Internet Applications Assignment

Honglin Li 19315272

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>☁️Vuether 🌡️</title>
    <!-- Include Vue -->
    <script type="importmap">
        {
            "imports": {
                "vue":
"https://unpkg.com/vue@3/dist/vue.esm-browser.js"
            }
        }
    </script>
    <script type="module">
        import { createApp } from 'vue'
        createApp({
            data() {
                return {
                    city: '',
                    hasError: false,
                    errorMsg: '',
                    weatherJSON: null,
                    willRain: null,
                    maskAdvised: null,
                    tempSentiment: null,
                    airPolJSON: null,
                    cityJSON:null
                }
            },
            methods: {
                getWeatherForecast() {
```

```
                    console.log(`Requesting weather forecast for
${this.city} from OpenWeather...`);

                    fetch(`/forecast/${this.city}`)
                        .then((response) => {
                            if (response.status === 200) return
response.json();
                            else throw Error(response.statusText);
                        }).then(responseJSON => {
                            console.log(responseJSON);
                            this.weatherJSON =
responseJSON.forecastData;
                            this.willRain = responseJSON.willRain;
                            this.tempSentiment =
responseJSON.tempSentiment;
                            this.maskAdvised =
responseJSON.maskAdvised;
                            this.airPolJSON =
responseJSON.airPollutionData;
                            this.cityJSON = responseJSON.cityInfo;
                            // Clear errors
                            this.hasError = false;
                            this.errorMsg = '';
                        })
                        .catch(error => {
                            console.error(error);

                            // Set error
                            this.hasError = true;
                            this.errorMsg = `Can't fetch weather data
for ${this.city} :(`;
                        });
                }
            }
        }).mount('#app')
    </script>
</head>

<body>
    <div id="app">
        <h1>☁️Vuether 🌡️</h1>
        <br />
```

```
        <h2>Enter city name for today's weather and its forecast for
the next four days:</h2>
        <div>
            <div>
                <input v-model="city" type="text" v-model="city"
placeholder="City..." required />
            </div>
            <button v-on:click="getWeatherForecast">Search🔍</button>
        </div>

        <span v-if="hasError">
            Error: {{errorMsg}}
        </span>
        <!-- Innovative Feature City Information -->
        <div v-if="!hasError && cityJSON!= null">
            <h1>{{cityJSON.name}}, {{cityJSON.country}}</h1>
            <h3>Latitude: {{cityJSON.lat}}, Longitude:
{{cityJSON.lon}}</h3>
            <h3 v-if="cityJSON.timeZone<0">Time Zone: GMT
{{cityJSON.timeZone}}</h3>
            <h3 v-else>Time Zone: GMT +{{cityJSON.timeZone}}</h3>
            <h3>Sunrise: {{cityJSON.sunRise}}, Sunset:
{{cityJSON.sunSet}} (Local Time)</h3>
        </div>
        <!-- Umbrella Packing Tips -->
        <div v-if="!hasError && willRain != null">

            <h2>🌂 Should I bring an umbrella? ☔</h2>
            <span v-if="willRain">☔ Bring an umbrella with ya, you
don't wanna get soaked. 💦</span>
            <span v-if="!willRain">🌞 Keep your umbrella at home, and
enjoy some sunny days, or maybe cloudy. ⛅</span>

        </div>

        <!-- Clothes Packing Tips by Temperature Sentiment-->
        <div v-if="!hasError && tempSentiment != null">

            <h2>👔 What kind of clothes should I pack? 🧳</h2>
            <span v-if="tempSentiment.tempFeel == 'hot'">🥵Hot
weather({{tempSentiment.min}}°C -
                {{tempSentiment.max}}°C).
                <br />👕 Pack some light cloths and stay hydrated! 💧
```

```
            </span>
            <span v-if="tempSentiment.tempFeel == 'mild'">🤗Mild
weather({{tempSentiment.min}}°C -
                {{tempSentiment.max}}°C).
                <br />👘 Pack a few extra layers just in case.
🍃</span>
            <span v-if="tempSentiment.tempFeel == 'cold'">🥶Cold
weather({{tempSentiment.min}}°C -
                {{tempSentiment.max}}°C).
                <br />🏂 Pack some warm clothes and maybe enjoy a hot
cuppa. ☕</span>

        </div>

        <!-- Masking tips -->
        <div v-if="!hasError && maskAdvised != null">

            <h2>🏭 Should I wear a mask? 🫁</h2>
            <span v-if="maskAdvised">😷 The air quality is going to be
rough. You should wear a mask!</span>
            <span v-if="!maskAdvised">😃 The air quality is going be
good. Enjoy the fresh air!</span>

        </div>


        <div v-if="!hasError && weatherJSON">
            <h2>☁️Next 4 Days Weather Forecast🌡️</h2>
            <table>
                <thead>
                    <tr>
                        <th>📆</th>
                        <th>Lowest 🌡️(°C)</th>
                        <th>Avg.🌡️ (°C)</th>
                        <th>Highest 🌡️(°C)</th>
                        <th>🌬️(m/s)</th>
                        <th>☔(mm)</th>
                    </tr>
                </thead>
                <tbody>
                    <template v-for="(dayForecastData, date) in
weatherJSON">
                        <tr>
```

```
                            <td>|<u>{{ date }}</u></td>
                            <td>|<u>{{ dayForecastData.minTemp
}}</u></td>
                            <td>|<u>{{ dayForecastData.avgTemp
}}</u></td>
                            <td>|<u>{{ dayForecastData.maxTemp
}}</u></td>
                            <td>|<u>{{ dayForecastData.avgWind
}}</u></td>
                            <td>|<u>{{ dayForecastData.rainfallLevels
}}</u></td>
                        </tr>
                    </template>
                </tbody>
            </table>
        </div>

        <div v-if="!hasError&&airPolJSON">
            <h2>🫁 Next 5 Days Air Pollution 🏭
                <h3>Remember to bring a mask if any day's PM2_5 is over
10!</h3>
            </h2>
            <table>
                <thead>
                    <tr>
                        <th>📆</th>
                        <th>Daily Average PM2_5</th>
                        <!-- <th>pm2_5</th> -->
                    </tr>
                </thead>
                <tbody>
                    <template v-for="(dayAirPolData, date) in
airPolJSON">
                        <tr>
                            <td>|<u>{{ date }}</u></td>
                            <td>|<u>{{ dayAirPolData.avgPM2_5
}}</u></td>
                            <!-- <td>|<u>{{ airPolData.pm2_5
}}</u></td> -->
                        </tr>
                    </template>
                </tbody>
            </table>
```

```
        </div>
    </div>
</body>

</html>
```

server.js

```
require("dotenv").config()
const axios = require('axios')
const cors = require('cors');
const express = require('express');
const { get } = require("http");
const app = express();
app.use(cors());
const path = require('path');
const port = 3000

const URL_base = `https://api.openweathermap.org/data/2.5`
const API_key = process.env.OPENWEATHER_API_KEY

// Functions for temperature calculations
const average = arr => (arr.reduce((p, c) => p + c, 0) /
arr.length).toFixed(2);
const sum = arr => (arr.reduce((p, c) => p + c, 0)).toFixed(2);
const kel_to_cel = k => Math.round((k - 273.12) * 100) / 100;
const min = arr => (Math.min(...arr));
const max = arr => (Math.max(...arr));
// formatting date from seconds
function dtToDate(dt) {
    let date = new Date(dt * 1000);
    date.setHours(0, 0, 0, 0);
    return date.toLocaleDateString();
}
//get timeZone from DT dt
function timezoneFromDT(dt) {
    let sign = 1;
    if (dt < 0) sign = -1;
    let date = new Date(Math.abs(dt) * 1000);
    let hr = date.getHours() - 1;
    return hr * sign;
}
//get hour and minute from dt
```

```
function timeFromDT(dt) {
    let date = new Date(dt * 1000);
    let hours = date.getHours();
    let mins = date.getMinutes();
    if (hours < 10) { hours = "0" + hours; }
    if (mins < 10) { mins = "0" + mins; }
    let hrMin = hours + ':' + mins;
    return hrMin;
}
app.get('/', (req, res) =>
res.sendFile(path.join(__dirname,'../public/index.html')));
app.get('/forecast/:city', getForecast);
app.listen(port, () => console.log(`Vuether listening on port
${port}!`));

//PM2_5, mask advised if any day's avg PM2_5 exceeds 10
function getMaskAdvice(airPollutionData) {
    for (date in airPollutionData) {
        if (airPollutionData[date].avgPM2_5 !== null &&
airPollutionData[date].avgPM2_5 !== undefined
            && airPollutionData[date].avgPM2_5 > 10) {
            //console.log(airPollutionData[date].avgPM2_5);
            return true;
        }
    }
    return false;
}

// Temperature Sentiment - hot/mild/cold
function getTempSentiment(forecastData) {
    let max = forecastData[Object.keys(forecastData)[0]].maxTemp;
    let min = forecastData[Object.keys(forecastData)[0]].minTemp;
    let tempFeel = null;

    for (date in forecastData) {
        currMinTemp = forecastData[date].minTemp;
        currMaxTemp = forecastData[date].maxTemp;

        if (currMinTemp <= min)
            min = currMinTemp;
        if (currMaxTemp >= max)
            max = currMaxTemp;
    }
```

```
    if (max > 24) tempFeel = "hot";
    else if (min >= 12 && max <= 24) tempFeel = "mild";
    else tempFeel = "cold";

    return {
        tempFeel: tempFeel,
        max: max,
        min: min
    }

}

function getForecast(req, res) {
    var city = req.params.city;
    console.log(`Requesting weather forecast data for ${city} from
OpenWeather...`);

    var forecastData = {};
    var airPollutionData = {};
    var willRain = false;
    var cityLat = 0;
    var cityLon = 0;
    var cityInfo = {}

    axios.get(`${URL_base}/forecast?q=${city}&APPID=${API_key}`).then(
        (response) => {
            const { lat, lon } = response.data.city.coord;
            cityLat = lat;
            cityLon = lon;

            //Innovative Feature
            //get city information
            cityInfo.name = response.data.city.name;
            cityInfo.country = response.data.city.country
            cityInfo.lat = lat;
            cityInfo.lon = lon;
            cityInfo.timeZone =
timezoneFromDT(response.data.city.timezone);
            // console.log("TimeZone:",response.data.city.timezone)
            // console.log("Sunrise:",response.data.city.sunrise)
            // console.log("Sunset:",response.data.city.sunset)
```

```
            cityInfo.sunRise = timeFromDT(response.data.city.sunrise +
response.data.city.timezone);
            cityInfo.sunSet = timeFromDT(response.data.city.sunset +
response.data.city.timezone);

            var fetchedWeatherData = response.data.list;

            // Loop through forecast of each day
            var days = 0
            for (weatherEntry in fetchedWeatherData) {
                date = dtToDate(response.data.list[weatherEntry].dt);
                // Forecast for only today and the next 4 days
                if (days > 4) break;
                // Init forecast data for a day if undefined or null
                if (!forecastData[date]) {
                    days++;
                    forecastData[date] = {
                        temperaturesK: [],
                        tempMinsK: [],
                        tempMaxsK: [],
                        windSpeeds: [],
                        rainfallLevels: []
                    }
                }


forecastData[date].temperaturesK.push(fetchedWeatherData[weatherEntry].
main.temp);

forecastData[date].tempMinsK.push(fetchedWeatherData[weatherEntry].main
.temp_min);

forecastData[date].tempMaxsK.push(fetchedWeatherData[weatherEntry].main
.temp_max);

forecastData[date].windSpeeds.push(fetchedWeatherData[weatherEntry].win
d.speed);

                // Add rain level if any rain
                if (fetchedWeatherData[weatherEntry].rain &&
fetchedWeatherData[weatherEntry].rain['3h']) {
                    willRain = true;
```

```
forecastData[date].rainfallLevels.push(fetchedWeatherData[weatherEntry]
.rain['3h']);
                }

            }

        }
        //after weather forecast, get air pollution data with lat and
lon
    ).then(() => {

axios.get(`${URL_base}/air_pollution/forecast?lat=${cityLat}&lon=${city
Lon}&APPID=${API_key}`).then((responseAirPol) => {
            const fetchedAirPollutionData = responseAirPol.data.list;
            var days = 0
            for (airPollutionEntry of fetchedAirPollutionData) {
                date = dtToDate(airPollutionEntry.dt);
                // Air pollution next 5 days
                if (days > 5) break;
                // Init air pollution data if undefined or null
                if (!airPollutionData[date]) {
                    days++;
                    airPollutionData[date] = {
                        pm2_5: []
                    }
                }

airPollutionData[date].pm2_5.push(parseInt(airPollutionEntry.components
.pm2_5))
            }

            //  Calculate daily averages
            for (date in forecastData) {
                forecastData[date].avgTemp =
kel_to_cel(average(forecastData[date].temperaturesK));
                //Getting min/max temperature by getting min/max of
each day's min/max temps
                forecastData[date].minTemp =
kel_to_cel(min(forecastData[date].tempMinsK));
                forecastData[date].maxTemp =
kel_to_cel(max(forecastData[date].tempMaxsK));
```

```
                forecastData[date].avgWind =
average(forecastData[date].windSpeeds);
                forecastData[date].rainfallLevels =
sum(forecastData[date].rainfallLevels);

            }
            for (date in airPollutionData) {
                if (airPollutionData[date].pm2_5 !== null &&
airPollutionData[date].pm2_5 !== undefined)
                    airPollutionData[date].avgPM2_5 =
average(airPollutionData[date].pm2_5);
            }
            // Overall temperature sentiment (4 days) and mask advice
(5 days)
            tempSentiment = getTempSentiment(forecastData);
            maskAdvised = getMaskAdvice(airPollutionData);

            res.json({
                forecastData: forecastData,
                willRain: willRain,
                tempSentiment: tempSentiment,
                maskAdvised: maskAdvised,
                airPollutionData: airPollutionData,
                cityInfo: cityInfo
            })

        }).catch((error) => {
            console.error(error);
            res.status(400);
            res.json({
                error: "Bad Request!"
            });
        })
    }

    ).catch((error) => {
        console.error(error);
        res.status(400);
        res.json({
            error: "Bad Request!"
        });
    })
```

```
}
```