Ivan Banchev

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Weather App</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="style.css">
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
    <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/gh/mwasil/Leaflet.Rainviewer/leaflet.rainviewer.css"/>
    <script
src="https://cdn.jsdelivr.net/gh/mwasil/Leaflet.Rainviewer/leaflet.rainviewer.js"></script>
    <script src="https://kit.fontawesome.com/f423f669aa.js" crossorigin="anonymous"></script>

</head>
<body>
    <div id="app">
        <div class="grid-container">

            <div class="grid-item left-column">
                <div class="background-half container">
                    <div class="dialogue-box">
                        <form @submit.prevent="handleFormSubmit">
                            <input type="text" v-model="city" placeholder="Enter city" id="userInput">
                            <i class="fa fa-search" aria-hidden="true"></i>
                        </form>
                        <div id="errorMessage" style="color: red;">{{ errorMessage }}</div>
                    </div>

                    <div class="content">
                        <button @click="handleTodayButtonClick">Today</button>
                        <button @click="handle3DaysButtonClick">3 Days</button>
                        <button @click="handleAirQualityButtonClick">Air Quality</button>
                        <button @click="handleNewFeatButtonClick">Landscape & Tourism</button>
                    </div>
                    <div class="weathergrid">
                        <div id="leftGrid"v-if="weatherData">
                            <h3> <i class="fa fa-calendar" aria-hidden="true"></i> {{ formattedDateToday
}} </h3>
                            <h3> <i class="fa fa-map-marker" aria-hidden="true"></i> {{ cityName }} , {{
countryName}} </h3>
```

```
                    <div class="weather-card">
                        <p class="temperature">Temperature: {{ weatherData.main.temp }} ºC </p>
                        <p>Weather: {{ description }}</p>
                        <p>Humidity: {{ weatherData.main.humidity }}%</p>
                        <p>Wind Speed: {{ weatherData.wind.speed }} m/s</p>
                        <p>Pressure: {{ weatherData.main.pressure }} hPa</p>
                    </div>

                </div>
            </div>
            <div v-if="forecastData">
                <h3>3-Day Forecast</h3>
                <table>
                    <thead>
                        <tr>
                            <th>Date</th>
                            <th>Max / Min Temp</th>
                            <th>Wind Speed</th>
                            <th>Weather</th>
                            <th>Rain Chance</th>
                            <th>Umbrella</th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr v-for="(day, index) in forecastData" :key="index">
                            <td>{{ day.date}}</td>
                            <td>{{ day.maxTemp }}°C / {{ day.minTemp }}°C</td>
                            <td>{{ day.avgWindSpeed }} m/s</td>
                            <td> {{ day.weatherDescription }} </td>
                            <td>{{ day.rainChance }}%</td>
                            <td>
                                <img v-if="day.rainChance > 50" src="Red-Umbrella-PNG.webp"
alt="Umbrella" width="30" height="30">
                                <img v-if="day.rainChance <= 50" src="NoUmbrella.png" alt="No
Umbrella" width="30" height="30">
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
```

```
        <div class="grid-item right-column">
            <div id="map" class="map-container"></div>
            <div id="air" class="airqualityTable" v-if="airData">
                <h3>Air Quality</h3>

                    <div class="grid-item airquality">

                        <div class="airquality-grid">
                            <p>SO2: {{ so2 }} µg/m³</p>
                            <p>NO2: {{ no2 }} µg/m³</p>
                            <p>PM10: {{ pm10 }} µg/m³</p>
                            <p>PM2_5: {{ pm2_5 }} µg/m³</p>
                            <p>O3: {{ o3 }} µg/m³</p>
                            <p>CO: {{ co }} µg/m³</p>
                            <p>NH3: {{ nh3 }} µg/m³</p>
                            <p>NO: {{ no }} µg/m³</p>
                         </div>
                    </div>

                    <p v-if="airquality === 1"> The quality of the air is GOOD </p>
                    <p v-if="airquality > 1"> The quality of the air is FAIR <br> {{
getCauseOfLowerAQI() }}</p>
                    <p v-if="airquality === 3"> The quality of the air is MODERATE <br> {{
getCauseOfLowerAQI() }} </p>
                    <p v-if="airquality === 4"> The quality of the air is POOR <br> {{
getCauseOfLowerAQI() }} </p>
                    <p v-if="airquality === 5"> The quality of the air is VERY POOR <br> {{
getCauseOfLowerAQI() }} </p>

            </div>
        </div>
    </div>
    <div v-if="loading" class="loading-spinner">
        <div class="spinner"></div>
    </div>
</div>



    <script src="app.js"></script>
</body>
</html>

const apiKey = '47f3ad18870adefc1fd086cb168886d5';
```

```
const days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];
const months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September',
'October', 'November', 'December'];

new Vue({
    el: '#app',
    data: {
        city: '',
        errorMessage: '',
        displayInput: '',
        weatherData: null,
        forecastData: null,
        airData: null,
        forecastWeatherData: null,
        description: '',
        userCoordinates: null,
        formattedDate: '',
        formattedDateToday : '',
        loading: false,
        cityName: '',
        countryName: '',
        latitudeProperty: '',
        longitudeProperty: '',
        co: '',
        co2: '',
        no2: '',
        o3: '',
        pm2_5: '',
        pm10: '',
        so2: '',
        nh3: '',
        airquality: '',
        activated3days : false,
        activatedAirQuality : false


    },
    methods: {

        getCauseOfLowerAQI() {
            const pollutants = {
                O3: this.o3,
                PM10: this.pm10,
                PM2_5: this.pm2_5,
```

```
        NO2: this.no2,
        CO: this.co,
        SO2: this.so2,
    };

    const pollutantThresholds = {
        O3: 60,
        PM10: 20,
        PM2_5: 10,
        NO2: 40,
        CO: 4400,
        SO2: 20,
    };

    let causeOfLowerAQI = [];
    for (const pollutant in pollutants) {
        if (pollutants[pollutant] > pollutantThresholds[pollutant]) {
            causeOfLowerAQI.push(pollutant);
        }
    }

    if (causeOfLowerAQI.length > 0) {
        return `The pollutant(s) causing AQI to be worse: ${causeOfLowerAQI.join(', ')}`;
    } else {
        return "All pollutants are within good levels.";
    }
},


auxDisplay(temp) {

    if (temp < 8.0) {
        return 'Cold';
    } else if (temp >= 8.0 && temp <= 24.0) {
        return 'Mild';
    } else if (temp > 24.0) {
        return 'Hot';
    }
},

async handleFormSubmit() {
    if (this.city.trim() === "") {
        this.errorMessage = "Input cannot be empty.";
        this.displayInput = "";
```

```
            return;
        }
        this.errorMessage = "";
        this.loading = true;

        const apiUrl =
`https://api.openweathermap.org/data/2.5/weather?q=${this.city}&appid=${apiKey}&units=metric
`;

        try {
            const response = await fetch(apiUrl);
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            const data = await response.json();
            this.weatherData = data;
            this.weatherData.main.temp = Math.round(this.weatherData.main.temp);
            this.weatherData.rain = data.rain;

            this.description = this.auxDisplay(data.main.temp);

            this.displayInput = `You entered: ${this.city}`;

            const d = new Date();
            const dayName = days[d.getDay()];
            const monthName = months[d.getMonth()];
            const day = d.getDate();
            const year = d.getFullYear();

            this.formattedDateToday = `${dayName}, ${day} ${monthName} ${year}`;


            let geoApiUrl =
`https://api.openweathermap.org/geo/1.0/direct?q=${this.city}&limit=1&appid=${apiKey}`;
            try {
                const response = await fetch(geoApiUrl);
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            const data = await response.json();

            let {name, lat, lon, country} = data[0];
            this.cityName = name;
            this.countryName = country;
```

```
            this.latitudeProperty = lat;
            this.longitudeProperty = lon;

            } catch (error) {
            this.errorMessage = `Error: ${error.message}`;
            }
        } catch (error) {
            this.errorMessage = `Error: ${error.message}`;
        }
        finally {
            this.loading = false;
            if(this.activated3days){
                this.handle3DaysButtonClick();
            }
            if(this.activatedAirQuality){
                this.handleAirQualityButtonClick();
            }
        }
    },
    async handle3DaysButtonClick() {
        if (this.city.trim() === "") {
            this.errorMessage = "Input cannot be empty.";
            this.displayInput = "";
            return;
        }
        this.loading = true;
        this.errorMessage = "";

        const apiUrl =
`https://api.openweathermap.org/data/2.5/forecast?q=${this.city}&appid=${apiKey}&units=metric
`;

        try {
            const response = await fetch(apiUrl);
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            const data = await response.json();
            this.activated3days = true;

            const groupedData = {};

            data.list.forEach(item => {
                const date = item.dt_txt.split(' ')[0];
```

```
            console.log(item.weather[0]);
            if (!groupedData[date]) {
                groupedData[date] = {
                    temps: [],
                    weather: item.weather[0],
                    rainChance: Math.round(item.pop * 100).toFixed(0),
                    windSpeeds: []

                };
            }

            groupedData[date].temps.push(item.main.temp);
            groupedData[date].windSpeeds.push(item.wind.speed);

        });


        this.forecastData = Object.keys(groupedData).slice(0, 3).map(date => {
            const d = new Date(date);
            const dayName = days[d.getDay()];
            const monthName = months[d.getMonth()];
            const day = d.getDate();
            const year = d.getFullYear();
            this.formattedDate = `${dayName}, ${day},    ${monthName} ${year}`;

            const temps = groupedData[date].temps;
            let maxTemp = Math.round(Math.max(...temps));
            let minTemp = Math.round(Math.min(...temps));
            let averageTemp = maxTemp + minTemp / 2;
            return {
                date: this.formattedDate,
                maxTemp: maxTemp,
                minTemp: minTemp,
                weatherDescription: this.auxDisplay(averageTemp),
                rainChance: groupedData[date].rainChance,
                avgWindSpeed: (groupedData[date].windSpeeds.reduce((a, b) => a + b, 0) /
groupedData[date].windSpeeds.length).toFixed(2)
            };
        });
        this.forecastWeatherData = { temp: this.forecastData[0].maxTemp };
    } catch (error) {
        this.errorMessage = `Error: ${error.message}`;
    }
    finally {
```

```
            this.loading = false;
        }
    },
    handleTodayButtonClick() {
        this.handleFormSubmit();
    },
    async handleAirQualityButtonClick() {
        if (this.city.trim() === "") {
            this.errorMessage = "Input cannot be empty.";
            this.displayInput = "";
            return;
        }
        this.errorMessage = "";
        this.loading = true;
        const airApiCall =
`http://api.openweathermap.org/data/2.5/air_pollution?lat=${this.latitudeProperty}&lon=${this.lon
gitudeProperty}&appid=${apiKey}`;

        try {
            const response = await fetch(airApiCall);
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            const data = await response.json();
            this.activatedAirQuality = true;
            this.airData = data;
            let {co, no, no2, o3, so2, pm2_5, pm10, nh3} = data.list[0].components;
            this.co = co;
            this.no = no;
            this.no2 = no2;
            this.o3 = o3;
            this.pm2_5 = pm2_5;
            this.pm10 = pm10;
            this.so2 = so2;
            this.nh3 = nh3;
            this.airquality = data.list[0].main.aqi;

        } catch (error) {

            this.errorMessage = `Error: ${error.message}`;
        } finally {
            this.loading = false;
        }
```

```
    },
    handleNewFeatButtonClick() {
        // Implement the logic for fetching and displaying landscape & tourism data
    },

    getUserCoordinates() {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(position => {
                this.userCoordinates = {
                    latitude: position.coords.latitude,
                    longitude: position.coords.longitude
                };
                console.log('User coordinates:', this.userCoordinates);
                this.displayMap();
            }, error => {
                this.errorMessage = `Error getting location: ${error.message}`;
            });
        } else {
            this.errorMessage = "Geolocation is not supported by this browser.";
        }
    },
    displayMap() {
        if (this.userCoordinates) {

            let map = L.map('map').setView([this.userCoordinates.latitude,
this.userCoordinates.longitude], 4);



L.tileLayer(`https://api.maptiler.com/maps/hybrid/{z}/{x}/{y}.jpg?key=WA2dNh7InZuBzttxGdeH`, {

            }).addTo(map);
            L.control.rainviewer({
                position: 'bottomleft',
                nextButtonText: '>',
                playStopButtonText: 'Play/Stop',
                prevButtonText: '<',
                positionSliderLabelText: "Hour:",
                opacitySliderLabelText: "Opacity:",
                animationInterval: 2000,
                opacity: 0.5
            }).addTo(map);
```

```
            const marker = L.marker([this.userCoordinates.latitude,
this.userCoordinates.longitude]).addTo(map);
            marker.bindPopup('You are here!').openPopup();
        }
    }



  },

  mounted() {
    this.getUserCoordinates();
  }
});
/* style.css */

body {
  font-family: Arial, sans-serif;
  background-color: #423f3f;
  margin: 0;
  padding: 0;
  height: 100vh;
}

.grid-container {
  display: grid;
  grid-template-columns: 2fr 1fr;
  grid-template-rows: 2fr;
  gap: 20px;
  padding: 20px;
  height: 100%;
  box-sizing: border-box;
}

.grid-item {
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  padding: 20px;
  height: 100%;
}

.left-column {
  display: flex;
```

```
    flex-direction: column;
    justify-content: space-between;
}

.right-column {
    display: flex;
    flex-direction: column;
    height: 100vh;
}

.grid-container {
    display: grid;
    grid-template-columns: 2fr 1fr;
    grid-template-rows: 2fr;
    gap: 20px;
    padding: 20px;
    height: 100%;
    box-sizing: border-box;
}
.airqualityTable{
    justify-content: center;
    align-items: center;
    display: flex;
    flex-direction: column;
    min-width: 80%;
}
.airquality-grid {
    display: grid;

    grid-template-columns: 1fr 1fr; /* Two columns */
    grid-template-rows: repeat(2, auto); /* Four rows */
    gap: 10px; /* Space between grid items */
    box-sizing: border-box;
}

.airquality-grid .grid-item {
    background-color: #f9f9f9;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}
.background-half {
    width: 100%;
```

```
    height: 50%;
}

.map-container {
    width: 100%;
    height: 50%;
    min-height: 200px;
    background-color: #e0e0e0;
    border-radius: 10px;
}


.dialogue-box {
    margin: 0 auto;
    padding: 20px;
    border-radius: 10px;
    background-color: #f9f9f9;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.dialogue-box input[type="text"] {
    width: 90%;
    padding: 10px;
    border: none;
    border-radius: 10px;
    background-color: #e9e9e9;
    box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.1);
    font-size: 16px;
    outline: none;
    margin-bottom: 20px;
}

.content {
    text-align: center;
}

.content button {
    padding: 10px 20px;
    margin: 5px;
    border: none;
    border-radius: 10px;
    background-color: #4CAF50;
    color: white;
    font-size: 16px;
```

```
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.content button:hover {
    background-color: #45a049;
}
.weatherGrid {
    display: grid;
    grid-template-columns: 1fr 1fr; /* Two columns */
    gap: 10px; /* Space between grid items */
    box-sizing: border-box;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    padding: 12px;
    text-align: left;
}

th {
    background-color: #f2f2f2;
}

.loading-spinner {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(255, 255, 255, 0.8);
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 9999;
```

```
}

.spinner {
    border: 8px solid #f3f3f3;
    border-top: 8px solid #3498db;
    border-radius: 50%;
    width: 60px;
    height: 60px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
/* Weather data card styling */
.weather-card {
    background-color: white;
    padding: 20px;
    margin-right: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    font-size: 18px; /* Increase font size for readability */
    color: #333; /* Use darker color for the text */
    line-height: 1.6; /* Improve line spacing */
    width: 96%; /* Ensure full width */
    margin-top: 20px; /* Add spacing at the top */
}

.weather-card p {
    margin: 5px 0; /* Add margin between lines */
}

/* Temperature styling */
.weather-card .temperature {
    font-size: 24px;
    font-weight: bold;
    color: #3498db; /* Highlight temperature in blue */
}

const express = require('express');
const path = require('path');

const app = express();
```

```
const port = 3000;

__dirname = path.resolve();

// Serve static files from the 'public' directory
app.use(express.static(path.join(__dirname, 'public')));

app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```