

**KGiSL INSTITUTE OF TECHNOLOGY**

(Approved By AICTE, New Delhi, Affiliate to Anna University

Recognized by UGC, Accredited by NBA(IT)

265, KGISL Campus, Thudiyalur Road, Saravanampatti, Coimbatore-641035**.)**

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**NAAN MUDHALVAN - INTERNET OF THINGS**

**NOISE POLLUTION MONITORING**

**NAME: JOSHUA PAUL.M**

**REG NO:** 711721243307

**NM ID:** au21AIA70

**TEAM MENTOR:** Mr**.** Mohankumar M

**TEAM EVALUATOR:** Ms. Akilandeeshwari M

**Phase 4: Development Part 2**

**Building a Noise pollution Monitoring system using IoT sensors and Raspberry Pi integration**

**PROBLEM STATEMENT**

The IoT-Based Noise Pollution Monitoring System project aims to design, develop  
and deploy an innovative, cost-effective solution for monitoring and mitigating  
noise pollution in urban and residential environments. Noise pollution, often  
underestimated in its impact on public health and quality of life, is a growing  
concern in densely populated areas. This project seeks to address this issue by  
creating a comprehensive noise monitoring infrastructure that leverages the  
capabilities of the Internet of Things (IoT) technology.  
  
**Procedure:**

**Step 1:**

Define Project Requirements and Scope - Determine the goals and objectives of your noise monitoring application. - Identify the target audience and stakeholders. - Define the scope of the project, including the specific features and functionalities you want to include**.**

**Step 2:**

Plan and Design - Create a project plan, outlining tasks, milestones, and timelines. - Design the user interface (UI) and user experience (UX) for your web application. - Determine the technologies and tools you'll use for development**.**

**Step 3:**

Set Up the Backend - Choose a server-side technology (e.g., Node.js, Python, Ruby on Rails) and a database (e.g., MongoDB, PostgreSQL) for your backend. - Implement the server, including API routes for data collection and retrieval. - Set up a database to store noise data. - Implement data validation and error handling on the server.

**Step 4:**

Choose Hardware or APIs - Decide whether you will use physical sensors or integrate with noise monitoring APIs to collect real-time noise data. - If using hardware, select appropriate sensors and equipment, and connect them to your backend.

**Step 5:**

Develop the Backend - Implement the API endpoints for receiving noise data from sensors or external APIs. - Implement data storage and retrieval functions in the backend.

**Step 6:**

Set Up Authentication and Security - Implement user authentication if your application requires it. - Apply security best practices to protect your application and data.

**Step 7:**

Frontend Development - Develop the user interface using HTML, CSS, and JavaScript. - Create the user interface for displaying real-time noise data and historical data. - Implement interactive features, such as start/stop monitoring, charts, and maps.

**Step 8:**

Real-Time Data Display - Integrate the real-time noise data from sensors or APIs into the frontend. - Use JavaScript libraries or frameworks to update the noise level in real-time on the UI.

**Step 9:**

Data Visualization - Implement charts and graphs to visualize historical noise data. - Use charting libraries like Chart.js or D3.js for data visualization.

**Step 10:**

Deployment - Deploy your backend server and database to a hosting service (e.g., AWS, Heroku). - Deploy the frontend to a web server or hosting service. - Configure domain and DNS settings if you have a custom domain**.**

**Step 11:**

Testing and Quality Assurance - Thoroughly test your application for functionality, performance, and security. - Debug and fix any issues or errors. - Conduct user acceptance testing (UAT) with stakeholders and potential users.

**Step 12:**

User Documentation - Create user documentation and guides on how to use the noise monitoring application. - Provide troubleshooting instructions.

**Step 13:**

Compliance and Legal Considerations - Ensure that your application complies with any relevant laws and regulations related to noise monitoring. - Consider privacy policies and data handling regulations.

**Step 14:**

Monitoring and Maintenance - Set up monitoring tools to keep an eye on server performance and data quality. - Regularly update and maintain your application to address issues and add new features.

**Step 15:**

User Training and Education (if necessary) - Provide training or educational resources to users who will be interacting with the noise monitoring system. Please note that this is a general guide, and the actual implementation can vary depending on your specific project requirements, available resources, and expertise. Building a comprehensive noise monitoring web application can be a complex task, and you may need to consult with experts in relevant fields, such as environmental monitoring or web development, to ensure the success of your project.

**SOURCE CODE:**

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Noise Pollution Monitor</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <h1>Noise Pollution Monitor</h1>

    <div id="noise-level">

        <h2>Noise Level (dB):</h2>

        <p id="noise-value">0</p>

    </div>

    <button id="start-button">Start Monitoring</button>

    <button id="stop-button">Stop Monitoring</button>

    <script src="script.js"></script>

</body>

</html>

**CSS:**

body {

    font-family: Arial, sans-serif;

    text-align: center;

    background-color: #f4f4f4;

}

h1 {

    color: #333;

}

#noise-level {

    margin: 20px;

    background-color: #fff;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

}

button {

    padding: 10px 20px;

    margin: 10px;

    font-size: 16px;

    background-color: #0074d9;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

button:hover {

    background-color: #0056b3;

}

**JAVASCRIPT:**

let isMonitoring = false;

let noiseValue = 0;

let noisePollutionMonitor;

const noiseValueDisplay = document.getElementById('noise-value');

const startButton = document.getElementById('start-button');

const stopButton = document.getElementById('stop-button');

function startMonitoring() {

    isMonitoring = true;

    noisePollutionMonitor = setInterval(updateNoiseValue, 1000); // Update noise level every second

    startButton.disabled = true;

    stopButton.disabled = false;

}

function stopMonitoring() {

    isMonitoring = false;

    clearInterval(noisePollutionMonitor);

    startButton.disabled = false;

    stopButton.disabled = true;

}

function updateNoiseValue() {

    // Simulate getting noise data from a sensor or API

    noiseValue = Math.floor(Math.random() \* 101); // Random value between 0 and 100 dB

    noiseValueDisplay.textContent = noiseValue;

}

startButton.addEventListener('click', startMonitoring);

stopButton.addEventListener('click', stopMonitoring);

stopButton.disabled = true; // Initially disable the "Stop Monitoring" button

// You would typically integrate this with real noise monitoring hardware or APIs for a production application.

**CONCLUSION:**

In summary, developing a noise monitoring web app involves:

**Planning:** Clearly define project goals and create a roadmap.

**Backend Setup:** Build a secure backend for data storage and management.

**Data Sources:** Connect to sensors or APIs for real-time noise data.

**User Interface:** Design an intuitive UI for data display and visualization.

**Deployment:** Host the app and configure server settings.

**Testing:** Thoroughly test for functionality and security.

**Documentation**: Create user guides and comply with regulations.

**Maintenance:** Continuously monitor and update the app.

**Expertise:** Seek professional advice when needed.

Developing a noise monitoring app is a complex process, but it can lead to valuable insights for addressing noise pollution.