CS-499: Capstone

Amber Law

15 December 2024

7-1 Final Project Submission: Reflection

As I near the completion of my Computer Science degree, I reflect on the coursework and projects that have shaped my knowledge, skills, and professional growth. Throughout this program, I have gained a solid foundation in various core areas, including software engineering, data structures, algorithms, database design, and cybersecurity. The process of developing my ePortfolio has been an integral part of this journey, allowing me to highlight my strengths and demonstrate my competence in these fields.

Developing a professional portfolio to highlight my distinct abilities and expertise is an essential tool for effectively communicating my value to potential employers. The ePortfolio serves as a comprehensive foundation, presenting clear examples of my proficiency in key areas of Computer Science. The selected artifacts from various courses within the program demonstrate my progression and competence in core domains such as software design and engineering, algorithms and data structures, and database management.

My ePortfolio encapsulates the knowledge and skills I have cultivated throughout my academic journey in the Computer Science program at SNHU. It reflects both my academic growth and achievements, including honor roll recognitions for consistently high performance. Through this portfolio, I have curated a professional-quality representation of my capabilities, combining written and visual elements that are cohesive, technically robust, and tailored to meet the needs of specific audiences and contexts.

One of the most valuable lessons I’ve learned is the importance of collaboration in a team environment. Working on projects such as the **ABCU Advising Program** (CS300), which required designing and implementing a sorting algorithm, I learned how to work effectively with others, exchange ideas, and contribute to the success of a project. Whether working on group assignments or solo tasks, I’ve developed my ability to communicate clearly with stakeholders, such as professors, peers, and potential employers. This experience has refined my communication skills, helping me to translate complex technical concepts into understandable language for non-technical audiences.

The code implements a sorting algorithm used to organize items. It sorts a list of student records based on specific criteria such as name or academic performance. The sorting algorithm takes an unsorted list of student data and organizes it in ascending or descending order based on the specified criteria. It handles different data types, such as numeric and string values. The algorithm is designed to be efficient in terms of time complexity. It uses an appropriate sorting technique based on the size and nature of the data set. The algorithm is functional but could benefit from optimizations, such as handling edge cases or improving the runtime for larger datasets.

I had implemented relevant code review criteria to support my stated findings for artifact algorithm. This included efficiency, edge cases, testing, and any additional comments I had added regarding the sorting algorithm. The code was functional, but its time complexity needed be improved. Depending on the dataset, it had to be optimized by using more efficient algorithms, such as merge sort or quicksort. The algorithm was to be improved by handling edge cases like empty lists or very large data sets. While the algorithm worked for normal input, additional test cases should’ve been added to ensure its robustness against edge cases and large inputs. The code lacked detailed comments, which makes it difficult to understand the specific steps of the sorting algorithm, especially for complex sections. Target areas for improvement such as structure, logic, efficiency, functionality, security, testing, commenting, and documenting. To improve the algorithm, I had: (1) implemented a more efficient sorting algorithm (e.g., quicksort or merge sort) to reduce the time complexity for larger datasets, (2) added checks for edge cases like empty or extremely large datasets, (3) wrote unit tests to ensure the sorting algorithm works correctly across various test cases, (4) improved the documentation to explain the steps of the sorting algorithm more clearly. These enhancements demonstrate skills in algorithm optimization, edge case handling, and software testing, aligning with the course outcomes of developing and improving algorithms and applying data structures efficiently.

In terms of technical expertise, I’ve developed a strong understanding of data structures and algorithms, as demonstrated in the sorting algorithm artifact. By optimizing sorting methods for efficiency, I not only deepened my understanding of algorithmic complexity but also honed my problem-solving skills, which are essential for building scalable and efficient systems. Additionally, through coursework like **CS340: Salvare Search and Rescue Web App**, I gained hands-on experience in database management, learning how to design and implement client-server models. This experience allowed me to create practical, real-world solutions using relational databases, including query optimization and data integrity techniques.

The code interacts with a database to store and retrieve information about rescue operations, including mission details, personnel, and equipment. The code provides basic CRUD (Create, Read, Update, Delete) functionality to interact with a database. Users can search for and update records related to rescue missions, equipment, and personnel in real time. The system allows users to search for specific mission data, update details about equipment or personnel, and delete obsolete information. The database interaction is efficient, but there was room for improvement in terms of security and optimization.

I had implemented relevant code review criteria to support my stated findings for artifact database. This included database structure, security, efficiency, and any additional comments I added regarding the web application. The database structure is well-defined but could’ve benefitted from additional normalization to prevent data redundancy and ensure data integrity. The code could’ve use more secure methods for handling user authentication and database access. For example, implementing parameterized queries would prevent SQL injection attacks. The code performed database queries efficiently for small datasets, but there were a risk of inefficiency when the dataset grows larger. Indexing could’ve be added to frequently queried fields to speed up search results. The database interaction code was not well-documented, which made it difficult to maintain and extend in the future. Target areas for improvement such as structure, logic, efficiency, functionality, security, testing, commenting, and documenting. Improvements had included: (1) normalizing the database schema to reduce redundancy, (2) adding parameterized queries to prevent SQL injection, (3) optimizing queries and indexing frequently used fields, (4) adding comments to the database interaction sections for better documentation. These enhancements demonstrate skills in database normalization, security best practices, and query optimization, aligning with the course outcomes of developing efficient and secure database systems and ensuring data integrity.

Another critical area I’ve focused on is software engineering, particularly through the **CS360: Inventory App for Mobile Architecture and Programming** artifact. In this project, I designed a secure login system, applying best practices in software development, security protocols, and mobile application architecture. By working on this artifact, I not only strengthened my coding skills but also gained insight into the importance of security in software design, particularly in handling sensitive user data. Understanding security risks and mitigating them through encryption and secure authentication practices is an area I plan to continue developing as I move forward in my career.

The code functions as a secure login system. It ensures that users are authenticated before they gain access to the app. The system uses a standard authentication process, which includes checking the user’s credentials against a database and providing appropriate error messages when the login fails. The code takes user input (username and password), validates the input, hashes the password for security, and checks the credentials against a database. If the user is authenticated, they are granted access; otherwise, an error message is displayed. The primary function is to authenticate users. It includes basic validation of input fields, hashing of passwords, and querying the database for credential matching. The code also contains error handling to deal with incorrect logins.

I had implemented relevant code review criteria to support my stated findings for artifact software design. This included code efficiency, error handling, security, and any additional comments I had added regarding the inventory application. The login logic was efficient in terms of user input handling but could’ve been optimized by integrating more advanced hashing algorithms or multi-factor authentication for added security. While the code did include some error handling, it could’ve been improved by providing more specific error messages, such as distinguishing between incorrect usernames and passwords. The password hashing mechanism was secure but could’ve been further enhanced with salting or transitioning to more secure methods like bcrypt or Argon2. Some sections of the code could’ve used more comments explaining the logic, especially around security measures and how the password validation process works. Target areas for improvement such as structure, logic, efficiency, functionality, security, testing, commenting, and documenting. Improvements included: (1) refactoring the authentication logic to be more modular, (2) enhancing security with better password hashing techniques, (3) adding multi-factor authentication, and (4) improving error handling and messaging. To enhance the code, I had: (1) implemented a more secure password hashing mechanism such as bcrypt, (2) added multi-factor authentication to increase login security, (3) improved error handling by providing more specific feedback on login failures, and (4) refactored the code to make the authentication process more modular and maintainable. These enhancements demonstrate skills in secure coding practices, error handling, and code modularity, aligning with the course outcomes of designing and implementing secure software systems and improving the functionality and security of applications.

The artifacts I have chosen for my ePortfolio—**Secure Login** (CS360), **Sorting Algorithm** (CS300), and **Database System** (CS340)—serve as tangible representations of my technical abilities and growth throughout this program. The secure login project demonstrates my understanding of mobile app architecture and security, while the sorting algorithm showcases my grasp of algorithmic design and data structures. The database system artifact reflects my ability to design and implement database solutions, combining my knowledge of data modeling and server-client development. Together, these projects highlight my versatility and well-rounded expertise in computer science.

As I prepare to enter the workforce, my ePortfolio serves as both a reflection of my educational achievements and a roadmap for my future career. The skills I have gained, coupled with my experience in software design, data analysis, and security, position me to contribute meaningfully in the technology field. This ePortfolio not only demonstrates my technical competence but also my ability to collaborate, communicate, and apply these skills in real-world settings, making me a valuable candidate for potential employers.