SUPERWISH

SUPERWISH IS AN E-COMMERCE, ASP.NET RAZOR PAGES APPLICATION FOR TOYS.

BY KRISTY HOFFMAN, EDGAR TOWNSEND, AND XUAN WANG.

SUPERWISH'S PURPOSE

The purpose of our application is to facilitate the purchase of toys. In creating it, we aimed to solve problems related to loss of business from limiting purchases to in-store.

We also wanted to offer toys that are difficult to find in other children's stores by specializing in one type of toy.

ROLES

Within the scope of Superwish, there are two roles: administrator and user. Both are capable of registering and logging in.

Everyone can browse the index page without login

User will be able to login and make selections of the item, they can edit the items in the cart page

FUNCTIONALITIES

Originally, we sought to implement the following features and functionalities:

Administrators could perform CRUD for toy listings and inventory, as well as customer profiles.

Users can perform CRUD for their carts and checkout with toys they wished to purchase.

IMPLEMENTATION

Working with a code-first approach, we successfully implemented our database. We also implemented roles, storage, and users can check out with their purchases.

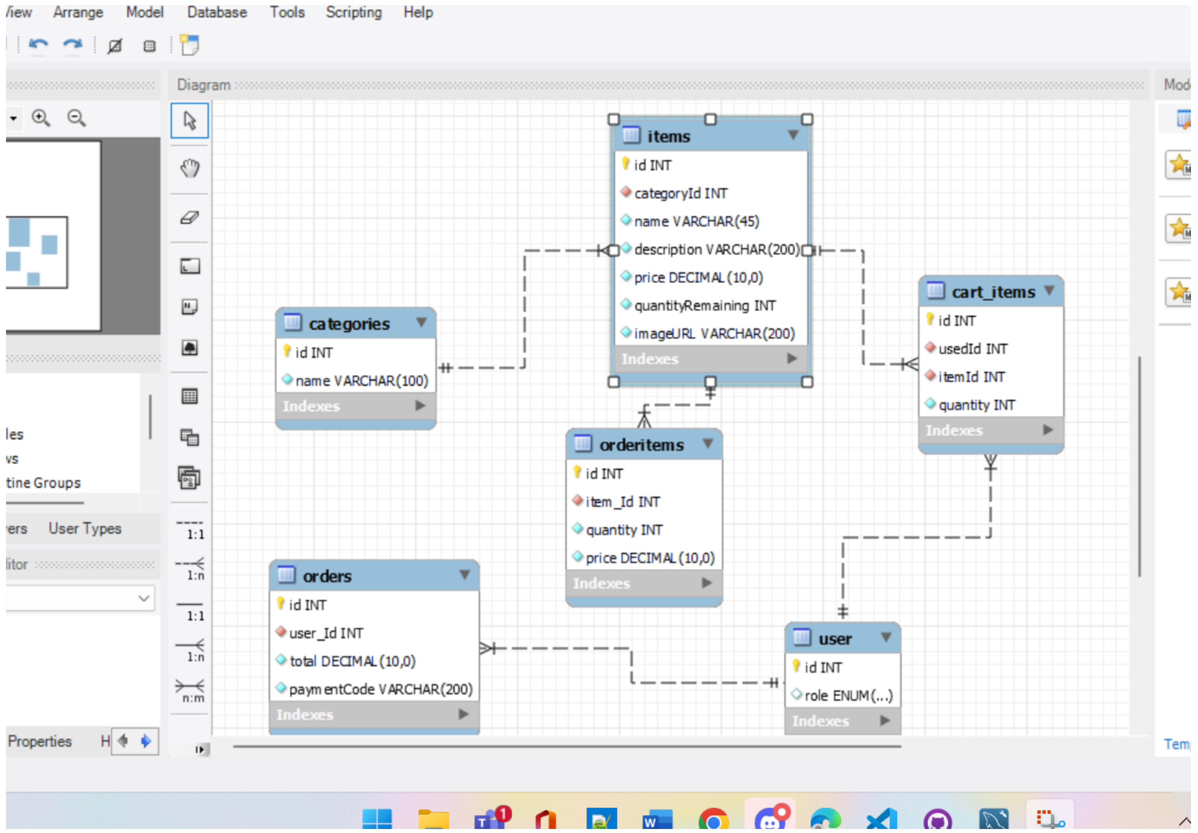
TECHNOLOGIES

We used HTML5, CSS, Bootstrap and javascript for the UI. We used C# asp.net core and EntityFramework razor pages to build our application's physical structure, design it, and manage its behavior, respectively.

We also used Azure for cloud hosting and Azure SQL Server to build our cloud database.

Storage was also on Azure in a blob Storage Container and accessed through a web app and plan

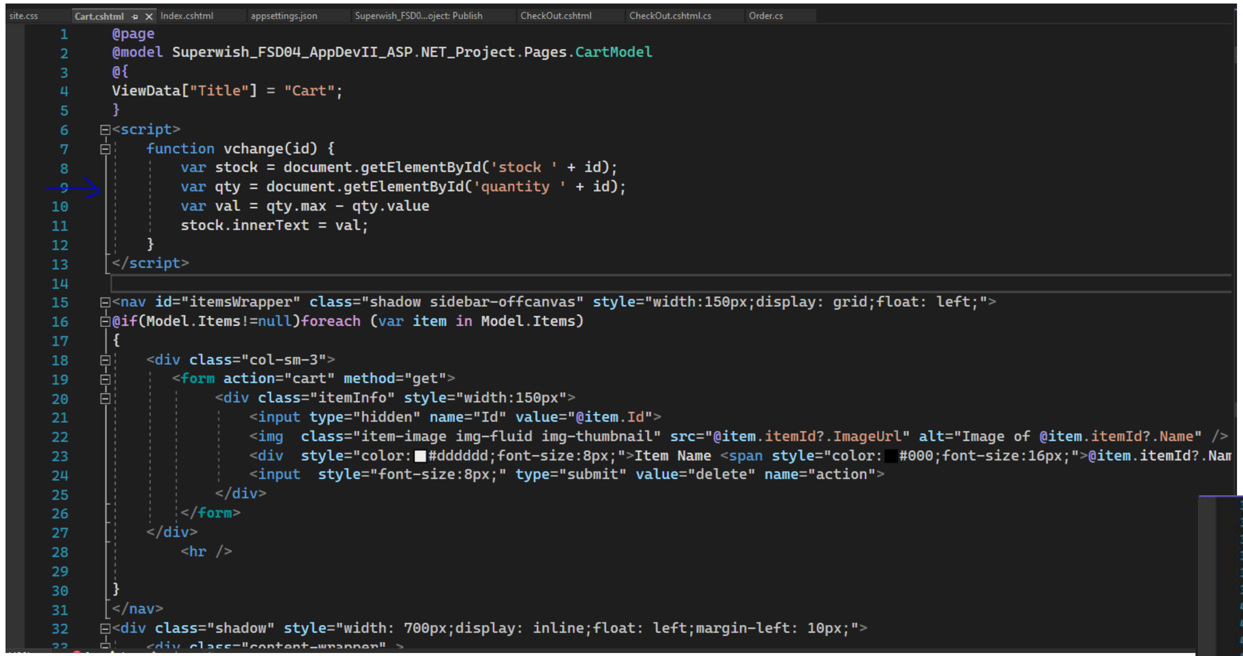
The Database ERD:



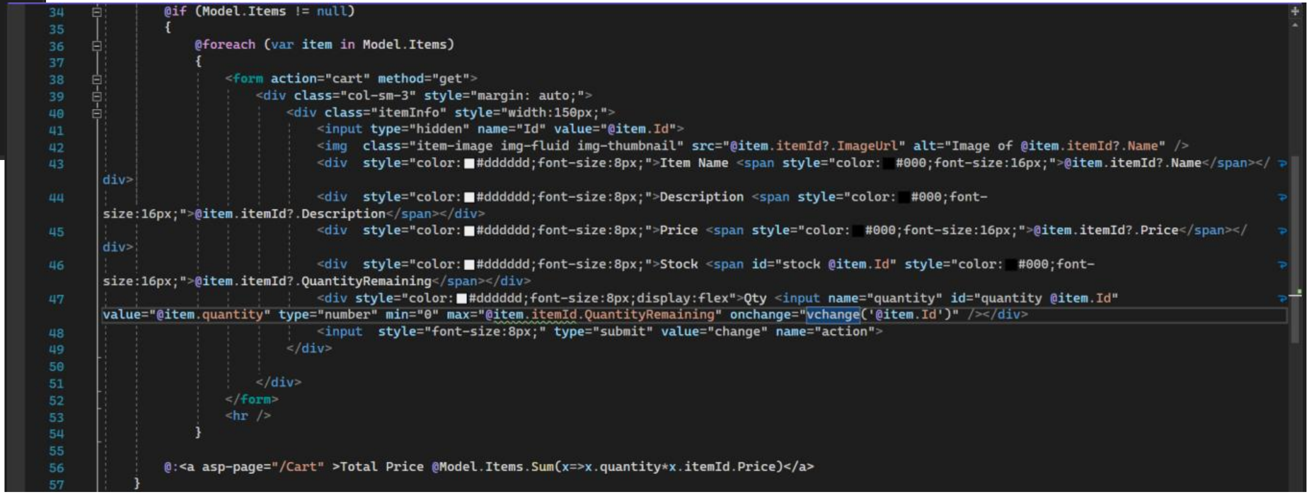
Challenge 1

- Database First with Azure turned out to be deceptively easy
- Our First Real Challenge was configuring the Blob Storage
- There were multiple moving parts and the resources we found had tricky ways to implement it… like setting up the web app and plan to rs group and configuring in program.cs and within the razor pages themselves as well as connecting the storage back to our db

Challenge 2



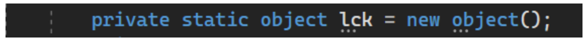
```
@page
@model Superwish_FSD04_AppDevII_ASP.NET_Project.Pages.CartModel
@{
ViewData["Title"] = "Cart";
}
<script>
    function vchange(id) {
        var stock = document.getElementById('stock ' + id);
        var qty = document.getElementById('quantity ' + id);
        var val = qty.max - qty.value
        stock.innerText = val;
    }
</script>

<nav id="itemsWrapper" class="shadow sidebar-offcanvas" style="width:150px;display: grid;float: left;">
@if(Model.Items!=null)foreach (var item in Model.Items)
{
    <div class="col-sm-3">
      <form action="cart" method="get">
           <div class="itemInfo" style="width:150px">
               <input type="hidden" name="Id" value="@item.Id">
               <img  class="item-image img-fluid img-thumbnail" src="@item.itemId?.ImageUrl" alt="Image of @item.itemId?.Name" />
               <div  style="color:□#dddddd;font-size:8px;">Item Name <span style="color:■#000;font-size:16px;">@item.itemId?.Nam
               <input  style="font-size:8px;" type="submit" value="delete" name="action">
           </div>
        </form>
    </div>
        <hr />

}
</nav>
<div class="shadow" style="width: 700px;display: inline;float: left;margin-left: 10px;">
```



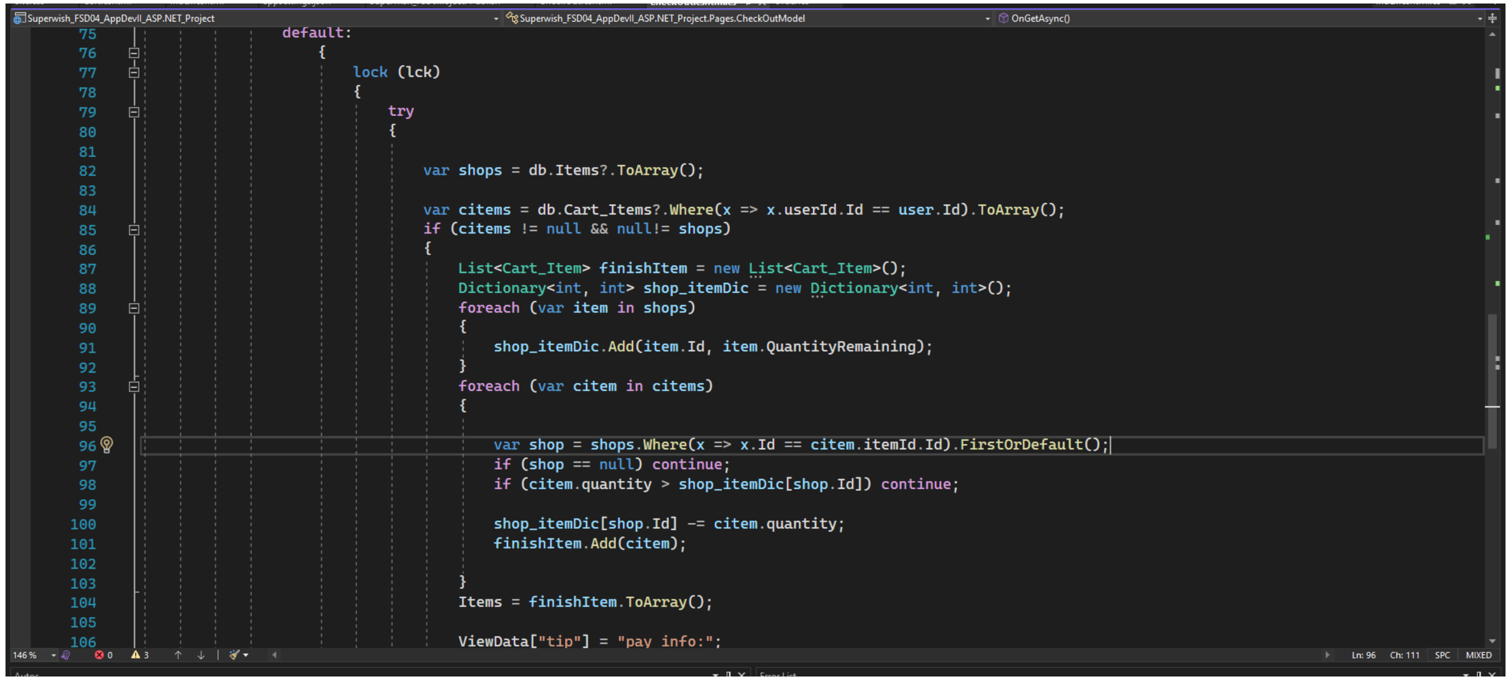
```
            @if (Model.Items != null)
            {
                @foreach (var item in Model.Items)
                {
                    <form action="cart" method="get">
                        <div class="col-sm-3" style="margin: auto;">
                            <div class="itemInfo" style="width:150px;">
                                <input type="hidden" name="Id" value="@item.Id">
                                <img  class="item-image img-fluid img-thumbnail" src="@item.itemId?.ImageUrl" alt="Image of @item.itemId?.Name" />
                                <div  style="color:□#dddddd;font-size:8px;">Item Name <span style="color:■#000;font-size:16px;">@item.itemId?.Name</span></div>
                                <div  style="color:□#dddddd;font-size:8px;">Description <span style="color:■#000;font-size:16px;">@item.itemId?.Description</span></div>
                                <div  style="color:□#dddddd;font-size:8px;">Price <span style="color:■#000;font-size:16px;">@item.itemId?.Price</span></div>
                                <div  style="color:□#dddddd;font-size:8px;">Stock <span id="stock @item.Id" style="color:■#000;font-size:16px;">@item.itemId?.QuantityRemaining</span></div>
                                <div style="color:□#dddddd;font-size:8px;display:flex">Qty <input name="quantity" id="quantity @item.Id" value="@item.quantity" type="number" min="0" max="@item.itemId.QuantityRemaining" onchange="vchange('@item.Id')" /></div>
                                <input  style="font-size:8px;" type="submit" value="change" name="action">
                            </div>

                        </div>
                    </form>
                    <hr />
                }

            @:<a asp-page="/Cart" >Total Price @Model.Items.Sum(x=>x.quantity*x.itemId.Price)</a>
        }
```

Challenge 3

- To get the cart emptied after placing the order
- To update the stock in the database once the order is placed



```
    private static object lck = new object();
```



Superwish_FSD04_AppDevII_ASP.NET_Project
Superwish_FSD04_AppDevII_ASP.NET_Project.Pages.CheckOutModel
OnGetAsync()

```
default:
    {
        lock (lck)
        {
            try
            {

                var shops = db.Items?.ToArray();

                var citems = db.Cart_Items?.Where(x => x.userId.Id == user.Id).ToArray();
                if (citems != null && null!= shops)
                {
                    List<Cart_Item> finishItem = new List<Cart_Item>();
                    Dictionary<int, int> shop_itemDic = new Dictionary<int, int>();
                    foreach (var item in shops)
                    {
                        shop_itemDic.Add(item.Id, item.QuantityRemaining);
                    }
                    foreach (var citem in citems)
                    {

                        var shop = shops.Where(x => x.Id == citem.itemId.Id).FirstOrDefault();
                        if (shop == null) continue;
                        if (citem.quantity > shop_itemDic[shop.Id]) continue;

                        shop_itemDic[shop.Id] -= citem.quantity;
                        finishItem.Add(citem);

                    }
                    Items = finishItem.ToArray();

                    ViewData["tip"] = "pay info:";
```

Roles And Scaffolding

- This was another challenging part… scaffolding ultimately failed because I believe I tried to implement it too late… as Kyle said the issue with reflection of the db Context was very difficult to resolve and eventually I made them by hand.

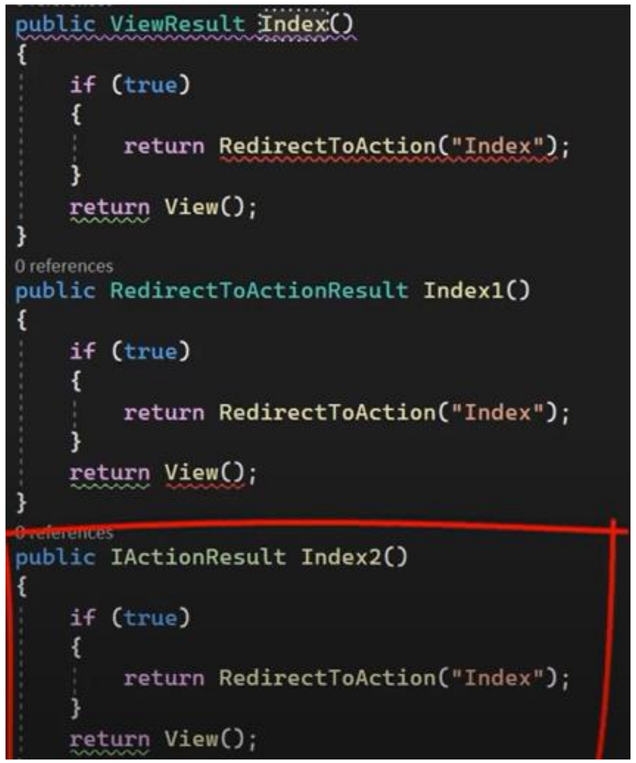
Admin Solution

- Not a perfect fix, still having trouble in layout fixing the nav but the crud functions are there however accessing the urls is still an issue
- The pages and nav on login do render and permission for User is working

Resolving Blob Storage

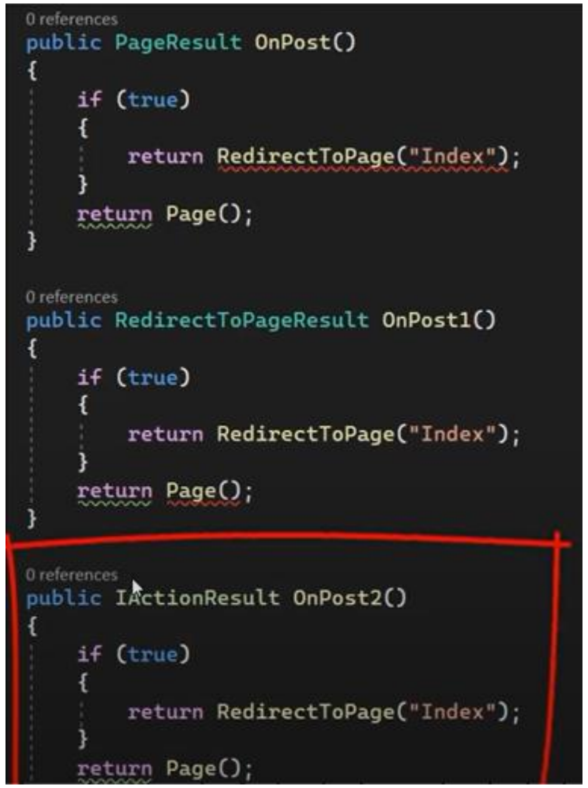
- We managed to get the upload working
- This was a notable milestone and now we can upload to our container
- We're hoping to get the urls back to our database today and this will fix most of our issues with create and edit

Learning 1 – the difference of the MVC application and the razor page application



```
public ViewResult Index()
{
    if (true)
    {
        return RedirectToAction("Index");
    }
    return View();
}
0 references
public RedirectToActionResult Index1()
{
    if (true)
    {
        return RedirectToAction("Index");
    }
    return View();
}
0 references
public IActionResult Index2()
{
    if (true)
    {
        return RedirectToAction("Index");
    }
    return View();
```

MVC Application



```
0 references
public PageResult OnPost()
{
    if (true)
    {
        return RedirectToPage("Index");
    }
    return Page();
}

0 references
public RedirectToPageResult OnPost1()
{
    if (true)
    {
        return RedirectToPage("Index");
    }
    return Page();
}

0 references
public IActionResult OnPost2()
{
    if (true)
    {
        return RedirectToPage("Index");
    }
    return Page();
```

RazorPage Application

Future work

- Stripe for the real-payment
- Admin panel finalized
- Update the UI of the checkout page

Summary

- This application allows users to create a new order, add Items into a cart, update the cart, and checkout the order.
- It helped us learn about ASP.NET core, and C#.
- Further improvements can be made with more development time.