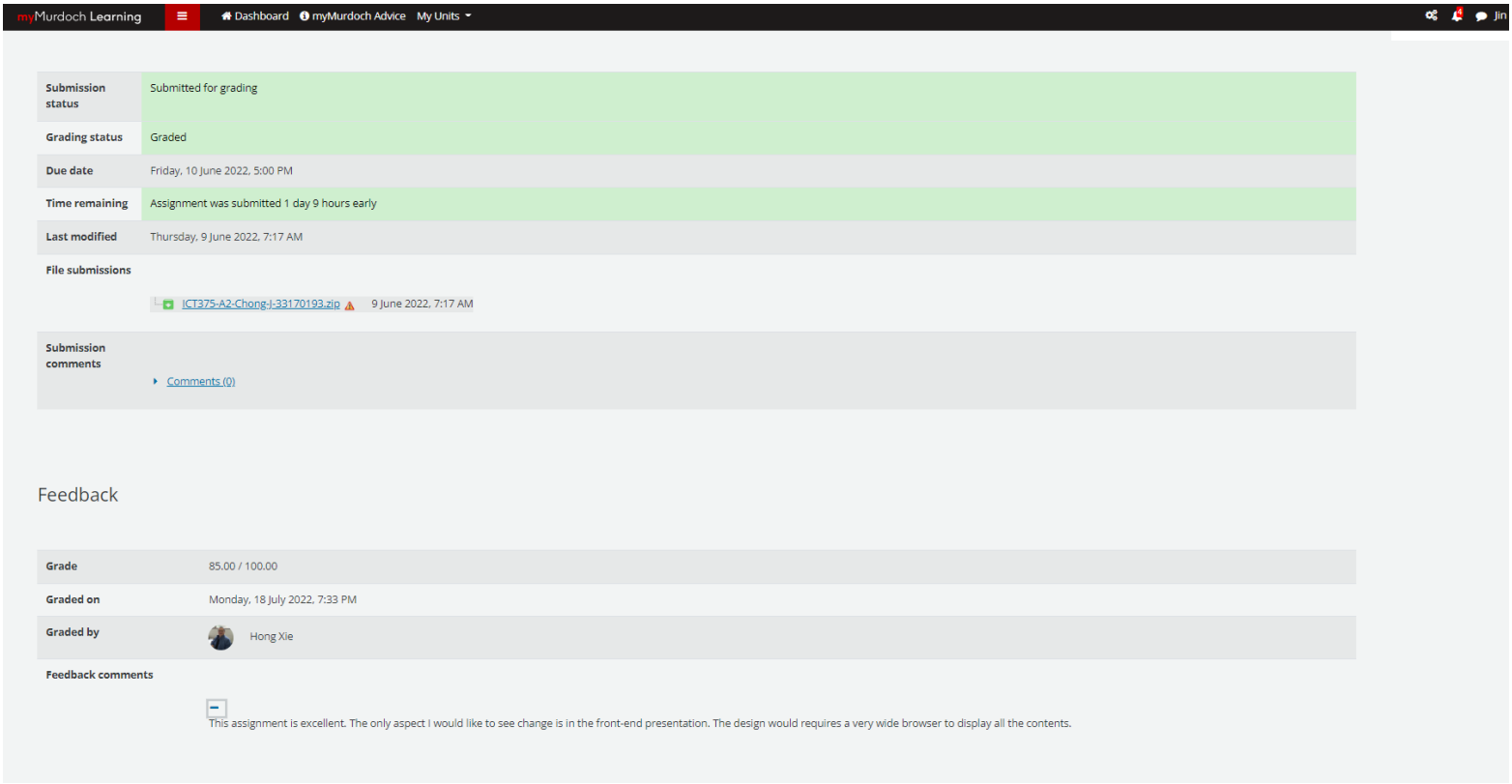


1) Introduction

My assignment is a system that allows a client to enter the weather data information they would like and the server will display the information back to them. In terms of the implementation the client entered nominated year dictates whether an XML document weather data or JSON document weather will be handled. This implementation decision was made in the event where a year file 2016 had both 2016.XML and 2016.JSON (refer to loadData.js). Also, when loading data to the application my assignments prioritize application security which is valid whenever data from external source is being handled or used. The calculations were validated with both excel and my assignment 2 program In ICT283 which had the same calculations. Also, ICT283 program calculations were marked as correct.

Assumption:

- Assume that for graph month(s) that has no data and is within the selected time frame the month will be extrapolated meaning the value for the month with no data is assigned based on extrapolation. However, if the graph month that has no data is at the first or last spot in x-axis then extrapolation will obviously not occur
- Assume when a line (in data file) has a solar radiation that is not valid and a wind speed that is valid don't ignore the whole line just throw away the invalid solar radiation value and keep the wind speed as part of the calculation
- Assume when a line (in data file) has a wind speed that is not valid and a solar radiation that is valid don't ignore the whole line just throw away the invalid wind speed value and keep the solar radiation as part of the calculation
- Assume when date/time (in data file) for a line is invalid throw away the entire line
- Assume solar radiation and wind speed is rounded to two decimal places

- Assume sphinx url referred to in marking guide is in this format
    - http://sphinx.murdoch.edu.au/~20010930/ICT375/2010.json
- Assume 0 wind speed is valid

2) XML OR JSON technologies and why used

My application can handle both XML and JSON documents. When client submits a search. The year in which is selected will dictate where the server will handle a JSON document or XML document. This decision can be seen in LoadDataFromWebsite(..) in loadData.js.

When a JSON document is determined to be handled the entire JSON file will be read asynchronously from website URL first. When dealing with file operations especially large files, asynchronous request are appropriate. In this case, it provides better user experience by keeping user interface responsive and not frozen when reading the JSON file. This is seen in LoadJSONFromWebsite(...) in loadData.js. Because the request is very large it is sent to server through chunks. Once all chunks have been accumulated it will parse the JSON document into Javascript object using built in JSON.parse(…). Similarly, if the JSON document from the website URL is not reachable the JSON document stored locally is read asynchronously. This is seen in LoadJSONFromLocal(..) in loadData.js. Once local JSON document is read it is is parsed into JavaScript object using built in JSON.parse(…). The object is processed and transversed using dot notation. This is illustrated in ExtractWindSpeeds(…) in windSpeedCalculation.js . Where for example the windspeed property (ws) outlined in the JSON schema is used to retrieve the windspeed value. This approach was selected because both the client side and server side will handle data and having JSON for both makes it easier. Also, JSON format provides dot notation for accessing which is simple since the JSON schema is not complex

When an XML document is determined to be handled the entire XML file will be read asynchronously from website url first. This is done in LoadXMLFromWebsite(...) in loadData.js. The XML data will be parsed using xml2js SAX library asynchronously. The parseString method in xml2js will parse and process XML data to JSON format. This is illustrated in LoadXMLFromWebsite(..) in loadData.js. In the event the XML document is unreachable on the website it reverts to local XML document and the same process is followed. This approach was selected since it enables reusability of functions such as GetSolarRadiation(…) and GetWindSpeed(…) in windSpeedCalculation.js and solarRadiationCalculation.js respectively. Also allows the use of light weight easy to use JSON format which allows for dot notation.

The disadvantage of taking and XML document and converting it to JSON format with xml2js SAX is that it increased the load time. Reading a JSON document and converting JSON format is quite fast. However, when it comes to the XML document the document needs to be converted JSON format by a third-party library. This extra step adds extra time as opposed to reading XML document and parsing XML data with your own parser. However, the extra load time is justified in that I could reuse

existing well tested code to process the JSON format for calculations. In addition, the xml2js parser is well tested since it has been used widely for so long. Creating your own parser does not have the benefit of being well tested or robust incase the XML format changes in which we would need to modify our created parsers and test.

3) Detailed description of design

Search weather measurements

- index.html
  - Provides the client form to submit new search to access weather information
- searchFormValidator.js
  - Responsible for the client-side validation of index.html
  - Ensures time-frame is valid meaning start month < end month, a year is a number, and at least a display format and weather measurement is selected
  - Provides client-side feedback when submission is not valid
  - The importance of client-side validation is it provides feedback fast to the client and avoids server being accessed on when it doesn't need to be. Also calls the ProcessSearchForm() in SearchFormFrontEnd.js if valid
- SearchFormFrontEnd.js
  - Handles initiating the asynchronous JQuery AJAX post sending the client weather measurement information request to the backend through request handler(s) in order to receive the weather data from server. The follow functions encapsulate this AJAX post-
    - GetSolarRadiation(…), GetWindSpeed(..), and GetSolarRadiationAndWindspeed(…)
  - Handles constructing the table and line graph containing requested weather data
- requestHandlers.js
  - Either reqSolarRadiation(…), reqWindSpeed(…) or reqSolarRadiationAndWindSpeed(…) request handler is called depending on client's submitted Weather Search Form selection.
    All of three request handlers will generally perform these operations
    - Call LoadDataFromWebsite(…) from loadData.js
    - Call GetSolarRadiation(…), GetWindSpeed(..) or both from windSpeedCalculation.js and solarRadiationCalculation.js respectively
    - Return JSON string containing client's requested weather data to SearchFormFrontEnd.js via $.post(…) call-back function
  - The importance for the requestHandler file is to provide a place where when a resource is requested by a client or browser, often through URL, an appropriate response is provided by server

- loadData.js
  - Responsible for accessing appropriate weather data files and extracting the year weather data into JavaScript object

- Contains LoadDataFromWebsite(…) and this function determines whether to load data from web XML or JSON document depending on client selected year
- Contains LoadJSONFromWebsite(...) and LoadXMLFromWebsite(…) responsible for accessing the weather data file from web and converting it to JavaScript object. These two function will also call LoadDataFromLocalFile(…) if network is unreachable
- Once extracted to JavaScript object a callback is executed sending JavaScript object back to reqSolarRadiation(…), reqWindSpeed(…) or reqSolarRadiationAndWindSpeed(…) request handler in requestHandlers.js
- The importance of a separate loadData.js is firstly to make code easier to read and understand. So, any modifications to loading data can be located in one file making modification to existing code easier especially when the code base gets larger.

- windSpeedCalculation.js & solarRadiationCalculation.js
  - Both the files' handles their respective calculation the function(s) that start the calculations are - GetSolarRadiation(…), GetWindSpeed(..)
    - Total Monthly Solar Radiation and Average Monthly Windspeeds
  - In order to get their respective calculation, the data needs to extracted from the JavaScript object and then processed
  - The importance of both files is to ensure server-code is modular in design and single responsibility principle is maintained. Also only the GetSolarRadiation(…), GetWindSpeed(..) are exposed to the client. The other functions are only accessible to the functions inside the file so file remains tightly encapsulated and abstraction is promoted by hiding implementation. This increased readability

- windSpeedDatabase.js & solarRadiationDatabase.js
  - This file contains a class that encapsulates the map data structure that stores the windspeed/solarRadiation records
  - Ensures that records are added in a controlled way this is seen in AddRecord(…) function. This is responsible for adding to the map
  - The creation of both files makes sure the client can't accidentally add a single record to the month key and override the array already associated to the month key

- databaseUtilities.js
  - Contains functions that are utility/helper functions for database class meaning it is reusable for both windSpeedDatabase.js & solarRadiationDatabase.js
  - The file contains functions that are not appropriate to be placed with utilitiesFunctions.js since that file is for general functions that can be used in multiple scenarios. For example, CalcAverage
  - The creation of a standalone database utilities encourages code reuse providing functionality to both windSpeedDatabase.js & solarRadiationDatabase.js

- utilitiesFunctions.js
  - This file contains utility/helper functions that are reusability in many different scenarios even in situations not related to the assignment context

- The file was created to encourage reusability as well as ensuring the general functions were not tied another files code such as requestHandler.js code. Had the utility functions been defined there then in order for other files to access the functions we would need to expose the requestHandler.js file thus breaking file encapsulation

4) Description of data structures and why

Role of array:

The array structure was used heavily in this assignment. An array stores its elements in contiguous memory which means every element will be located close to each other. This means when every element in the data structure needs to be accessed the array is efficient. There were many instances where a lot of data had to be stored in a data structure, order of the data did not matter, and every element in the data structured had to be accessed. The mathematical calculations functions such as- CalcTotal and CalcAverage was where an array was most appropriate. In this case, the calcTotal would take an array as the parameter. Order in the array did not matter and every element would need to be accessed since the total is basically calculating the total of all the elements in the array. Thus, for this situation the use of array is best since it provides best time complexity for accessing every element with a time complexity always performed at O(1).

Role of windSpeedDatabase.js and solarRadiationDatabase.js:

Both of the files encapsulate a map data structure. The reason for the encapsulation is the need to provide controlled access or more specifically controlled insertion of data. The general format for the encapsulated map data structure is <MonthNumber, ArrayofData[]>. The class ensure that when a new record is added to the data structure the record gets added to the array associated with the correct month key. We also want the client to be able to accidentally override a month key with just one record thereby deleting the entire array of records associated with month key. This is where encapsulating the data structure comes in to assist. Allowing the class to be responsible for this eliminates the need for code duplication of the complex insertion process. The reason why having an array of data values for a month is appropriate is because a month can contain many data values. Also, the processing required such as calculating total solar radiation (per month) required all elements for a month to be accessed.

The map data structure was also used in this assignment in the windSpeedDatabase and solarRadiationDatabase class. The map provided the following-

- The keys were the months in a year which is unique

- Allowed for the value that was paired with the month key to be an array of data for that month (key)
- The map keeps the orders of the keys preserved

Role of result data structure:

Finally the var result = {}; data structure in reqSolarRadiation(…), reqWindSpeed(…) or reqSolarRadiationAndWindSpeed(…) in requestHandlers.js file was the last data structure used to return the requested information for the specified timeframe back to client. The result data structure had to be a JavaScript object so ES6 map was not allowed. The form of the result data structure was the following

{ws: [avgMonthlyWindSpeedVal1, avgMonthlyWindSpeedVal2 …]

sr: [totalSRMonth1, totalSRMonth2, …] }

Where the ws and sr were the keys and the values was an array of calculated results for nominated time frame as per the specifications. When a month inside time frame had no value, the element was represented in the array as “”. The result data structure was stringify in order to send it to the front end via AJAX.

5) Feature testing table (refer to conclusion to see how the test are relevant to specific requirements)

Evidence is found here-

https://www.youtube.com/playlist?list=PLUktbenWQI5iisFuY0Z6Rf7HxU_hAEO2G

Test #	Test objective(s)	Test steps	Expected results	Pass/ fail	Evidence (video title)
1	Validate ability to correctly graph wind speeds	Go to http://ceto.murdoch.edu.au:40004/ Select WindSpeed only Select graph format only Fill out search form- Year: 2014 Timeframe: Feb to Aug	Displays graph with wind speed(s) plotted- Feb Mar 20.28 Apr 13.71 May 17.08 Jun 4.65 Jul 12.71 Aug 18.97	Pass	(Test Case validate ability to correctly graph)
2	Validate ability to correctly graph solar radiations	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations only Select graph format only	Displays graph with solar radiations(s) plotted- May 104.89 Jun 75.14	Pass	(Test Case validate ability to

		Fill out search form- Year: 2008 Timeframe: May to June			correctly graph)
3	Validate ability to correctly graph both solar radiations and windspeeds	Go to http://ceto.murdoch.edu.au:40004/ Select both solar radiations and windspeed Select graph format only Fill out search form- Year: 2012 Timeframe: Jan to Dec	Displays graph with solar radiations(s)- Jan 219.44 Feb 201.73 Mar 209.17 Apr 127.99 May 103.54 Jun 62.29 Jul 100.71 Aug 119.36 Sep 155.61 Oct 201.50 Nov 223.16 Dec 244.69 Displays graph with wind speed(s) plotted- Jan 27.14 Feb 22.41 Mar21.49 Apr 16.77 May 15.22 Jun 19.92 Jul 12.73 Aug 14.89 Sep 21.97 Oct 21.19 Nov 21.49 Dec 19.88	Pass	(Test Case validate ability to correctly graph)
4	Validate ability to correctly create table of wind speeds	Go to http://ceto.murdoch.edu.au:40004/ Select WindSpeed only Select table format only Fill out search form- Year: 2014 Timeframe: Feb to Aug	Displays table with- ws (km/h) Jan Feb Mar 20.28 Apr 13.71 May 17.08 Jun 4.65 Jul 12.71 Aug 18.97 Sep Oct Nov Dec	Pass	(Test Case validate ability to correctly create table)
5	Validate ability to correctly create table of solar radiations	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations only Select table format only Fill out search form- Year: 2010 Timeframe: Feb to Mar	Displays table with- sr (kWh/m^2) Jan Feb 188.33 Mar 182.45 Apr May Jun Jul Aug Sep Oct Nov Dec	Pass	(Test Case validate ability to correctly create table)
6	Validate ability to correctly create table of both wind speeds and solar radiations	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations and wind speeds Select table format only Fill out search form- Year: 2010 Timeframe: Feb to Mar	Displays table with- ws (km/h) sr (kWh/m^2) Jan Feb 21.50 188.33 Mar 20.60 182.45 Apr May Jun Jul	Pass	(Test Case validate ability to correctly create table)

			Aug Sep Oct Nov Dec		
7	Validate ability to both correctly graph & create table of wind speed	Go to http://ceto.murdoch.edu.au:40004/ Select WindSpeed only Select graph format and table format Fill out search form- Year: 2014 Timeframe: Feb to Aug	Displays graph with wind speed(s) plotted- Feb Mar 20.28 Apr 13.71 May 17.08 Jun 4.65 Jul 12.71 Aug 18.97 Displays table with- ws (km/h) Jan Feb Mar 20.28 Apr 13.71 May 17.08 Jun 4.65 Jul 12.71 Aug 18.97 Sep Oct Nov Dec	Pass	(Test Case validate ability to both correctly graph & create table)
8	Validate ability to both correctly graph & create table of solar radiation	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations only Select table format and graph format Fill out search form- Year: 2010 Timeframe: Feb to Mar	Displays table with- sr (kWh/m^2) Jan Feb 188.33 Mar 182.45 Apr May Jun Jul Aug Sep Oct Nov Dec Displays graph with solar radiations(s) plotted- Feb 188.33 Mar 182.45	Pass	(Test Case validate ability to both correctly graph & create table)
9	Validate ability to both correctly graph & create table of solar radiation & wind speed	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiation and wind speed Select table format and graph format Fill out search form- Year: 2010 Timeframe: Feb to Mar	Displays table with- ws (km/h) sr(kWh/m^2) Jan Feb 21.50 188.33 Mar 20.60 182.45 Apr May Jun Jul Aug Sep Oct Nov Dec Displays graph with solar radiations(s) plotted- Feb 188.33 Mar 182.45	Pass	(Test Case validate ability to both correctly graph & create table)

			Displays graph with wind speed plotted- Feb 21.50 Mar 20.60		
10	Validate application is a single page application	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiation and wind speed Select table format and graph format Fill out search form- Year: 2009 Timeframe: Feb to Sep Submit Change Year: 2011 Submit	Current page is rewritten with new data from the server and thus allows client to submit more searches without needing to reload .	Pass	(Test Case validate SPA)
11	Validate there is a script to start and control application	node index.js	Display in terminal- Output server running indicating application has started	Pass	(Test Case validates adopts assignment 1 modular server application structure)
12	Validate a script to start a server	node index.js	Display in terminal- "Server running http://ceto.murdoch.edu.au:40004/"	Pass	(Test Case validates adopts assignment 1 modular server application structure)
13	Validate there is a script to route the different client request to appropriate request handlers	Uncomment line 4 in router.js Save router.js Type node index.js Go to http://ceto.murdoch.edu.au:40004/	Display in terminal- Server running http://ceto.murdoch.edu.au:40004/ Proces ID: entered router(...) router.js entered reqStart(...) in requestHandlers.js entered router(...) router.js entered router(...) router.js entered reqSearchFormFrontEndJs(...) in requestHandlers.js	Pass	(Test Case validates adopts assignment 1 modular server application structure)
14	Validate searchSolarRadiation request handler	Go to http://ceto.murdoch.edu.au:40004/ Fill in valid search form with only solar radiation measurement ticked	Display in terminal- … entered reqSolarRadiation(...) in requestHandlers.js ….	Pass	(Test case validate all requestHandlers)
15	Validate searchWindSpeed request handler	Go to http://ceto.murdoch.edu.au:40004/ Fill in valid search form with only wind speed measurement ticked	Display in terminal- … entered reqWindSpeed(...) in requestHandlers.js …	Pass	(Test case validate all requestHandlers)
16	Validate searchSolarRadiationAndWindSpeed request handler	Go to http://ceto.murdoch.edu.au:40004/ Fill in valid search form with both wind speed and solar radiation measurement ticked	Display in terminal- … entered reqSolarRadiationAndWindSpeed(...) in requestHandlers.js …	Pass	(Test case validate all requestHandlers)

17	Validate mainStyleSheet request handler	Go to http://ceto.murdoch.edu.au:40004/mainStyleSheet	Output main.css code on browser	Pass	(Test case validate all requestHandlers)
18	Validate searchFormFrontEnd request handler	Go to http://ceto.murdoch.edu.au:40004/searchFormFrontEnd	Output searchFormFrontEnd.js code on browser	Pass	(Test case validate all requestHandlers)
19	Validate searchFormValidator request handler	Go to http://ceto.murdoch.edu.au:40004/searchFormValidator	Output searchFormValidator.js code on browser	Pass	(Test case validate all requestHandlers)
20	Validate the ability to download (at runtime) the user nominated XML File	Go to http://ceto.murdoch.edu.au:40004 Select WindSpeed only Select table format only Fill out search form- Year: 2009 Timeframe: Feb to Aug	Display in terminal- … entered LoadXMLFromWebsite(...) in loadData.js …	Pass	(Test Case download weather data or load local files)
21	Validate defaults to locally stored XML file if network (i.e. sphinx website) is unreachable	Edit loadData.js line 13 Change const websitePath = "http://sphinx.murdoch.edu.au/~20010930/CT375/" Save to emulate network down Go to http://ceto.murdoch.edu.au:40004 Select WindSpeed only Select table format only Fill out search form- Year: 2009 Timeframe: Feb to Aug	Display in terminal- … entered LoadDataFromLocalFile(...) in loadData.js …	Pass	(Test Case download weather data or load local files)
22	Validate the ability to download (at runtime) the user nominated JSON File	Go to http://ceto.murdoch.edu.au:40004 Select solar radiation only Select table format only Fill out search form- Year: 2016 Timeframe: Feb to Aug	Display in terminal- … entered LoadJSONFromWebsite(...) in loadData.js …	Pass	(Test Case download weather data or load local files)
23	Validate defaults to locally stored JSON file if network (i.e. sphinx website) is unreachable	Edit loadData.js line 13 Change const websitePath = "http://sphinx.murdoch.edu.au/~20010930/CT375/" Save to emulate network down Go to http://ceto.murdoch.edu.au:40004 Select solar radiation only Select table format only Fill out search form- Year: 2016 Timeframe: Feb to Aug	Display in terminal- … entered LoadDataFromLocalFile(...) in loadData.js …	Pass	(Test Case download weather data or load local files)
24	Validate data structures used to store extracted data for windspeeds of nominated time frame	Go to http://ceto.murdoch.edu.au:40004/ Select WindSpeed only Select table format only Fill out search form- Year: 2016 Timeframe: Feb to Apr	Display in terminal a map with only key 2, 3 both with array of values	Pass	(Test Case data structure used to store extracted data)
25	Validate data structures used to store extracted data for solar radiations of nominated time frame	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations only Select table format only Fill out search form- Year: 2016 Timeframe: Feb to Apr	Display in terminal a map with only key 2, 3 both with array of values	Pass	(Test Case data structure used to store extracted data)

26	Validate wind speed calculations data structure is sent back to client (AJAX)	Go to http://ceto.murdoch.edu.au:40004/ Select WindSpeed only Select table format only Fill out search form- Year: 2014 Timeframe: Feb to Apr	Display in terminal- { ws: ['', '20.28', '13.71'] } Display in browser- ws: Array(3) 0: "" 1: "20.28" 2: "13.71"	Pass	(Test case data structure sent to client AJAX)
27	Validate solar radiations data structure is sent back to client (AJAX)	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations only Select table format only Fill out search form- Year: 2011 Timeframe: Mar to Jun	Display in terminal- { sr: ['207.39', '144.16', '107.07', '71.74'] } Display in browser- sr: Array(4) 0: "207.39" 1: "144.16" 2: "107.07" 3: "71.74"	Pass	(Test case data structure sent to client AJAX)
28	Validate wind speed and solar radiation data structure is sent back to client (AJAX)	Go to http://ceto.murdoch.edu.au:40004/ Select solar radiations and wind speed Select table format only Fill out search form- Year: 2016 Timeframe: Feb to Apr	Display in terminal- { sr: ['225.45', '0.96', ''], ws: ['20.58', '9.03', ''] } Display in browser- sr: Array(3) 0: "225.45" 1: "0.96" 2: "" ws: Array(3) 0: "20.58" 1: "9.03" 2: ""	Pass	(Test case data structure sent to client AJAX)

29	Validate server calculates wind speed calculations from XML document correctly for given time frame	Go to (Microsoft Edge)
		http://ceto.murdoch.edu.au:40004/
		Select wind speed
		Select table format only
		Fill out search form-
		Year: 2008
		Timeframe: Jan to Dec
			Display in terminal-
			{
			ws: [
			'24.85', '44.56',
			'43.94', '40.94',
			'44.26', '32.70',
			'31.41', '20.04',
			'33.36', '37.38',
			'38.53', '45.23'
			]
			}
				Pass	(Test Case validate server calculation)

30	Validates server calculates wind speed calculations from JSON document correctly for given time frame	Go to (Microsoft Edge)
		http://ceto.murdoch.edu.au:40004/
		Select wind speed
		Select table format only
		Fill out search form-
		Year: 2014
		Timeframe: Jan to Dec
			Display in terminal-
			{
			ws: [
			'9.95', '',
			'20.28', '13.71',
			'17.08', '4.65',
			'12.71', '18.97',
			'20.58', '18.89',
			'20.34', '21.74'
			]
			}
				Pass	(Test Case validate server calculations)

31	Validate server calculates solar radiation calculations from XML document correctly for given time frame	Go to (Microsoft Edge) http://ceto.murdoch.edu.au:40004/ Select solar radiation Select table format only Fill out search form- Year: 2009 Timeframe: Aug to Dec	Display in terminal- { sr: ['42.25', '48.44', '', '99.14', '271.19'] }	Pass	(Test Case validate server calculations)

32	Validate server calculates solar radiation calculations from JSON document correctly for given time frame	Go to
		(Microsoft Edge)
		http://ceto.murdoch.edu.au:40004/
		Select solar radiation
		Select table format only
		Fill out search form-
		Year: 2013
		Timeframe: Mar to Oct
			Display in terminal-
			{
			sr: [
			'185.29', '118.69',
			'103.45', '87.77',
			'87.31', '109.31',
			'130.42', '201.42'
			]
			}
				Pass	(Test Case validate server calculations)

33	Client program is pleasant with IPHONE SE	Go to http://ceto.murdoch.edu.au:40004/ Change dimensions to Iphone SE Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent	Pass	(Test Case application works on different devices)
34	Client program is pleasant with Samsung Galaxy S8+	Go to http://ceto.murdoch.edu.au:40004/ Change dimensions to Samsung Galaxy S8+ Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent	Pass	(Test Case application works on different devices)
35	Client program is pleasant with Ipad Mini	Go to http://ceto.murdoch.edu.au:40004/ Change dimensions to Ipad Mini Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent	Pass	(Test Case application works on different devices)
36	Client program is pleasant with Pixel 5	Go to http://ceto.murdoch.edu.au:40004/ Change dimensions to Pixel 5 Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent	Pass	(Test Case application works on different devices)
37	Client program is pleasant with laptop	Go to http://ceto.murdoch.edu.au:40004/ Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent	Partial pass (not aesthetically pleasing)	(Test Case application works on different devices)
38	Client program is pleasant with IPad Air	Go to http://ceto.murdoch.edu.au:40004/ Change dimensions to IPad Air Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent on any device	Partial pass (not aesthetically pleasing	(Test Case application works on different devices)
39	Client program is pleasant with Surface pro 7	Go to http://ceto.murdoch.edu.au:40004/ Change dimensions to Surface pro 7 Select wind speed and solar radiation Select table and graph Fill out search form- Year: 2013 Timeframe: Jan to Dec	Program is aesthetically pleasing, proportional and consistent on any device	Partial pass (not aesthetically pleasing	(Test Case application works on different devices)

6) Conclusion

Achieved requirements-

Client program provides a table output (displays all months, even if some months have no data) and a graphical output (displays user specified months only)

- Client program provides a table output (displays all months, even if some months have no data) and a graphical output (displays user specified months only)
    - Test #1 – Test #9
- A table output displays all months, even if some months have no data and a graphical output displays user specified months only
    - Test #7

Client program provides 4 appropriate gathering mechanisms to get user input (year, start and end months, wind speed and/or solar radiation, and table and/or graph display).

- Client program provides 4 appropriate gathering mechanisms (year, start and end months, wind speed and/or solar radiation, and table and/or graph display) to get user input
    - Test #1 - Test #9

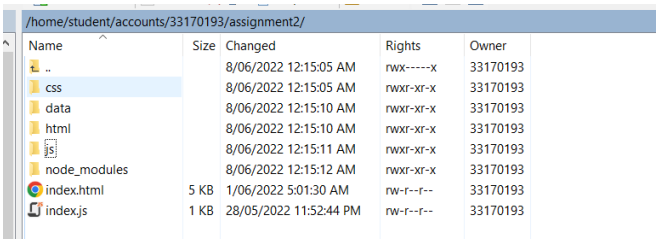
Client program presented in an aesthetically pleasing manner (using css). Must use of single page application (i.e. AJAX). It is suggested that the client program uses three div's. Top most div for input, middle div for table output, and bottom most div for graphical output.

- Single page application for all client-side output
    - Test #10
- Single page application uses AJAX for all client-side output
    - Test #26, Test #27, Test #28
- Client program presented in an aesthetically pleasing manner (using css) mobile
    - Test #33 – Test #36
- Client program works with different browsers (edge)
    - Test #29 – Test #31

Server program (i.e. Node.js). Must follow the instruction from assignment 1 to make your application modular in design. Must not use port 80; use the one given to you in week 1. Must include all server-side scripts used in processing; these should be well organized in appropriate directories. Also include the node_modules directory and any sub-directories (i.e. npm modules).

- Must follow the instruction from assignment 1 to make your application modular in design.
    - A script to start and control the application
        - Test #11
    - A script to start a server (port)
        - Test #12
    - A script to route the different client request to appropriate request handlers
        - Test #13

    - Request handler for requesting solar radiation
        - Test #14
    - Request handler for requesting wind speed
        - Test #15
    - Request handler for requesting solar radiation & wind speed
        - Test #16
    - Request handlers for style sheet (resource)
        - Test #17
    - Request handlers for searchFormFrontEnd (resource)
        - Test #18
    - Request handlers for searchFormValidator (resource)
        - Test #19
- Uses port 40004
    - All the browser test cases show the use of this port never port 80
- The application is organised in appropriate directories according to the “Coding Standard and File Organization”



Interface with sphinx URL; i.e. the ability to download (at runtime) the user nominated XML OR JSON file for parsing and processing. Defaults to locally stored data files if network (i.e. sphinx website) is unreachable.

- Interface with sphinx URL; i.e. the ability to download (at runtime) the user nominated XML File
    - Test #20
- Defaults to locally stored XML file if network (i.e. sphinx website) is unreachable
    - Test #21
- Interface with sphinx URL; i.e. the ability to download (at runtime) the user nominated JSON file
    - Test #22
- Defaults to locally stored JSON file if network (i.e. sphinx website) is unreachable.
    - Test #23

Data structures used to store extracted data. That is, arrays, associative arrays, objects, etc.

- Data structures used to store extracted data for windspeeds of nominated time frame
    - Test #24
- Data structures used to store extracted data for solar radiations of nominated time frame
    - Test #25

Use of XSLT, OR JSON equivalent, for processing on server-side to facilitate the display of formatted table and graph (proportionately and consistent on any device).

Includes data retrieval (i.e. parsing) from XML OR JSON file. Also, application design (i.e. accessing correct data and calculation of results required to be sent to the client for display). That is, all components achieving the task of parsing and processing within the Node.js environment using XML OR JSON technologies. This must be done on the server-side.

- Server returns the calculated average monthly windspeed for specified time frame back to the client via JSON string (AJAX)
    - Test #26
- Sever returns the total solar radiations per month for specified time frame back to client via JSON string (AJAX)
    - Test #27
- Server returns both calculated average monthly windspeed & total solar radiations per month for specified time frame back to client via JSON string (AJAX)
    - Test #28
- XML file parse and process for wind speed for nominated month in Node.js environment
    - Test #29
- JSON file parse and process for wind speed for nominated month in Node.js environment
    - Test #30
- XML file parse and process for solar radiation for nominated month in Node.js environment
    - Test #31
- JSON file parse and process for solar radiation for nominated month in Node.js environment
    - Test #32
- Application is consistent on any device and proportional and consistent on any device
    - Test #33 – Test #39

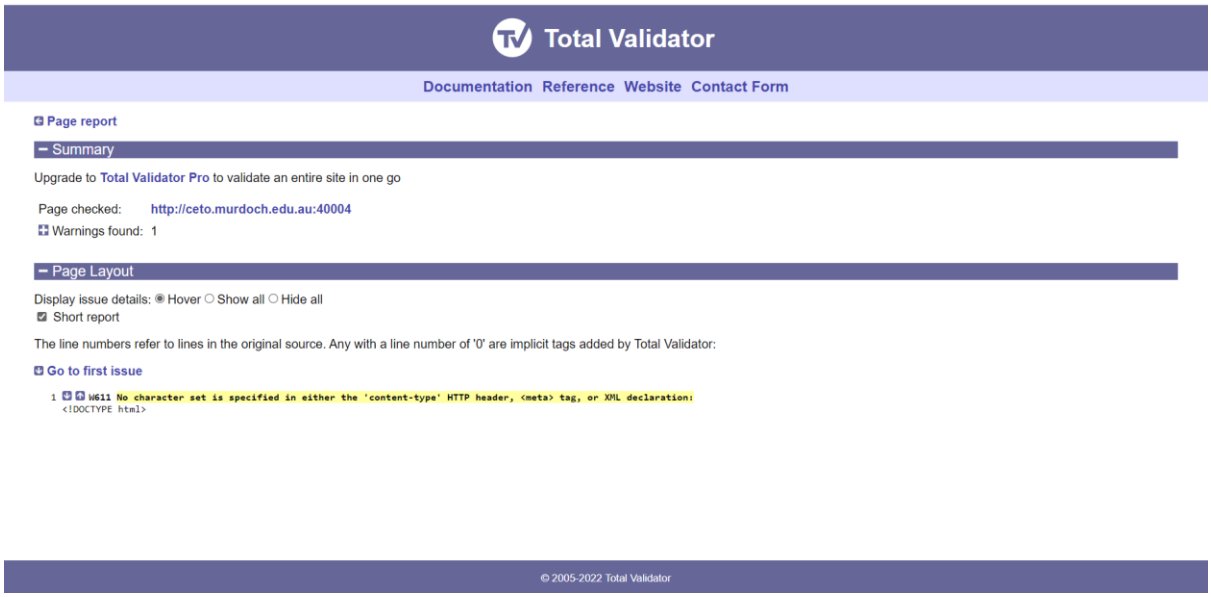
The limitation with my assignment is the following-

- Parsing and processing XML document is slower than JSON document since an external library xml2js is required to parse to JSON format
- When table outputs all months and their data the CSS does not look too aesthetically pleasing on laptop. However, it is proportional and consistent for all devices. Also, since it was designed with mobile first approach in mind it is aesthetically pleasing on mobile and has no issues when output all months and their data
    - Test #37

The highlights in my assignment are-

- I’ve added comments to describe my line of thinking and it illustrates that every decision was well thought out. Also, the comments are useful in that often when we are asked to review old code, we tend to forget why we did something. The comments answer those questions
- CSS adopted a mobile first approach which is appropriate for modern user interfaces since most people use mobiles to view sites. The site on mobile is pleasing
- SOLID principles were taken into consideration when designing the application especially when extracting functions into new files in order to follow single responsibility principle.
    - For example- windSpeedCalculation.js & solarRadiationCalculation.js. While a single file could have stored both windSpeedCalculation.js & solarRadiationCalculation.js that would break single responsibility because calculations had to handle two things windSpeedCalculation and solarRadiationCalculation. Also, if somebody wants to just use windSpeedCalculation then they are exposed to solarRadiationCalculation if a single file was used
- My requestHandler.js was able to handle both request and response for all by utilizing function callbacks to take it back to requestHandler.js

7) Additional information



W3C CSS Validator results for http://ceto.murdoch.edu.au:40004/mainStyleSheet (CSS level 3 + SVG)

Congratulations! No Error Found.

This document validates as CSS level 3 + SVG !

To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that validates. Here is the XHTML you could use to add this icon to your Web page:

```
<p>
    <a href="http://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="http://jigsaw.w3.org/css-validator/images/vcss"
            alt="Valid CSS!" />
    </a>
</p>
```

```
<p>
<a href="http://jigsaw.w3.org/css-validator/check/referer">
    <img style="border:0;width:88px;height:31px"
        src="http://jigsaw.w3.org/css-validator/images/vcss-blue"
        alt="Valid CSS!" />
    </a>
</p>
```

(close the img tag with > instead of /> if using HTML <= 4.01)

Interested in understanding what new technologies are coming out of W3C? Follow @w3cdevs on Twitter to keep track of what the future looks like!

Donate and help us build better tools for a better web.

If you like, you can download a copy of this image to keep in your local web directory, and change the XHTML fragment above to reference your local image rather than the one on this server.

If you would like to create a link to this page (i.e., this validation result) to make it easier to re-validate this page in the future or to allow others to validate your page, the URI is: