Quintin B. Rozelle
6/10/2025
Professional Self-Assessment
CS-499

I have always had an interest in computers and programming, though I 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:

```
private static ArrayList<Dog> dogList = new ArrayList<Dog>();
```

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

```
// Adds dogs to a list for testing
public static void initializeDogList() {
    Dog dog1 = new Dog("Spot", "German Shepherd", "male", 1, 25.6, "05-12-2019", "United States", "intake", false, "United States");
    Dog dog2 = new Dog("Rex", "Great Dane", "male", 3, 35.2, "02-03-2020", "United States", "Phase I", false, "United States");
    Dog dog3 = new Dog("Bella", "Chihuahua", "female", 4, 25.6, "12-12-2019", "Canada", "in service", true, "Canada");
    Dog dog4 = new Dog("Lady", "Basset", "female", 6, 55.6, "01-02-2018", "United States", "in service", false, "United States");
    Dog dog5 = new Dog("Hank", "Yellow Lab", "male", 4, 70.2, "10-22-2019", "Canada", "in service", false, "Canada");

    dogList.add(dog1);
    dogList.add(dog2);
    dogList.add(dog3);
    dogList.add(dog4);
    dogList.add(dog5);
}
```

```
public static void intakeNewDog(Scanner scanner) {
    String name = getUserString(scanner, "What is the dog's name?");
    // Check for dog already in system
    for(Dog dog: dogList) {
        if(dog.getName().equalsIgnoreCase(name)) {
            System.out.println("\nThis dog is already in our system\n");
            return; //returns to menu
        }
    }
```

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:

```python
# Create new document and add to database
# data: new document to add to database. Needs to be a Python dictionary
# returns: boolean; true if successful, false if failure
def create(self, data: dict) -> bool:
    try:
        if data is not None:
            insertSuccess = self.database.animals.insert_one(data)
            return insertSuccess.acknowledged
        else:
            raise Exception('Nothing to save, because data parameter is empty')
    except Exception as e:
        print(repr(e))
        return False


# Searches for and returns list of documents from database
# data: document to search for in database. Needs to be a Python dictionary
# returns: list of documents found
def read(self, data: dict) -> list:
    try:
        if data is not None:
            return list(self.database.animals.find(data))
        else:
            raise Exception('Nothing to search for, because data parameter is empty')
    except Exception as e:
        print(repr(e))
        return list()


# Search for and update document(s) in the database
# searchData: document to search for in database. Needs to be a Python dictionary
# updateData: key/value pairs to update in found documents. Needs to be a Python dictionary
# returns: the number of documents updated
def update(self, searchData: dict, updateData: dict) -> int:
    try:
        if searchData is not None or updateData is not None:
            return self.database.animals.update_many(searchData, updateData).modified_count
        else:
            raise Exception('Nothing to search for, because searchData or updataData parameter is empty')
    except Exception as e:
        print(repr(e))
        return 0


# Deletes document(s) in the database
# data: document to search for in database. Needs to be a Python dictionary
# returns: the number of documents deleted
def delete(self, data: dict) -> int:
    try:
        if data is not None:
            return self.database.animals.delete_many(data).deleted_count
        else:
            raise Exception('Nothing to search for, because data parameter is empty')
    except Exception as e:
        print(repr(e))
        return 0
```

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:

```
class ServerController{

    /**
     * Function to convert byte array to hexadecimal.
     * <p>Found at: https://stackoverflow.com/questions/9655181/how-to-convert-a-byte-array-to-a-hex-string-in-java
     * @param bytes  The byte array to convert to hexadecimal
     * @return  String containing the converted byte array
     */
    public static String bytesToHex(byte[] bytes) {
        final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }

    @RequestMapping("/hash")
    public String myHash() throws NoSuchAlgorithmException{
        String data = "Quintin Rozelle's unique data string";

        //New message digest that creates hash
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] hash = md.digest(data.getBytes());

        //Convert the hash to hexadecimal
        String hexHash = bytesToHex(hash);

        return "<p>data: "+data+"<p>Name of Cipher Algorithm Used: SHA-256<p>CheckSum Value: "+hexHash;
    }
}
```

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:

```
//char user_input[20];
const std::string account_number = "CharlieBrown42";
char user_input[20];
std::cout << "Enter a value: ";

// Added call to getline which terminates the character extraction once a certain number of characters are reached.
// The second argument calculates the total number of elements assigned to the array. This prevents issues with
// extracting the wrong number of characters from cin if the code is changed to define user_input with a different
// number of elements in the array
std::cin.getline(user_input, sizeof(user_input)/sizeof(*user_input));

std::cout << "You entered: " << user_input << std::endl;
std::cout << "Account Number = " << account_number << std::endl;
```

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

```
bool run_query(sqlite3* db, const std::string& sql, std::vector< user_record >& records)
{
  // TODO: Fix this method to fail and display an error if there is a suspected SQL Injection
  //  NOTE: You cannot just flag 1=1 as an error, since 2=2 will work just as well. You need
  //  something more generic

  // clear any prior results
  records.clear();

  // Uses a regular expression to search for the presence of " or *** = ***" where *** is any valid SQL value and "or" is case insensitive
  std::regex sqlInjectionSearch(" [Oo][Rr] .*=.*");
  if (std::regex_search(sql, sqlInjectionSearch))
  {
      // SQL injection attempt found. Print notification to console and fail search
      std::cout << "Attempted SQL injection identified. Aborting search" << std::endl;
      return false;
  }

  char* error_message;
  if(sqlite3_exec(db, sql.c_str(), callback, &records, &error_message) != SQLITE_OK)
  {
    std::cout << "Data failed to be queried from USERS table. ERROR = " << error_message << std::endl;
    sqlite3_free(error_message);
    return false;
  }

  return true;
}
```

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:

Item	Response
Business Case/Vision (value to attain)	SNHU Travel is in need of a functioning website that allows users to search for and book travel deals.
Mission Statement (result to accomplish)	To create a website for SNHU Travel that allows users to book their own adventures with the goals of making the process simple for users while allowing SNHU Travel to retain their foothold as a top travel agency.
Project Team (team members and roles)	Project Owner • Give direction to team and make decisions for the project • Prioritize work • Represent the business sponsor • Ensure the best value of the product and work of the team Scrum Master • Remove barriers that prevent the team from doing their best work • Facilitate and lead Scrum events (e.g., Daily Standup, Scrum Review, Scrum Retrospective, etc.) • Coaching the team • Effectively manage product backlog Developer • Designing and writing the code for the product • Produce a potentially releasable product at the end of every sprint • Be responsible for own work Tester • Determine acceptance criteria • Test the product against those criteria • Work with team to resolve issues
Success Criteria	Start date: 10/30/2022 Expected completion date: 5 weeks from start date (approximately 12/5/2022) Final deliverable: Functioning travel website that allows users to search for travel deals and book them Key project objectives: • Attractive and easy to use website • Secure website that allows for financial transactions • Responsive and quick so users are certain they have booked the package they want
Key Project Risks	• Not finishing on time or not producing a product that meets the client's needs • Overly cumbersome product that is difficult for the end user

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 improve 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:

User Story Number:	2
User Story Name:	Top travel based on history or profile
User Story Size:	Large
User Story Value Stater	As an End User, I want to see top travel options based on prior travel or my profile so that I can choose between popular locations that I would likely want to travel to.

Acceptance Criteria:	• A link that takes user to a page displaying top destinations tailored to them
	• Ordered list of tailored locations
	• Each item on the list will have:
	  • Name
	  • Short description
	  • Picture
	  • Link to package details

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

Questions

1. Can you describe your current process for monitoring soil moisture and nitrogen levels?
2. Is there anything in your current process that is lacking?
3. How would you feel about being able to see current soil moisture and nitrogen levels? Both equally?
4. How would you feel about being able to see historical soil moisture and nitrogen levels? Both equally?
5. Would it help to see current soil moisture and nitrogen levels plotted over the whole field instead of just individual trees? Both equally?
6. Would it help to see historical soil moisture and nitrogen levels plotted over the whole field instead of just individual trees? Both equally?
7. Would you need this app to tie into other services (e.g., automated irrigation systems, etc.)?
8. Would notifications from the app be useful?
9. How would you expect an app like this too look (e.g., color scheme, playful vs professional, etc.)?
10. Anything else that you would like to add?

Michelle

1. Sticking finger in dirt. Look at leaves for health
2. Accuracy, consistency
3. Very helpful, take out guesswork. Moisture more important
4. Yes, planning planting time frames. Both equally important
5. Yes. Helps with finding issues (e.g., water leaks or irrigation line plugs)
6. Most definitely.
7. Yes. Could almost "AI" itself
8. Yes. Need to be able to set ranges for when notifications are triggered
9. Need a plot/map. Different themes would be nice
10. Weather integration/forecast. Insect infestation monitoring. Historical logging of fertilizing/watering

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.

Course Outcomes

Through the completion of this self-assessment and the code enhancements, I have successfully achieved the outcomes for this course. Each narrative provides an update to the

completion of each outcome up to that point, but this serves as a summary of the status of each at the end of this course.

1. Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.
    - I have achieved this through my documentation in the code and the creation of an API. Each enhancement includes well written comments throughout the code to describe its functionality. Additionally, each function includes a complete docstring indicating its purpose, its parameters, and its output. Lastly, the CRUD API developed for enhancement three provides an easy and consistent way to interact with the SQLite database. These help to make the code more clear while also providing the documentation needed for other developers to use and build off my code.
2. Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
    - I have achieved this through the narratives, code review, completed ePortfolio, and this professional self-assessment. All combine to show my ability to provide written, oral, and visual communication that is professional, accurate, and adapted to the audience.
3. Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.

- I have completed this outcome through the implementation of a Red-Black Tree in enhancement two. Prior to this, the code used a standard binary search tree. While easy to implement and effective for many situations, these are less than optimal in certain edge cases (e.g., presorted data, etc.) in which they have a time complexity of O(n). Using a self-balancing tree, like a Red-Black Tree, guarantees a time complexity of O(log n).

4. Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.
    - I have achieved this outcome through the use of prebuilt libraries to speed development while also utilizing best practices in the areas that these libraries focus. All three enhancements used Python's CSV module to handle CSV manipulation. Additionally, enhancement three used Python's sqlite3 module to manage interaction with the SQLite database. Both streamlined the development of the code by preventing the need to reinvent the wheel and also ensured the functionality that these performed was done in an optimal and efficient manner since I was using well vetted and bug free code.
5. Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.
    - I achieved this outcome through the use of data validation and exception handling throughout the enhancements. The data validation ensured that information given to the programs was reasonable and in the format intended to prevent crashes or

unintended side effects (e.g., SQL injection, etc.). Additionally, the exception handling allowed the code to catch thrown errors and deal with them in a way that prevented the program from crashing.