

Lab Activity – 2

Objectives

- To practice C# backend
- To learn hash functions

Exercise

Storing user passwords is a critical component for any application. When you store a user's password, you must ensure that you have it secured in such a way that if your data is compromised, you don't expose your user's password.

“Hashing” passwords is the common approach to storing passwords securely. A “Hash” is a one-way function that generates a representation of the password. So when a user signs up for an account and they choose a password, the password is stored as the generated hash, rather than the actual characters that the user typed in. When you run a password through a hashing function, it will always produce the same output. When the user tries to log in with their username and password, the entered password is hashed again and then compared to what is stored in the database. If the two hashes are the same, the user has entered the correct password.

In this application, you are expected to use hash to hide user data. To hash a data, use SHA2-256 algorithm. You can use this link to verify your hash output: <https://www.miraclesalad.com/webtools/sha256.php>

Design and implement an application that consists of two forms. Users will sign up and sign in using the first form. It ought to allow users to sign up with a user name and password and sign in after doing so. Design a user interface for first form with textboxes, labels, and button (Figure 1).

- ❖ User name (label&textbox)
- ❖ Password (label&textbox)
- ❖ Password confirm (label&textbox, enable on sign up)
- ❖ Sign up (button)
- ❖ Sign in (button)
- ❖ Warning label (label, enable when an undesirable situation occurs)

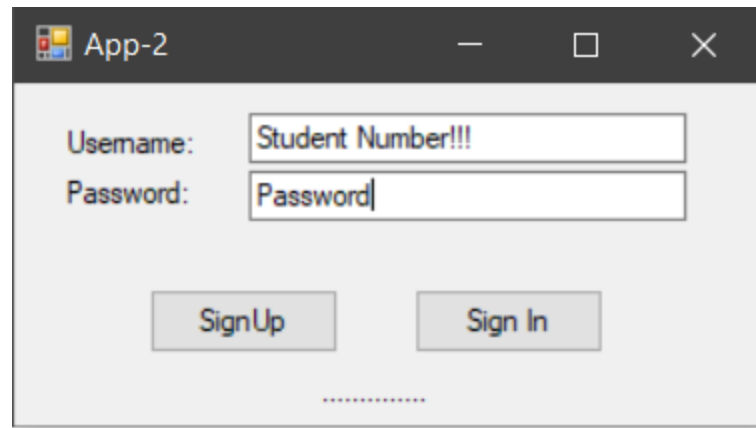


Figure 1

On the user control screen, make sure to examine the following items:

- ❖ Username and password must be at least 6 characters long.
 - ❖ Username and password must not be blank.
 - ❖ Configure that username and password are filled when the application launches.
 - ❖ Passwords must match on password confirmation.
 - ❖ If another user already exists with the same name, registration should not be allowed and user should be informed.
 - ❖ Username and password should be saved in csv file (UserInfo.csv, Figure 2).
 - ❖ Password should be saved as a hash (Figure 2).
 - ❖ User records should be checked when sign in (username text, password hash).
 - ❖ Administrator records should be checked using testUserCheck function (with testScript.dll file), function will return true if user information is verified (username text, password hash).
- For test -> username: administrator
Password: esoguce2023
- ❖ Second form should be launched after verifying the user registration during sign in, and the username should be transferred here.

```
20230001,7e5d8f0263cfd0059237c3d7f4ae7666085d306d75c14947063ecb54ec76fe16
20230002,eca5a247bd4839b9d26aefd9f206090174c86d9c67105cc967026d3a3d2d7c6d
20230003,844b61b82978b1fa0e2e999b7a9b17bc94271eaf226c8de9ea1d87b79dc84aeb
```

Figure 2

Design the second form with textboxes, labels, and button (Figure 3).

- ❖ User name (label)
- ❖ Score (label)
- ❖ Diary text (textbox)
- ❖ Save (button)

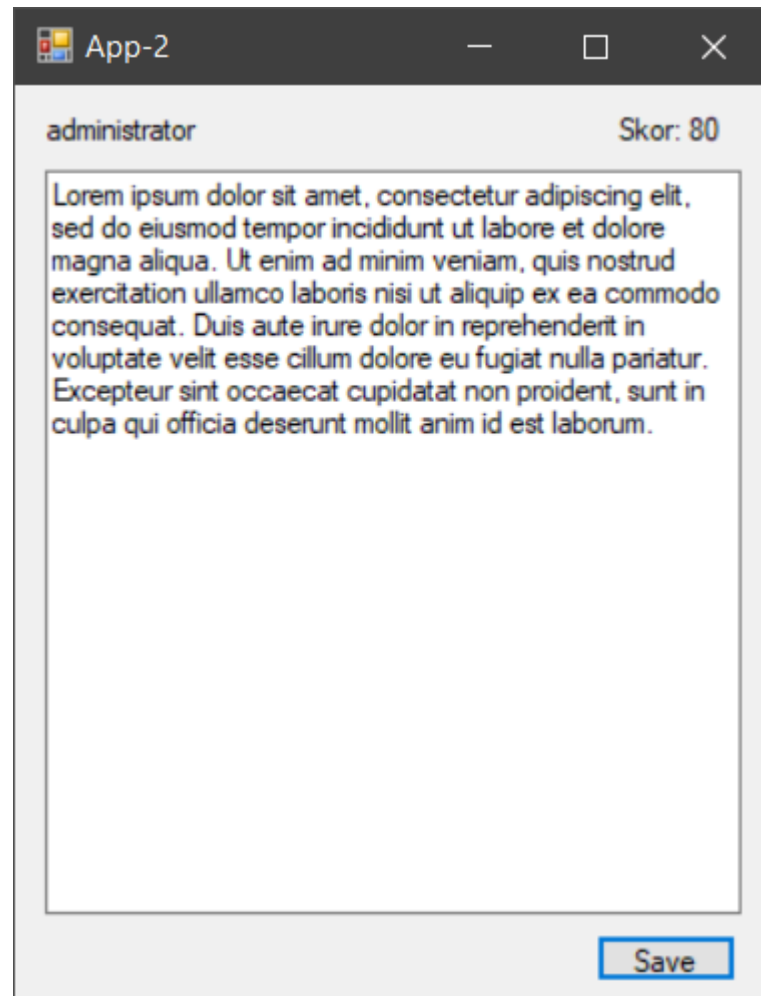
The image shows a screenshot of a Windows application window titled "App-2". The window has a standard Windows title bar with minimize, maximize, and close buttons. Inside the window, there is a label "administrator" on the left and "Skor: 80" on the right. Below these labels is a large text area containing Lorem Ipsum placeholder text. At the bottom right of the window is a "Save" button.

Figure 3

On the diary screen, make sure to examine the following items:

- ❖ Username should be brought from the first form.
- ❖ Diary text should be saved as a hash
- ❖ Each user's diary file must be its own and be in the "username.csv" format (the file content and name will be checked).
- ❖ Diary records and save file should be checked using testFuncApp2 function and test text (with testScript.dll and DiaryTestText.txt file), function will return int score value for score label (username text, diaryText text).

What is Expected

- Create proper user interface
- Carefully review and apply the above-mentioned requirements.
- Configure that textboxes are filled when the application launches
- Import testScript.dll file and use test functions for evaluation
- Properly show what is requested
- Apply object-oriented programming principles
- Report simply with step-by-step images and explanations of what has been completed
- Upload report (only .pdf) and **complete project file** (only .zip/.rar)
- The titles of the report and project files will be “<student number>_<student name>_lab<no>.<file format>”
(for example, 152120230000_firstname_lastname_lab1.pdf)
(for example, 152120230000_firstname_lastname_lab1.zip)
- Cheating is at your own initiative, but you also accept the consequences!