

## Weather App

- Started by making the index page (main page), which has a header, with 3 links to the side.

The 3 links that lead to the home page and lead to the two other pages of this website. Then, the home page also has a search bar that allows you to search any city, and have it appeared on the map. The map was taken by the openweather API, and the weather information is also taken from the API. Lastly the footer also has the the 3 links for each webpage.

```
index.html > html > head > link
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <title>Weather</title>
7   <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
8
9   <link rel="icon" href="images/weather-news.png">
10  <meta name="viewport" content="width=device-width, initial-scale=1">
11  <link rel="stylesheet" type="text/css" media="screen" href="styles.css">
12
13 </head>
14 <body>
15   <!-- Header -->
16   <header>
17     <h1>Weather App</h1>
18     <!-- The navigation on the right side of header with 2 links -->
19     <nav>
20       <a href="index.html">Home</a>
21       <a href="save.html">Saved Cities</a>
22       <a href="forecast.html">Forecast</a>
23     </nav>
24   </header>
25
26   <main>
27     <!-- Search bar to search the city. -->
28     <div id="searchCity">
29       <input type="text" id="inputCity" placeholder="Enter city name">
30       <button id="searchButton">Search</button>
31     </div>
32
33     <!-- This is where the IFrame will be located/ -->
34     <section id="mapSection">
35       <h2>Select a Location on the Map</h2>
36       <div id="map" style="width: 100%;></div>
37     </section>
38
39     <!-- This section will hold weather info -->
40     <section id="weatherDisplay">
41
42
43
44
45
46
47
48
49
50   </main>
51
52   <!-- Footer -->
53   <footer>
54     <nav> <!--Links-->
55       <a href="index.html">Home</a>
56       <a href="save.html">Saved City</a>
57       <a href="forecast.html">Forecast</a>
58     </nav>
59     <p>&copy; 2023 Weather App</p>
60   </footer>
61
62   <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
63
64   <script src="script.js"></script>
65 </body>
66 </html>
```

The leaflet styles and leaflet script is for the map put on the webpage. Makes the map interactive.

This is the javascript for the index page:

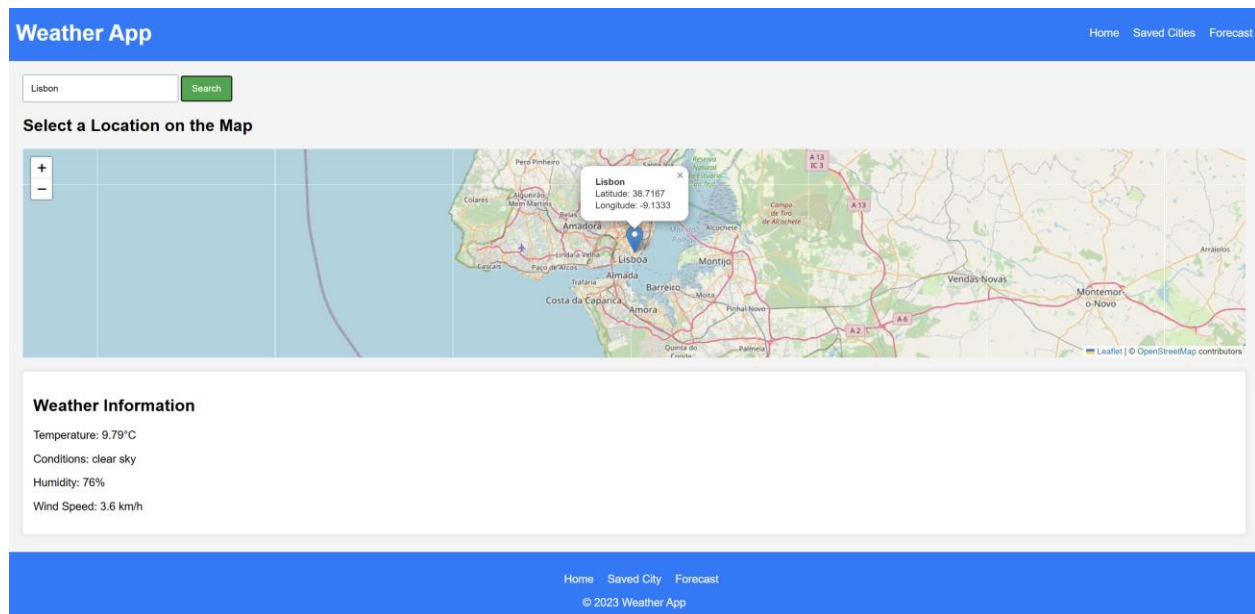
```

15 script.js > document.addEventListener("DOMContentLoaded") callback > searchButton.addEventListener("click") callback
1 // Ensure DOM is loaded before executing scripts
2 document.addEventListener("DOMContentLoaded", () => {
3     const API_KEY = "86253551bBe42e3573ca08d5326f1392"; // OpenWeather API Key
4
5     // ----- INDEX PAGE FUNCTIONALITY----- //
6
7     // adds a map on the webpage where the #map div is located
8     if (document.body.contains(document.getElementById("map"))) {
9         const map = L.map('map').setView([51.505, -0.09], 5);
10        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
11            attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
12        }).addTo(map);
13
14        const searchButton = document.getElementById('searchButton');
15        const inputCity = document.getElementById('inputCity');
16        // event handler -> when the search button is created, it searches the city typed in in the searchbox
17        searchButton.addEventListener('click', () => {
18            const cityName = inputCity.value.trim();
19            if (!cityName) {
20                alert("Please enter a city name!"); // if user doesn't put any city and just presses search, it will give an alert message.
21                return;
22            }
23
24            // fetched the searched city in the api, and fetches it using AJAX
25            fetch("https://api.openweathermap.org/data/2.5/weather?q=${cityName}&appid=${API_KEY}&units=metric")
26                .then(response => response.json())
27                .then(data => {
28                    if (data.cod === 200) {
29                        const lat = data.coord.lat;
30                        const lon = data.coord.lon;
31
32                        // Update map view and add marker to the city
33                        map.setView([lat, lon], 10);
34                        L.marker([lat, lon]).addTo(map)
35                            .bindPopup(`<b>${cityName}</b><br>Latitude: ${lat}<br>Longitude: ${lon}`)
36                            .openPopup();
37
38                        // Display weather information
39                        document.getElementById("temperature").innerText = `Temperature: ${data.main.temp}°C`;
40                        document.getElementById("conditions").innerText = `Conditions: ${data.weather[0].description}`;
41                        document.getElementById("humidity").innerText = `Humidity: ${data.main.humidity}%`;
42                        document.getElementById("windSpeed").innerText = `Wind Speed: ${data.wind.speed} km/h`;
43                    } else {
44                        alert("City not found. Try again.");
45                    }
46                }) // is there is an error fetching the data, or the city typed doesn't exists.
47                .catch(err => {
48                    console.error("Error fetching weather data:", err);
49                    alert("Unable to fetch weather data. Please try again.");
50                });
51        });
52    });
53

```

Event listener for “DOMContentLoaded” makes sure that the entire DOM HTML is loaded. Then, it puts the map from the api on the page on the div “map” where the world map will be located. There is also a search feature, where when the user searches for a city, it will fetch the city weather information in an API, using AJAX. Once the data is fetched, it will appear on the weather information div and will display each information. Also, if no city was searched or city searched doesn’t exists, it will give an alert message.

## First Page Result:



- The Second page is the saved city page, where the user can enter a city and save it in a list of cities, where the weather is displayed for each city. This is implemented so the user can see the weather of the city he likes to see every time.

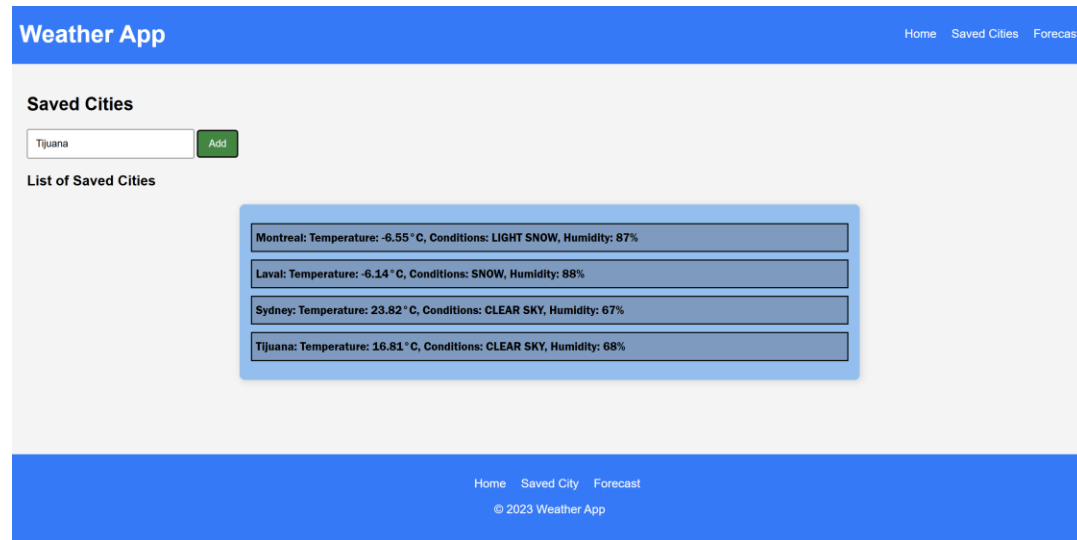
```

15 script.js > document.addEventListener("DOMContentLoaded") callback > updateForecastCards
2 document.addEventListener("DOMContentLoaded", () => {
3
4 // ----- SAVE PAGE FUNCTIONALITY ----- //
5
6 if (document.body.contains(document.getElementById("savedCities"))) {
7   const addButton = document.getElementById("addButton");
8   const inputSavedCity = document.getElementById("inputSavedCity");
9   const savedCitiesList = document.getElementById("savedCities");
10
11   // event listener for save button, trigger the function where it will save the city's weather info in a list.
12   addButton.addEventListener('click', () => {
13     const cityName = inputSavedCity.value.trim();
14     if (!cityName) { // if user doesn't enter anything.
15       alert("Please enter a city name!");
16       return;
17     }
18
19     // fetches from the API using AJAX
20     fetch('https://api.openweathermap.org/data/2.5/weather?q=${cityName}&appid=${API_KEY}&units=metric')
21       .then(response => response.json())
22       .then(data => {
23         if (data.cod === 200) {
24           const lat = data.coord.lat;
25           const lon = data.coord.lon;
26
27           // Create list item for the saved city
28           const li = document.createElement("li");
29           // in the list it will have all the weather info.
30           li.innerHTML = `
31             <strong>${cityName}</strong>:
32             Temperature: ${data.main.temp}°C,
33             Conditions: ${data.weather[0].description.toUpperCase()},
34             Humidity: ${data.main.humidity}%
35           `;
36           savedCitiesList.appendChild(li); // it will keep appending it for each city saved in a list.
37         } else {
38           alert("City not found. Try again."); // if the city is not found.
39         }
40       })
41     }
42
43     // If there are any error fetching the data.
44     .catch(err => {
45       console.error("Error fetching weather data:", err);
46       alert("Unable to fetch weather data. Please try again.");
47     });
48   });
49 }
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97

```

The search features is the same as the previous page. The only difference is that once you search a city, it will fetch the weather information from the API and make a `<li>` element where it displays the weather information. Then you can add multiple cities to a list and see all the weather information for each city. This allows the user to see all the cities it wants to see on a regular basis.

The Second webpage:



```

1  <!-- Saved Cities Page -->
2  <html>
3  <head>
12 </head>
13 <body>
14   <header>
15     <h1>Weather App</h1>
16     <!-- The navigation on the right side of header with 2 links -->
17     <nav>
18       <a href="index.html">Home</a>
19       <a href="save.html">Saved Cities</a>
20       <a href="forecast.html">Forecast</a>
21     </nav>
22   </header>
23   <main>
24     <!-- Saved Cities Title -->
25     <section id="savedCitiesSection">
26       <h2>Saved Cities</h2>
27       <!-- Search bar for adding new cities -->
28       <div id="addCity">
29         <input type="text" id="inputSavedCity" placeholder="Enter city name to add"> <!--The search bar-->
30         <button id="addButton">Add</button> <!--The search button-->
31       </div>
32
33       <!-- List of saved cities -->
34       <div id="cityList">
35         <h3>List of Saved Cities</h3>
36         <ul id="savedCities">
37           <!-- Dynamically populated saved cities -->
38         </ul>
39       </div>
40     </section>
41   </main>
42
43   <!-- Footer -->
44   <footer>
45     <nav> <!--Links-->
46       <a href="index.html">Home</a>
47       <a href="save.html">Saved City</a>
48       <a href="forecast.html">Forecast</a>
49     </nav>
50     <p>&copy; 2023 Weather App</p>
51   </footer>
52
53   <script src="script.js"></script>
54
55 </body>
56 </html>

```

- The third webpage is the webpage that shows the forecast for the next 5 days:

```

JS script.js > document.addEventListener("DOMContentLoaded") callback > searchButton.addEventListener("click") callback > then() callback
2 document.addEventListener("DOMContentLoaded", () => {
99
100 // ----- FORECAST PAGE FUNCTIONALITY ----- //
101 if (document.body.contains(document.getElementById("forecastSection"))) {
102     // gets all the info about the where the divs are.
103     const forecastButton = document.getElementById("forecastButton");
104     const forecastInput = document.getElementById("forecastInput");
105     const forecastCards = document.getElementById("forecastCards");
106     const cityDisplay = document.getElementById("cityDisplay");
107
108     // When searching a city
109     forecastButton.addEventListener("click", () => {
110         const cityName = forecastInput.value.trim();
111         if (!cityName) {
112             alert("Please enter a city name!");
113             return;
114         }
115
116         // Fetch forecast data
117         fetch(`https://api.openweathermap.org/data/2.5/forecast?q=${cityName}&appid=${API_KEY}&units=metric`)
118             .then(response => response.json())
119             .then(data => {
120                 if (data.cod === "200") {
121                     updateCityDisplay(cityName); //updates the displays
122                     updateForecastCards(data.list); //updates the card
123                 } else {
124                     alert("City not found. Please try again.");
125                 }
126             })
127             .catch(err => {
128                 console.error("Error fetching data:", err);
129                 alert("Failed to fetch forecast data. Please try again later.");
130             });
131     });
132
133     // Update the city display section
134     function updateCityDisplay(cityName) {
135         cityDisplay.innerHTML = `
136             <h3>${cityName}</h3>
137         `;
138     }
139

```

```

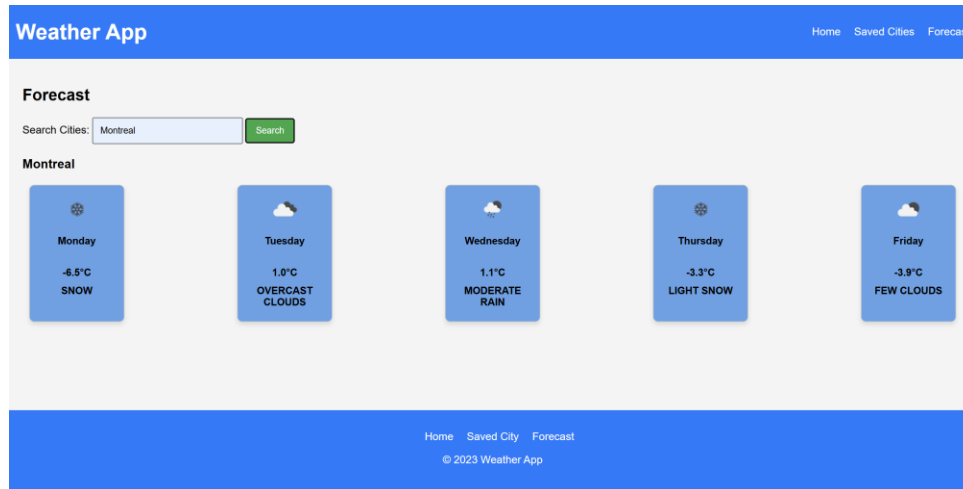
139
140 // It will show the forecast of 5 days
141 // Update the forecast cards
142 function updateForecastCards(forecastList) {
143     const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]; // depending
144     let dayIndex = new Date().getDay(); // So if use
145
146     // Clear previous forecast cards
147     const dayCards = forecastCards.querySelectorAll(".dayCard");
148     dayCards.forEach(card => card.innerHTML = "");
149
150     // Populate new forecast data
151     for (let i = 0; i < 7; i++) {
152         const forecast = forecastList[i * 8]; // Every 8th index gives roughly a day apart
153         if (forecast) {
154             // the information on each card
155             const card = dayCards[i];
156             const iconCode = forecast.weather[0].icon; // the weather icon based on the forecast.
157             const temperature = forecast.main.temp.toFixed(1).toUpperCase(1);
158             const description = forecast.weather[0].description.toUpperCase();
159
160             card.innerHTML =
161             // the icon for the forecast is also taken from openweather, where I got my API
162             `
163                 <div class="weatherIcon">
164                     
165                 </div>
166                 <div class="dayDetails">
167                     <h4>${days[(dayIndex + i) % 7]}</h4>
168                     <p>${temperature}°C</p>
169                     <p>${description}</p>
170                 </div>
171             `;
172         }
173     }
174 }
175 }
176 });

```

The search feature is the same as the previous two pages. When the user presses the button search, it fetches the data from the API and will display the forecast for the next 5 days in the cards. It will show the forecast for the next 5 days based on the day you searched. For example, if you

search on Wednesday, it will show the forecast for Wednesday, Thursday, Friday, Saturday and Sunday. Each card shows an icon of the weather based on the forecast that day, so if it is snowing, it will have a snowing icon. It will also show the Temperature of that day, along side the weather conditions of that day. The icon was taken from OpenWeather, the same website I got the API for the weather information.

The result of the third webpage:



```
1 forecast.html > ...
2 <html>
11 <body>
12 <header>
13 <div>Weather App</div>
14 <!-- The navigation on the right side of header with 2 links -->
15 <div>
16 <a href="#index.html" class="Home">Home</a>
17 <a href="#save.html" class="Saved Cities">Saved Cities</a>
18 <a href="#forecast.html" class="Forecast">Forecast</a>
19 </div>
20 </header>
21 <main>
22 <!-- Forecast Section -->
23 <div>ForecastSection</div>
24 <div>Forecast</div>
25 <div>SearchForecast</div>
26 <div>
27 <input type="text" id="forecastInput" placeholder="Enter city name" /> <!-- The search bar -->
28 <button id="forecastButton" class="Search" value="Search" /> <!-- The search button -->
29 </div>
30
31 <!-- City Display -->
32 <div>CityDisplay</div>
33 <div>City</div>
34 <div>CityImagePlaceholder</div>
35 </div>
36
37 <!-- Daily Forecast Cards -->
38 <div>ForecastCards</div>
39 <!-- Placeholder for 7 day forecast -->
40 <!-- Each DayCard represents a day where it will show the weather of the day -->
41 <!-- Day 1 -->
42 <div>DayCard</div>
43 <div>weatherIcon</div> <!-- Icon will change depending of the weather that day -->
44 <div>dayDetails</div> <!-- Days should change according to current day date -->
45 </div>
46 <!-- Day 2 -->
47 <div>DayCard</div>
48 <div>weatherIcon</div> <!-- Icon will change depending of the weather that day -->
49 <div>dayDetails</div> <!-- Days should change according to current day date -->
50 </div>
51 <!-- Day 3 -->
52 <div>DayCard</div>
53 <div>weatherIcon</div> <!-- Icon will change depending of the weather that day -->
54 <div>dayDetails</div> <!-- Days should change according to current day date -->
55 </div>
56 <!-- Day 4 -->
57 <div>DayCard</div>
58 <div>weatherIcon</div> <!-- Icon will change depending of the weather that day -->
59 <div>dayDetails</div> <!-- Days should change according to current day date -->
60 </div>
61 <!-- Day 5 -->
62 <div>DayCard</div>
63 <div>weatherIcon</div> <!-- Icon will change depending of the weather that day -->
64 <div>dayDetails</div> <!-- Days should change according to current day date -->
65 </div>
66 </div>
```

The CSS for the webpage looks like this:

```
# styles.css > ...
1  /* Main page / Index */
2  body {
3      font-family: Arial, sans-serif;
4      margin: 0;
5      padding: 0;
6      background-color: #f4f4f4;
7      display: flex;
8      flex-direction: column;
9      height: 100%;
10 }
11
12 header {
13     display: flex; /* Use flexbox */
14     justify-content: space-between; /* Space out the items */
15     align-items: center; /* Center items vertically */
16     background: #007bff;
17     color: #fff;
18     padding-left: 10px;
19     padding-right: 10px;
20 }
21
22 nav {
23     display: flex; /* Use flexbox for navigation links */
24 }
25
26 nav a {
27     color: #fff; /* Set link color */
28     text-decoration: none;
29     margin-left: 20px;
30 }
31
32 /* On hover it will change color and underline it. */
33 nav a:hover {
34     text-decoration: underline;
35     color: #00008b;
36 }
37
38 /* The search bar */
39 input[type="text"] {
40     padding: 10px;
41     width: 200px;
42 }
```

```
43  /* search button */
44  button {
45      padding: 10px 15px;
46      background: #28A745;
47      color: #fff;
48      border: none;
49      cursor: pointer;
50  }
51
52  button:hover {
53      background: #218838;
54  }
55  /* Main area with content */
56  main {
57      padding: 20px;
58      flex: 1;
59  }
60
61  /* The Iframe Map section */
62  #mapSection {
63      margin-bottom: 20px;
64  }
65  /* map size */
66  #map {
67      width: 100%;
68      height: 300px;
69  }
70
71  /* Display with weather informations */
72  #weatherDisplay {
73      background: #fff;
74      padding: 15px;
75      border-radius: 5px;
76      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
77  }
78
79  html {
80      height: 100%;
81  }
```



```
82  /* Footer */
83  footer {
84      background-color: #007BFF;
85      padding: 10px;
86      text-align: center;
87      color: #fff;
88      padding: 30px;
89      display: flex;
90      flex-direction: column;
91      align-items: center;
92  }
93
94
95  /* Saved Cities */
96  /* ----- */
97
98
99  /* Makes like a box around the main content */
00  #savedCities {
01      width: 90%;
02      max-width: 800px;
03      margin: 20px auto;
04      padding: 15px;
05      font-size: 16px;
06      background-color: #86c0f2;
07      border: 1px solid #dddddd;
08      border-radius: 8px;
09      box-shadow: 2px 2px 8px rgba(0, 0, 0, 0.1);
10      list-style-type: none;
11  }
12
13  #savedCities li {
14      background-color: #769bc2;
15      font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
16      padding: 5px;
17      margin: 10px 0; /* Add space between individual city entries */
18      line-height: 1.6;
19      border: 1px solid #000000;
20  }
21
22
23  /* Forecast */
24  /* ----- */
25
26  /* Forecast Section Styling */
27  /* Puts the info in some sort of box */
28  .forecast-container {
29      display: flex;
30      flex-wrap: wrap;
31      gap: 1rem;
32  }
```

```
# styles.css > ...
133
134 .forecast-card {
135   background-color: #f0f8ff;
136   border: 1px solid #ccc;
137   padding: 1rem;
138   border-radius: 8px;
139   width: 200px;
140   text-align: center;
141   box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.1);
142 }
143
144 .forecast-card h4 {
145   font-size: 1.2rem;
146   margin-bottom: 0.5rem;
147 }
148
149 #forecastCards {
150   display: flex;
151   justify-content: space-between;
152   flex-wrap: wrap;
153   gap: 10px;
154   margin-top: 20px;
155   margin-left: 10px;
156   margin-right: 10px;
157 }
158
159 .dayCard {
160   background: #61a2e7;
161   border: 1px solid #ccc;
162   border-radius: 8px;
163   padding: 10px;
164   text-align: center;
165   width: 120px;
166   box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
167 }
168
169 .dayCard img {
170   width: 50px;
171   height: 50px;
172 }
173
174 .weatherIcon {
175   margin-bottom: 10px;
176 }
177
178 .dayDetails h4 {
179   font-size: 15px;
180   margin: 10px 0;
181   padding-bottom: 20px;
182 }
183
184 .dayDetails p {
185   margin: 5px;
186   font-size: 15px;
187   padding-bottom: 5px;
188   font-weight: bold;
189 }
190
```