# ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE- 641046

## NAAN MUDHALVAN COURSE

## PHASE 5

| | |
|---|---|
| NAME | ROOBASRI.B |
| NAAN MUDHALVAN REGISTER NO. | au710021106031 |
| DOMAIN | CLOUD COMPUTING |
| PROJECT | URBAN NOISE MAPPING SYSTEM |

### INTRODUCTION:

Noise is being recognised as a serious environmental problem, and one which must be accounted for in a sustained development policy, which is designed to improve the quality of life for citizens. European cities have developed in the recent past mostly around their historic centres. The fast social and economical growth in the 20th century was not always accompanied by adequate land planning and environment management measures.

## NOISE:

Noise refers to any unwanted, unpleasant, or random sound or disturbance that interferes with the normal transmission or perception of a signal. It can manifest as background sounds, static on a phone line, or disruptions in various forms, such as in audio, data, or even environmental contexts. Noise can be a nuisance in communication, technology, and daily life, and efforts are often made to reduce or eliminate it in various applications.

## NEED FOR NOISE MANAGEMENT:

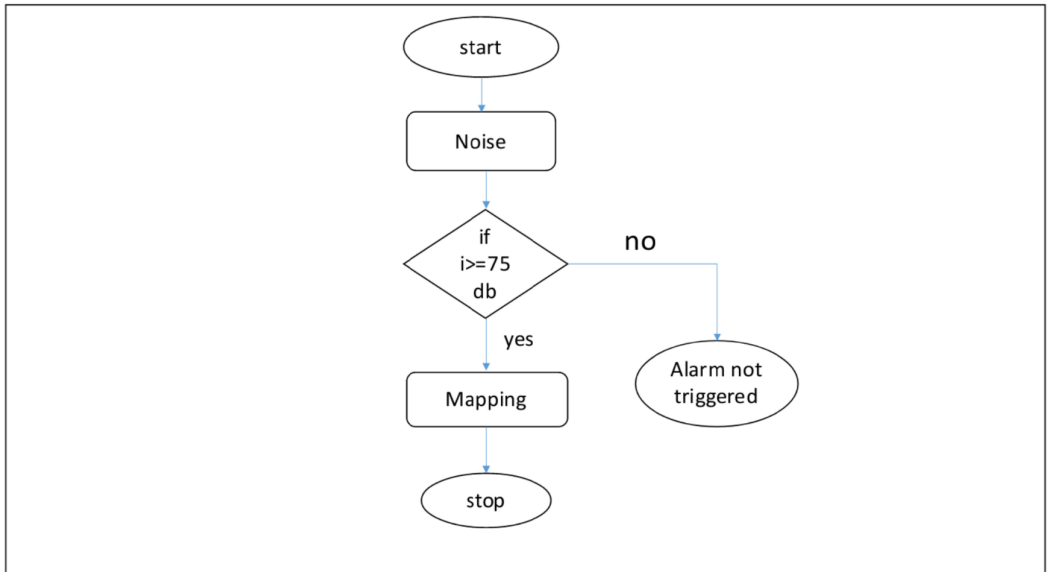
Constant exposure to elevated levels of noise can lead to serious mental and physical health issues. Industrial noise pollution is causing many problems for living beings. Many industrial noise control products are available nowadays. These products help in reducing noise, resonance, and echo in any area.

- Hearing Loss
- Stress and Anxiety
- Sleep Disturbances
- Cardiovascular Effects
- Communication Interference
- Impact on Learning and Performance
- Workplace Productivity
- Quality of Life
- Social and Recreational Disruptions
- Effects on Wildlife

## NOISE MAPPING

Noise mapping is a technique used to create visual representations of noise levels in a specific area. It involves collecting data on noise pollution from various sources, such as traffic, industry, and other environmental factors, and then using this data to generate maps that show noise levels in different parts of the area. These maps can be valuable for urban planning, environmental impact assessments, and identifying areas with high noise pollution that may require mitigation measures


## Flow chat




## KEY CLOUD COMPUTING COMPONENTS:

To understand the project, it's essential to recognize the key cloud computing components involved:

### 1.SENSORS AND DATA SOURCES

These are the physical devices that capture noise data, such as microphones or sound level meters. Data is collected from various sources across a

geographic area.


## 2. DATA INGESTION

A component that collects and processes data from the sensors, which is then transmitted to the cloud. This could involve data preprocessing and filtering.

## 3. CLOUD INFRASTRUCTURE

The core of the system, including virtual servers, storage, and databases hosted on cloud platforms like AWS, Azure, or Google Cloud. This is where data is stored and processed.

## 4.GIS

Integration with GIS tools for spatial mapping and visualization. This

helps in creating noise maps and geospatial analysis.

## 5. Cloud Services


Cloud services encompass a wide range of offerings, from computing power to databases, data storage, and machine learning. These services are designed to simplify the development and deployment of IoT applications. For IoT data processing, services like AWS Lambda, IBM Cloud Functions, and Azure Functions are examples of serverless compute platforms.

## 6. Scalability

The ability to scale resources up or down based on demand, which is a significant advantage of cloud computing.

## 7.COST MANAGEMENT


Monitoring and optimizing cloud computing costs, as they can escalate with increased data and usage.

## CODE IN PYTHON FLASK TO MONITOR THE NOISE

```
app = Flask(__name__)

# Simulated noise data (you would replace this with real data from sensors)
Def  get_noise_data():
  return random.uniform(40, 80)  # Simulated noise level between 40
and 80 dB

@app. route('/')
Def  index():
  return render_template('index.html')

@app. route('/get_ noise_data')
Def  get_noise_ data_ route():
  noise_ level = get_noise_ data()
  return j sonify ({'noise_level': noise_ level})

if __name__ == '__main__':
  app. run(debug=True)
```

## EXPLANATION FOR THE CODE:

### 1.Import necessary modules:

`from flask import Flask, render_template, request, jsonify`: This line imports the required modules from Flask, a micro web framework for Python. `Flask` is the main class for creating the web application, `render_template` is used to render HTML templates, `request` is used to handle HTTP requests, and `jsonify` is used to send JSON responses.

## 2. Create a Flask application instance:

`app = Flask(__name__)`: This line creates an instance of the Flask web application. `__name__` is a built-in Python variable that represents the name of the current module.

## 3. Define a function to simulate noise data:

`def get_noise_data()`: This function is a placeholder for collecting or generating noise data. In this example, it generates random noise data between 40 and 80 decibels using the `random.uniform` function. You would replace this with actual data from noise sensors in a real-world application.

## 4. Create a route for the root URL ('/'):

`@app.route('/')`: This is a decorator that associates the `/` URL path with the following function. When a user visits the root URL of the web application, it will execute the `index` function.

## 5. Define the 'index' function:

`def index()`: This function is responsible for rendering an HTML template. In this case, it renders the 'index.html' template.

## 6. Create a route for getting noise data ('/get_noise_data'):

`@app.route('/get_noise_data')`: This decorator associates the '/get_noise_data' URL path with the following function. This route is used to

retrieve noise data from the server asynchronously.

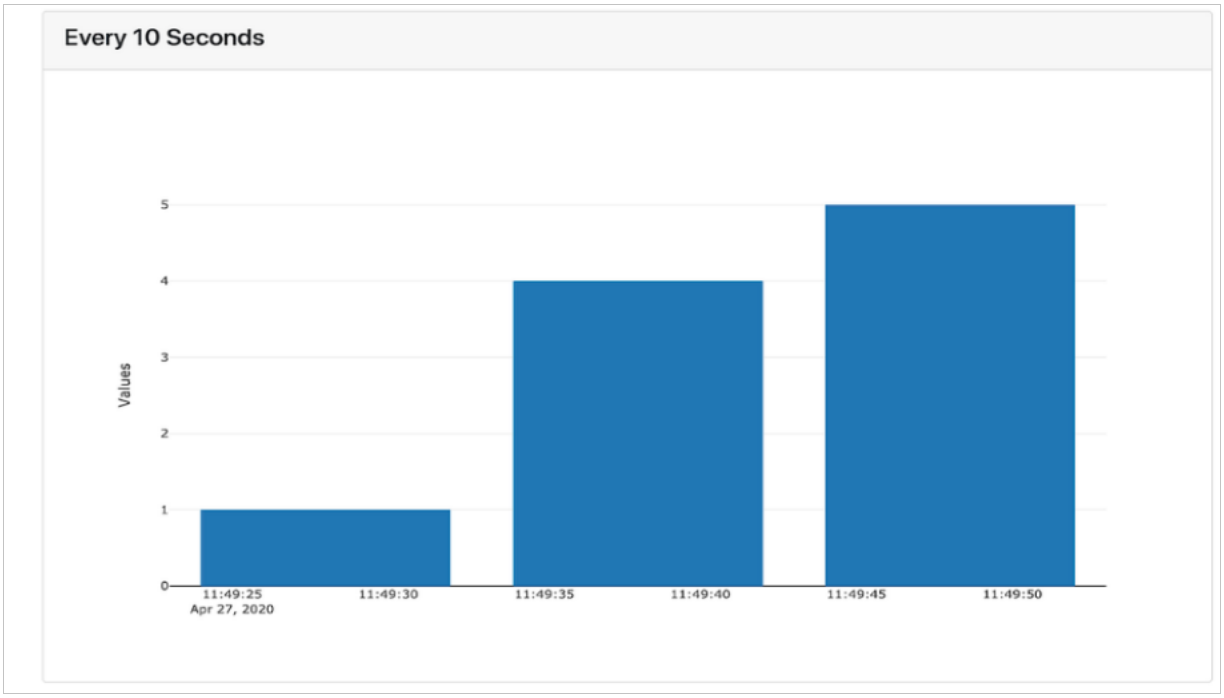
## 7. Define the 'get_noise_data_route' function:

`def get_noise_data_route()`: This function is responsible for generating noise data using the `get_noise_data` function and returning it in JSON format. This data can be fetched by a client-side script to display the current noise level.

## 8. Run the Flask application:

- `if __name__ == '__main__':`: This condition ensures that the application is only run if the script is executed directly (not imported as a module). When you run the script, it starts the Flask development server with debugging enabled.
- `app.run(debug=True)`: This line runs the Flask application on the local server. The `debug=True` argument enables debugging mode, which helps in development by providing detailed error messages.

## OUTPUT:




## DEVELOPMENT TECHNOLOGIES FOR NOISE MONITORING SYSTEM:

Creating an urban noise mapping system is a complex project that involves various web development technologies. Here's a high-level overview of the technologies you might need:

### 1. Front-end:

- HTML/CSS for structuring and styling the user interface.
- JavaScript and frameworks like React or Angular for interactive features.

### 2. Back-end:

- Server-side scripting using languages like Node.js, Python, or Ruby.
- A web framework like Express (for Node.js) or Django (for Python).

### 3. Database:

- A relational database like PostgreSQL or a NoSQL database like MongoDB to store noise data.

4. Geospatial Data:

- Use GeoJSON or other spatial data formats for representing geographical data.
- A geospatial database or library like PostGIS for handling spatial data.

5. Mapping:

- A mapping library like Leaflet or Mapbox for displaying noise data on maps.
- APIs like Google Maps or OpenStreetMap for geolocation services.

6. Noise Sensors:

- Integration with IoT devices or sensors to collect real-time noise data.

7. User Authentication and Authorization:

- Implement user registration and login systems to control access to the system.

8. Data Analysis:

- Use data analysis and visualization libraries like D3.js for presenting noise data insights.

9. API Development:

- Create RESTful APIs to expose data and functionalities to other applications.

10. Hosting and Deployment:

- Choose a hosting platform (e.g., AWS, Azure, Heroku) and deploy your system.
- Set up CI/CD pipelines for automated deployment.

11. Security:

- Implement security measures to protect user data and system integrity.

12. User Interface Design:

- Ensure a user-friendly design for your web application.

## CODE FOR WEBPAGE

```
<!DOCTYPE html>
<html>
<head>
  <title>Noise Detection</title>
</head>
<body>
  <button id="startButton">Start Noise Detection</button>
  <div id="status">Status: Inactive</div>
  <div id="noiseIndicator"></div>

  <script>
    const startButton = document.getElementById('startButton');
    const statusElement = document.getElementById('status');
    const noiseIndicator = document.getElementById('noiseIndicator');
    let audioContext;
    let analyser;

    startButton.addEventListener('click', () => {
      startButton.disabled = true;
      statusElement.textContent = 'Status: Active';

      // Initialize audio context and getUserMedia to capture audio from
the microphone
      audioContext = new (window.AudioContext ||
window.webkitAudioContext)();
      navigator.mediaDevices.getUserMedia({ audio: true })
        .then((stream) => {
          const microphone =
audioContext.createMediaStreamSource(stream);

          // Create an AnalyserNode to analyze audio levels
          analyser = audioContext.createAnalyser();
          analyser.fftSize = 256;
          microphone.connect(analyser);

          // Implement noise detection logic
          const bufferLength = analyser.frequencyBinCount;
```

```
            const dataArray = new Uint8Array(bufferLength);

            function detectNoise() {
              analyser.getByteFrequencyData(dataArray);
              const average = dataArray.reduce((acc, val) => acc + val, 0) /
bufferLength;

              // Adjust this threshold as needed
              const threshold = 100;
              noiseIndicator.style.backgroundColor = average > threshold ?
'red' : 'green';
              requestAnimationFrame(detectNoise);
            }

            detectNoise();
          })
          .catch((error) => {
            console.error('Error accessing microphone:', error);
          });
      });
    </script>
</body>
</html>
```

## EXPLANATION FOR THE CODE:

This HTML and JavaScript code is for creating a simple web page that allows you to start noise detection using a computer's microphone. It uses the Web Audio API and the get User Media API to access the microphone and analyze the audio input to detect noise levels. Here's an explanation of the code:

### 1. HTML Structure:

- `<!DOCTYPE html>`: This is the document type declaration, specifying that this is an HTML5 document.
- `<html>`: The root HTML element.
- `<head>`: The head section of the HTML document where you can define metadata and include external resources. In this case, it contains the title of the web page.
- `<title>`: Sets the title of the web page to "Noise Detection."
- `<body>`: The body of the HTML document where the visible content is placed.

## 2. Page Elements:

- `<button id="startButton">Start Noise Detection</button>`: This button element with the ID "startButton" is used to start the noise detection process.
  - `<div id="status">Status: Inactive</div>`: This div element with the ID "status" displays the status of noise detection. It initially shows "Status: Inactive."
  - `<div id="noiseIndicator"></div>`: This div element with the ID "noiseIndicator" will change its background color to indicate noise levels.

## 3. JavaScript Code:

- The JavaScript code is enclosed within a `<script>` tag and runs after the page is loaded.

## 4. Event Listener:

- `startButton` is a reference to the button element with the ID "startButton."
  - An event listener is attached to the `startButton` element, so when it is clicked, the noise detection process is initiated.

## 5. Noise Detection Logic:

- When the button is clicked, it is disabled (`startButton.disabled = true`) to prevent multiple clicks.
  - The text content of the `statusElement` (with the ID "status") is changed to "Status: Active."

  - An `AudioContext` is created. This is part of the Web Audio API and is used to handle audio operations.

  - `navigator.mediaDevices.getUserMedia({ audio: true })` is used to request access to the computer's microphone. If access is granted, a microphone audio stream is obtained.

- An `AnalyserNode` is created to analyze the audio levels. The `fftSize` property is set to 256, determining the size of the Fast Fourier Transform used for frequency analysis.

- The microphone audio stream is connected to the `analyser`.

- A function `detectNoise` is defined to perform noise detection.

- The `analyser` retrieves audio frequency data into a `Uint8Array` called `dataArray`.

- The average value of the data in `dataArray` is calculated, giving an indication of the audio level.

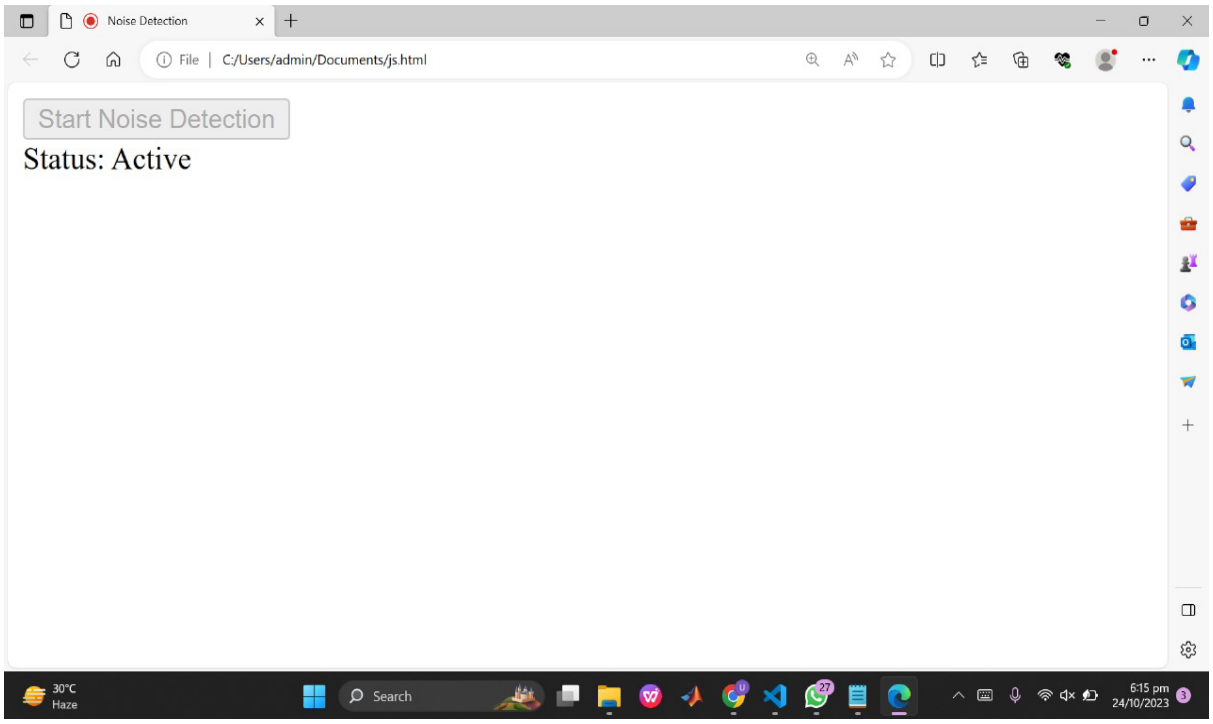
- A threshold value is set (currently 100) to determine what constitutes "noise." If the average value is greater than the threshold, the background color of the `noiseIndicator` div changes to red; otherwise, it changes to green.

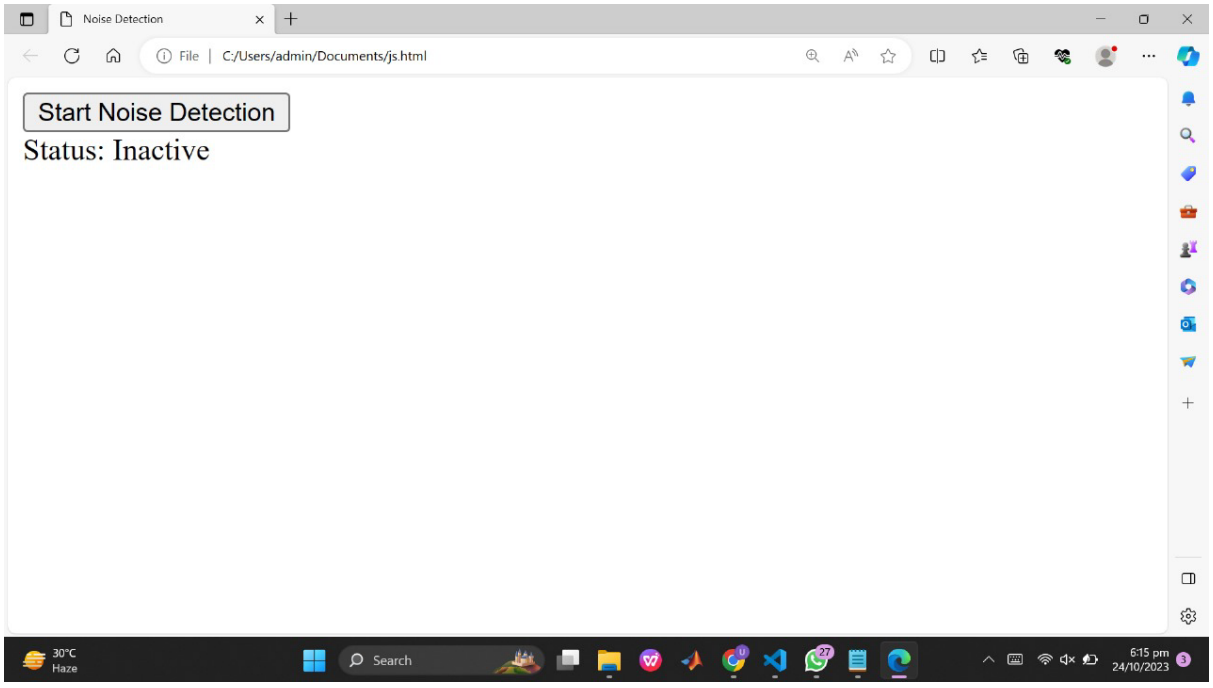
- The `requestAnimationFrame` function is used to continually call the `detectNoise` function, creating a real-time noise monitoring effect.

- If there is any error in accessing the microphone, an error message is logged to the console.

This code provides a simple web interface for monitoring noise levels from the computer's microphone. When you click the "Start Noise Detection" button, it activates the microphone and continuously updates the background color of the `noiseIndicator` to reflect the current noise level.

## OUTPUT:


Noise Detection
File | C:/Users/admin/Documents/js.html
Start Noise Detection
Status: Active
30°C
Haze
Search
6:15 pm
24/10/2023


Noise Detection
File | C:/Users/admin/Documents/js.html
Start Noise Detection
Status: Inactive
30°C
Haze
Search
6:15 pm
24/10/2023

# CONCLUSION:

Noise in cities has increased in the past decades, due to a growing urban development. In the last century, population movement to the greater cities, disorder planned city development and increase of the motor vehicle in the traffic have been produced noise pollution and other environmental problems. Management and reduction or urban noise has been called for in urban development plans. Noise community ordinances have been approved at national and local levels in various countries of the world. The comprehensive system for real-time noise monitoring is an effective tool for advancing urban quality of life.

.

.