The artifact selected for the Software Design and Engineering example was the IT-145 Zoo Authentication system. This artifact was selected as it was a very early example of my work and was one I had always thought could have been done better. This served to act as a prime example of how I’ve grown as a computer scientist during my time at SNHU. The initial critique I’ve always had with that code was that it was far too deeply nested into while loops and if statements. Taking the opportunity now to re-engineer it I started with pseudocode like I did back then. Now with a better handle on how to apply loops and use Booleans I managed to go from six layers deep into loops down to three. Cutting the over all width down significantly simplified the operation. This definitely served to meet the criteria of improving the original software as well as expand the projects complexity by slimming down and better using the variables.

The artifact selected for Algorithms and Data Structures was the CS-260 eBid assignment. The original was over-commented and all contained in one file. This needed a refactoring in order to make it reusable as well as to clean up the main file. The main trimmed nearly 300 lines of code out and other than some compiler issues was a successful transformation. Once everything was move to where it needed to be the compiler was throwing some errors that it certainly didn’t do when the code was initially written. Once I sorted out those I was left with 3 files and no errors. The header file contains what it needs to function, the main file contains the header file, it all went swimmingly. Then I turned my attention to the comments. The description at the top still read hello world and there were FIXME’s and other things that were indicative of being a school submission. Trimming it down to just what it needed to identify the sections also clean up the code dramatically. The efficiency of the code was tremendously improved and by involving more than just the one thing served to enhance the complexity while keeping intact the core functions.

The last artifact is for the Database category and for that I worked with the CS-340 Mongo database. The original was on Linux, with a local database, and the functions were written in Python. I took it from there and re-engineered it to work on Windows, with a web based Mongo database and with the functions written in JavaScript. When I created the code initially I had always wanted to include more error handling. I wanted to provide the user with more feedback when things went right or wrong. So after starting with getting the database ready I turned my attention to simply make a JavaScript file which I could define the CRUD functions I had previously made in Python. Once I had them working consistently I started building IF/ELSE IF/ ELSE statements into those functions. This would check if you passed any arguments to the functions. If you didn’t, it would give you feedback stating such. With create it would look to see if something already matched in the database. This was a lesson learned from the original course when I was adding the same thing over and over and had multiple entries but with unique Mongo “\_id” fields. The read function does more than just return a blank line if it fails to find your query, it comes back saying that no entry was found. This is something else from the original class I had noticed and wanted to make sure I included. Updated and delete work similarly in that they give you feedback on whether or not the entry was found as well as if the function carried out successfully. The sample database I used was a list of AirBNBs and each one contained many reviews. So I changed the read function to also allow the query filter to pass in and allow you to filter out the review documents. Not only does these functions finally work as well as I had hoped them to be, I completely changed the operating system I was working in as well as the language I used to make the functions and the location of the database. I also created a batch file to launch the shell as opposed to continually have to copy a huge string to copy into the command prompt, which was something I tried to figure out with Linux but I was spinning my tires and didn’t want to waste time in the class. Now given the opportunity to make improvements I was elated to do it.