CST-350 Activity 3 Database and Users

Kaya Nelson

Course Number: CST-350

Professor Donna Jackson

1/20/2025

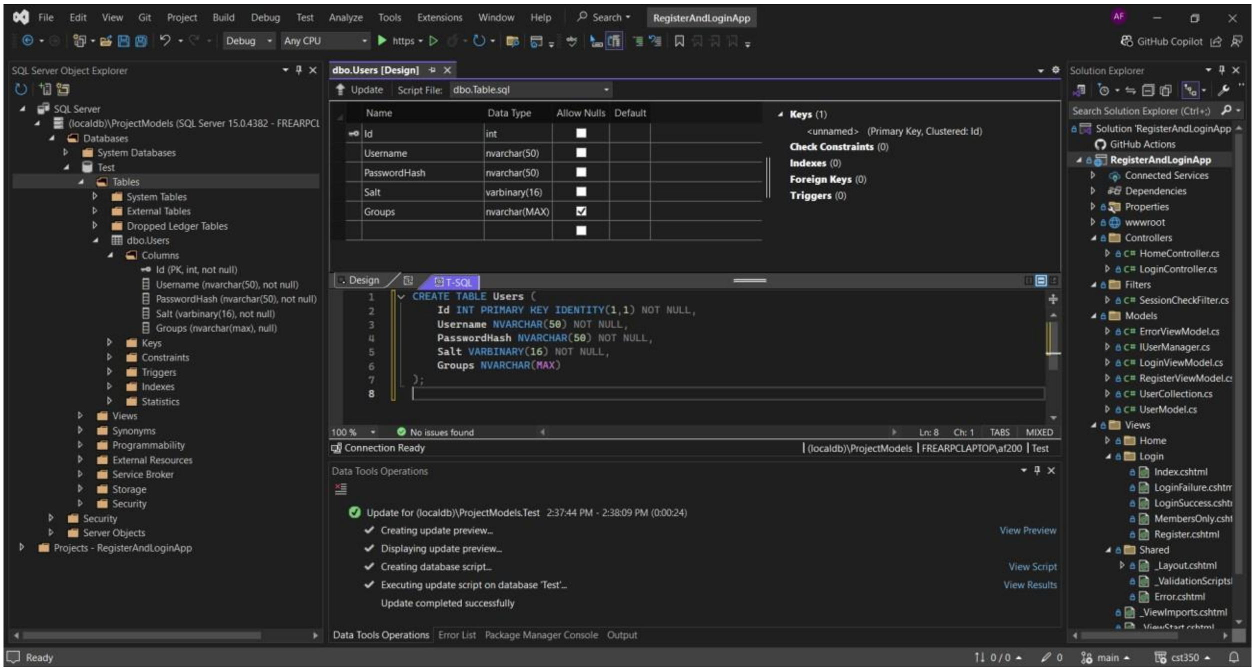
GitHub Link: CST-350/Topic 3 at main · Kdeshun/CST-350

Contents

Part 1: Database Setup and Integration Overview

Screenshots

Users Table Creation



Name	Data Type	Allow Nulls	Default
Id	int	□	
Username	nvarchar(50)	□	
PasswordHash	nvarchar(50)	□	
Salt	varbinary(16)	□	
Groups	nvarchar(MAX)	☑	
		□	

Keys (1)
<unnamed> (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```
CREATE TABLE Users (
    Id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    Username NVARCHAR(50) NOT NULL,
    PasswordHash NVARCHAR(50) NOT NULL,
    Salt VARBINARY(16) NOT NULL,
    Groups NVARCHAR(MAX)
);
```

100 % No issues found Ln: 8 Ch: 1 TABS MIXED

Connection Ready | (localdb)\ProjectModels | FREARPCLAPTOP\af200 | Test

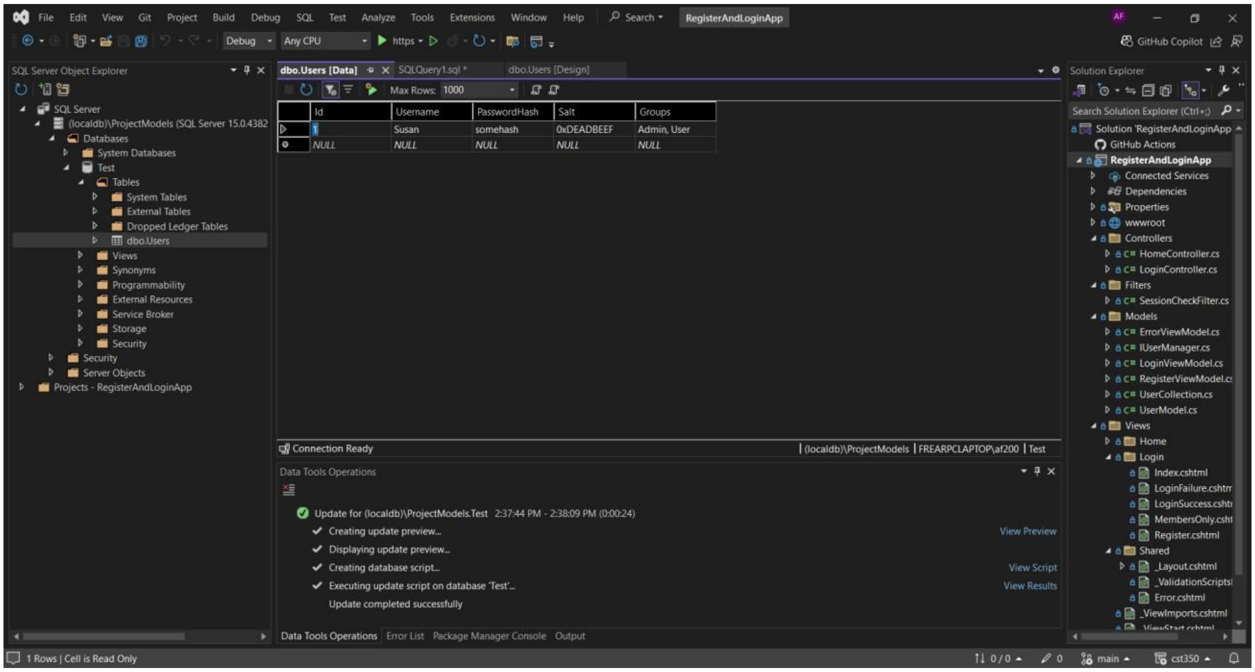
Data Tools Operations

```
Update for (localdb)\ProjectModels.Test   2:37:44 PM - 2:38:09 PM (0:00:24)
  ✓ Creating update preview...                          View Preview
  ✓ Displaying update preview...
  ✓ Creating database script...                         View Script
  ✓ Executing update script on database 'Test'...       View Results
    Update completed successfully
```

Data Tools Operations Error List Package Manager Console Output

Figure 1*: This image presents the Users table that has been set up within SQL Server. It includes crucial columns: Id, which serves as a unique identifier for each user; Username, used for user authentication; PasswordHash, which stores the hashed version of the user's password for security; Salt, an additional layer of security to prevent attacks; and Groups, which categorizes users based on their roles or permissions. These elements are fundamental to the overall database architecture and user management system.*

Users Table Data Insertion



Id	Username	PasswordHash	Salt	Groups
1	Susan	somehash	0xDEADBEEF	Admin, User
NULL	NULL	NULL	NULL	NULL

```
Connection Ready                                  | (localdb)\ProjectModels | FREARPCLAPTOP\af200 | Test
```

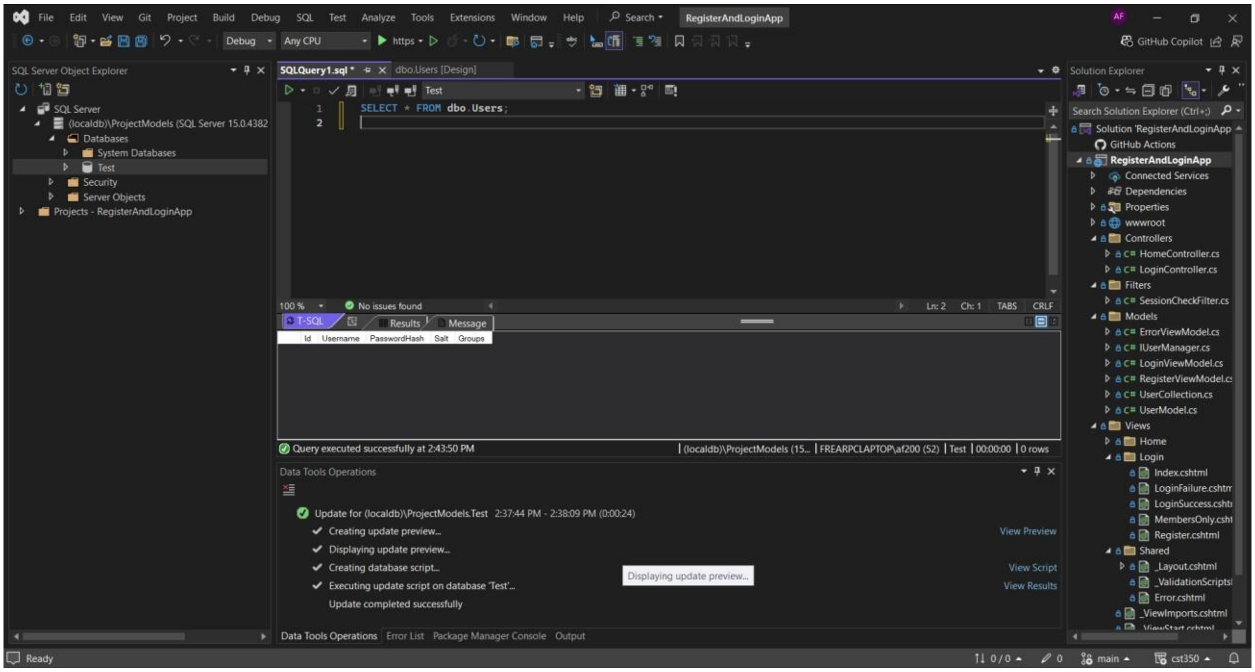
```
Data Tools Operations

✓ Update for (localdb)\ProjectModels.Test   2:37:44 PM - 2:38:09 PM (0:00:24)
   ✓ Creating update preview...                                   View Preview
   ✓ Displaying update preview...
   ✓ Creating database script...                                  View Script
   ✓ Executing update script on database 'Test'...                View Results
     Update completed successfully
```

```
1 Rows | Cell is Read Only
```

Figure 2: This image depicts the procedure for inserting a new user into the Users table. It highlights the entry of essential information, including the Username, which identifies the user; PasswordHash, the encrypted representation of the password for enhanced security; Salt, a unique value used in conjunction with the PasswordHash to further protect user credentials; and Groups, which assigns the user to specific categories or roles. This step is vital for expanding the database with new user accounts.

Users Table Data Validation



```
SELECT * FROM dbo.Users;
```

Id	Username	PasswordHash	Salt	Groups

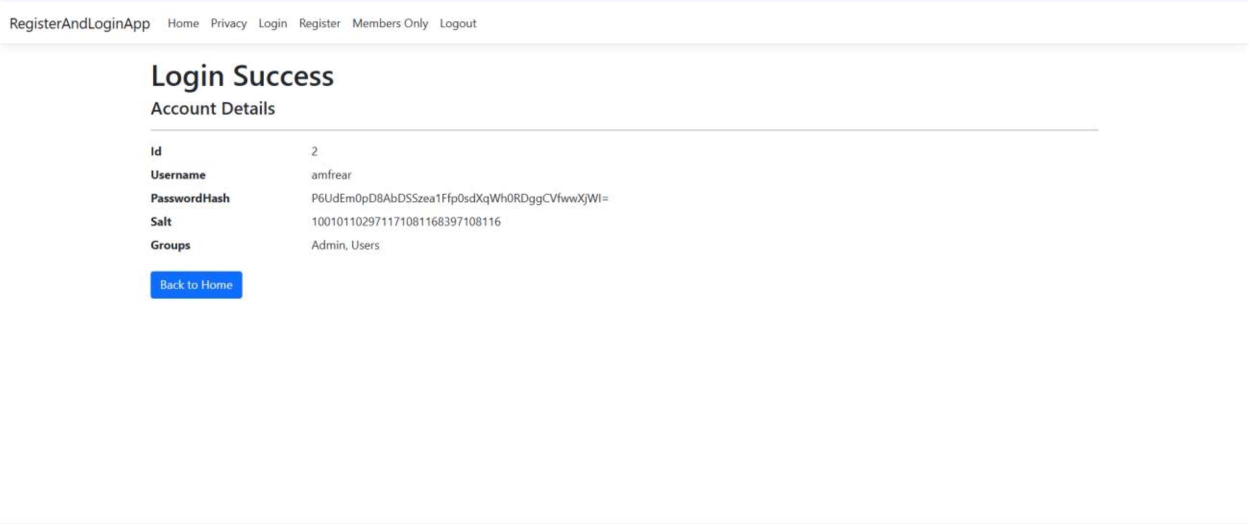
```
Query executed successfully at 2:43:50 PM    | (localdb)\ProjectModels (15... | FREARPCLAPTOP\af200 (52) | Test | 00:00:00 | 0 rows
```

Data Tools Operations

```
Update for (localdb)\ProjectModels.Test  2:37:44 PM - 2:38:09 PM (0:00:24)
  ✓ Creating update preview...
  ✓ Displaying update preview...
  ✓ Creating database script...
  ✓ Executing update script on database 'Test'...
    Update completed successfully
Displaying update preview...
```

Figure 3: This illustration captures the validation phase, showcasing the execution of the SELECT * FROM dbo.Users query. This query serves as a crucial check, confirming that the newly entered user information has been accurately recorded in the database. The results displayed include all relevant fields—such as Id, Username, PasswordHash, Salt, and Groups—providing a comprehensive view of the stored data. This step ensures data integrity and confirms the successful addition of user records.

New Registered User - Login Success



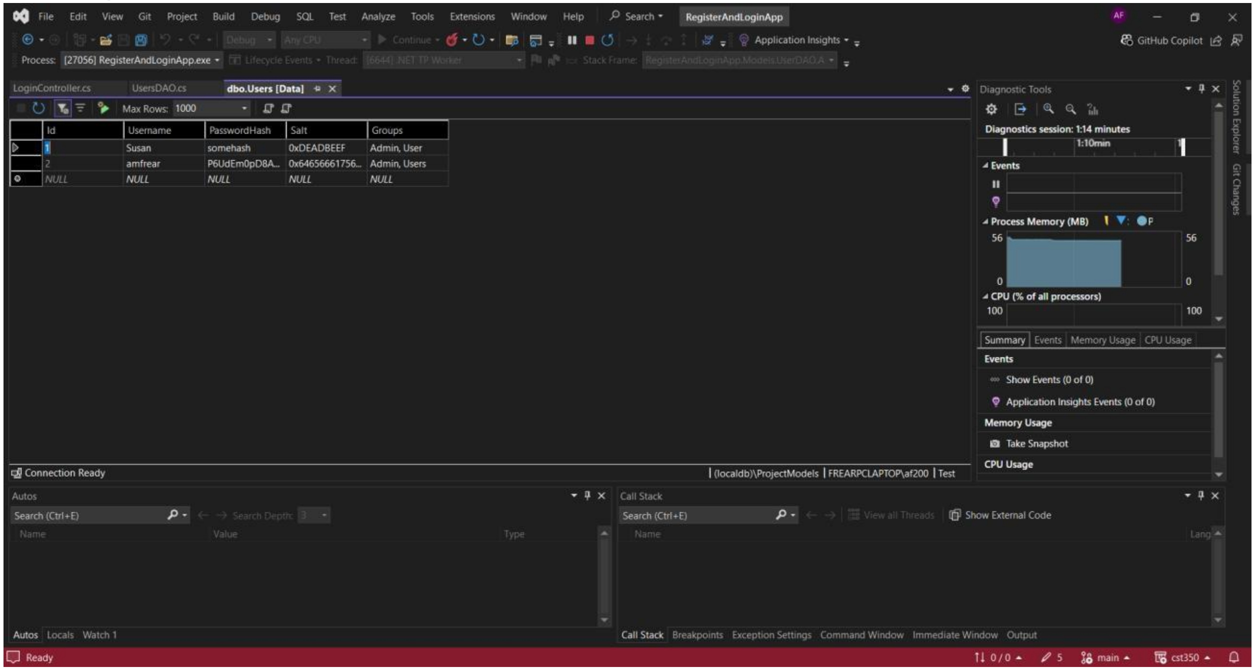
Login Success

Account Details

Id	2
Username	amfrear
PasswordHash	P6UdEm0pD8AbDSSzea1Ffp0sdXqWh0RDggCVfwwXjWI=
Salt	100101102971171081168397108116
Groups	Admin, Users

Figure 4: This image illustrates a successful login by the newly registered user. It prominently displays key account details, including the Username for identification, the PasswordHash representing the encrypted password, the Salt used for added security, and the Groups assigned to the user, indicating their role within the system. This visual not only confirms the user's access but also highlights the essential components of their account, demonstrating the effectiveness of the registration and authentication processes.

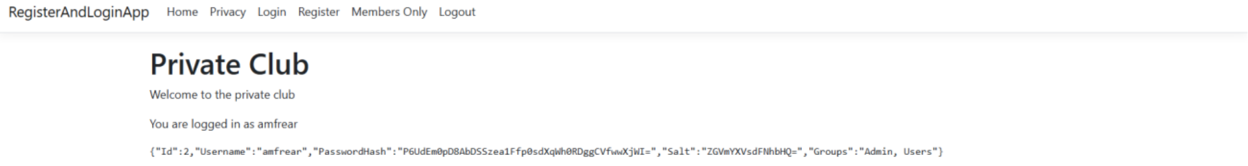
New Registered User in Database



Id	Username	PasswordHash	Salt	Groups
1	Susan	somehash	0xDEADBEEF	Admin, User
2	amfrear	P6UdEm0pD8A...	0x64656661756...	Admin, Users
NULL	NULL	NULL	NULL	NULL

***Figure 5**: This image illustrates the verification of the new user entry within the database. It confirms that the registration functionality has successfully added the user details to the Users table. The visual representation shows the relevant fields—Id, Username, PasswordHash, Salt, and Groups—demonstrating that the newly registered user's information is accurately stored and accessible. This step is crucial for ensuring that the registration process functions correctly and maintains the integrity of the database.*

Members Only Page Access by New User



Private Club

Welcome to the private club

You are logged in as amfrear

{"Id":2,"Username":"amfrear","PasswordHash":"P6UdEm0pD8AbDSSzea1Ffp0sdXqWh0RDggCVfwwXjWI=","Salt":"ZGVmYXVsdFNhbHQ=","Groups":"Admin, Users"}

***Figure 6**: This screenshot demonstrates that the newly registered user has successfully accessed the Members-Only page. It visually confirms the session validation process, indicating that the user's credentials have been authenticated and that they possess the necessary permissions for restricted access. The interface highlights the exclusive content available to members, underscoring the effectiveness of the login and user authentication mechanisms in controlling access to sensitive information. This step is essential for maintaining user security and ensuring that only authorized individuals can view the protected resources.*

Summary of Key Concepts (Part 1)

In Part 1 of this activity, I embarked on the development of a secure login and registration system integrated with a SQL Server database. This multifaceted project encompassed several critical components:

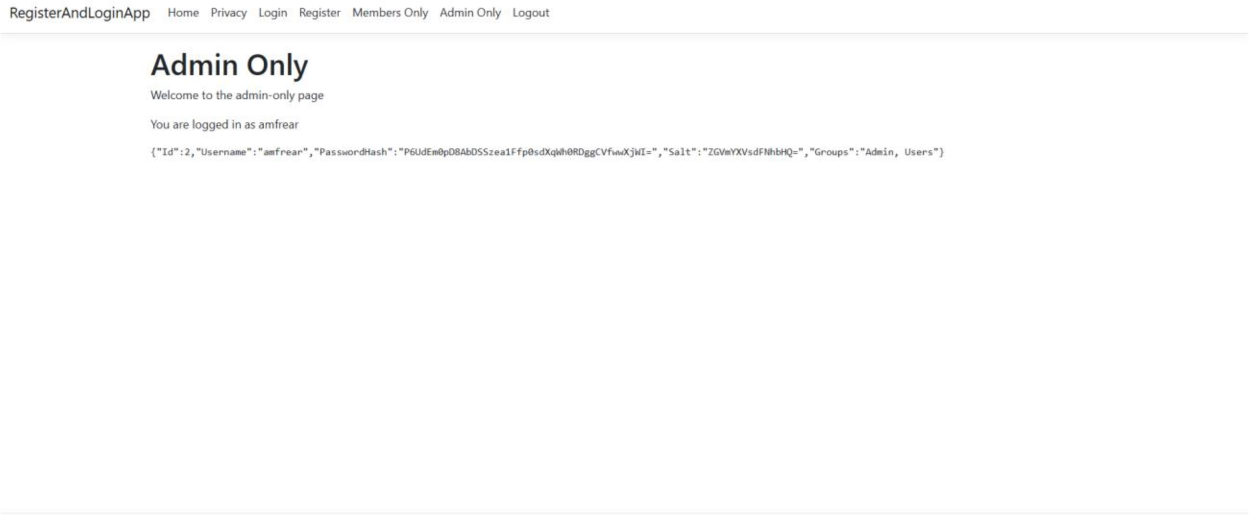
1. **Database Architecture**:
    - I meticulously designed and set up a **Users table** within the SQL Server environment. This table was structured to securely store essential user information, including:
        - **Username**: A unique identifier for each user.
        - **PasswordHash**: A securely hashed version of the user's password, ensuring that sensitive information is not stored in plain text.
        - **Salt**: An additional layer of security, this unique value is combined with the password before hashing to thwart common attacks like rainbow table attacks.
        - **Groups**: This field categorizes users into specific roles or permissions, facilitating role-based access control.
2. **C# Implementation**:
    - I implemented robust **SQL commands** in C# to handle user data operations. This included:
        - **Data Insertion**: Crafting SQL INSERT statements to add new user records to the database securely.
        - **Validation Queries**: Writing SELECT queries to verify user credentials during the login process, ensuring that only authorized users could gain access.
3. **Security Enhancements**:
    - To bolster the security of user credentials, I developed a comprehensive **registration and login system** that included:
        - **Password Hashing**: Utilizing a strong hashing algorithm to convert user passwords into secure hashes.
        - **Salting**: Generating unique salts for each password to enhance security further, making it significantly harder for attackers to compromise user accounts.
4. **Session Management and Access Control**:
    - I created a **Members-Only page** that employs sophisticated **session-based access control**. This feature ensures that:
        - Only authenticated users can access sensitive content, effectively protecting the integrity of user data and maintaining privacy.
        - Session management is carefully handled to track user logins, providing a seamless yet secure user experience.
5. **Acquired Skills and Experience**:
    - Through this activity, I gained invaluable practical experience in:
        - Managing user data within a relational database, including the intricacies of SQL Server.
        - Implementing essential security practices to safeguard user information.
        - Enforcing authentication requirements to control access to protected web pages, ensuring that only valid users can interact with sensitive features.

This project not only solidified my understanding of database management and web security practices but also equipped me with the skills necessary to build secure, user-oriented applications. I look forward to expanding on this foundation in future stages of development.

Part 2: User Group Access

Screenshots

Admin Only Page



Admin Only

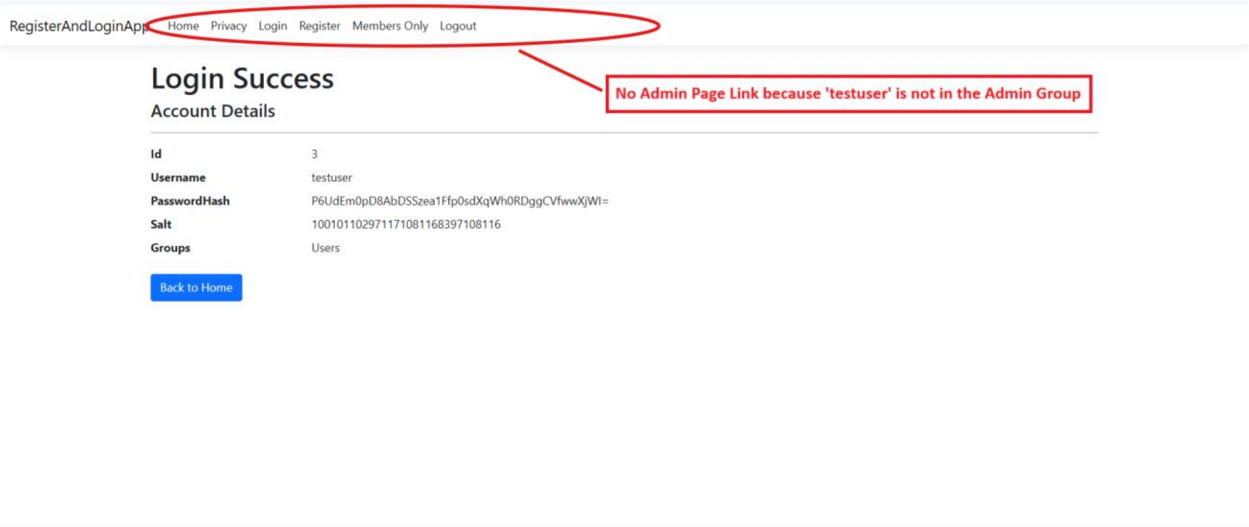
Welcome to the admin-only page

You are logged in as amfrear

```
{"Id":2,"Username":"amfrear","PasswordHash":"P6UdEm0pD8AbDSSzea1Ffp0sdXqWh0RDggCVfwwXjWI=","Salt":"ZGVmYXVsdFNhbHQ=","Groups":"Admin, Users"}
```

Figure 7*: This screenshot showcases the Admin-Only page, highlighting its accessibility exclusively to users who belong to the Admin group. The visual emphasizes the implementation of group-based access control, ensuring that only authorized personnel can view and interact with this sensitive section of the application. This feature not only reinforces security but also streamlines user permissions, demonstrating an effective approach to managing different user roles within the system. The content displayed on this page is tailored for administrative tasks, further underscoring the importance of access restrictions in safeguarding critical functionalities.*

User Only Access Restriction



No Admin Page Link because 'testuser' is not in the Admin Group

Login Success

Account Details

Id	3
Username	testuser
PasswordHash	P6UdEm0pD8AbDSSzea1Ffp0sdXqWh0RDggCVfwwXjWI=
Salt	100101102971171081168397108116
Groups	Users

Figure 8*: This screenshot illustrates that a regular user, who does not belong to the Admin group, is effectively restricted from accessing the Admin-Only page. This visual reinforces the implementation of role-based access control within the system, highlighting the security measures in place to prevent unauthorized access to sensitive administrative features. The display of a warning or error message serves as a clear indication that the user lacks the necessary permissions, emphasizing the importance of safeguarding administrative functionalities and maintaining the integrity of user roles within the application. This approach ensures that only users with appropriate authority can engage with critical system operations.*

Summary of Key Concepts (Part 2)

In Part 2 of this activity, I focused on enhancing the security of my application through the implementation of role-based access control, specifically by developing an **Admin-Only page**. This page is exclusively accessible to users who are members of the Admin group, ensuring that sensitive functionalities are shielded from unauthorized access.

To achieve this, I created a **custom action filter** named `AdminCheckFilter`. This powerful component plays a critical role in verifying user permissions by inspecting the session data for group membership. When a user attempts to access the Admin-Only page, the filter checks if the user belongs to the Admin group.

If the user is not authenticated as an admin, the filter seamlessly redirects them back to the login page, preventing any unauthorized access attempts. This mechanism not only enhances security but also improves user experience by providing immediate feedback regarding access restrictions.

Throughout this process, I gained invaluable insights into the use of custom filters within **ASP.NET Core MVC**. This experience deepened my understanding of how to enforce group-specific access restrictions effectively, allowing me to create a more secure and structured application. By implementing role-based access control, I ensured that administrative features remain safeguarded, thereby maintaining the integrity of sensitive operations and user data within the system. This endeavor significantly contributed to my skills in web application security and reinforced best practices in user management.