Ryan Coplien, Jack Rasmussen, Bailey Watkins, Michael Kelly  
CS 4230  
Project Part 5 Report

**Project Requirements**

Project part 5 consisted of the string processing in MongoDB and Javascript. We needed to add two templates to the program, once called QueryPanel, and the other called Dashboard. Within the QueryPanel template, there should be three buttons with different actions.

The first button should fetch all the records from MongoDB where the department is “radiotherapy” and then via the javascript side, count the number of records and also extract all records where the “Severity of Illness” is “Extreme”, and then return the count for those records. Then, the number will need to be displayed on the dashboard as a fraction of what amount of radiotherapy patients are filled with extreme cases.

The second button should do the same thing, but instead of depending on the javascript to handle the count and filtering, it should use mongoDB queries to count and filter all records. Then it will return the values into Javascript where it can be used to display the results on the dashboard.

Finally, the third button should create a table that will return all the records from MongoDB into an HTML table.

**Working on the Project**

To start this project part, we immediately ran into issues that occurred because of the previous project part where we could not get the project to properly compile on any of our machines. We were at a loss; however, we made the unanimous decision to move our code to a barebones system were we could properly expand on our project parts without the need for the Onstitute Education platform reliance.

We converted our project away from several HTML pages for the time being to better utilize the templates that we needed. The primary developers for this project part were Bailey Watkins and Jack Rasmussen. Their computer/VM specs are as follows:

* Bailey Watkins (Not using VM)
  + CPU: Intel i7-8700k
  + GPU: Nvidia GTX 1080
  + RAM: 16 GB DDR4 3000 Mhz
  + Drive: Kingston 2.5” Sata SSD
* Jack Rasmussen (Using VM)
  + Video Memory: 16 MB
  + Processors Allocated: 4 Cores
  + RAM Allocated: 6030 MB

Throughout the project, some stuff was easily reused. Such as parts of the commands in server (ClientSideCount and ServerSideCount) to get an array of results based on the query. Then using the given code in the project briefing to tabularize the array, and finally setting the array into the table within the html.

Additionally, we removed the iron router at the bottom of the page as there were no routes for the templates in the main.js client and a template in the main.html.

**Issues**

There were a few issues that occurred during development of the project part. To start, asynchronous functions where finishing faster depending on their complexity, and the DB query didn’t finish before the calculation happened which kept resulting in DIVIDE BY ZERO errors.

Additionally, issues that occurred included getting the table into a more legible format to prevent taking the entire page space and having to scroll for ages. This was fixed by putting the table into a divider and styling the divider. The only issue with this is that white space will hold the area until the template is called. This could possibly be fixed in future updates.

**Results**

The main objective of the project part was to find out how much time each process took. Here is a graph of the results:

|  |  |  |
| --- | --- | --- |
|  | Client-Side Processing | Server-Side Processing |
| Pros | * Data is readily-available for mass use in project as needed | * Function is built into the mongo software * Filters are much quicker to retrieve the data from * There is a higher level of security of data when processing through the server * Uses MongoDB optimization |
| Cons | * Functions to count and to filter need to be manually created through Javascript * Functions and filters will inherently be slower as all data needs to be pulled from MongoDB * Custom optimization is needed, and might never reach level of MongoDB * Lack of security as all data needs to be pulled from MongoDB | * Data will need to be retrieved again if used a lot |

**Conclusion**

Based on these results, it can be concluded that the time to processes the count and filter of records was much faster on the server-side processing method over MongoDB.