ASP.NET MVC -CA Project

SA49 – Team 6B

Member:

Phung Khanh Chi

Chai Cai

Yeo Shen Yean

Video Url: https://youtu.be/9gXiBSEXoVU

Document for CA_Project_Team6B

1. Introduction

We designed a video game e-commerce ASP.NET MVC application with the database which uses the Mode First model of the Entity Framework.

The customers can login, browse, search and purchase the games from the website.

2. Part I: Key functionalities

Account management

In the database, the passwords have already been hashed using CreatePasswordSalt(string password) function, with string password as submitted by the user after registration. When the user logs in again, password will be sent to the controller to be processed and matched with database's record.

```
private const int SALT_SIZE = 8;
private const int NUM_ITERATIONS = 1000;
private static readonly RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
10 references
public static string CreatePasswordSalt(string password)
{
    //random salt for each user
    byte[] buf = new byte[SALT_SIZE];
    rng.GetBytes(buf);
    string salt = Convert.ToBase64String(buf);
    Rfc2898DeriveBytes deriver2898 = new Rfc2898DeriveBytes(password.Trim(), buf, NUM_ITERATIONS);
    string hash = Convert.ToBase64String(deriver2898.GetBytes(16));
    //salt will be saved in the database with hash_value
    return salt + ':' + hash;
}
```

1)Login

After that, when the user key in password and press "Log in" button, the username and password will be sent to the controller.

```
private const int NUM_ITERATIONS = 1000;
1 reference
public static bool IsPasswordValid(string password, string saltHash)
{
    //after user sends username and plain-text password to the server, server will
    //check and get the salt of that user:

    //1.saltHash is the password saved in database and is retrieved by using function
    //GetPassword(string username). After that, password will be splited to salt and hash
    string[] parts = saltHash.Split(new[] { ':' }, StringSplitOptions.RemoveEmptyEntries);

    if (parts.Length != 2) return false;

    //2. Hash the plain-text password sent by user in the server side, using the stored salt
    //from database
    byte[] buf = Convert.FromBase64String(parts[0]);
    Rfc2898DeriveBytes deriver2898 = new Rfc2898DeriveBytes(password.Trim(), buf, NUM_ITERATIONS);
    string computedHash = Convert.ToBase64String(deriver2898.GetBytes(16));

    //3. Compare the result of step 2 with the hash in the database
    return parts[1].Equals(computedHash);
}
```

```
public ActionResult Login(Customer customer)
{
    //check if username is valid in the database
    if (CustomerData.GetPassword(customer.Username) != null)
    {
        //if username is valid, check if password is matched
        if (CustomerData.IsPasswordValid(customer.Password, CustomerData.GetPassword(customer.Username)))
        {
            //if username and password are valid, save customer info to a session and redirect
            //customer to the product gallery page
            Session["customer"] = CustomerData.GetCustomer(customer.Username);
            return RedirectToAction("Index", "Product");
        }
        ViewBag.errorMessage = "Wrong password. Please try again";
        return View("LoginPage");
    }
    //if username is not valid, show the error message "Invalid username"
    ViewBag.errorMessage = "Invalid username";
    return View("LoginPage");
}
```

2 & 3. List Products and Search

Browse and search for products:

Function ProductController.Index(string sortOrder, int?page, string searchStr= “”)

```
public ActionResult Index(string sortOrder, int?page, string searchStr = "")
{
    if (Session["customer"] == null)
        return RedirectToAction("LoginPage", "Home");
    else
    {
        //Retrieve all product from the database and send to the View
        List<Product> products = ProductData.GetAllProducts();
        //if the user enter a keyword to search, products will be filtered, those with matched
        //name and description will be showed
        products = products.Where(p => p.ProductName.ToLower().Contains(searchStr.ToLower())
                        || p.Description.ToLower().Contains(searchStr.ToLower())).ToList();

        switch (sortOrder)...

        if (Session["cart"] != null)
        {
            ViewData["cart"] = Session["cart"];
        }
        ViewBag.searchStr = searchStr;
        ViewBag.sortOrder = sortOrder;
        int pageSize = 8;
        int pageNumber = (page ?? 1);
        return View(products.ToPagedList(pageNumber, pageSize));
    }
}
```

Shopping Cart Management

4.Add products to cart

```
public void Buy(int productId, double newPrice)
{
    //after the user press "Add to cart" button, productId and price (after discounted) will
    //be sent to the server side. If there is no existing cart session, the product will be
    //saved in a new cart session
    if (Session["cart"] == null)
    {
        List<CartItem> cart = new List<CartItem>();
        cart.Add(new CartItem { Product = ProductData.find(productId),
            Quantity = 1, NewPrice = newPrice });
        Session["cart"] = cart;
    }
    else
    {
        //if a cart session is already existing, ProductId will be checked
        List<CartItem> cart = (List<CartItem>)Session["cart"];
        int index = isExit(productId);
        if (index != -1)
        {
            //if there is a matched ProductId in the existing session, product's quantity will be increased
            cart[index].Quantity++;
        }
        else
        {
            //if there is no matched ProductId in the existing session, new Product will be added
            cart.Add(new CartItem { Product = ProductData.find(productId), Quantity = 1, NewPrice = newPrice });
        }
        Session["cart"] = cart;
    }
}
```

5.View Cart

Change the quantity of cart items:

```
public void Change(int productId, int num)
{
    List<CartItem> cart = (List<CartItem>)Session["cart"];
    //if the user change the quantity, the product will be find by its Id and the old quantity will be
    //replaced by the new one sent from the View
    int index = isExit(productId);
    cart[index].Quantity = num;
    Session["cart"] = cart;
}
```

Remove products from the cart:

6.Manage Activation Codes

Shopping cart checkout and activation code management: Unique activation code will be given with each copy of purchased product.

```
List<CartItem> cart = (List<CartItem>)Session["cart"];

List<Product> products = new List<Product>();
List<int> quantities = new List<int>();
List<ActivationCode> codeLists = new List<ActivationCode>();
for(int i=0; i < cart.Count(); i++)
{
    //adding products in the cart to a new list of product
    products.Add(cart[i].Product);
    quantities.Add(cart[i].Quantity);
    int productId = cart[i].Product.ProductId;
    //for each productId, a new list of codes will be generated, with number of elements equal to
    //the quantity of that product
    List<string> codes = new List<string>();
    for (int j = 0; j < cart[i].Quantity; j++)
    {
        //activation codes for that products will be generated and saved to the list
        codes.Add(Guid.NewGuid().ToString());
    }
    ActivationCode actCode = new ActivationCode(productId, codes);
    codeLists.Add(actCode);
}
OrderDetail orderDetail = new OrderDetail(products, quantities, codeLists);
Customer customer = (Customer)Session["customer"];
int customerId = Convert.ToInt32(customer.CustomerId);
Order order = new Order()
{
    CustomerId = customerId,
    OrderDate = DateTime.Now,
    OrderDetail = orderDetail
};
//order details are saved to the database by OrderData.SaveOrder(order Order) method
OrderData.SaveOrder(order);
int orderId = OrderData.GetOrderId(order);
Session["cart"] = null;
return RedirectToAction("Index", "Order", new { orderId = orderId});
```

7. View Purchase History

Customer can view their order histories and see the details after logging into the account

```
public static List<Order> GetOrderHistory(int customerId)
{
    List<Order> orderHistory = new List<Order>();
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();
        string sql = @"select OrderId, OrderDate from Orders where CustomerId = @customerId";
        SqlCommand cmd = new SqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@customerId", customerId);
        SqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Order order = new Order()
            {
                OrderId = (int)reader["OrderId"],
                OrderDate = (DateTime)reader["OrderDate"]
            };
            orderHistory.Add(order);
        }
    }
    return orderHistory;
}
```

8.Logout

```
public ActionResult LogOff()
{
    Session.Abandon();
    return RedirectToAction("Index", "Home");
}
```

3 Part II: other functionalities

1) The sort functions

Customer can view the product order some specific needs.

```
switch (sortOrder)
{
    case "price_asc":
        products = products.OrderBy(p => p.Price).ToList();
        break;
    case "price_desc":
        products = products.OrderByDescending(p => p.Price).ToList();
        break;
    case "sales_desc":
        products = products.OrderByDescending(p => p.NumOfSales).ToList();
        break;
    case "discount_desc":
        products = products.OrderByDescending(p => p.Discount).ToList();
        break;
    case "release_desc":
        products = products.OrderByDescending(p => p.ReleaseDate).ToList();
        break;
    default:
        products = products.OrderBy(p => p.ProductId).ToList();
        break;
}
```

2). The customer can view the product detail after clicking the product image.

```
public ActionResult ProductDetail(int productId)
{
    Product product = ProductData.GetProductById(productId);
    ViewData["product"] = product;
    return View();
}
```

3). We add the remove function in cart so that the customer can remove the product they undesired.

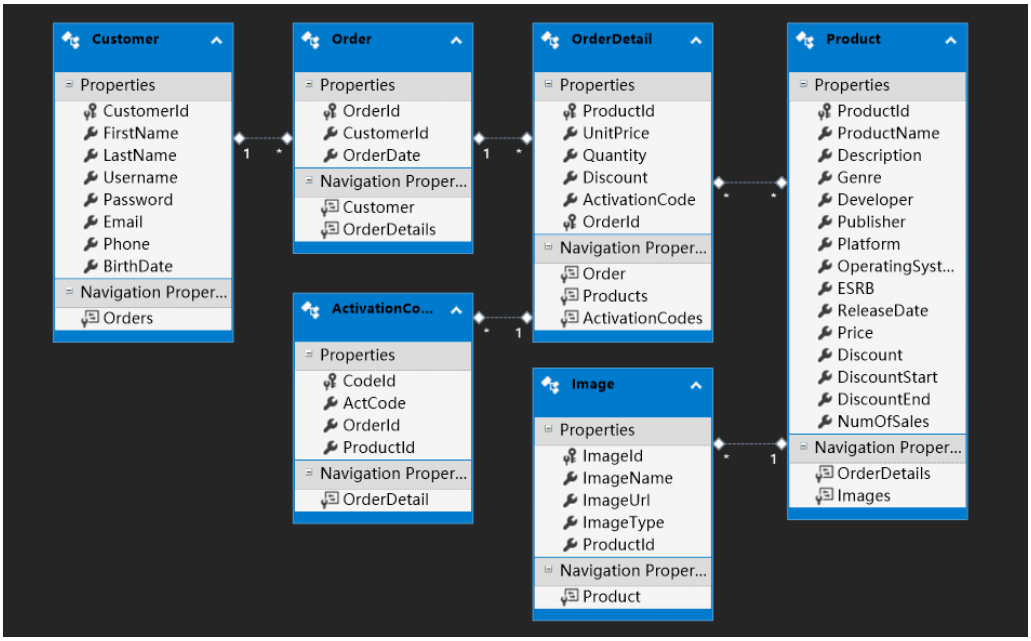
```
public void Remove(int productId)
{
    List<CartItem> cart = (List<CartItem>)Session["cart"];
    //find the product in cart session by its ProductId
    int index = isExit(productId);
    cart.RemoveAt(index);
    Session["cart"] = cart;
}
```

4). User can download file from server when click on download link.

```
public void Download()
{
    HttpResponse response = System.Web.HttpContext.Current.Response;
    response.ClearContent();
    response.Clear();
    response.ContentType = "text/plain";
    response.AddHeader("Content-Disposition",
                       "attachment; filename= Destiny2.jpg;");
    response.TransmitFile(Server.MapPath("../Images/Destiny2.jpg"));
    response.Flush();
    response.End();
}
```

4 Database schema

The main database is built in Model-First approach with Entity Framework.



Then the data will be seeded

And stored in the SQL server.

Video Url: https://youtu.be/9gXiBSEXoVU