**CS-499 4-2 Narrative**

**Briefly describe the artifact.**

The artifact in question is a Python-based authentication system, an enhanced version of a basic authentication system originally written in Java. It was created in java initially for my IT-145 course, in 2019, and last modified in March 2024. This system introduces several significant improvements over its Java predecessor, including the use of bcrypt for secure password hashing, admin functionalities for user account management, an algorithm for generating unique usernames, and advanced password management features aimed at improving both security and usability.

**Justify the inclusion of the artifact in your ePortfolio.**

The inclusion of this artifact in my ePortfolio was a deliberate choice, aimed at showcasing my skills and abilities in secure coding practices, algorithm design, and data structure implementation. This authentication system exemplifies my approach to problem-solving in software design, particularly in the areas of security and user management. The transition from MD5 to bcrypt hashing is a testament to my commitment to enhancing security. The unique username generation algorithm and the system for generating and managing passwords highlight my capability to innovate and implement practical solutions in data handling and security.

The artifact was significantly improved from its original version by:

* Integrating bcrypt hashing to address vulnerabilities associated with MD5 hashes.
* Adding admin functionality to manage user accounts, reflecting my ability to extend systems with new features.
* Implementing a unique username generation mechanism to avoid duplicates, showcasing my problem-solving skills.
* Introducing advanced password management features to ensure users have strong, secure passwords and the ability to change them when needed, underscoring my focus on security and user experience.

**Did you meet the course outcomes you planned to meet with this enhancement in Module One?**

The enhancement of this artifact was aligned with the course outcomes of (1) 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, and (2) 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.

The first outcome was satisfied by incorporating algorithmic principles and best practices in computing solutions, notably in its security features and user management functionalities. By transitioning to bcrypt for password hashing, the system adopts a more secure, though computationally intensive, approach to protect against brute-force attacks, showcasing an application of algorithmic principles that prioritize security over operational efficiency. The introduction of unique username generation and a password management system, which includes the creation of temporary passwords and enforces password change requirements, highlights my understanding of algorithmic design. These features not only improve the usability and management of user accounts but also reflect an adept handling of design trade-offs, balancing complexity and functionality to enhance overall system security and user experience.

For the second outcome, developing a security mindset, I took a proactive approach to anticipating adversarial exploits and mitigating potential vulnerabilities. The use of bcrypt, known for its robustness against password cracking techniques, directly addresses the need for enhanced security in storing and verifying user credentials. Furthermore, the system's mechanisms for generating temporary passwords and enforcing their subsequent change upon first login are strategic defenses against the common pitfalls of default or weak passwords, mitigating the risk of unauthorized access. By implementing admin functionalities for account management, the enhancement also underscores a commitment to maintaining the integrity and privacy of user data. Altogether, these improvements not only rectify limitations in the original design but also embody a comprehensive security mindset that anticipates and neutralizes potential exploits through meticulous architectural and design choices, ensuring a higher standard of data protection and system integrity.

Overall, the code has been optimized for security using bcrypt, with some computational expenses as the tradeoff. Time complexity can be represented as O(c) for the cost factor of bcrypt, O(n) where n is the number of usernames, and O(l) for length of the password. O(n) will impact username creation, user records and password generation. O(c) will impact the password check. O(l) will impact new password validation, increasing with length. O(n) will become O(1) once database functionality is added. As a result of O(n), the algorithmic code is moderately efficient, and will become less so as more user records are added, due to the linear search through the records as more account are added. Efficiency will significantly increase with the addition of database functionality.

As for updates to the outcome-coverage plans, the successful implementation of these enhancements reaffirms my initial goals. However, future developments, such as integrating a SQL database for better account status management, are planned to further align with and exceed the course outcomes.

**Reflect on the process of enhancing and modifying the artifact.**

The process of enhancing and modifying the artifact was both enlightening and challenging. Through it, I learned the importance of secure password handling and user account management in software development. I gained practical experience in applying bcrypt for password hashing and understood the intricacies of creating a user-friendly yet secure authentication system. One of the main challenges faced was ensuring that the username generation algorithm could efficiently handle duplicates and provide unique identifiers for each user. This required careful consideration of algorithm efficiency and scalability. Another challenge was implementing a system that could prompt users to change their temporary passwords upon first login, balancing security needs with user experience. These experiences taught me valuable lessons in software development, especially in the areas of security, data structure design, and algorithm implementation.