Paolo Valenti

Dr. Ogoh

CS-499 Computer Science Capstone

4 October 2025

Milestone Four – Enhancement Three

For the third and final enhancement, I am still making use of CS: 360's weight tracker app. The purpose of the app as previously stated was to allow local user accounts to track weight entries for specific dates, after entering a username and password to login. Along with being able to compare a set goal weight to their current weight on a specific entry. The other functions include editing, deleting, and adding multiple weights. After last week's work, a user can view their weight data in both labeled line and bar graphs. With this last enhancement the users are now automatically authenticated to a SQL Server, where their accounts can be managed my an admin. The app was created around April of 2024, towards the end of CS:360 course.

An app that is accessed via user accounts with passwords makes sense for its inclusion in my ePortfolio. It is especially fitting now that the accounts are managed directly from a database like the name of the enhancement suggests. The guidelines for the database enhancement concerned, "consider improving the efficiency of a project or expanding the complexity of the use of data structures and algorithms for your artifact". The modification of having the app store user accounts directly on a remote SQL server as opposed to a local SQLite database does exactly that. The efficiency has been improved because an admin can now perform SQL queries and operations directly on user accounts from Microsoft SQL Server Management Studio after authenticating through SQL Server Configuration Manager. The complexity involves establishing a reliable working connection to launch each time the app is opened, and average level competency with the two previously stated tools. There were a few aspects of this enhancement that showcase my skills, on top of knowledge on coding the connection class. Creating the working connection that launches on the app login screen shows that I am able to bridge the mobile and desktop systems for better control over the app's operations. I did this by making use of a Connection class object that utilizes JDBC (Java Database Connectivity framework) and its specific java class driver for SQL, JTDS. I then had to set parameters for the IP address, database name, admin username and password, and port number. This enhancement also required making a policy less strict so Android, which natively can avoid network connections, to allow one from the app. All these changes improved the artifact greatly, when successful connection to the SQL Server is established. An admin can access the SQL Database and view the users that currently exist by right clicking dbo.Users, checking the top 100 rows of the user's table, and perform whatever queries are desired on the needed account. The level of control over the app has not only increased drastically from having absolutely no overhead, to allowing an admin to modify accounts whenever needed. These modifications include, deletion, replication, role changes, and privilege promotions and demotions.

I would say that I exceeded in meeting the course outcomes mentioned in Module One. Originally, I had stated that this enhancement would cover only outcome five, but after reviewing the outcome one as well, I believe this enhancement covers that too. Outcome One involved, "Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science." The strategy I'm employing here for collaborative control over how the app is accessed and by who, is related to Access Management Framework or Role-based Access Control. This involves promotions and demotions of carefully selected persons to have access to the user accounts in SQL Server only as development and management requires. Once an employee's obligations have been fulfilled, regardless of their role, the admin credentials can be removed added or modified as needed. This encourages diversity as it concerns to app maintenance and development since multiple users from different departments in an organization have access to admin accounts at different times.

Outcome five states, "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." This enhancement leans in on the last part, "ensure privacy and enhanced security of data and resources". The reason this is so vital has to do with enhancement one, facilitating the apps access to the Android's SAF (Storage Access Framework), and the simplicity of registering accounts on the app. While using SAF is convenient for exporting CSV files, I'm not sure if enabling the app to access SAF opens any vulnerabilities into the user's device based on who logs in and what they have stored on their phones. Furthermore, account registration for the app is very simple, just username and password, no account verification of any type. So if either of these points opens the app to vulnerability, forced SQL Database control can allow an admin to prevent abuse. Since the admin can immediately target accounts and perform actions on them.

I learned quite a bit in creating this enhancement, not only in creating the connection class but about windows, SQL Server Management Studio, and SQL Server Configuration Manager. While I learned I spent a lot of time troubleshooting as well. The process of adding the connection class wasn't too hard, with the exception of making a non-strict policy for the connection object so network connections were allowed for the app. It took a lot of research on how to set the policy definition to make that exception. Once I had both of the SQL programs installed that I needed, I repeatedly ran into an issue where the database username and password were accepted but the connection wasn't being made. To remedy this I needed to make an inbound connection exception on Windows Firewall for port 1433, which is the default number for SQL Server connections. Then, I had to manually enable SQL Server, SQL Server Agent, and SQL Server Browser on the Configuration Manager to make sure that the database could be initialized and ran. Finally, I had to manually add the name of the database, “demo” in this case since that was what I originally used for testing and I decided to keep it. After those steps, inputting the database admin username and password successfully logged me in and connected to the database. Lastly, I needed to navigate to dbo.Users and figure out how to access the index that brought up the list of users with currently registered accounts for the app, to verify the connection was good. I then spent time repeatedly registering accounts and relaunching the app to ensure I saw the successful toast message appear about connectivity to SQL server working every time. While this project was lighter in the actual coding done to enable the SQL Server connectivity compared to the other enhancements, it was heavier in regards to SQL Server and Windows Firewall knowledge. Overall, this was a very rewarding and worthwhile enhancement.

References

Microsoft. (2023, November 21). *Windows Firewall overview - Windows Security*. Learn.microsoft.com. https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/windows-firewall/

rwestMSFT. (2025, March 18). *SQL Server Configuration Manager - SQL Server*. Microsoft.com. https://learn.microsoft.com/en-us/sql/tools/configuration-manager/sql-server-configuration-manager?view=sql-server-ver17

erinstellato-ms. (2025, May 19). *Install SQL Server Management Studio*. Microsoft.com. https://learn.microsoft.com/en-us/ssms/install/install

W3Schools. (2019). *SQL Syntax*. W3schools.com. https://www.w3schools.com/SQl/sql_syntax.asp