Quintin B. Rozelle

6/10/2025

Professional Self-Assessment

CS-499

I have always had an interest in computers and programming, though didn’t seriously explore these areas until starting this degree at SNHU. Prior to this, I had dabbled a bit with basic HTML and CSS and taught myself to use Excel to a highly proficient level, thinking that this would satisfy my curiosity for computers and programming. While the latter is not overly related to the field of computer science, I would argue that there are minor elements of it which are. Instead of sating my curiosity, this experience only piqued my interest and caused a desire to learn more which this computer science program has SNHU has been able to provide. I have thoroughly enjoyed each and every course of this degree and have taken it as an opportunity to learn about the various aspects of computer science while growing my skills in them. I consider myself to be a lifelong learner and SNHU has given me the knowledge necessary to enter into the field of computer science while also providing me a very solid foundation to build upon over my career. Through this degree I have gained knowledge and developed my strengths in a number of areas.

**Data Structures and Algorithms**

Appropriately using data structures and algorithms is essential to creating a well-functioning product. Using the incorrect data structure or creating a less-than-optimal algorithm can severely hamper the end-user’s experience. CS-300 (Data Structures and Algorithms) gave me the knowledge necessary to apply the most commonly used data structures. Based on this, I have the ability to create classes and objects that adhere to the needs of and work well with a data structure. Enhancements one and two for this portfolio are good examples of this in which I have created a Bid class that works well with two different types of binary search trees.

Another example comes early in this degree from IT-145 (Foundations in App Development). One of the assignments in this course was to build an intake system for rescue animals. As the total number of animals to store is unknown at runtime though likely relatively small, an ArrayList is a good choice:



This was easy to use to store prepopulated data and eventually to search to data on intake to prevent duplicate entries:

A black background with colorful text

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

**Software Engineering and Databases**

Software engineering and database use are also essential to a well-functioning program. Without using either appropriately, the program could lose data or malfunction. My concentration for this degree is Software Engineering so many of my courses incorporated this. A great example of this comes from CS-465 (Full Stack Development) in which I built MEAN stack website. A strength of this project was the separation and encapsulation of components into their own files. This allowed for easier development as each file was focused and simple to create and debug and makes the whole project easier to understand. This project created a full functioning end-user facing website with an admin focused backend for easing the management of data displayed on the front end. Both interact with a MongoDB through the use of a custom-built API.

In addition to the use of a MongoDB for this project, I have grown my abilities to use databases through DAD-220 (Introduction to Structured Database Environments) and CS-340 (Client-Server Development) in which I learned to use relational and document-based databases respectively. An example of my ability to use and interact with databases comes from CS-340 in which I built a CRUD API to handle the management of documents within a MongoDB:

A screenshot of a computer program

AI-generated content may be incorrect.

**Security**

Similar to the above two sections, software security is paramount to software development as it prevents both unintentional software crashes and intentional breaches. My skills in this area come from CS-305 (Software Security) and CS-405 (Secure Coding). In the former, I learned to use dependency checking tools to check for known security vulnerabilities and cryptography techniques to allow for transfer of sensitive information:

A screen shot of a computer program

AI-generated content may be incorrect.

In the latter class, I gained the ability to incorporate secure coding practices into my code to prevent security issues. Examples of this are the prevention of buffer overflow and SQL injection. In the code below, buffer overflow is prevented by using the getline function to read into an array no more than the size of that array:

A computer screen with text and images

AI-generated content may be incorrect.

SQL injection is prevented in the following code example by using a regex to search user input for SQL injection attempts:

A screenshot of a computer program

AI-generated content may be incorrect.

**Collaborating in a team environment**

In addition to the hard skills listed above, a successful developer needs to be proficient in some of the softer skills as well. Since most developers do not work alone, being able to work well on a team is vital. CS-250 (Software Development Lifecycle) gave me the skills to work with teams that use the Agile framework. In this course, I simulated working on a team like this through written assignments and a group project/discussion. I also created an Agile team charter that could be used in a theoretical team:

A document with text on it

AI-generated content may be incorrect.

**Communicating with stakeholders**

In addition to collaborating with a team, a developer must also communicate and collaborate with stakeholders. CS-250 and CS-319 (UI/UX Design and Development) provided me with the opportunity to grow these skills. As an example of this, in CS-250, I created multiple User Stories for a project to put myself in the end-user’s shoes to help ensure I was creating something they needed instead of something I thought they needed:

A screenshot of a computer

AI-generated content may be incorrect.

As another example, in CS-319 I conducted several interviews with potential user’s for a mobile app I was creating:

A questionnaire with text on it

AI-generated content may be incorrect.

Both cases showcase my ability to work with stakeholders to improve the product that I provide.

**Artifact Summary**

The enhancements for this portfolio are all applied to the same artifact. This artifact comes from CS-300 (Data Structures and Algorithms). In it, I built a tool for reading auction information from a CSV file, adding that information to Bid objects, and storing those Bids in a binary search tree. This tool would then be able to display all information, display a specific record or delete a record from the BST.

Enhancement One takes the original artifact and converts it from C++ to Python. This showcases my skills in software development through my ability to convert software from one language to another while understanding the nuances of both. In addition to this, I improved the documentation by adding well written comments where necessary and including docstrings for every class and function to help improve understanding for another developer who would be coming into my code blind. Security was improved though the use of data validation and exception handling. Lastly, software stability was ensured through the implementation of unit testing.

Enhancement Two takes Enhancement One a step further by improving the data structure and algorithms used. Enhancement One retained the use of a standard binary search tree from the original artifact, but this enhancement converts that to the more advanced Red-Black Tree. This ensures that the algorithms run at a guaranteed time complexity of O(log n) instead of the potential for a time complexity of O(n) that standard binary search trees can have in certain situations (e.g., presorted input data, etc.). This enhancement retains and continues to build upon the documentation, security, and stability that Enhancement One introduced.

Enhancement Three further improves upon the updates from Enhancement Two by implementing the use of a database to store the information between sessions. Since the imported data was stored in a data structure stored in memory, as soon as the program closes that information is lost. By using a database (in this case a SQLite database), that information is no longer volatile. To ease the use of and interaction with the database, a CRUD API was built which further helps to clean up the code and make it more readable. As with the other enhancements, the documentation, security, and stability improvements are maintained.