The artifact being presented for enhancement three was my final project in CS410 Software Reverse Engineering during 21EW4 and was originally named Loginapplication.cpp. As this enhancement is a continuation of the work presented in enhancement two, I have thus renamed the project LoginapplicationWSQL and is presented within a folder accompanied with required headers and c files. The project was created from reverse software engineering an application, interpreting the assembly code, and recreating the application using C++.

This artifact was selected for the enhancement of database functionality due to the simplistic and immutable nature of the arrays used to store customer information in the original program thus presenting an opportunity for improvement to the software by being able to store customer information within a database. Within this artifact and with the utilization of SQLite I was able to create functions that create a database and table within a given file system. Furthermore, upon opening the database the create table function creates a table with the given columns and rows required for customer information. If the table exists prior to the function call, then it simply connects to the table. Other SQLite functions were created including the ability to insert, remove, update, and delete data. A callback function had to also be created for the select all data function for output functionality. Each of the SQLite functions utilizes within the enhancement utilizes SQL commands to achieve application functionality allowing the user input, update, print and delete customer information stored within the database. I moved the ability to delete database information to the administrator menu previously created in enhancement two. Overall, I would state that I am most proud of not only my ability to initiate a table but also in the functions created to help utilizes the table created within the artifact.

In developing this enhancement, I feel that I have met a few course objectives and outcomes. In creating a database within a file system and allowing for user CRUD (Create, Read, Update, Delete) SQL functionality has met the objective of "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”. The functionality developed not only helped solved the problem of customer information storage but also allowed the user to be able to interact with the data for which the initial application did not accommodate. Secondly the outcome of “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” was partially met in the utilization of principle of least privilege as the ability to delete and insert within the database table was limited to the previously created administrator menu. Normal user access was limited to updating policy choice and printing all customer information. Lastly the course objective of “Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science”, I believe was also achieved through the architecture of the software application and the consistent use of syntax and comments. This application is a display of an environment that warrants collaborative input through its organizational nature allowing for an ease of understanding from outside parties using a consistent coding standard. In conclusion, this enhancement has been completed to the entirety of the initial prompt provided therefore I see no required additional coverage to complete.

In the creation of this enhancement, I learned how SQL and C++ can be interactive with each other to produce functionality within an application. In developing this application, I learned and understood a rhythm of how an SQL function is implemented within a C++ application from the creation of the database and table to the calls within the functions to connect to those objects as well as an ability to add error message outputs in the event of an issue with the given function. In my initial development I faced a few challenges. Originally, I had thoughts to create an online database, but this required the expense of a hosting service as well as an implementation of several proprietary APIs to ensure connect ability. Furthermore, upon experimentation with several services such as Single Store Database I found that certain database options are more receptive to certain coding languages. For example, with Single Store Database they provide an ease of use to implement JavaScript functionality to connect to the database, but the implementation of a C++ application requires much more in the form of installed headers and programs to implement. This led me to the understanding that different online database options cater to different types of clients. Upon this experimentation with Oracle and Single Store database options I moved to a more local database option utilizing SQLite. Upon creating the SQLite functions within the C++ application, I learned that a callback was required to output the data in database. This was something that proved to be challenging at first as I was unsure what was required to complete the function. Upon review of the SQLite website was able to understand the requirement as well as how to implement the callback function.