Assignment 1: Weather Web Application

**-KSHITIJ MALVANKAR (A00268796)**

**Introduction:**

The main goal of this assignment is to call the open weather API and get the weather forecast in xml format and then parse the data into the HTML page. The HTML page should have a unique design. The requirements were:

* **Connect to openweathermaps.org API using an AJAX request**
* **Parse the returned XML and present the data in a meaningful way**

**Challenges Faced:**

* Parsing the XML data received.
* Animations
* Setting the Cards in proper layout.

**Features:**

1. **Current Weather conditions**

* Current Temperature
* Humidity
* Windspeed
* Wind Description
* Wind Direction
* Weather Description
* Longitude and Latitude

1. **4 Day Weather Forecast**

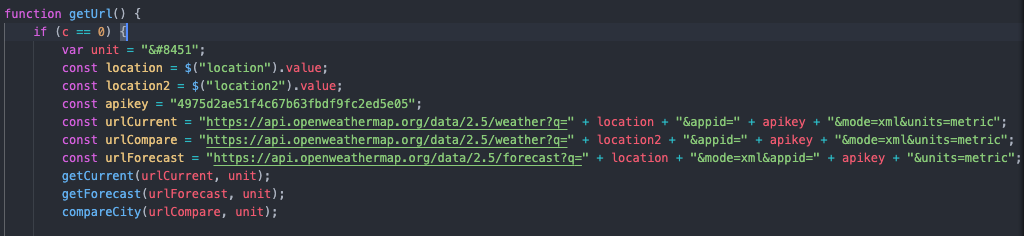
* High Temperature
* Low Temperature
* Wind speed
* Weather Description
* Longitude and Latitude

1. **Comparing Current weather of 2 cities**
2. **Informative charts**

* Max Temperature vs Time
* Pressure vs Time

**Working:**

An AJAX call was used to connect to the openweather API and get the data in XML format. 2 parts of the openweather API are used. 5day/3 hour forecast as well as the current weather forecast. API calls through AJAX are made to both the APIs. With AJAX, a web application can send and receive data from a server asynchronously without interfering with the look or display of the existing page.



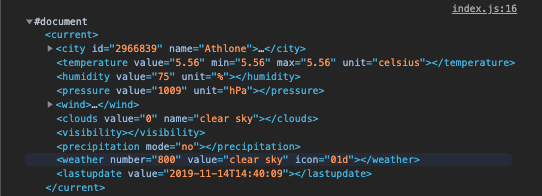
The url of the API was stored in a variable with the location part set to the location we get from the search field on the HTML page and the apikey is the key to use the API.





The url is passed into the “urlCurrent” variable and the request is sent. The data that is retrieved and stored in the variable “data”.

The next step was to parse the received data.

The data in the image above is an example of the xml file that was received after calling the API from AJAX. This is data for current forecast and was parsed first. 



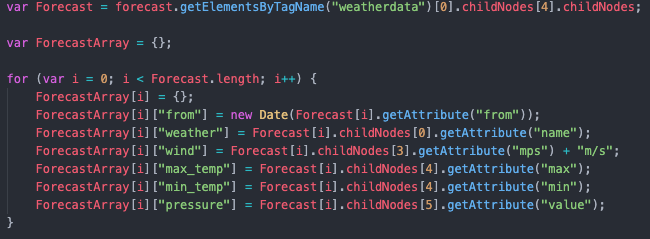
The City name, temperature, the humidity, the wind speed, direction and description were the data sets that were parsed and with the help of DOM scripting, they were injected into the HTML.. To get specific elements from the XML data file, “getElementsByTagName” was used. This returns an array of elements of the same tag. To further parse into the array, ”.childNodes[i]” was used, where “i” is the index of the childNode required.

Eg) var wind = data.getElementsByTagName("wind")[0];

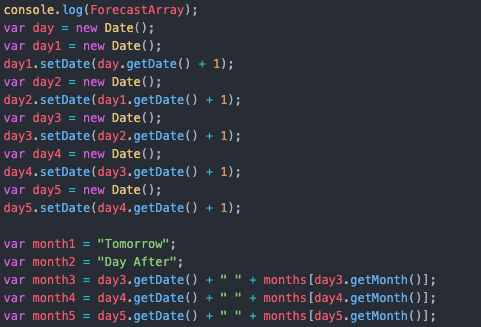
var windchild = wind.childNodes[0];

The next step was to show the forecast for the next 4 days. This was a challenging step as The 5 day/ 3 hour data from openweather was used. This returns the data for the next 5 days at 3 hour intervals. The goal was to show the Min and Max temperatures for the next 4 days. The data returned has many fields, but only a few fields were required.

A 2d array was made to store the required data. Data was parsed from the XML file that was returned after making the call to the 5day/3hour forecast API.



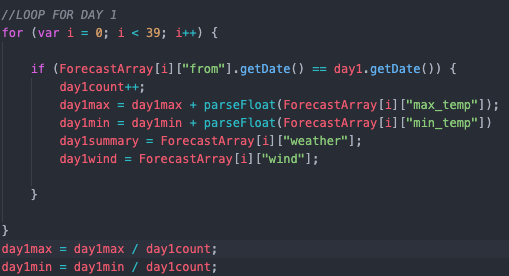
The current date was stored and the dates for the following days was obtained through that.



There are ideally 8 data sets for each day, but this number varies on the time of the day when you check the weather. For example, if you check the weather at 12 am, you will get 8 sets of data, but if you check at 6 pm that same day, you will get only 2 sets of data. The variables “month1”, “month2” etc are used to display the date and month on the forecast.

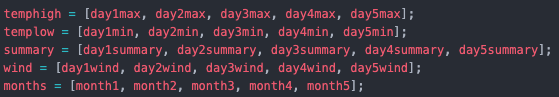
The getDate() function will return the date value and the getMonth() will return an integer value between 0 and 11. Using this value as an index, The month was obtained from the “months” array.

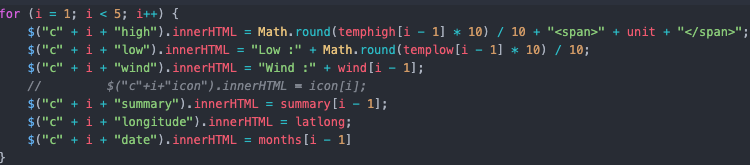
Each set of data has the date in it. The date was matched with the date that was obtained above and a variable was used to count the number of sets for that day. To get an average Min and Max values, the values from all sets were added and then divided by the count of sets. Code shown in the image below.



This was repeated for all the days required. The summary of the day and the wind speed was also obtained from the same loop.

Arrays were then made to store a value of the same type. For example, all the Max temp value in one array, all Min temp values in another array and so on.



Using a for loop, the values from the arrays were set into the HTML. 

The next feature was to compare the temperature of two cities. The same logic was applied that was used to get the forecast of the current weather.

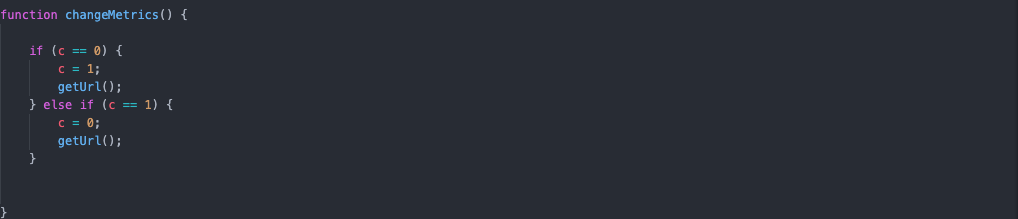
Tabs were used for displaying different content on the same HTML page. Followig is the code for switching between the tabs:

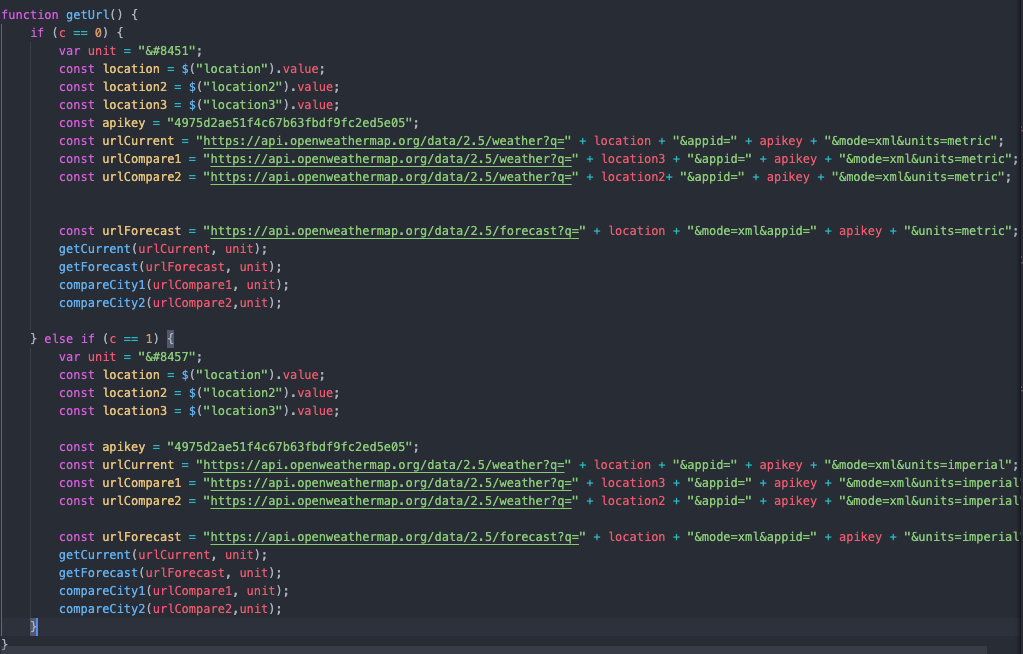


**Note:**

The last if statement in the code snippet above is used to hide the search bar on the page where 2 cities are being compares.

A function to change the units from Imperial to Metric is also implemented in the application. A variable “c” has been set to 0 at the start.

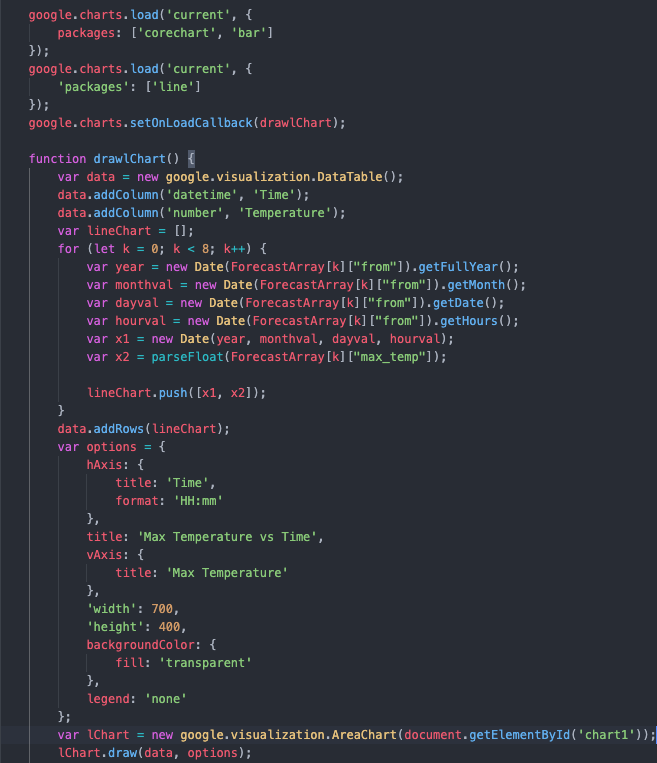




Based on the value of “c”, the units were being set to either imperial or metric and the appropriate URL was being sent to the AJAX function.

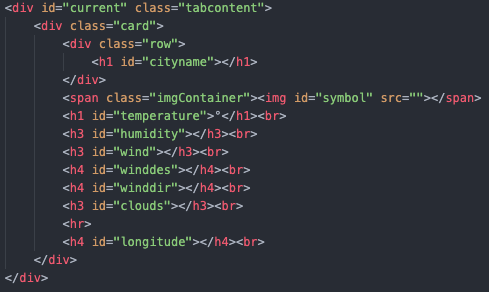
The final feature of the application was charts. Google charts was used for this. Google charts is a framework that lets you easily add charts to your application. Two charts were added to the application. One was a line chart of the Max temp over a period of 24 hours and the second was a bar graph of the pressure over a period of 24 hours.

As image below shows the code to get a line chart into the application. A foor loop is being created and year, month, day, hour values are used to make a new date variable “x1”. A variable “x2” is created in the same loop that contains the Max temperature. The variables are then pushed into the array “lineChart”.The object “options” is then created to hold the various options for the charts. A new variable is created to draw the chart and the id of the div where the chart is supposed to be is passed into that. The draw mehod is then called for the variable with the array and the options variables are passed as parameters. The same process was followed to create the bar chart.



**Design of the Application:**

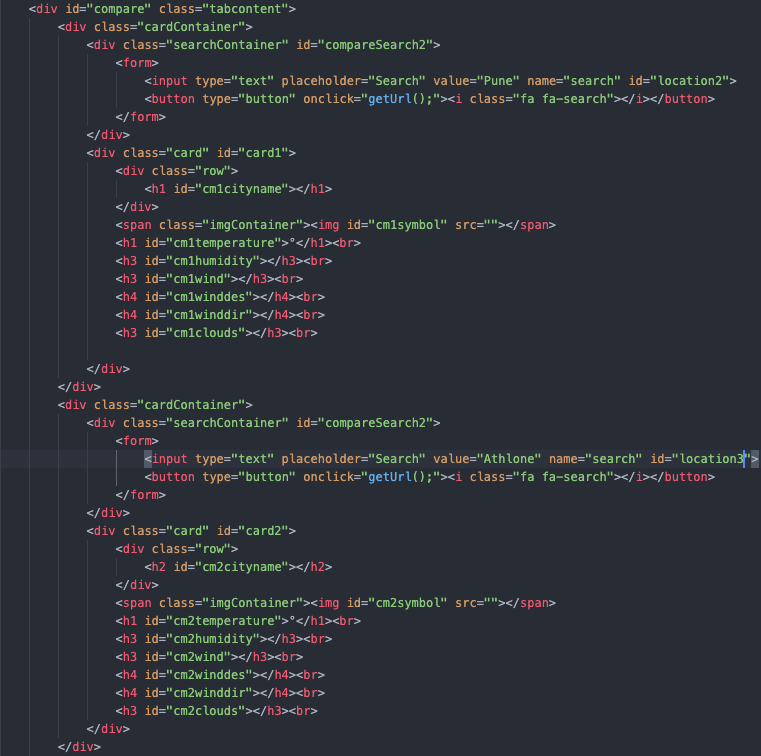
Tabs were used in the HTML design. The main HTML docuent was divided into 3 parts: header, main and footer. The header of the page includes a logo for the application. It also includes a search bar to search for the city for the weather and a button to change the units. The header also containts the tabs that are used.

The main was dividen into 4 tabs. The first tab has the current weather information. Cards format was used to display all types of information. The first tab only has one card. 

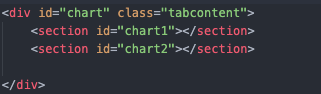
The second tab was the forecast information for the next 4 days. Card format was used to display the data. The following image shows the HTML code for 2 cards. The same was repeated for the remaining 2 cards.



The third tab contained the comparing cities feature. Two cards were used to showcase the information for 2 cities. This tab also includes two search bars for each city respectively.



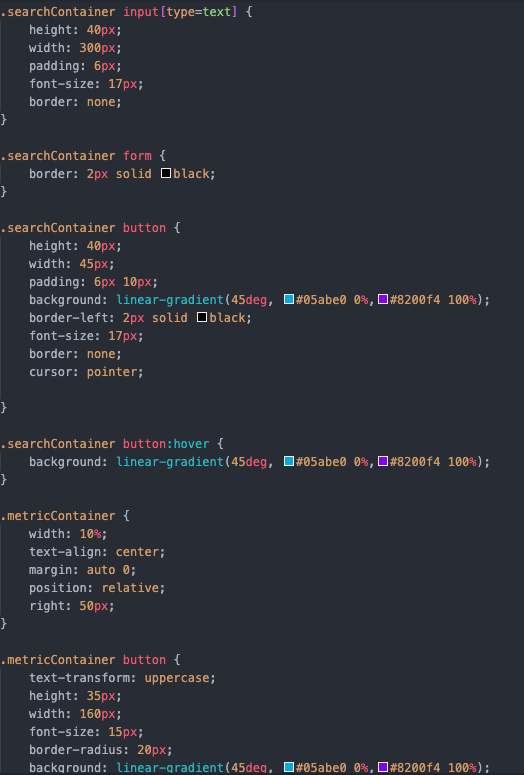
The last tab was used for displaying the charts.



**CSS:**

CSS playes an important role I any web page as it defines how everything looks.

The following code snippet is for the search bar.

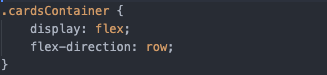
****

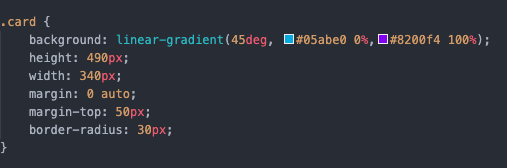
**.searchContainer input[type=text].** This identifier identifies all the input tags with type = text in the searchContainer class.

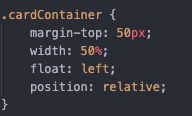
**.searchContainer button:hover**. This identifier identifies the behaviour that should happen when we hover over a button in th searchContainer class.

The following code is for the tab links.



The following code is for cards: 

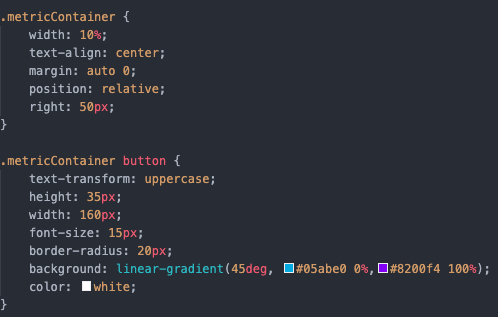


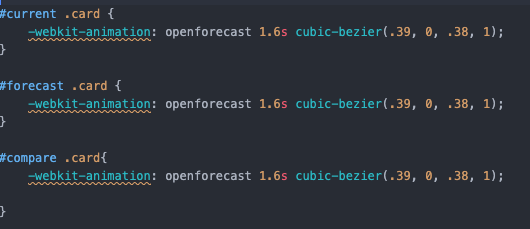


Keyframes has been used to make the cards look like they’re opening in a top-down manner.



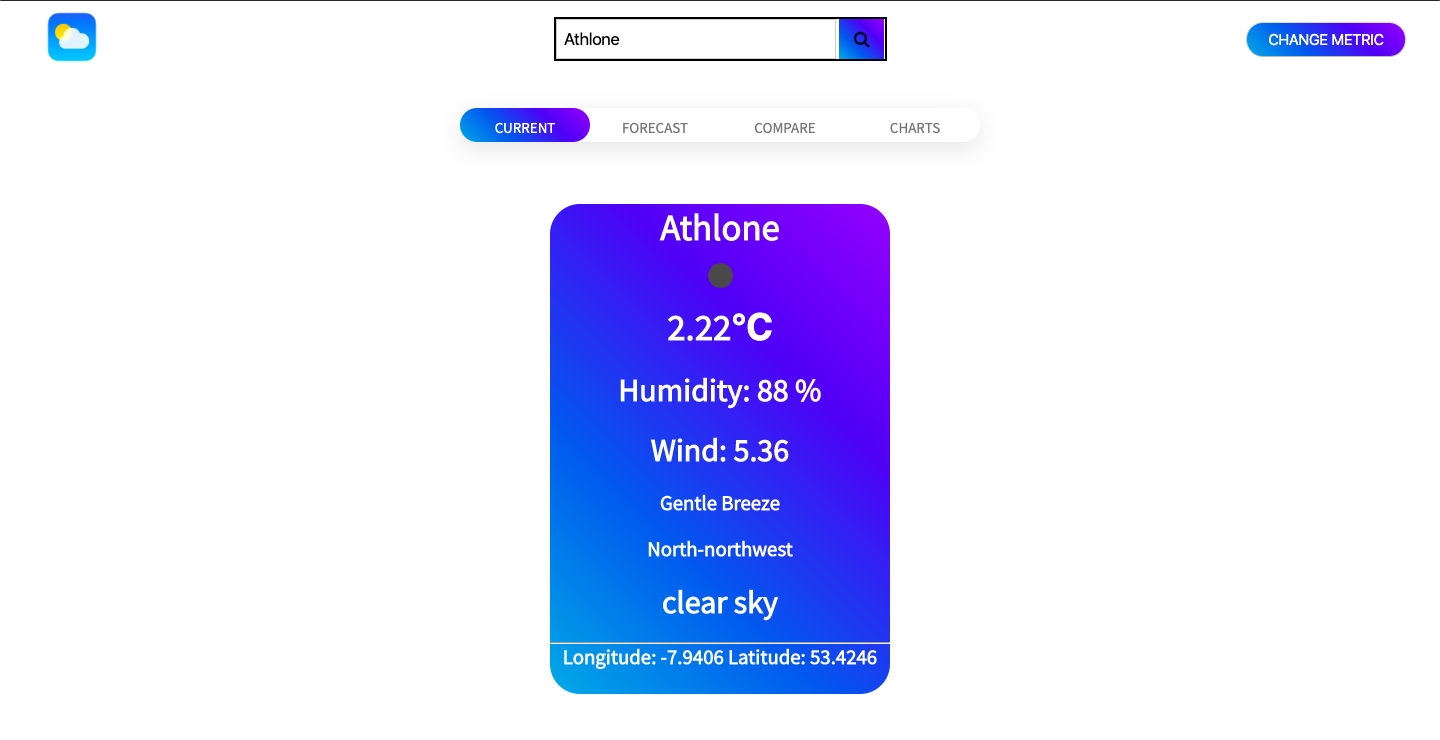
The following css code is for the change metrics button:

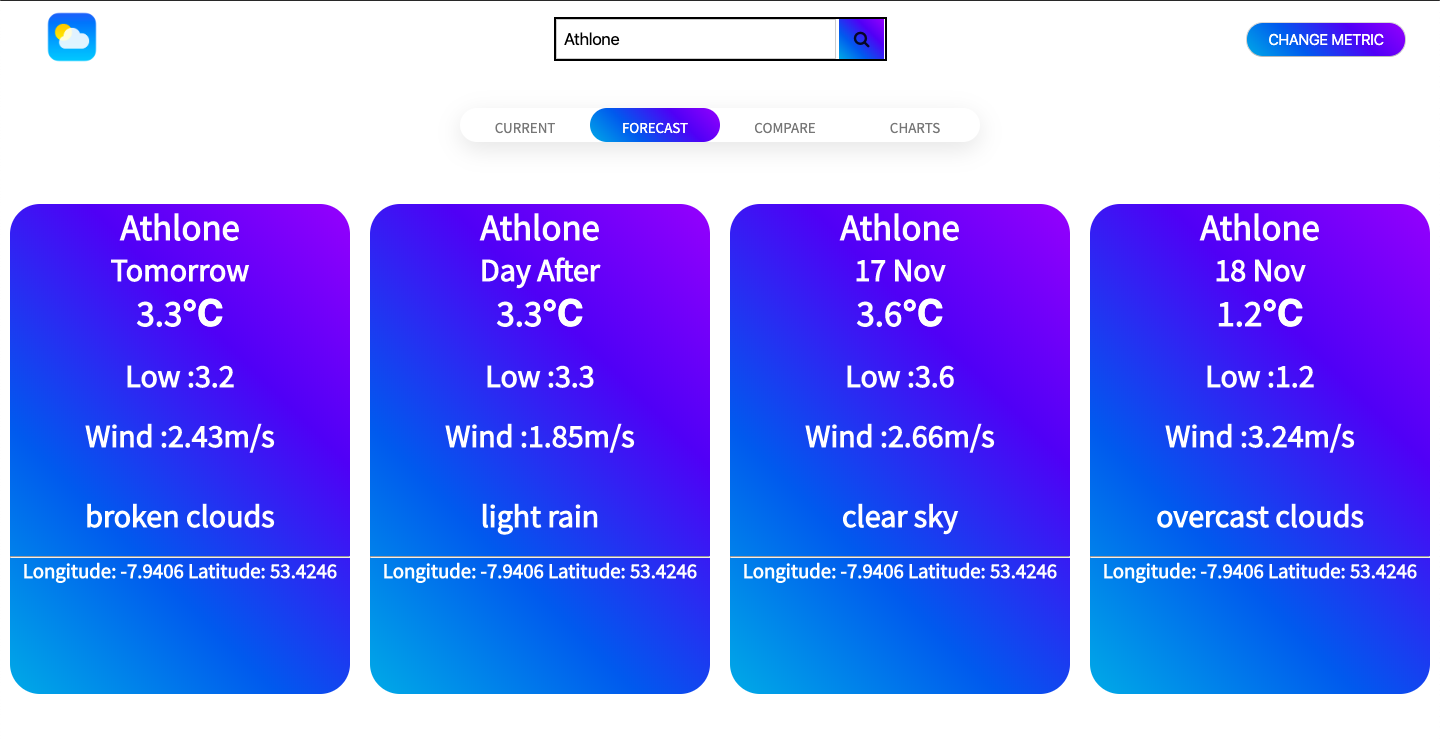


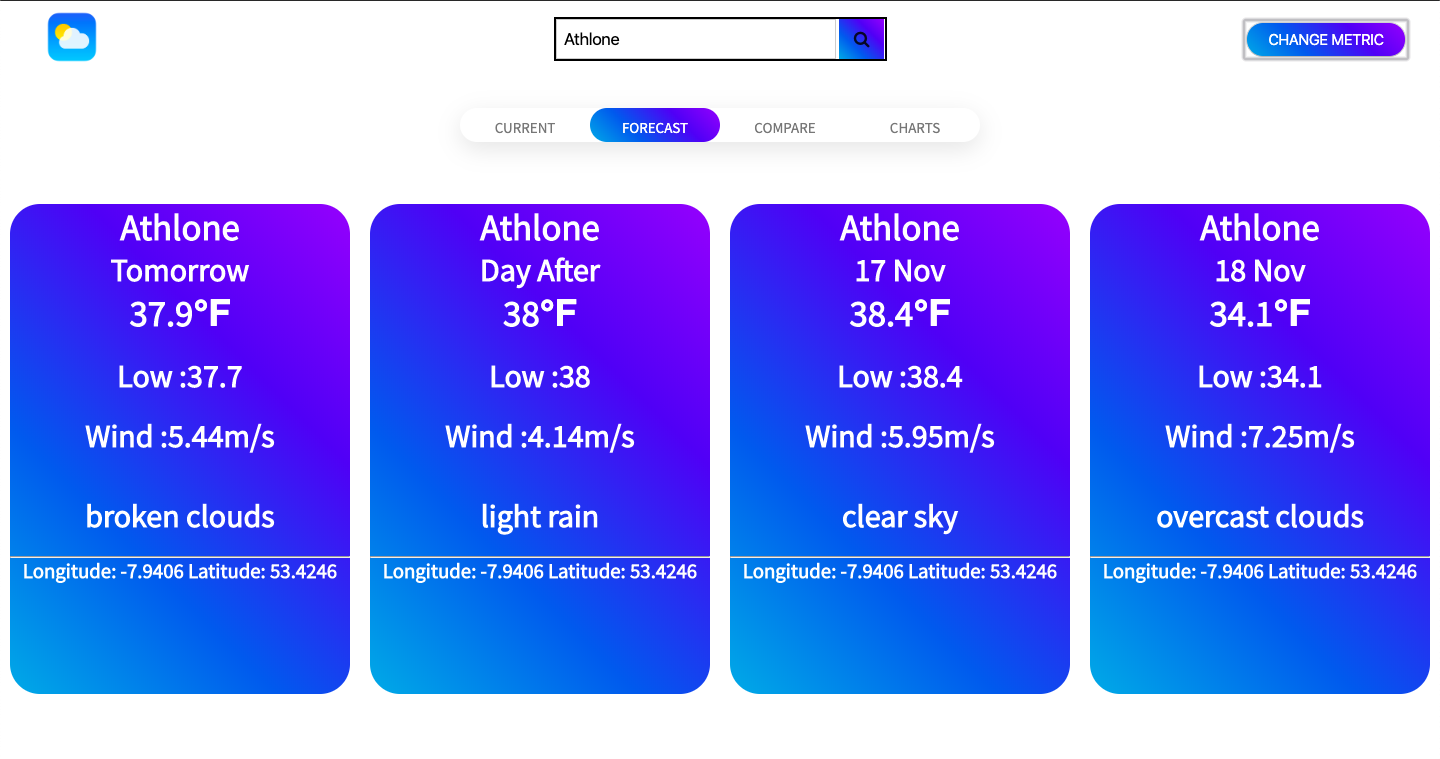
The cubic-bezier function was used for the animation. The cubic-bezier() function defines a Cubic Bezier curve.A Cubic Bezier curve is defined by four points P0, P1, P2, and P3. P0 and P3 are the start and the end of the curve and, in CSS these points are fixed as the coordinates are ratios. P0 is (0, 0) and represents the initial time and the initial state, P3 is (1, 1) and represents the final time and the final state.

Animation is the short-hand property for all animations. The animation name is “openforecast” and will run for1.6 seconds. The cubic-bezier function is used to define how the animation speed will change over time.

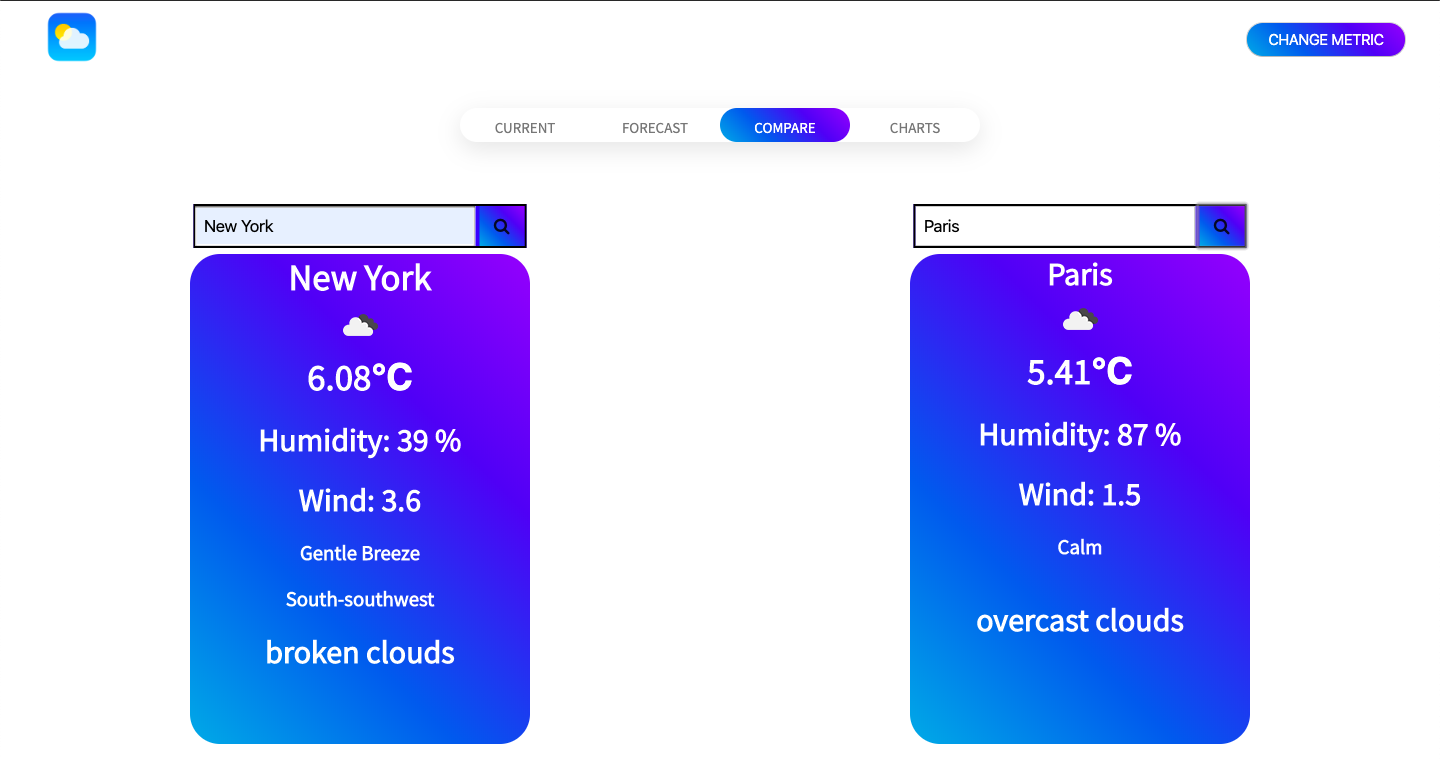
**Final Output**

****









**References:**

* **w3schools.com**
* **openweathermap.org**
* **developers.google.com/charts**
* **Burhan Mullamitha**