ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE- 641046

NAAN MUDHALVAN COURSE

PHASE 3

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.

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.

NOISE:

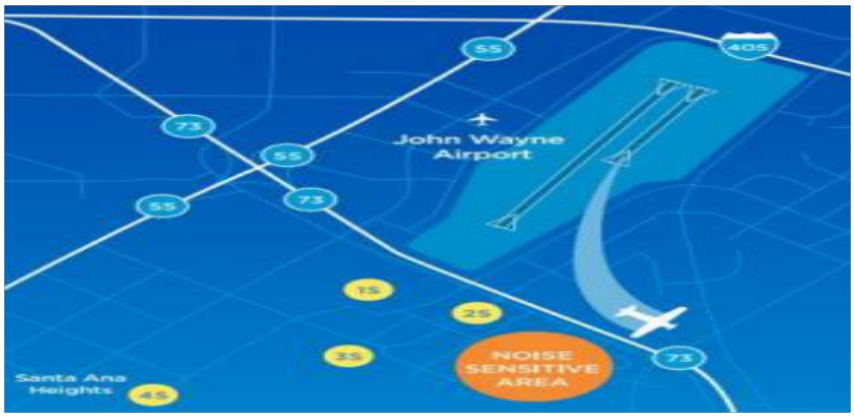
Noise refers to any unwanted, unpleasant, or random sound or disturbance that interferes with the normal transmission or perception of a signal. It can manifest as background sounds, static on a phone line, or disruptions in various forms, such as in audio, data, or even environmental contexts. Noise 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.

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



CODE

```
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 jsonify ({'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.

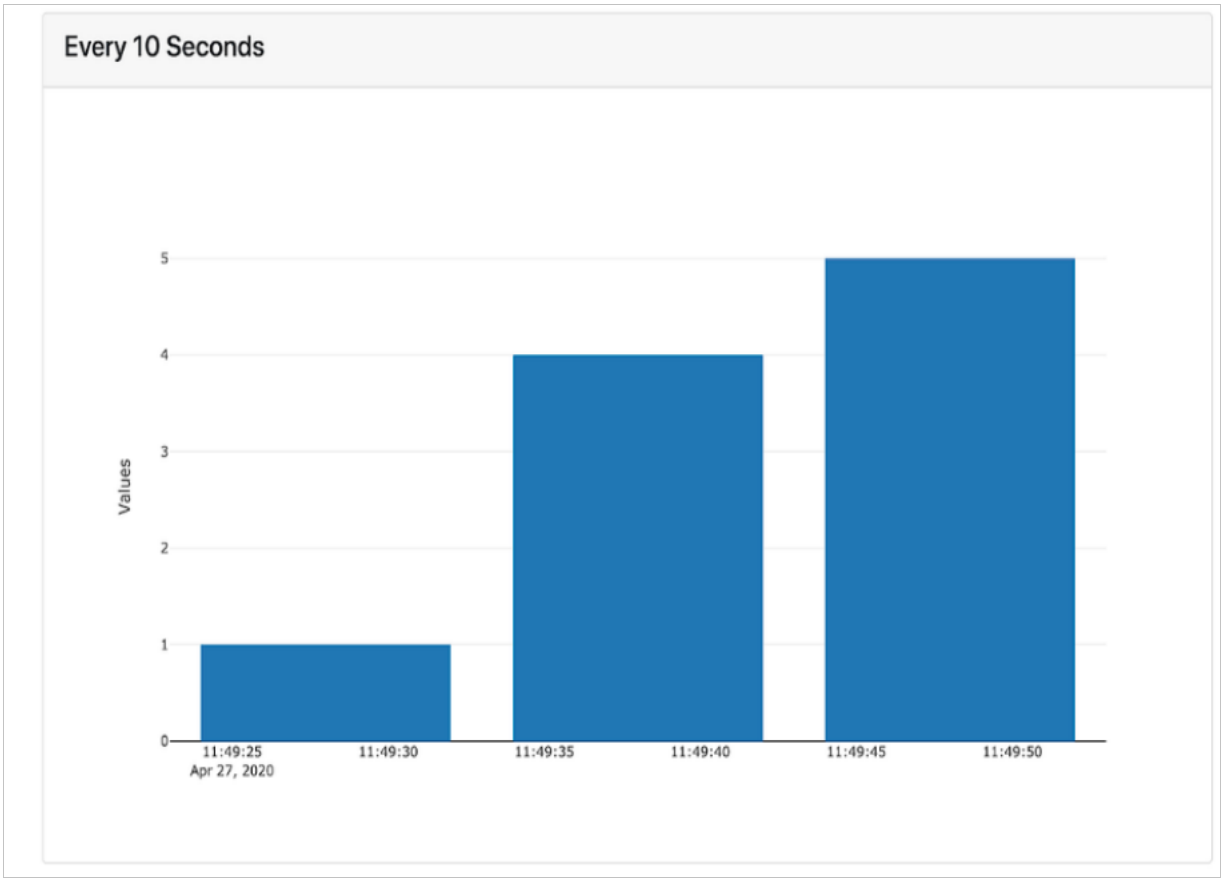
8. Run the Flask application:

- `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:



CONCLUSION:

In this we can monitor the noise in software by using the application python flask and we can store the collected data's of noise in the cloud then we make a noise mapping methodology.