```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Health Monitoring System</title>
  <style>
    body {
      padding: 25px;
      font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #0f2027, #203a43, #2c5364);
      color: #b2ebf2;
    }
    .title {
      color: #5C6AC4;
      text-align: center;
      font-size: 2rem;
      margin-bottom: 20px;
      position: relative;
      letter-spacing: 0.5px;
      text-shadow: 0 0 4px rgba(92,106,196,0.8), 0 0 12px rgba(92,106,196,0.6);
    }
    .container {
      background: rgba(0, 77, 96, 0.9);
      padding: 30px;
      border-radius: 15px;
      max-width: 750px;
      margin: auto;
    }
    label { display: block; margin-top: 10px; margin-bottom: 5px; }
    input {
      width: 100%; padding: 10px; border-radius: 8px; border: none; margin-
bottom: 10px;
    }
    button {
      margin-top: 10px; padding: 10px; border-radius: 10px;
      border: none; cursor: pointer; background: #00e5ff; font-weight: bold;
    }
    button#resetBtn { background: #f44336; color: white; }
    table { width: 100%; border-collapse: collapse; margin-top: 15px; }
    th, td { padding: 8px; border-bottom: 1px solid #00bcd4; text-align: left; }
    .normal { color: #4caf50; font-weight: bold; }
    .abnormal { color: #f44336; font-weight: bold; }
    .summary { margin-top: 15px; text-align: center; font-size: 1.2rem; font-
weight: bold; }
    .summary.abnormal { color: #f44336; }
    .summary.normal { color: #4caf50; }
    .location { margin-top: 10px; font-size: 1rem; font-weight: bold; color:
#ffeb3b; }
    #currentTime { margin-top: 15px; font-size: 1.1rem; text-align: center;
color: #ffcc80; font-weight: bold; }
  </style>
</head>
<body>
  <div class="container" role="main">
    <h1 class="title">Wireless Health Monitoring System</h1>
    <form id="healthForm">
      <label for="name">Patient Name:</label>
      <input type="text" id="name" name="name" placeholder="Enter patient name"
required />
      <label for="age">Age:</label>
      <input type="number" id="age" name="age" placeholder="Enter patient age"
min="0" max="120" step="1" required />
      <button type="submit" id="submitBtn">Submit</button>
      <button type="button" id="resetBtn">Reset</button>
    </form>

    <div id="results" class="results" style="display:none;">
      <div class="patient-info" id="patientInfo"></div>
      <div class="location" id="locationInfo">📍 Detecting location...</div>
      <table>
        <thead><tr><th>Parameter</th><th>Value</th></tr></thead>
        <tbody id="parametersTable"></tbody>
      </table>
    </div>

    <div id="summary" class="summary"></div>
    <p id="currentTime"></p>
  </div>

  <script>
    const form = document.getElementById('healthForm');
    const resultsDiv = document.getElementById('results');
    const summaryDiv = document.getElementById('summary');
    const patientInfoDiv = document.getElementById('patientInfo');
    const parametersTableBody = document.getElementById('parametersTable');
    const locationInfo = document.getElementById('locationInfo');

    const indianNames = 
["Aarav","Vivaan","Aditya","Vihaan","Arjun","Sai","Krishna","Ishaan","Rohan","Ar
yan",

"Karthik","Pranav","Rudra","Siddharth","Rishi","Kabir","Anirudh","Manoj","Puneet
","Harsha",

"Lakshmi","Ananya","Kavya","Divya","Sneha","Aishwarya","Priya","Sahana","Shreya"
,"Radhika",

"Meera","Pooja","Keerthi","Nandini","Swathi","Bhavana","Chaitra","Navya","Sindhu
","Bindu",

"Rajesh","Suresh","Mahesh","Ramesh","Gopal","Venkatesh","Sanjay","Anand","Sunil"
,"Vinay",

"Abhishek","Ashok","Deepak","Kiran","Naveen","Ravi","Shiva","Mohan","Ajay","Uday
",

"Tejas","Darshan","Varun","Rahul","Omkar","Harish","Yash","Nikhil","Chetan","Sha
rath",

"Smitha","Geetha","Jyothi","Rekha","Radha","Madhuri","Seema","Vidya","Rupa","Sun
itha",

"Padma","Gayathri","Archana","Latha","Malini","Pushpa","Sowmya","Uma","Anitha","
Sandhya"
    ];

    function randomFloat(min, max, decimals = 1) {
      return parseFloat((Math.random() * (max - min) + min).toFixed(decimals));
    }
    function randomInt(min, max) {
      return Math.floor(Math.random() * (max - min + 1)) + min;
    }

    function generateParameters() {
      const temperature = randomFloat(35, 39);
      const spo2 = randomInt(90, 100);
      const heartRate = randomInt(50, 110);
      const bedNumber = randomInt(1, 100);
      return {
        bedNumber,
        parameters: [
          { name: 'Body Temperature', value: `${temperature} °C`, normal:
temperature >= 36.1 && temperature <= 37.2 },
          { name: 'SpO2 Level', value: `${spo2} %`, normal: spo2 >= 95 && spo2
<= 100 },
          { name: 'Heart Rate', value: `${heartRate} bpm`, normal: heartRate >=
60 && heartRate <= 100 }
        ]
      };
    }

    function renderResults(name, age, bedNumber, parameters) {
      patientInfoDiv.textContent = `Patient: ${name}, Age: ${age}, Bed No: $
{bedNumber}`;
      parametersTableBody.innerHTML = '';
      parameters.forEach(p => {
        const tr = document.createElement('tr');
        tr.innerHTML = `<td>${p.name}</td><td class="${p.normal ? 'normal' :
'abnormal'}">${p.value}</td>`;
        parametersTableBody.appendChild(tr);
      });
    }

    function fetchLocation() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(success, error);
      } else {
        locationInfo.textContent = "❌ Geolocation not supported by browser.";
      }
      function success(position) {
        const lat = position.coords.latitude;
        const lon = position.coords.longitude;
        fetch(`https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat=$
{lat}&lon=${lon}`)
          .then(res => res.json())
          .then(data => {
            if (data.address) {
              locationInfo.textContent =
                `📍 Location: ${data.address.village || data.address.town ||
data.address.city || ''}, `
                + `${data.address.county || ''}, ${data.address.state || ''}, $
{data.address.country}`;
            } else {
              locationInfo.textContent = "⚠️ Location details not found.";
            }
          })
          .catch(() => { locationInfo.textContent = "⚠️ Unable to fetch location
details."; });
      }
      function error() {
        locationInfo.textContent = "⚠️ Location not available.";
      }
    }

    function updateRandomPatient() {
      const name = indianNames[randomInt(0, indianNames.length - 1)];
      const age = randomInt(1, 90);
      const { bedNumber, parameters } = generateParameters();
      renderResults(name, age, bedNumber, parameters);
      const allNormal = parameters.every(p => p.normal);
      summaryDiv.textContent = allNormal ? "✅ All parameters are normal. Stay
```

```
healthy!" : "⚠️ Warning: Some parameters are abnormal. Alert sent to doctor!";
      summaryDiv.className = allNormal ? "summary normal" : "summary abnormal";
      resultsDiv.style.display = 'block';
      fetchLocation();
    }

    // ✅ Form submit → show entered details first
    form.addEventListener('submit', (e) => {
      e.preventDefault();
      const name = document.getElementById('name').value;
      const age = document.getElementById('age').value;

      // Only show name & age initially
      patientInfoDiv.textContent = `Patient: ${name}, Age: ${age}`;
      parametersTableBody.innerHTML = '';
      summaryDiv.textContent = "⏳ Waiting for monitoring data...";
      summaryDiv.className = "summary";
      resultsDiv.style.display = 'block';

      // After 10 seconds → start random updates
      setTimeout(() => {
        updateRandomPatient();
        setInterval(updateRandomPatient, 10000);
      }, 10000);
    });

    document.getElementById('resetBtn').addEventListener('click', () => {
      form.reset();
      resultsDiv.style.display = 'none';
      summaryDiv.textContent = '';
      summaryDiv.className = "summary";
    });

    async function fetchGoogleTime() {
      try {
        const response = await
fetch("https://worldtimeapi.org/api/timezone/Asia/Kolkata");
        const data = await response.json();
        const dateTime = new Date(data.datetime);
        const dayName = dateTime.toLocaleDateString('en-US', { weekday:
'long' });
        const dateStr = dateTime.toLocaleDateString('en-US', { year: 'numeric',
month: 'long', day: 'numeric' });
        const timeStr = dateTime.toLocaleString('en-US', {
          hour: 'numeric', minute: 'numeric', second: 'numeric', hour12: true
        });
        document.getElementById('currentTime').innerHTML =
          `🕒 Internet Time: ${timeStr} | 📅 ${dayName}, ${dateStr}`;
      } catch (error) {
        document.getElementById('currentTime').innerHTML = "⚠️ Using local
time.";
      }
    }
    setInterval(fetchGoogleTime, 1000);
    fetchGoogleTime();
  </script>
</body>
</html>
```