**INTRODUCTION**

This report will be detailing the completion of the entire project, the challenges faced during the implementation of Task 3, and how they were overcome and sharing the insights into the key learnings acquired throughout the project.

**COMPLETION OF PROJECT**

From part 1, up until this point in part 3, I always attempted to complete each part in sprints. But before that was possible, I would plan for which day I wanted to begin the part for programming. I would measure the number of features and quantify in my head in terms of knowledgeability, ability to implement and time, mainly. For each feature in part 1, for example, I already had an idea of how to go about implementing what I needed to implement after reading the instructions. I would usually implement a feature/aspect of the application a day. I usually try and “prepare” the back end by dedicating a part of the day to making the UI that I think I would need to complete what was required of the certain part as well. Starting with the UI helps with getting a more refined idea of what I was going to do in the back-end as well.

After quantifying it, I would select a date that was well before the due date to begin working on it. The date is very inconsistent, especially this semester, considering the load of work that we need to complete it. Usually, a week and a half before the due date was what I settled on. The reason I picked an amount of time like that is because I would do a sprint every day from the date that I’d chosen. In the case where I run into a problem that I’m having a hard time solving, I’d be able to have enough days to be on the safe side of things. For example, figuring out the binary Search Tree did take quite a while and that’s why the “extra days” I’d assigned for myself were useful in that case. The extra days are also useful to implement additional error handling that I hadn’t discovered in testing on previous days.

**CHALLANGES IN PART 3**

The main challenge that I encountered in part 3 was the use of data structures that I selected to use for this part. After some research and comparison, I decided to the Binary Search Tree and Heap data structures to complete the tasks. Visual Studios didn’t have any of these built in the .Net Framework version I was using, so I decided to make my own one that would fit the use case. This was hard and challenging. I understood how you were meant to iterate through a Binary Search Tree when speaking about it in theory, but to implement via code was a challenge that I sort of saw coming. The constant nesting of nodes kept disturbing my thought process, but I tried to take it one step at a time by making different versions (and doing some research as well for already existing binary search trees) of the binary search tree until I made an authentic binary search tree. That’s why my Binary Search Tree class seems a little too empty for a typical class, it just has what it needed to get the task done.

Now the heap was a little simpler to implement, I did struggle just a little bit with the Swap function that is in the class, but it works, nonetheless.

**ACQUIRED KEY LEARNINGS**

With the entire POE complete, I have learnt some new key skills and lessons that I think are important as a programmer.

* Data structures and algorithms are important: In a real-world scenario, using a normal list or LinkedList data structure to get and sort data is not going to cut it. Companies usually deal with hundreds and thousands of entries of data into a database. Problem-solving is important here because what works in one case wouldn’t work as best in the other. For example, using a normal list for a family friend who wants to have a digital calendar could be fine, but a company asking you to sort through thousands of data to find a single user’s information with a normal list won’t be as efficient as another data structure that exists here.
* Approaching programming with a more attention to detail: Every time I make an application of some sort; I’m now thinking of what the best data structure would be to use to hold the data would be and what would be the best algorithm to sort through that data.
* Research is crucial: As a programmer, I have to admit that researching what you need for your application is, in itself, a skill. To be able to phrase things in a way that Google interprets in the way you want it to so that you can receive the results you want is important. My skills in that have improved and I’ve grown attached to some popular programming sites because of how they explain certain concepts and visualize solutions.
* To stay calm: The world of programming alone in IT is very overwhelming. There are hundreds of techniques, there are thousands of terminologies that exist within the space and the worst/best thing about programming is that it is constantly growing. I’m still new to this, and even though I’ve been passing with distinctions, there's still a lot to learn and it's important to take it one step at a time. I felt that way learning about data structures because there are so many types of data structures, and we need to be able to construct them ourselves.
* Refined my skills in the use of LINQ: In part 2, I’ve learnt more about the flexibilities of LINQ and the way to use it. I have improved with my use for it.

**CONCLUSION**

Therefore, this report has detailed the completion of the entire project, the challenges faced during the implementation of Task 3, and how they were overcome and shared the insights into the key learnings acquired throughout the project.