Unit 9 OWASP Improvements

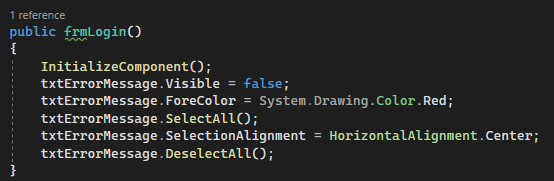
**Corey Crooks**

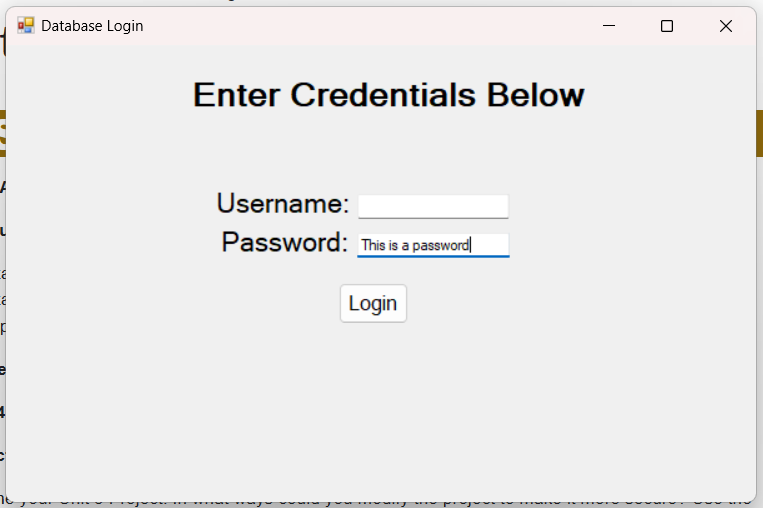
**Purdue University Global**

**IT481 – Advanced Software Development**

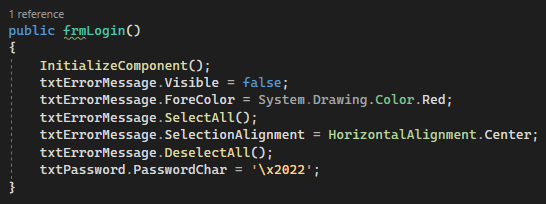
**Ahmad Kassem**

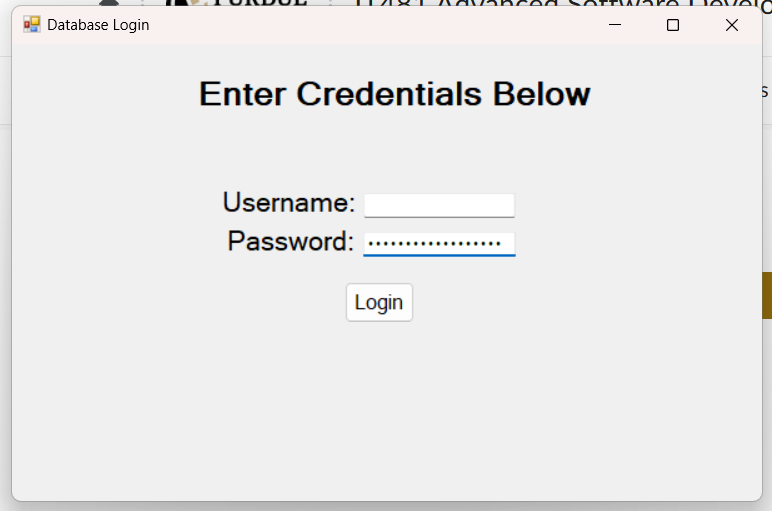
**June 21st, 2023**

**Change 1: The Password field:**



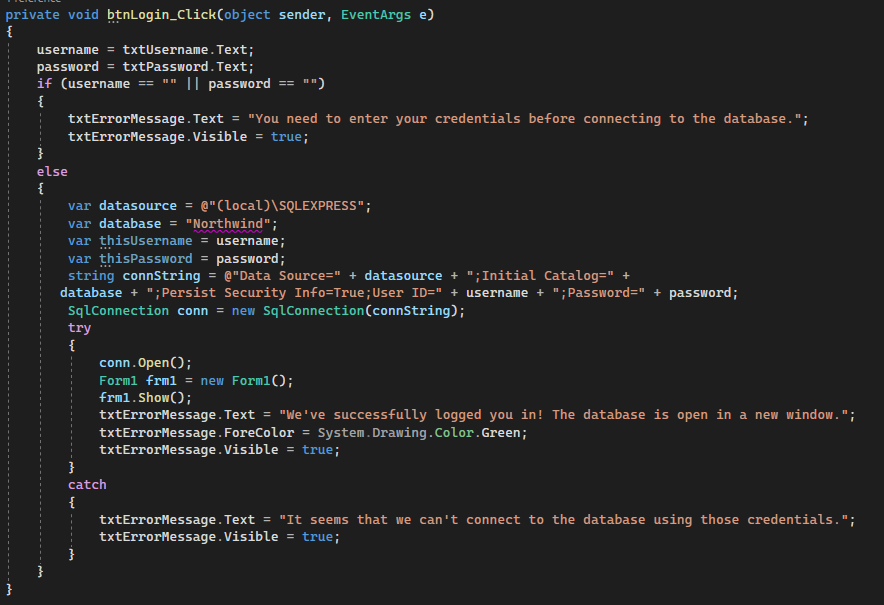
The above setup is relatively problematic from a security standpoint. As left in the Unit 3 assignment, this form allows the user to enter a Username and Password in order to access the database. As it stood, anyone looking over the shoulder of the user may be able to gleam the credentials from the screen with no repercussion.





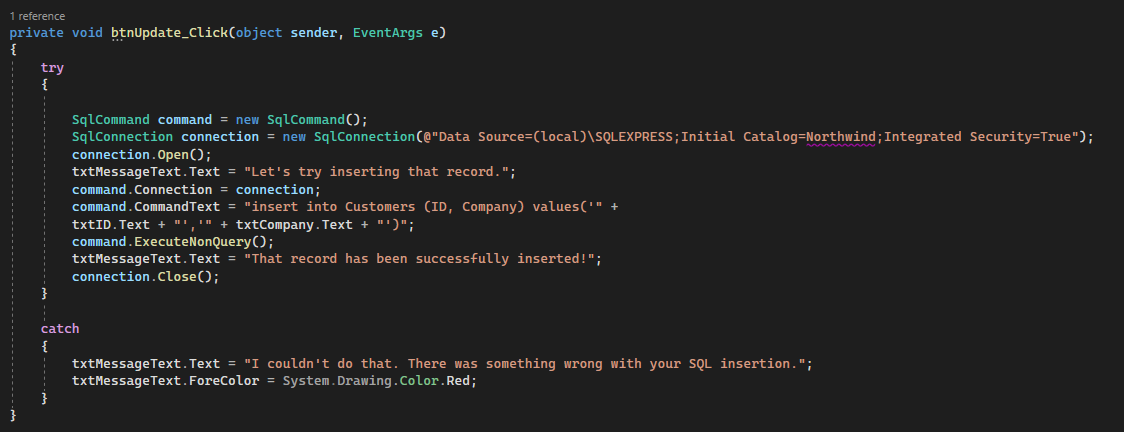
In the two screenshots above, I have made improvements to the security of the program. This coincides with the OWASP Secure Coding Practices guide on “Secure Passwords” (The Open Web Application Security Project, 2023). Instead of displaying the password field in plaintext, I instead use the Hexadecimal value for a bullet point in place of the character that would normally be displayed utilizing the “PasswordChar” attribute of the text box in visual studio.

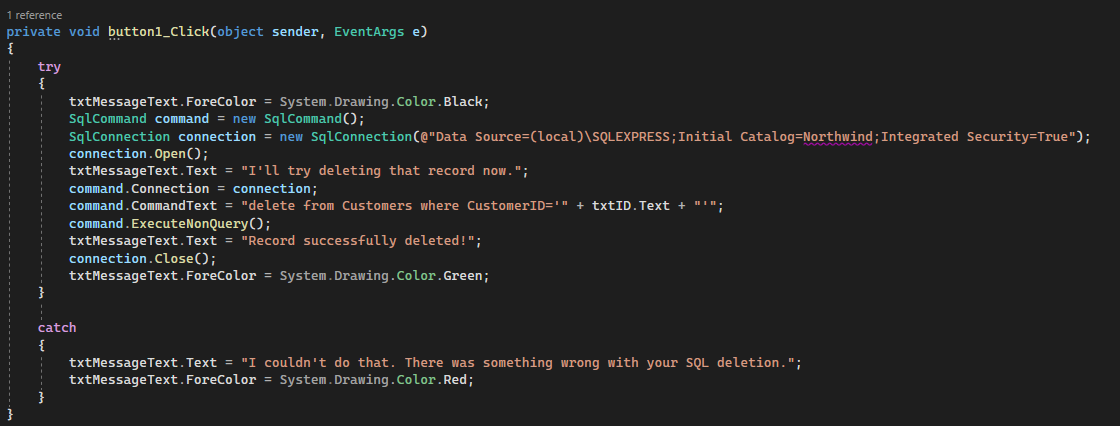
**Change 2: Try and Catch**

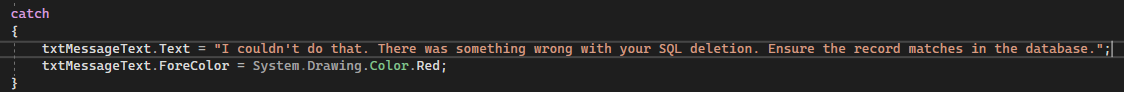
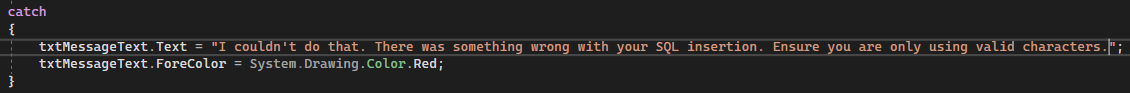


Above is a screenshot from the same login form showing the process of logging in. In particular, this is the method that is called whenever the “login” button is clicked. This focuses on the OWASP Secure Coding Practices guide on “Error Handling and Logging” (The Open Web Application Security Project, 2023). The program will try to connect to the database using this username and password, but if it is unsuccessful, it will alert the user to an error notifying of what went wrong.

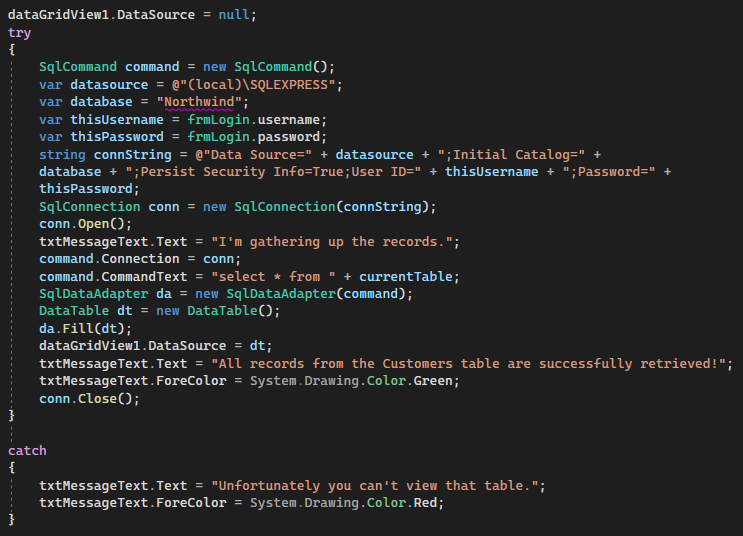
**Change 3: Input Validation**



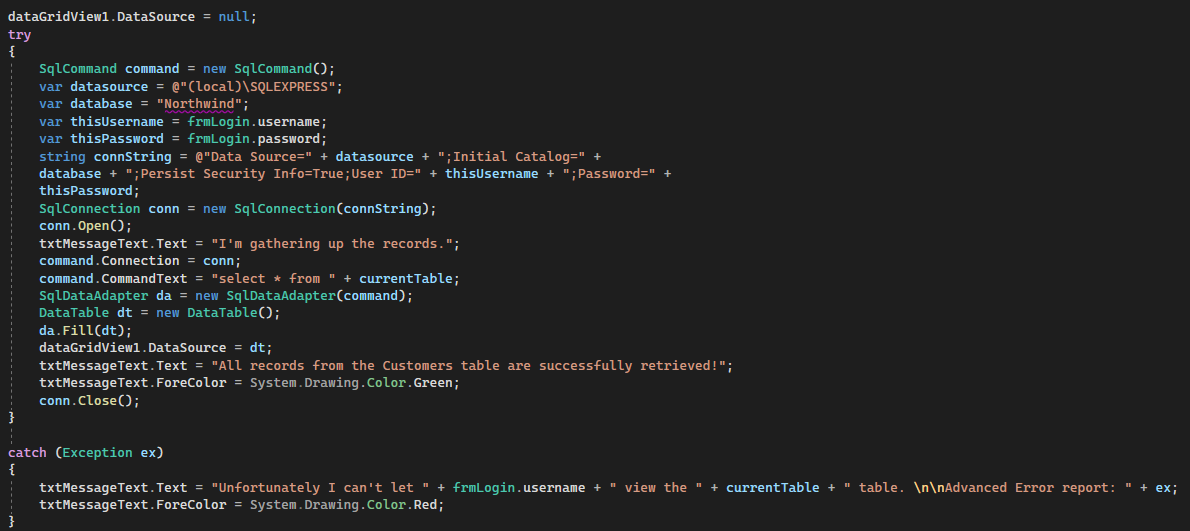


Although this also applies to the Error Handling approach as detailed in the OWASP Secure Coding Practices guide, Input Validation may be more applicable to this particular scenario (The Open Web Application Security Project, 2023). I have implemented a standard in which input is parsed, and tried as the user commands. In the first situation, a user will try to insert a record into the database. If the user inputs garbage data that cannot be parsed, the system will instead notify the user that something has gone wrong with the SQL insertion. In the second screenshot, the user will try to delete a record. If the user enters incorrect information to delete, they will be informed that the record cannot be deleted. I have tuned these strings to be more informative.

**Change 4: Access Control**



In the above screenshot, implementation has been done on a system that rejects certain users from accessing the parts of a database that they should not have access to. This is done to ensure that malicious actors are mitigated in the damage they can do to the sensitive information available. This is in accordance with the OWASP Secure Coding Practices guide on Access Control (The Open Web Application Security Project, 2023). I have also tuned the error message to be more informative. This now details what user is trying to access which table, and that they don’t have access to it. It also details the exact error message given by the system.



# **References**

The Open Web Application Security Project. (2023). *OWASP Secure Coding Practices-Quick Reference Guide*. Retrieved from owasp.org: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/