Mattias Wickberg, 1dv600

Personal reflections on Project 1dv600

#Assignment 1

Task 1

Subtask A

The most complicated aspect of this task was really that I don't understand what the book object in the dao folder should be? In the end, I just made a file there and went on to create the book list in GetBooksResource. Again, I made the mistake of adding the book list to the callback to make it show on the website instead of console.log, so had to retrace and log it to the terminal instead. As always, the hardest part of any task is usually to actually understand what the task giver is saying/writing. This is something I try to remind myself of every day as a teacher, because it's often the most challenging task for both me and my students.

Subtask B

Converting to JSON is the work of a few minutes, so not much problem with the work this time.

Improvements: Firstly having a list of books in a function is really not a workable strategy for a site where you'll need to add new books through an interface, so first of all I need to separate my books from my functions, and then require that json file into my function. This is probably not good enough for the final version, but it's better than what's there now.

Secondly, I need to comment my code so that I know what I'm doing, and add more details to my books.

Subtask C

Really didn't need much time for this part, and I think maybe I should have come up with more complex improvements - however, without being quite sure what the requirements for the system is, I don't want to rush ahead with something major before I have implemented the basics thoroughly. I realise that my json file is very transitory, as there will be xml in the works, but still, it bugged me having a messy function.

Task 2

The difficulty with writing a vision is that I still have no clear idea of how complex this system can be during these few weeks, or whether to see as a library system for a full-fledged library, as a small scale version mostly used at home, or as a system for a smallish bookshop. They all have different demands. I went with the real library-angle, since I'm guessing that's what the basics of the system was intended for, and it may be difficult to work against what the course idea is. In reality it appeals more to build a system for my home or my school, sicne we don't really have one at the moment, but generally speaking it's better to avoid doing things you actually need in university assignments - they need to meet the requirements of the course first, and your own needs second.

Task 3

The project planning was mostly just a nice was of doing something with a bit more structure than I probably would have otherwise. Planning has been a crucial part of my life for many years now, juggling a complex work, studies and home life. The risk analysis was one part that I wouldn't have put into words had this not been part of the course literature, but it did help me focus on what the real risks are. This of course being a smallish project where I work alone rather than a large project with many people involved, it is a radically more simple plan than the ones I do work with for a living. Time wise I did overestimate the time it would take to do some parts of the first assignment, but the stimates were based on me usually being interrupted while I work, and this last two days I've had perfect calm to focus on this course!

#Assignment 2

##Task 1 ###Subtask A I have encountered UML-diagrams previously, and I do find them very useful in the planning stage for development. Use cases are of course very similar to ways that I've worked in Interaction Design, since I have a lot more experience from that field than programming, and so they do feel quite natural to have here as well. In general, I think the difficult part here is to decide on the scope of the planning stages of this project, knowing that a full scope library system would need more functionality than we can really make during a course like this.

Subtask B

Robustness diagrams in this very case seem a bit overkill for documentation and planning, but I can definitely see the usefulness of it to provide a link between the slightly abstract use cases to the actual implementation of the functionality. Especially for a larger sclae product where the people implementing and the ones designing the system may not be the same people or even in the same location, this seems to provide a good basis for clear technical communication of how the different steps should work.

Subtask C

Sequence diagrams, again feel quite useful. But like with subtask B I'm starting to wonder how many different diagrams are really useful for this fairly limited project. For this kind of project, I do feel that I would do use cases and either sequence diagrams or robustness diagrams, but to add more would start to feel a bit redundant. Again though, I can see that the story is different for larger scale projects. After doing the sequence diagrams I feel a have a better grasp of the system architecture.

##Task 2 This way of doing things does really force me to think through thigs before I do the coding. Of course, it also means I can't test and see if I'm mistaken in some assumption of how things work - but then if I do find out in the end that I made a planning mistake I guess I can go back and redo the sequence diagram. It probably reduces the amount of errors in the end anyways. It is akin to what I tell my students about essay writing (my profession is language teaching), that is is a process that is more about planning than actual writing if you want the end result to be of good quality and reduce the number of errors and unnecessary work you do.

Task 3

Ok so this one took a lot longer than I thought, possibly pasrtly because I started it while I was at the

students' LAN party at work, and was quite tired and unfocused, but I really got tangled in how to transform the xml to json. It really should have been ten or twenty minutes work, but it took me two hours to figure that part out. The rest was really quite easy. Since the only real tangle here was the xml, I don't really feel that there was a huge difference between the designed and undesigned approach to implementing a function - the delete function was so much faster and easier to implement, because I had figured out the xml and understood the flow between different functions. I think the lessons I take away from this assignment is:

- What takes time is rarely what you think will take time - it's true about a lot of things in life.
- Converting information between formats is important, but can be time consuming (my partner who works as a bioinfomatician says that this is mostly what working as a scientist is about).
- Planning the flow between functions and what information travels is important, but planning the overall architecture is more important than the details.