td style="color: #4a7c59; font-weight: 600;">${row.savings}</td>
        `;
        tbody.appendChild(tr);
    });
}

// Report Generation
function downloadReport() {
    const reportContent = generateReportHTML();

    // Create a new window for printing
    const printWindow = window.open('', '_blank');
    printWindow.document.write(reportContent);
    printWindow.document.close();
```

```
    // Trigger print dialog
    printWindow.onload = function() {
        printWindow.print();
    };
}

function generateReportHTML() {
    let totalMonthlyCO2 = 0;
    let totalYearlyCO2 = 0;
    let totalMonthlyCost = 0;
    let totalYearlyCost = 0;

    Object.values(calculationResults).forEach(result => {
        totalMonthlyCO2 += result.monthlyCO2;
        totalYearlyCO2 += result.yearlyCO2;
        totalMonthlyCost += result.monthlyCost;
        totalYearlyCost += result.yearlyCost;
    });

    const moduleDetails = selectedModules.map(module => {
        const result = calculationResults[module];
        const names = {
            electricity: 'Solar Electricity',
            water: 'Water Conservation',
            building: 'Green Building',
            iot: 'IoT Controls'
        };

        return `
            <tr>
                <td>${names[module]}</td>
```

```
            <td>${result.monthlyCO2.toFixed(1)} kg</td>
            <td>${(result.yearlyCO2 / 1000).toFixed(1)} tons</td>
            <td>${result.monthlyCost.toFixed(0)}</td>
            <td>${result.yearlyCost.toFixed(0)}</td>
          </tr>
        `;
      }).join('');

      return `
        <!DOCTYPE html>
        <html>
        <head>
          <title>Green-Builder Carbon Savings Report</title>
          <style>
            body { font-family: Arial, sans-serif; margin: 40px; color: #333; }
            .header { text-align: center; margin-bottom: 40px; }
            .header h1 { color: #4a7c59; margin: 0; }
            .header p { color: #6b8e23; margin: 10px 0 0 0; }
            .section { margin-bottom: 30px; }
            .section h2 { color: #4a7c59; border-bottom: 2px solid #d4d4aa; padding-bottom: 5px; }
            table { width: 100%; border-collapse: collapse; margin-top: 15px; }
            th, td { padding: 12px; text-align: left; border-bottom: 1px solid #ddd; }
            th { background-color: #f0f8f0; color: #4a7c59; font-weight: 600; }
            .summary-box { background: #f8fdf8; padding: 20px; border-radius: 8px; border: 1px
solid #d4d4aa; }
            .highlight { color: #4a7c59; font-weight: 600; font-size: 18px; }
            .date { text-align: right; color: #6b8e23; font-size: 12px; margin-top: 20px; }
            @media print {
              body { margin: 20px; }
              .no-print { display: none; }
            }
```

```
    </style>
  </head>
  <body>
    <div class="header">
      <h1>🌱 Green-Builder Carbon Savings Report</h1>
      <p>Environmental Impact Analysis for ${userInputs.buildingType} Building</p>
    </div>

    <div class="section">
      <h2>Building Profile</h2>
      <table>
        <tr><td><strong>Building Area:</strong></td><td>${userInputs.area.toLocaleString()} sq ft</td></tr>
        <tr><td><strong>Building Type:</strong></td><td>${userInputs.buildingType.charAt(0).toUpperCase() + userInputs.buildingType.slice(1)}</td></tr>
        <tr><td><strong>Occupants:</strong></td><td>${userInputs.occupants}</td></tr>
        <tr><td><strong>Green Features:</strong></td><td>${selectedModules.length} modules analyzed</td></tr>
      </table>
    </div>

    <div class="section">
      <h2>Executive Summary</h2>
      <div class="summary-box">
        <p>By implementing the selected green building features, this ${userInputs.buildingType} building can achieve:</p>
        <ul>
          <li class="highlight">${(totalYearlyCO2 / 1000).toFixed(1)} tons of CO₂ emissions avoided annually</li>
          <li class="highlight">${totalYearlyCost.toFixed(0)} in annual cost savings</li>
          <li class="highlight">${totalMonthlyCO2.toFixed(1)} kg of CO₂ reduced monthly</li>
```

```
      </ul>
    </div>
  </div>

  <div class="section">
    <h2>Detailed Savings by Feature</h2>
    <table>
      <thead>
        <tr>
          <th>Green Feature</th>
          <th>Monthly CO₂ Saved</th>
          <th>Annual CO₂ Saved</th>
          <th>Monthly Cost Savings</th>
          <th>Annual Cost Savings</th>
        </tr>
      </thead>
      <tbody>
        ${moduleDetails}
        <tr style="background: #f0f8f0; font-weight: 600;">
          <td><strong>Total</strong></td>
          <td><strong>${totalMonthlyCO2.toFixed(1)} kg</strong></td>
          <td><strong>${(totalYearlyCO2 / 1000).toFixed(1)} tons</strong></td>
          <td><strong>${totalMonthlyCost.toFixed(0)}</strong></td>
          <td><strong>${totalYearlyCost.toFixed(0)}</strong></td>
        </tr>
      </tbody>
    </table>
  </div>

  <div class="section">
    <h2>Assumptions & Methodology</h2>
```

```
                <ul>
                    <li>CO₂ emission factors: Electricity ${EMISSION_FACTORS.electricity} kg CO₂e/kWh, Water ${EMISSION_FACTORS.water} kg CO₂e/gallon</li>
                    <li>Energy costs: Electricity ${COSTS.electricity}/kWh, Water ${COSTS.water}/gallon</li>
                    <li>Building type multiplier: ${BUILDING_MULTIPLIERS[userInputs.buildingType]}x for ${userInputs.buildingType} buildings</li>
                    <li>Calculations based on industry-standard emission factors and regional averages</li>
                    <li>Results are estimates for planning purposes; actual savings may vary</li>
                </ul>
            </div>

            <div class="date">
                Report generated on ${new Date().toLocaleDateString()} | Green-Builder Carbon Calculator v2.0
            </div>
        </body>
        </html>
    `;
}

// Reset Function
function resetWidget() {
    if (confirm('Are you sure you want to start over? All data will be lost.')) {
        currentStep = 1;
        selectedModules = [];
        calculationResults = {};
        userInputs = {};

        // Clear all inputs
        document.querySelectorAll('input, select').forEach(input => {
            if (input.type === 'checkbox') {
```

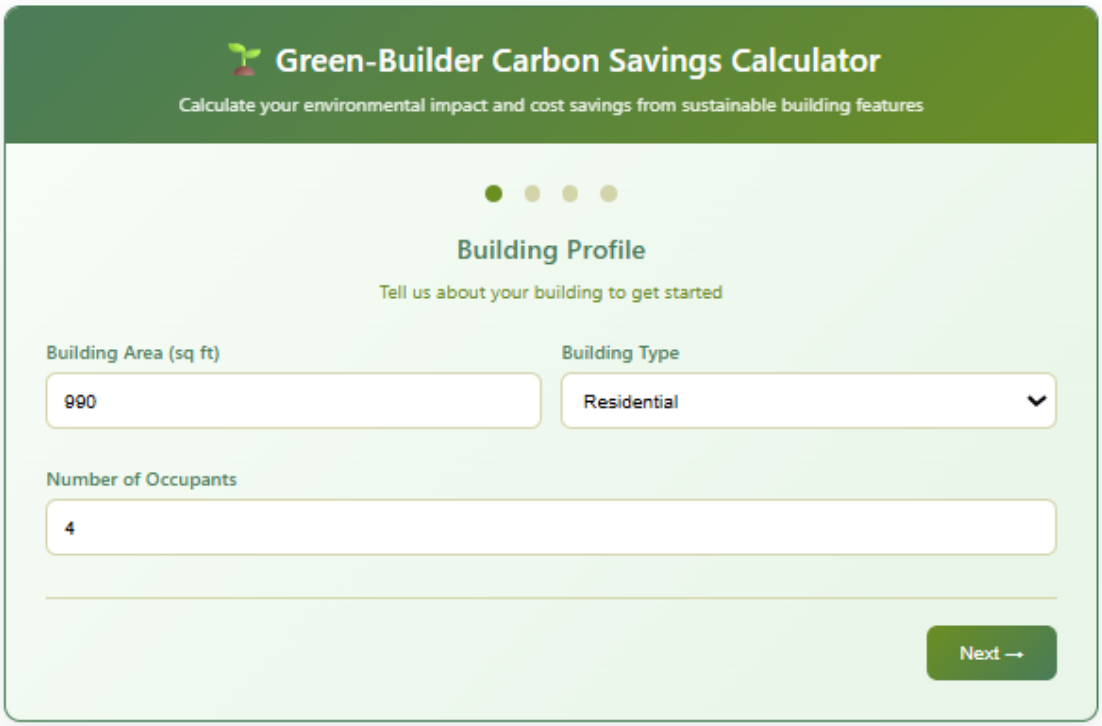
```
                input.checked = false;
            } else {
                input.value = '';
            }
        });

        // Reset module cards
        document.querySelectorAll('.module-card').forEach(card => {
            card.classList.remove('selected');
        });

        // Show first step
        showStep(1);
        updateNavigation();
        updateStepIndicator();
        clearErrors();
    }
}

// Initialize the widget when page loads
document.addEventListener('DOMContentLoaded', function() {
    initializeWidget();
});
</script>
</body>
</html>
```

Output -


Green-Builder Carbon Savings Calculator
Calculate your environmental impact and cost savings from sustainable building features
Building Profile
Tell us about your building to get started
Building Area (sq ft)
990
Building Type
Residential
Number of Occupants
4
Next →


Green-Builder Carbon Savings Calculator
Calculate your environmental impact and cost savings from sustainable building features
Select Green Features
Choose which sustainable features you want to analyze
Solar Electricity
Water Conservation
Green Building
IoT Controls
← Previous
Next →

Green-Builder Carbon Savings Calculator

Calculate your environmental impact and cost savings from sustainable building features

Feature Details

Provide specific information for your selected features

Solar Electricity

Average Monthly Units (kWh)

350

Monthly Bill Amount ($)

3500

Planned Solar Capacity (kW)

3

Water Conservation

Monthly Water Usage

Use default (135 LPCD)

Water Efficiency Improvement (%)

50

Green Building Features

Air Quality Index

Moderate (51-100)

Insulation Quality

Average

← Previous

Calculate Results →

Green-Builder Carbon Savings Calculator

Calculate your environmental impact and cost savings from sustainable building features

Your Carbon Savings Report

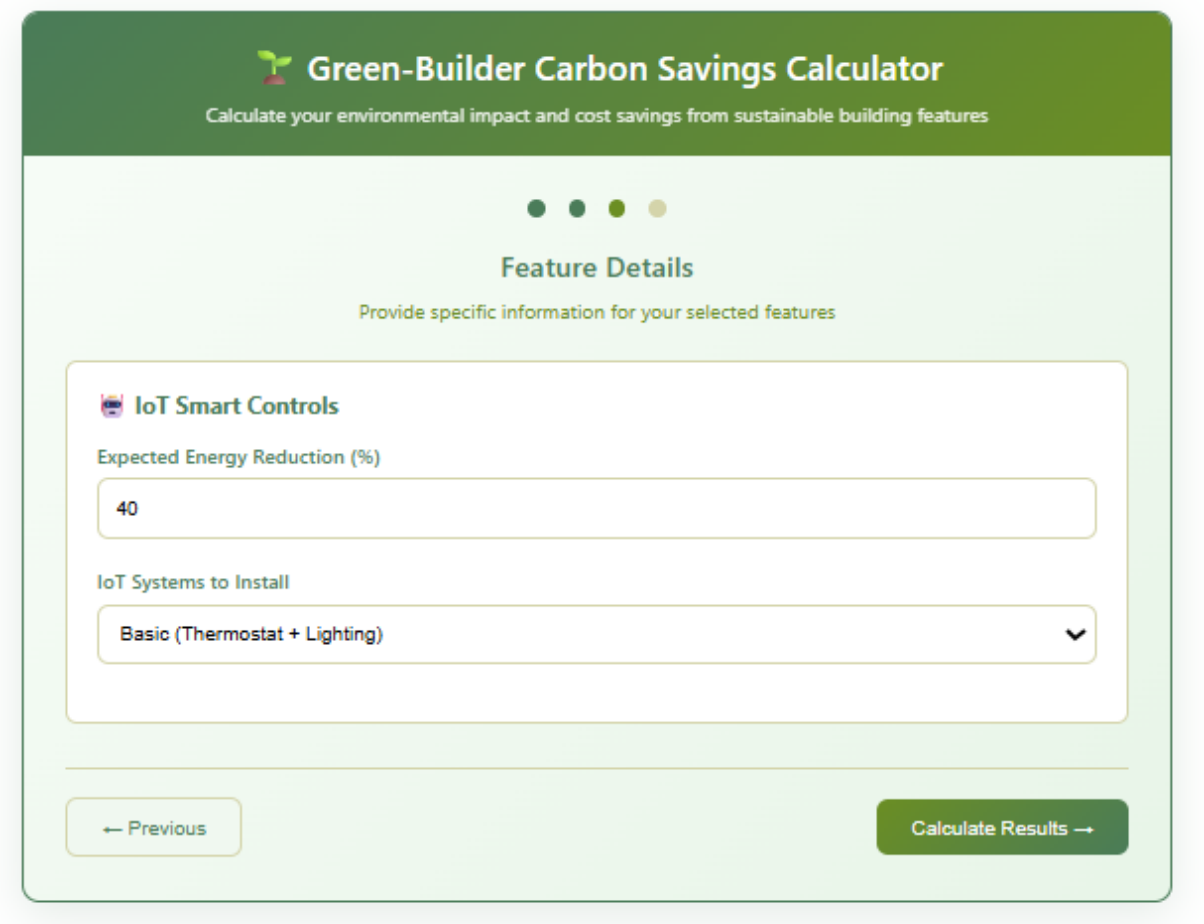
546.5 kg	6.6 tons	77	927
MONTHLY CO_2 SAVED	ANNUAL CO_2 SAVED	MONTHLY COST SAVINGS	ANNUAL COST SAVINGS

Monthly Carbon Savings by Feature

Metric	Conventional	Sustainable	Savings
Annual CO_2 Emissions	13.1 tons	6.6 tons	6.6 tons
Annual Energy Costs	2317	1390	927

Download Report | Start Over

← Previous



Next Task PRD : -

Product Requirements Document (AI-Driven Implementation v2.1)

Project: Green-Builder Carbon Savings Widget

Version: 2.1 | Date: July 28, 2025

1. Mission & Vision

Mission: AI will generate a single embed-ready HTML/CSS/JS widget that estimates operational and embodied carbon savings by analyzing key green features—Electricity, Water, Green Building Profile, IoT Controls, Routine Emissions, and Material Green Score—without exposing complex inputs to end users.

2. Overall AI Tasks & Boundaries

What AI Must Do:

Generate Widget Structure: modular code snippets (HTML container, CSS styles, JS logic) that assemble into a complete widget.

Implement Input Flow: a wizard with:

Select Green Features: Electricity, Water, Green Building Profile, IoT Controls, Routine Emissions Calculator, Material Green Score Checker.

Conditional Q&A: when “Green Building Profile” is selected, embed Building Profile questions inside that step.

Customize Green Building Profile:

If Residential Individual: ask for plot size (sq m/yd) and number of floors.

If Flat: ask for apartment size and floor number.

Default Water Efficiency:

Remove user input for water efficiency.

Use defaults: Residential 40%, Commercial 25%, Industrial 60%, Mixed Use 35%.

In report, state: “Average water efficiency improvement: X%”.

Maintenance Cost Section:

Ask for House Maintenance Cost (crack fixing, repairs): slider 0–₹10 Lakh+ per year.

Ask for Society Maintenance Cost: separate slider, same range.

Clearly label each cost type with helper text.

Remove All Pre-Existing Green Building Inputs: AI must discard any previous form fields under Green Building and replace with the above.

Add New Features:

Routine Emissions Calculator: integrate Q&A from reference document’s “Carbon Emissions Calculator” section—questions to quantify daily routines (e.g. driving, cooking, appliances).

Material Green Score Checker: integrate Q&A from reference document’s “Material Green Score Checker” section to assess product/material sustainability.

Scoring & Dashboard:

After all inputs, compute per-feature CO_2 savings and material green scores.

Provide a feature-level score and overall sustainability index on dashboard.

Stepwise AI Guidance:

At each code snippet, include instructions: “Paste this into previous code’s <head>”, “Append to widget JS” etc.

Annotate “What to do next” and “What not to do” clearly.

What AI Must Not Do:

Reference WordPress or server-side integrations.

Ask users for complex uploads or additional modules outside the specified features.

Include heavy external dependencies; keep to vanilla JS/CSS/HTML.

3. Stepwise Implementation Plan for AI

Step A: HTML Container

Generate and include the <script> tag reference.Next: AI will provide CSS.

Step B: CSS Styles

Provide scoped CSS for widget (colors, layout, responsive), avoiding global selectors.Next: AI will supply base JS to initialize wizard.

Step C: Base JavaScript Skeleton

Create initWidget(), step management, constants for emission factors, water defaults, maintenance cost ranges.

Outline feature list and placeholders for each Q&A module.Next: AI will code “Select Green Features” UI and event handlers.

Step D: Feature Selection UI

Render checkboxes/cards for seven features:

Electricity Module

Water Module

Green Building Profile

Building Green Score Checker

IoT Controls Module

Routine Emissions Calculator

Material Green Score Checker

On selection, mark the feature in the widget state; on deselection, clear related data.
Next: AI will implement conditional Q&A for “Green Building Profile” and “Building Green Score Checker.”

Step E: Green Building Profile & Building Green Score Checker Q&A

Green Building Profile (when selected):

Building Type: Residential Individual, Flat, Commercial, Industrial, Mixed.

If Residential Individual: ask for plot size (sq m or sq yd) and number of floors.

If Flat: ask for apartment size (sq m or sq yd) and floor number.

Building Green Score Checker (when selected): present the 10 yes/no questions from the reference document to compute a 0–10 green score:

Energy Efficiency: Does your home/building have energy-saving features like LED lights, good insulation, or energy-efficient appliances?

Renewable Energy: Do you use renewable energy sources (solar panels, solar water heater, wind, etc.)?

Water Conservation: Do you have water-saving fixtures (low-flow taps, dual-flush toilets, rainwater harvesting)?

Waste Management: Do you recycle or compost home/building waste?

Healthy Indoor Air: Do you use low-VOC paints and non-toxic materials indoors?

Green Materials: Was your home/building built or renovated using eco-friendly or recycled materials?

Smart Controls: Do you use smart thermostats, sensors, or automation to reduce energy use?

Local Sourcing: Were most building materials sourced locally (within 200 km)?

Green Landscaping: Does your property have native plants, green roofs, or permeable paving?

Transport & Community: Is your home/building close to public transport, bike paths, or walkable services?

Each “Yes” = 1 point; sum to a 0–10 score. In the report, interpret score as: 9–10: Excellent; 7–8: Good; 5–6: Moderate; 3–4: Low; 0–2: Poor.
Next: AI will integrate default water efficiency logic in Step F.

Step F: Water Module Defaults Green Building Profile Q&A**

When selected, inject sub-step:

Type: Individual Residential vs. Flat vs. Commercial vs. Industrial vs. Mixed.

If Residential Individual: ask plot size, number of floors.

If Flat: ask size, floor number.

Remove any old AQI/insulation questions.Next: AI will integrate default water efficiency logic.

Step F: Water Module Defaults

Omit user prompts for water efficiency.

Assign efficiency based on building type.

Display read-only efficiency % in summary.Next: AI will build Maintenance Cost sliders.

Step G: Maintenance Cost Inputs

Two sliders with labels: House Maintenance, Society Maintenance; 0–1,000,000 INR with “+” label for above max.

Store values for reporting.Next: AI will add Routine Emissions Calculator Q&A (see reference file).

Step H: Routine Emissions Q&A

Parse questions from reference doc’s “Carbon Emissions Calculator” section.

Dynamically inject text fields or selects as per doc.Next: AI will add Material Green Score Checker Q&A.

Step I: Material Green Score Checker

Parse questions from reference doc’s “Material Green Score Checker” section.

Create inputs and compute a green score (e.g. 0–100).Next: AI will code calculation engine.

Step J: Calculation Engine

For each feature:

Electricity, Water, Green Profile, IoT, Routine Emissions, Material Score.

Summarize CO_2 saved, cost savings, green score.

Compute overall Sustainability Index.Next: AI will render dashboard elements.

Step K: Dashboard & Reporting

Render summary cards and SVG/Canvas charts.

Add sustainability index gauge.

Generate printable report via window.print().Next: AI will include error handling and final integration instructions.

Step L: Error Handling & Integration Notes

Provide validations and fallback defaults.

Clearly document how to continue in next chat if truncated.

4. References

QnA Tools Document: Contains detailed questions for Routine Emissions Calculator and Material Green Score Checker

End of PRD (AI-Driven v2.1)