import logo from './logo.svg';

import './App.css';

import React, { useEffect, useState, PureComponent } from "react";

import io from 'socket.io-client'; // Import the socket.io client library

import Stack from '@mui/material/Stack';

import Box from '@mui/material/Box';

import { SparkLineChart } from '@mui/x-charts/SparkLineChart';

import { LineChart } from '@mui/x-charts/LineChart';

import { BarChart } from '@mui/x-charts/BarChart';

import { Gauge } from '@mui/x-charts/Gauge';

import { GaugeComponent } from 'react-gauge-component';

import 'bootstrap/dist/css/bootstrap.css';

import { Container, Row, Col, ResponsiveEmbed,Toast,Button,ToastContainer } from 'react-bootstrap';

import { Sparklines, SparklinesLine, SparklinesSpots, SparklinesReferenceLine } from 'react-sparklines';

import Alert from 'react-bootstrap/Alert';

import Speech from 'react-speech';

// Establish a socket connection to the server at the specified URL

const socket = io.connect('http://localhost:5050');

function App()  {

  const [receiveMessage, setReceiveMessage] = useState("");

  const [vib , setVib] = useState("");

  const [hum, sethum] = useState("");

  const [arr ,setarr] = useState([]);

  const [temp ,settemp] = useState([]);

  const [avrg, setavrg] = useState();

  const [vibarr, setvibarr] = useState([]);

  const [humarr, sethumarr] = useState([]);

  const[crdate, setcrdate] = useState([]);

 // let avr = [];

 const url = "http://localhost:5050/params";

 async function gettemp () {

  try {

    const response = await fetch(url);

    if (!response.ok){

      throw new Error(`res status : ${response.status}`);

    }

    const res = await response.json();

    //console.log( res[100].temp);

   for (let i = 0; i < res.length; ++i){

    settemp([...temp,res[i].temp]);

   }

      //const temp = res.map((item)=> item.temp.reduce((a,b)=> a + b) / res.length, res)

      setcrdate(res.createdAt);

  } catch (error) {

    console.error(error);

  }

 }

  useEffect(() => {

   // gettemp();

    // Listen for incoming messages from the server

    try{

      socket.on("receive\_message", (ac,co2,t) => {

       // socket.on("receive\_message", (co2,t) => {

        //const ac = 0;

          setReceiveMessage(ac);

        setVib(co2);

        sethum(t);

        function getRandomInt(max) {

          return Math.floor(Math.random() \* max);

        }

        const sum = temp.reduce((a,b) => a + b ,0);

        const avg = (sum / temp.length);

        setavrg(avg);

         if (arr.length >=  8){

            let avr = [...arr];

            let vrr= [...vibarr];

            let hrr = [...humarr];

            avr.shift();

            vrr.shift();

            hrr.shift();

            setarr(avr);

            setvibarr(vrr);

            sethumarr(hrr);

          }else{

              setarr([...arr, ac]);

              arr.push(ac);

              setvibarr([...vibarr, co2]);

              vibarr.push(co2);

              sethumarr([...humarr,t]);

              humarr.push(t);

          }

      });

    }catch(e){

      console.log(e)

    }

    // Cleanup the effect by removing the event listener when the component unmounts

    return () => {

      socket.off("receive\_message");

    };

  }, []); // Empty dependency array ensures this runs only once when the component mounts

  const msgv = () => {

   const msg = new SpeechSynthesisUtterance()

   msg.text = "Warning warning High Speed Movement"

   window.speechSynthesis.speak(msg)

  }

    const msgh = () => {

      const msg1 = new SpeechSynthesisUtterance()

      msg1.text = "Warning warning High Tvoc in Air "

      window.speechSynthesis.speak(msg1)

    }

  const iac = Number(receiveMessage)

  const ih = Number(hum);

  const iv = Number(vib);

  return (

    <Row>

      <Col md={6} className="mb-2">

      <ToastContainer

          className="p-3"

          position={'middle-start'}

          style={{ zIndex: 1 }} >

        <Toast >

          <Toast.Header>

            <strong className="me-auto">accelration :</strong>

            <small>{receiveMessage} m/s</small>

          </Toast.Header>

          <Toast.Body>

          {iac >0.8 || iac < -2 ? <> <Alert variant='danger'> Warning warning High Speed Movement  </Alert> {msgv()}</> : ""}

          <Sparklines data={arr} margin={6}>

           <SparklinesLine style={{ strokeWidth: 3, stroke: "#336aff", fill: "none" }} />

           <SparklinesSpots size={4} style={{ stroke: "#336aff", strokeWidth: 3, fill: "white" }} />

           <SparklinesReferenceLine type="min" />

          </Sparklines>

      </Toast.Body>

        </Toast>

        </ToastContainer>

      </Col>

    <Col md={6} className="mb-2">

    <ToastContainer

          className="p-3"

          position={'middle-end'}

          style={{ zIndex: 1 }}>

        <Toast >

          <Toast.Header>

            <strong className="me-auto">TVOC & Temprature:</strong>

            <small>{vib} ppb - {hum} C</small>

          </Toast.Header>

          <Toast.Body>

          {iv > 9000000 || ih > 29 ? <> <Alert variant='danger'> Warning!! warning!! High Tvoc in Air  </Alert> {msgh()}</> : ""}

            <div style={{height:10}}>

            <Sparklines data={vibarr} margin={6} >

           <SparklinesLine style={{ strokeWidth: 3, stroke: "#336aff", fill: "none" }} />

           <SparklinesSpots size={4} style={{ stroke: "#336aff", strokeWidth: 3, fill: "white" }} />

           <SparklinesReferenceLine type="min" />

          </Sparklines>

          </div>

          <Sparklines data={humarr} margin={6} >

           <SparklinesLine style={{ strokeWidth: 3, stroke: "green", fill: "none" }} />

           <SparklinesSpots size={4} style={{ stroke: "#336aff", strokeWidth: 3, fill: "white" }} />

           <SparklinesReferenceLine type="min" />

          </Sparklines>

      </Toast.Body>

        </Toast>

        </ToastContainer>

      </Col>

    </Row>

  );

}

export default App;