Introduction

The objective of this assignment was to develop an Internet Application that displays a 5-day weather forecast to the user. It requests the city that the person intends to travel to and displays the weather for the next 5 days for that city.

Technologies Used

In summary, the technology stack I employed for this assignment encompasses Vue.js for the frontend and Express.js for the backend.

Frontend:

The frontend of my weather web application relies on Vue.js, as required for the assignment. It is a highly regarded JavaScript framework for building user interfaces. It excels in creating dynamic, interactive web pages by employing components to manage data and control application behaviour efficiently. Vue components structure the application and facilitate code organization.

For structuring and styling the user interface, my code employs standard HTML and CSS. To enhance the visual aesthetics of the application, I chose the Semantic UI, a CSS framework, along with my internal "styles.css" file. I used the Vue.js directives such as v-model, v-on:click, and v-if for data binding, event handling, and conditional rendering, resulting in a responsive and engaging user experience. This was needed for the packing element of the assignment, for notifying the user if they needed an umbrella for potential rainfall, for temperature alerts and for wearing a mask or not.

The frontend communicates with the backend using the Fetch API, enabling the application to make HTTP requests for the weather and air pollution (PM2_5) data from the openweathermap API resource.

Backend:

The backend of my application is constructed using Express.js, which was also a requirement. It is a powerful Node.js web application framework. It helps to simplify the creation of server-side applications by handling incoming HTTP requests and generating appropriate responses. My web application defines two critical routes, '/forecast/:city' and '/air_pollution/forecast/:lat/:lon,' which are responsible for managing requests for the weather forecast and air pollution (PM2_5) data. Express.js adeptly manages the routing and processing of these requests.

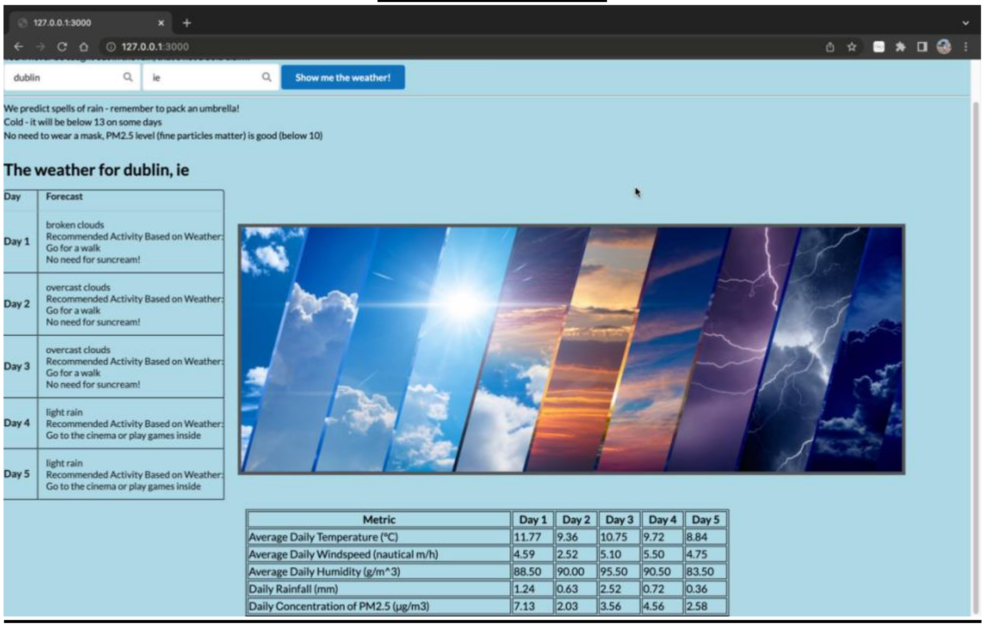
API Integration:

The application integrates external data through the OpenWeatherMap API, a reliable source for weather forecast information. My API key, which is a very sensitive piece of information, is securely stored in an environmental variable in my dot env file. For marking purposes, my API key is: [redacted]

Static Files:

The static files, comprising HTML, CSS, and JavaScript, are made accessible through the Express.js server. These files are stored in my "public" directory and are served by Express.js, guaranteeing that the frontend user interface is readily available to users for a seamless and user-friendly experience.

Design Process



We predict spells of rain - remember to pack an umbrella!
Cold - it will be below 13 on some days
No need to wear a mask, PM2.5 level (fine particles matter) is good (below 10)

The weather for dublin, ie

Day	Forecast
Day 1	broken clouds Recommended Activity Based on Weather: Go for a walk No need for suncream!
Day 2	overcast clouds Recommended Activity Based on Weather: Go for a walk No need for suncream!
Day 3	overcast clouds Recommended Activity Based on Weather: Go for a walk No need for suncream!
Day 4	light rain Recommended Activity Based on Weather: Go to the cinema or play games inside
Day 5	light rain Recommended Activity Based on Weather: Go to the cinema or play games inside

Metric	Day 1	Day 2	Day 3	Day 4	Day 5
Average Daily Temperature (°C)	11.77	9.36	10.75	9.72	8.84
Average Daily Windspeed (nautical m/h)	4.59	2.52	5.10	5.50	4.75
Average Daily Humidity (g/m^3)	88.50	90.00	95.50	90.50	83.50
Daily Rainfall (mm)	1.24	0.63	2.52	0.72	0.36
Daily Concentration of PM2.5 (µg/m3)	7.13	2.03	3.56	4.56	2.58

My server.js file is located in the project root directory, along with package.json, package-lock.json and my .env file. My index.html file and styles.css are kept in a public folder in the project root directory. My index.html client-side performs all of the rendering for getting the API data onto the screen in the frontend, and my server-side server.js performs the API call with the necessary request variables and retrieves the API responses in the backend.

I constantly used the console.log() operation for debugging purposes. This was very helpful as it helped me trace my steps if a certain function was not working as intended.

I wanted to ensure that my weather app was as user friendly as possible, and as efficient as possible at giving essential weather forecast information for a city to the user. I realised that since there are many cities in different countries worldwide that share the same name, I wanted to give the user the ability to be as specific as possible to receive the most accurate information. With this goal in mind, I added an option for the user to input the country code of their desired city. For example, this would differentiate the city of Dublin in Ireland from the city of Dublin in Ohio in the US.

I chose to show two tables for the API data. This was done to separate the numerical data from the descriptive data of each day. The table shown at the bottom of the screen accommodates for the core features of rainfall, temperature and wind speed, the numerical data. My second table, at the left-hand side, made with the Semantic UI framework, outputs the overall description of the weather for each day in the forecast, giving the user a quick overview of what to expect for each day. This was taken from the "description" element of the JSON converted response from the API.

To keep my API safe and secure, I used a dot env file in my project. This was then imported into the Express backend using "require('dotenv').config()" and used as an environmental variable with "const apiKey = process.env.MY_WEATHER_API_KEY;".

I also chose to keep my functions in the client-side HTML separate from the Vue app function. By doing so, I was required to use app.variable_name instead of this.variable_name. I then called all my functions in the Vue app for running my "GetForecast" methods when the user opts to be shown the weather.

I added an image between both tables to make the application more appealing to the user. It shows a collage of the different types of weather forecasts, such as clear skies, sun, and lightning.

Core Features

To meet the requirements for this project, I needed to display a range of information to the user through the frontend. I utilised the HTML Table element to store various metrics, such as rainfall measure (in mm), temperature (in Celsius) and the wind speed (m/s), over the 5-day forecast. These are calculated in my packingForWeather function and is further utilised with the v-if directives for notifying the user of temperature conditions and packing recommendations, e.g. pack an umbrella for rain.

For the rainfall level per day, it was important to keep in mind that the "rain" element is not always present in the JSON response. If it is sunny all day, then the "rain" element does not exist. Therefore, I first had to check if the "rain" element existed in the JSON response in my getRainfall() function for each day in the forecast. If it existed, I added the measure for each day to a total rainfall variable and computed the average of it. If it did not exist, I returned zero in the table.

I decided to get the average measure of these weather elements per day, as the API data updates every 3 hours. To display the air pollution (PM2_5) API, I took the latitude and longitude elements from the first API call for the weather forecast and passed them in as variable parameters to my getPollution() function in my client side. This function uses these variables and pulls the corresponding API data from the backend. I then got the average for each day. I introduced a Boolean into this function, called "warning". If the PM2_5 for a city over any day is above 10, it triggers my pollution_mask variable and notifies the user to wear a mask, using the Vue v-if directive.

Additionally, I decided to show the humidity as it was available in the JSON response from the openweathermap API. A high humidity would indicate to the user that it may feel hotter outside than the data shows, which would give them further insight into what kind of day to expect in their desired city.

Unique Feature

My unique feature for my weather application is an activity suggestion for each day, based on the weather. If the weather is sunny, I recommend that the user goes to the beach and gets ice cream and remind them to wear sun cream. If the weather is cloudy, but with slim chances of rain, I recommend that the user goes for a walk, and there is no need for them to apply sun cream. If the weather is predicted to be wet, I suggest that the user visits the cinema or plays games inside. I accomplished this using the Vue directives v-if and v-else-if, along with my sortedWeatherDescriptions API response variable. To check what type of weather is predicted, I utilised the JavaScript String "includes()" method, which checks if the weather description string contains certain substrings. For example, If the description contained the substring “sun”, I concluded that the weather must be sunny, and gave recommended activities accordingly. If it included “rain”, I suggest that the user stays indoors to avoid getting wet. For weather that relates to extreme conditions such as hurricanes and other storms, I recommend the user to consult with their local authorities. This information is displayed in my table that shows the short description for each day in the forecast.