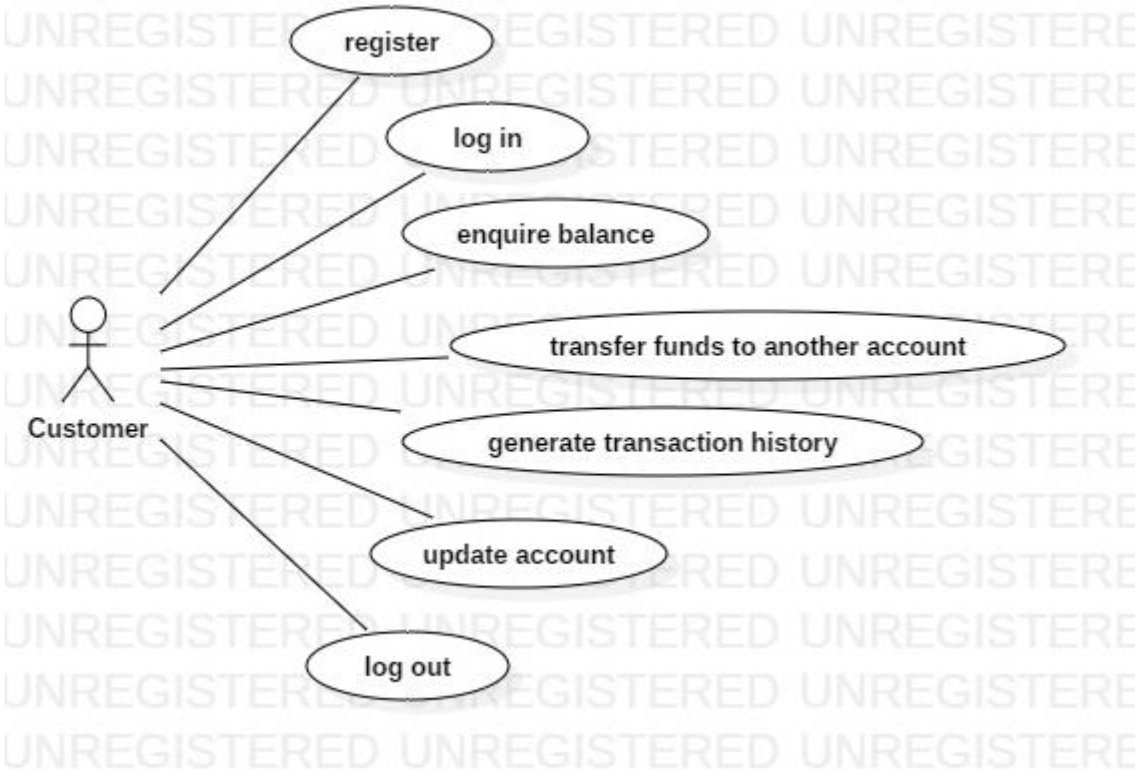
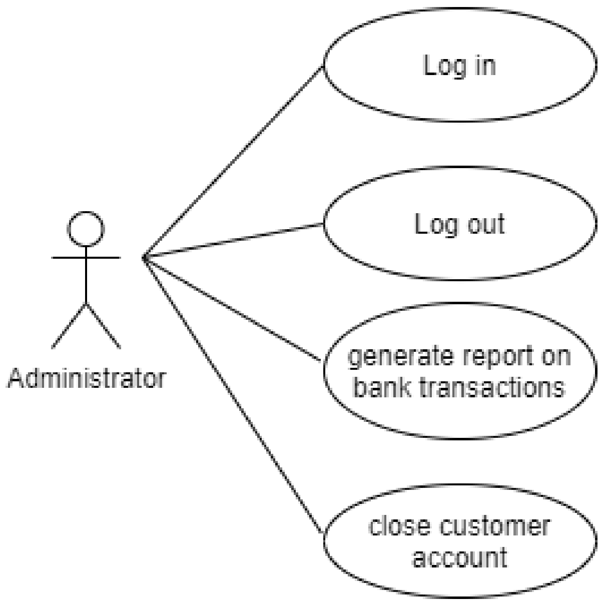
Software Engineering
Online Banking
C# Project

Team: Chereji Iulia and Iepure Ana
Group: 30434-1
Author: Chereji Iulia

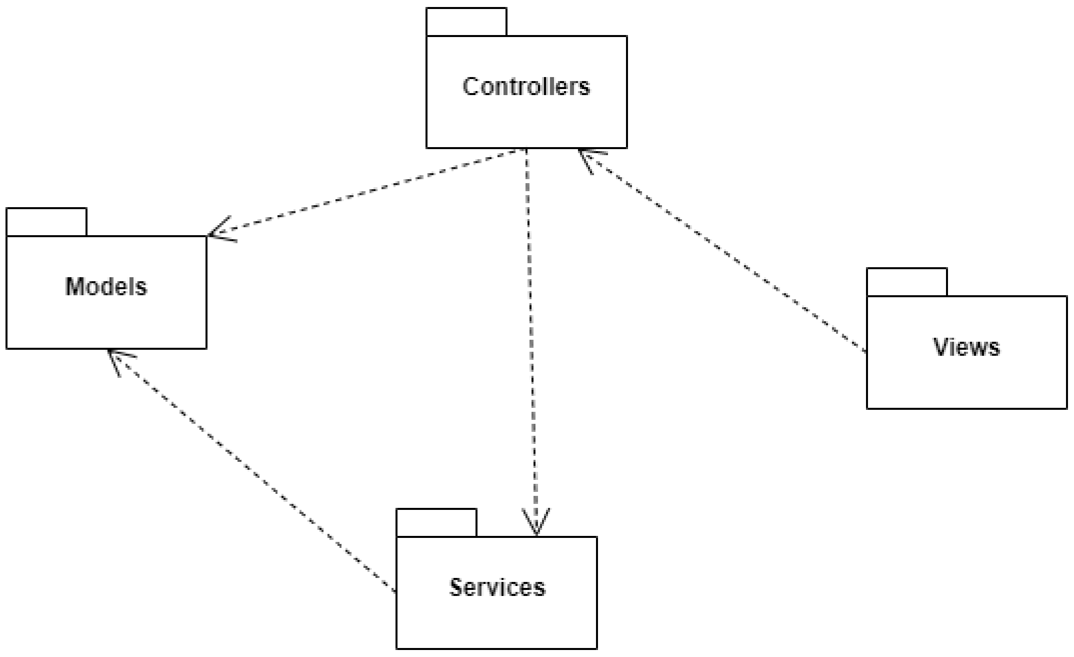
Contents

1. Use Case Diagram

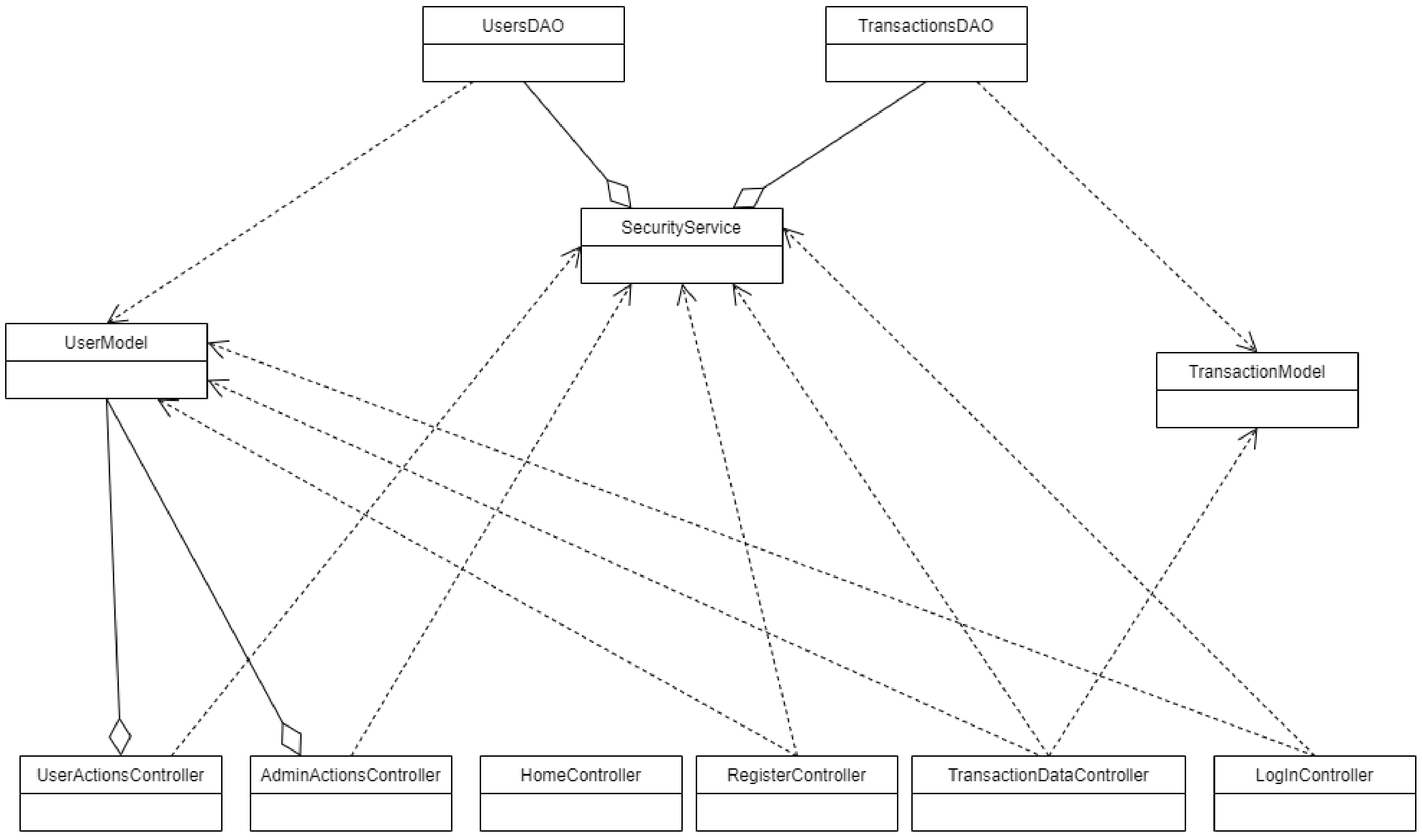

register
log in
enquire balance
transfer funds to another account
generate transaction history
update account
log out
Customer

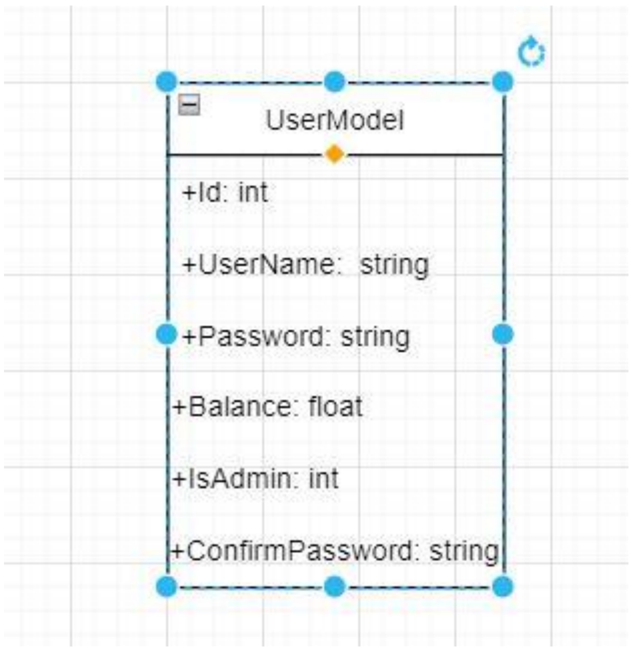


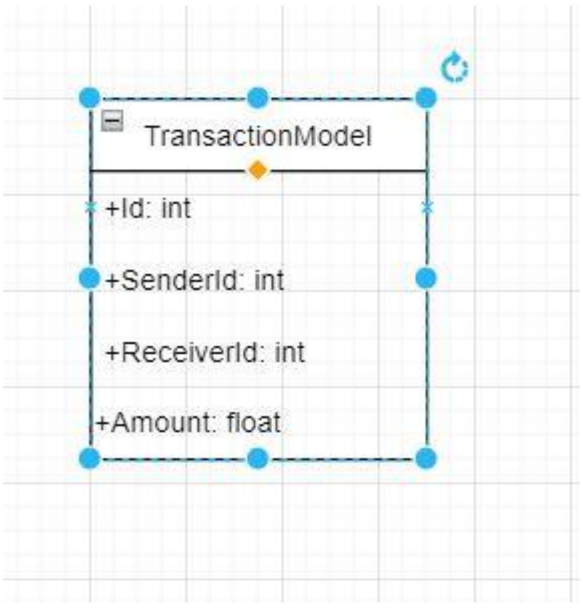
2. Package Diagram

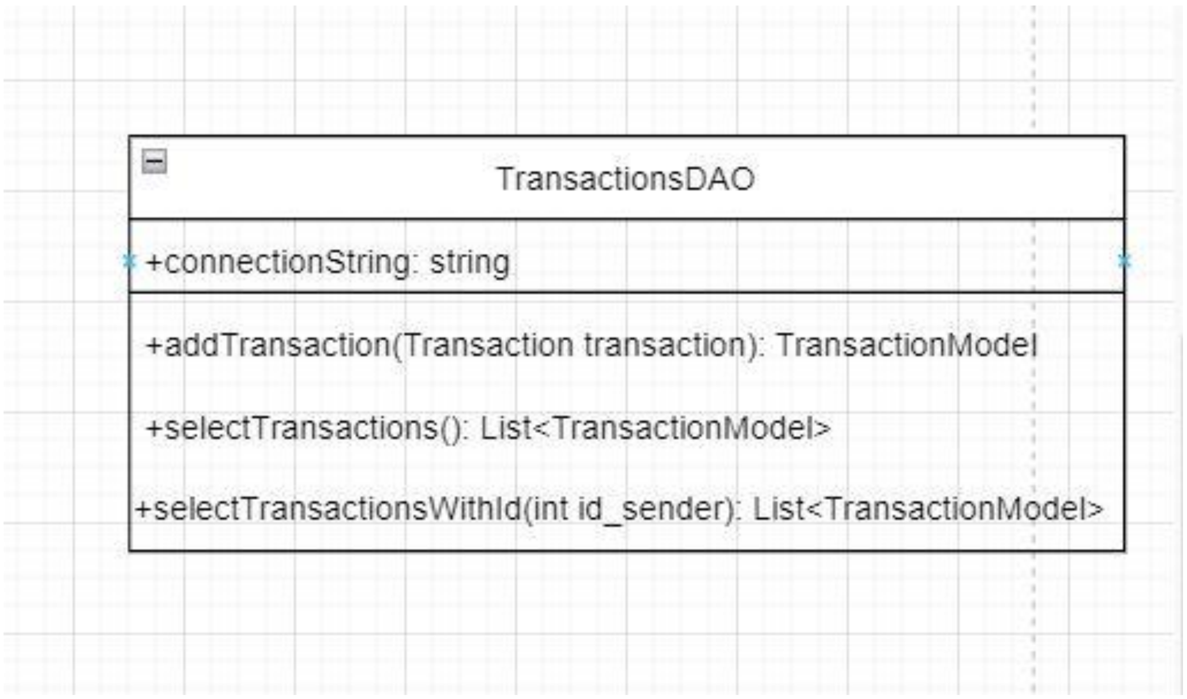


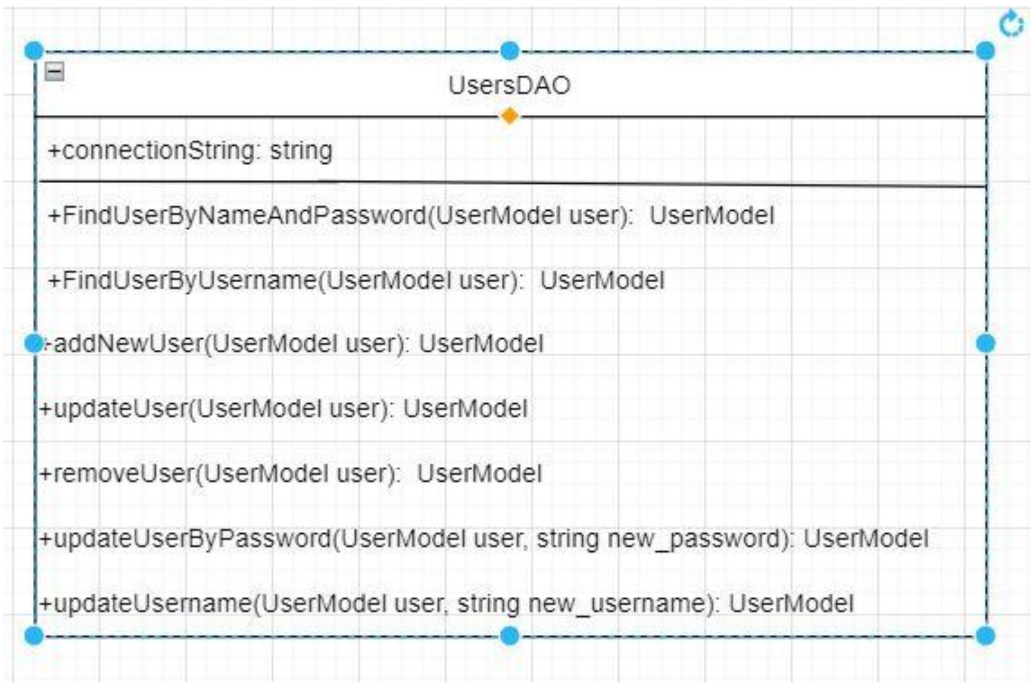
3. Class Diagram

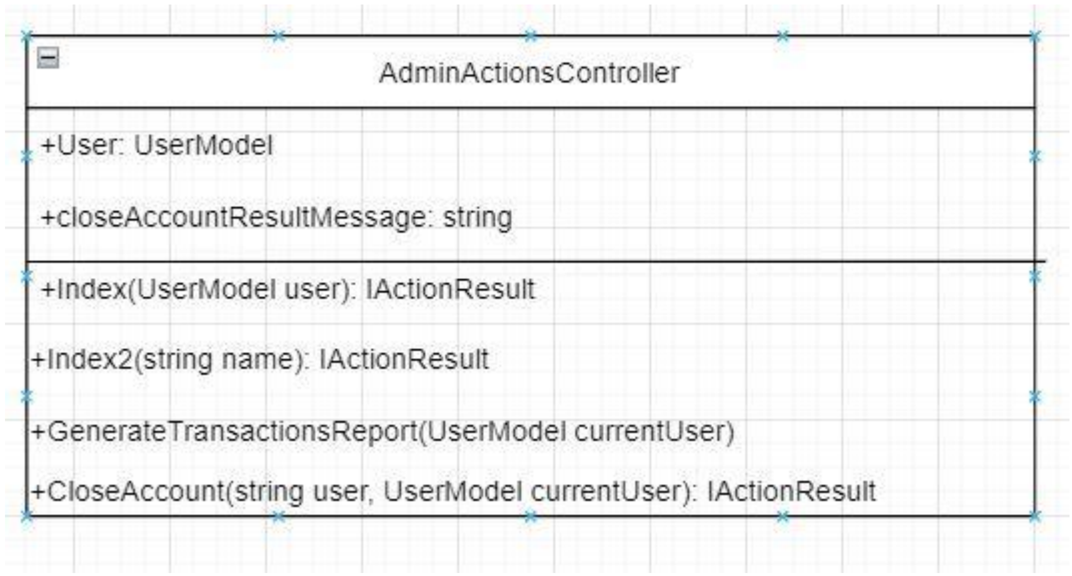


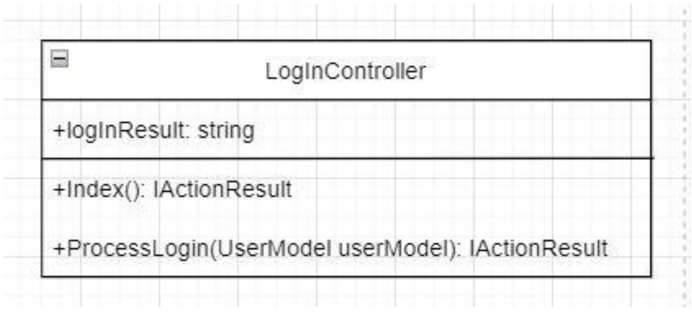

UserModel
+Id: int
+UserName: string
+Password: string
+Balance: float
+IsAdmin: int
+ConfirmPassword: string

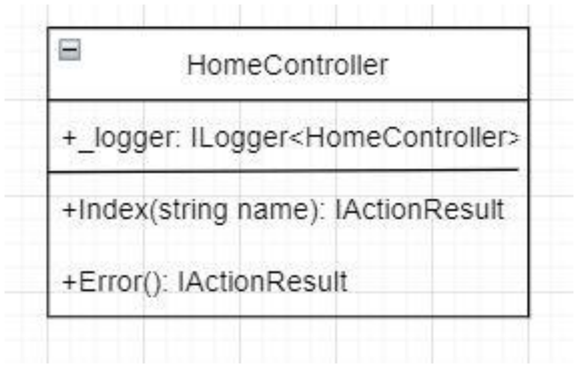

TransactionModel
+Id: int
+SenderId: int
+ReceiverId: int
+Amount: float

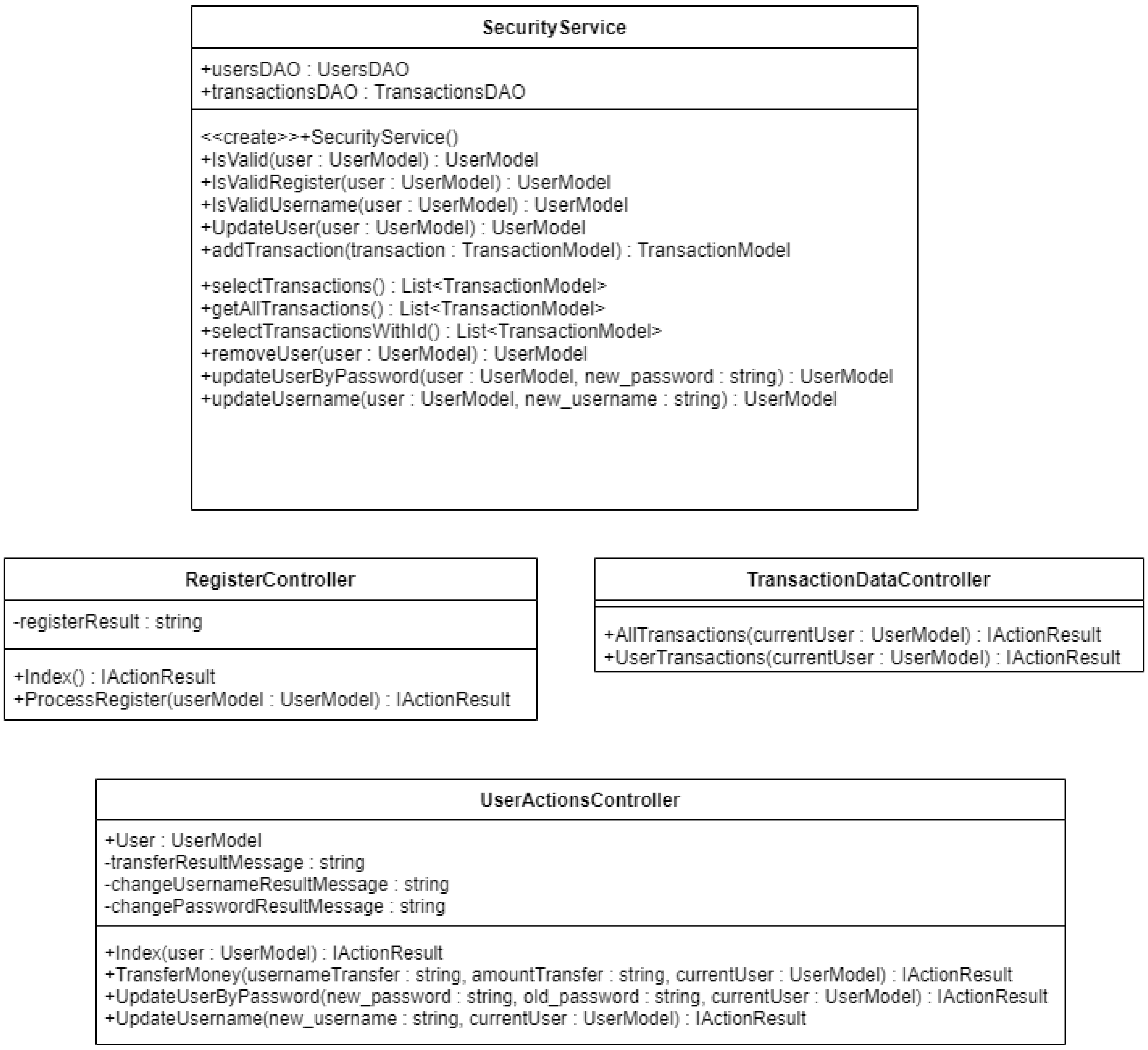

TransactionsDAO
+connectionString: string
+addTransaction(Transaction transaction): TransactionModel
+selectTransactions(): List<TransactionModel>
+selectTransactionsWithId(int id_sender): List<TransactionModel>


UsersDAO
+connectionString: string
+FindUserByNameAndPassword(UserModel user): UserModel
+FindUserByUsername(UserModel user): UserModel
+addNewUser(UserModel user): UserModel
+updateUser(UserModel user): UserModel
+removeUser(UserModel user): UserModel
+updateUserByPassword(UserModel user, string new_password): UserModel
+updateUsername(UserModel user, string new_username): UserModel


AdminActionsController
+User: UserModel
+closeAccountResultMessage: string
+Index(UserModel user): IActionResult
+Index2(string name): IActionResult
+GenerateTransactionsReport(UserModel currentUser)
+CloseAccount(string user, UserModel currentUser): IActionResult


LogInController
+logInResult: string
+Index(): IActionResult
+ProcessLogin(UserModel userModel): IActionResult


HomeController
+_logger: ILogger<HomeController>
+Index(string name): IActionResult
+Error(): IActionResult



SecurityService

```
+usersDAO : UsersDAO
+transactionsDAO : TransactionsDAO
```

```
<<create>>+SecurityService()
+IsValid(user : UserModel) : UserModel
+IsValidRegister(user : UserModel) : UserModel
+IsValidUsername(user : UserModel) : UserModel
+UpdateUser(user : UserModel) : UserModel
+addTransaction(transaction : TransactionModel) : TransactionModel

+selectTransactions() : List<TransactionModel>
+getAllTransactions() : List<TransactionModel>
+selectTransactionsWithId() : List<TransactionModel>
+removeUser(user : UserModel) : UserModel
+updateUserByPassword(user : UserModel, new_password : string) : UserModel
+updateUsername(user : UserModel, new_username : string) : UserModel
```

RegisterController

```
-registerResult : string
```

```
+Index() : IActionResult
+ProcessRegister(userModel : UserModel) : IActionResult
```

TransactionDataController

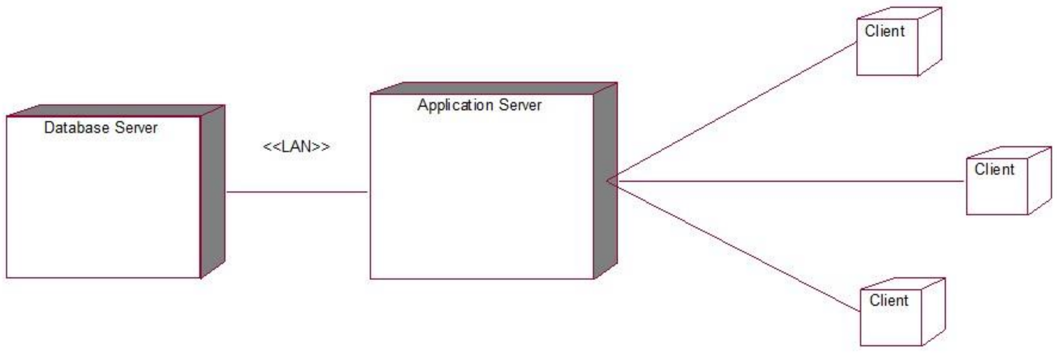
```
+AllTransactions(currentUser : UserModel) : IActionResult
+UserTransactions(currentUser : UserModel) : IActionResult
```

UserActionsController

```
+User : UserModel
-transferResultMessage : string
-changeUsernameResultMessage : string
-changePasswordResultMessage : string
```

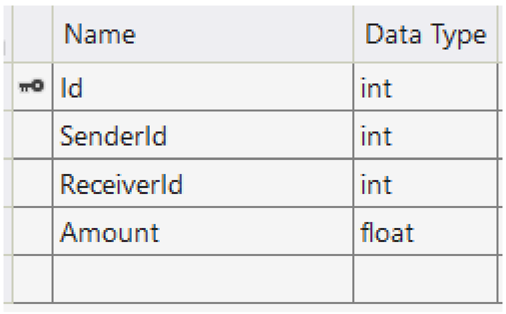
```
+Index(user : UserModel) : IActionResult
+TransferMoney(usernameTransfer : string, amountTransfer : string, currentUser : UserModel) : IActionResult
+UpdateUserByPassword(new_password : string, old_password : string, currentUser : UserModel) : IActionResult
+UpdateUsername(new_username : string, currentUser : UserModel) : IActionResult
```

4. Deployment Diagram



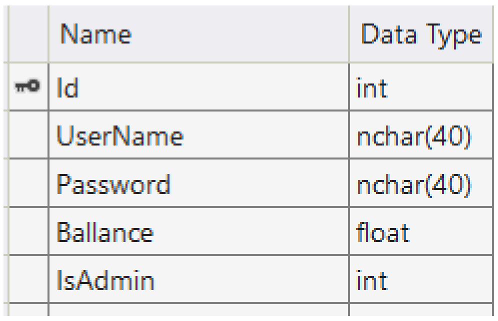
5. Database tables

5.1. Transactions table



	Name	Data Type
🔑	Id	int
	SenderId	int
	ReceiverId	int
	Amount	float

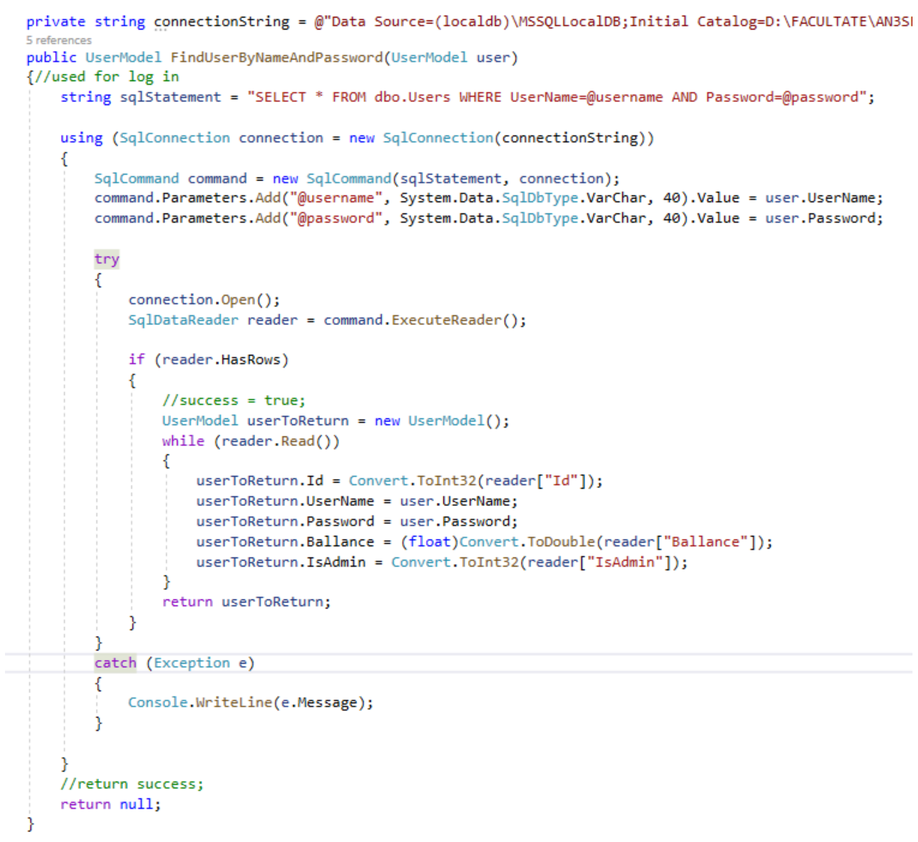
5.2. Users table



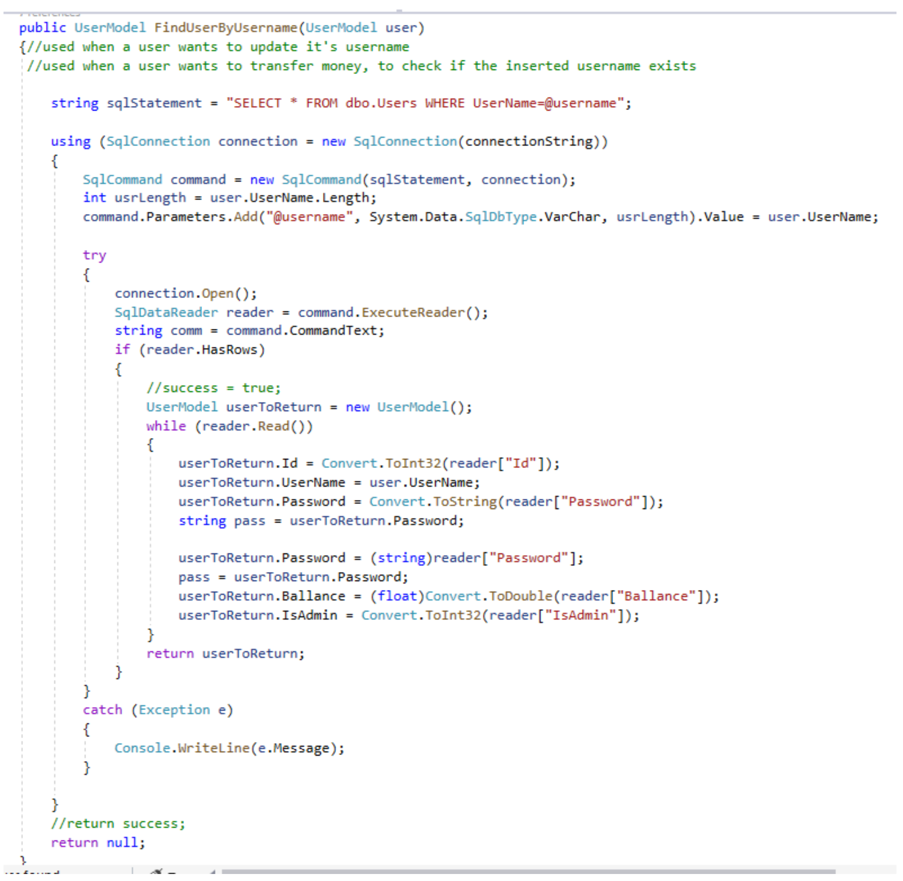
	Name	Data Type
🔑	Id	int
	UserName	nchar(40)
	Password	nchar(40)
	Ballance	float
	IsAdmin	int

6. Code

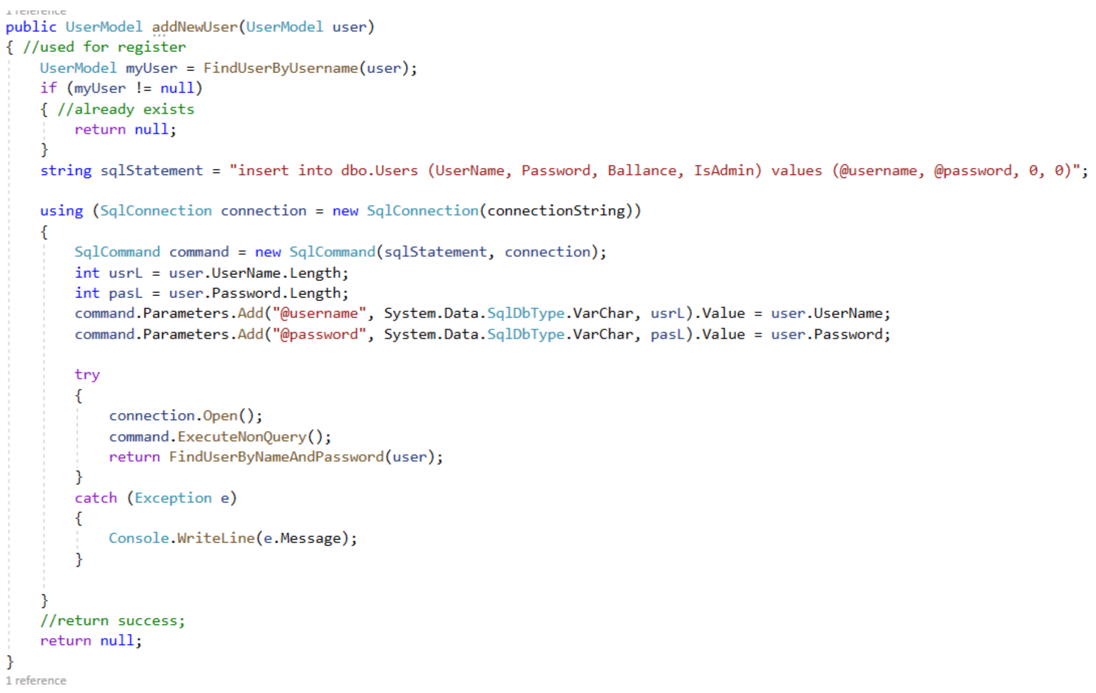
6.1. UsersDAO class



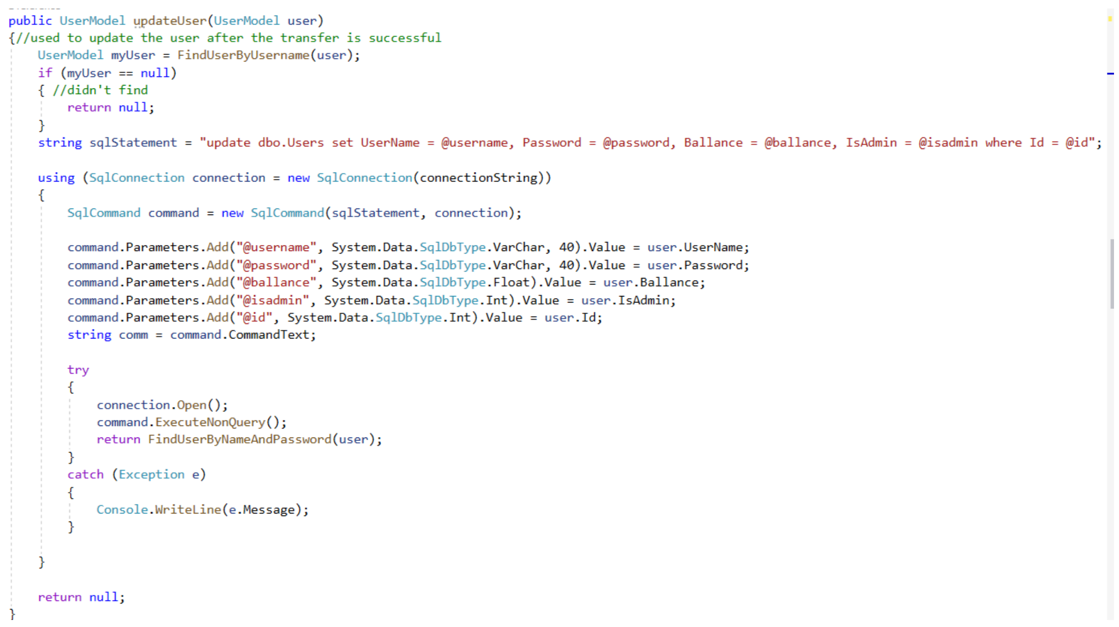
```
private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=D:\FACULTATE\AN3SI
5 references
public UserModel FindUserByNameAndPassword(UserModel user)
{//used for log in
    string sqlStatement = "SELECT * FROM dbo.Users WHERE UserName=@username AND Password=@password";

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);
        command.Parameters.Add("@username", System.Data.SqlDbType.VarChar, 40).Value = user.UserName;
        command.Parameters.Add("@password", System.Data.SqlDbType.VarChar, 40).Value = user.Password;

        try
        {
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            if (reader.HasRows)
            {
                //success = true;
                UserModel userToReturn = new UserModel();
                while (reader.Read())
                {
                    userToReturn.Id = Convert.ToInt32(reader["Id"]);
                    userToReturn.UserName = user.UserName;
                    userToReturn.Password = user.Password;
                    userToReturn.Ballance = (float)Convert.ToDouble(reader["Ballance"]);
                    userToReturn.IsAdmin = Convert.ToInt32(reader["IsAdmin"]);
                }
                return userToReturn;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }

    }
    //return success;
    return null;
}
```



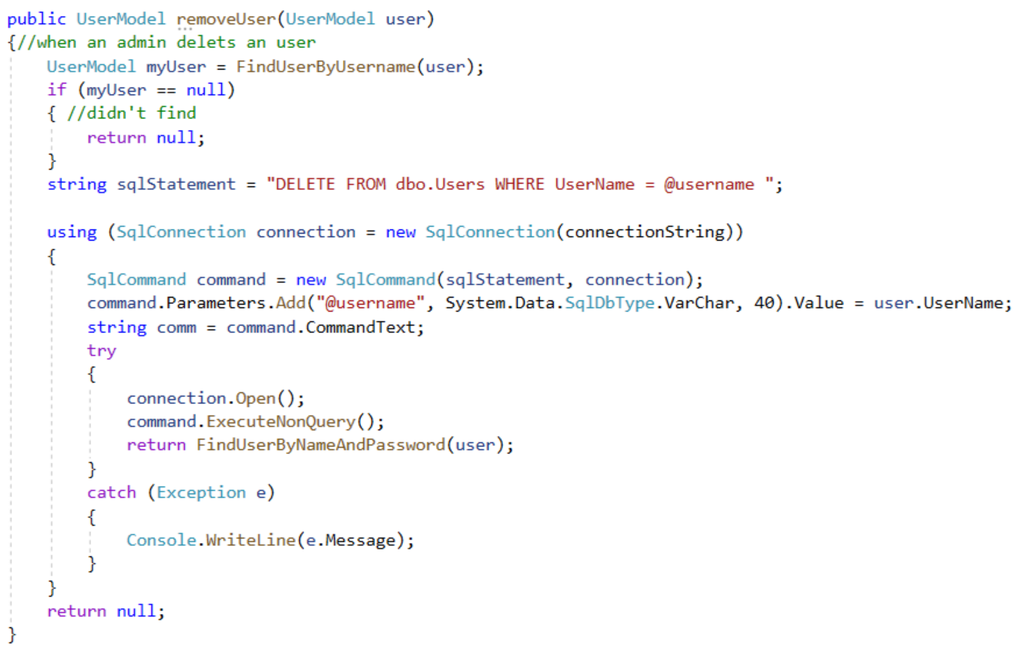
```
public UserModel FindUserByUsername(UserModel user)
{//used when a user wants to update it's username
 //used when a user wants to transfer money, to check if the inserted username exists

    string sqlStatement = "SELECT * FROM dbo.Users WHERE UserName=@username";

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);
        int usrLength = user.UserName.Length;
        command.Parameters.Add("@username", System.Data.SqlDbType.VarChar, usrLength).Value = user.UserName;

        try
        {
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();
            string comm = command.CommandText;
            if (reader.HasRows)
            {
                //success = true;
                UserModel userToReturn = new UserModel();
                while (reader.Read())
                {
                    userToReturn.Id = Convert.ToInt32(reader["Id"]);
                    userToReturn.UserName = user.UserName;
                    userToReturn.Password = Convert.ToString(reader["Password"]);
                    string pass = userToReturn.Password;

                    userToReturn.Password = (string)reader["Password"];
                    pass = userToReturn.Password;
                    userToReturn.Ballance = (float)Convert.ToDouble(reader["Ballance"]);
                    userToReturn.IsAdmin = Convert.ToInt32(reader["IsAdmin"]);
                }
                return userToReturn;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }

    }
    //return success;
    return null;
}
```



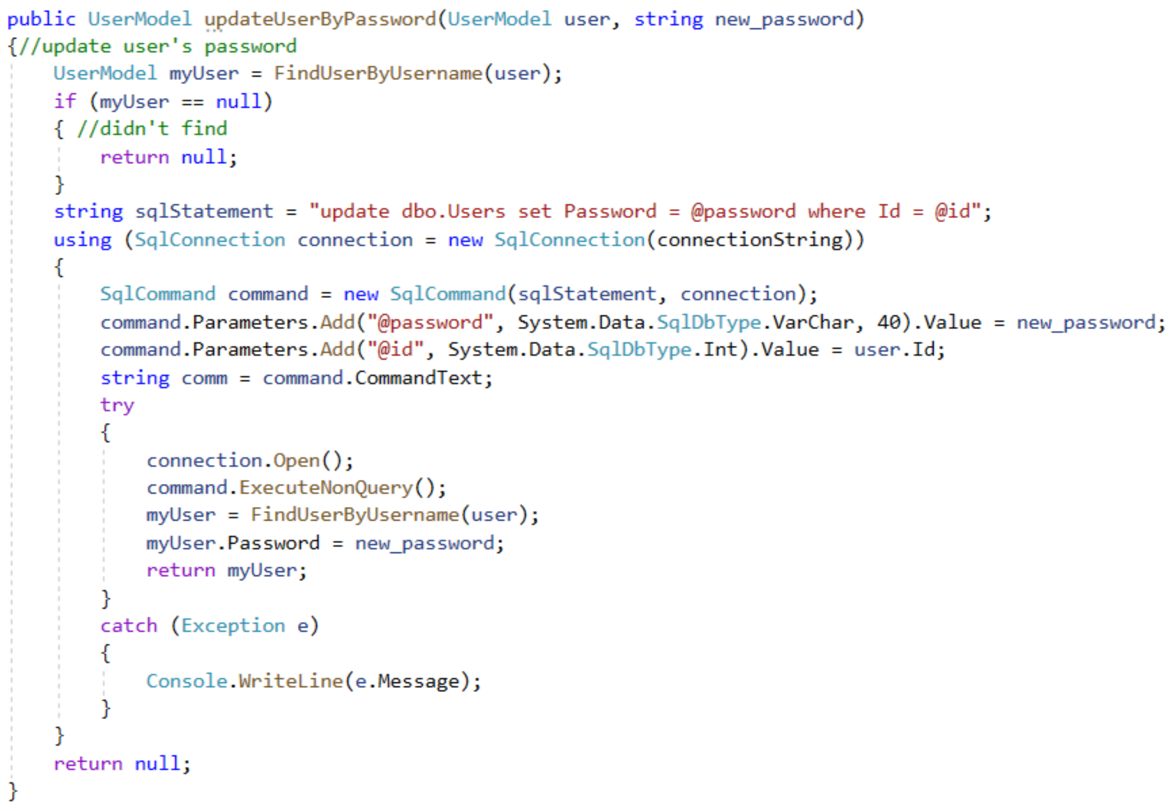
```
public UserModel addNewUser(UserModel user)
{ //used for register
    UserModel myUser = FindUserByUsername(user);
    if (myUser != null)
    { //already exists
        return null;
    }
    string sqlStatement = "insert into dbo.Users (UserName, Password, Ballance, IsAdmin) values (@username, @password, 0, 0)";

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);
        int usrL = user.UserName.Length;
        int pasL = user.Password.Length;
        command.Parameters.Add("@username", System.Data.SqlDbType.VarChar, usrL).Value = user.UserName;
        command.Parameters.Add("@password", System.Data.SqlDbType.VarChar, pasL).Value = user.Password;

        try
        {
            connection.Open();
            command.ExecuteNonQuery();
            return FindUserByNameAndPassword(user);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }

    }
    //return success;
    return null;
}
1 reference
```



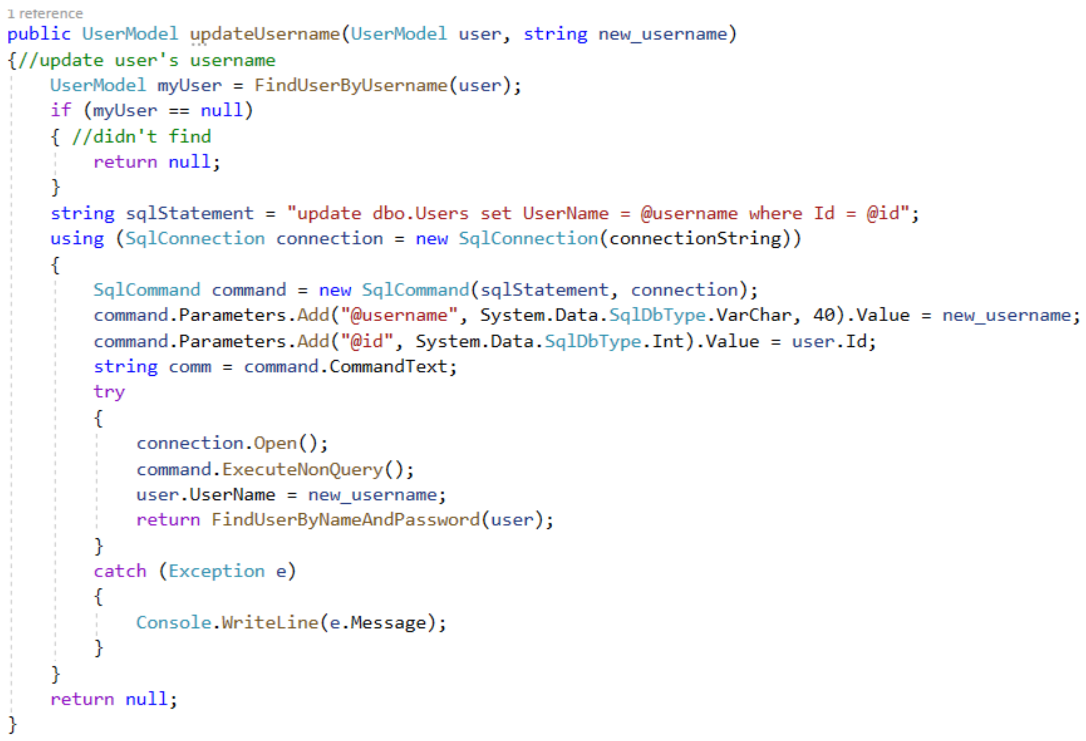
```
public UserModel updateUser(UserModel user)
{//used to update the user after the transfer is successful
    UserModel myUser = FindUserByUsername(user);
    if (myUser == null)
    { //didn't find
        return null;
    }
    string sqlStatement = "update dbo.Users set UserName = @username, Password = @password, Ballance = @ballance, IsAdmin = @isadmin where Id = @id";

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);

        command.Parameters.Add("@username", System.Data.SqlDbType.VarChar, 40).Value = user.UserName;
        command.Parameters.Add("@password", System.Data.SqlDbType.VarChar, 40).Value = user.Password;
        command.Parameters.Add("@ballance", System.Data.SqlDbType.Float).Value = user.Ballance;
        command.Parameters.Add("@isadmin", System.Data.SqlDbType.Int).Value = user.IsAdmin;
        command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = user.Id;
        string comm = command.CommandText;

        try
        {
            connection.Open();
            command.ExecuteNonQuery();
            return FindUserByNameAndPassword(user);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }

    }

    return null;
}
```



```
public UserModel removeUser(UserModel user)
{//when an admin delets an user
    UserModel myUser = FindUserByUsername(user);
    if (myUser == null)
    { //didn't find
        return null;
    }
    string sqlStatement = "DELETE FROM dbo.Users WHERE UserName = @username ";

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);
        command.Parameters.Add("@username", System.Data.SqlDbType.VarChar, 40).Value = user.UserName;
        string comm = command.CommandText;
        try
        {
            connection.Open();
            command.ExecuteNonQuery();
            return FindUserByNameAndPassword(user);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
    return null;
}
```

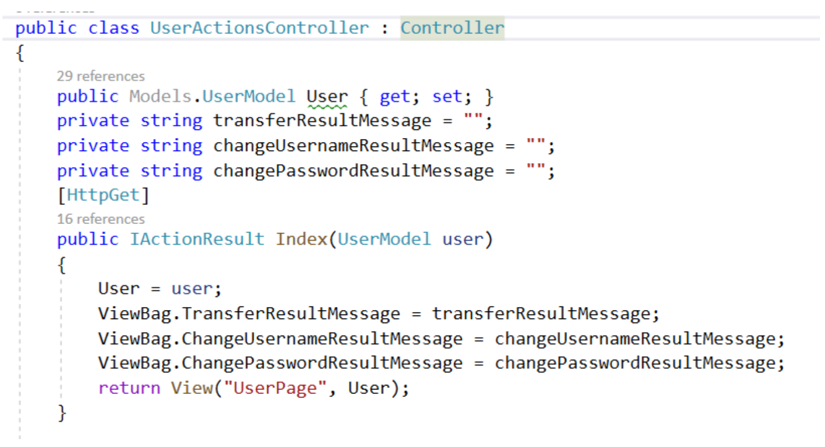


```
public UserModel updateUserByPassword(UserModel user, string new_password)
{//update user's password
    UserModel myUser = FindUserByUsername(user);
    if (myUser == null)
    { //didn't find
        return null;
    }
    string sqlStatement = "update dbo.Users set Password = @password where Id = @id";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);
        command.Parameters.Add("@password", System.Data.SqlDbType.VarChar, 40).Value = new_password;
        command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = user.Id;
        string comm = command.CommandText;
        try
        {
            connection.Open();
            command.ExecuteNonQuery();
            myUser = FindUserByUsername(user);
            myUser.Password = new_password;
            return myUser;
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
    return null;
}
```

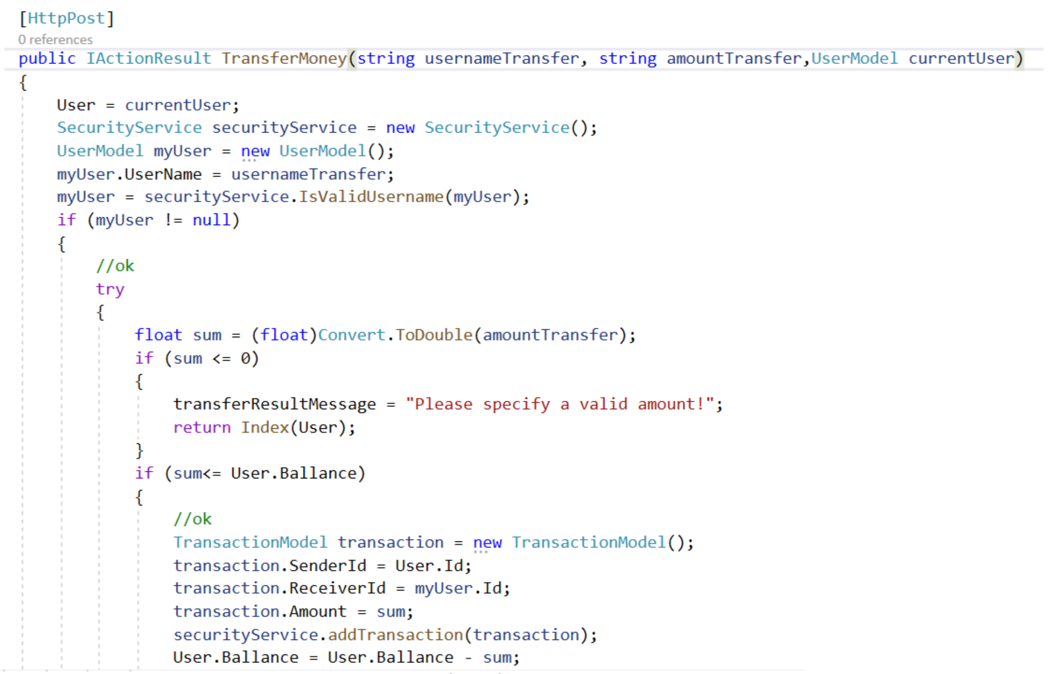
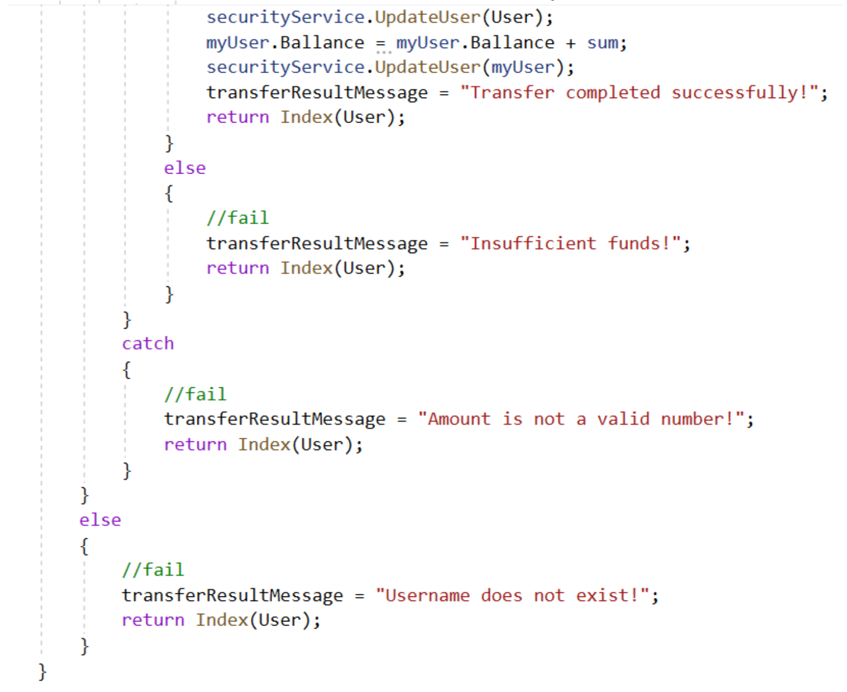


```
1 reference
public UserModel updateUsername(UserModel user, string new_username)
{//update user's username
    UserModel myUser = FindUserByUsername(user);
    if (myUser == null)
    { //didn't find
        return null;
    }
    string sqlStatement = "update dbo.Users set UserName = @username where Id = @id";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(sqlStatement, connection);
        command.Parameters.Add("@username", System.Data.SqlDbType.VarChar, 40).Value = new_username;
        command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = user.Id;
        string comm = command.CommandText;
        try
        {
            connection.Open();
            command.ExecuteNonQuery();
            user.UserName = new_username;
            return FindUserByNameAndPassword(user);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
    return null;
}
```

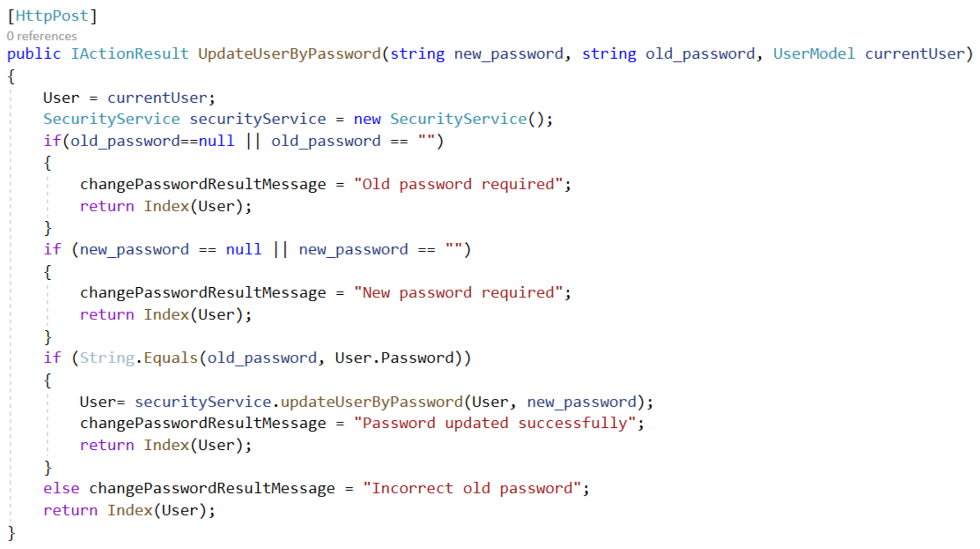
6.2. UserActionsController class



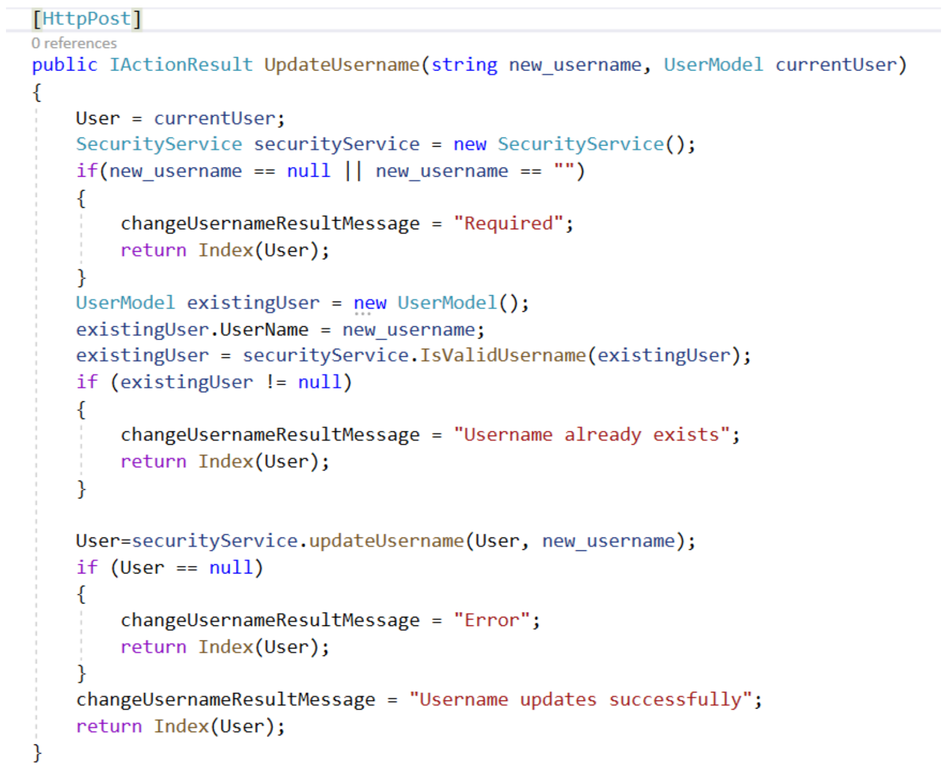
```
public class UserActionsController : Controller
{
    29 references
    public Models.UserModel User { get; set; }
    private string transferResultMessage = "";
    private string changeUsernameResultMessage = "";
    private string changePasswordResultMessage = "";
    [HttpGet]
    16 references
    public IActionResult Index(UserModel user)
    {
        User = user;
        ViewBag.TransferResultMessage = transferResultMessage;
        ViewBag.ChangeUsernameResultMessage = changeUsernameResultMessage;
        ViewBag.ChangePasswordResultMessage = changePasswordResultMessage;
        return View("UserPage", User);
    }
```

```
[HttpPost]
0 references
public IActionResult TransferMoney(string usernameTransfer, string amountTransfer,UserModel currentUser)
{
    User = currentUser;
    SecurityService securityService = new SecurityService();
    UserModel myUser = new UserModel();
    myUser.UserName = usernameTransfer;
    myUser = securityService.IsValidUsername(myUser);
    if (myUser != null)
    {
        //ok
        try
        {
            float sum = (float)Convert.ToDouble(amountTransfer);
            if (sum <= 0)
            {
                transferResultMessage = "Please specify a valid amount!";
                return Index(User);
            }
            if (sum<= User.Ballance)
            {
                //ok
                TransactionModel transaction = new TransactionModel();
                transaction.SenderId = User.Id;
                transaction.ReceiverId = myUser.Id;
                transaction.Amount = sum;
                securityService.addTransaction(transaction);
                User.Ballance = User.Ballance - sum;
                securityService.UpdateUser(User);
                myUser.Ballance = myUser.Ballance + sum;
                securityService.UpdateUser(myUser);
                transferResultMessage = "Transfer completed successfully!";
                return Index(User);
            }
            else
            {
                //fail
                transferResultMessage = "Insufficient funds!";
                return Index(User);
            }
        }
        catch
        {
            //fail
            transferResultMessage = "Amount is not a valid number!";
            return Index(User);
        }
    }
    else
    {
        //fail
        transferResultMessage = "Username does not exist!";
        return Index(User);
    }
}
```

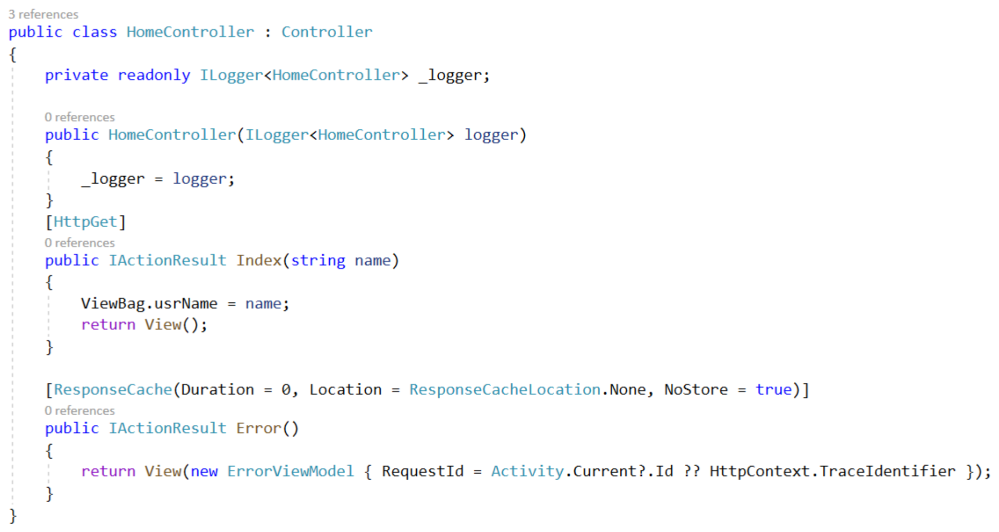


```
[HttpPost]
0 references
public IActionResult UpdateUserByPassword(string new_password, string old_password, UserModel currentUser)
{
    User = currentUser;
    SecurityService securityService = new SecurityService();
    if(old_password==null || old_password == "")
    {
        changePasswordResultMessage = "Old password required";
        return Index(User);
    }
    if (new_password == null || new_password == "")
    {
        changePasswordResultMessage = "New password required";
        return Index(User);
    }
    if (String.Equals(old_password, User.Password))
    {
        User= securityService.updateUserByPassword(User, new_password);
        changePasswordResultMessage = "Password updated successfully";
        return Index(User);
    }
    else changePasswordResultMessage = "Incorrect old password";
    return Index(User);
}
```



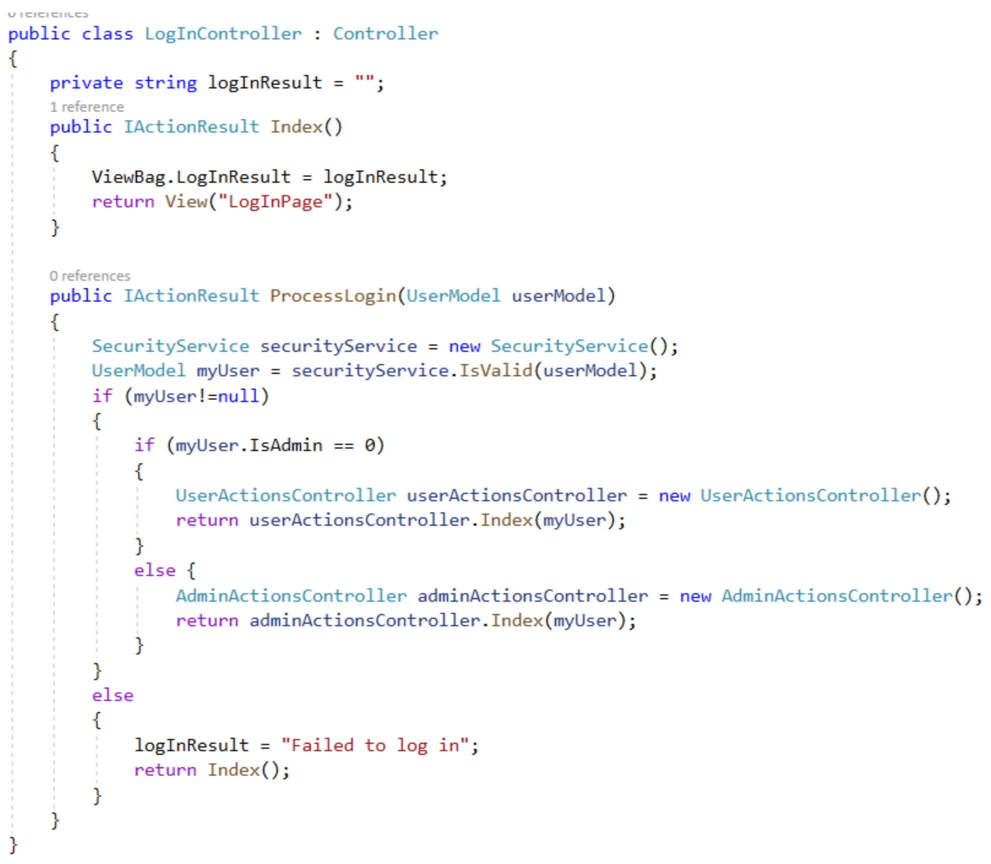
```
[HttpPost]
0 references
public IActionResult UpdateUsername(string new_username, UserModel currentUser)
{
    User = currentUser;
    SecurityService securityService = new SecurityService();
    if(new_username == null || new_username == "")
    {
        changeUsernameResultMessage = "Required";
        return Index(User);
    }
    UserModel existingUser = new UserModel();
    existingUser.UserName = new_username;
    existingUser = securityService.IsValidUsername(existingUser);
    if (existingUser != null)
    {
        changeUsernameResultMessage = "Username already exists";
        return Index(User);
    }

    User=securityService.updateUsername(User, new_username);
    if (User == null)
    {
        changeUsernameResultMessage = "Error";
        return Index(User);
    }
    changeUsernameResultMessage = "Username updates successfully";
    return Index(User);
}
```

6.3. HomeController class



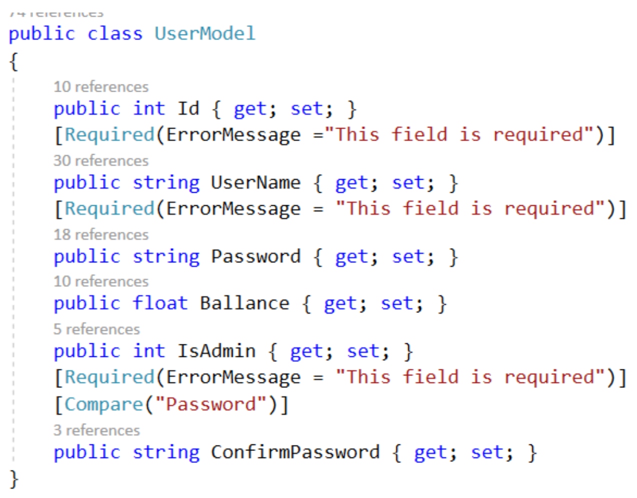
```
3 references
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    0 references
    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }
    [HttpGet]
    0 references
    public IActionResult Index(string name)
    {
        ViewBag.usrName = name;
        return View();
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    0 references
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }
}
```

6.4. LogInController class



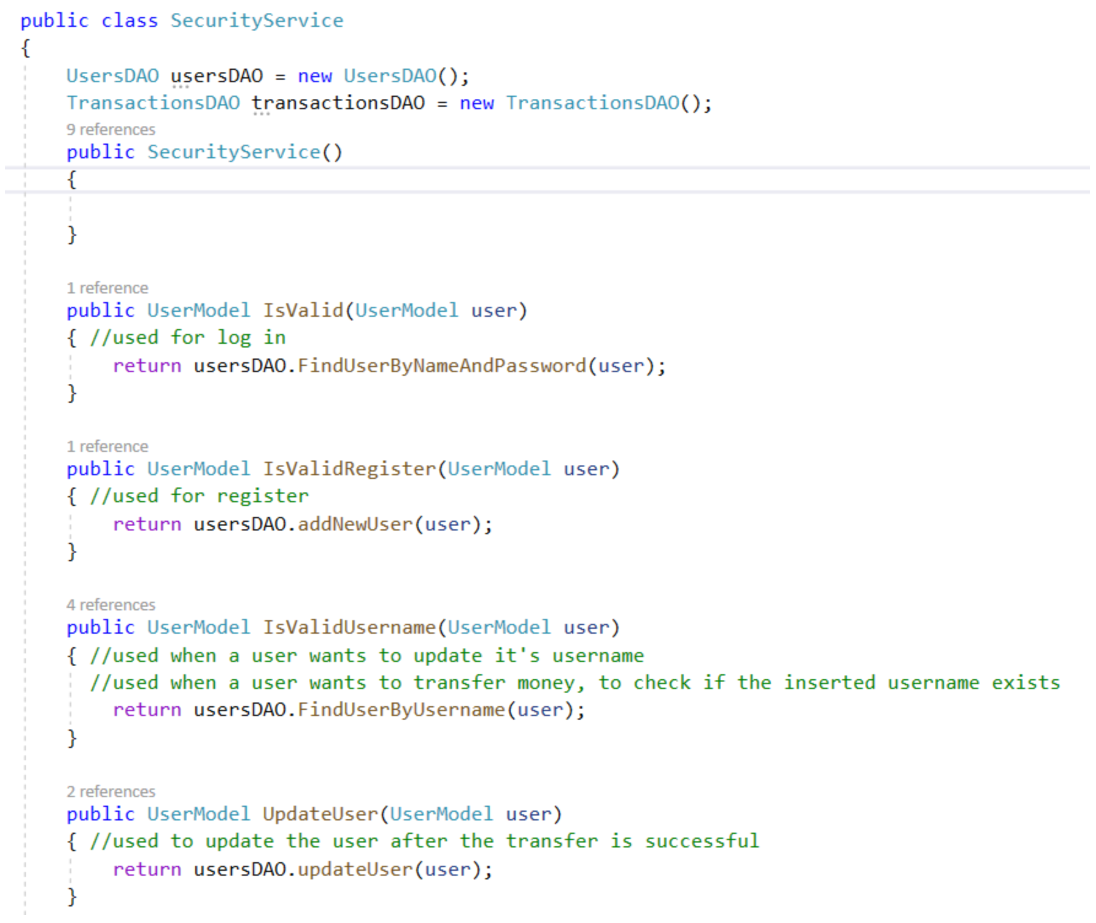
```
0 references
public class LogInController : Controller
{
    private string logInResult = "";
    1 reference
    public IActionResult Index()
    {
        ViewBag.LogInResult = logInResult;
        return View("LogInPage");
    }

    0 references
    public IActionResult ProcessLogin(UserModel userModel)
    {
        SecurityService securityService = new SecurityService();
        UserModel myUser = securityService.IsValid(userModel);
        if (myUser!=null)
        {
            if (myUser.IsAdmin == 0)
            {
                UserActionsController userActionsController = new UserActionsController();
                return userActionsController.Index(myUser);
            }
            else {
                AdminActionsController adminActionsController = new AdminActionsController();
                return adminActionsController.Index(myUser);
            }
        }
        else
        {
            logInResult = "Failed to log in";
            return Index();
        }
    }
}
```

6.5. UserModel class

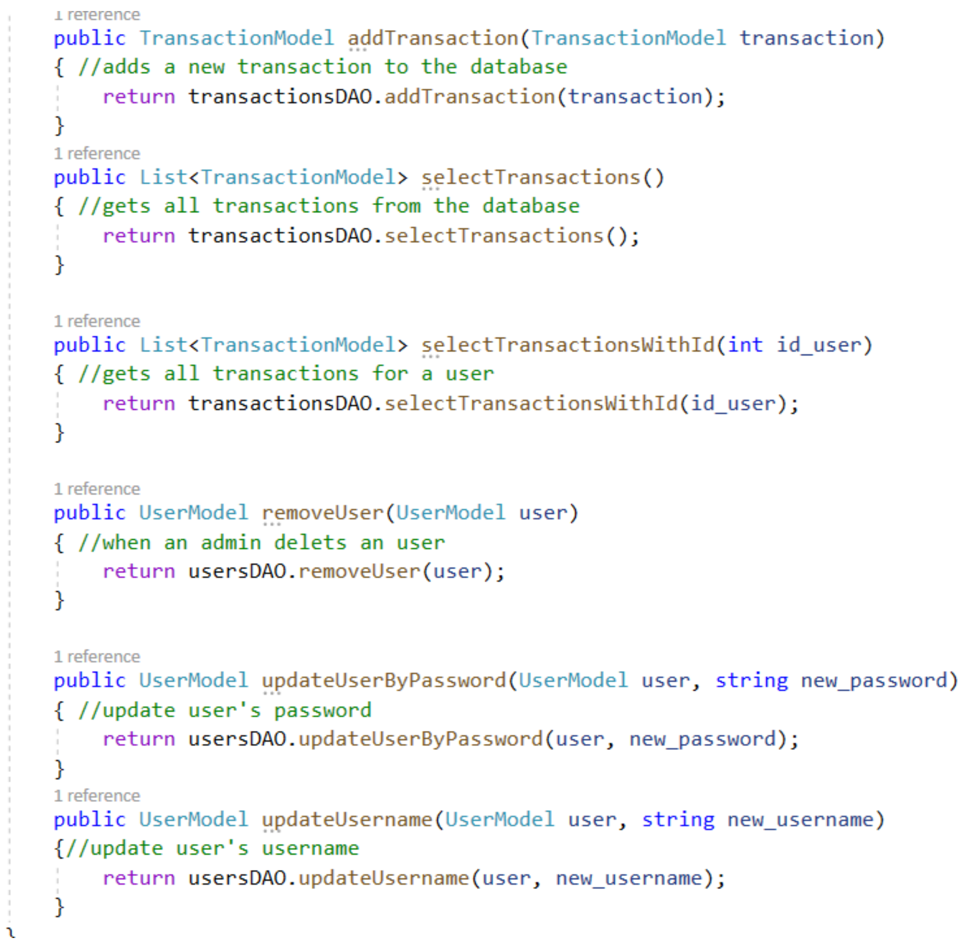


```
74 references
public class UserModel
{
    10 references
    public int Id { get; set; }
    [Required(ErrorMessage ="This field is required")]
    30 references
    public string UserName { get; set; }
    [Required(ErrorMessage = "This field is required")]
    18 references
    public string Password { get; set; }
    10 references
    public float Ballance { get; set; }
    5 references
    public int IsAdmin { get; set; }
    [Required(ErrorMessage = "This field is required")]
    [Compare("Password")]
    3 references
    public string ConfirmPassword { get; set; }
}
```

6.6. SecurityService class

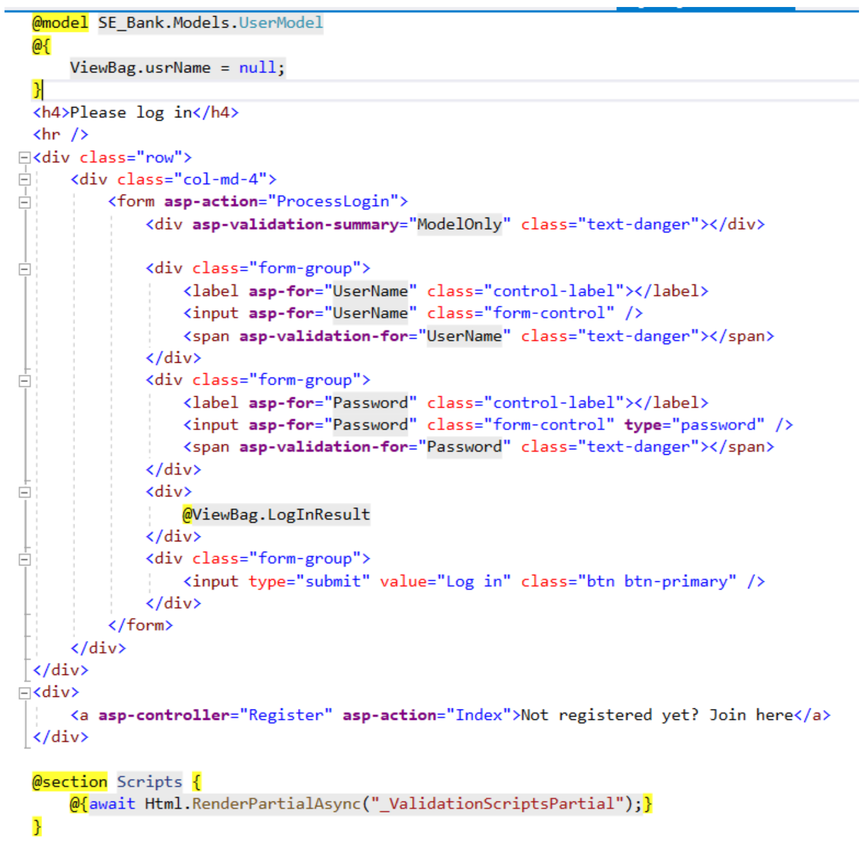


```
public class SecurityService
{
    UsersDAO usersDAO = new UsersDAO();
    TransactionsDAO transactionsDAO = new TransactionsDAO();
    9 references
    public SecurityService()
    {

    }

    1 reference
    public UserModel IsValid(UserModel user)
    { //used for log in
        return usersDAO.FindUserByNameAndPassword(user);
    }

    1 reference
    public UserModel IsValidRegister(UserModel user)
    { //used for register
        return usersDAO.addNewUser(user);
    }

    4 references
    public UserModel IsValidUsername(UserModel user)
    { //used when a user wants to update it's username
      //used when a user wants to transfer money, to check if the inserted username exists
        return usersDAO.FindUserByUsername(user);
    }

    2 references
    public UserModel UpdateUser(UserModel user)
    { //used to update the user after the transfer is successful
        return usersDAO.updateUser(user);
    }
```



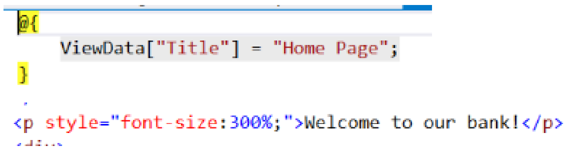
```
1 reference
public TransactionModel addTransaction(TransactionModel transaction)
{ //adds a new transaction to the database
    return transactionsDAO.addTransaction(transaction);
}
1 reference
public List<TransactionModel> selectTransactions()
{ //gets all transactions from the database
    return transactionsDAO.selectTransactions();
}

1 reference
public List<TransactionModel> selectTransactionsWithId(int id_user)
{ //gets all transactions for a user
    return transactionsDAO.selectTransactionsWithId(id_user);
}

1 reference
public UserModel removeUser(UserModel user)
{ //when an admin delets an user
    return usersDAO.removeUser(user);
}

1 reference
public UserModel updateUserByPassword(UserModel user, string new_password)
{ //update user's password
    return usersDAO.updateUserByPassword(user, new_password);
}
1 reference
public UserModel updateUsername(UserModel user, string new_username)
{//update user's username
    return usersDAO.updateUsername(user, new_username);
}
}
```

6.7. LogInPage page

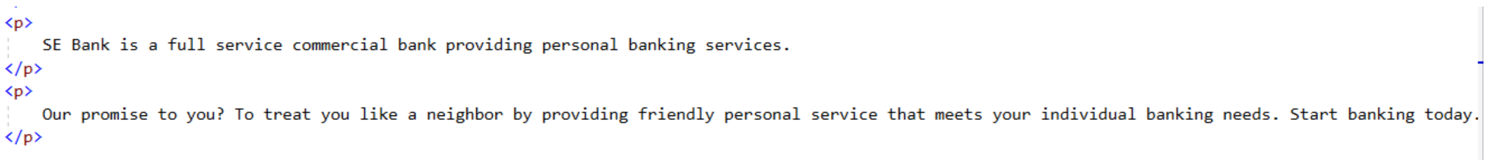


```
@model SE_Bank.Models.UserModel
@{
    ViewBag.usrName = null;
}
<h4>Please log in</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="ProcessLogin">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

            <div class="form-group">
                <label asp-for="UserName" class="control-label"></label>
                <input asp-for="UserName" class="form-control" />
                <span asp-validation-for="UserName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password" class="control-label"></label>
                <input asp-for="Password" class="form-control" type="password" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div>
                @ViewBag.LogInResult
            </div>
            <div class="form-group">
                <input type="submit" value="Log in" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
<div>
    <a asp-controller="Register" asp-action="Index">Not registered yet? Join here</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

6.8. Index page (Home)

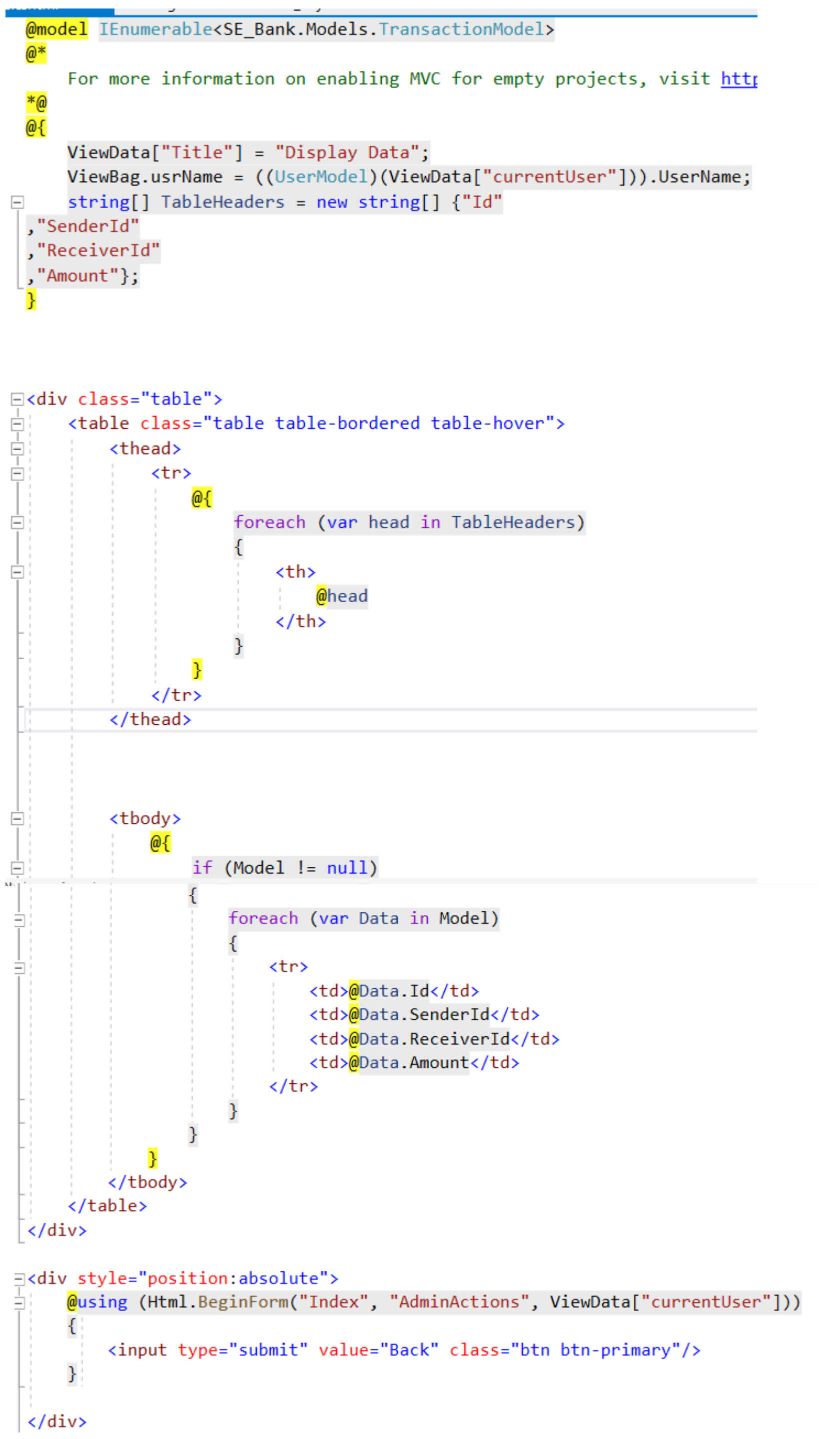


```
@{
    ViewData["Title"] = "Home Page";
}

<p style="font-size:300%;">Welcome to our bank!</p>
```



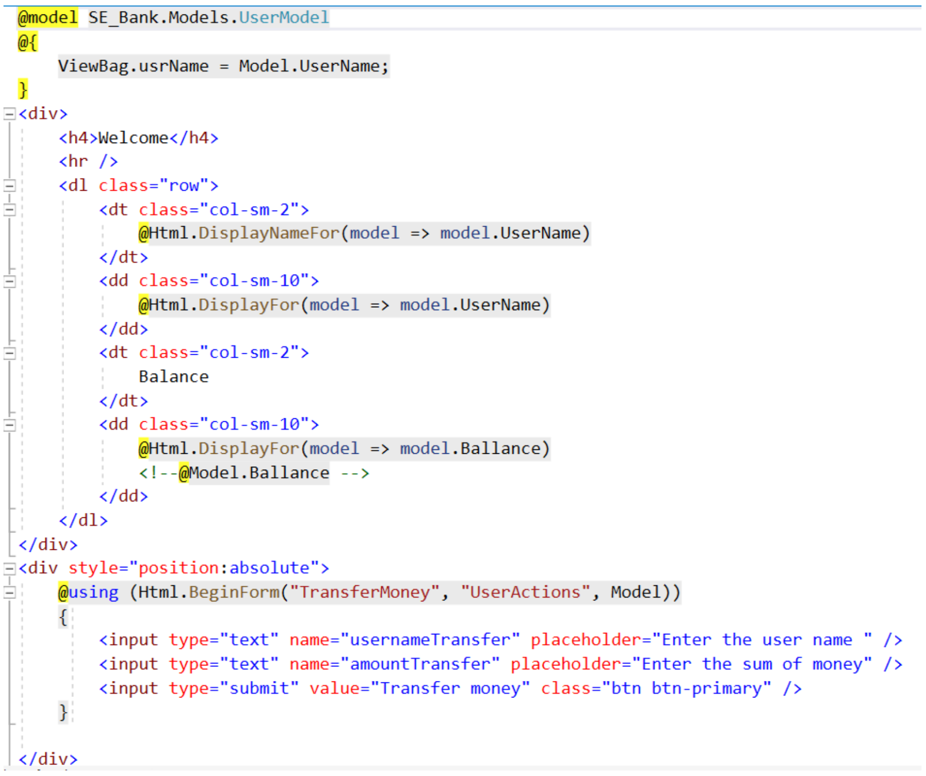
```
<p>
    SE Bank is a full service commercial bank providing personal banking services.
</p>
<p>
    Our promise to you? To treat you like a neighbor by providing friendly personal service that meets your individual banking needs. Start banking today.
</p>
```

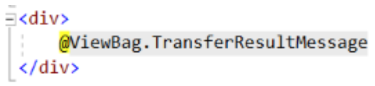
6.9. UserTransactions page

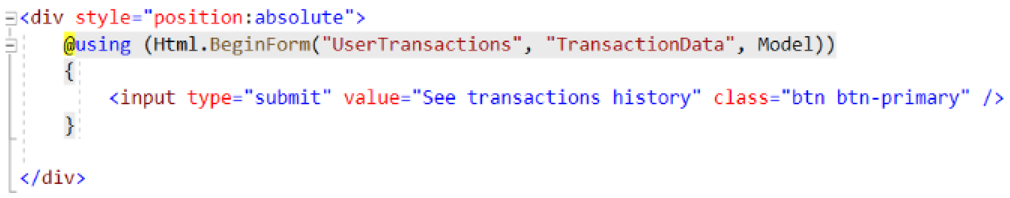


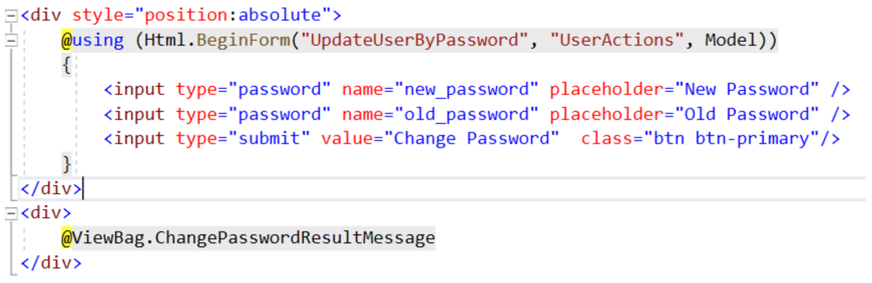
```
@model IEnumerable<SE_Bank.Models.TransactionModel>
@*
    For more information on enabling MVC for empty projects, visit http
*@
@{
    ViewData["Title"] = "Display Data";
    ViewBag.usrName = ((UserModel)(ViewData["currentUser"])).UserName;
    string[] TableHeaders = new string[] {"Id"
,"SenderId"
,"ReceiverId"
,"Amount"};
}


<div class="table">
    <table class="table table-bordered table-hover">
        <thead>
            <tr>
                @{
                    foreach (var head in TableHeaders)
                    {
                        <th>
                            @head
                        </th>
                    }
                }
            </tr>
        </thead>



        <tbody>
            @{
                if (Model != null)
                {
                    foreach (var Data in Model)
                    {
                        <tr>
                            <td>@Data.Id</td>
                            <td>@Data.SenderId</td>
                            <td>@Data.ReceiverId</td>
                            <td>@Data.Amount</td>
                        </tr>
                    }
                }
            }
        </tbody>
    </table>
</div>

<div style="position:absolute">
    @using (Html.BeginForm("Index", "AdminActions", ViewData["currentUser"]))
    {
        <input type="submit" value="Back" class="btn btn-primary"/>
    }

</div>
```

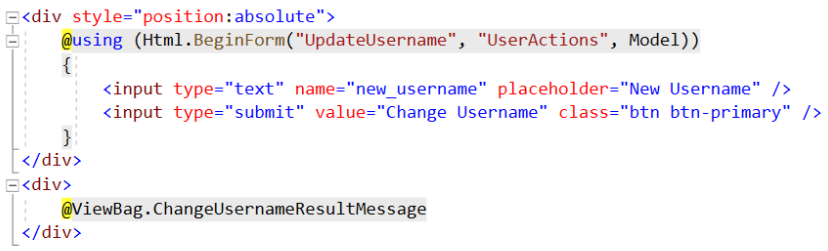
6.10. UserPage page


```
@model SE_Bank.Models.UserModel
@{
    ViewBag.usrName = Model.UserName;
}
<div>
    <h4>Welcome</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.UserName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.UserName)
        </dd>
        <dt class="col-sm-2">
            Balance
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Ballance)
            <!--@Model.Ballance -->
        </dd>
    </dl>
</div>
<div style="position:absolute">
    @using (Html.BeginForm("TransferMoney", "UserActions", Model))
    {
        <input type="text" name="usernameTransfer" placeholder="Enter the user name " />
        <input type="text" name="amountTransfer" placeholder="Enter the sum of money" />
        <input type="submit" value="Transfer money" class="btn btn-primary" />
    }

</div>
```

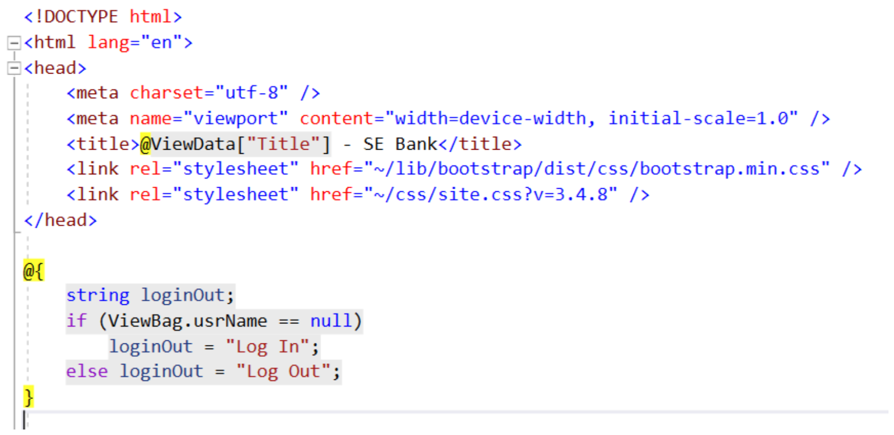

```
<div>
    @ViewBag.TransferResultMessage
</div>
```


```
<div style="position:absolute">
    @using (Html.BeginForm("UserTransactions", "TransactionData", Model))
    {
        <input type="submit" value="See transactions history" class="btn btn-primary" />
    }

</div>
```

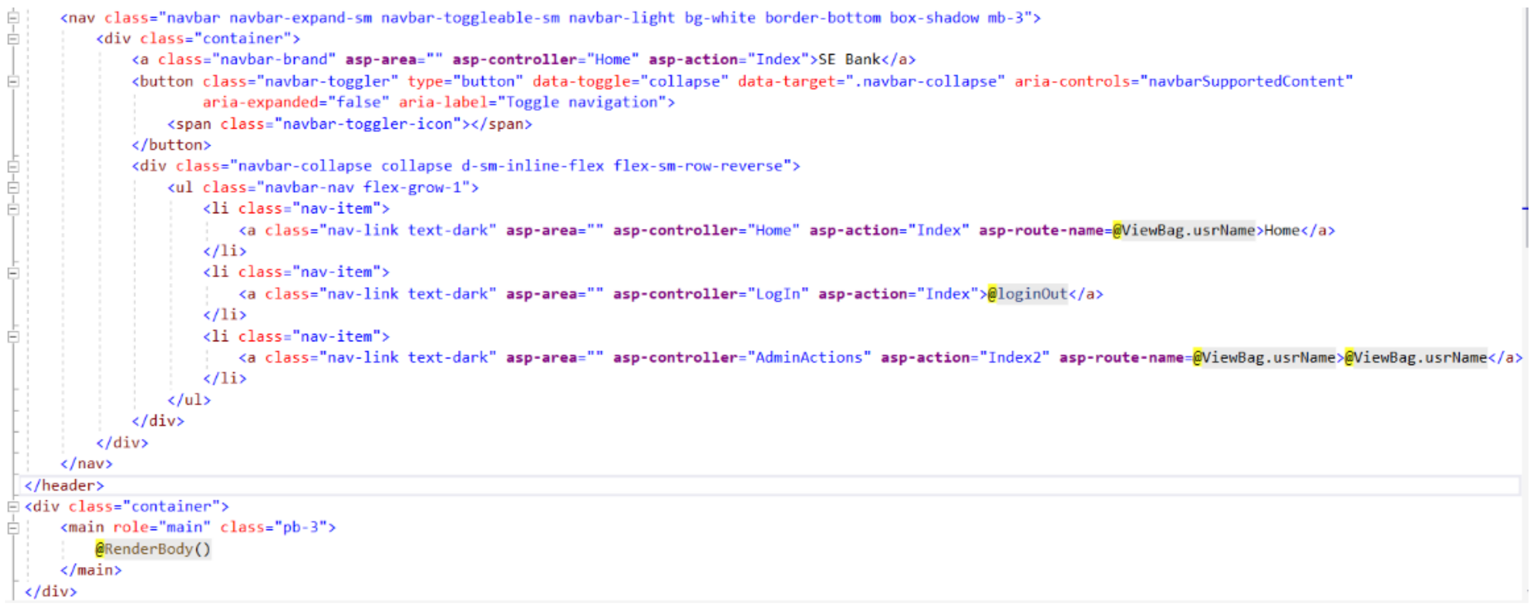

```
<div style="position:absolute">
    @using (Html.BeginForm("UpdateUserByPassword", "UserActions", Model))
    {
        <input type="password" name="new_password" placeholder="New Password" />
        <input type="password" name="old_password" placeholder="Old Password" />
        <input type="submit" value="Change Password"  class="btn btn-primary"/>
    }
</div>
<div>
    @ViewBag.ChangePasswordResultMessage
</div>
```


```
<div style="position:absolute">
    @using (Html.BeginForm("UpdateUsername", "UserActions", Model))
    {
        <input type="text" name="new_username" placeholder="New Username" />
        <input type="submit" value="Change Username" class="btn btn-primary" />
    }
</div>
<div>
    @ViewBag.ChangeUsernameResultMessage
</div>
```

6.11. _Layout page



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - SE Bank</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css?v=3.4.8" />
</head>

@{
    string loginOut;
    if (ViewBag.usrName == null)
        loginOut = "Log In";
    else loginOut = "Log Out";
}
```



```
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
        <div class="container">
            <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">SE Bank</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                    aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
                <ul class="navbar-nav flex-grow-1">
                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index" asp-route-name=@ViewBag.usrName>Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="LogIn" asp-action="Index">@loginOut</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="AdminActions" asp-action="Index2" asp-route-name=@ViewBag.usrName>@ViewBag.usrName</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
</header>
<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>
```

7. References

1. https://www.youtube.com/watch?v=BfEjDD8mWYg
2. https://creately.com/blog/diagrams/deployment-diagram-tutorial/
3. https://www.youtube.com/watch?v=RPTpC1O87_Q
4. https://docs.microsoft.com/en-us/dotnet/api/system.string.length?view=net-6.0
5. https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.button?view=windowsdesktop-6.0
6. https://stackoverflow.com/questions/3081916/convert-int-to-string
7. https://stackoverflow.com/questions/16112982/how-to-use-the-varchar-data-type-in-c/16112990
8. https://stackoverflow.com/questions/9264835/how-to-align-an-input-tag-to-the-center-without-specifying-the-width
9. https://www.w3schools.com/tags/tag_center.asp
10. https://www.quackit.com/css/css_color_codes.cfm
11. https://css-tricks.com/strategies-for-cache-busting-css/
12. https://stackoverflow.com/questions/21351004/css-body-background-image-doesnt-work
13. https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-6.0