CS 4230-01

Project Part 5 Report

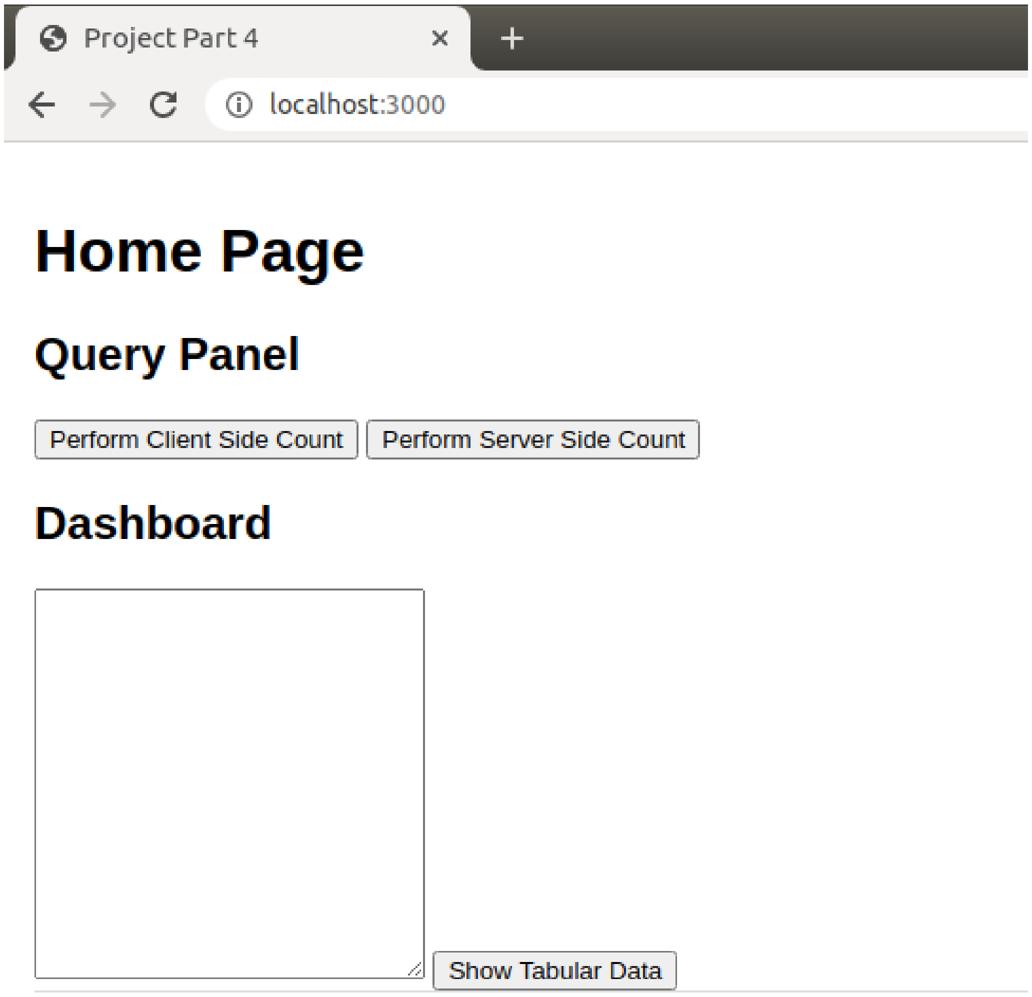
Austin Lee, Jacob Mulroy, Conner Hundt, D’Angelo Abell

10/30/2020

Part 1: Templates

Sucesses

Creating Templates: This was the easiest portion of the project assignment. We just added two new js files, QueryPanel.js and DashBoard.js to our project/public directory, and then imported them into our client/main.js. We then added the querypanel and dashboard templates into our main.html code as well.



Main Issues

N/A: We had no major issues with this portion of the assignment.

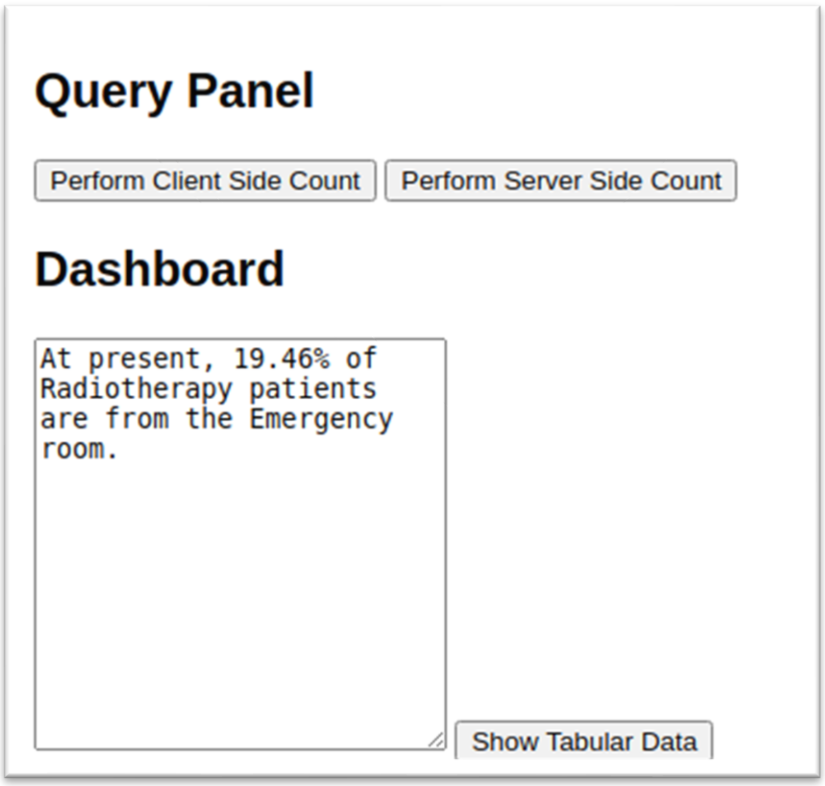
Part 2: Client-side Count

Sucesses

Implementation: This was the easier of the two counts that we coded. We simply ran a query to return all records from the database where the “Department” was “radiotherapy.” After returning that to the client side, we iterated through the array using keys to determine if the value “Severity of Illness” was equal to “Extreme.” After getting the counts we divided them to get the percentage and formatted that value into the display area.

Main Issues

N/A: Again, we really did not have any issues of note for this part.



```
At present, 19.46% of
Radiotherapy patients
are from the Emergency
room.
```

Successful query count: client-side

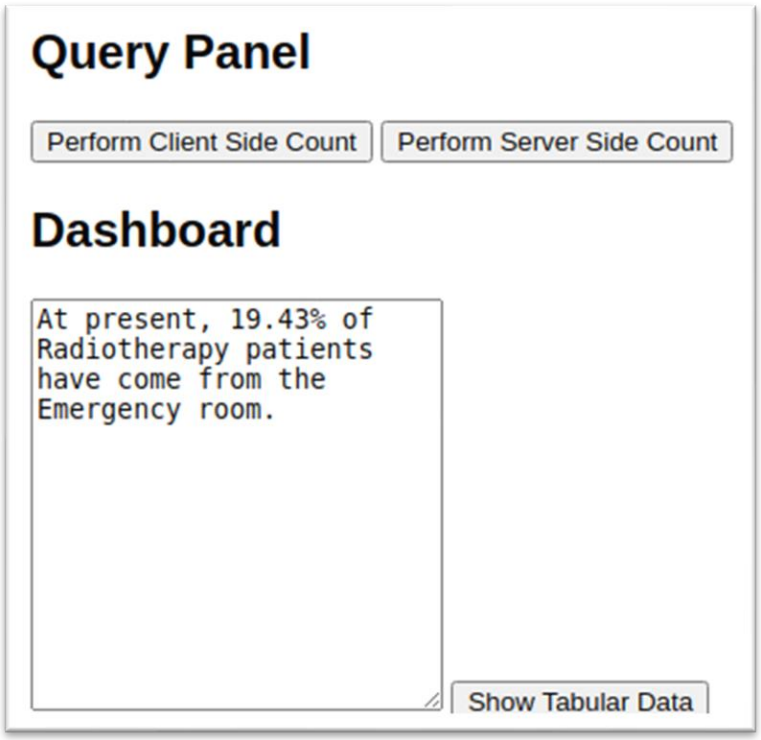
Part 3: Server-side Count

Sucesses

Implementation: We were successfully able to implement a count that was handled fully on the server-side of the program.

Main Issues

Query Formatting: Our main issue with this was trying to figure out how to structure the query, specifically getting multiple connections to execute in order. Because of the asynchronous functions, the query would sometimes execute one before the other, and vice versa, or sometimes complete and return without executing the second one at all. This caused us to return undefined or NaN counts to our client side. After restructuring the server-side access code a few times, and combining both connections into the same function, we were able to get the proper counts, perform our calculations, and return the percentage (in decimal format) back to the client side.



```
At present, 19.43% of
Radiotherapy patients
have come from the
Emergency room.
```

Successful query count: server-side

Part 4: Tabular Formatting

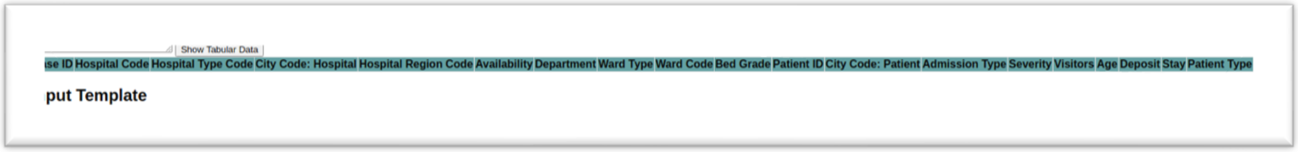
Sucesses

Implementation: Able to dynamically generate a table on our DashBoard that contains all the records matching our query.

Main Issues

Table Generation: This was an issue from our last project part as well: dynamically generating an HTML table and loading it with values returned from the database. We encountered two main errors during this part, and were able to resolve both as of submitting:

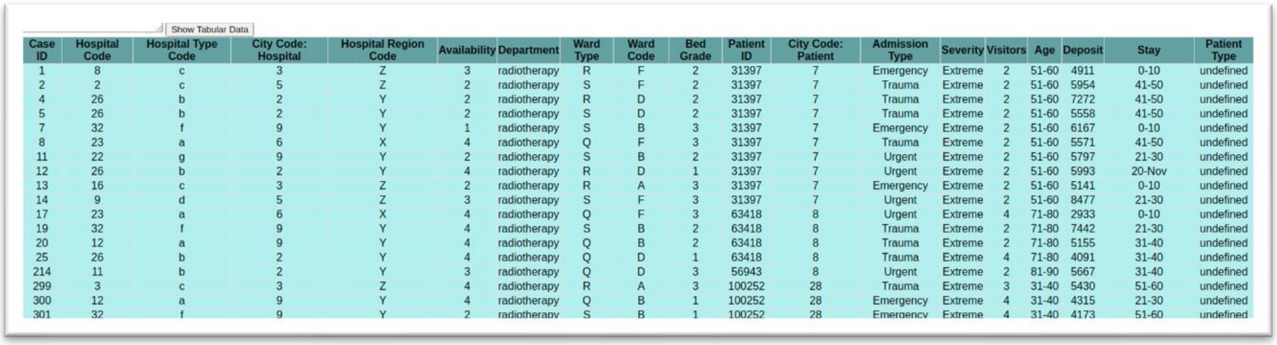
Error	Explanation	Solution
“Unexpected token ‘o’ in JSON at position 1”	This occurred as a result of trying to use the method “JSON.parse(res) on our returned array. Because we had already used JSON.stringify on the server-side, we were effectively trying to parse a file that was already in JSON format, and so it would not parse correctly.	Removed JSON.stringfiy from the server-side, and just passed the returned array as-is to the client.
Table.insertRow is not a function	This occurred while trying to generate the table in our HTML page. Researching the error led to the possibility that we wre unable to generate a table within a <p> element.	Manually structured the table in our HTML page, and referred to that “<table id=” value when generating our table.



Show Tabular Data

ıse ID	Hospital Code	Hospital Type Code	City Code: Hospital	Hospital Region Code	Availability	Department	Ward Type	Ward Code	Bed Grade	Patient ID	City Code: Patient	Admission Type	Severity	Visitors	Age	Deposit	Stay	Patient Type

put Template



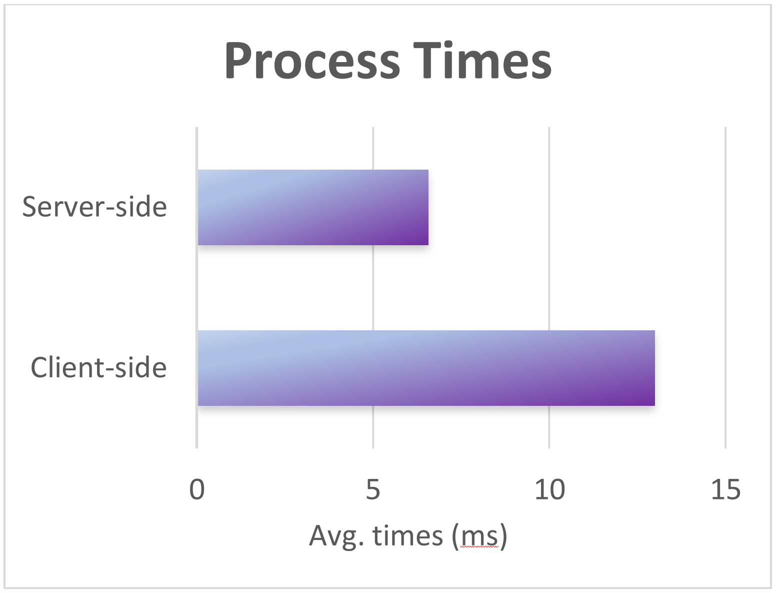
Show Tabular Data

Case ID	Hospital Code	Hospital Type Code	City Code: Hospital	Hospital Region Code	Availability	Department	Ward Type	Ward Code	Bed Grade	Patient ID	City Code: Patient	Admission Type	Severity	Visitors	Age	Deposit	Stay	Patient Type
1	8	c	3	Z	3	radiotherapy	R	F	2	31397	7	Emergency	Extreme	2	51-60	4911	0-10	undefined
2	2	c	5	Z	2	radiotherapy	S	F	2	31397	7	Trauma	Extreme	2	51-60	5954	41-50	undefined
4	26	b	2	Y	2	radiotherapy	R	D	2	31397	7	Trauma	Extreme	2	51-60	7272	41-50	undefined
5	26	b	2	Y	2	radiotherapy	S	D	2	31397	7	Trauma	Extreme	2	51-60	5558	41-50	undefined
7	32	f	9	Y	1	radiotherapy	S	B	3	31397	7	Emergency	Extreme	2	51-60	6167	0-10	undefined
8	23	a	6	X	4	radiotherapy	Q	F	3	31397	7	Trauma	Extreme	2	51-60	5571	41-50	undefined
11	22	g	9	Y	2	radiotherapy	S	B	2	31397	7	Urgent	Extreme	2	51-60	5797	21-30	undefined
12	26	b	2	Y	4	radiotherapy	R	D	1	31397	7	Urgent	Extreme	2	51-60	5993	20-Nov	undefined
13	16	c	3	Z	2	radiotherapy	R	A	3	31397	7	Emergency	Extreme	2	51-60	5141	0-10	undefined
14	9	d	5	Z	3	radiotherapy	S	F	3	31397	7	Urgent	Extreme	2	51-60	8477	21-30	undefined
17	23	a	6	X	4	radiotherapy	Q	F	3	63418	8	Urgent	Extreme	4	71-80	2933	0-10	undefined
19	32	f	9	Y	4	radiotherapy	S	B	2	63418	8	Trauma	Extreme	2	71-80	7442	21-30	undefined
20	12	a	9	Y	4	radiotherapy	Q	B	2	63418	8	Trauma	Extreme	2	71-80	5155	31-40	undefined
25	26	b	2	Y	4	radiotherapy	Q	D	1	63418	8	Trauma	Extreme	4	71-80	4091	31-40	undefined
214	11	b	2	Y	3	radiotherapy	Q	D	3	56943	8	Urgent	Extreme	2	81-90	5667	31-40	undefined
299	3	c	3	Z	4	radiotherapy	R	A	3	100252	28	Trauma	Extreme	3	31-40	5430	51-60	undefined
300	12	a	9	Y	4	radiotherapy	Q	B	1	100252	28	Emergency	Extreme	4	31-40	4315	21-30	undefined
301	32	f	9	Y	2	radiotherapy	S	B	1	100252	28	Emergency	Extreme	4	31-40	4173	51-60	undefined

Dashboard before and after btnTab is clicked, and table is generated

Part 5: Process Times

Summary: Overall, the server-side counting performed an average of twice as fast as the client-side method. Client-side had an average time of 12.999ms per run, while server-side showed results in an average of 6.573ms.

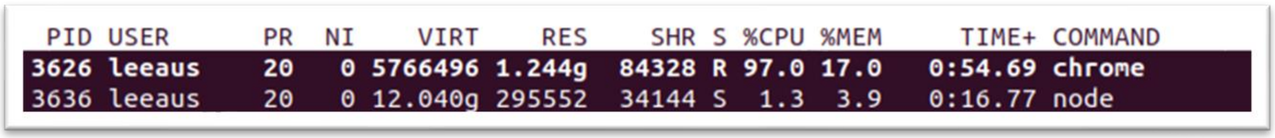


Analysis: We believe this is due to the implementation of the sorting methods. The client-side method must sort through an entire array, parse each entry into keys, and then perform a comparison operation on each record to determine if the value matches. During the server-side method, the program has direct access to the database. It can make both queries, and only has to return a number, rather than one or two arrays to the client-side.

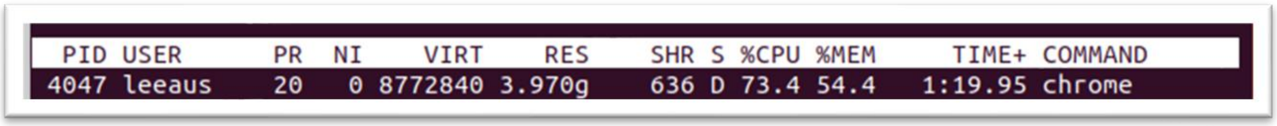
Overall, server-side processing seems to be much better than client-side. For this particular requirement, the efficiency is clearly greater. It is also much more secure to do this task on the server-side. The program only returns a number to the client, rather than an array containing all of the records (and personal information within). If we were performing multiple tasks, however, it may be more efficient to do more client-side processing, so as to avoid the latency between client/server, and avoid basically requesting a new dataset each time we want to perform an operation.

Part 6: Other Notes

Project part 4: As of submission last time, we were unable to get the tabular display of data working on our page. By using the site recommended for this project part, we were able to amend our earlier code very quickly and now have a functional method of displaying all records from a database. However, we did encounter one main error when attempting to load the entire database into this table:

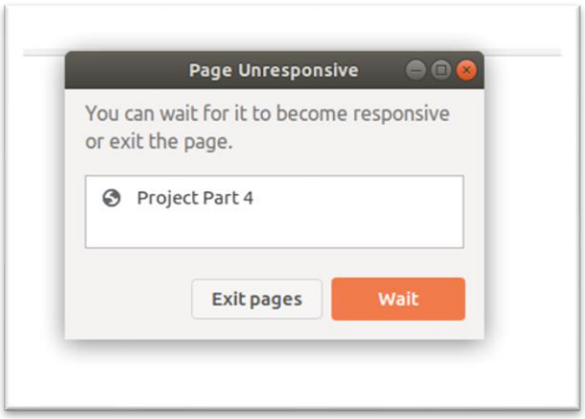


```
  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 3626 leeaus    20   0 5766496 1.244g  84328 R 97.0 17.0   0:54.69 chrome
 3636 leeaus    20   0 12.040g 295552  34144 S  1.3  3.9   0:16.77 node
```



```
  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 4047 leeaus    20   0 8772840 3.970g    636 D 73.4 54.4   1:19.95 chrome
```

The memory usage of Chrome beings to gradually rise, before capping out at around 54%. At this point, Chrome becomes altogether unresponsive:

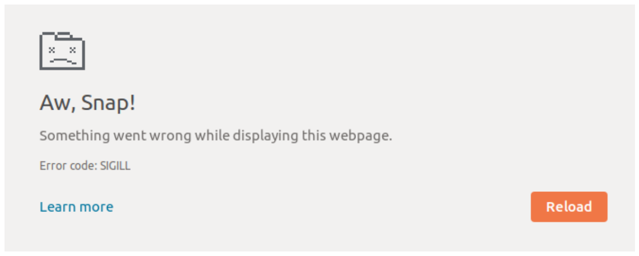


Page Unresponsive

You can wait for it to become responsive or exit the page.

Project Part 4

Before unceremoniously crashing:



Aw, Snap!

Something went wrong while displaying this webpage.

Error code: SIGILL

We believe this is simply due to the program running out of memory from trying to load so many records into the table, because we were able to generate the tabular display using a smaller sample database.