<!-- <!DOCTYPE html>

<html>

<head>

    <title>Satellite Telemetry Data</title>

</head>

<body>

    <h1>Satellite Telemetry Data</h1>

    <table border="1">

        <tr>

            <th>Temperature</th>

            <th>Pressure</th>

            <th>Altitude</th>

            <th>Latitude</th>

            <th>Longitude</th>

            <th>Camera Angle</th>

            <th>Timestamp</th>

        </tr>

        {% for row in data %}

        <tr>

            <td>{{ row.temperature }}</td>

            <td>{{ row.pressure }}</td>

            <td>{{ row.altitude }}</td>

            <td>{{ row.latitude }}</td>

            <td>{{ row.longitude }}</td>

            <td>{{ row.camera\_angle }}</td>

            <td>{{ row.timestamp }}</td>

        </tr>

        {% endfor %}

    </table>

</body>

</html>-->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Telemetry Dashboard</title>

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

  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css">

  <script src="https://cdn.jsdelivr.net/npm/particles.js@2.0.0/particles.min.js"></script>

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

  <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>

  <style>

    :root {

      --primary: #00ff9d;

      --primary-dark: #00cc7d;

      --bg-dark: #0a0a0a;

      --bg-card: rgba(0, 255, 157, 0.15);

      --text: #ffffff;

      --border: rgba(0, 255, 157, 0.2);

      --shadow: rgba(0, 255, 157, 0.2);

      --shadow-hover: rgba(0, 255, 157, 0.3);

    }

    \* {

      margin: 0;

      padding: 0;

      box-sizing: border-box;

    }

    body {

      font-family:'JetBrains Mono', monospace;

      background: var(--bg-dark);

      color: var(--text);

      height: 100vh;

      margin: 0;

      overflow: hidden;

    }

    #particles-js {

      position: fixed;

      width: 100%;

      height: 100%;

      top: 0;

      left: 0;

      z-index: -1;

    }

    .dashboard {

      display: grid;

      grid-template-columns: repeat(4, 1fr);

      grid-template-rows: auto auto auto;

      gap: 0.5rem;

      padding: 0.5rem;

      height: 100vh;

      width: 100%;

      max-width: 1200px;

      margin: 0 auto;

      box-sizing: border-box;

    }

    .card {

      background: var(--bg-card);

      border-radius: 6px;

      padding: 0.5rem;

      box-shadow: 0 2px 8px var(--shadow);

      border: 1px solid var(--border);

      backdrop-filter: blur(6px);

      transition: transform 0.2s ease, box-shadow 0.2s ease;

      overflow: hidden;

      cursor: pointer;

    }

    .card:hover {

      transform: translateY(-1px);

      box-shadow: 0 4px 10px var(--shadow-hover);

    }

    .card-header {

      display: flex;

      justify-content: space-between;

      align-items: center;

      margin-bottom: 0.3rem;

    }

    .card-title {

      font-size: 0.8rem;

      font-weight: 600;

      color: var(--primary);

    }

    .icon {

      color: var(--primary);

      width: 16px;

      height: 16px;

    }

    .metric {

      font-size: 1.5rem;

      font-weight: 700;

      color: var(--primary);

    }

    .unit {

      font-size: 0.7rem;

      color: var(--primary-dark);

      margin-left: 0.1rem;

    }

    .small-metric {

      grid-row: span 1;

    }

    .chart-container {

      grid-column: span 2;

      grid-row: span 1;

    }

    .map-container {

      grid-column: span 2;

      grid-row: span 1;

      height: 100%;

    }

    .chart-wrapper {

      height: calc(100% - 24px);

    }

    #map {

      height: calc(100% - 24px);

      border-radius: 4px;

      z-index: 1;

    }

    /\* Timer Styles \*/

    .timer-container {

      background: var(--bg-card);

      border-radius: 6px;

      padding: 0.5rem;

      text-align: center;

      height: calc(100% - 24px);

      display: flex;

      flex-direction: column;

      justify-content: space-between;

    }

    .timer-title {

      color: var(--primary);

      font-size: 0.9rem;

      margin-bottom: 0.3rem;

      font-weight: bold;

    }

    .timer-display {

      background: rgba(0, 0, 0, 0.3);

      border-radius: 4px;

      padding: 0.5rem;

      margin-bottom: 0.3rem;

    }

    .countdown {

      font-family: 'JetBrains Mono', monospace;

      font-size: 1.2rem;

      color: var(--primary);

      text-shadow: 0 0 6px rgba(0, 255, 157, 0.3);

    }

    .timer-controls {

      display: flex;

      gap: 0.3rem;

      justify-content: center;

    }

    .timer-btn {

      padding: 0.3rem 0.6rem;

      border: none;

      border-radius: 0.2rem;

      font-weight: 600;

      color: white;

      cursor: pointer;

      transition: all 0.2s;

      text-transform: uppercase;

      letter-spacing: 0.3px;

      font-size: 0.6rem;

    }

    .timer-btn:disabled {

      opacity: 0.5;

      cursor: not-allowed;

    }

    .timer-btn.start {

      background: var(--primary);

    }

    .timer-btn.start:hover:not(:disabled) {

      background: var(--primary-dark);

    }

    .timer-btn.stop {

      background: #dc2626;

    }

    .timer-btn.stop:hover:not(:disabled) {

      background: #b91c1c;

    }

    .timer-btn.reset {

      background: #ca8a04;

    }

    .timer-btn.reset:hover:not(:disabled) {

      background: #a16207;

    }

    /\* Speedometer Styles \*/

    .speedometer {

      width: 100%;

      height: calc(100% - 24px);

      background: linear-gradient(135deg, rgba(0, 40, 0, 0.95), rgba(0, 25, 0, 0.95));

      border-radius: 12px;

      box-shadow: 0 0 20px rgba(51, 255, 51, 0.15),

                  inset 0 0 12px rgba(51, 255, 51, 0.05);

      display: flex;

      flex-direction: column;

      justify-content: center;

      align-items: center;

      position: relative;

      padding: 12px;

      text-align: center;

      backdrop-filter: blur(6px);

      border: 1px solid rgba(51, 255, 51, 0.1);

      overflow: hidden;

    }

    .speedometer::before {

      content: '';

      position: absolute;

      top: -1px;

      left: -1px;

      right: -1px;

      bottom: -1px;

      border-radius: 12px;

      background: linear-gradient(45deg, rgba(51, 255, 51, 0.2), transparent, rgba(51, 255, 51, 0.2));

      z-index: -1;

      animation: borderGlow 4s ease-in-out infinite;

    }

    .header {

      position: absolute;

      top: 10px;

      left: 0;

      right: 0;

      display: flex;

      justify-content: space-between;

      padding: 0 12px;

      font-size: 0.6rem;

      letter-spacing: 0.3px;

      color: #33ff33;

    }

    .status {

      display: flex;

      align-items: center;

      gap: 4px;

    }

    .status-dot {

      width: 5px;

      height: 5px;

      background: #33ff33;

      border-radius: 50%;

      animation: blink 2s infinite;

    }

    .coordinates {

      opacity: 0.7;

    }

    .speed-display {

      font-size: 3rem;

      font-weight: 700;

      text-shadow: 0 0 8px rgba(51, 255, 51, 0.5);

      position: relative;

      margin: 15px 0;

      letter-spacing: -0.5px;

      color: #33ff33;

    }

    .speed-display::after {

      content: 'km/h';

      position: absolute;

      font-size: 0.8rem;

      bottom: -8px;

      right: -25px;

      opacity: 0.7;

      font-weight: 400;

      letter-spacing: 0;

    }

    .progress-container {

      width: 90%;

      position: relative;

      margin-top: 20px;

    }

    .progress-bar {

      width: 100%;

      height: 8px;

      background: rgba(51, 255, 51, 0.1);

      border-radius: 4px;

      overflow: hidden;

      position: relative;

      border: 1px solid rgba(51, 255, 51, 0.2);

    }

    .progress {

      height: 100%;

      width: 0%;

      background: linear-gradient(90deg,

          rgba(51, 255, 51, 0.8),

          rgba(51, 255, 51, 1));

      border-radius: 4px;

      transition: width 0.5s ease-in-out;

      position: relative;

      box-shadow: 0 0 8px rgba(51, 255, 51, 0.3);

    }

    .progress::after {

      content: '';

      position: absolute;

      top: 0;

      left: 0;

      right: 0;

      bottom: 0;

      background: linear-gradient(90deg,

          transparent 0%,

          rgba(255, 255, 255, 0.2) 50%,

          transparent 100%);

      animation: shimmer 2s infinite;

    }

    .markers {

      width: 100%;

      display: flex;

      justify-content: space-between;

      margin-top: 4px;

      padding: 0 4px;

      font-size: 0.6rem;

      opacity: 0.6;

      color: #33ff33;

    }

    .satellite-icon {

      position: absolute;

      top: -15px;

      right: -15px;

      width: 30px;

      height: 30px;

      animation: orbit 6s linear infinite;

      filter: drop-shadow(0 0 3px rgba(51, 255, 51, 0.3));

    }

    .footer {

      position: absolute;

      bottom: 10px;

      left: 0;

      right: 0;

      text-align: center;

      font-size: 0.6rem;

      opacity: 0.5;

      letter-spacing: 0.5px;

      color: #33ff33;

    }

    .particles {

      position: absolute;

      width: 100%;

      height: 100%;

      pointer-events: none;

    }

    .particle {

      position: absolute;

      background: #33ff33;

      border-radius: 50%;

      pointer-events: none;

      opacity: 0;

      box-shadow: 0 0 1px #33ff33;

    }

    /\* Modal Styles \*/

    .modal {

      display: none;

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

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

      z-index: 1000;

      justify-content: center;

      align-items: center;

    }

    .modal-content {

      background: var(--bg-card);

      border-radius: 8px;

      padding: 1rem;

      width: 80%;

      max-width: 800px;

      max-height: 80vh;

      overflow: auto;

      border: 1px solid var(--border);

      box-shadow: 0 0 20px rgba(0, 255, 157, 0.3);

    }

    .modal-header {

      display: flex;

      justify-content: space-between;

      align-items: center;

      margin-bottom: 1rem;

    }

    .modal-title {

      color: var(--primary);

      font-size: 1.2rem;

      font-weight: bold;

    }

    .close-modal {

      background: none;

      border: none;

      color: var(--primary);

      font-size: 1.5rem;

      cursor: pointer;

    }

    .modal-chart-container {

      height: 400px;

      width: 100%;

    }

    @keyframes borderGlow {

      0%, 100% { opacity: 0.3; }

      50% { opacity: 0.8; }

    }

    @keyframes shimmer {

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

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

    }

    @keyframes blink {

      0%, 100% { opacity: 1; }

      50% { opacity: 0.3; }

    }

    @keyframes orbit {

      from { transform: rotate(0deg) translateX(10px) rotate(0deg); }

      to { transform: rotate(360deg) translateX(10px) rotate(-360deg); }

    }

    @media (max-width: 1000px) {

      .dashboard {

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

        grid-template-rows: auto auto auto auto;

        height: auto;

        min-height: 100vh;

        overflow-y: auto;

        max-width: 800px;

      }

      .chart-container, .map-container {

        grid-column: span 1;

      }

      .speed-display {

        font-size: 2.5rem;

      }

      .header {

        padding: 0 8px;

        font-size: 0.5rem;

      }

      .modal-content {

        width: 95%;

      }

    }

    @media (max-width: 600px) {

      .dashboard {

        grid-template-columns: 1fr;

        padding: 0.3rem;

        max-width: 500px;

      }

      .card {

        grid-column: span 1 !important;

      }

      .metric {

        font-size: 1.2rem;

      }

      .countdown {

        font-size: 1.1rem;

      }

      .speed-display {

        font-size: 2rem;

      }

      .speed-display::after {

        font-size: 0.7rem;

        bottom: -6px;

        right: -20px;

      }

    }

  </style>

</head>

<body>

  <div id="particles-js"></div>

  <div class="dashboard">

    <!-- First Row - Metrics -->

    <div class="card small-metric" id="temperature-card">

      <div class="card-header">

        <span class="card-title">Temperature</span>

        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="#00ff9d" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="icon"><path d="M14 4v10.54a4 4 0 1 1-4 0V4a2 2 0 0 1 4 0Z"></path></svg>

      </div>

      <div class="metric" id="temperature">24.5°C</div>

    </div>

    <div class="card small-metric" id="altitude-card">

      <div class="card-header">

        <span class="card-title">Altitude</span>

        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="#00ff9d" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="icon"><path d="M3 3v18h18"></path><path d="M7 17v-4"></path><path d="M11 17v-8"></path><path d="M15 17v-4"></path><path d="M19 17v-8"></path></svg>

      </div>

      <div class="metric" id="altitude">1,234<span class="unit">m</span></div>

    </div>

    <div class="card small-metric" id="pressure-card">

      <div class="card-header">

        <span class="card-title">Pressure</span>

        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="#00ff9d" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="icon"><path d="m12 14 4-4"></path><path d="M3.34 19a10 10 0 1 1 17.32 0"></path></svg>

      </div>

      <div class="metric" id="pressure">1,013<span class="unit">hPa</span></div>

    </div>

    <div class="card small-metric" id="angle-card">

      <div class="card-header">

        <span class="card-title">Angle</span>

        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="#00ff9d" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="icon"><circle cx="12" cy="12" r="10"></circle><path d="M16.24 7.76a6 6 0 0 1 0 8.49m-8.48-.01a6 6 0 0 1 0-8.49m11.31-2.82a10 10 0 0 1 0 14.14m-14.14 0a10 10 0 0 1 0-14.14"></path></svg>

      </div>

      <div class="metric" id="angle">45.2<span class="unit">°</span></div>

    </div>

    <!-- Second Row - Map and Satellite Position -->

    <div class="card map-container">

      <div class="card-header">

        <span class="card-title">Live Location Tracking</span>

      </div>

      <div id="map"></div>

    </div>

    <div class="card chart-container">

      <div class="card-header">

        <span class="card-title">Satellite Position</span>

      </div>

      <div class="chart-wrapper">

        <canvas id="satelliteChart"></canvas>

      </div>

    </div>

    <!-- Third Row - Timer and Speedometer -->

    <div class="card" style="grid-column: span 1;">

      <div class="timer-container">

        <h2 class="timer-title">Mission Timer</h2>

        <div class="timer-display">

          <div class="countdown" id="countdown">05:55:25</div>

        </div>

        <div class="timer-controls">

          <button id="startBtn" class="timer-btn start">Start</button>

          <button id="stopBtn" class="timer-btn stop" disabled>Stop</button>

          <button id="resetBtn" class="timer-btn reset">Reset</button>

        </div>

      </div>

    </div>

    <div class="card" style="grid-column: span 3;">

      <div class="speedometer">

        <div class="header">

          <div class="status">

            <div class="status-dot"></div>

            SATELLITE LINK: ACTIVE

          </div>

          <div class="coordinates">LAT: 51.5074° N &nbsp; LON: 0.1278° W</div>

        </div>

        <div class="satellite-icon">🛰️</div>

        <div class="speed-display" id="satellite-speed">0</div>

        <div class="progress-container">

          <div class="progress-bar">

            <div class="progress" id="satellite-progress"></div>

          </div>

          <div class="markers">

            <span>0</span>

            <span>2,500</span>

            <span>5,000</span>

            <span>7,500</span>

            <span>10,000</span>

          </div>

        </div>

        <div class="particles" id="particles"></div>

        <div class="footer">TELEMETRY DATA v2.5.0</div>

      </div>

    </div>

  </div>

  <!-- Modal for showing metric graphs -->

  <div class="modal" id="metricModal">

    <div class="modal-content">

      <div class="modal-header">

        <h2 class="modal-title" id="modalTitle">Metric History</h2>

        <button class="close-modal" id="closeModal">&times;</button>

      </div>

      <div class="modal-chart-container">

        <canvas id="metricChart"></canvas>

      </div>

    </div>

  </div>

  <script>

    // Particles.js configuration

    const particlesConfig = {

      particles: {

        number: {

          value: 40,

          density: {

            enable: true,

            value\_area: 400

          }

        },

        color: {

          value: "#00ff9d"

        },

        shape: {

          type: "circle",

          stroke: {

            width: 0,

            color: "#000000"

          }

        },

        opacity: {

          value: 0.3,

          random: true

        },

        size: {

          value: 2,

          random: true

        },

        line\_linked: {

          enable: true,

          distance: 100,

          color: "#00ff9d",

          opacity: 0.2,

          width: 1

        },

        move: {

          enable: true,

          speed: 0.5,

          direction: "none",

          random: true,

          straight: false,

          out\_mode: "out",

          bounce: false

        }

      },

      interactivity: {

        detect\_on: "canvas",

        events: {

          onhover: {

            enable: true,

            mode: "grab"

          },

          onclick: {

            enable: true,

            mode: "push"

          },

          resize: true

        },

        modes: {

          grab: {

            distance: 100,

            line\_linked: {

              opacity: 1

            }

          },

          push: {

            particles\_nb: 2

          }

        }

      },

      retina\_detect: true

    };

    // Data storage for metrics

    const metricData = {

      temperature: { values: [], timestamps: [] },

      altitude: { values: [], timestamps: [] },

      pressure: { values: [], timestamps: [] },

      angle: { values: [], timestamps: [] }

    };

    // Initialize the satellite speedometer

    function initSatelliteSpeedometer() {

      let speed = 0;

      const maxSpeed = 500;

      function createParticle() {

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

        if (!particles) return;

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

        particle.className = 'particle';

        const size = Math.random() \* 1.5 + 0.5;

        const speedX = (Math.random() - 0.5) \* 8;

        const speedY = (Math.random() - 0.5) \* 8;

        const startX = Math.random() \* 100;

        const startY = Math.random() \* 100;

        particle.style.width = `${size}px`;

        particle.style.height = `${size}px`;

        particle.style.left = `${startX}%`;

        particle.style.top = `${startY}%`;

        particles.appendChild(particle);

        let opacity = 0.8;

        let posX = startX;

        let posY = startY;

        function updateParticle() {

          opacity -= 0.01;

          posX += speedX \* 0.1;

          posY += speedY \* 0.1;

          particle.style.opacity = opacity.toString();

          particle.style.left = `${posX}%`;

          particle.style.top = `${posY}%`;

          if (opacity > 0) {

            requestAnimationFrame(updateParticle);

          } else {

            particle.remove();

          }

        }

        requestAnimationFrame(updateParticle);

      }

      function updateCoordinates() {

        const coordsElement = document.querySelector('.coordinates');

        if (!coordsElement) return;

        const lat = (Math.random() \* 0.001 + 51.5074).toFixed(4);

        const lon = (Math.random() \* 0.001 - 0.1278).toFixed(4);

        coordsElement.textContent = `LAT: ${lat}° N   LON: ${lon}° W`;

      }

      function addSpeedGlow() {

        const speedDisplay = document.getElementById('satellite-speed');

        if (!speedDisplay) return;

        speedDisplay.style.textShadow = '0 0 10px rgba(51, 255, 51, 0.5)';

        setTimeout(() => {

          speedDisplay.style.textShadow = '0 0 6px rgba(51, 255, 51, 0.5)';

        }, 300);

      }

      function increaseSpeed() {

        if (speed < maxSpeed) {

          const increment = Math.floor(Math.random() \* 500);

          speed = Math.min(maxSpeed, speed + increment);

          const speedElement = document.getElementById("satellite-speed");

          const progressElement = document.getElementById("satellite-progress");

          if (speedElement) {

            speedElement.innerText = speed.toLocaleString();

          }

          if (progressElement) {

            progressElement.style.width = (speed / maxSpeed \* 100) + "%";

          }

          addSpeedGlow();

          updateCoordinates();

          if (increment > 200) {

            for (let i = 0; i < 2; i++) {

              createParticle();

            }

          }

          const delay = Math.max(100, 500 - (speed / maxSpeed \* 300));

          setTimeout(increaseSpeed, delay);

        }

      }

      setInterval(createParticle, 2000);

      setInterval(updateCoordinates, 3000);

      increaseSpeed();

    }

    // Timer functionality

    function initializeTimer() {

      let timer = null;

      const INITIAL\_TIME = 5 \* 3600 + 55 \* 60 + 25; // 5:55:25

      let timeLeft = INITIAL\_TIME;

      let isRunning = false;

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

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

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

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

      function formatTime(time) {

        const hours = Math.floor(time / 3600);

        const minutes = Math.floor((time % 3600) / 60);

        const seconds = time % 60;

        return {

          hours: String(hours).padStart(2, '0'),

          minutes: String(minutes).padStart(2, '0'),

          seconds: String(seconds).padStart(2, '0')

        };

      }

      function updateDisplay() {

        const { hours, minutes, seconds } = formatTime(timeLeft);

        countdown.textContent = `${hours}:${minutes}:${seconds}`;

      }

      function startTimer() {

        if (!isRunning && timeLeft > 0) {

          isRunning = true;

          startBtn.disabled = true;

          stopBtn.disabled = false;

          if (timer !== null) {

            clearInterval(timer);

          }

          timer = setInterval(() => {

            if (timeLeft > 0) {

              timeLeft--;

              updateDisplay();

              if (timeLeft === 0) {

                stopTimer();

                startBtn.disabled = true;

              }

            }

          }, 1000);

        }

      }

      function stopTimer() {

        if (isRunning) {

          isRunning = false;

          startBtn.disabled = false;

          stopBtn.disabled = true;

          if (timer !== null) {

            clearInterval(timer);

            timer = null;

          }

        }

      }

      function resetTimer() {

        stopTimer();

        timeLeft = INITIAL\_TIME;

        startBtn.disabled = false;

        stopBtn.disabled = true;

        updateDisplay();

      }

      // Event Listeners

      startBtn.addEventListener('click', startTimer);

      stopBtn.addEventListener('click', stopTimer);

      resetBtn.addEventListener('click', resetTimer);

      // Initialize display

      updateDisplay();

    }

    // Modal functionality

    function initializeModal() {

      const modal = document.getElementById('metricModal');

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

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

      const metricChartCtx = document.getElementById('metricChart').getContext('2d');

      let metricChart = null;

      // Close modal when clicking X or outside

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

        modal.style.display = 'none';

      });

      window.addEventListener('click', (event) => {

        if (event.target === modal) {

          modal.style.display = 'none';

        }

      });

      // Function to show metric graph

      function showMetricGraph(metricName) {

        const metricInfo = {

          'temperature': { title: 'Temperature', unit: '°C' },

          'altitude': { title: 'Altitude', unit: 'm' },

          'pressure': { title: 'Pressure', unit: 'hPa' },

          'angle': { title: 'Angle', unit: '°' }

        };

        modalTitle.textContent = `${metricInfo[metricName].title} History`;

        modal.style.display = 'flex';

        // Destroy previous chart if exists

        if (metricChart) {

          metricChart.destroy();

        }

        // Create new chart

        metricChart = new Chart(metricChartCtx, {

          type: 'line',

          data: {

            labels: metricData[metricName].timestamps,

            datasets: [{

              label: metricInfo[metricName].title,

              data: metricData[metricName].values,

              borderColor: '#00ff9d',

              backgroundColor: 'rgba(0, 255, 157, 0.2)',

              borderWidth: 2,

              pointRadius: 3,

              pointHoverRadius: 5,

              tension: 0.4,

              fill: true

            }]

          },

          options: {

            responsive: true,

            maintainAspectRatio: false,

            plugins: {

              legend: {

                labels: {

                  color: '#00ff9d',

                  font: { size: 12 }

                }

              },

              tooltip: {

                callbacks: {

                  label: function(context) {

                    return `${context.dataset.label}: ${context.raw} ${metricInfo[metricName].unit}`;

                  }

                },

                backgroundColor: 'rgba(0, 0, 0, 0.8)',

                titleColor: '#00ff9d',

                bodyColor: '#00ff9d',

                borderColor: 'rgba(0, 255, 157, 0.2)',

                borderWidth: 1

              }

            },

            scales: {

              x: {

                ticks: {

                  color: '#00ff9d',

                  font: { size: 10 }

                },

                grid: {

                  color: 'rgba(0, 255, 157, 0.1)'

                },

                title: {

                  display: true,

                  text: 'Time',

                  color: '#00ff9d',

                  font: { size: 12 }

                }

              },

              y: {

                ticks: {

                  color: '#00ff9d',

                  font: { size: 10 }

                },

                grid: {

                  color: 'rgba(0, 255, 157, 0.1)'

                },

                title: {

                  display: true,

                  text: metricInfo[metricName].unit,

                  color: '#00ff9d',

                  font: { size: 12 }

                }

              }

            }

          }

        });

      }

      // Add click event listeners to metric cards

      document.getElementById('temperature-card').addEventListener('click', () => showMetricGraph('temperature'));

      document.getElementById('altitude-card').addEventListener('click', () => showMetricGraph('altitude'));

      document.getElementById('pressure-card').addEventListener('click', () => showMetricGraph('pressure'));

      document.getElementById('angle-card').addEventListener('click', () => showMetricGraph('angle'));

    }

    // Initialize everything when the DOM is loaded

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

      // Initialize particles.js

      if (window.particlesJS) {

        window.particlesJS('particles-js', particlesConfig);

      }

      // Initialize Map

      let map = L.map('map').setView([12.9716, 77.5946], 10);

      L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

        attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

      }).addTo(map);

      let marker = L.marker([12.9716, 77.5946]).addTo(map).bindPopup('Live Location').openPopup();

      // Initialize Chart

      const satelliteCtx = document.getElementById('satelliteChart').getContext('2d');

      const chartLabels = [];

      const latitudeData = [];

      const longitudeData = [];

      const satelliteChart = new Chart(satelliteCtx, {

        type: 'line',

        data: {

          labels: chartLabels,

          datasets: [

            {

              label: 'Latitude',

              data: latitudeData,

              borderColor: '#00ff9d',

              backgroundColor: 'rgba(0, 255, 157, 0.2)',

              borderWidth: 1,

              pointRadius: 2,

              pointHoverRadius: 4,

              tension: 0.4,

              fill: true

            },

            {

              label: 'Longitude',

              data: longitudeData,

              borderColor: '#00cc7d',

              backgroundColor: 'rgba(0, 204, 125, 0.2)',

              borderWidth: 1,

              pointRadius: 2,

              pointHoverRadius: 4,

              tension: 0.4,

              fill: true

            }

          ]

        },

        options: {

          responsive: true,

          maintainAspectRatio: false,

          plugins: {

            legend: {

              labels: {

                color: '#00ff9d',

                font: { size: 10 }

              }

            },

            tooltip: {

              backgroundColor: 'rgba(0, 0, 0, 0.8)',

              titleColor: '#00ff9d',

              bodyColor: '#00ff9d',

              borderColor: 'rgba(0, 255, 157, 0.2)',

              borderWidth: 1

            }

          },

          scales: {

            x: {

              ticks: {

                color: '#00ff9d',

                font: { size: 8 }

              },

              grid: {

                color: 'rgba(0, 255, 157, 0.1)'

              },

              title: {

                display: true,

                text: 'Time',

                color: '#00ff9d',

                font: { size: 10 }

              }

            },

            y: {

              ticks: {

                color: '#00ff9d',

                font: { size: 8 }

              },

              grid: {

                color: 'rgba(0, 255, 157, 0.1)'

              },

              title: {

                display: true,

                text: 'Degrees',

                color: '#00ff9d',

                font: { size: 10 }

              }

            }

          }

        }

      });

      // Update metrics function

      function updateMetrics() {

        const now = new Date();

        const timestamp = now.toLocaleTimeString();

        // Temperature

        const tempValue = (20 + Math.random() \* 10).toFixed(1);

        document.getElementById('temperature').textContent = `${tempValue}°C`;

        metricData.temperature.values.push(tempValue);

        metricData.temperature.timestamps.push(timestamp);

        // Altitude

        const altValue = Math.floor(1000 + Math.random() \* 1000);

        document.getElementById('altitude').innerHTML = `${altValue}<span class="unit">m</span>`;

        metricData.altitude.values.push(altValue);

        metricData.altitude.timestamps.push(timestamp);

        // Pressure

        const pressValue = Math.floor(980 + Math.random() \* 40);

        document.getElementById('pressure').innerHTML = `${pressValue}<span class="unit">hPa</span>`;

        metricData.pressure.values.push(pressValue);

        metricData.pressure.timestamps.push(timestamp);

        // Angle

        const angleValue = (Math.random() \* 360).toFixed(1);

        document.getElementById('angle').innerHTML = `${angleValue}<span class="unit">°</span>`;

        metricData.angle.values.push(angleValue);

        metricData.angle.timestamps.push(timestamp);

        // Keep only the last 20 data points

        Object.keys(metricData).forEach(metric => {

          if (metricData[metric].values.length > 20) {

            metricData[metric].values.shift();

            metricData[metric].timestamps.shift();

          }

        });

      }

      // Update map location function

      function updateLocation() {

        const lat = 12.9 + Math.random() \* 0.2;

        const lng = 77.5 + Math.random() \* 0.2;

        marker.setLatLng([lat, lng]);

        map.setView([lat, lng], 10);

      }

      // Update satellite chart function

      function updateSatelliteGraph() {

        const currentTime = new Date().toLocaleTimeString();

        const latitude = 12.9 + Math.random() \* 0.2;

        const longitude = 77.5 + Math.random() \* 0.2;

        chartLabels.push(currentTime);

        latitudeData.push(latitude);

        longitudeData.push(longitude);

        // Keep only the last 6 points

        if (chartLabels.length > 6) {

          chartLabels.shift();

          latitudeData.shift();

          longitudeData.shift();

        }

        satelliteChart.update();

      }

      // Set up intervals for updates

      setInterval(updateMetrics, 2000);

      setInterval(updateLocation, 2000);

      setInterval(updateSatelliteGraph, 5000);

      // Initial updates

      updateMetrics();

      updateLocation();

      updateSatelliteGraph();

      // Initialize the satellite speedometer

      initSatelliteSpeedometer();

      // Initialize the timer

      initializeTimer();

      // Initialize the modal

      initializeModal();

      // Handle window resize for chart responsiveness

      window.addEventListener('resize', function() {

        satelliteChart.resize();

      });

    });

  </script>

</body>

</html>