NOISE POLLUTION MONITORING

PHASE 4 DEVELOPMENT PART 2

INTRODUCTION :

A noise monitor is used to measure and evaluate the levels or characteristics of various types of environmental noise. Complying with ISO 1996-2 standards, noise monitors are designed to assess primary noise sources such as road traffic, rail traffic, air traffic, and industrial plants.

TABLE OF CONTENT :

1. Define Project Requirements:

Clearly outline the requirements and objectives of your noise pollution monitoring platform. What kind of data do you want to collect? What are the specific features you need?

2. Select Hardware and Sensors:

Choose the appropriate hardware and sensors for measuring noise pollution. This may include microphones, sound level meters, or other specialized equipment.

3. Data Collection and Analysis:

Develop the software to collect data from the sensors. This could involve real-time data streaming or periodic data logging. Implement algorithms for noise level analysis.

4. Data Visualization:

Create a user-friendly interface for visualizing the collected data. This could be a web-based dashboard, a mobile app, or desktop software. Use charts, maps, and graphs to present data effectively.

5. Data Storage and Management:

Set up a database to store historical noise pollution data. Consider using a database system that is suitable for time-series data.

6. User Authentication and Security:

Implement user authentication to restrict access to authorized personnel. Ensure that data security and privacy are maintained.

7. Alerting and Notification System:

Develop an alerting system that can notify relevant parties when noise pollution levels exceed predefined thresholds.

8. Integration with IoT and GIS:

If applicable, integrate your platform with Internet of Things (IoT) devices for remote monitoring. Consider integrating Geographic Information Systems (GIS) for spatial analys**is.**

9. Testing and Quality Assurance:

Thoroughly test your platform to ensure data accuracy, system reliability, and usability. Perform both unit testing and system testing.

10. Deployment and Maintenance:

Deploy the platform in the intended monitoring locations. Establish a maintenance plan to keep the sensors and software up-to-date and operational.

11. Documentation:

Create comprehensive documentation that explains how to set up, use, and maintain the platform. This documentation will be crucial for onboarding new users and maintaining the system.

12. Compliance and Regulations:

Ensure that your platform complies with local noise pollution monitoring regulations and standa**rds.**

13. Data Analysis and Reporting:

Develop tools for in-depth data analysis and reporting, which can be valuable for research or policy-making.

14. Community Engagement:

Consider involving local communities in your project. Make data accessible to the public to raise awareness and encourage participation in noise pollution reduction efforts.

15. Scaling:

Plan for the scalability of your platform to accommodate future growth and additional monitoring locations.

16. Feedback and Iteration:

Continuously gather feedback from users and stakeholders to make improvements and add new features.

CREATING A PLATFORM USING WEB DEVELOPMENT TECHNOLOGIES (HTML,

CSS, AND JAVASCRIPT)

PYTHON SCRIPT :

```
import time
import requests
import sound_sensor_library # Replace with the actual library for your noise sensor
# Configuration
SENSOR_ID = "sensor_001" # Unique identifier for the sensor
```

```
API_ENDPOINT = "https://your-api-endpoint.com/data" # Replace with your API endpoint

# Initialize the noise sensor (use the appropriate library or code for your specific sensor)

sound_sensor = sound_sensor_library.initialize()

# Function to read noise level

def read_noise_level():

 noise_level = sound_sensor.get_noise_level() # Replace with the appropriate method

 return noise_level

# Function to send data to the API

def send_data_to_api(data):

 headers = {"Content-Type": "application/json"}

 payload = {

 "sensor_id": SENSOR_ID,

 "timestamp": int(time.time()),

 "noise_level": data,

 }

 try:

 response = requests. Post(API_ENDPOINT, json=payload, headers=headers)

 if response.status_code == 200:

 print("Data sent successfully")

 else:

 print("Failed to send data. Status code:", response.status_code)

 except Exception as e:

print("Error:", str(e))

# Main loop

while True:
```

noise_level = read_noise_level()

send_data_to_api(noise_level)

time. Sleep(60) # Send data every 60 seconds (adjust as needed)Send Data to Noise Pollution Platform

:

Replace

"https://your-noise-platform.com/api/send_data" with the actual URL of your noise pollution information platform. Adjust the payload structure as per the API requirements of the platform.

Run the Script:

Execute the Python script on your IoT device, and it will record noise level data and send it to the

platform in real-time.

Data Processing on the Platform:

On the noise pollution information platform, make sure you have a way to receive and process the data sent by the IoT device.

JavaScript Noise Pollution Monitoring

Audio Input: Capture audio from the microphone using Web Audio API. You can create an audio context and use the getUserMedia stream as an audio source.

Analysing Sound: Analyzed the audio data to measure noise levels. You can use various techniques, such as Fast Fourier Transform (FFT) to convert the audio signal into frequency domain data.

Thresholds: Define noise pollution thresholds or criteria. Depending on your project, you might consider certain sound levels as noise pollution.

Real-time Monitoring: Continuously monitor the noise levels and update the user interface in real-time. You can use JavaScript's requestAnimationFrame to update the display.

Data Visualization: Create charts or graphs to visually represent noise levels over time. You can use JavaScript libraries like Charts or D3.js for this purpose.

Alerts and Notifications: Implement alerts or notifications when noise pollution exceeds predefined thresholds. You can use browser notifications or custom pop-ups.

Data Storage: Optionally, you can store the noise pollution data in a database or local storage for historical tracking and analysis.

Geolocation: If you want to monitor noise pollution at specific locations, you can use the Geolocation API to associate noise data with geographic coordinates.

User Interface: Design a user-friendly interface to display noise pollution information, charts, and alerts.

JavaScript

```
function issue Notification(message) {
 if ('Notification' in window) {
 if (Notification. Permission === 'granted') {
 new Notification('Noise monitoring ', { body: message });
 } else if (Notification.permission !== 'denied') {
 Notification.requestPermission()
 .then(permission => {
 if (permission === 'granted') {
 new Notification(‘noise monitoring', { body: message });
 }
 });
 }
 }
```

Testing: Thoroughly test your application on different browsers and devices to ensure compatibility

Deployment: Host your web application on a web server or deploy it using a cloud platform, making it accessible to us.

CONCLUSION:

Noise pollution can cause health issues such as hearing loss, mental health, increased anxiety and stress, stress-induced issues such as heart diseases. For all the negative impacts that noise pollution can have it's important that local areas take action to reduce it.

SUBMITTED BY

S.KAVIYA

311421106026