CS-499 Capstone Narrative

Jason M. Gross

Southern New Hampshire University

February 19, 2023

**Software Design & Engineering Artifact Justification**

The artifact chosen for software engineering and design is a mobile weight loss tracking application written in Java for the Android platform. It was originally developed for CS-390 Mobile Architecture and Programming course. I am translating the code from Java to VB.NET to allow it to run on the Microsoft Windows operating system. .NET code has the added advantage of being able to target platforms other than Microsoft Windows with few or no code changes. I chose this artifact to demonstrate my ability to work with multiple platforms and multiple programming languages as well as being able to convert applications from one platform or language to another. It requires careful analysis of the original code to determine if any improvements can be made to both design and security. This artifact provides the potential for both types of improvements.

**Overview**

The following are side-by-side screenshots of the four primary screens from both original Android Java application on the left and the newly translated VB.NET Windows desktop application on the right. Each of the four primary screens was first designed to resemble the original application’s layout and object placement. Then the original code behind each screen was translated into the VB.NET language for each new form. All data validation code and MySQL functions were placed in their own respective modules and can be used in other projects with minor changes promoting code modularity and reusability.

|  |  |
| --- | --- |
| **Original Java Android Application** | **New VB.NET Windows Application** |
| User Login |  |
| Set Target Weight Screen |  |
| Add New Weight Screen |  |
| Main Display Screen |  |

**Software Design & Engineering Challenges and Reflections**

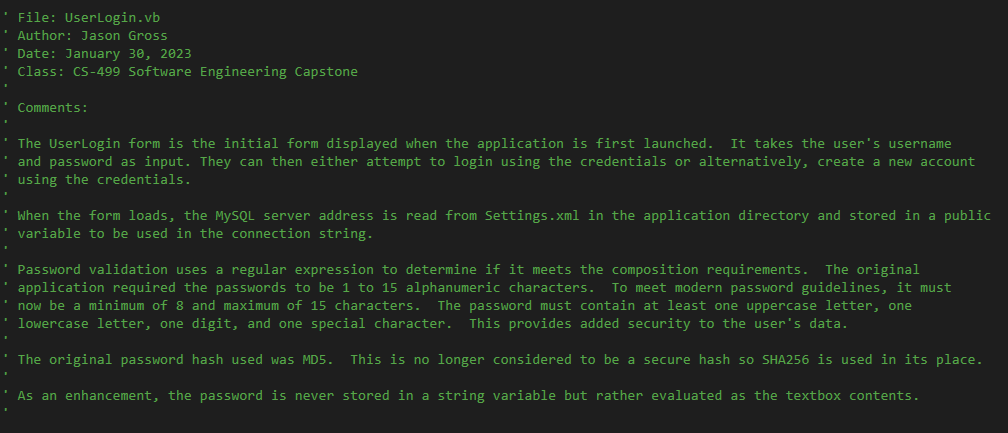
Because different operating environments are involved, there were trade-offs in the translation. For example, the main data display of the Android app used a custom control that displayed each row of data with its own edit and delete buttons. This was a good design choice for a small touchscreen app but an unnecessary complexity on a desktop application with mouse functionality. I chose a simpler DataGridView control in .NET to make working with and displaying data easier in the new application.

I am familiar with Microsoft SQL Server and working with those databases in .NET applications but as an added challenge, I chose to use MariaDB as the database platform for this project instead. MariaDB is a free and open-source fork of MySQL and has its own .NET objects for database interaction. I had to take some time to go through the documentation and example code to familiarize myself with how to implement the .NET MySQL data objects. I created a separate test application to experiment with the new objects until I felt comfortable implementing them in the project application.

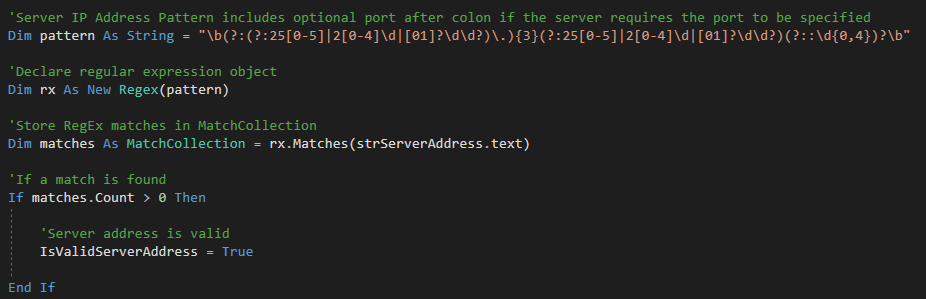
I added a chart control on the very first iteration of the software with the intention of adding functionality in future iterations. It turned out to be a fortunate decision because the latest .NET core does not support that chart control. Rather than try to recreate a chart object manually or use an untested third-party control, I recreated the project using an earlier version of .NET that supports the chart objects. If this had been discovered later, it would have involved significantly more work to migrate the project to a different version of .NET. This also illustrates the importance of researching availability and compatibility of objects that will be used when developing .NET applications.

**Software Design & Engineering Objectives Review**

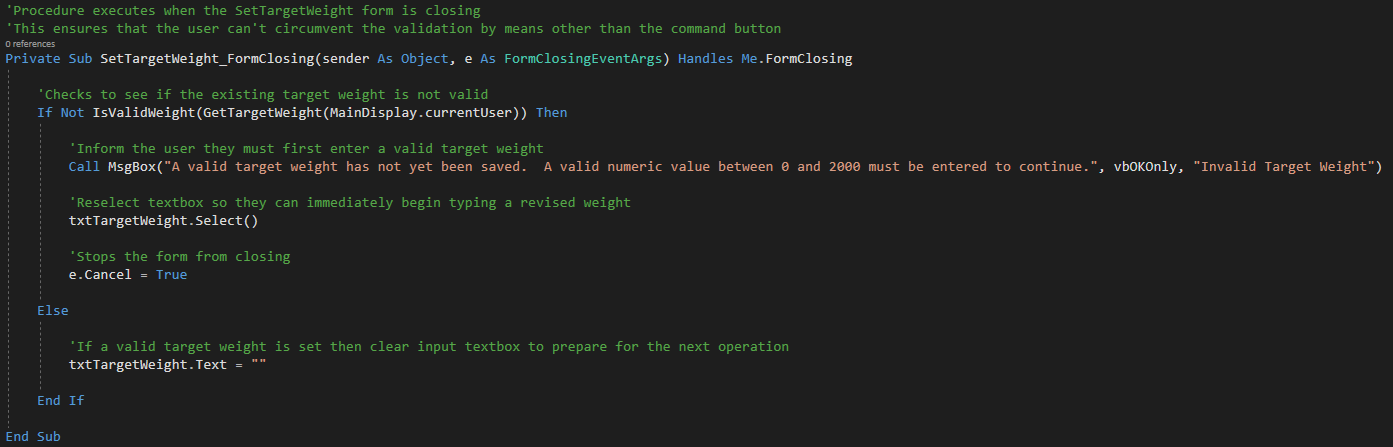
I have added detailed comments throughout each of the forms and modules to explain exactly what is happening in the code and why various design choices were made. This demonstrates the ability to articulate my ideas behind the code to other programmers. They should be able to open the code and immediately follow what is happening and be able to make changes with confidence. The following is an example header from the User Login form giving an overview of the functionality and design choices. Each statement of code is equally commented to be as informative as possible to the readers.



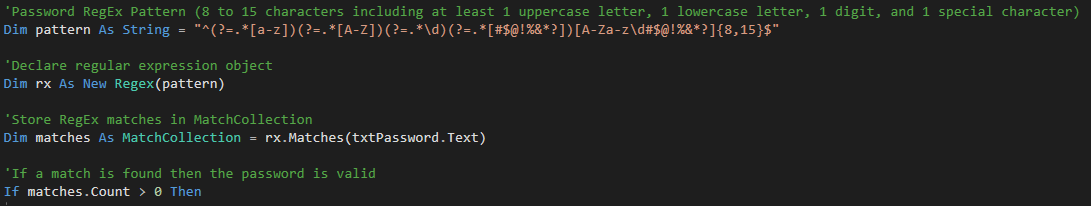
I have demonstrated the ability to program solutions to logic problems by implementing more complex regular expressions for data validation rather than using multiple logic conditions in a nested conditional structure. This provides more robust input validation using fewer conditions and lines of code. The following excerpt is from the Settings form. The code validates a server address with optional port number entered by the user. This example demonstrates the flexibility of validating complex strings with optional parameters using just a few lines of code.



I have demonstrated innovative skills in form navigation design by taking advantage of various language features including the ShowDialog() method, style properties, and the loading and closing events of the forms to control form navigation and ensure the user is unable to circumvent the intended flow by clicking an unexpected button.



I have demonstrated my ability to identify and address design flaws related to security by not storing plaint text passwords in any additional variables or passing them to any function in plaintext form. In the following example, the user input is evaluated directly from the textbox’s text property and hashed before being stored in memory and passed between functions.



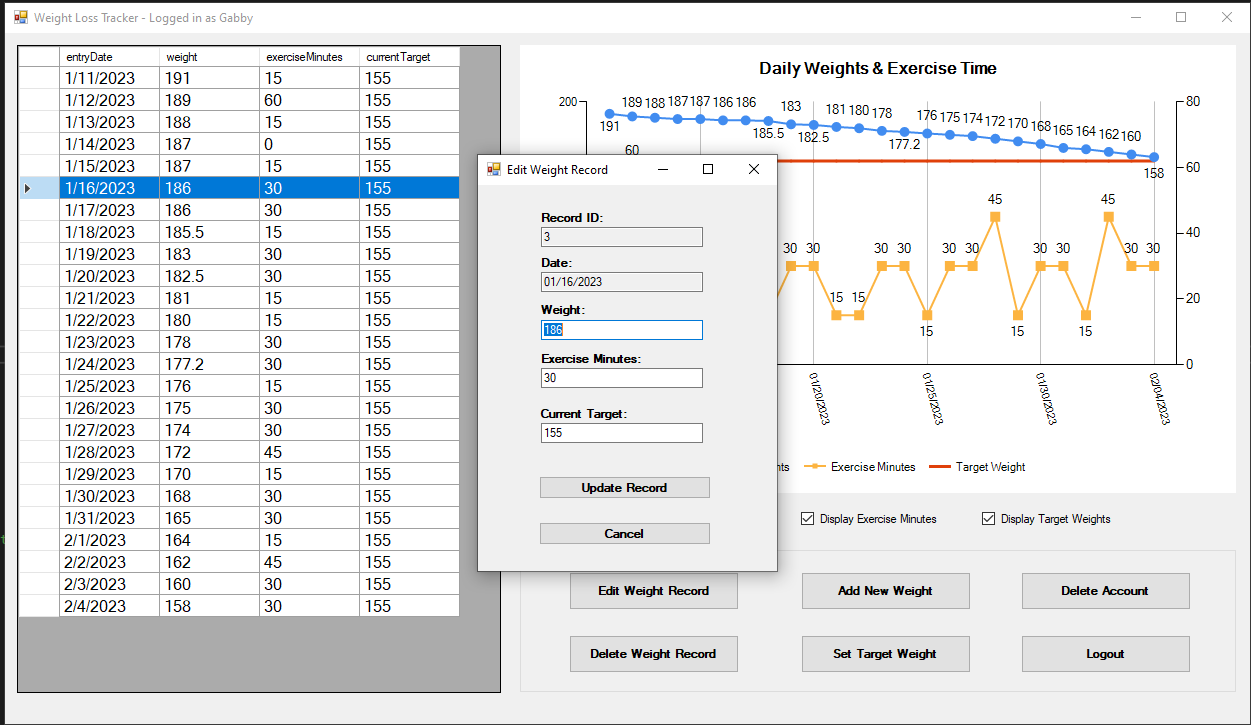
**Algorithms & Data Structure Artifact Justification**

The artifact for algorithms and data structures will be a continuation of the weight tracking android application that has been translated to VB.NET. Now that all of the original functionality is implemented, I will be implementing several value-added enhancements to both functionality and security. The data tracked by the original application will be expanded to include an additional metric of exercise minutes that the user can track along with their weight. A data chart will be added demonstrating the ability to manipulate data structures and translate them to visual representations for the user. Additionally, several security enhancements will bring the application in line with modern standards when dealing with passwords and user privacy.

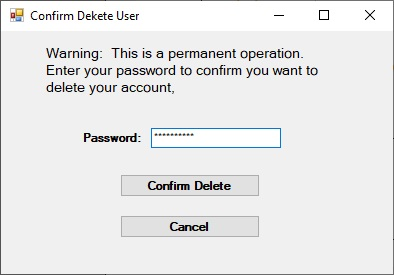
**Algorithms & Data Structure Challenges and Reflections**

The data chart was one of the primary enhancements planned for the algorithms and data structure category. I wanted both the chart and the DataGridView control to share the same data structure to ensure consistency and eliminate the possibility of having one control display data that the other does not. I chose to use the DataTable object because it is a very powerful data structure that can be automatically populated with data returned from the database taking on the necessary structure without having to first be defined. Stored procedures will return all the records for the given user and store the data directly in a DataTable object. That object can be immediately set as the data source for the DataGridView control with another single line of code. The DataGridView control can go on to use its internal functionality to allow the user to select records or order the data to their liking but will not affect the DataTable object or the series points in the chart display.

The original application had record editing and delete capabilities on each row of data with a single button tap. By default, each cell in the DataGridView control is clickable. To select an entire row, the user must click the left-most gutter cell. However, this would not be immediately apparent to the user. To compensate, I added additional code in the click event so whenever any cell is clicked, the entire row is automatically selected. This modification still requires the edit and delete operations to be a two-step process of first selecting a row and then clicking a button. I added additional code in the mouse event procedures so if a user double-left-clicks on any data row, it immediately opens the editing form with the selected record. Additionally, a right click of any field initiates the delete operation asking for permission to delete the selected record. These additions provide single-step functionality to match the original application.

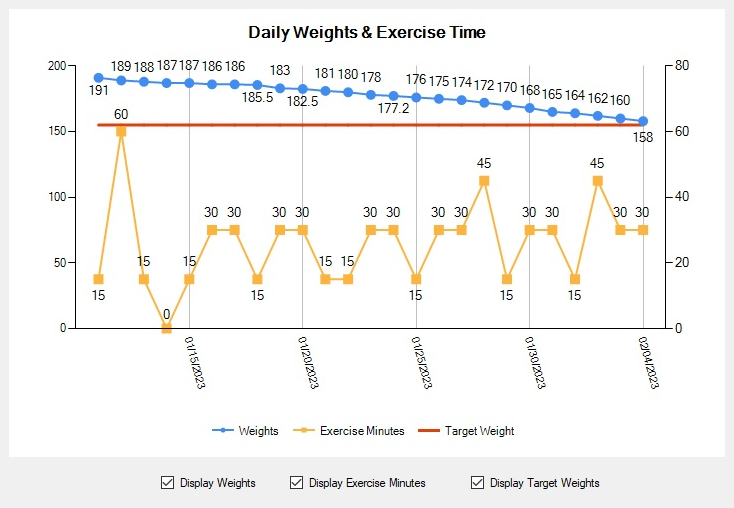


Finally, a new button was added to allow the user to delete their account and all the associated weight records. This provides peace of mind that the user’s data can be fully controlled by them and completely removed from the system if they wish. A stored procedure performs the operation, but I quickly realized a security risk existed where the procedure only took the username as a parameter and did not perform any other user authentication. I closed this gap by requiring the user to re-enter their password before calling the stored procedure and the hashed password is passed and authenticated before carrying out the operation. This way, if someone had discovered the credentials to execute the stored procedure without the application, they would still not be able to delete an account without first discovering the user’s password.

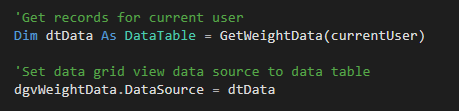
****

**Algorithms & Data Structure Objectives Review**

As more data points are entered during testing, the chart started to become cluttered. I demonstrated my ability to use innovative skills by implementing a series of checkboxes to allow the user to turn each data series in the chart on or off on demand. This not only alleviates the clutter when desired but also allows the user to quickly show or hide information giving them the flexibility to specify how their data is displayed.



The use of DataTable objects to hold data from the database and share them between the different forms and controls demonstrates the ability to code solutions to a logic problem involving data structures. The database calls happen only when necessary and few if any coding changes would need to be performed if the source data changed or expanded. For instance, a new field was added to the weight table to track exercise minutes but the code to pull the data and display it in the DataGridView remained the same. The DataTable object adapted to the change based on the new data and a new field was also displayed in the DataGridView control with no additional code changes. The following shows how just two lines of code are needed to import data into the data table object and assign it to the data grid control. This code will work even if fields are added or removed from the database.



I have demonstrated my ability to identify and address design flaws related to security by replacing the MD5 password hashing algorithm in favor of the more secure SHA256 algorithm making password hashes more difficult to compromise if discovered. Additionally, I strengthened user security by increasing the minimum length and composition requirements of their passwords to align them with modern best practices.

**Databases Artifact Justification**

The original mobile weight tracking application utilized a local MySQL Lite installation. The database components of the primary artifact will be a recipe for a complete back-end solution to support the weight tracking application. The result will be a SQL script that creates the database, user account, data tables, and all stored procedures used by the front-end application. With this modification, the back-end no longer needs to exist on the same PC as the front-end application. Rather, it will exist on a separate secured server and be shared by multiple client applications. This facilitates greater security since the data and queries are not stored in the client binary. It also provides better performance of the client machines since the data processing is performed on the server. This enhancement will demonstrate many applications of advanced MySQL concepts to expand the software with geographically multi-user capabilities.

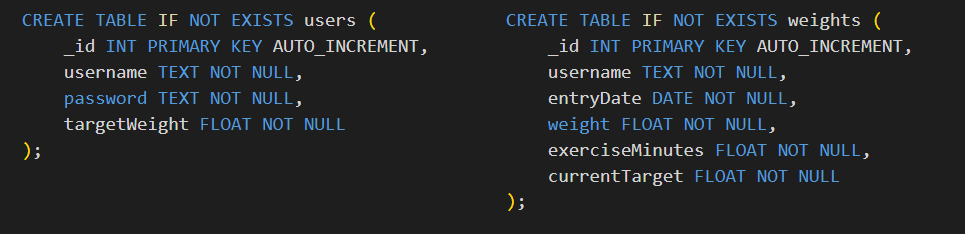
**Databases Challenges and Reflections**

The first challenge in this phase of the project was establishing a MySQL database server on a separate PC. A complete server installation is beyond the scope of this project but as a brief overview, I used a computer running Arch Linux and installed MariaDB, a modern MySQL fork, using a series of pre-built scripts to get a secure working database server up and running in just a few minutes.

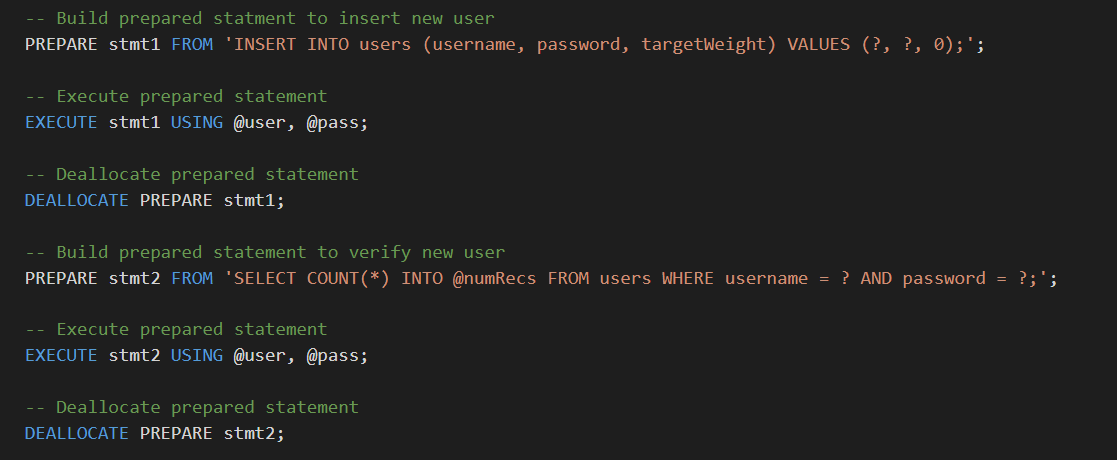
I created a user account called WeightAppUser that will be used exclusively by the application. The account is only granted permission to execute stored procedures contained in the database. This way, if the account credentials were reverse engineered from the binary, they could not be used to access the tables directly or execute commands that could reveal additional exploitative information about the database. This strategy follows the security best practice of least privilege which is granting users only enough access to successfully perform their specific tasks.



When creating the new data tables, I chose to consolidate the user’s target weight into the users table rather than in a third table. This reduces complexity and eliminates unnecessary data repetition. This only required adding a new field to the user table and the original target weight table could be eliminated. There was a second design challenge involving data types for the fields in the data tables. The original application exclusively used strings for all fields. This was likely a design choice to make the application easier to program. I chose instead to use appropriate numeric data types for each floating point numeric field and implement proper parsing and error handling in the application code. Storing numerical data properly will eliminate confusion with future troubleshooting and development.



The biggest challenge wass implementing the different functions of the application using server-side stored procedures. I designed them to be intrinsically secure by using prepared statements. This prevents maliciously crafted parameters from causing unintended SQL code to execute. This behavior is known as SQL-Injection attacks and prepared statements ensure the parameters are treated accordingly and not treated as text to be arbitrarily appended to a SQL statement. Additionally, I designed each procedure to first perform the intended operation and a second set of instructions to validate that the operation completed successfully. A return value indicates success or failure to the calling application. In the event of failure on some procedures, the value also indicates which part of the procedure failed to give the user more meaningful error messages and allow the application to react accordingly.

****

**Databases Objectives Review**

The final SQL script can be run on any MySQL database server to create the components necessary to support the application. Migrating the server to other hardware would be a trivial matter of running this script on the destination server and copying over the data tables from the previous server. This demonstrates my ability to use innovative skills and techniques to implement a complete database solution for the .NET weight tracking application.

The library of secured stored procedures was designed to provide all of the database functionality for the .NET weight tracking application and does so completely independent of the client software. This makes the client extremely light on resource utilization as the database handles the bulk of the processing. This solution demonstrates my ability to solve problems in storing/accessing/modifying data in a client/server model.

I addressed two potential design flaws related to security. The first was the user account used by the application. By restricting access to only execute select stored procedures, the compromised credentials could do far less damage than an account with access to the data tables and permissions to view the stored procedure code. Additionally, using prepared statements in the stored procedures makes them intrinsically secure and less susceptible to SQL Injection attacks.

The database script is logically segmented and fully commented detailing the functionality of each SQL statement and the reasoning behind them. This demonstrates my ability to clearly articulate my ideas to other developers that will need to evaluate or modify the code in the future. The following excerpt is a full stored procedure for updating the user’s target weight. It is presented in its entirety to demonstrate the concise commenting and structure throughout.

