**ICT375 Assignment 2 report**

**Name: Ow Jing Wei**

**Student No.: 34053405**

# Introduction

It is a weather aggregator website where user selects the type of data (windspeed and/or solar radiation) and result they want to display based on the year selected. User options will be sent and communicate with the server, and this is implemented using Node.js. The web browser serves as the client and HTTP protocol of Node.js core module will be used for all communication between client and server.

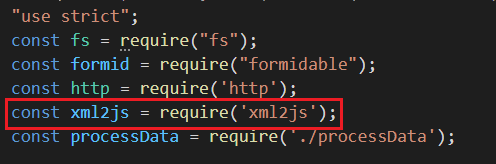
There is only 1 page for this website and user is required to select options of data they want to display, the year, range of months and how the data will be displayed.

# Full description of XML and JSON

## XML

For parsing of XML data, I have decided to use xml2js node npm module to parse XML data to JavaScript object after the retrieval of the data from URL provided. Below shows some of the screenshots of code on how this technology is implemented in my solution.

**The xml2js npm module was installed and imported to the project:**



**On line 83, the parser was instantiated: Text

Description automatically generated**

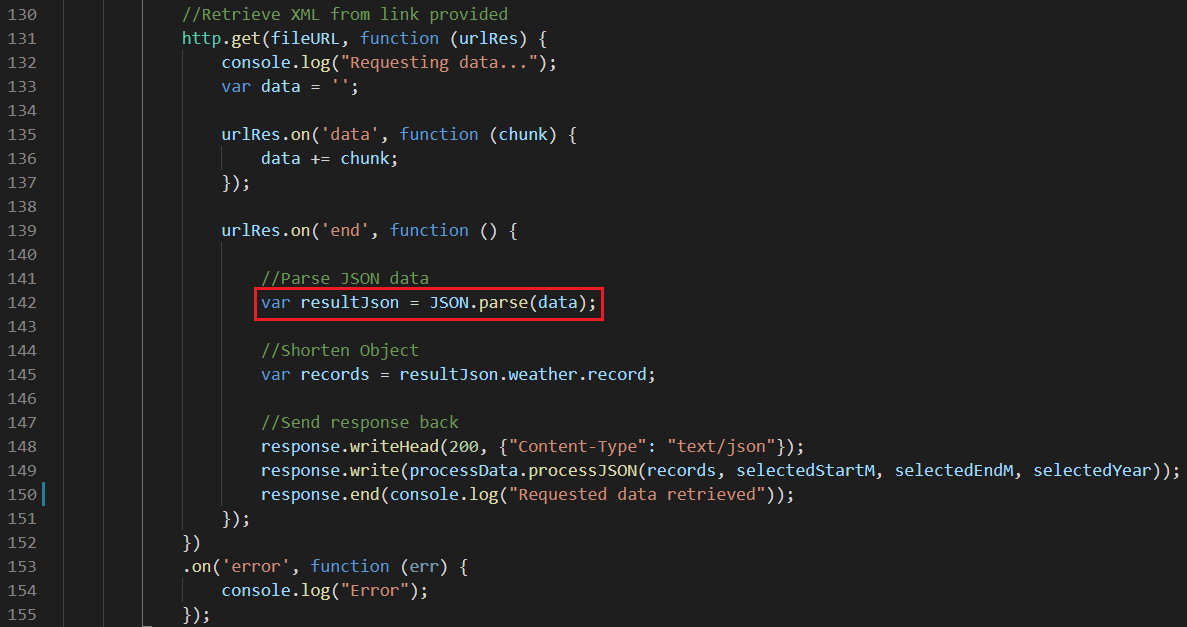
**On line 91, the parser was used to parse the XML data to JavaScript object with error handling: Text

Description automatically generated**

## JSON

For parsing of JSON data, no external node npm was used. JSON.parse() method was implemented to parse the string from JSON data retrieved from the URL provided. Below shows some of the screenshots of code on how this technology is implemented in my solution.

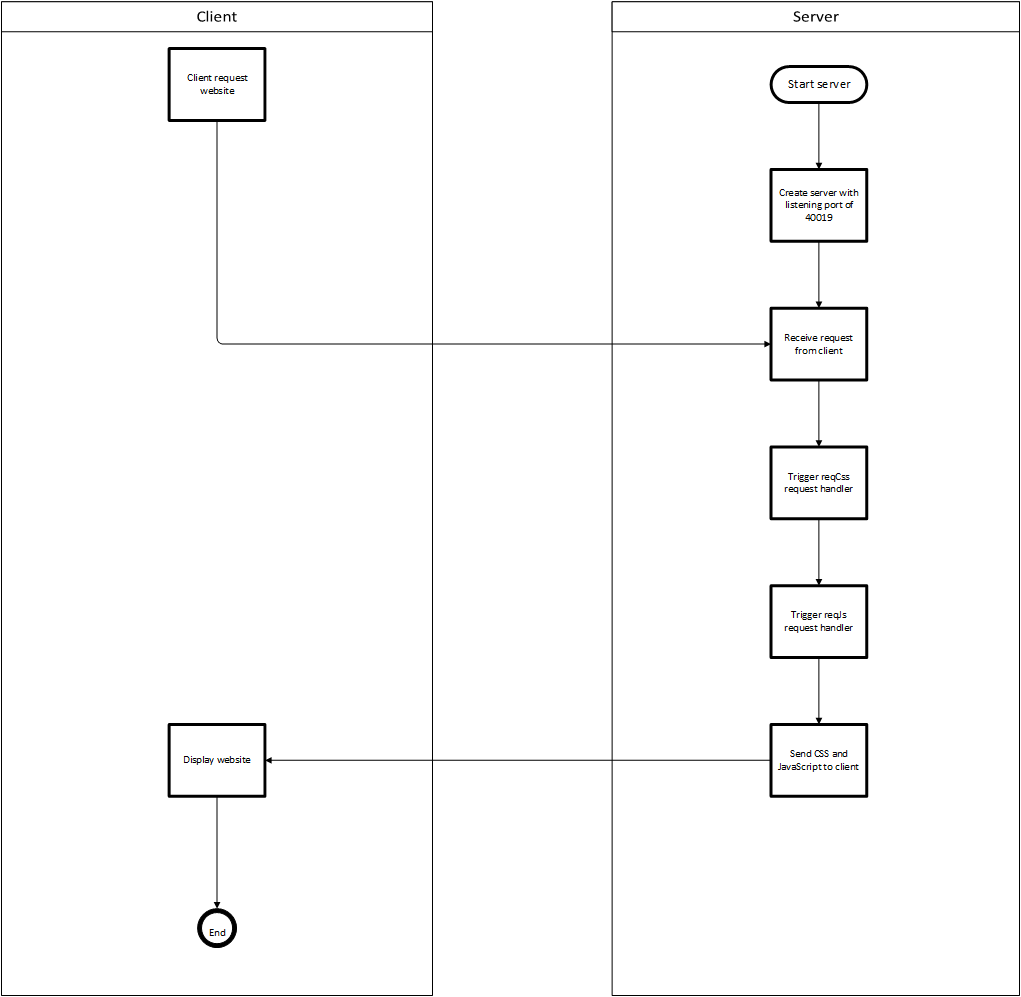
**On line 142, the JSON.parse() method was used to parse the JSON data**



# Overall design

## Requesting of website

When requesting website, user will enter the URL of the website, such as ceto.murdoch.edu.au:40019, and the browser will send request to 2 of the request handlers “/reqCss” and “/reqJs” with the path of “/css/styles.css” and “/js/client.js ” on the server for the CSS and JavaScript of the website, and the server will send it back to the client to render the design and functionality of the website.



## Requesting XML

**Sending GET request (Client-side)**

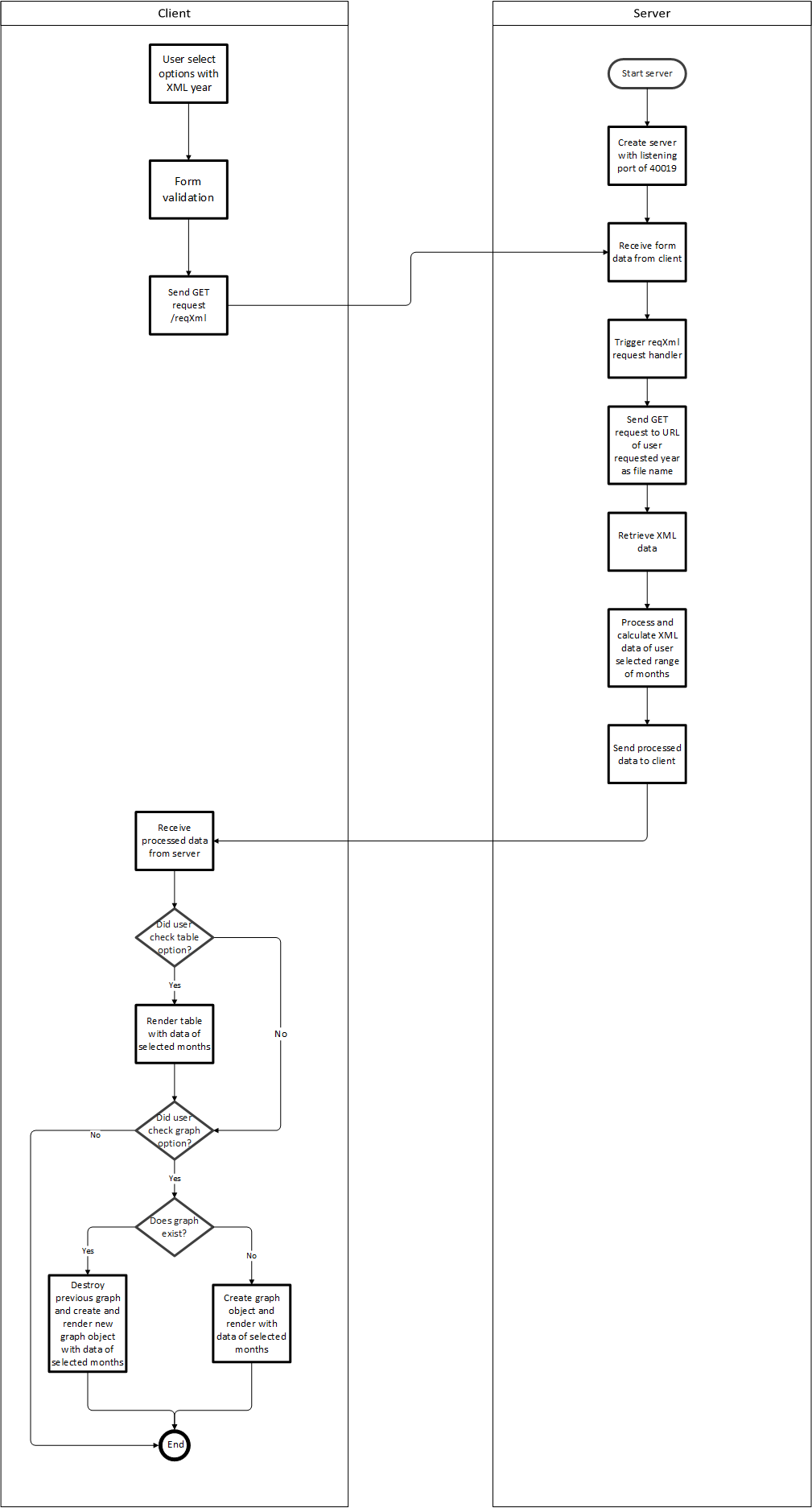
After the form is checked and selected with the correct information of the user’s choice, the form will be sent using XMLHttpRequest to the server requesting “/reqXml” request handler.

**Receiving the data from client (Server-side)**

After receiving the data from the client, the server will route to trigger “/reqXml” request handler and it will run the process.

**“/reqXML” request handler**

The reqXML request handler will first parse the form data using an external Node npm module, formidable, in order to get the form data of user selected data, year, months. The request handler will then send a http GET request using the node HTTP core module to retrieve the specific year user selected and when the data is fully retrieved, another external Node npm module will be used, xml2js, it is used to parse the XML data to an array of JavaScript objects in order to process the data easily and conveniently. If the parser failed to parse the XML data, it will throw an error to indicate failure to parse XML data, else the data will be used to process the selected range of months the client requested. This XML data will then be pass into the function that I have created in another JavaScript file “processData.js”. Full description of “processData.js” will be available under the section of “Description of data structures” below. After the processing of data, it will be sent back to the client script to process user selected output format.

****

## Requesting JSON

**Sending GET request (Client-side)**

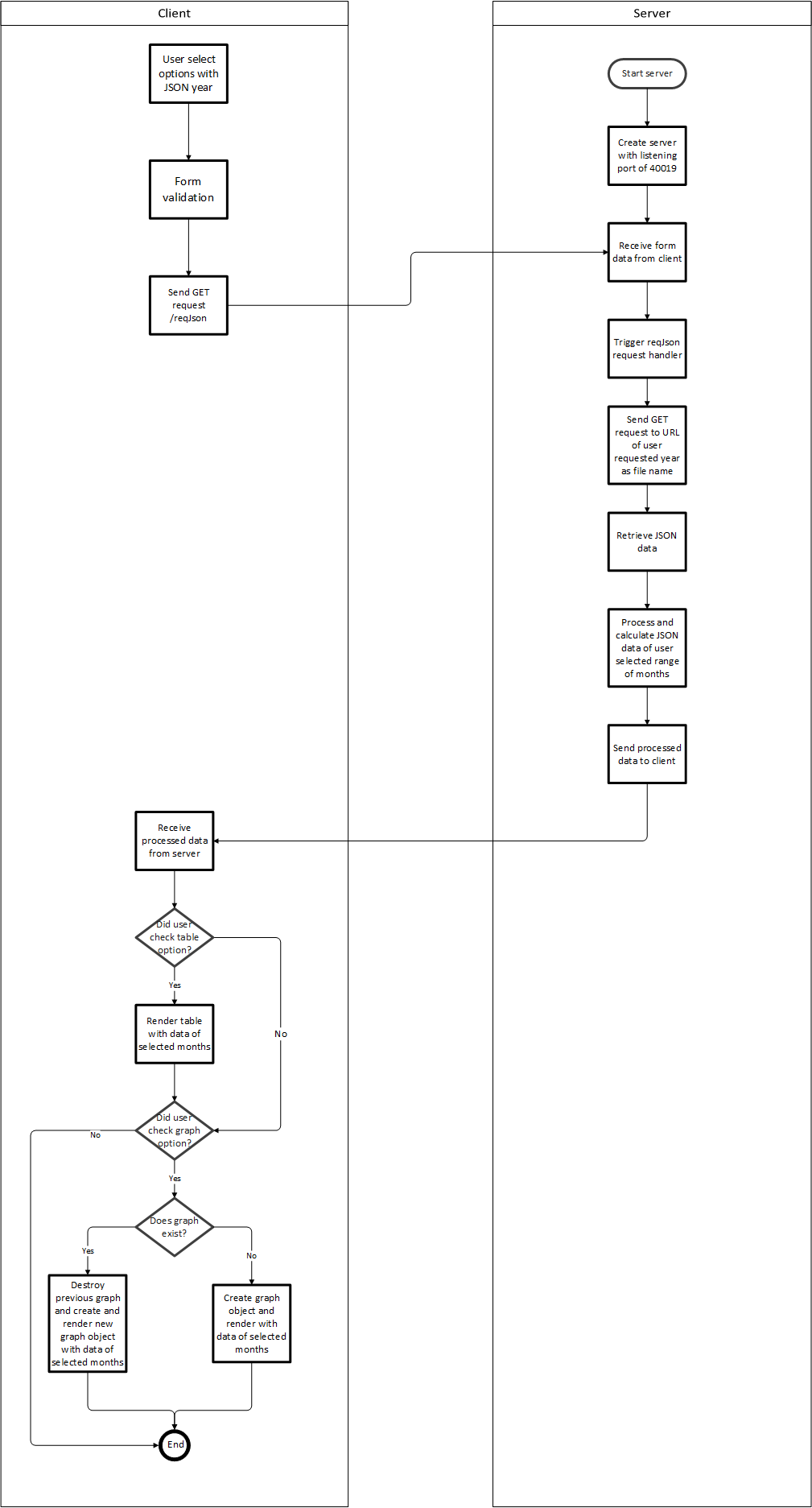
After the form is checked and selected with the correct information of the user’s choice, the form will be sent using XMLHttpRequest to the server requesting “/reqJson” request handler.

**Receiving the data from client (Server-side)**

After receiving the data from the client, the server will route to trigger “/reqJson” request handler and it will run the process.

**“/reqJSON” request handler**

The reqJSON request handler will first parse the form data using an external Node npm module, formidable, in order to get the form data of user selected data, year, months. The request handler will then send a http GET request using the node HTTP module to retrieve the specific year user selected and when the data is fully retrieved, no external npm node module will be used to parse the JSON data. For parsing JSON data, I have used the function “JSON.parse()” to parse JSON data to an array of JavaScript objects in order to process the data easily and conveniently. This JSON data will then be pass into the function “processJSON” that I have created in another JavaScript file “processData.js”. Full description of “processData.js” will be available under the section of “Description of data structures” below. After the processing of data, it will be sent back to the client script to process user selected output format.

****

## Request Error

**Sending request (Client-side)**

When a non-existing path is requested from the client, for example, “ceto.murdoch.edu.au:40019/windspeed”, request will be sent to “/reqError” request handler to handle error request.

**Receiving the data from client (Server-side)**

After receiving the data from the client, the server will route to trigger “/reqError” request handler and it will run the process.

**“/reqError” request handler**

When a non-existing path is requested from the client the reqError request handler will be called and send a “404 error not found” message back to client to indicate no such path.

# Description of data structures

**processData.js**

This JavaScript file is for processing both XML and JSON data in order to make it modular and makes it easier to read. This script has function of processXML and processJSON.

Function processXML will take in the parameters of XML data retrieved from the URL given, user requested startMonth and endMonth. The file name requested will dynamically change depending on the user’s selection. For example, when user selects year 2007, the program will then send GET request to: <http://it.murdoch.edu.au/~S900432D/ict375/data/2007.xml>.

Function processJSON will take in the parameters of JSON data retrieved from the URL given, user requested startMonth and endMonth. The file name requested will dynamically change depending on the user’s selection. For example, when user selects year 2010, the program will then send GET request to: <http://it.murdoch.edu.au/~S900432D/ict375/data/2010.json>.

Both of the code might look the same but selecting the data of XML and JSON is different. For example, for XML getting the date from the data is “value.date[0]” while getting the data for JSON is “value.date”, and this is the reason of having 2 different function of processing data.

This JavaScript file will first process the requested range of months the client requested. For example, user selects, “January” and “May”, this function first will process the range of months from January to May, for example, Jan, Feb, Mar, Apr, May.

After processing the months, function .map() is used to process the data to get all the data of all months but with different timing and data, for example**, {date: ‘01/01/2010’, time: ’09:00’, ws: 7, sr: 813}, { date: ‘01/01/2010’, time: ’09:10’, ws: 6, sr: 845}, …** . For every month, I have converted the date to month for example when the program finds the string of “/01/” and this will be converted to “Jan” and making sure the WS and SR data was converted to Number for calculation.

After getting all the data and conversion, using “result2”, I have used function .map() again in order to remove the object property value of array to just value and converting property name “date” to “Month” and make sure the data is still Number. For example, from **{date: [‘Jan’], ws: [4], sr:[12]}** to **{Month:** **‘Jan’, Ws: 6, Sr: 447}.**

Next, using “result3”, I created another array called “avgArray” just to store all the calculations. I used the .reduce() function to do some calculations, for adding the previous data read to the current data. If that month is had not data yet, all the data will be 0 and push it to “avgArray”, else it will push the count, sum and average of Wind Speed, sum and total of Solar Radiation. I have done the conversion from m/s to km/h and w/m^2 to kWh/m^2 in this function. The end result of this function will be:

**{Month: ‘Jan’, Count: 4411, WsAvg: 2.425119020063024, WsSum: 27477, SrSum: 1482299, SrTotal: 1482.299},**

**{Month: ‘Feb’, Count: 3573, WsAvg: 23.588916876574306, WsSum: 23412, SrSum: 1082013, SrTotal: 1082.013}, …**

The last function is to find the specific month that is not appended into the array as some XML or JSON does not have data for the specific month. So .map() and .find() function is used, when the function did not find that specific month comparing it to “monthNumber” array it will append all the that data for that specific month with 0. For example, in 2007.xml, the months of March and April have no data, hence, the program will append **{Month: ‘Mar’, Count: 0, WsAvg:0, WsSum:0, SrSum:0, SrTotal: 0}, {Month: ‘April’, Count: 0, WsAvg:0, WsSum:0, SrSum:0, SrTotal: 0}, …**

Before returning the data back to the 2 request handlers reqXml or reqJson, both of XML and JSON data will be converted using JSON.stringify(), so it can be sent back to the client-side as string. Both of the processing of XML and JSON data is the same but the only difference between function “processJSON” and “processXML” is that the selecting of data is different as mentioned.

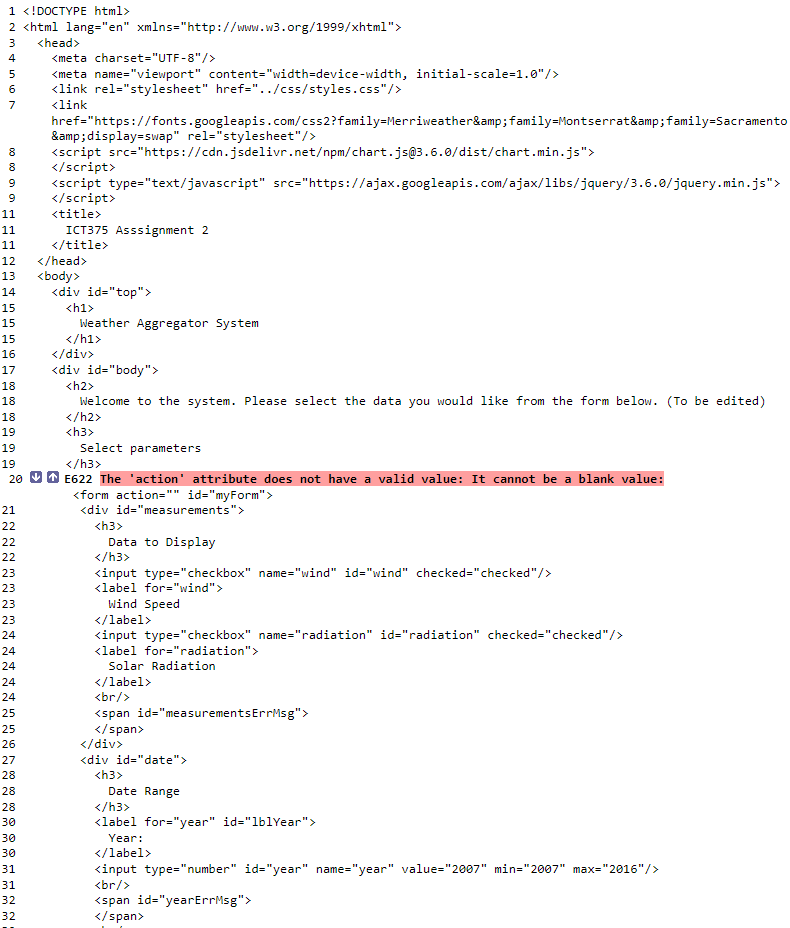
# Application test (Screenshots)

## HTML TotalValidator

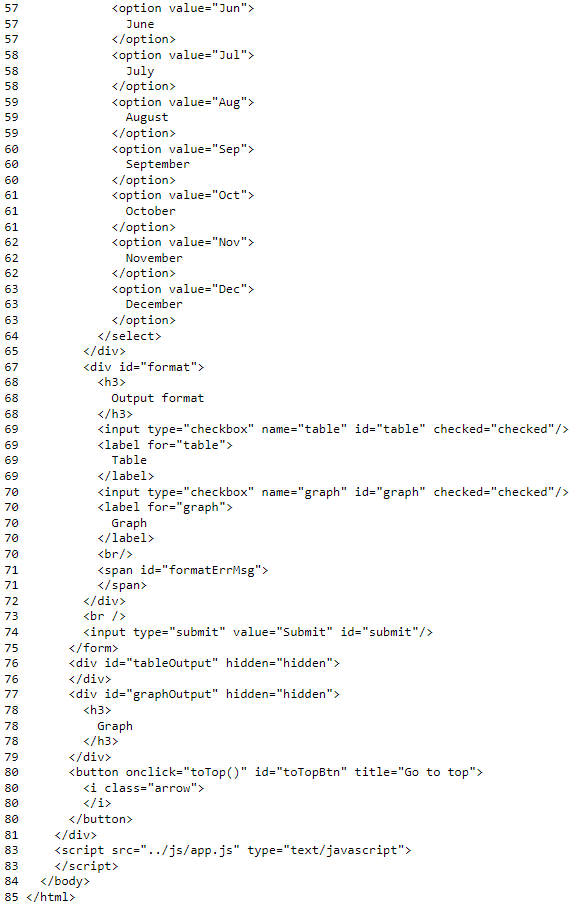
The result of HTML TotalValidator, there is an error found, but it is for the functionality of the website that causes this error: Graphical user interface, application, Teams

Description automatically generated

The only error of the HTML was the action was blank as the form needs to be validate before submitting it to the server, I have leave the action attribute blank to prevent it to submit to server first before validation and changed the action of the form in the JavaScript after validation is completed, just to make sure user entered the required fields in the form so that it is valid to submit to server:



A picture containing graphical user interface

Description automatically generated

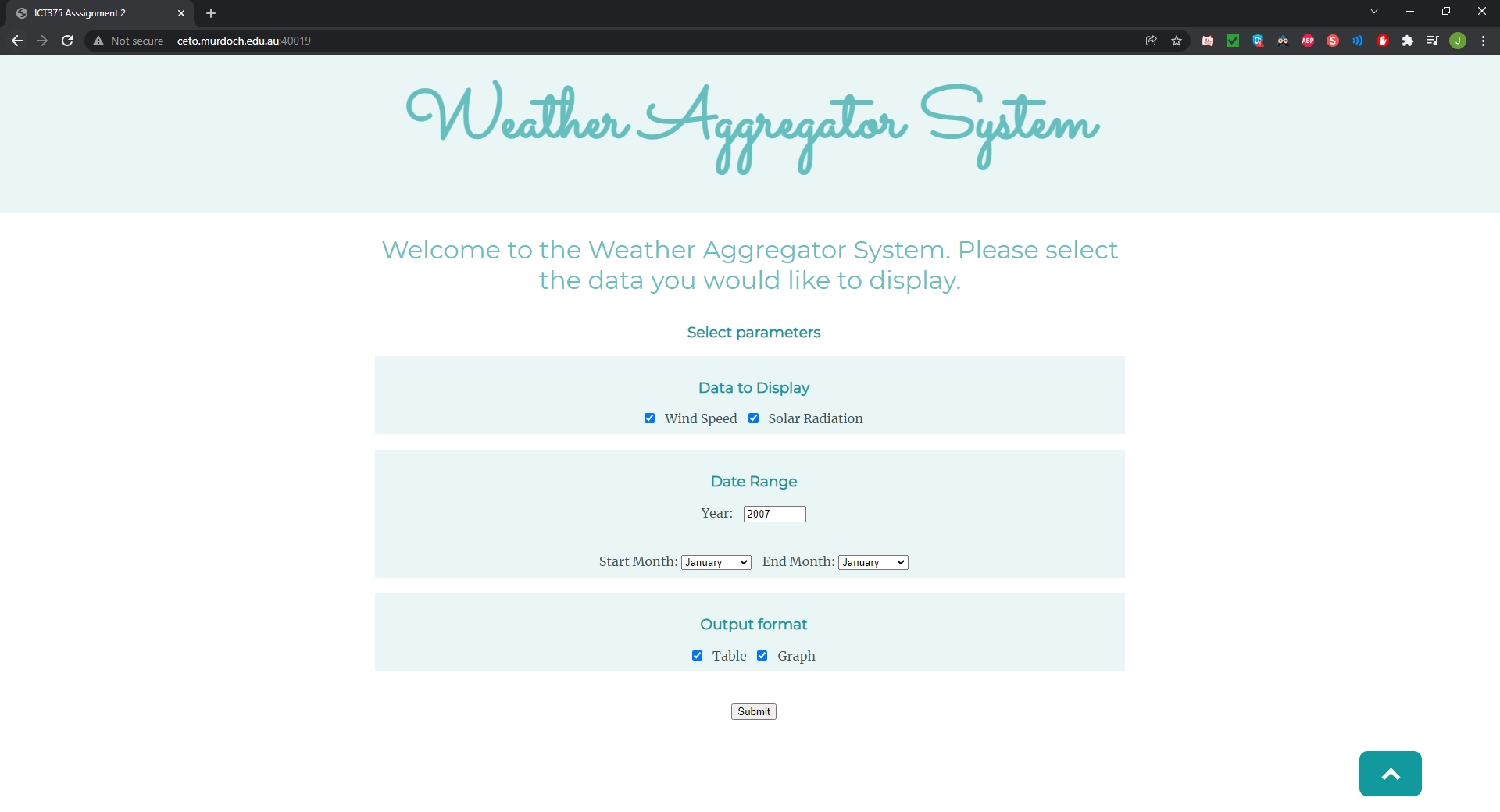
## CSS validation

Graphical user interface, text, application, email

Description automatically generated

## Form validation

### Initial page



### Data checkbox validation

**The initial Wind Speed and Solar Radiation checkbox when it is both checked: A picture containing graphical user interface

Description automatically generated**

**When Wind Speed is unchecked: A picture containing timeline

Description automatically generated**

**When Solar Radiation is unchecked: A picture containing graphical user interface

Description automatically generated**

**When both checkbox is unchecked: A picture containing graphical user interface

Description automatically generated**

### Year input validation

**The initial Table and Graph checkbox when it is both checked: A picture containing graphical user interface

Description automatically generated**

**When user remove the text and leave it blank:A picture containing text

Description automatically generated**

**When user enters an invalid year: A picture containing text

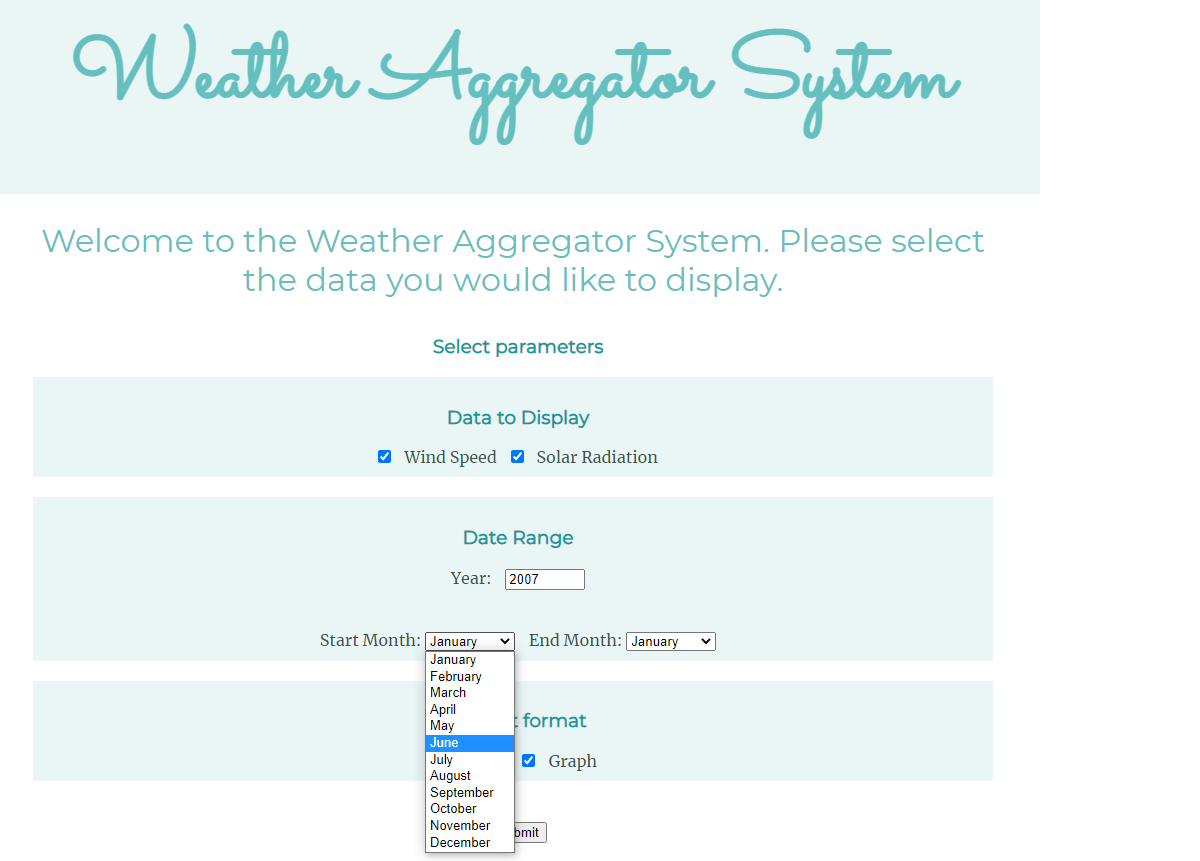
Description automatically generated**

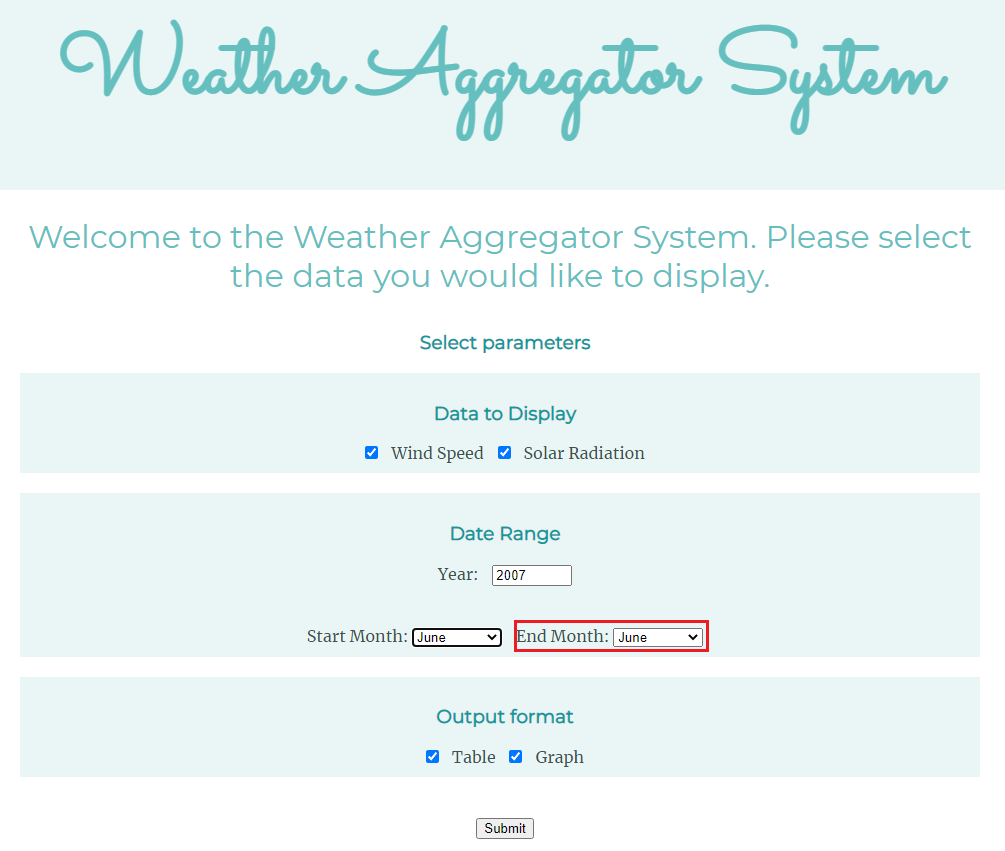
### Month dropdown validation

**Initial start month and end monthA picture containing graphical user interface

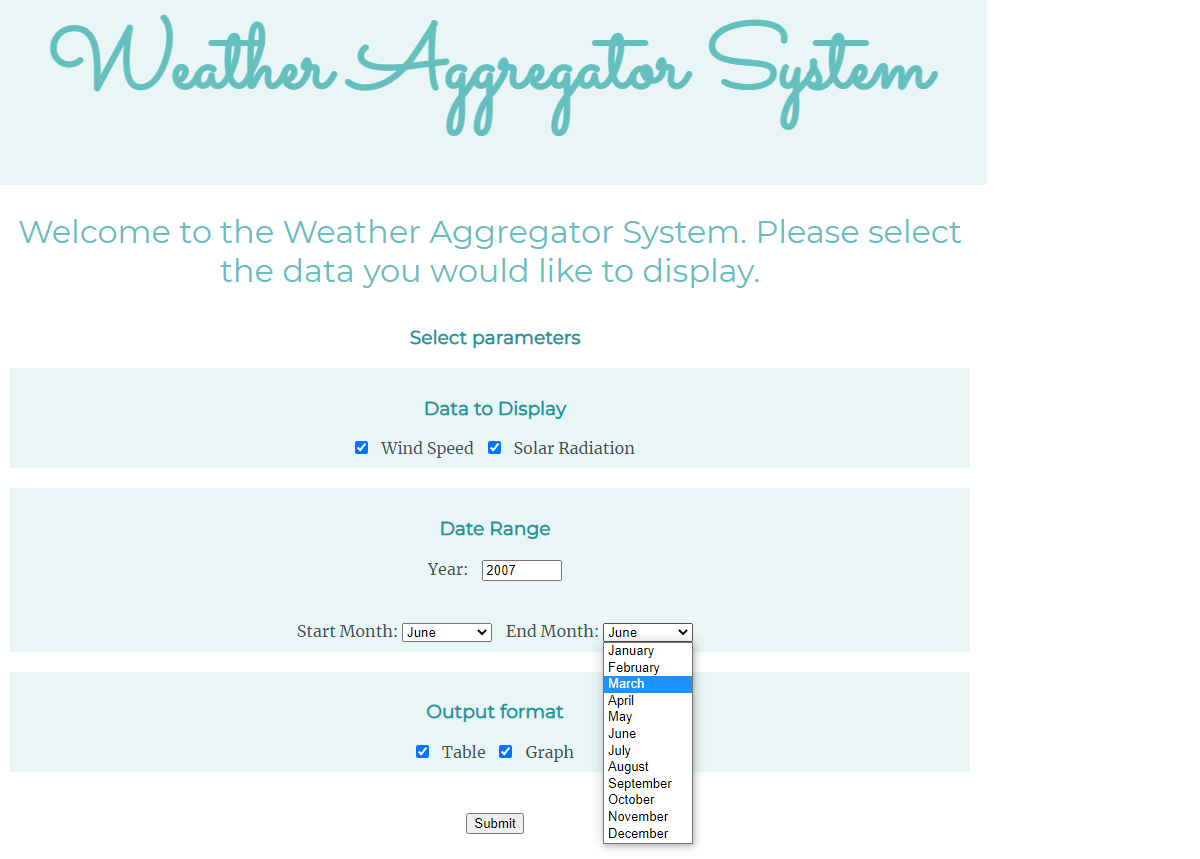
Description automatically generated**

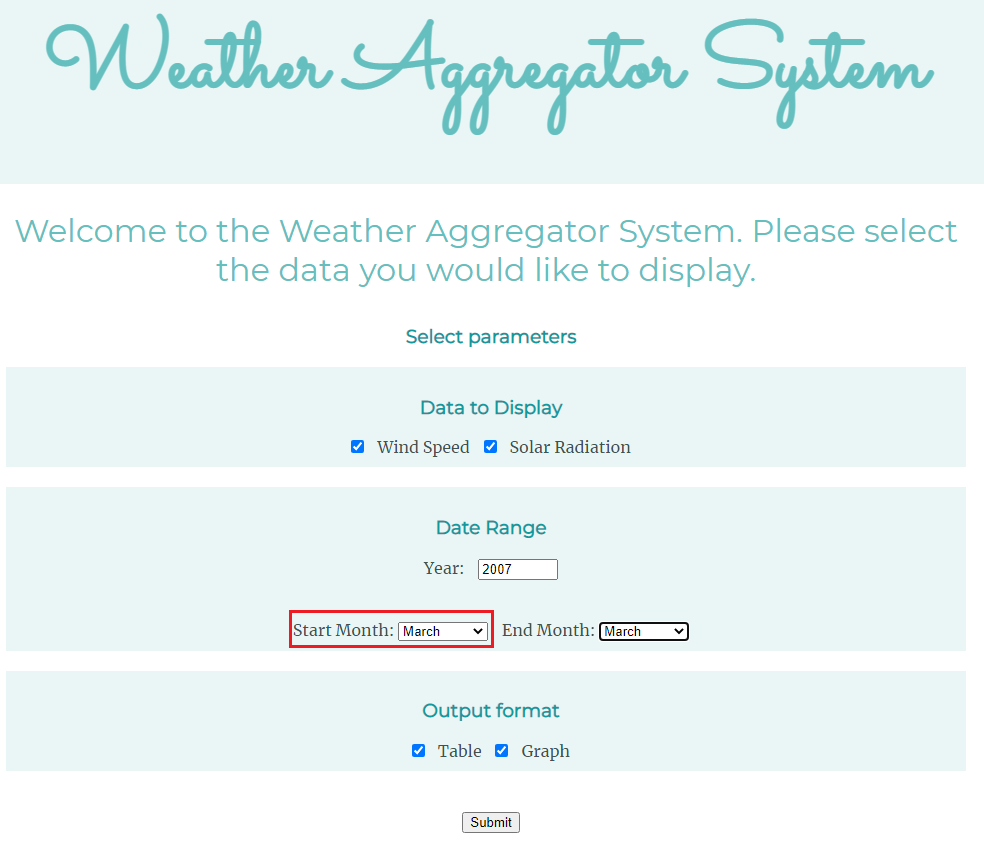
**When End Month is earlier than Start Month:**

****

****

**When End Month is earlier than Start Month:**

****

****

### Output format checkbox validation

**The initial Table and Graph checkbox when it is both checked:**

**A picture containing graphical user interface

Description automatically generated**

**When Table checkbox is unchecked:**

**A picture containing graphical user interface

Description automatically generated**

**When Graph checkbox is unchecked:A picture containing timeline

Description automatically generated**

**When both checkbox is unchecked: A picture containing timeline

Description automatically generated**

## Root request: “/”

When the root is requested, the server will respond with the website to the user with proper JavaScript and CSS.

**Client-side response when root is requested: Text

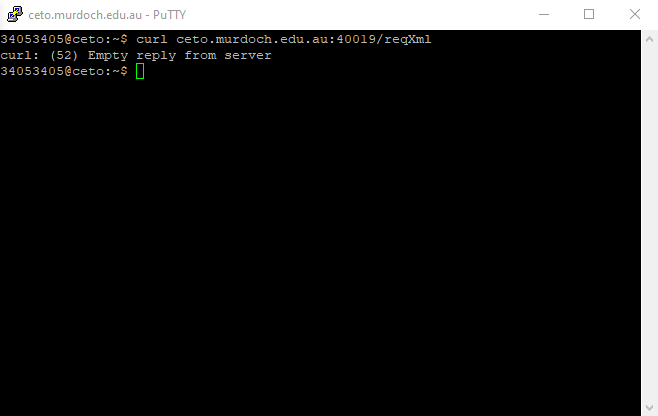
Description automatically generated**

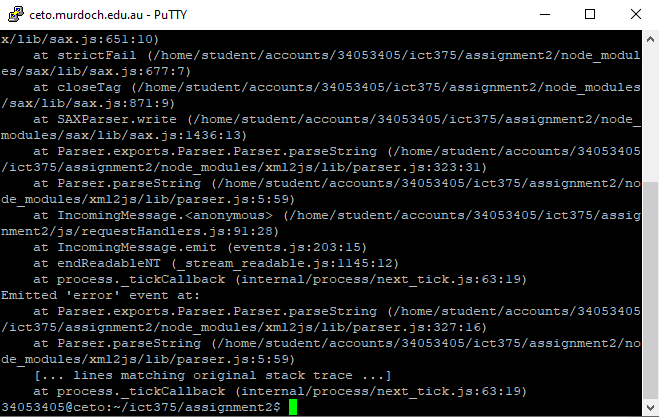
**Server-side when root is requested: Text

Description automatically generated**

## XML request: “/reqXml”

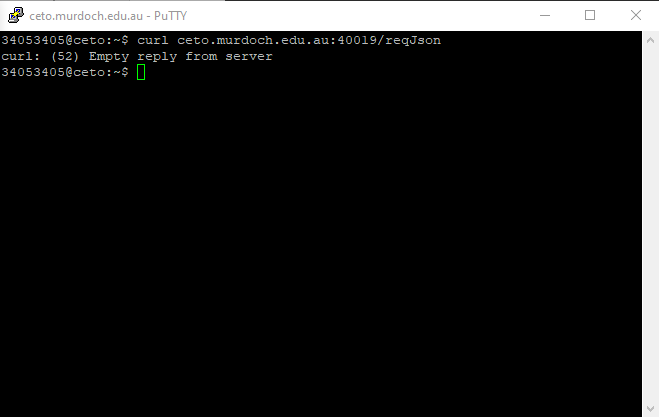
An error occurred when the client directly request “/reqXml” because there is no Year specified for the program to retrieve any data from the URL provided of this assignment.

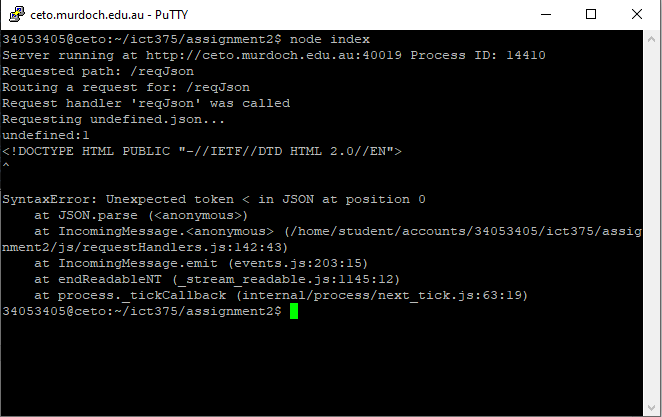
**Client-side response when “/reqXml” is requested:**

**Server-side when “/reqXml” is requested:**

## JSON request: “/reqJson”

An error occurred when the client directly request “/reqJson” because there is no Year specified for the program to retrieve any data from the URL provided of this assignment.

**Client-side response when “/reqJson” is requested:**

**Server-side when “/reqJson” is requested:**

## JavaScript request: “/js/app.js”

Requesting the JavaScript of website when user access the website for the functionality of client-side of the website.

**Client-side response when JavaScript is requested: Text

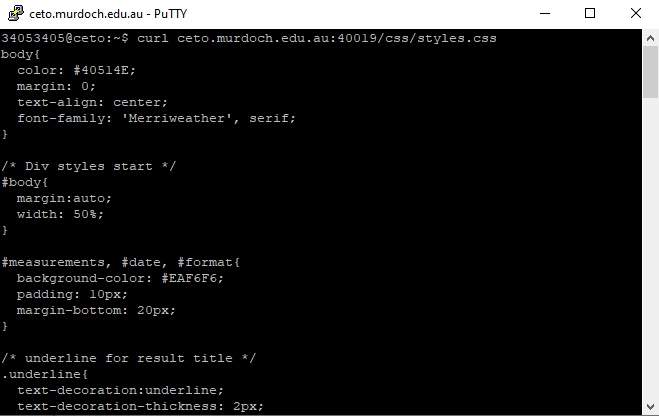
Description automatically generated**

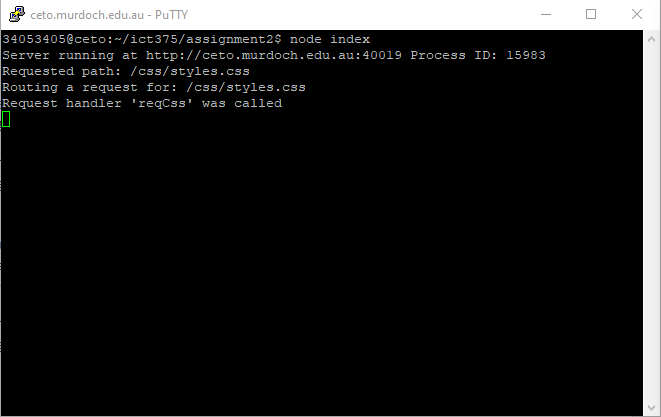
**Server-side when JavaScript is requested: Text

Description automatically generated**

## CSS request: “/css/styles.css”

Requesting the CSS of website when user access the website to render the design of website.

**Client-side response when CSS is requested:**

**Server-side when CSS is requested:**

## Error request: “/error”

Error request handler is when users entered an invalid URL of the website for example, ceto.mudoch.edu.au:40019/windspeed

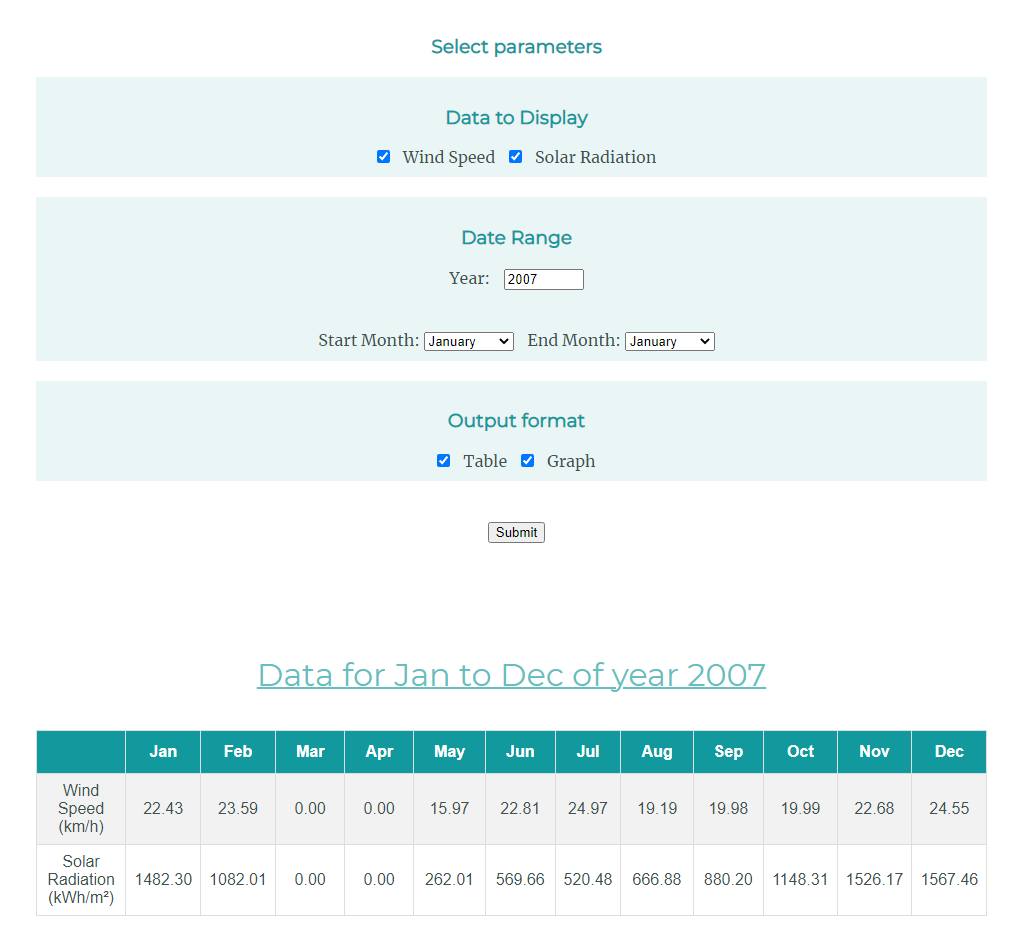
**Client-side response when error is requested: Text

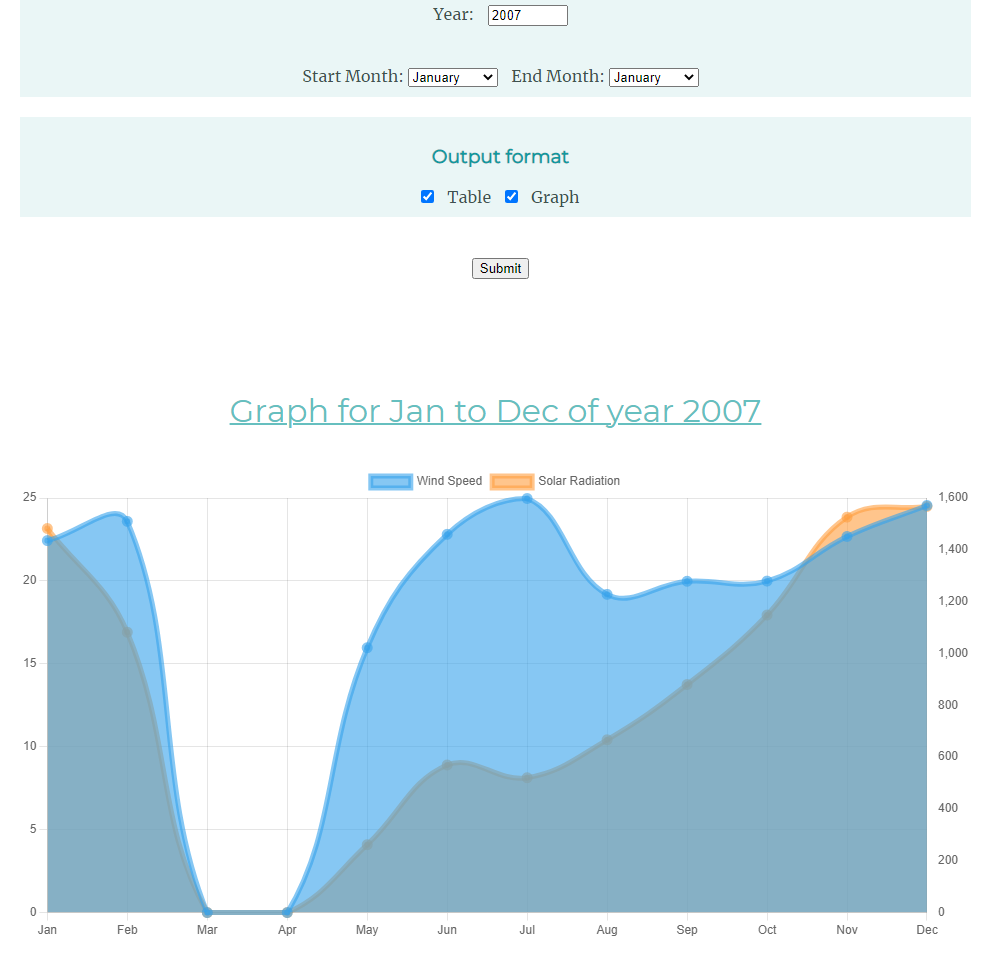
Description automatically generated**

**Server-side when error is requested: Text

Description automatically generated**

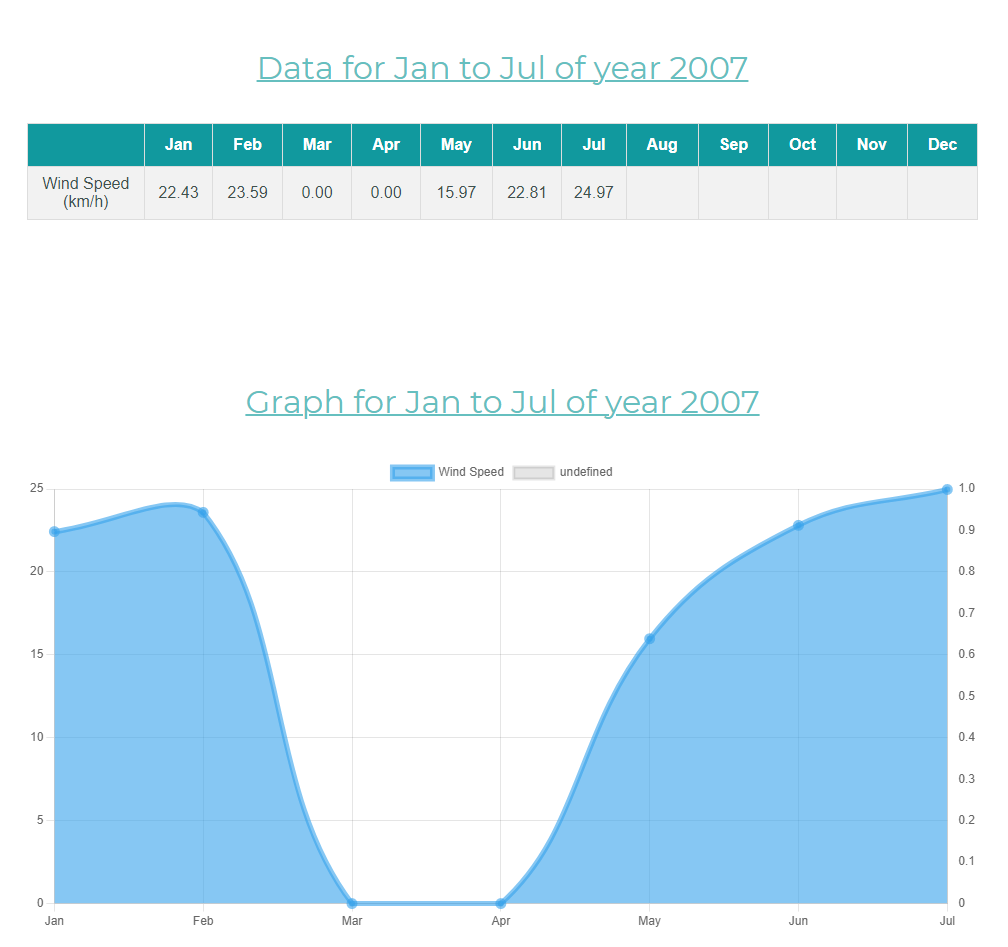
## Result of XML data

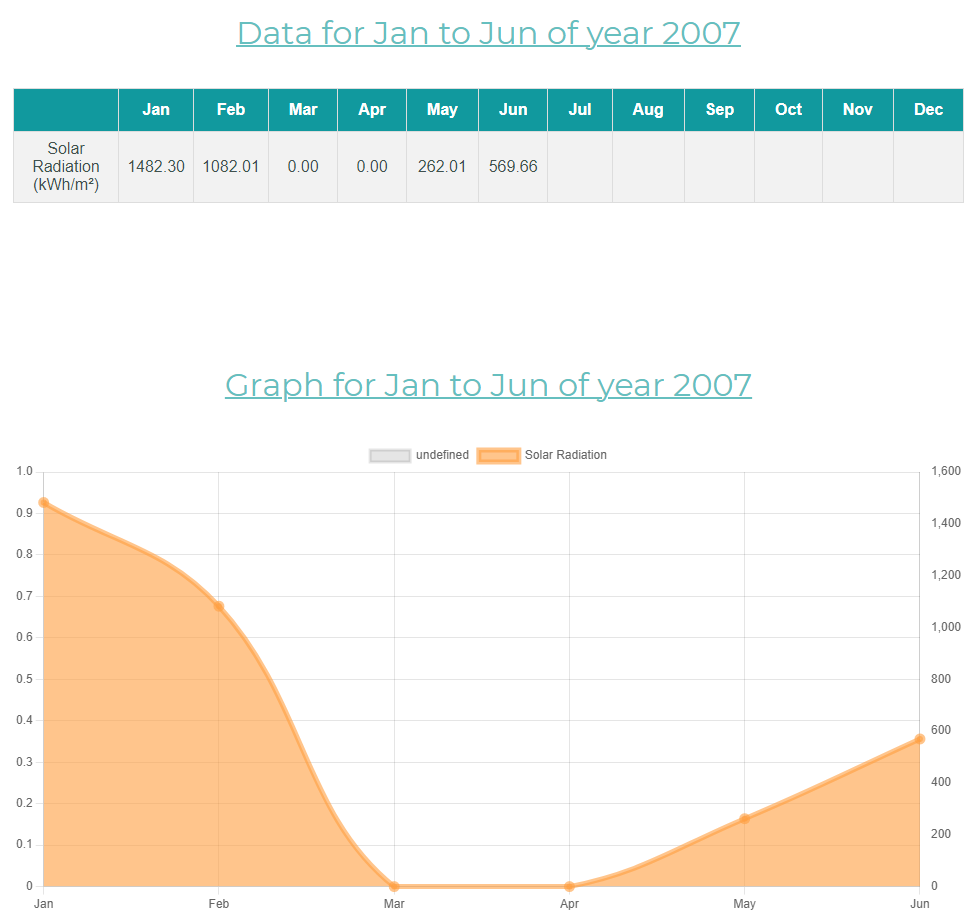
**XML result with only table data**

**XML result with only graph data**

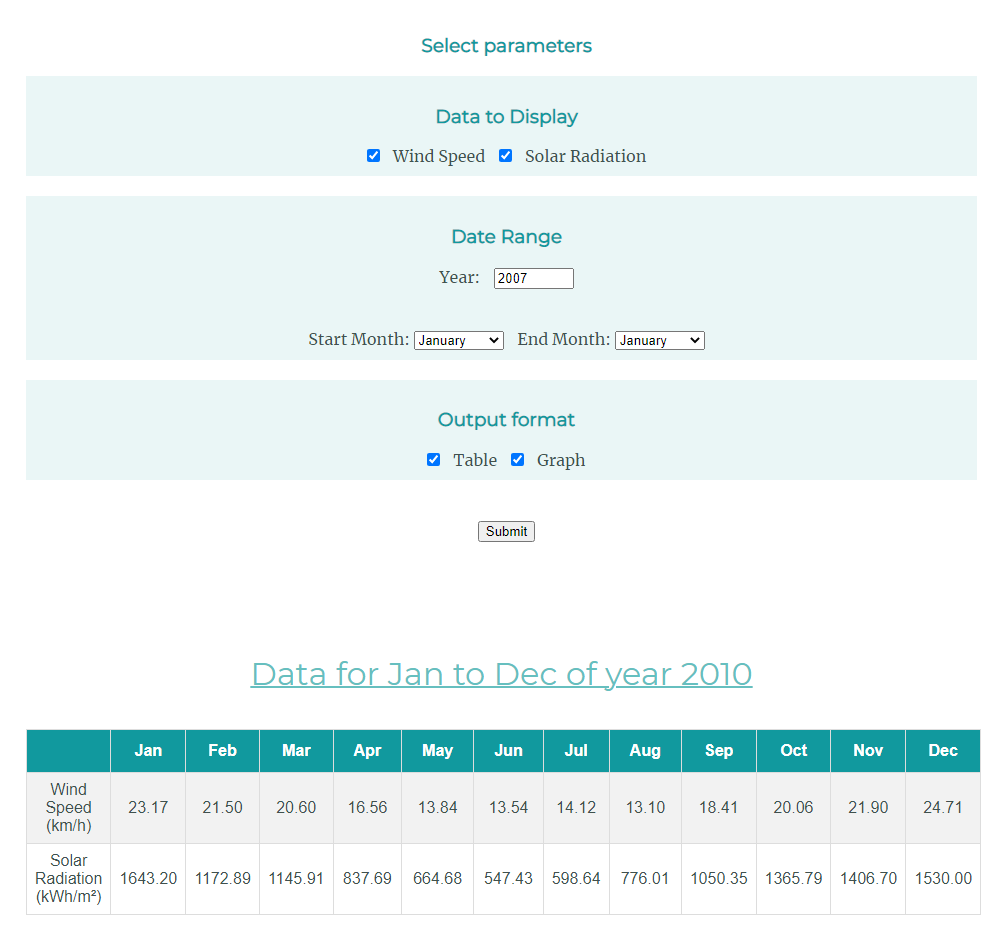
**XML result with both table and graph data**Chart

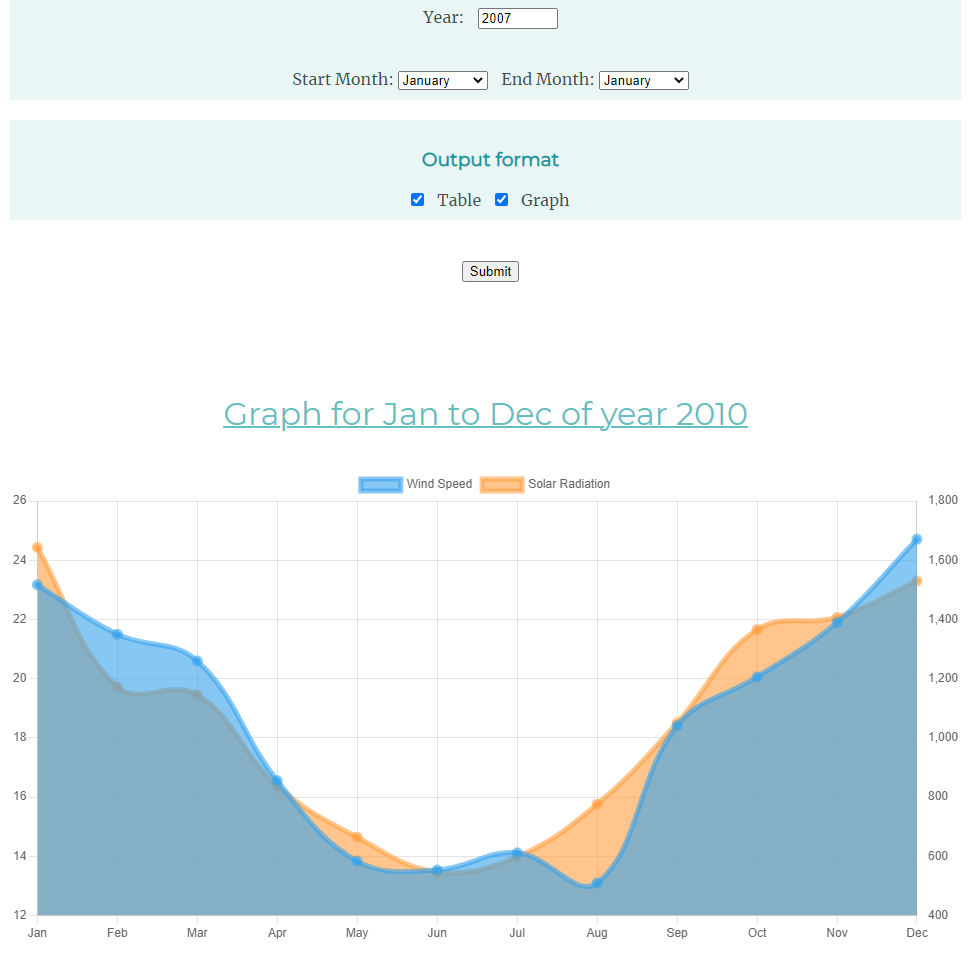
Description automatically generated

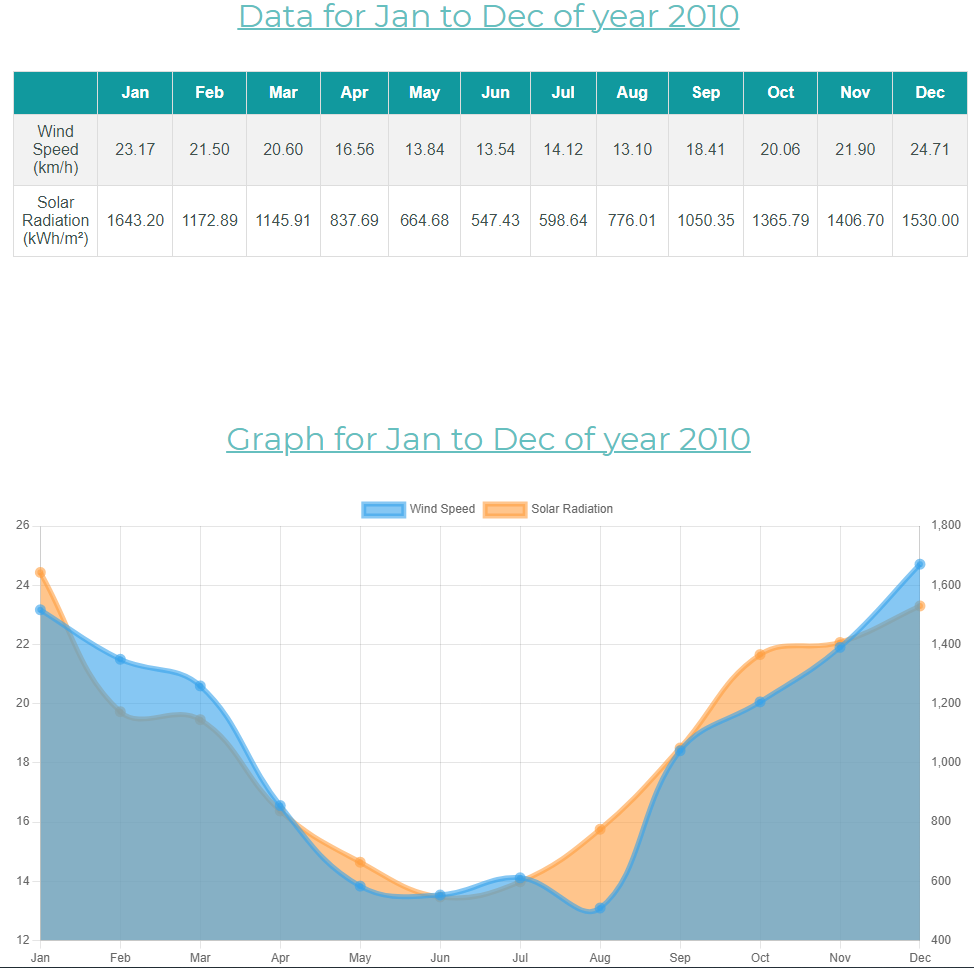
**XML result with only Wind Speed data**

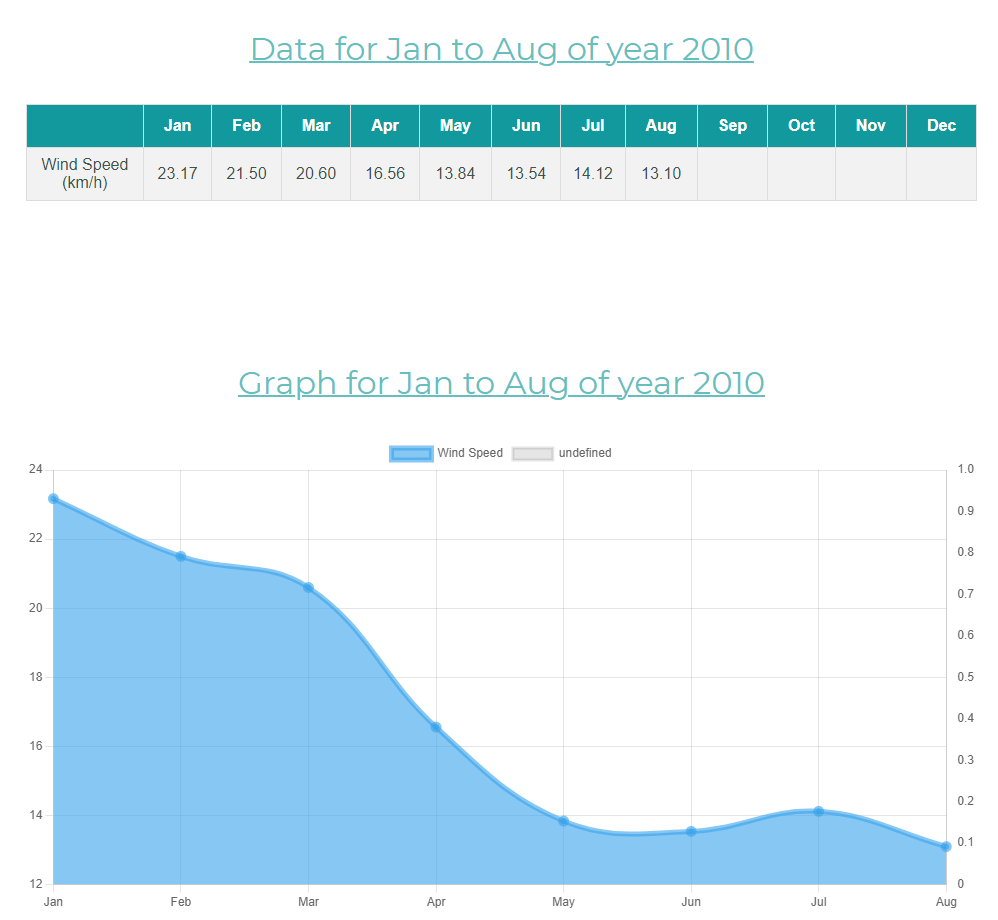
**XML result with only Solar Radiation data**

## Result of JSON data

**JSON result with only table data**

**JSON result with only graph data**

**JSON result with both table and graph data**

**JSON result with only Wind Speed data**

**JSON result with only Solar Radiation dataChart, line chart

Description automatically generated**

# Conclusion

Retrieval of XML and JSON data from specified URL (http://it.murdoch.edu.au/~S900432D/ict375/data/) without downloading the files was achieved, it is able to process the data retrieved and calculate the data, the requested data was successfully displayed on the table and/or graph. When the XML or JSON files does not provide data for some of the months, it successfully displays value of 0 for table and graph and when the months that is not requested by the user, it displays no value on the table. I am able to achieve average of Wind Speed and converting m/s to km/h. However, I was not able to achieve the correct calculation for Solar Radiation and getting all the Solar Radiation data of 1000 and above only, my solution calculates all the Solar Radiation data.

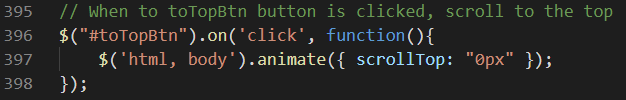
Table and/or graph was displayed successfully with the correct XML or JSON data depending on the options user selected. When the user unchecked the table and/or graph display option, it is hidden.

I have added additional animation feature, after the user clicks on submit, and when the data is received, the screen will automatically smooth scroll to the bottom of the screen to show user the data they have selected. And when user wants to go back to the filling and selecting options for the form again, there is an arrow button on the bottom right of the screen to assist user to click on it and conveniently go back to the top of the screen without scrolling a few times on the mouse’s scroll wheel. This is achieved by the code below:

**Scrolling to bottom of the screen when data is received:**



**Scrolling to top of the screen when “to top” button is clicked:**



**“To top” button location on the website:**

Graphical user interface

Description automatically generated with medium confidence