KINGS ENGINEERING COLLEGE

**PROJECT TITTLE**: NOISE POLLUTION MONTIOR (IOT\_PHASE4)

**BATCH MEMBERS**: INDRIS.P, IMMANUEL JOSHUA.P, MOHAN BABU.R, PRASANNA KUMAR.S, GUNASEKAR.V

**DEPARTMENT:** B. TECH ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

**MENTOR NAME:** MR. SUNDAR RAJ V

# Front-End Development:

# HTML/CSS/JavaScript: Create the user interface for your platform. This includes designing the dashboard, charts, and other elements for visualizing water consumption data.

# React, Vue, or Angular: Consider using a modern JavaScript framework to build interactive and responsive user interfaces.

# Back-End Development:

# Node.js, Python, or Java: Choose a server-side technology to handle data processing, user authentication, and communication with IoT devices.

# Database: Select a database system (e.g., MongoDB, PostgreSQL) to store real-time water consumption data securely.

# RESTful or GraphQL APIs: Create APIs for your front-end to communicate with the back-end, allowing data retrieval and updates.

# 1. Project Planning:

# - Define the scope and objectives of your platform.

# - Determine your target audience and what features you want to include, such as real-time noise level displays, maps, and educational content.

# 2. Data Sources:

# - Identify sources of real-time noise data. This may involve using environmental sensors, APIs from environmental agencies, or user-generated noise reports.

# 3. Technology Stack:

# - Choose your web development technologies, such as HTML, CSS, and JavaScript. You might also consider using libraries or frameworks like React or Vue.js for building dynamic user interfaces.

# 4. Design the User Interface:

# - Create wireframes and designs for your platform, ensuring that it's user-friendly and easy to navigate.

# 5. HTML and CSS:

# - Write HTML and CSS code to build the structure and style of your platform. This includes creating pages, layouts, and responsive design for mobile users.

# 6. JavaScript and Real-Time Data:

# - Use JavaScript to fetch and display real-time noise data. You can use technologies like Web Sockets, server-sent events, or AJAX to update noise levels in real time.

# 7. Interactive Maps:

# - If you plan to include maps, consider using a mapping library like Leaflet or Google Maps API to visualize noise data geospatially.

# 8. User Reporting:

# - Create a form or interface for users to report noise levels if your platform supports user-generated content.

# 9. Data Visualization:

# - Develop charts or graphs to present noise data in a clear and understandable manner.

# 10. Notifications and Alerts:

# - Implement a system for notifying users of significant noise level changes or events.

# 11. Testing and Debugging:

# - Thoroughly test your platform to ensure that it works correctly on various devices and browsers.

# 12. Security and Privacy:

# - Ensure the security of your platform and protect user data if you're collecting any personal information.

# 13. Deployment:

# - Host your platform on a web server or cloud hosting service.

# 14. Domain and DNS:

# - Register a domain name if desired and configure DNS settings to point to your platform.

# 15. Monitoring and Maintenance:

# - Set up monitoring to track the performance and stability of your platform and establish a maintenance plan to address issues promptly.

# 16. User Education and Outreach:

# - Provide educational content about noise pollution on your platform and reach out to your target audience to promote it.

# PROGRAM CODE:

# 1. HTML (index.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 Platform</title>

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

</head>

<body>

<header>

<h1>Noise Pollution Data</h1>

</header>

<main>

<div id="noise-level">

<h2>Real-Time Noise Level</h2>

<p id="noise-value">Loading...</p>

</div>

</main>

<script src="app.js"></script>

</body>

</html>

**2. CSS (styles.css):**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

background-color: #333;

color: #fff;

text-align: center;

padding: 20px;

}

main {

max-width: 800px;

margin: 0 auto;

padding: 20px;

}

#noise-level {

border: 1px solid #ccc;

padding: 20px;

}

#noise-value {

font-size: 24px;

}

**3. JavaScript (app.js):**

// Simulated real-time noise data (for demonstration)

function getRandomNoiseLevel() {

return (Math.random() \* 100).toFixed(2);

}

function updateNoiseLevel() {

const noiseValueElement = document.getElementById("noise-value");

// Fetch real-time noise data here and update the value

const noiseLevel = getRandomNoiseLevel();

noiseValueElement.textContent = `${noiseLevel} dB`;

}

// Update noise level every 5 seconds (for demonstration)

setInterval(updateNoiseLevel, 5000);

// You would typically use AJAX or WebSockets to fetch real data from sensors or APIs.