**PHASE 4:DEVELOPMENT PART2**

**INDOOR AIR QUALITY MONITORING**

**MODEL PERFORMANCE TESTING:**

The existing air quality monitoring systems generally monitor PM 2.5 along with temperature and humidity but the proposed system focuses on the multiple air pollutants including Carbon Dioxide (CO2), Particulate Matter (PM) 2.5, Nitrogen Dioxide (NO2), Carbon Monoxide (CO), Methane (CH4), temperature and humidity. The predictive analysis of air quality based on these pollutants make the proposed system more attractive and reliable as compared to the existing systems. • The web portal is developed which provide real time data of air quality which further helps to compute Air Quality Index (AQI), since time series data of 24 hour is required to compute AQI.

**SIMULATION:**

import random

import time

class AirQualitySimulator:

def \_\_init\_\_(self):

self.temperature = 25.0 # in Celsius

self.humidity = 50.0 # in percentage

self.co2\_level = 400 # in parts per million (ppm)

self.pm25\_level = 10 # in micrograms per cubic meter (µg/m³)

self.simulation\_duration = 60 # in seconds

def simulate(self):

start\_time = time.time()

while time.time() - start\_time < self.simulation\_duration:

# Simulate changes in air quality parameters

self.temperature += random.uniform(-1, 1)

self.humidity += random.uniform(-1, 1)

self.co2\_level += random.uniform(-1, 1)

self.pm25\_level += random.uniform(-1, 1)

# Ensure values stay within reasonable ranges

self.temperature = max(0, min(self.temperature, 40))

self.humidity = max(0, min(self.humidity, 100))

self.co2\_level = max(300, min(self.co2\_level, 1000))

self.pm25\_level = max(0, min(self.pm25\_level, 50))

# Display simulated data

self.display\_data()

time.sleep(1)

def display\_data(self):

print(f"Temperature: {self.temperature:.2f}°C | Humidity: {self.humidity:.2f}% | CO2 Level: {self.co2\_level} ppm | PM2.5 Level: {self.pm25\_level} µg/m³")

if \_\_name\_\_ == "\_\_main\_\_":

air\_quality\_simulator = AirQualitySimulator()

air\_quality\_simulator.simulate()

**SIMULATOR CODE :**

+-------------------+ +-------------------------+ +------------------+

| Simulation Script | ---> | Air Quality Simulator | ---> | Display/Logging |

| | | | | |

+-------------------+ +-------------------------+ +------------------+

|

v

+------------------+

| Data Store |

| (e.g., Database) |

+------------------+

**WEB TECHNOLOGIES :**

For building an indoor air quality monitoring system using IoT (Internet of Things), various web technologies can be employed to enable data communication, visualization, and user interaction. Here are some key web technologies commonly used in IoT projects:

1. **Web Development Frameworks:**
   * **Django or Flask (Python):** These frameworks can be used for building the backend of your web application. They provide tools for handling HTTP requests, managing databases, and integrating with IoT devices.
   * **Node.js (JavaScript):** Particularly suitable for projects with a large number of simultaneous connections, Node.js can be used to build a lightweight and efficient backend.
2. **Frontend Frameworks and Libraries:**
   * **React, Angular, or Vue.js:** These frameworks can be used for building dynamic and responsive user interfaces. They are especially useful for creating real-time dashboards and interactive visualizations of air quality data.
   * **Charting Libraries (e.g., Chart.js, D3.js):** Incorporate these libraries to create graphical representations of air quality metrics.
3. **Communication Protocols:**
   * **MQTT (Message Queuing Telemetry Transport):** A lightweight and efficient publish-subscribe messaging protocol that is commonly used in IoT applications for communication between devices and servers.
   * **HTTP/HTTPS:** Traditional web protocols can be used for communication between IoT devices and the server.
4. **Database Systems:**
   * **MySQL, PostgreSQL, MongoDB:** Choose a database system based on the requirements of your project. SQL databases (e.g., MySQL, PostgreSQL) are suitable for structured data, while NoSQL databases (e.g., MongoDB) can handle more unstructured data.
5. **IoT Platforms:**
   * **ThingSpeak, Blynk, Ubidots:** These IoT platforms provide tools and services to facilitate the integration of IoT devices, data storage, and visualization. They often offer APIs for connecting your devices and web application.
6. **Security Measures:**
   * **HTTPS (SSL/TLS):** Ensure secure communication between the IoT devices and the server.
   * **Token-based Authentication:** Implement secure authentication mechanisms to control access to the monitoring system.
7. **Cloud Services:**
   * **AWS IoT, Google Cloud IoT, Microsoft Azure IoT:** These cloud platforms provide scalable and reliable infrastructure for managing IoT devices, storing data, and hosting web applications.
8. **Containerization and Orchestration:**
   * **Docker, Kubernetes:** Use containerization for easy deployment and scaling of your web application.
9. **WebSockets:**
   * **Socket.io (JavaScript), Django Channels (Python):** Implement real-time communication between the server and the web browser for instant updates in air quality data.
10. **Mobile App Development (Optional):**
    * **React Native, Flutter:** If you plan to develop a mobile app for monitoring air quality, these frameworks can be used to create cross-platform applications.

Top of Form