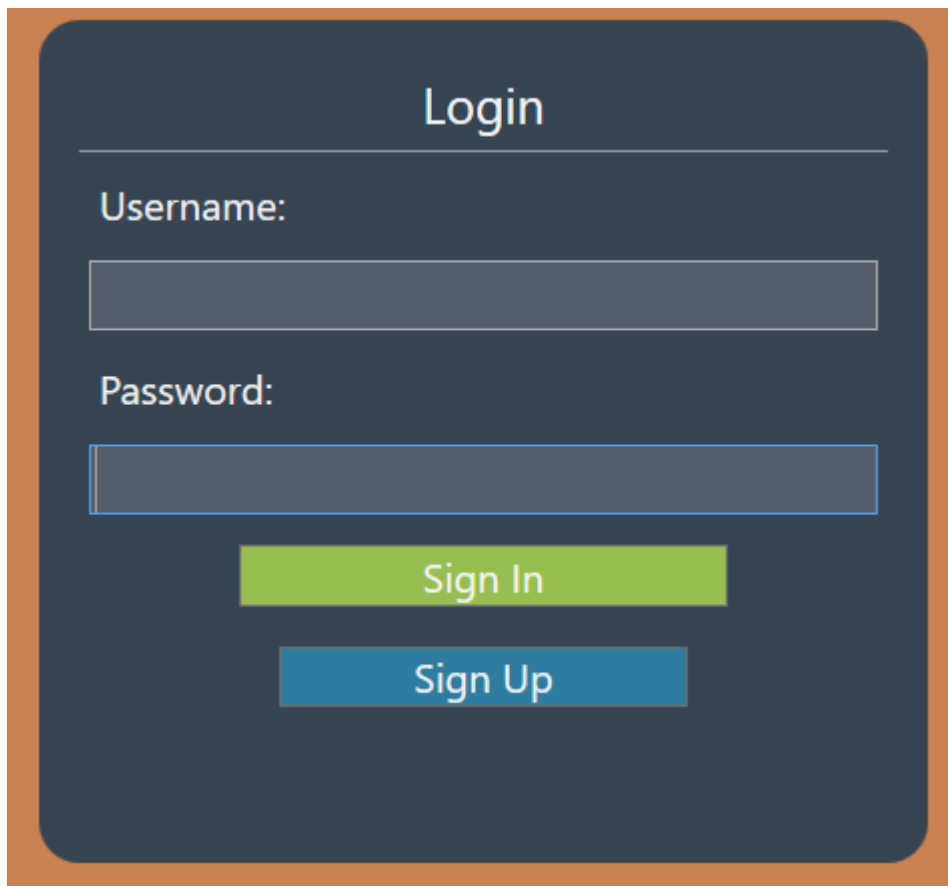


# To Do List Application

This is a simple application that allows the user login to its account, create a list and put items on it. The application give opportunity the user for creating multiple lists and modify them. In addition, the user can filter items by its status and order them by their name, created date, deadline or status. When the created item is completed, it can be marked as completed and its status changes as well. Moreover, the items and lists can be deleted easily. One of the features of the application is that if a list deleted, its items also deleted automatically.

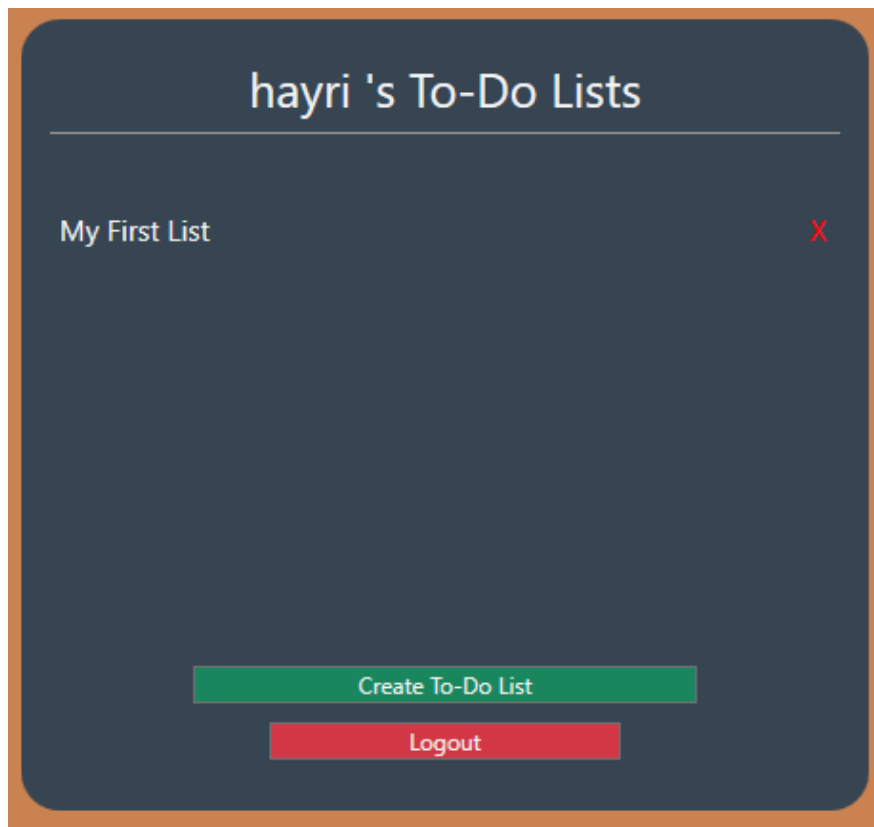
## Getting Started

A screenshot of a login window with a dark blue background and an orange border. The window has a title bar with the word "Login" in white. Below the title bar, there are two input fields: "Username:" and "Password:". The "Username:" field is a simple text box, and the "Password:" field is a text box with a blue border. Below the password field, there are two buttons: a green "Sign In" button and a blue "Sign Up" button.

This is the Login Window in order to login or register, thanks to “Sign In” and “Sign Up” buttons to the application. Before click these buttons, required fields which are username and password should be filled. I use “System.Security.Cryptography” library and prefer its “SHA256” class to hold the passwords encrypted.

```
private static string ComputeSha256Hash( string rawData )
{
    // Create a SHA256
    using ( SHA256 sha256Hash = SHA256.Create() )
    {
        // ComputeHash - returns byte array
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

        // Convert byte array to a string
        StringBuilder stringBuilder = new StringBuilder();
        for ( int i = 0; i < bytes.Length; i++ )
        {
            stringBuilder.Append( bytes[i].ToString( "x2" ) );
        }
        return stringBuilder.ToString();
    }
}
```



After login, in this window, the user can see its username on title and existing lists on body. The Control Objects created dynamically when the window constructor called.

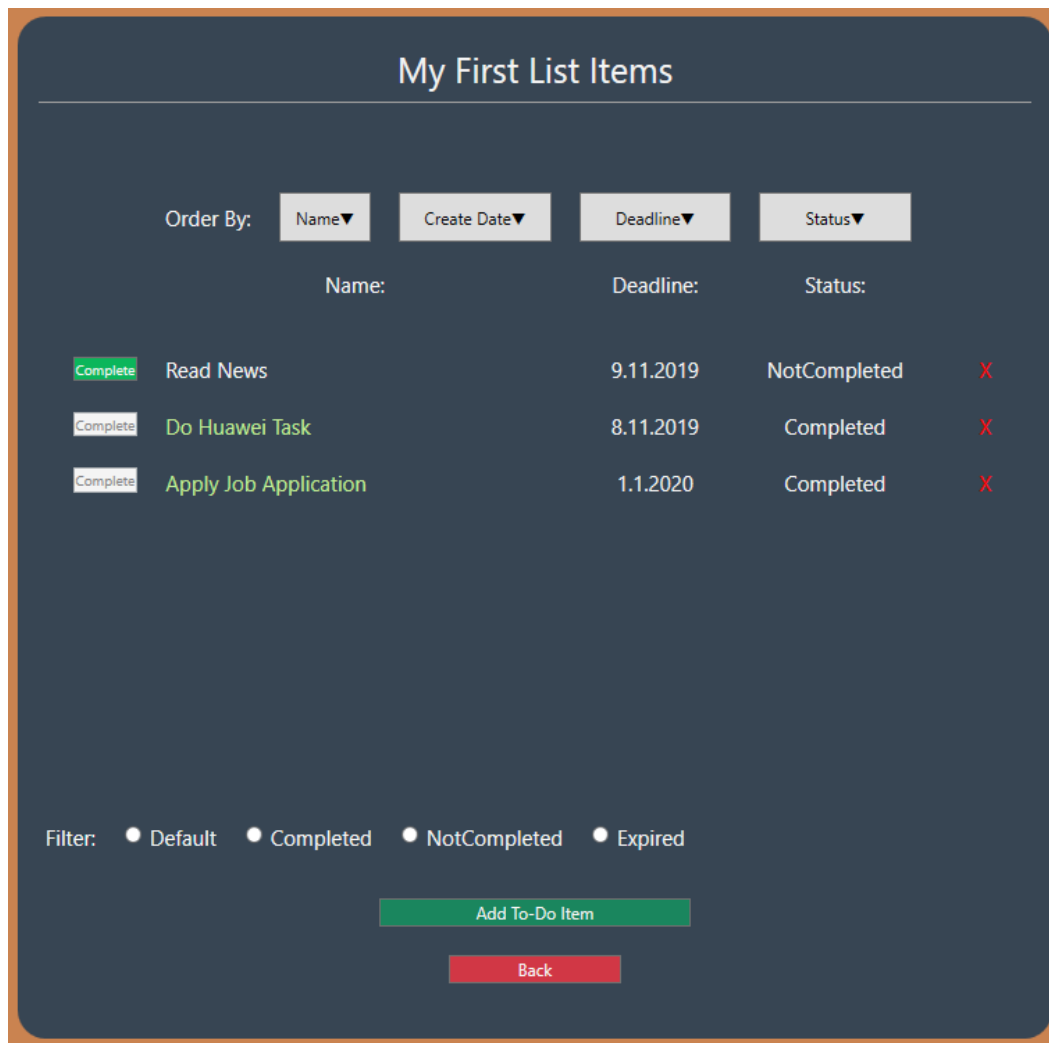
```
private void FillBody()
{
    List<ToDoList> lists = new List<ToDoList>();
    lists = process.GetLists();

    if ( lists.Count > 0 )
    {
        foreach ( ToDoList list in lists )
        {
            Label listNameLabel = new Label()
            {
                Content=list.listName.ToString(),
                Style = (Style)FindResource("nameLabelStyle"),
                Tag = list
            };
            this.listNameStackPanel.Children.Add( listNameLabel );

            Label xLabel = new Label()
            {
                Style = (Style)FindResource("deleteLabelStyle"),
                Tag=list
            };
            this.deleteIconStackPanel.Children.Add( xLabel );

            xLabel.MouseDown += XLabel_MouseDown;
            listNameLabel.MouseDown += ListNameLabel_MouseDown;
        }
    }
}
```

The user can easily log out when click “Logout” and returns to Login Window. If click on a list, next window which has its items going to be appear.

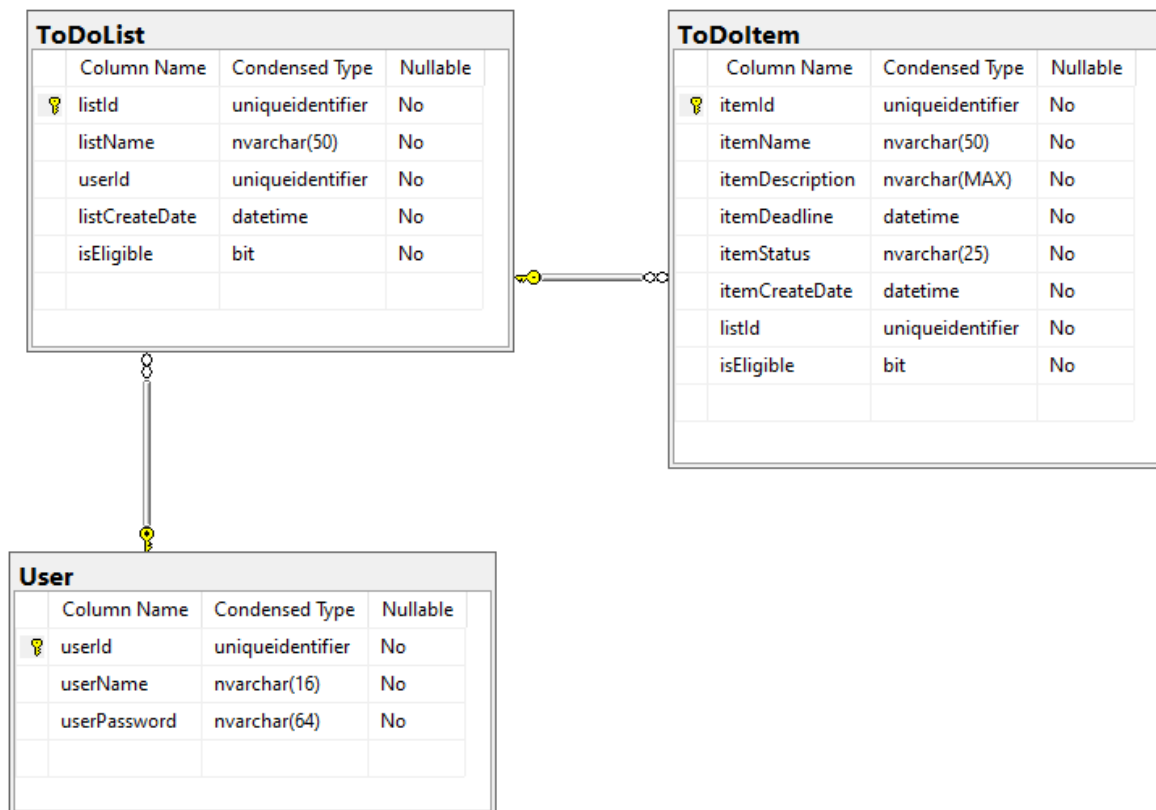


This window shows up the items from selected list. They can be deleted, ordered, filtered and marked as completed. New items will be part of the list's item after they added and they will be appeared immediately. In addition, the descriptions can be checked when the click on items' name.

```
public List<ToDoItem> OrderItems( OrderBySelection orderBySelection )
{
    List<ToDoItem> orderedItems = new List<ToDoItem>();
    ToDoItemRepository toDoItemRepository = new ToDoItemRepository();

    switch ( orderBySelection )
    {
        case OrderBySelection.CreateDate:
            orderedItems = toDoItemRepository.OrderByCreateDate( selectedList.listId );
            break;
        case OrderBySelection.Deadline:
            orderedItems = toDoItemRepository.OrderByDeadline( selectedList.listId );
            break;
        case OrderBySelection.Name:
            orderedItems = toDoItemRepository.OrderByName( selectedList.listId );
            break;
        case OrderBySelection.Status:
            orderedItems = toDoItemRepository.OrderByStatus( selectedList.listId );
            break;
        default:
            break;
    }
    return orderedItems;
}
```

## Database Diagram



## Techniques and technologies

- 3 Tier Architecture
- Singleton Design Pattern
- Microsoft Visual Studio 2017
- Entity Framework
- Microsoft SQL Server Management Studio 2017
- ADO.NET Entity Data Model
- Azure SQL Server