**Narrative**

This program started as a project for the class IT-145. It was a Zoo Authentication program that would allow users to login to the system and it would detect their role that was listed for them in a file with their username and password. It would then print out a message based on the role. It specifically would only allow three attempts to login before the program closed. I chose this artifact for several reason. The first reason is that it never worked. The artifact had a bug that it was not reading the password in correctly and would never let the user into the system. I wanted to fix this and show that I was able to do this properly now. I also thought that this program is something that an employer might like to see because it could show a login system for their employees.

Fixing this program and making it usable, and then converting it from a zoo system to a general use system, showed my ability to design and engineer this system. I not only fixed the system but added the ability to change their password after a default password is set by the system when a new user is added. This allows for an added layer of security for the users. The system is still able to have the user login and exit if they use the wrong username or password three times, but now is also able to have a messaging system, password change, and an admin/employee menu.

The algorithms and data structures are demonstrated with the messaging system. The messaging system uses linked lists to add a message to the list. It also uses a search to find the correct message to delete when the user requests to remove a message from the system. It also searches through the users to make sure that the correct user with the given username is the one who is being send the message. This had its own struggles of making sure that the user sending the message could not see the messages of the user it sent it to so keeping track of a sent messages list and a current user message list was needed for security. You don’t want someone to be able to read someone else’s messages!

Finally, I demonstrated my ability to use databases by converting the system to use a database instead of text files for user information. This allowed the system to automatically give the user a user id and that user id was then able to be used for the messaging system. This also made sure it was kept unique. The database is included in the program, so it didn’t need an external server to run it. This kept keep it self-reliant and also means that a company doesn’t have to pay for a server host unless they want to use one as backup.

Although this is one artifact, the additions fit together nicely. The same code that read from files before was updated to read from the database. When a user is removed from the database it also clears out the text file created by their message list to keep the system storage needed lower. This prevents it from building up many old message files after a employee leaves the company. Although I could have used databases for the messaging system, using the linked list allowed me to better demonstrate other abilities that I learned throughout my time in school. Before creating this program, I had never built a database into a program like this before. I had only worked on them on servers. This allowed me to learn how to use code to access a database. This also gave me more practice on how to use code to not only read and write from a file, but also to create and delete a file.