The artifact I chose for my ePortfolio is a C++ user authentication program I created a few months ago. The goal of the program is to allow users to register with a username and password, and then later log in by validating their credentials against what's stored in a text file. This project demonstrates basic but essential programming concepts like file input and output, loops, conditionals, and user interaction. While simple, it models a common real-world feature like account management and gave me a solid foundation for exploring software design and security considerations.

I selected this artifact because it represents a complete, functioning program that handles file operations and simulates a basic login system. In its original version, the program worked fine, but there were some clear areas where it could be improved both in terms of code quality and user experience. For example, there were no checks for empty inputs, error messages were minimal, and the structure of the main function was a bit cluttered. Enhancing this artifact gave me the chance to apply what I’ve learned about cleaner design, better error handling, and writing code that’s easier to maintain and extend.

In the enhanced version, I added input validation to ensure users can’t register or log in with empty usernames or passwords. I also added checks to confirm the file opens successfully before trying to write to or read from it, and I updated the messaging to give clearer feedback when something goes wrong. Functionally, I broke out the main menu logic into its own showMenu() function, which made the main() function much cleaner and improved readability overall. I also made sure to add comments throughout the code, including a clear warning that storing passwords in plain text is insecure, just to show I understand the risks and how I would approach it differently in a real-world scenario.

Additionally, this artifact demonstrates my understanding of algorithms and data structures. Specifically, I used a simple linear search algorithm to scan through the text file and match the entered username and password to what was stored. Even though this is a basic algorithm, it allowed me to apply concepts like data retrieval and comparison. I also worked with file streams and strings as my data structures, which gave me experience in handling external data sources and manipulating string data within the program. By enhancing the search logic and improving how the program reads and processes stored credentials, I strengthened my skills in both algorithms and data handling.

On top of that, this project also covers database concepts, even though it doesn’t use a formal database system like SQL. The users.txt file in this program acts as a flat-file database, storing persistent user data that the program can read from and write to. I implemented file input and output operations to mimic common database actions such as inserting new records (when registering a user) and querying data (when searching for matching credentials during login). While basic, this use of a file as a data store helped me understand the importance of data persistence, retrieval, and integrity, all key concepts in database management. By enhancing the way the file is handled, including adding checks to make sure the file opens properly and handling missing files gracefully, I was able to apply foundational database practices within a simple program.

This updated version reflects my growth as a developer. I’m not just writing code that works. I’m writing code that makes sense to others, anticipates user errors, and keeps future improvements in mind. It shows that I’m thinking critically about usability, structure, security, and data management even in a basic project. This artifact connects directly to the software design and engineering outcomes for the Computer Science program and now also demonstrates my ability to apply algorithms, data structures, and database handling in a functional project.

Working through the enhancements taught me how small changes can make a big difference. One of the trickier parts was deciding where to stop. I didn’t want to completely rebuild the project with encryption and full error logging, but I still wanted it to feel polished and intentional. Overall, the process reinforced the importance of writing clean, modular code, understanding basic algorithm logic, handling data persistence, and considering edge cases, all of which are skills that I will carry with me into future development work.