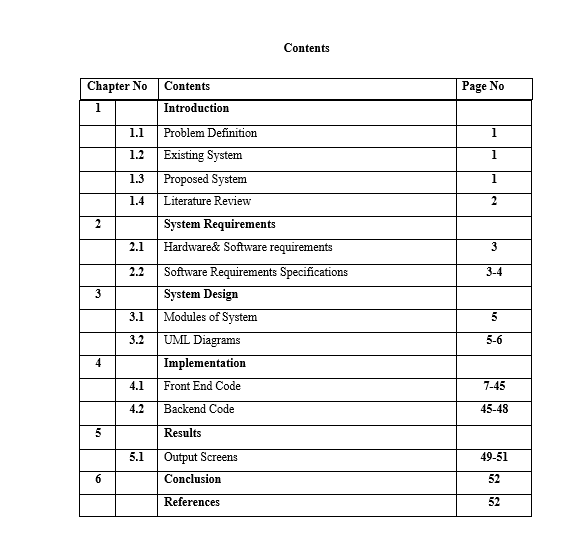


****

**1. Introduction**

In today’s world, where energy resources are becoming increasingly scarce and environmental sustainability is a growing concern, efficient energy management has become a critical priority. Educational institutions, such as colleges, are significant consumers of electricity due to the large number of rooms, appliances, and equipment in use daily. However, the lack of proper monitoring and control mechanisms often leads to unnecessary energy wastage, resulting in higher operational costs and a larger carbon footprint. This chapter introduces the context and necessity of implementing a system to monitor and optimize electricity consumption within a college block.

**1.1 Problem Definition:**

Electricity wastage is a prevalent issue in many college buildings, primarily due to the uncontrolled and unmonitored use of electrical appliances such as lights, fans, air conditioners, and computers. Without a system to track energy consumption, it becomes challenging to identify areas where energy is being used inefficiently. For instance, lights and fans may remain switched on in unoccupied rooms, or high-energy-consuming appliances may be used unnecessarily. This lack of awareness and control leads to significant energy wastage, increased electricity bills, and unnecessary strain on the power grid. The problem is further exacerbated by the absence of real-time data, which could otherwise help in making informed decisions to optimize energy usage.

**1.2 Existing System:**

Currently, the college block operates without any system to monitor or track electricity consumption at the room level. The absence of such a system means that energy usage is neither measured nor analysed. As a result, there is no visibility into which rooms or appliances are consuming the most energy, making it impossible to implement targeted energy-saving measures. The existing setup relies solely on manual observation and ad hoc interventions, which are neither efficient nor scalable. This lack of a structured approach to energy management has led to inefficiencies, higher energy costs, and a missed opportunity to contribute to environmental sustainability.

**1.3 Proposed System:**

A software-based system that simulates energy consumption data for each room based on predefined energy consumption patterns for various appliances (e.g., lights, fans, computers, etc.). The system will provide insights into energy usage for each room and allow the college management to identify areas of improvement.

In summary, this project aims to bridge the gap between energy consumption and energy management by introducing a simulation-based monitoring system. By leveraging technology to track and analyze electricity usage, the college can take a significant step toward reducing its energy footprint, lowering operational costs, and promoting a culture of sustainability.

**1.4 Literature Review:**

This section reviews existing technologies and systems for energy monitoring, such as power meters, smart meters, and software tools that track energy consumption in real-time. It discusses previous works in energy management systems, particularly focusing on those that do not rely on sensors or automation.

**2. System Requirements**

**2.1 Hardware & Software Requirements:**

**Hardware:**

* Smart energy meters or power monitoring devices for each room (to measure electricity usage).
* Microcontroller (e.g., Raspberry Pi, Arduino) for data collection.
* Communication modules (e.g., Wi-Fi, Ethernet) to transmit data to the server or monitoring system.
* Power supply and other supporting electronic components.

**Software:**

**Front-End Technologies:**

* **HTML** for the structure of the web pages.
* **CSS** for styling and making the pages responsive.
* **JavaScript** for interactivity and visualizing energy consumption data.

**Back-End Technologies:**

* MongoDB
* NodeJS
* ReactJS

**Database:**

* **SQL** for lightweight data storage.

**2.2 Software Requirements Specification (SRS):**

**Functional Requirements:**

The system must be capable of tracking the energy consumption for each room within the college. This tracking involves simulating the usage of common appliances such as lights, fans, and air conditioners, and calculating the energy consumed in kilowatt-hours (kWh). For each room, the energy consumption data will be calculated dynamically based on predefined parameters such as the average usage of appliances over time. The system will then provide a way to visualize the energy consumption data through graphical representations such as bar charts or line graphs. This will be done using Chart.js, a powerful JavaScript library that allows for easy rendering of interactive and visually appealing graphs.

**Non-Functional Requirements:**

The system should prioritize an easy-to-use interface that ensures administrators or users can interact with the system efficiently. This means the design should be intuitive, with clear navigation and a clean, responsive layout that adapts to different screen sizes. The system should also allow users to easily access real-time data as well as historical energy usage data for each room. In addition to usability, the system must be **scalable** to handle a large number of rooms, users, and data points. As the number of rooms in the college increases or if the system is deployed in larger institutions, the system should be able to process and display energy usage data without performance degradation.

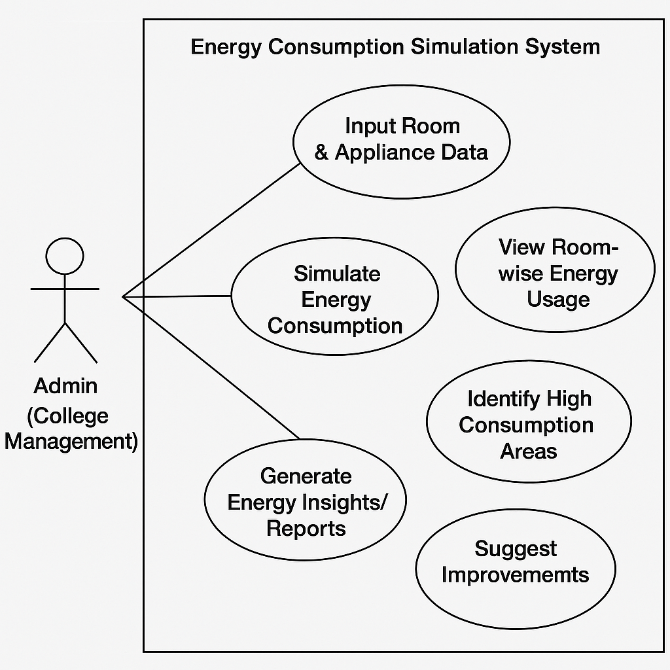
**3. System Design**

**3.1 Modules of the System:**

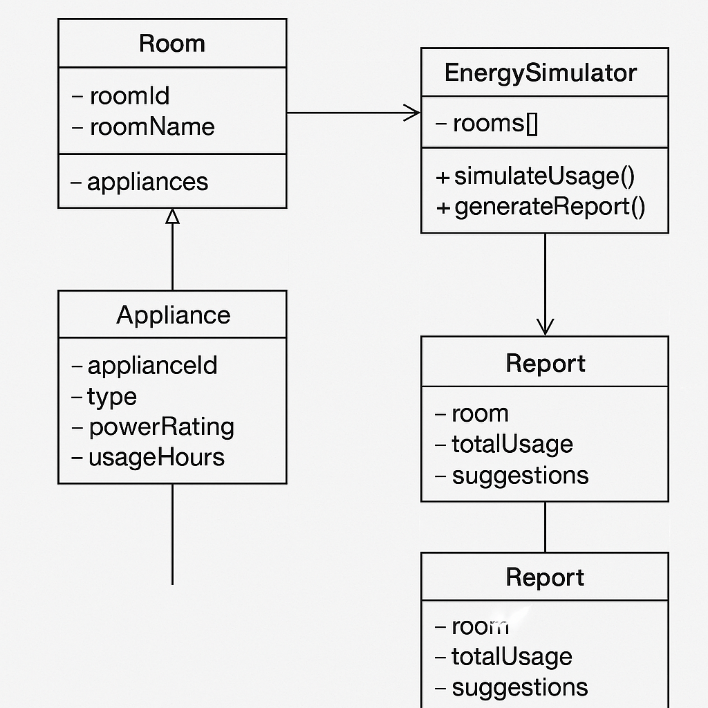
The system tracks and visualizes energy consumption in a college by simulating the usage of appliances like lights, fans, and air conditioners. The Energy Consumption Simulation Module generates dynamic energy usage data for each room based on predefined parameters. This data is stored in a Database Module (SQLite) for easy retrieval. The User Interface Module uses HTML, CSS, and JavaScript to display real-time data and historical trends, with Chart.js visualizing the data in graphs. The Reporting Module generates detailed energy consumption reports for each room over specified time periods. The system, built with NodeJS & ReactJS, handles back-end calculations and data management, while the front-end provides an interactive interface for users to view and analyse energy usage.

**3.2 UML Diagrams:**

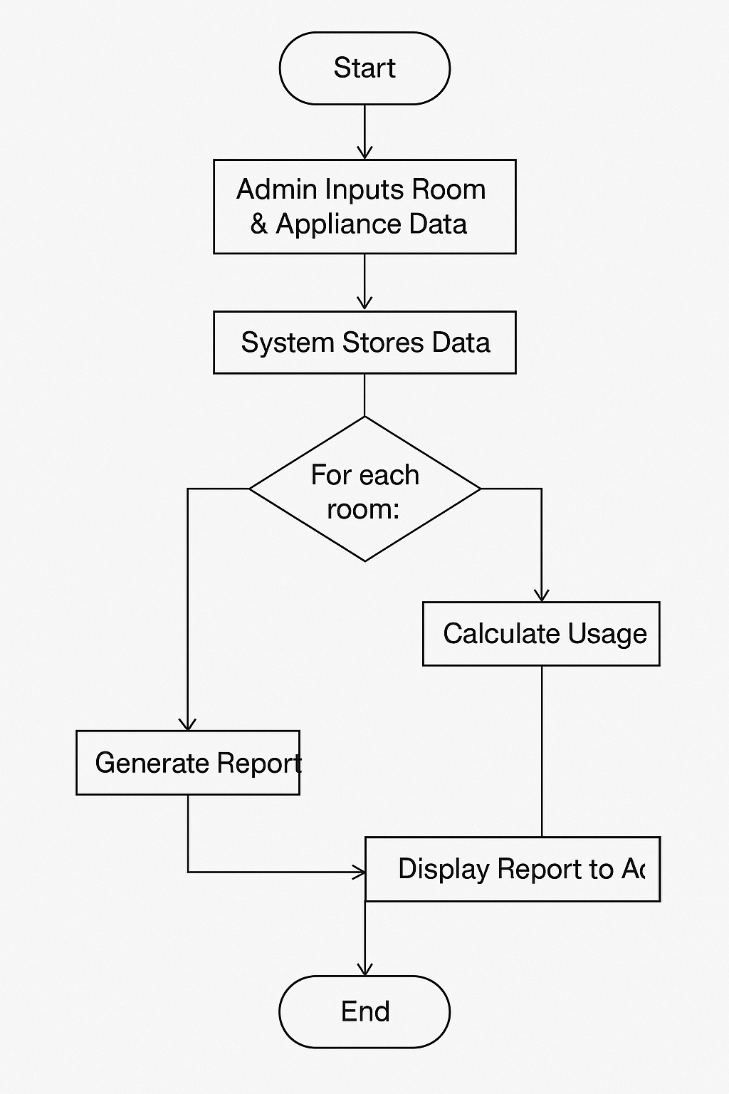
**3.2.1 Class Diagram**



**3.2.2 Activity Diagram** – Energy Usage Simulation Flow



**3.2.3 Component Diagram**



**4. Implementation**

**4.1 Front-end Code:**

**Login Page**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Animated Electricity Login Page</title>

<style>

/\* (Your existing CSS remains unchanged) \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

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

background-color: #000; /\* Dark background for electricity effect \*/

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

overflow: hidden;

position: relative;

}

.electricity-background {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: radial-gradient(circle, rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 1));

z-index: -1;

overflow: hidden;

}

.electricity-background::before,

.electricity-background::after {

content: '';

position: absolute;

width: 100%;

height: 100%;

background: linear-gradient(45deg, #00ffcc, #ff00cc, #00ffcc, #ff00cc);

background-size: 400% 400%;

animation: electricFlow 10s linear infinite;

opacity: 0.3;

z-index: -1;

}

.electricity-background::after {

animation-delay: -5s;

opacity: 0.2;

}

@keyframes electricFlow {

0% { background-position: 0% 50%; }

50% { background-position: 100% 50%; }

100% { background-position: 0% 50%; }

}

.container {

width: 100%;

max-width: 400px;

padding: 20px;

background-color: rgba(0, 0, 0, 0.7); /\* Semi-transparent dark background \*/

border-radius: 12px;

box-shadow: 0 8px 16px rgba(0, 255, 204, 0.3);

animation: fadeIn 1s ease-in-out;

position: relative;

overflow: hidden;

border: 2px solid #00ffcc;

}

@keyframes fadeIn {

from { opacity: 0; transform: scale(0.9); }

to { opacity: 1; transform: scale(1); }

}

.form-container {

display: none;

animation: slideIn 0.5s ease-in-out;

opacity: 0;

transform: translateY(20px);

}

@keyframes slideIn {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.form-container.active {

display: block;

opacity: 1;

transform: translateY(0);

}

.form-container h2 {

text-align: center;

margin-bottom: 20px;

color: #00ffcc; /\* Electric cyan color \*/

font-size: 24px;

font-weight: bold;

animation: textPop 0.5s ease-in-out;

@keyframes textPop {

0% { transform: scale(0.9); opacity: 0; }

100% { transform: scale(1); opacity: 1; }

}

.input-group {

margin-bottom: 20px;

position: relative;

animation: fadeInInput 0.5s ease-in-out;

}

@keyframes fadeInInput {

from { opacity: 0; transform: translateX(-20px); }

to { opacity: 1; transform: translateX(0); }

}

.input-group label {

font-size: 14px;

color: #00ffcc; /\* Electric cyan color \*/

display: block;

margin-bottom: 5px;

font-weight: bold;

}

.input-group input {

width: 100%;

padding: 12px;

margin-top: 5px;

border: 2px solid #00ffcc;

border-radius: 6px;

font-size: 16px;

background-color: rgba(0, 0, 0, 0.5);

color: #00ffcc;

transition: border-color 0.3s ease-in-out, box-shadow 0.3s ease-in-out;

}

.input-group input:focus {

border-color: #000dff; /\* Electric pink color \*/

box-shadow: 0 0 8px rgba(166, 0, 255, 0.3);

outline: none;

}

.btn {

width: 100%;

padding: 14px;

background-color: #00ffcc; /\* Electric cyan color \*/

color: #000;

font-size: 16px;

border: none;

border-radius: 6px;

cursor: pointer;

transition: background 0.3s ease-in-out, transform 0.2s ease-in-out;

}

.btn:hover {

background-color: #ff00cc; /\* Electric pink color \*/

transform: translateY(-2px);

}

.btn:active {

transform: translateY(0);

}

.error-message {

color: #ff4444;

font-size: 14px;

text-align: center;

margin-top: 10px;

animation: shake 0.5s ease-in-out;

}

@keyframes shake {

0%, 100% { transform: translateX(0); }

25% { transform: translateX(-10px); }

50% { transform: translateX(10px); }

75% { transform: translateX(-10px); }

}

.toggle-link {

text-align: center;

margin-top: 15px;

font-size: 14px;

color: #00ffcc; /\* Electric cyan color \*/

}

.toggle-link a {

color: #ff00cc; /\* Electric pink color \*/

cursor: pointer;

text-decoration: none;

font-weight: bold;

transition: color 0.3s ease-in-out;

}

.toggle-link a:hover {

color: #00ffcc; /\* Electric cyan color \*/

}

</style>

</style>

</head>

<body>

<!-- Electricity Background -->

<div class="electricity-background"></div>

<!-- Login & Signup Forms -->

<div class="container">

<div id="signup-form" class="form-container">

<h2>Signup</h2>

<form id="signup-form-submit">

<div class="input-group">

<label for="signup-email">Email</label>

<input type="email" id="signup-email" required>

</div>

<div class="input-group">

<label for="signup-username">Username</label>

<input type="text" id="signup-username" required>

</div>

<div class="input-group">

<label for="signup-password">Password</label>

<input type="password" id="signup-password" required>

</div>

<button type="submit" class="btn">Sign Up</button>

<p id="signup-error-message" class="error-message"></p>

</form>

<p class="toggle-link">Already have an account? <a onclick="toggleForms('login')">Login</a></p>

</div>

<div id="login-form" class="form-container active">

<h2>Login</h2>

<form id="login-form-submit">

<div class="input-group">

<label for="login-email">Email</label>

<input type="email" id="login-email" required>

</div>

<div class="input-group">

<label for="login-username">Username</label>

<input type="text" id="login-username" required>

</div>

<div class="input-group">

<label for="login-password">Password</label>

<input type="password" id="login-password" required>

</div>

<button type="submit" class="btn">Login</button>

<p id="login-error-message" class="error-message"></p>

</form>

<p class="toggle-link">Don't have an account? <a onclick="toggleForms('signup')">Sign Up</a></p>

</div>

</div>

<script>

document.addEventListener("DOMContentLoaded", function () {

function toggleForms(formType) {

document.getElementById('signup-form').classList.toggle('active', formType === 'signup');

document.getElementById('login-form').classList.toggle('active', formType === 'login');

}

document.getElementById('signup-form-submit').addEventListener('submit', async function (e) {

e.preventDefault();

const email = document.getElementById('signup-email').value;

const username = document.getElementById('signup-username').value;

const password = document.getElementById('signup-password').value;

try {

const response = await fetch('http://localhost:3000/signup', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ email, username, password })

});

const result = await response.json();

if (response.ok) {

alert("Signup successful! You can now login.");

toggleForms('login');

} else {

document.getElementById('signup-error-message').textContent = result.message || "Signup failed.";

}

} catch (err) {

document.getElementById('signup-error-message').textContent = "Error connecting to server.";

}

});

document.getElementById('login-form-submit').addEventListener('submit', async function (e) {

e.preventDefault();

const email = document.getElementById('login-email').value;

const username = document.getElementById('login-username').value;

const password = document.getElementById('login-password').value;

try {

const response = await fetch('http://localhost:3000/login', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ email, username, password })

});

const result = await response.json();

if (response.ok) {

alert("Login successful!");

window.location.href = "Extend1.html";

} else {

document.getElementById('login-error-message').textContent = result.message || "Invalid credentials.";

}

} catch (err) {

document.getElementById('login-error-message').textContent = "Error connecting to server."; }});});

</script>

</body>

</html>

**Bulidings.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Building Selection - Electricity Theme</title>

    <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet">

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

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

            background-color: #000;

            color: #00ffcc;

            text-align: center;

            overflow-x: hidden;

        }

        .electricity-background {

            position: fixed;

            top: 0;

            left: 0;

            width: 100%;

            height: 100%;

            background: radial-gradient(circle, rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 1));

            z-index: -1;

        }

        .electricity-background::before {

            content: '';

            position: absolute;

            width: 100%;

            height: 100%;

            background: linear-gradient(45deg, #00ffcc, #ff00cc, #00ffcc, #ff00cc);

            background-size: 400% 400%;

            animation: electricFlow 10s linear infinite;

            opacity: 0.2;

        }

        @keyframes electricFlow {

            0% { background-position: 0% 50%; }

            50% { background-position: 100% 50%; }

            100% { background-position: 0% 50%; }

        }

        nav {

            background-color: rgba(0, 0, 0, 0.9);

            padding: 15px;

            text-align: center;

            box-shadow: 0 4px 6px rgba(0, 255, 204, 0.3);

            position: sticky;

            top: 0;

            z-index: 1000;

        }

        nav a {

            color: #00ffcc;

            text-decoration: none;

            margin: 0 20px;

            font-size: 18px;

            transition: color 0.3s ease, transform 0.3s ease;

        }

        nav a:hover {

            color: #ff00cc;

            transform: translateY(-3px);

        }

        .login-btn {

            background-color: #00ffcc;

            color: #000;

            padding: 10px 20px;

            border-radius: 25px;

            transition: background-color 0.3s ease, transform 0.3s ease;

        }

        .login-btn:hover {

            background-color: #ff00cc;

            transform: scale(1.05);

        }

        .container {

            max-width: 800px;

            margin: 50px auto;

            background: rgba(0, 0, 0, 0.7);

            padding: 30px;

            border-radius: 15px;

            box-shadow: 0 10px 30px rgba(0, 255, 204, 0.5);

            animation: fadeIn 1s ease-in-out;

            border: 2px solid #00ffcc;

        }

        @keyframes fadeIn {

            from { opacity: 0; transform: translateY(-20px); }

            to { opacity: 1; transform: translateY(0); }

        }

        h2 {

            font-size: 2.5rem;

            margin-bottom: 20px;

            color: #00ffcc;

            animation: flicker 10s infinite;

        }

        @keyframes flicker {

            0%, 18%, 22%, 25%, 53%, 57%, 100% { opacity: 1; }

            20%, 24%, 55% { opacity: 0.2; }

        }

        .building-grid {

            display: grid;

            grid-template-columns: repeat(2, 1fr);

            gap: 20px;

            justify-items: center;

        }

        .building {

            cursor: pointer;

            transition: transform 0.3s ease, box-shadow 0.3s ease;

            background: rgba(0, 0, 0, 0.8);

            border-radius: 15px;

            padding: 20px;

            box-shadow: 0 4px 6px rgba(0, 255, 204, 0.3);

            width: 100%;

            max-width: 200px;

            border: 2px solid #00ffcc;

            animation: pulse 1.5s infinite;

        }

        .building img {

            width: 100%;

            aspect-ratio: 16 / 9;

            border-radius: 10px;

            object-fit: cover;

            border: 2px solid #00ffcc;

            transition: border-color 0.3s ease;

        }

        .building:hover {

            transform: translateY(-10px);

            box-shadow: 0 8px 15px rgba(255, 0, 204, 0.5);

        }

        .building:hover img {

            border-color: #ff00cc;

        }

        .building p {

            margin-top: 10px;

            font-size: 1.2rem;

            font-weight: 600;

            color: #00ffcc;

            transition: color 0.3s ease;

        }

        .building:hover p {

            color: #ff00cc;

        }

        @keyframes pulse {

            0% { transform: scale(1); }

            50% { transform: scale(1.05); }

            100% { transform: scale(1); }

        }

        footer {

            margin-top: 50px;

            padding: 20px;

            background-color: rgba(0, 0, 0, 0.7);

            color: #00ffcc;

            font-size: 14px;

        }

        footer a {

            color: #ff00cc;

            text-decoration: none;

            transition: color 0.3s ease;      }

        footer a:hover {

            color: #00ffcc;}

    </style>

</head>

<body>

    <!-- Electricity Background -->

    <div class="electricity-background"></div>

    <!-- Navbar -->

    <nav>

        <a href="Home.html">Home</a>

        <a href="#">Buildings</a>

        <a href="#">Random Room</a>

        <a href="#">Contact</a>

        <a href="login.html" class="login-btn">Login/Signup</a> </nav>

    <!-- Main Container -->

    <div class="container">

        <h2>Select a Building</h2>

        <div class="building-grid">

            <div class="building" onclick="selectBuilding('A')">

                <img src="assets/A.jpg" alt="A Block">

                <p>A Block</p>  </div>

            <div class="building" onclick="selectBuilding('N')">

                <img src="assets/N.jpg" alt="N Block">

                <p>N Block</p></div>

            <div class="building" onclick="selectBuilding('U')">

                <img src="assets/U.png" alt="U Block">

                <p>U Block</p></div>

            <div class="building" onclick="selectBuilding('H')">

                <img src="assets/images.jpg" alt="H Block">

                <p>H Block</p></div>

        </div>

    </div>

    <footer>

        &copy; 2024 Building Selection. All rights reserved. | <a href="#">Privacy Policy</a> | <a href="#">Terms of Service</a>

    </footer>

    <script>

        function selectBuilding(building) {

            window.location.href = "floors.html?building=" + encodeURIComponent(building);}

    </script>

</body>

</html>

**Calculate.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Energy Consumption Monitor</title>

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap" rel="stylesheet">

  <style>

    body {

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

      background-color: #0a0f1a;

      color: white;

      text-align: center;

      margin: 0;

      padding: 0;

      overflow-x: hidden;

    }

    @keyframes glow {

      0% { box-shadow: 0 0 10px #00d4ff; }

      50% { box-shadow: 0 0 20px #00d4ff; }

      100% { box-shadow: 0 0 10px #00d4ff; }

    }

    .container {

      display: flex;

      justify-content: space-around;

      align-items: flex-start;

      flex-wrap: wrap;

      margin-top: 20px;

      padding: 20px;

    }

    .calculation-table, .chart-container {

      background: rgba(15, 20, 40, 0.9);

      padding: 20px;

      border-radius: 12px;

      width: 45%;

      min-width: 350px;

      height: 420px;

      display: flex;

      flex-direction: column;

      justify-content: space-between;

      border: 2px solid #00d4ff;

      animation: glow 1.5s infinite alternate;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      margin-top: 10px;

    }

    th, td {

      border: 1px solid #00d4ff;

      padding: 10px;

      text-align: center;

      color: white;

    }

    th {

      background: #0077ff;

      color: white;

    }

    input {

      width: 60px;

      padding: 5px;

      text-align: center;

      border: none;

      border-radius: 5px;

      background: #1a1f2e;

      color: white;

      border: 1px solid #00d4ff;

    }

    .btn {

      background: linear-gradient(45deg, #0077ff, #00d4ff);

      color: white;

      border: none;

      padding: 10px 15px;

      margin-top: 10px;

      cursor: pointer;

      border-radius: 5px;

      font-weight: bold;

      transition: 0.3s ease;

    }

    .btn:hover {

      background: linear-gradient(45deg, #00d4ff, #0077ff);

      transform: scale(1.05);

    }

    /\* Digital Clock \*/

    .clock {

      font-size: 22px;

      font-weight: bold;

      margin-bottom: 15px;

      color: #00d4ff;

      text-shadow: 0 0 10px #00d4ff;

    }

    /\* Chart \*/

    .chart-container canvas {

      height: 250px !important;

    }

    /\* Responsive \*/

    @media (max-width: 900px) {

      .container {

        flex-direction: column;

        align-items: center;

      }

      .calculation-table, .chart-container {

        width: 90%;

        height: auto;

      }}

  </style>

</head>

<body>

  <div class="container">

    <div class="calculation-table">

      <h2>Energy Consumption Calculator</h2>

      <table>

        <tr>

          <th>Appliance</th>

          <th>Quantity</th>

          <th>Power (W)</th>

        </tr>

        <tr>

          <td>Ceiling Fans</td>

          <td><p id="fans">Loading...</p></td>

          <td><p id="powerFan">Loading...</p></td>

        </tr>

        <tr>

          <td>Computers</td>

          <td><p id="computers">Loading...</p></td>

          <td><p id="powerComputer">Loading...</p></td>

        </tr>

        <tr>

          <td>Light Bulbs</td>

          <td><p id="lights">Loading...</p></td>

          <td><p id="powerLight">Loading...</p></td>

        </tr>

      </table>

      <button class="btn" onclick="startTimer()">Start Timer</button>

      <button class="btn" onclick="stopTimer()">Stop Timer</button>

      <div class="results">

        <p>Time Elapsed: <span id="timeElapsed">0</span> seconds</p>

        <p><strong>Total Energy Consumed:</strong> <span id="energyConsumed">0</span> Wh</p>

      </div>

    </div>

    <div class="chart-container">

      <h2>Energy Consumption Chart</h2>

      <div class="clock" id="clock">00:00:00</div>

      <canvas id="energyChart"></canvas>

    </div>

  </div>

  <script>

    let timerInterval;

    let startTime;

    let elapsedTime = 0;

    let energyData = [];

    let timeData = [];

    const ctx = document.getElementById('energyChart').getContext('2d');

    const energyChart = new Chart(ctx, {

      type: 'line',

      data: {

        labels: timeData,

        datasets: [{

          label: 'Energy Consumed (Wh)',

          data: energyData,

          borderColor: '#00d4ff',

          backgroundColor: 'rgba(0, 212, 255, 0.2)',

          borderWidth: 2,

          fill: true,

        }]

      },

      options: {

        responsive: true,

        maintainAspectRatio: false,

        scales: {

          x: { ticks: { color: '#fff' } },

          y: { ticks: { color: '#fff' } }

        },

        plugins: { legend: { labels: { color: '#fff' } } }

      }

    });

    function startTimer() {

      if (!startTime) startTime = Date.now() - elapsedTime \* 1000;

      if (!timerInterval) timerInterval = setInterval(updateTime, 1000);

    }

    function stopTimer() {

      clearInterval(timerInterval);

      timerInterval = null;

      calculateEnergy();

    }

    function updateTime() {

      elapsedTime = Math.floor((Date.now() - startTime) / 1000);

      document.getElementById("timeElapsed").textContent = elapsedTime;

      updateClock();

    }

    function updateClock() {

      const now = new Date();

      document.getElementById("clock").textContent = now.toLocaleTimeString();

    }

    function calculateEnergy() {

      let totalPower = parseInt(document.getElementById("powerFan").textContent) \* parseInt(document.getElementById("fans").textContent) +

                       parseInt(document.getElementById("powerComputer").textContent) \* parseInt(document.getElementById("computers").textContent) +

                       parseInt(document.getElementById("powerLight").textContent) \* parseInt(document.getElementById("lights").textContent);

      document.getElementById("energyConsumed").textContent = ((totalPower \* elapsedTime) / 3600).toFixed(2);

    }

    async function fetchRoomData(roomNumber) {

      try {

        const response = await fetch(`http://localhost:3000/room/${roomNumber}`);

        const data = await response.json();

        if (data.error) {

          console.error('Room data not found');

          return;

        }

        document.getElementById('fans').textContent = data.cfans;

        document.getElementById('powerFan').textContent = data.cpow;

        document.getElementById('computers').textContent = data.computers;

        document.getElementById('powerComputer').textContent = data.compow;

        document.getElementById('lights').textContent = data.bulbs;

        document.getElementById('powerLight').textContent = data.bulpow;

      } catch (error) {

        console.error('Error fetching data:', error);

      }

    }

    document.addEventListener('DOMContentLoaded', () => {

      fetchRoomData('1');});

  </script>

</body>

</html>

**Floors.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Floor Selection - Electricity Theme</title>

    <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet">

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

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

            background-color: #000;

            color: #00ffcc;

            text-align: center;

            overflow-x: hidden;

        }

        .electricity-background {

            position: fixed;

            top: 0;

            left: 0;

            width: 100%;

            height: 100%;

            background: radial-gradient(circle, rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 1));

            z-index: -1;

        }

        .electricity-background::before {

            content: '';

            position: absolute;

            width: 100%;

            height: 100%;

            background: linear-gradient(45deg, #00ffcc, #ff00cc, #00ffcc, #ff00cc);

            background-size: 400% 400%;

            animation: electricFlow 10s linear infinite;

            opacity: 0.2;

        }

        @keyframes electricFlow {

            0% { background-position: 0% 50%; }

            50% { background-position: 100% 50%; }

            100% { background-position: 0% 50%; }

        }

        .container {

            max-width: 800px;

            margin: 50px auto;

            background: rgba(0, 0, 0, 0.8);

            padding: 30px;

            border-radius: 15px;

            box-shadow: 0 10px 30px rgba(0, 255, 204, 0.5);

            animation: fadeIn 1s ease-in-out;

            border: 2px solid #00ffcc;

        }

        @keyframes fadeIn {

            from { opacity: 0; transform: translateY(-20px); }

            to { opacity: 1; transform: translateY(0); }

        }

        h2 {

            font-size: 2.5rem;

            margin-bottom: 20px;

            color: #00ffcc;

            animation: flicker 10s infinite;

        }

        @keyframes flicker {

            0%, 18%, 22%, 25%, 53%, 57%, 100% { opacity: 1; }

            20%, 24%, 55% { opacity: 0.2; }

        }

        #floorButtons {

            display: grid;

            grid-template-columns: repeat(2, 1fr); /\* 2 buttons per row \*/

            gap: 20px;

            justify-content: center;

            padding: 20px;

        }

        .floor-button {

            padding: 15px 30px;

            background: linear-gradient(90deg, #00ffcc, #ff00cc);

            color: white;

            border: none;

            border-radius: 10px;

            cursor: pointer;

            font-size: 1.2rem;

            transition: transform 0.3s ease, box-shadow 0.3s ease;

            box-shadow: 0 0 10px rgba(0, 255, 204, 0.8);

            font-weight: bold;

            text-transform: uppercase;

            width: 100%; /\* Full width of the grid column \*/

        }

        .floor-button:hover {

            transform: scale(1.1);

            box-shadow: 0 0 20px rgba(255, 0, 204, 1);

            background: linear-gradient(90deg, #ff00cc, #00ffcc);

        }

        footer {

            margin-top: 50px;

            padding: 20px;

            background-color: rgba(0, 0, 0, 0.7);

            color: #00ffcc;

            font-size: 14px;

        }

        footer a {

            color: #ff00cc;

            text-decoration: none;

            transition: color 0.3s ease;

        }

        footer a:hover {

            color: #00ffcc;

        }

    </style>

</head>

<body>

    <!-- Electricity Background -->

    <div class="electricity-background"></div>

    <!-- Main Container -->

    <div class="container">

        <h2>Select a Floor</h2>

        <div id="floorButtons"></div>

    </div>

    <footer>

        &copy; 2024 Floor Selection. All rights reserved. | <a href="#">Privacy Policy</a> | <a href="#">Terms of Service</a>

    </footer>

    <script>

        const urlParams = new URLSearchParams(window.location.search);

        const building = urlParams.get('building');

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

        for (let i = 0; i <= 4; i++) {

            let button = document.createElement('button');

            button.className = "floor-button";

            button.innerText = "Floor " + i;

            button.onclick = function() {

                window.location.href = "rooms.html?building=" + encodeURIComponent(building) + "&floor=" + i;

            };

            floorButtons.appendChild(button);

        }

    </script>

</body>

</html>

**Fp.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Smart Energy Management System</title>

</head>

<body>

    <!-- Header -->

    <header>

        <div class="header-container">

            <h1 class="logo">

                <span class="logo-icon">⚡</span>

                Smart Energy Management System

            </h1>

            <div class="time" id="current-time"></div>

        </div>

    </header>

    <!-- Main Content -->

    <main>

        <!-- Dashboard Summary -->

        <div class="summary-cards">

            <div class="summary-card">

                <div class="card-icon power">⚡</div>

                <div class="card-content">

                    <h3>Total Power Consumption</h3>

                    <p id="total-consumption">0 kWh</p>

                </div>

            </div>

            <div class="summary-card">

                <div class="card-icon building">🏢</div>

                <div class="card-content">

                    <h3>Total Blocks</h3>

                    <p id="total-blocks">0</p>

                </div>

            </div>

            <div class="summary-card">

                <div class="card-icon users">👥</div>

                <div class="card-content">

                    <h3>Occupancy</h3>

                    <p id="total-occupancy">0 / 0 Rooms</p>

                </div>

            </div>

        </div>

        <!-- Block Overview (Initial View) -->

        <div id="blocks-overview">

            <h2 class="section-title">Building Blocks Overview</h2>

            <div class="block-grid" id="block-grid"></div>

        </div>

        <!-- Block Detail View (Hidden Initially) -->

        <div id="block-detail" class="block-detail">

            <div class="detail-header">

                <button class="back-button" id="back-button">Back</button>

                <h2 class="section-title" id="detail-title">Block Details</h2>

            </div>

            <div class="block-summary">

                <div class="block-summary-grid" id="block-summary-grid">

                    <!-- Will be populated by JavaScript -->

                </div>

            </div>

            <div class="room-list" id="room-list">

                <!-- Will be populated by JavaScript -->

            </div>

        </div>

    </main>

    <!-- Footer -->

    <footer>

        <p>Smart Energy Management System (SEMS) - Saving Energy and Reducing Carbon Footprint</p>

    </footer>

    <script>

        const data = {

            blocks: [

                {

                    id: 'block-a',

                    name: 'Block A',

                    totalConsumption: 234.5,

                    rooms: [

                        { id: 'a-101', name: 'Room 101', occupancy: true, lights: true, fans: true, ac: true, consumption: 45.2 },

                        { id: 'a-102', name: 'Room 102', occupancy: false, lights: false, fans: false, ac: false, consumption: 0 },

                        { id: 'a-103', name: 'Room 103', occupancy: true, lights: true, fans: true, ac: false, consumption: 18.7 },

                        { id: 'a-104', name: 'Room 104', occupancy: false, lights: false, fans: false, ac: false, consumption: 0 },

                    ]

                },

                {

                    id: 'block-b',

                    name: 'Block B',

                    totalConsumption: 187.3,

                    rooms: [

                        { id: 'b-201', name: 'Room 201', occupancy: true, lights: true, fans: true, ac: true, consumption: 52.8 },

                        { id: 'b-202', name: 'Room 202', occupancy: true, lights: true, fans: true, ac: false, consumption: 21.4 },

                        { id: 'b-203', name: 'Room 203', occupancy: false, lights: false, fans: false, ac: false, consumption: 0 },

                    ]

                },

                {

                    id: 'block-c',

                    name: 'Block C',

                    totalConsumption: 312.8,

                    rooms: [

                        { id: 'c-301', name: 'Room 301', occupancy: true, lights: true, fans: true, ac: true, consumption: 65.3 },

                        { id: 'c-302', name: 'Room 302', occupancy: false, lights: false, fans: false, ac: false, consumption: 0 },

                        { id: 'c-303', name: 'Room 303', occupancy: true, lights: true, fans: true, ac: true, consumption: 58.9 },

                        { id: 'c-304', name: 'Room 304', occupancy: true, lights: true, fans: false, ac: false, consumption: 12.6 },

                    ]},]};

        const blocksOverview = document.getElementById('blocks-overview');

        const blockDetail = document.getElementById('block-detail');

        const blockGrid = document.getElementById('block-grid');

        const detailTitle = document.getElementById('detail-title');

        const blockSummaryGrid = document.getElementById('block-summary-grid');

        const roomList = document.getElementById('room-list');

        const backButton = document.getElementById('back-button');

        const currentTimeElement = document.getElementById('current-time');

        const totalConsumptionElement = document.getElementById('total-consumption');

        const totalBlocksElement = document.getElementById('total-blocks');

        const totalOccupancyElement = document.getElementById('total-occupancy');

        function updateTime() {

            const now = new Date();

            currentTimeElement.textContent = now.toLocaleTimeString();

        }

        setInterval(updateTime, 1000);

        updateTime();

        function updateSummaryStats() {

            const totalConsumption = data.blocks.reduce((sum, block) => sum + block.totalConsumption, 0);

            const totalRooms = data.blocks.reduce((sum, block) => sum + block.rooms.length, 0);

            const occupiedRooms = data.blocks.reduce((sum, block) => {

                return sum + block.rooms.filter(room => room.occupancy).length;

            }, 0);

            totalConsumptionElement.textContent = totalConsumption.toFixed(1) + ' kWh';

            totalBlocksElement.textContent = data.blocks.length;

            totalOccupancyElement.textContent = occupiedRooms + ' / ' + totalRooms + ' Rooms';}

        function renderBlocks() {

            blockGrid.innerHTML = '';

            data.blocks.forEach(block => {

                const occupiedRooms = block.rooms.filter(room => room.occupancy).length;

                const blockCard = document.createElement('div');

                blockCard.className = 'block-card';

                blockCard.dataset.blockId = block.id;

                blockCard.innerHTML = `

                    <h3 class="block-name">${block.name}</h3>

                    <div class="block-stats">

                        <div>

                            <p class="stat-label">Power Consumption</p>

                            <p class="stat-value">${block.totalConsumption.toFixed(1)} kWh</p>

                        </div>

                        <div>

                            <p class="stat-label">Occupancy</p>

                            <p class="stat-value">${occupiedRooms} / ${block.rooms.length} Rooms</p>

                        </div>

                    </div>`;

                blockCard.addEventListener('click', () => showBlockDetail(block.id));

                blockGrid.appendChild(blockCard);

            });

        }

        function showBlockDetail(blockId) {

            const block = data.blocks.find(b => b.id === blockId);

            if (!block) return;

            detailTitle.textContent = `${block.name} Details`;

            const occupiedRooms = block.rooms.filter(room => room.occupancy).length;

            const vacantRooms = block.rooms.length - occupiedRooms;

            blockSummaryGrid.innerHTML = `

                <div>

                    <p class="stat-label">Power Consumption</p>

                    <p class="stat-value">${block.totalConsumption.toFixed(1)} kWh</p>

                </div>

                <div>

                    <p class="stat-label">Number of Rooms</p>

                    <p class="stat-value">${block.rooms.length}</p>

                </div>

                <div>

                    <p class="stat-label">Occupied Rooms</p>

                    <p class="stat-value">${occupiedRooms}</p>

                </div>

                <div>

                    <p class="stat-label">Vacant Rooms</p>

                    <p class="stat-value">${vacantRooms}</p>

                </div>`;

            roomList.innerHTML = '';

            block.rooms.forEach(room => {

                const roomItem = document.createElement('div');

                roomItem.className = 'room-item';

                const roomHeader = document.createElement('div');

                roomHeader.className = 'room-header';

                roomHeader.innerHTML = `

                    <div class="room-name">

                        <span class="room-status ${room.occupancy ? 'occupied' : 'vacant'}"></span>

                        <span>${room.name}</span>

                    </div>

                    <div class="room-consumption">

                        <span>${room.consumption} kWh</span>

                        <span class="toggle-icon">+</span>

                    </div>  `;

                const roomDetails = document.createElement('div');

                roomDetails.className = 'room-details';

                roomDetails.innerHTML = `

                    <div class="appliance-grid">

                        <div class="appliance-item">

                            <span class="appliance-label">Occupancy:</span>

                            <span class="appliance-status ${room.occupancy ? 'status-on' : 'status-off'}">

                                ${room.occupancy ? 'Occupied' : 'Vacant'}

                            </span>

                        </div>

                        <div class="appliance-item">

                            <span class="appliance-label">Lights:</span>

                            <span class="appliance-status ${room.lights ? 'status-on' : 'status-off'}">

                                ${room.lights ? 'On' : 'Off'}

                            </span>

                        </div>

                        <div class="appliance-item">

                            <span class="appliance-label">Fans:</span>

                            <span class="appliance-status ${room.fans ? 'status-on' : 'status-off'}">

                                ${room.fans ? 'On' : 'Off'}

                            </span>

                        </div>

                        <div class="appliance-item">

                            <span class="appliance-label">AC:</span>

                            <span class="appliance-status ${room.ac ? 'status-on' : 'status-off'}">

                                ${room.ac ? 'On' : 'Off'}

                            </span>

                        </div>

                    </div> `;

                roomHeader.addEventListener('click', () => {

                    roomDetails.classList.toggle('expanded');

                    const toggleIcon = roomHeader.querySelector('.toggle-icon');

                    toggleIcon.textContent = roomDetails.classList.contains('expanded') ? '-' : '+';

                });

                roomItem.appendChild(roomHeader);

                roomItem.appendChild(roomDetails);

                roomList.appendChild(roomItem);

            });

            blocksOverview.style.display = 'none';

            blockDetail.classList.add('active');

        }

        backButton.addEventListener('click', () => {

            blocksOverview.style.display = 'block';

            blockDetail.classList.remove('active');

        });

        updateSummaryStats();

        renderBlocks();

        function simulateUpdates() {

            const randomBlockIndex = Math.floor(Math.random() \* data.blocks.length);

            const randomBlock = data.blocks[randomBlockIndex];

            const randomRoomIndex = Math.floor(Math.random() \* randomBlock.rooms.length);

            const randomRoom = randomBlock.rooms[randomRoomIndex];

            randomRoom.occupancy = !randomRoom.occupancy;

            if (randomRoom.occupancy) {

                randomRoom.lights = true;

                randomRoom.fans = true;

                randomRoom.ac = Math.random() > 0.5; // 50% chance of AC being on

                randomRoom.consumption = randomRoom.ac ? Math.random() \* 30 + 20 : Math.random() \* 15 + 5;

            } else {

                randomRoom.lights = false;

                randomRoom.fans = false;

                randomRoom.ac = false;

                randomRoom.consumption = 0;

            }

            randomBlock.totalConsumption = randomBlock.rooms.reduce((sum, room) => sum + room.consumption, 0);

            updateSummaryStats();

            renderBlocks();

            if (blockDetail.classList.contains('active')) {

                const currentBlockId = document.querySelector('.room-item')?.parentNode.dataset.blockId;

                if (currentBlockId === randomBlock.id) {

                    showBlockDetail(randomBlock.id);

                }}}

    </script>

</body>

</html>

**Floors.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Floor Selection - Electricity Theme</title>

    <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&display=swap" rel="stylesheet">

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

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

            background-color: #000;

            color: #00ffcc;

            text-align: center;

            overflow-x: hidden;

        }

        .electricity-background {

            position: fixed;

            top: 0;

            left: 0;

            width: 100%;

            height: 100%;

            background: radial-gradient(circle, rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 1));

            z-index: -1;

        }

        .electricity-background::before {

            content: '';

            position: absolute;

            width: 100%;

            height: 100%;

            background: linear-gradient(45deg, #00ffcc, #ff00cc, #00ffcc, #ff00cc);

            background-size: 400% 400%;

            animation: electricFlow 10s linear infinite;

            opacity: 0.2;

        }

        @keyframes electricFlow {

            0% { background-position: 0% 50%; }

            50% { background-position: 100% 50%; }

            100% { background-position: 0% 50%; }

        }

        .container {

            max-width: 800px;

            margin: 50px auto;

            background: rgba(0, 0, 0, 0.8);

            padding: 30px;

            border-radius: 15px;

            box-shadow: 0 10px 30px rgba(0, 255, 204, 0.5);

            animation: fadeIn 1s ease-in-out;

            border: 2px solid #00ffcc;

        }

        @keyframes fadeIn {

            from { opacity: 0; transform: translateY(-20px); }

            to { opacity: 1; transform: translateY(0); }

        }

        h2 {

            font-size: 2.5rem;

            margin-bottom: 20px;

            color: #00ffcc;

            animation: flicker 10s infinite;

        }

        @keyframes flicker {

            0%, 18%, 22%, 25%, 53%, 57%, 100% { opacity: 1; }

            20%, 24%, 55% { opacity: 0.2; }

        }

        #floorButtons {

            display: grid;

            grid-template-columns: repeat(2, 1fr); /\* 2 buttons per row \*/

            gap: 20px;

            justify-content: center;

            padding: 20px;

        }

        .floor-button {

            padding: 15px 30px;

            background: linear-gradient(90deg, #00ffcc, #ff00cc);

            color: white;

            border: none;

            border-radius: 10px;

            cursor: pointer;

            font-size: 1.2rem;

            transition: transform 0.3s ease, box-shadow 0.3s ease;

            box-shadow: 0 0 10px rgba(0, 255, 204, 0.8);

            font-weight: bold;

            text-transform: uppercase;

            width: 100%; /\* Full width of the grid column \*/

        }

        .floor-button:hover {

            transform: scale(1.1);

            box-shadow: 0 0 20px rgba(255, 0, 204, 1);

            background: linear-gradient(90deg, #ff00cc, #00ffcc);

        }

        footer {

            margin-top: 50px;

            padding: 20px;

            background-color: rgba(0, 0, 0, 0.7);

            color: #00ffcc;

            font-size: 14px;

        }

        footer a {

            color: #ff00cc;

            text-decoration: none;

            transition: color 0.3s ease;

        }

        footer a:hover {

            color: #00ffcc;

        }

    </style>

</head>

<body>

    <!-- Electricity Background -->

    <div class="electricity-background"></div>

    <!-- Main Container -->

    <div class="container">

        <h2>Select a Floor</h2>

        <div id="floorButtons"></div>

    </div>

    <footer>

        &copy; 2024 Floor Selection. All rights reserved. | <a href="#">Privacy Policy</a> | <a href="#">Terms of Service</a>

    </footer>

    <script>

        const urlParams = new URLSearchParams(window.location.search);

        const building = urlParams.get('building');

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

        for (let i = 0; i <= 4; i++) {

            let button = document.createElement('button');

            button.className = "floor-button";

            button.innerText = "Floor " + i;

            button.onclick = function() {

                window.location.href = "rooms.html?building=" + encodeURIComponent(building) + "&floor=" + i;

            };

            floorButtons.appendChild(button);

        }

    </script>

</body>

</html>  
**4.2 Back-end Code:**

**Server.js**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const app = express();

const PORT = 3000;

app.use(express.json());

app.use(cors());

const dbURI = 'mongodb://localhost:27017/saveelec';

mongoose.connect(dbURI, {

useNewUrlParser: true,

useUnifiedTopology: true

});

const roomSchema = new mongoose.Schema({

floor: Number,

roomno: Number,

cfans: Number,

cpow: Number,

computers: Number,

compow: Number,

bulbs: Number,

bulpow: Number

});

const Room = mongoose.model('saveelec1', roomSchema);

app.get('/room/:roomNumber', async (req, res) => {

try {

const room = await Room.findOne({ roomno: parseInt(req.params.roomNumber) });

if (!room) {

return res.status(404).json({ error: 'Room not found' });

}

res.json(room);

} catch (err) {

res.status(500).json({ error: 'Server error' });

}

});

app.listen(PORT, () => {

console.log(Server running on http://localhost:${PORT});

});

**Login Details Storage**

document.getElementById("loginForm").addEventListener("submit", async function(event) {

event.preventDefault();

const email = document.getElementById("email").value;

const username = document.getElementById("username").value;

const password = document.getElementById("password").value;

const response = await fetch("http://localhost:3000/login", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ email, username, password })

});

const result = await response.json();

alert(result.message);

});

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

const bodyParser = require("body-parser");

const app = express();

app.use(cors());

app.use(bodyParser.json());

mongoose.connect("mongodb://localhost:27017/userDB", {

useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => console.log("Connected to MongoDB"))

.catch(err => console.error("Could not connect to MongoDB", err));

const userSchema = new mongoose.Schema({

email: String,

username: String,

password: String

});

const User = mongoose.model("User", userSchema);

app.post("/login", async (req, res) => {

try {

const { email, username, password } = req.body;

const newUser = new User({ email, username, password });

await newUser.save();

res.json({ message: "User saved successfully" });

} catch (error) {

res.status(500).json({ message: "Error saving user" });

}

});

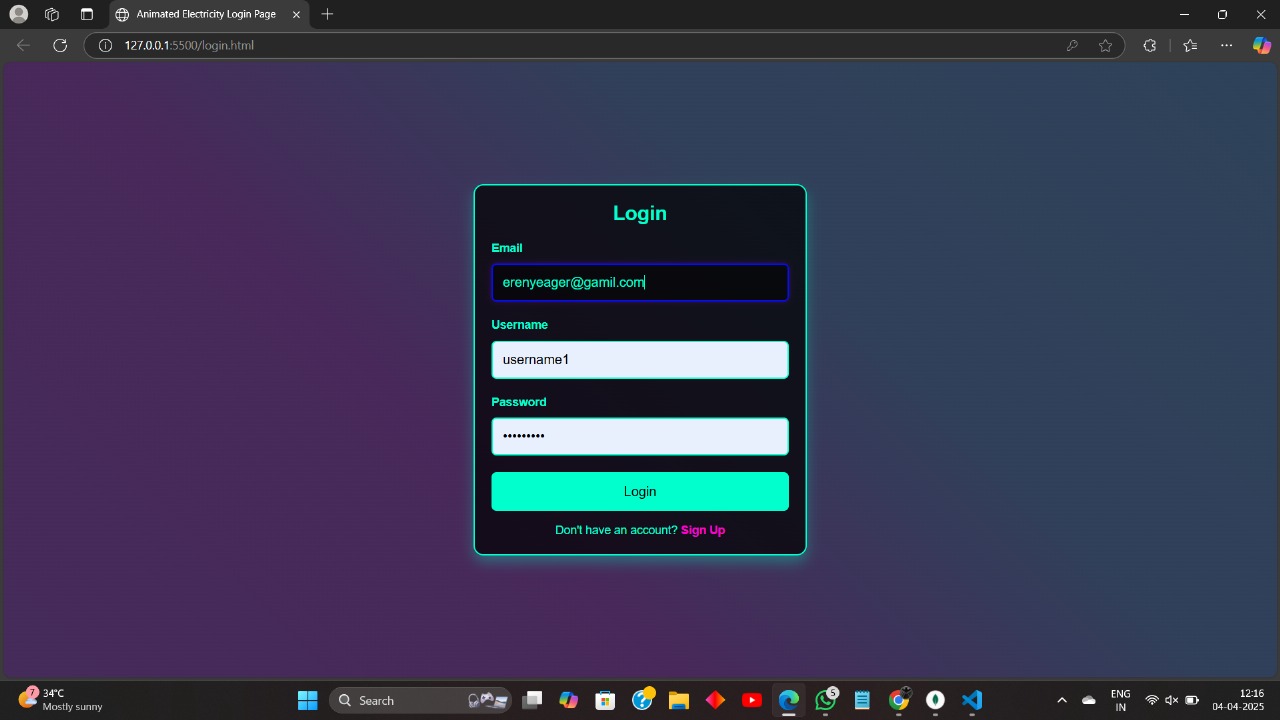
app.listen(3000, () => {

console.log("Server running on port 3000");

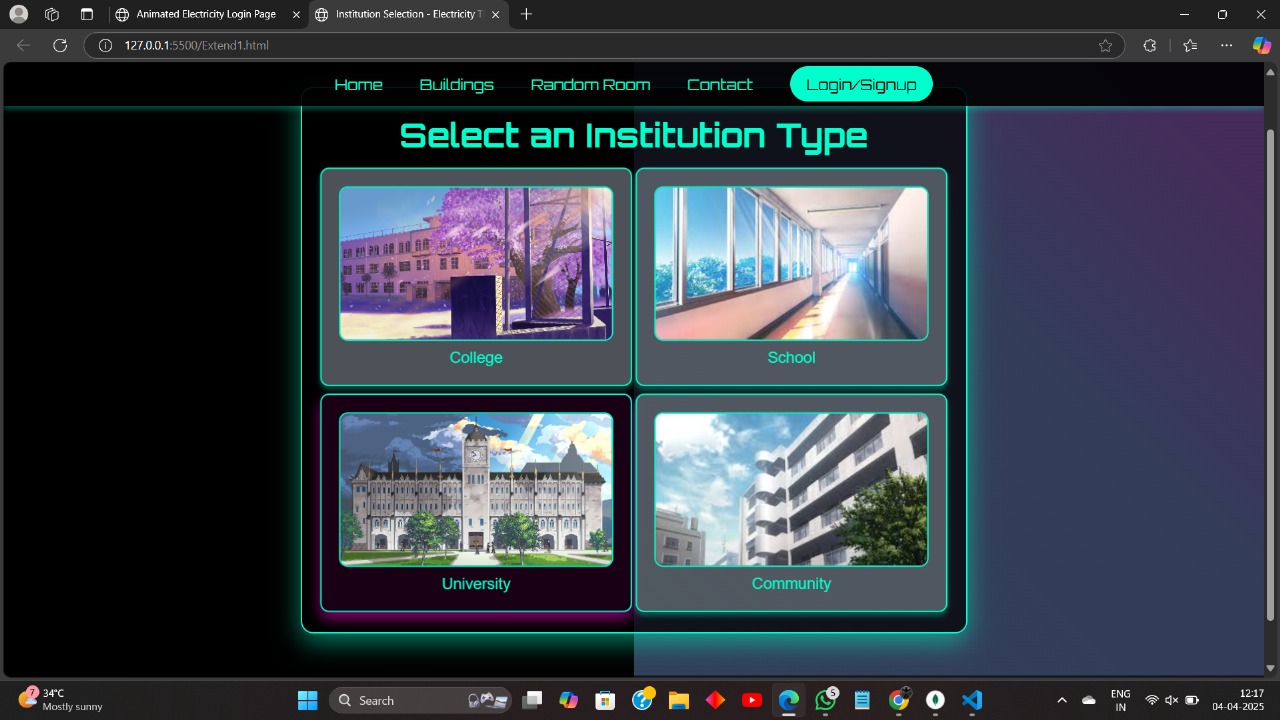
});

**5. Results**

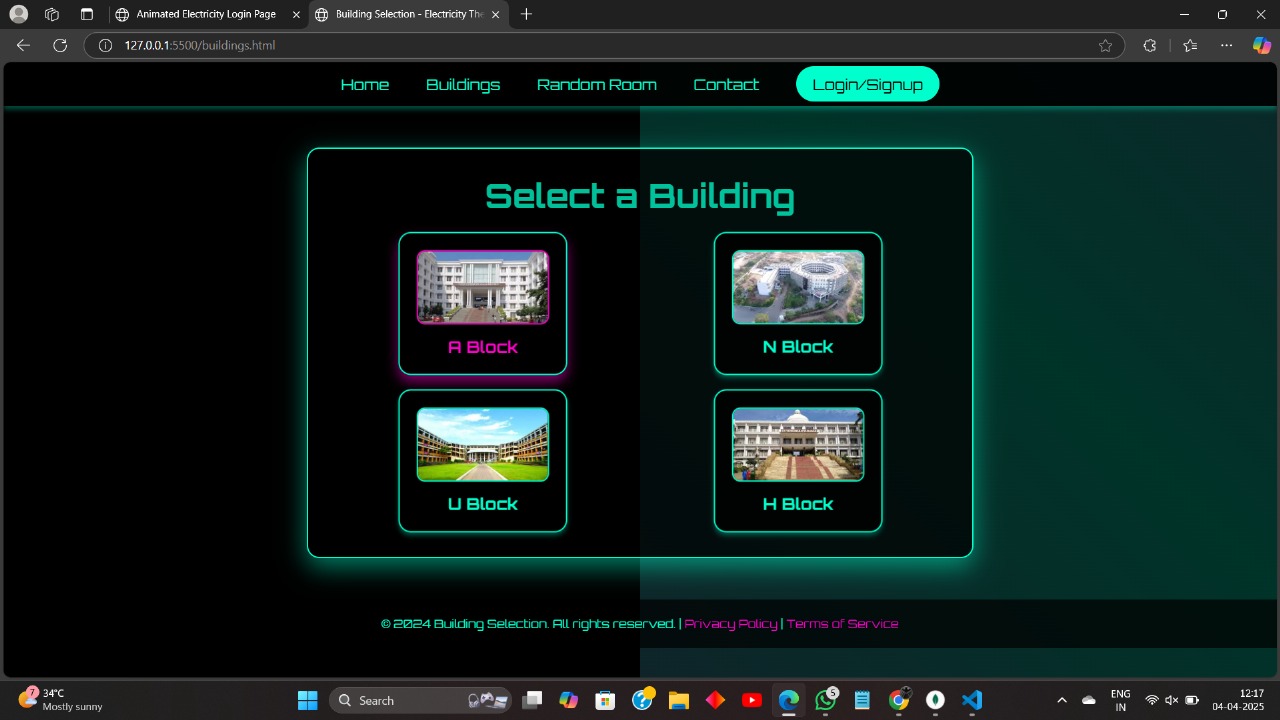
**5.1 Output Screens**



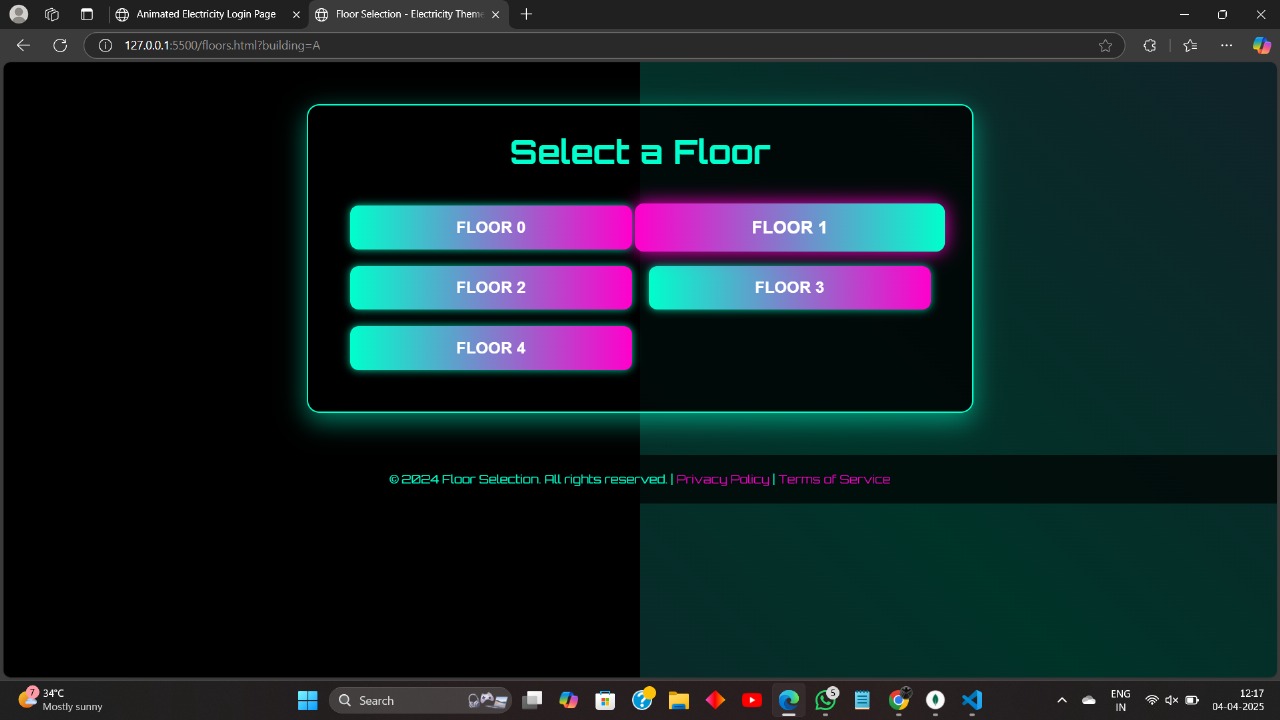
**Fig 5.1.1 Screenshot for Login Page**



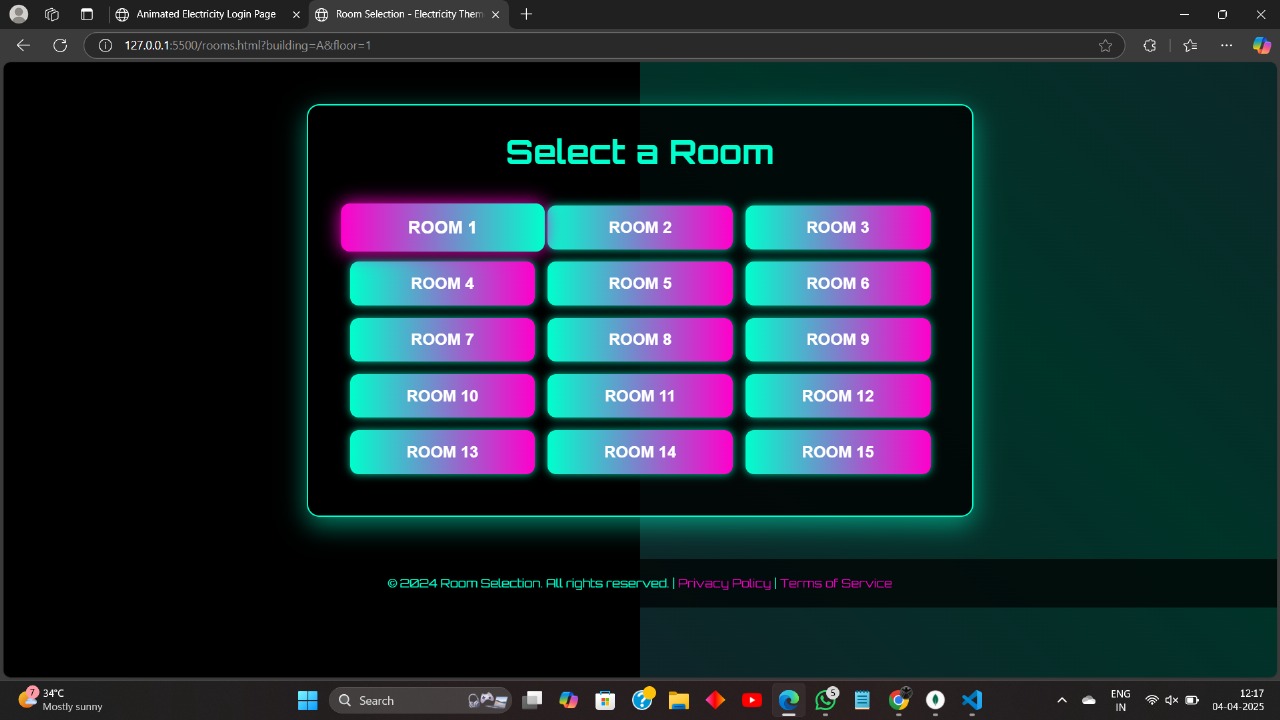
**Fig 5.1.2 Screenshot for Selection of Institution**



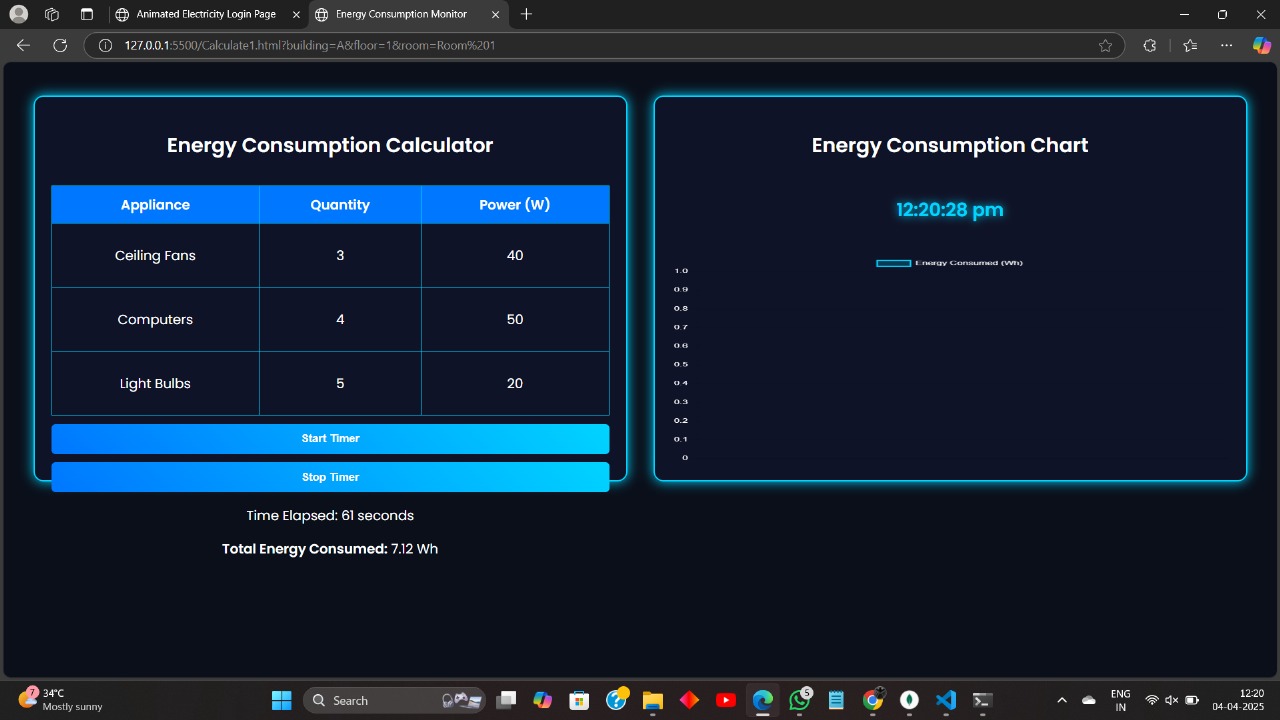
**Fig 5.1.3 Screenshot for selecting Building**



**Fig 5.1.4 Screenshot for selecting Floor**



**Fig 5.1.5 Screenshot for selecting Room No.**



**Fig 5.1.6 Screenshot for Energy Consumption and Statistics**

**6.1 Conclusion**

The Smart Energy Management System, developed using HTML, CSS, and JavaScript, effectively tracks and visualizes energy consumption data. The responsive and visually appealing front-end ensures a smooth user experience across devices, while Chart.js provides interactive graphs for users to easily analyse energy trends. On the back-end, Node.js with MongoDB enables efficient data management, and React.js ensures a dynamic, scalable user interface. SQL is used for lightweight data storage, ensuring fast and responsive performance even with large data sets.

Looking forward, there are opportunities to enhance the system by adding more advanced reporting features, such as analysing data by room type or building. Future versions could also integrate real-time data from energy meters or sensors, offering more accurate and precise energy monitoring for real-world applications. This system offers a comprehensive solution for energy optimization, with potential for expansion into a fully integrated energy management tool.

**6.2 References:**

* Research Papers & Articles
* Retrieved from <https://www.matellio.com>
* Node.js Official Documentation: <https://nodejs.org/en/docs>
* Node.js & Express.js Guide: <https://expressjs.com>