2023

WeatherAPP

DOCUMENTATION 1.0

PHILIPP SCHMID

PHILIPP SCHMID | Flühliweg 34 3624 Goldiwil

Idea Weather Location APP

Table of Contents

Pitch

The WeatherLocation app is there to conveniently display the weather in multiple locations on one single page. The user will be able to customize their list of locations to be displayed so that they can always have a quick glance at all their relevant travel points.

The customization of locations should include adding locations so that they can have as many locations where the weather interests them as they'd like. Of course, just as important as adding is the Possibility of deleting old Locations that are no longer of interest to the user. And for a third option the User will be able to filter their saved locations for various criteria entered by the User so that they can only display the information they truly need.

With this app searching the weather for multiple locations one after the other just to plan a single day's trip is a thing of the past, now you can easily see everything on one page.

Use Cases

In this section we will further explain the Use of this app. The reason why some one wants to sue it is to keep an eye on regional weather for daily cross region travel.

1. The user has to be able to add his preferred locations to the displayed list.
2. The user has to be able to delete a location once it is no longer necessary
3. The user must be able to see the Temperature and rainfall for the selected locations
4. The user must be able to search trough his list of cities by label or city name
5. The app needs to get the updated weather daily from the weatherstack API

Use case	Actor	Description
Location Create	User	This Use case allows the user to add a new location entry to the location list in the weather app for future display. Once created the location will be stored with in the app for future display
Location Delete	User	This Use case allows the User to remove preciously added locations form their weather app, The user selects a location they want to delete, and the program will remove it from the list so it will no longer be displayed in the future.
Weather View	User	This is the main use case of the app and allows the user to see the weather for all the specified locations at once. The app shows detailed weather information for each location.
Location filter	User	This use case allows the User to only display specific locations from the list.

Table 1 Use Cases

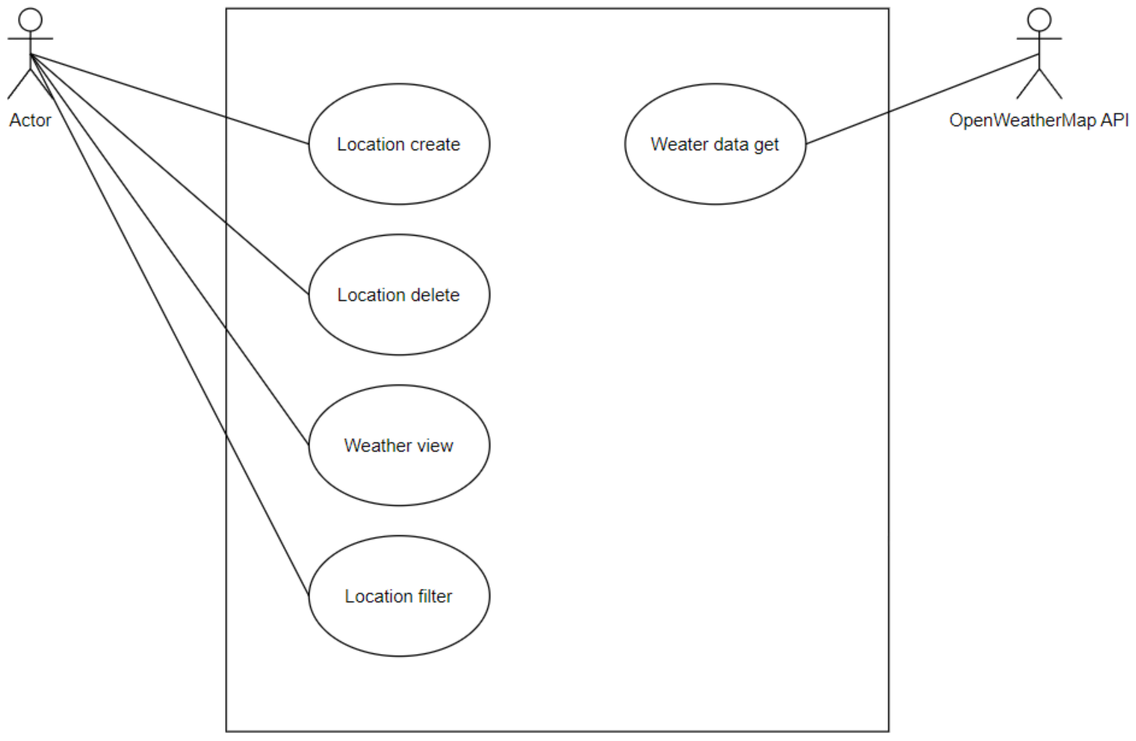


Figure 1 Use Cases

Requirements Catalogue

Functional Requirements

- The user must be able to create new Locations
- The user must be able to delete any and all locations from the list
- The program must be able to save locations
- The user must be able to filter locations by label or by name
- The user must be able to view the Temperature and precipitation in all the saved locations

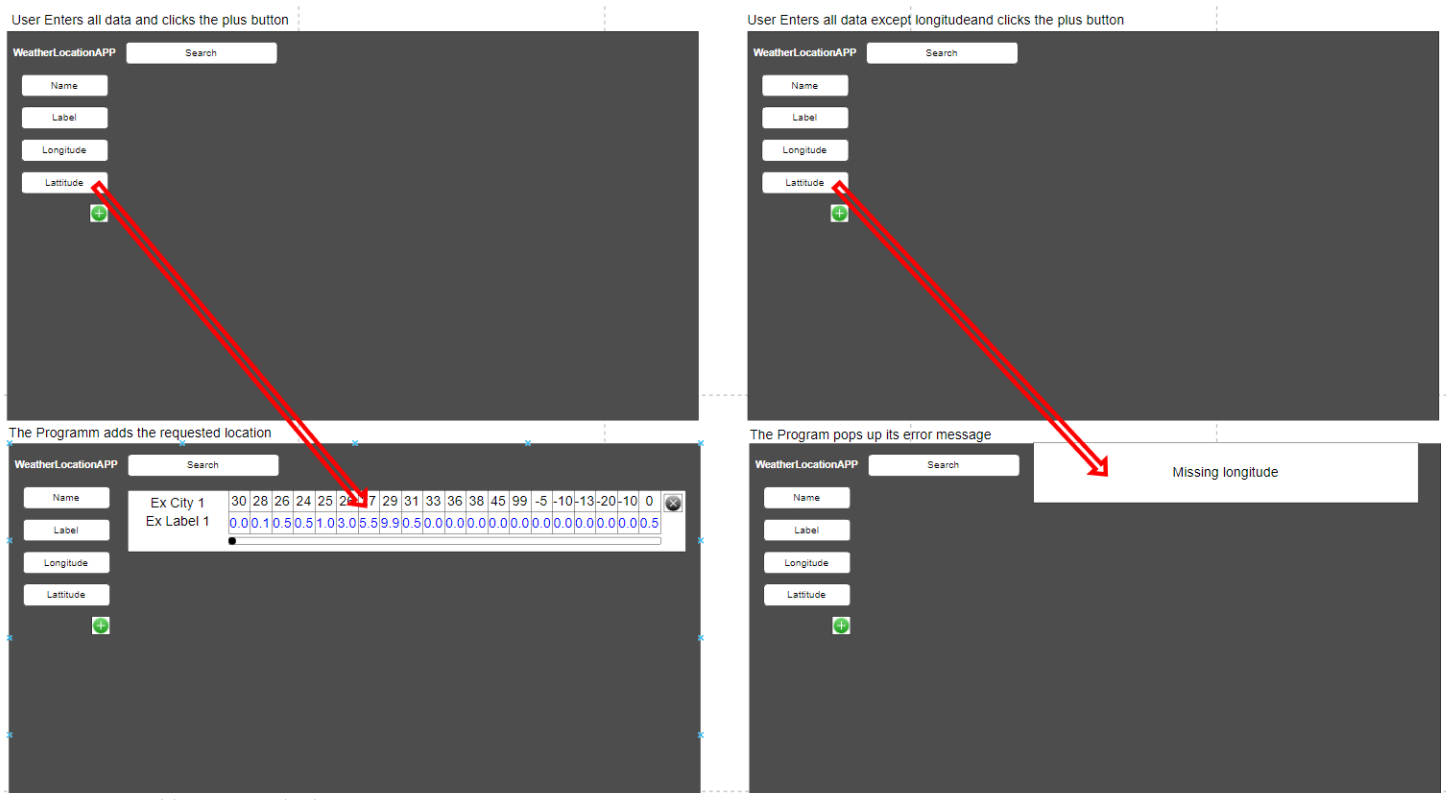
Technical Requirements

- The program must be capable of storing up to 50 Locations that the user can see
- The Program must run on a 1080p resolution
- The website must be displayed on a Firefox

Storyboard

Use case 2

The user can add a location to have the weather displayed, the Program gives feed back if one of the required fields is not filled in. If everything is correct the Program will add dte described Location to the list and display its temperature

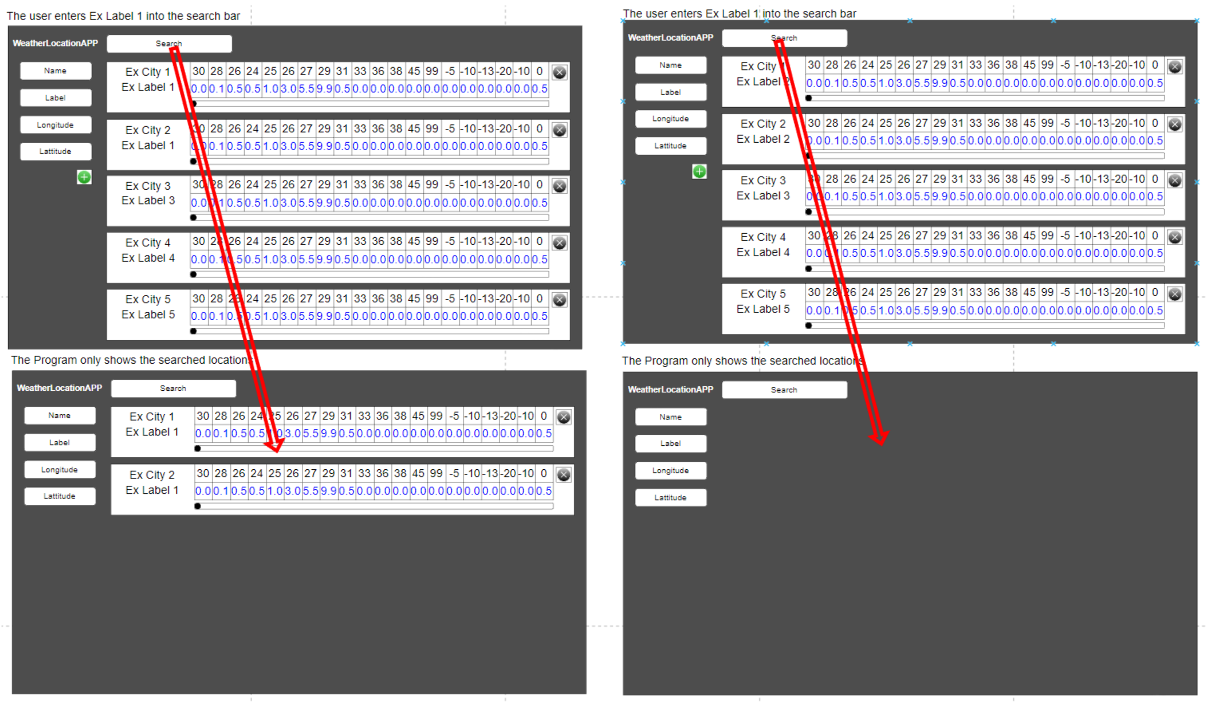


30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

Figure 2 User adds a city to the list

Use case 2

The User searches for “Ex Label 1”. The search bar Takes in a String and searches the City names and labels for a matching string regardless of capitalization, the results will be displayed live as the text is being entered. If no Label or city name matches the inserted string then the page should appear empty.



The user enters Ex Label 1 into the search bar
WeatherLocationAPP
Search
Name
Label
Longitude
Lattitude
The Program only shows the searched location
The user enters Ex Label 1 into the search bar
The Program only shows the searched locations

Ex City 1	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 2	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 3	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 3	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 4	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 4	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 5	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 5	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

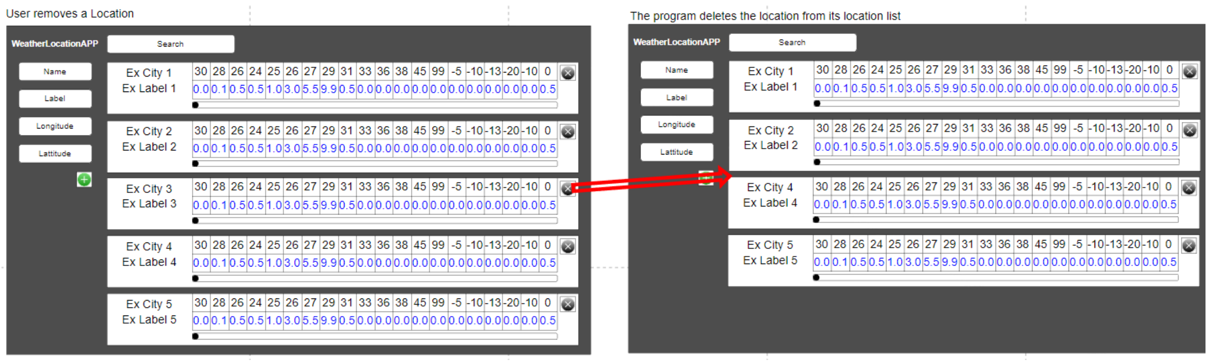
Ex City 1	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 2	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

Ex City 1	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 2	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 2	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 2	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 3	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 3	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 4	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 4	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 5	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 5	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

Figure 3 user searches the app

Use case 3

User deletes one of their previously saved locations. Every Location object has a delete button attached to it. This button specifically deletes that item immediately after being pressed. There is no additional query to ask for confirmation as the creation process is rather simple.



User removes a Location

Ex City 1	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 2	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 2	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 3	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 3	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 4	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 4	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 5	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 5	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

The program deletes the location from its location list

Ex City 1	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 2	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 2	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 4	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 4	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 5	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 5	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

Figure 4 User deletes a city

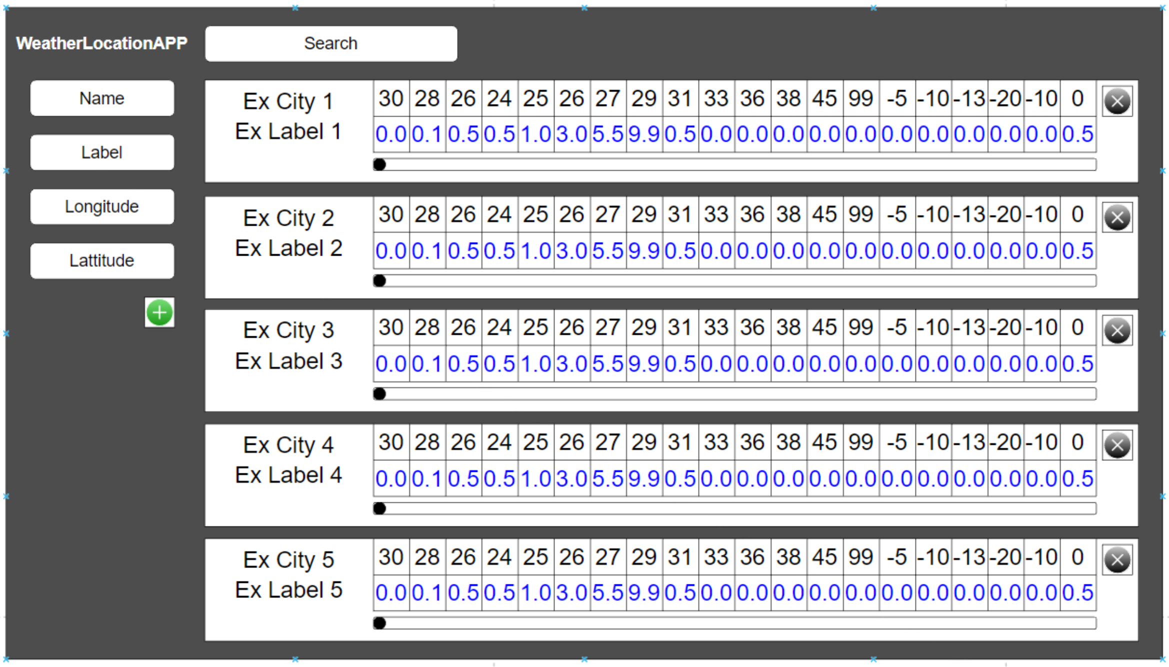
Screen-Mockup

The Mockup shows the completed Product as it has to be.

To the left are insert field that were Name,Label,Longitude,and Lattitude have to be entered This is one React component that then inserts an entry to the city list. All the fields are required. Latitude and longitude must be in the valid on earth existing range -90 to -90 and -180 to 180 respectively.

The city list is the React component next to the insert/ city creation component. This includes a search function that sifts trough all the cities inserted into the City list.

City is its own React component that sits inside the CityList. This element has all the attributes given by the Creation component as well as the weather data for that location delivered by the weatherstackAPI.



Ex City 1	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 1	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 2	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 2	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 3	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 3	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 4	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 4	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
Ex City 5	30	28	26	24	25	26	27	29	31	33	36	38	45	99	-5	-10	-13	-20	-10	0
Ex Label 5	0.0	0.1	0.5	0.5	1.0	3.0	5.5	9.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

Figure 5 Mockup

API-Interaction

This code is a React component called CityList that is designed to display weather data for a list of cities. It utilizes the weatherstackAPI to fetch weather information for each city and allows users to search for cities. Let's break down how it works:

```
 const fetchWeatherData = async (city) => {
      try {
        const response = await axios.get(

`http://api.weatherstack.com/current?access_key=f2c7de1ae140db82f7d9cc88985142
96&query=${city.latitude},${city.longitude}`
        );

        // Extract relevant weather data from the API response
        const temperature = response.data.current.temperature;
        const rainfall = response.data.current.precip;

        // Update weatherData state with the new data
        setWeatherData((prevWeatherData) => ({
          ...prevWeatherData,
          [city.location]: { temperature, rainfall },
        }));
      } catch (error) {
        console.error(`Error fetching weather data for ${city.location}:`,
error);
      }
    };
```

State Variables:

weatherData:	This state variable is used to store weather data for each city. It's an object where the keys are city locations, and the values are objects containing temperature and rainfall data.
searchTerm:	This state variable is used to store the user's search query for filtering cities.
filteredCities:	This state variable holds the list of cities to display after applying the search filter. Initially, it's set to the same list of cities passed as a prop.

useEffect for Fetching Weather Data:

The first useEffect is responsible for fetching weather data for each city. It runs whenever the cities prop changes (when the component mounts or when the list of cities is updated). Inside this effect, a function called fetchWeatherData is defined. It makes an asynchronous request to the Weatherstack API for the current weather data of a specific city using Axios.

The weather data, including temperature and rainfall, is extracted from the API response and then added to the weatherData state object using the setWeatherData function.

```
// Update weatherData state with the new data
        setWeatherData((prevWeatherData) => ({
          ...prevWeatherData,
          [city.location]: { temperature, rainfall },
        }));
```

If there is an error during the API request, it is caught, and an error message is logged to the console.

```
      } catch (error) {
        console.error(`Error fetching weather data for ${city.location}:`,
error);
```

useEffect for Filtering Cities:

```
  useEffect(() => {
    // Filter the cities based on the search term for location or label
    const filtered = cities.filter(
      (city) =>
        city.location.toLowerCase().includes(searchTerm.toLowerCase()) ||
        city.label.toLowerCase().includes(searchTerm.toLowerCase())
    );
    setFilteredCities(filtered);
  }, [cities, searchTerm]);

  const handleSearchChange = (event) => {
    setSearchTerm(event.target.value);
  };
```

The second useEffect is responsible for filtering the list of cities based on the searchTerm. It runs whenever the cities prop or searchTerm state variable changes.

It filters the list of cities based on whether the city's location or label contains the search term (case-insensitive), and the filtered list is stored in the filteredCities state variable using setFilteredCities.

In summary, this CityList component fetches weather data for a list of cities when it mounts and updates it when the list of cities changes. The weather data for each city is stored in the weatherData state variable, and the filtered city list is displayed based on the search term in the filteredCities state variable.

Test Plan

This Test plan is for white box testing the finished product, To ensure functionality and handling of edge cases.

Test 1

ID / Name	T-001	All ok
Description	In this Test the User enters correct data.	
Test requirements	The program must be installed Npm start has been successful The program is displayed in Firefox	
Test Steps	1. Enter the name "Thun" 2. Enter the tag "Work" 3. Enter coordinates of Thun copied from google maps 46.764051849039205, 7.624559011672853 4. Click submit	
Expected Result	The Program should add the Location to the list and lets you filter them by either.	

Test 2

ID / Name	T-002	Edge Case
Description	The test user enters valid coordinate values at the edge of the acceptable range	
Test requirements	The program must be installed Npm start has been successful The program is displayed in Firefox	
Test Steps	1. Enter the name "Edge" 2. Enter the tag "Case" 3. Enter in the coordinates 90,180 4. Click submit	
Expected Result	Weather data the app lets this entry trough with out alert	

Test 3

ID / Name	T-003	Out of bounds
Description	The test user enters invalid coordinates values	
Test requirements	The program must be installed Npm start has been successful The program is displayed in Firefox	
Test Steps	1. Enter the name "Edge" 2. Enter the tag "Case" 3. Enter in the coordinates 91,180 4. Click submit	
Expected Result	The program will send out an alert to the user giving information as to the proper formatting of the coordinates. As well as log an entry in the browser console	

Test 4

ID / Name	T-004	Filter Name
Description	The user utilizes the powerful Filter function to sort trough their locations	
Test requirements	The program must be installed Npm start has been successful The program is displayed in Firefox T- 001 Completed successfully	
Test Steps	1. The user enters letters into the search field 2. T 3. H 4. O	
Expected Result	Step 2 and 3 should still display the previously created location THUN but once step 4 is fulfilled the location must disappear from the list.	

Test 5

ID / Name	T-005	Filter Label
Description	The user utilizes the powerful Filter function to sort trough their locations	
Test requirements	The program must be installed Npm start has been successful The program is displayed in Firefox T- 001 Completed successfully	
Test Steps	1. The user enters letters into the search field 2. W 3. O 4. K	
Expected Result	Step 2 and 3 should still display the previously created location THUN but once step 4 is fulfilled the location must disappear from the list.	

Test 6

ID / Name	T-006	Delete
Description	The user Deletes an entry in the city list	
Test requirements	The Program must be installed Npm start has been successful The program is displayed in Firefox T- 001 Completed successfully	
Test Steps	1. Click delete on the Thun Location	
Expected Result	The program removes Thun from the list of displayed places with out any error messages.	

Tested System Requirements

These settings were the ones the system was tested on, as such

- Windows 11 Home
- Firefox 117.0.1

Installation instructions

1. Download the WeatherApp zip file
2. Unpack the zip file into the folder you prefer
3. Switch to the src folder
4. Open the windows Terminal in that folder
5. type “npm install” to make sure all the relevant files are up to date
6. wait until the installation is complete
7. npm install axios
8. wait until installation is complete
9. Then type “npm run”
10. Now the website should be open in your preferred browser

Figure Index

Table Index

References

W3School – Used for code references and to iron out details.

ChatGPT 3.5 – General code provider and help to structure certain paragraphs in the documentation.

Hugo Lucca – Teacher of the class.