Client.html

```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>Weather App</title>
   <script type="importmap">
       {
           "imports": {
               "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"
           }
       }
   </script>
   <script src="https://cdn.tailwindcss.com"></script>
</head>

<body>
   <div id="app" class="p-8">
       <h1 class="text-4xl font-semibold">🌦️
           Weather App</h1>
       <br />
       <p>Please enter the town name for the weather forecast:</p>
       <div class="flex items-center">
           <div class="relative w-full">
               <div class="flex absolute inset-y-0 left-0 items-center pl-3
pointer-events-none">
                   <svg aria-hidden="true" class="w-5 h-5 text-gray-500 "
fill="currentColor" viewBox="0 0 20 20"
                       xmlns="http://www.w3.org/2000/svg">
                       <path fill-rule="evenodd"
                           d="M8 4a4 4 0 100 8 4 4 0 000-8zM2 8a6 6 0 1110.89
3.476l4.817 4.817a1 1 0 01-1.414 1.414l-4.816-4.816A6 6 0 012 8z"
                           clip-rule="evenodd"></path>
                   </svg>
               </div>
               <input v-model="town" type="text" v-model="town"
                   class="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full pl-10 p-2.5 "
                   placeholder="Enter a town name..." required />
           </div>
           <button v-on:click="getWeatherForecast"
               class="inline-flex items-center py-2.5 px-3 ml-2 text-sm font-medium
text-white bg-blue-700 rounded-lg border border-blue-700 hover:bg-blue-800 focus:ring-4
focus:outline-none focus:ring-blue-300 dark:bg-blue-600 dark:hover:bg-blue-700
dark:focus:ring-blue-800">
               <svg aria-hidden="true" class="mr-2 -ml-1 w-5 h-5" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
                   xmlns="http://www.w3.org/2000/svg">
                   <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
                       d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z"></path>
               </svg>Search
           </button>
       </div>
       <!-- <button v-on:click="getWeatherForecast"
           class="h-10 px-6 font-semibold rounded-full bg-violet-600 text-white">Show me
the weather!</button> -->
       <span v-if="isError">
           <br />
           <br />
           <font color="red">{{errorMsg}}</font>
       </span>

       <!-- Rain Packing Tips -->
       <div v-if="!isError && doesRain != null">
           <p>
           <h2 class="text-lg font-semibold">☔ Rain:</h2>
           <span v-if="doesRain">Pack an Umbrella, it might rain.</span>
           <span v-if="!doesRain">No need to pack an Umbrella, it doesn't look like it's
gonna rain.</span>
           </p>
       </div>

       <!-- Weather Type Packing Tips -->
       <div v-if="!isError && temperatureAnalysis != null">
           <p>
           <h2 class="text-lg font-semibold">👔 Packing:</h2>
           <span v-if="temperatureAnalysis.weatherType == 'hot'">It's gonna be hot (more
than 20°C). Pack light
               clothes to keep cool.</span>
           <span v-if="temperatureAnalysis.weatherType == 'warm'">It's gonna be
warm({{temperatureAnalysis.max}}°C -
               {{temperatureAnalysis.min}}°C). Pack some extra layers incase it gets
colder.</span>
           <span v-if="temperatureAnalysis.weatherType == 'cold'">It's gonna be cold
               ({{temperatureAnalysis.max}}°C - {{temperatureAnalysis.min}}°C). Pack
some winter clothes to keep
               warm.</span>
           </p>
       </div>

       <!-- Mask Advise -->
       <div v-if="!isError && maskAdvised != null">
           <p>
           <h2 class="text-lg font-semibold">😷 Mask Advise: </h2>
           <span v-if="maskAdvised">High Air Pollution Expected (PM2_5 > 10). You should
wear a Mask
               outside.</span>
           <span v-if="!maskAdvised">There is not much air pollution these dates. No
need to wear a mask!</span>
           </p>
       </div>

       <table v-if="!isError && weatherJSON" class="w-full text-sm text-left
text-gray-500 ">
           <thead class="text-xs text-gray-700 uppercase bg-gray-5">
               <tr>
                   <th scope="col" class="border border-slate-300">Date</th>
                   <th scope="col" class="border border-slate-300">Avg. Temperature
(°C)</th>
                   <th scope="col" class="border border-slate-300">Highest (°C)</th>
                   <th scope="col" class="border border-slate-300">Lowest (°C)</th>
```

```
                    <th scope="col" class="border border-slate-300">Wind Speed (m/s)</th>
                    <th scope="col" class="border border-slate-300">Air Pollution
PM2_5</th>
                    <th scope="col" class="border border-slate-300">Rainfall Level
(mm)</th>
                </tr>
            </thead>
            <tbody>
                <template v-for="(forecast, date) in weatherJSON">
                    <tr>
                        <td scope="row" class="border">{{ date }}</td>
                        <td class="border">{{ forecast.avgTemp }}</td>
                        <td class="border">{{ forecast.temperatureRange.min }}</td>
                        <td class="border">{{ forecast.temperatureRange.max }}</td>
                        <td class="border">{{ forecast.avgWind }}</td>
                        <td class="border">{{ forecast.avgPM2_5 }}</td>
                        <td class="border">{{ forecast.rainfallLevels }}</td>
                    </tr>
                </template>
            </tbody>
        </table>
    </div>
</body>

</html>


<script type="module">
    import { createApp } from 'vue'
    createApp({
        data() {
            return {
                message: '',
                count: 0,
                town: '',
                isError: false,
                errorMsg: '',
                weatherJSON: null,
                doesRain: null,
                maskAdvised: null,
                temperatureAnalysis: null,
            }
        },
        methods: {
            getWeatherForecast() {
                console.log(`Requesting weather forecast for ${this.town}...`);

                fetch(`http://localhost:3000/weather/${this.town}`)
                    .then((response) => {
                        // if the response is ok
                        if (response.status === 200) return response.json();
                        else throw Error(response.statusText);
                    }).then(responseJSON => {
                        console.log(responseJSON);
                        this.weatherJSON = responseJSON.forecastData;
                        this.temperatureAnalysis = responseJSON.temperatureAnalysis;
                        this.doesRain = responseJSON.doesRain;
                        this.maskAdvised = responseJSON.maskAdvised;
                        // Reset any errors
                        this.isError = false;
                        this.errorMsg = '';
                    })
                    .catch(error => {
                        console.error(error);

                        // Set error messages
                        this.isError = true;
                        this.errorMsg = `Unable to fetch weather data for ${this.town}`;
                    });
            }
        }
    }).mount('#app')
</script>
```

Sever.js

```
// console.log('Vue App Backend')
require("dotenv").config()

const axios = require('axios')
const cors = require('cors');
const express = require('express');
const { get } = require("http");

// Setting up Express App
const app = express();
app.use(cors());
const port = 3000

// API Key from .env file and the base url
const base_url = `https://api.openweathermap.org/data/2.5`
const API_key = process.env.API_key

// Some helper functions
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 kelvin_to_celsius = k => (k < 0) ? '0K' : Math.round((k - 273.12)
* 100) / 100;
const min_max = (arr) => {
   const min = kelvin_to_celsius(Math.min(...arr));
   const max = kelvin_to_celsius(Math.max(...arr));
   return { min: min, max: max }
}


app.get('/', (req, res) => res.send('Weather App Server Side'));
app.get('/weather/:town', getForecast);

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

// Air Pollution PM2_5 Analysis
function getMaskAdvise(forecastData) {
   var pm2_5 = [];
   for (forecastDate in forecastData) {
       if (forecastData[forecastDate].avgPM2_5 === null ||
forecastData[forecastDate].avgPM2_5 === undefined)
           pm2_5.push(parseInt(0));
       else
           pm2_5.push(parseInt(forecastData[forecastDate].avgPM2_5));
   }
   // console.log(pm2_5)
   const pm2_5_avg = average(pm2_5);
   return (pm2_5_avg > 10);
}

// Temperature Analysis - hot/warm/cold
function getTemperatureAnalysis(forecastData) {
   let max = 0;
   let min = forecastData[Object.keys(forecastData)[0]].avgTemp;
   let weatherType = null;

   let tempRange = {};

   for (forecastDate in forecastData) {
       tempRange = forecastData[forecastDate].temperatureRange;
       if (tempRange.max >= max)
           max = tempRange.max;
       if (tempRange.min <= min)
           min = tempRange.min;
   }

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

   return {
       weatherType: weatherType,
       max: max,
       min: min
   }

}
```

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

    var forecastData = {};
    var doesRain = false;
    var airPollutionData = {};
    var townLat = 0;
    var townLon = 0;


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

            var weatherData = response.data.list;
            // Iterating over each day forecast

            var days = 0
            for (weatherEntry in weatherData) {
                // formatting date
                let date = new Date(response.data.list[weatherEntry].dt *
1000);
                date.setHours(0, 0, 0, 0);
                date = date.toLocaleDateString();
                // Making sure we only have next four days forercast
                if (days > 4) break;
                // Initiliazing if undefined or null
                if (!forecastData[date]) {
                    days++;
                    forecastData[date] = {
                        windSpeeds: [],
                        temperatures: [],
                        rainfallLevels: [],
                    }
                }


forecastData[date].windSpeeds.push(weatherData[weatherEntry].wind.speed)
;

forecastData[date].temperatures.push(weatherData[weatherEntry].main.temp
);
```

```
                // Check if there is any rain
                if (weatherData[weatherEntry].rain &&
weatherData[weatherEntry].rain['3h']) {
                    doesRain = true;

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

            }

        }
    ).then(() => {

axios.get(`${base_url}/air_pollution/forecast?lat=${townLat}&lon=${townL
on}&APPID=${API_key}`).then((response1) => {
            const airPollutionData = response1.data.list;
            var days = 0
            for (airPollutionEntry of airPollutionData) {
                let date = new Date(airPollutionEntry.dt * 1000);
                date.setHours(0, 0, 0, 0);
                date = date.toLocaleDateString();
                // Making sure we only have next four days forercast
                if (days > 4) break;
                // Initiliazing if undefined or null
                if (!airPollutionData[date]) {
                    days++;
                    airPollutionData[date] = {
                        pm2_5: []
                    }
                }
                // console.log(airPollutionEntry.components.pm2_5)

airPollutionData[date].pm2_5.push(parseInt(airPollutionEntry.components.
pm2_5))
                // console.log(`${date} - PM2_5 -
${airPollutionEntry.components.pm2_5}`)
            }


            //  Calculating averages once compiled

            for (forecastDate in forecastData) {

                // console.log(airPollutionData[forecastDate].pm2_5)
                forecastData[forecastDate].avgTemp =
kelvin_to_celsius(average(forecastData[forecastDate].temperatures));
```

```
                forecastData[forecastDate].temperatureRange =
min_max(forecastData[forecastDate].temperatures);
                forecastData[forecastDate].avgWind =
average(forecastData[forecastDate].windSpeeds);
                forecastData[forecastDate].rainfallLevels =
sum(forecastData[forecastDate].rainfallLevels);
                if (airPollutionData[forecastDate] !== null &&
airPollutionData[forecastDate] !== undefined)
                    forecastData[forecastDate].avgPM2_5 =
average(airPollutionData[forecastDate].pm2_5);
            }

            // Get overall temperature weatherType and air pollution
analysis
            temperatureAnalysis = getTemperatureAnalysis(forecastData);
            maskAdvised = getMaskAdvise(forecastData);



            res.json({
                forecastData: forecastData,
                doesRain: doesRain,
                temperatureAnalysis: temperatureAnalysis,
                maskAdvised: maskAdvised
            })


        }).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!"
        });
    })

}
```