```
"use strict";

/**ERROR CODES:
 *
 * 0: No error: the request was successful
 * 1: No city could be extracted from the req.params object - usually means the user has made a mistake and the input is undefined or null
 * 2: The city provided in params failed the regex test - contains something other than alphabet characters
 * 3: There was a problem making the open weather api request - done inside callWeatherAPI function. Usually means city name is not found from the API.
 */

const express = require('express');
const axios = require("axios");
const app = express();
const cors = require("cors");
const port = 3000;
const OPEN_KEY = "3e2d927d4f28b456c6bc662f34350957";


app.use(cors());

// Fix our cors issues, by adding middleware to allow requests from localhost
// app.use(function(req, res, next) {
//     //res.header("Access-Control-Allow-Origin", "no-cors");
//     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
//     next();
//   });

//Function returns a JSON object of the weather metrics we require
const callWeatherAPI = async (cityName) => {
    try {
        const urlForWeather = `https://api.openweathermap.org/data/2.5/forecast?q=${cityName}&appid=${OPEN_KEY}&units=metric`; //url taken from API on Open Weather website
        const response = await axios.get(urlForWeather); //returns a promise, going to use async await rather than a .then

        //OpenWeatherMap will return 200 if it succeeded in getting a valid response i.e. works properly
        if (response.status === 200){
            const data = response.data;         //data = the data object OpenWeatherMap returns

                if (data.cnt === 40)            //error checking to ensure number of days is 40 as it should be
                {
                //Check each day for rain
                //some(): tests whether at least one element in the array passes the test implemented by the provided function. It returns a Boolean value.
                const doesRainIn5days = data.list.some((element) => {
                    if (element.rain !== undefined) return (element.rain["3h"] > 0);
                    else return false;
                });

                //Get the average temp over the next 5 days to check if they should pack for cold warm hot
```

```
48                //reduce(): executes a reducer function on each element of the array using the accumulator value and the current value, resulting in single output value.
49                var packFor = ""; // Variable to store what the user should pack for- options can be["Cold", "Warm", "Hot"]
50                const averageTemp = (data.list.reduce((accum, current) => accum + current.main.temp, 0)) / data.cnt;
51                if (averageTemp < 10) packFor = "Cold";
52                else if (averageTemp >= 10 && averageTemp < 20) packFor = "Warm";
53                else packFor = "Hot";
54
55                //Get forecast for each day (weather taken 3 times a day for 5 days is 40 forecasts, we want one for each day. 40/5 =8 so ensure the forecast is divisible by 8)
56                //filter(); creates a new array with all elements that pass the test implemented by the provided function.
57                const weatherList = data.list.filter((element, index) => {
58                    //return the temp at midday
59                    const timestamp = (element.dt * 1000);  //dt is the timestamp in seconds, it needs to be in milliseconds for a javascript object so *1000
60                    const date = new Date(timestamp);       //made a js date object from the date
61                    // console.log(date.toLocaleString());
62                    // console.log(date.getHours());
63                    if(date.getHours() === 12) return true;
64                    else return false;
65                });
66
67                const mappedWeatherList = weatherList.map(element => {
68                    //check if rain element exists and storing it in variable rain if it does, will put 0 as the rainfall amount if rain element does not exist
69                    var rain = 0;
70                    if (element.rain !== undefined && element.rain["3h"] !== null) rain = element.rain["3h"];
71                    //return weather list with metrics we require mapped
72                    return {
73                        temp: element.main.temp,
74                        weather: element.weather[0].main,
75                        windSpeed: element.wind.speed,
76                        rain: rain
77                    };
78                });
79
80
81                //Oject we will return to our front end
82                return {
83                    code: 0,                                    //Success code
84                    doesRainIn5days: doesRainIn5days,       //tell them if they should bring umberella - Boolean
85                    packFor: packFor,                           //tell them if they should pack for cold warm or hot - String
86                    weatherList: mappedWeatherList          //includes weather metrics for each day - Array of objects
87                }
88
89            } else return null;
90
91        } else return null;
92    }
93    //catch any exceptions axios catches, print to console what went wrong
94    catch (e){
```

```
        console.log(e);
        return null;
    }
}

//Function will return false if the city name is invalid due to irregular charachters found. Uses regular expressions to check.
const cityNameIsValid = (cityName) => {
    const re = /[^a-zA-Z]/;
    return !re.test(cityName);
}

//Whenever it receives a request on port 30000, it reads where it gets directed to, finds the / (the root its given), creates a request
//object which then can be accessed inside the function. When you want to send something back you return a response object.
//You are defining what will get sent back.
app.get('/:cityName', async (req,res, next) => {

    const cityName = req.params.cityName;
    console.log(cityName);
    //Make sure city name is input correctly, else return error code so can decide what to do with front-end UI based on error code returned
    if (cityName === undefined || cityName === null) return res.status(400).json({code: 1, error: "City name is not defined"});
    else if (!cityNameIsValid(cityName)) return res.status(400).json({code: 2, message: "City name is not valid"});


    //callWeatherAPI function returns a promise as it uses async await.
    //If the method returns null then there was an error inside the callWeatherAPI function.
    const weatherResponse = await callWeatherAPI(cityName);
    if (weatherResponse === null) return res.status(500).json({code: 3, message: "Server encountered an error while making request"});

    return res.status(200).json(weatherResponse);
});



app.listen(port, () => console.log(`example app listening on port ${port}!`));

```