# COMP 215 – Final Project: Create a weather logging Application DUE: APRIL 29 2019

## Project Report

Create a JavaScript application using HTML, CSS, JQuery and AJAX techniques to implement a client-side weather logger. Begin with your Lab 12 (or the demo code) and extend the application following the requirements detailed in the next sections.

No formal report is required for this project. When you have completed and tested your application briefly demonstrate your running application to the instructor and double zip your project folder and email the archive to: [**brett.shiers@saskpolytech.ca**](mailto:brett.shiers@saskpolytech.ca) for marking

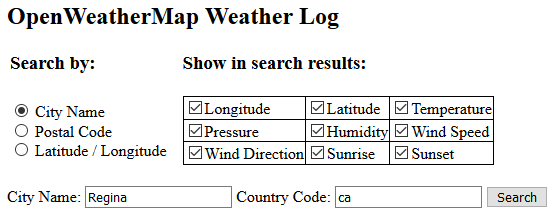
**Marking Scheme**:

1. Demonstration 10%
2. Code format, clarity and comments 20%
3. Input form functionality, validation and styles 20%
4. Weather Log “Add to Log” functionality 30%
5. Weather Log “Delete from Log” functionality 20%

100%

## Requirments: Create A Client-side Weather Logging application

1. Create your application in a web page named **weatherlog.htm**. Place your CSS rules in **weather.css** and your JavaScript in **weather.js**
2. Your application will consist of a **“Weather Search” input form** for the user to select a city or location, and a **“Weather Log” table** listing all the weather data from the user’s search results.
3. **Weather Search Form**:
   1. Begin with the form from Lab 12, and add inputs similar to the sample shown:

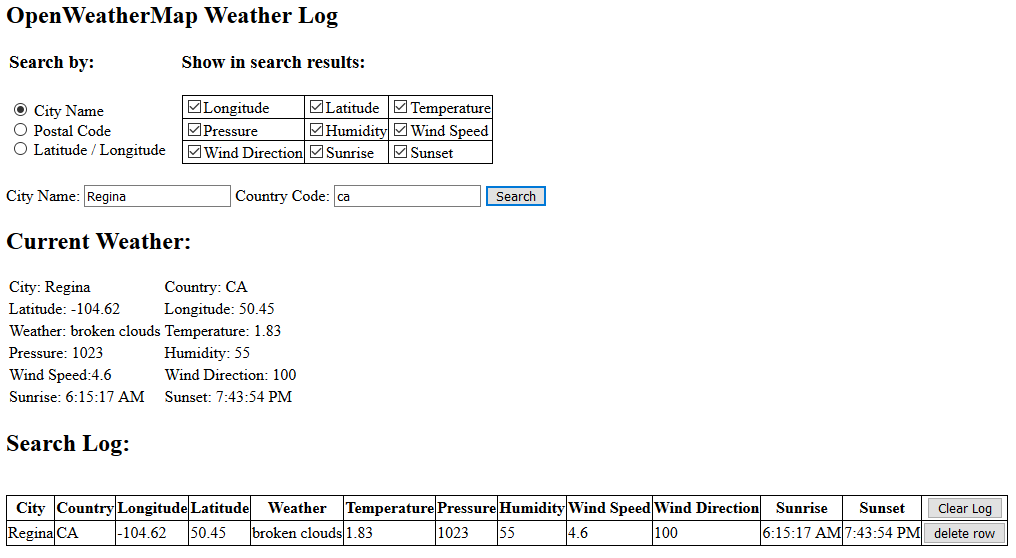


* 1. The form sample is an example only - You may layout your form as you like, but use **Bootstrap Form classes and styles** for consistent formatting.
  2. As done in Lab 12, when the user selects each search option, the appropriate input boxes are displayed:
* For the “City Name” option, show City Name and Country Code input boxes
* For the “Postal Code” option show Postal Code and Country Code input boxes
* For the “Latitude / Longitude” option show Latitude and Longitude input boxes
  1. Use **CSS validation formatting rules** to show when data is not valid: For example: change the background or border color of the input textbox of invalid fields to a red color when not valid, and a green color when valid.

The validation rules are:

* All input textboxes are Required
* Country Code: Must be 2 characters: Pattern=”[a-zA-Z]{2}”
* Postal Code: Must be a Letter, Number, Letter: Pattern=”[A-Z][0-9] [A-Z]”
* Latitude: Must be a number between -90 to + 90
* Longitude: Must be a number between -180 to + 180
  1. Use **JQuery to attach an oninvalid() event handler** function to each of the fields with validation rules.   
     Your oninvalid() functions should **use the setCustomValidity() method to set appropriate error messages**. For example, for the Country Code field you may prompt the user with “You must enter a 2 uppercase country code”.

1. **Display Weather Search Results**: Create Outputs on your html page similar to the sample below. Again, this is only a schematic sample – use bootstrap to format your results. The output requirements are described in sections (5) – (8) below.



The outputs shown here depend on the checkbox options selected

The inputs shown here depend on the radio option selected

1. **Current Weather**: Below your search form, display the results from the weather search. Show the City, Country Code, Weather description, and only the other weather details that are selected in the check boxes
2. **Search Log Table**: Below your Current Weather, create a table to log the results of each search that is made. Unlike section (5) show and log all the search result details independent of the checkbox options selected.
   1. When the page is loaded or refreshed, load any previously saved items from Local Storage and display them in the table.
   2. Each search that is made should be displayed in the Search Log table and also saved in the browser’s Local Storage
3. **Search Button**: When clicked, your Search button should:
   1. Verify that the form data is valid ( use the form’s checkValidity() method )
   2. If the form data is valid:

* Compose a proper OpenWeatherMap search string and use AJAX to pass the search request to OpenWeatherMap.org.

For example, the search sample shown used the search query:

"http://api.openweathermap.org/data/2.5/weather?q=REGINA,ca&APPID=52453f34dee0d65b1a41a02656142c6b"

* Retrieve the weather search results and convert the result into a JavaScript object.
* Display the properties of your Weather object in the Current Weather section of your webpage.
* The temperature is in Kelvin, subtract 273.15 to get degrees Celsius
* The Sunrise and Sunset times are retrieved in UNIX Epoch UTC format. To get current local time, use a JavaScript Date object expression similar to:

var sunrise = new Date(weatherObj["sys"]["sunrise"] \* 1000);

and output the time using:

sunrise.toLocaleTimeString()

* As shown in Lab11, use the system clock $.now() as a Weather Log Item ID. This ID will be used when deleting a row from the table.
* Compose a new row for the Search Log table containing the new weather log item details and append this new row to the end of the log table.
* Save the new weather log item in Local Storage. Use the array technique from Lab 11 to load existing saved weather log items into an array, push your new log item to the end of the array, and resave the array in local storage.

1. **Clear Log and Delete Row buttons**: create a new “Clear Log” column on the shopping cart table:
   1. Place a clickable “Clear Log” element (a link, checkbox or button) to allow the user to empty (delete) all the items from the Search Log:

* When clicked, clear the local storage and the table of all items.
  1. Place a clickable “delete row” element (a link, checkbox or button) in each row of the log table to allow the user to delete the single log item in that row.
* When clicked, remove the item in that row from the table AND remove the item from Local Storage.

1. **Code Comments and Documentation**:
   1. Choose appropriate names for all variables, elements, functions and CSS classes etc. that clearly indicate the meaning and purpose of that object.
   2. For each function and object you declare, include brief code comments to describe the purpose and arguments for that function.
   3. You may work and problem-solve in groups. However - all coding, object naming, comments and features of your solution must be your own work.
2. **Code Samples**: The lab demonstration code from labs 9, 10, 11 and 12 may be of use.

(end of COMP215 Final Project requirements)