SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19SB602 FULL STACK DEVELOPMENT FOR NEXT GENERATION IOT

III YEAR / VI SEMESTER

Unit IV- **INTEGRATION OF NG IoT WITH WEB DEVELOPMENT**

Topic :IoT and Web Development Work Together

Understanding IoT and Web Development:

IoT refers to a network of interconnected devices that communicate and exchange data over the internet.

Web development involves creating websites or web applications that users can access through web browsers.

Setting up the IoT Device:

- ✓ Choose an IoT device (e.g., Raspberry Pi, Arduino) and connect sensors or actuators to it.
- ✓ Write code on the IoT device to collect data from sensors or control actuators.
- ✓ Set up the IoT device to send data to a server or cloud platform. This could be done using MQTT, HTTP, or other protocols.

Creating a Web Application:

Choose a web development framework or library (e.g., React.js, Angular, Vue.js).

Set up a new project and create necessary components for your web application.

Design the user interface to display data from IoT devices or control them.

Establishing Communication:

Set up a server-side application (e.g., using Node.js, Flask, Django) to handle requests from IoT devices and web clients.

Define API endpoints to receive data from IoT devices and send commands to them.

Implement authentication and authorization mechanisms to secure communication between devices and the server.

Integrating IoT Data with the Web Application:

Use AJAX or WebSocket to fetch real-time data from the server and update the UI accordingly.

Display sensor data on the web interface using charts, graphs, or tables.

Allow users to interact with IoT devices through the web interface (e.g., turning on/off devices, adjusting settings).

Enhancing Functionality:

Implement features like notifications or alerts based on IoT data thresholds.

Incorporate machine learning algorithms to analyze IoT data and provide insights.

Optimize the web application for different devices and screen sizes (responsive design).

Testing and Deployment:

Test the IoT-device-to-server and server-to-web-client communication thoroughly.

Deploy the IoT device code, server-side application, and web application to respective environments (e.g., production server, cloud platform).

Monitoring and Maintenance:

1. Set up monitoring tools to track the performance of the IoT devices and the server.
2. Regularly update and maintain both the IoT device code and the web application to ensure security and reliability.

Example

A temperature monitoring system using a Raspberry Pi (emulated) as the IoT device and a React web application as the frontend.

Create a simple Node.js server to simulate the MQTT broker and RESTful API endpoints

```
// server.js

const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 5000;

// Simulated temperature data
let currentTemperature = 25;

app.use(bodyParser.json());
```

```
// Endpoint to get current temperature
app.get('/api/temperature', (req, res) => {
  res.json({ temperature: currentTemperature });
});

// Endpoint to set temperature threshold
app.post('/api/threshold', (req, res) => {
  const { threshold } = req.body;
  console.log('New threshold set:', threshold);
  res.sendStatus(200);
});

app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

create a React web application to display the temperature and allow users to set a temperature threshold:

```
// App.js

import React, { useState, useEffect } from 'react';
import axios from 'axios';

function App() {
  const [temperature, setTemperature] = useState(0);
  const [threshold, setThreshold] = useState(0);

  useEffect(() => {
    const fetchTemperature = async () => {
      const response = await axios.get('/api/temperature');
      setTemperature(response.data.temperature);
    };
```

```
fetchTemperature();
}, []);

const handleThresholdChange = async () => {
  await axios.post('/api/threshold', { threshold });
  console.log('Threshold updated successfully');
};

return (
  <div className="App">
    <h1>Temperature Monitoring System</h1>
    <div>
      <h2>Current Temperature: {temperature}°C</h2>
    </div>
    <div>
```

```
<input
        type="number"
        value={threshold}
        onChange={(e) => setThreshold(e.target.value)}
        placeholder="Set Threshold"
       />
       <button onClick={handleThresholdChange}>Set
Threshold</button>
     </div>
   </div>
  );
}
export default App;
```

Any Query????

Thank you……