**Milestone 2 Narrative**

**Briefly describe the artifact. What is it? When was it created?**

The project was initially used for Unit Testing in Java, and utilized a design structure of Services that communicated and managed base objects (Contacts and Appointments). I believe I initially wrote it early last year. The Services could be called to (Singleton objects) that could create, update, delete, and recall their managed objects. No base entry point was established in the code to make it usable to a client though. I chose to port it from Java to CPP and expand its functionality.

**Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

I chose the artifact because I wanted to have a deeper dive into both the porting process from a higher-level language (Java) to a lower-level language (C++), as well as further explore the services design.

The entire project showcases my ability to port code from one language to another, as well as design around one language’s strengths and the other’s weaknesses. This could be from Garbage Collection to data types/structures not existing between the two, teaching myself header files, since Java doesn’t use them, as well as relearning references and pointers.

Since I decided to add entry point code, which included a menu system for clients to use the software, I had to design a system to manage the menu traversal and calling to their respective services.

**Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

I think I met most of my objectives. I have most of the code ported over, and expanded (I added more functions to the C++ code). I have 2 existing bugs that I cannot figure out:

Appointment Deletion – I use vectors to store the Appointment list, when I run appointmentList.erase(appointment), I get an error at compile-time about a reference to a deleted function, when the code is the exact code I use for contacts (same data structure), so not sure what’s up there. I do delete certain methods to enable the service to work as a singleton, but even commenting those out and recompiling produce the error (and those same methods are deleted in the header for the ContactService class, so I don’t think that’s the issue, and nothing is const).

Appointment Rescheduling – The user input properly converts to a tm struct, then to a time\_point, and is passed to the Appointment class properly, it retains the data, and if the output, is correct, but the existing time\_point on the class will not change to the new one. The Java code used the Date datatype, and from all my research, C++’s closest approximation is the chrono::system\_clock::time\_point, but I cannot get the existing value to overwrite.

**Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

I have learned a lot, I was never an experienced C++ developer, and I know Java/C# well, but C++ is a completely different beast. I took the challenge of C++, because I wanted something more difficult, and wanted to get practice before touching my DSA milestone, where the codebase will be heavily C++ and OpenGL code.

I had to teach myself references, pointers, and header files.

I had to design around things that are so common in more recent languages, that I am surprised C++ didn’t have (to be fair, I developed with C++ 14, and some of this does exist in C++ 20). Error messages can be cryptic in C++, and lead to hours of head scratching.

I honestly thought that porting code from Java to C++ would have been easy and fast, but taking a 4-5 file Java project, with less than 500 lines into Java turned out to be a week-long endeavor, and that was before I touched the Main Menu that I planned to add.

Aside from the 2 bugs that have me completely stumped, I really enjoyed this process and getting to take something from Java and convert it to C++ where it could be far more modular, faster, and portable to far more devices.