**Task 5 – Closing Review**

**A & B)**

I was able to use the Project Charter to guide my design and development activities so that I didn’t lose focus of what I was trying to achieve. It stopped me going out of scope and building other functionality that I thought would have been nice to have, but on further reflection did not align to the original requirements identified in the Project Charter. It was really good to have a clearly defined scope that I could refer back to, and the specified success criteria meant I was able to quickly see which parts of the project had met the requirements, as well as the areas that still needed work. During each phase of the project I was able to use the Project Charter as a sort of checklist to make sure that what I had developed matched with the needs expressed in the user stories, which helped me to confirm whether or not I was on track.

The Quality Management Plan showed me what a successful application looked like and helped me to remember why I had made key design decisions early on in the project. This allowed me to avoid a few potential pitfalls as I moved through the development phase to implement the design. The Quality Management Plan also helped me a lot when it came time to write the developer tests. It spelled out exactly what I needed to cover off with my test cases and spelled out the types of the testing, verification and validation tasks that I needed to complete as the project progressed. This gave me a roadmap so I could easily see which task(s) needed to be completed next.

By following the Quality Management Plan I was forced to implement a requirements traceability matrix right from the start. That traceability made me feel accountable for each method I wrote because I knew I would have to justify its existence later on, so it prevented me from adding superfluous features on more than one occasion. By continuously coming back to the Project Charter I was further reminded of what was in and out of scope and I found I had forgotten to build a few small pieces of the required software package which I was then able to rectify relatively easily.

In future I would probably try to spend a bit more time writing out the user stories because I found some of them were a bit generic. There were a few aspects of the requirements that were not well explained in the user stories which made it more difficult to write tests that aligned to an existing user story while ensuring I had built all of the required functionality.

There were a few occasions where I found that my understanding of the design had changed slightly and some parts of the plan just didn’t quite fit how I wanted them to anymore. I was tempted to re-write some areas of the Project Charter and Quality Management Plan, however I think it was important to maintain them as they are. Those documents acted as a compass as I navigated the application development, and if they did not remain constant then it would be very easy to keep changing the project to a point where it is no longer recognizable, in which case the original objectives may not have been met.

**C)**

One change I would like to make to my software development practices in future would be to collaborate with others instead of developing in isolation. I believe that discussing design and implementation considerations with other developers throughout the project would enable me to more easily identify areas of the project that could be improved, avoid pitfalls that I may not have foreseen on my own, and it would provide a sounding board to help me figure out whether my intended approach is going to be practical or not before I spent hours writing the code.

Engaging in peer code reviews would be another great way to figure out which areas of the project are lacking. Having a second person look at the code from their own perspective would help to identify areas in the code base that have been created in an overly complicated way, which could assist with refactoring to simplify and make the code more readable and extensible. Having someone peer-review my code would also potentially help identify bugs that I might have missed on my own. If the person is more experienced than me, they may also be able to offer tips to improve my code or the application I am building, which would be a good way for me to upskill.

Another change I would make to my development practice would be to make more of an effort to identify areas where I can reuse code, rather than writing the same thing more than once. I think this is something I should be considering earlier in the development process, otherwise I end up building a unit only to discover I will now need to re-build the same thing again later for another part of the project. By thinking this through earlier in the process I could potentially change the way I implement it in the first place to make it reusable for both parts. In this project for example, I created a separate dialog box for each of the add task, view task, and edit task functions on the listView screen, then I re-created those same dialog boxes for the CalendarView screen. This meant I had to spend significantly more time in the testing phase as it meant I had to write twice as many test cases to cover both implementations.

**D)**

Reflection such as that outlined above helps to identify the lessons that can be learned from a project so that the next time similar issues arise on another project I can take what I have learned and use it to avoid making the same mistakes twice. By taking the time to think through the good, the bad and the ugly parts of the current project, I can apply the ‘so what’ principle to identify the implications of my decisions. This will allow me to more accurately analyse the potential implications associated with planning, design, logistical, or implementation decisions that I will need to make on my next project, thus (hopefully) increasing my chances of achieving successful outcomes.

**E)**

One way this software could evolve would be for a client-server approach to be implemented. If the database was moved to a centralized location then data could be accessed through an API. This would make the application data accessible from any of the devices on the network that have server access (e.g. the user could access their to-do list from any of the computers in their office). If a cloud-based server was used then the data could potentially be accessed from anywhere and different types of devices could be utilised to run the application, such as phones, tablets, laptops or desktop computers.

Another potential evolution would be for multiple users to have concurrent access to the to-do list so that it could be used to manage tasks in a team setting, for example as a project management tool. For this to be achieved user profiles would need to be created and stored somewhere, and login credentials would need to be securely managed.

These evolutions would require some kind of authentication to prevent unauthorized data access and ensure the user is able to access their data securely. Ongoing maintenance and security patches would be required to maintain a secure environment.

The issue of concurrent database transactions would also need to be managed (e.g. what happens if data is modified on one device and at the same time a user on a second device tries to access or modify the same data). Another consideration would be the development and implementation of new views to better accommodate different screen sizes (such as a mobile phone screen vs a desktop monitor). The database may also need to be refactored to keep track of when changes were made and who made them.

As new versions of the software become available it would need to be tested thoroughly before being released in a controlled manner. Considering that the application would potentially be used on different devices and operating systems, further testing would need to be completed in those environments to verify the software runs correctly and functions as it should. This means ongoing verification and validation procedures would need to be established, and a release plan would need to be developed and implemented prior to any new distributions of the software being made available. Supporting documentation would also be required to ensure uses are aware of any new features.