

## Spice Projects Note:

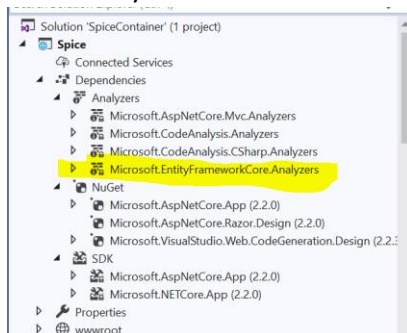
1. When Creating the Project ensure to choose asp.net core and Individual Authentication.
2. Go to startup.cs and edit the connection string with short name such as asp-Spice of data base that you want to create, highlighted below (dark)



```
1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "server=LAPTOP-ALI\\SQLEXPRESS;Database=aspnet-Spice;Trusted_Connection=True;MultipleActiveResultSets=true"
4   },
5   "Logging": {
6     "LogLevel": {
7       "Default": "Warning"
8     }
9   },
10  "AllowedHosts": "*"
11 }
12
```

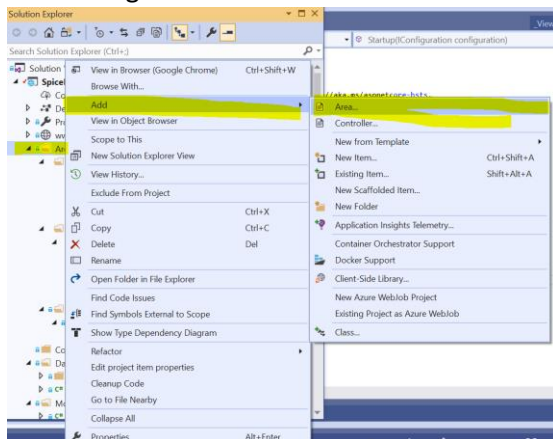
3. Open NuGet Console and type update-database, it will create database in server, If it cannot something is wrong, I have not installed any Entity framework for it to work. Make sure to change the sql server name in connection, highlighted above in yellow.

The above is necessary otherwise you may experience Entity framework errors. These packages are installed by default. See below and it works as it works with additional packages.



## Creating Separate Areas for both Admin and Customer.

1. Right click and create two areas for each one (Customer and Admin)



2. You can actually can make one home, make the customer, in the startup file , in the startup file replace it with original as the following:

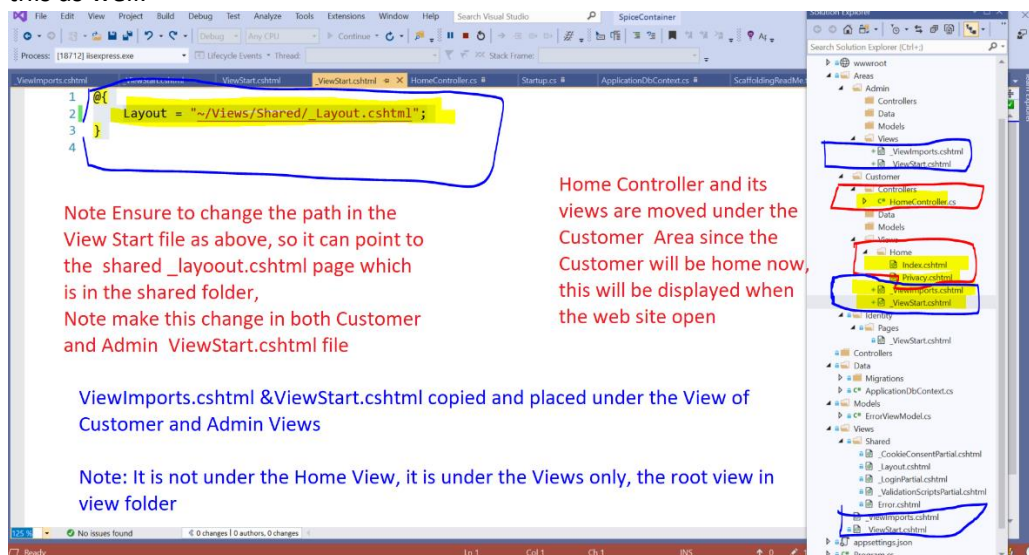
```
routes.MapRoute(
    name: "areas",
    template: "{area=Customer}/{controller=Home}/{action=Index}/{id?}");
```

```

5         }
6     }
7     else
8     {
9         app.UseExceptionHandler("/Home/Error");
10        // The default HSTS value is 30 days. You may want to change this for production s
11        app.UseHsts();
12    }
13
14    app.UseHttpsRedirection();
15    app.UseStaticFiles();
16    app.UseCookiePolicy();
17
18    app.UseAuthentication();
19
20    app.UseMvc(routes =>
21    {
22        routes.MapRoute(
23            name: "areas",
24            template: "{area=Customer}/{controller=Home}/{action=Index}/{id?}");
25    });

```

- 3.
4. Now you need to have the views and separate controller in both customer and admin. Example for customer is shown below: You may not need to copy the ViewStart.cshtml as it work without this as well.



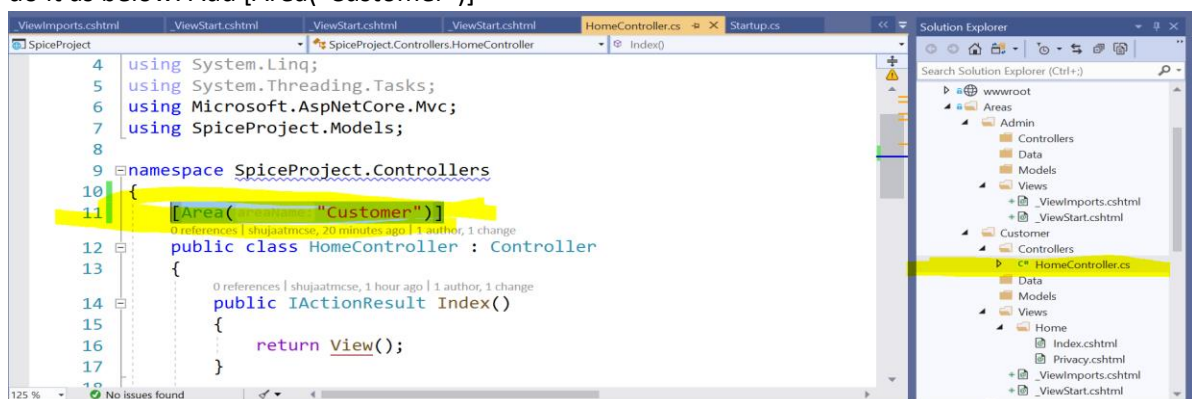
Note Ensure to change the path in the View Start file as above, so it can point to the shared \_layout.cshtml page which is in the shared folder,  
Note make this change in both Customer and Admin ViewStart.cshtml file

Home Controller and its views are moved under the Customer Area since the Customer will be home now, this will be displayed when the web site open

ViewImports.cshtml & ViewStart.cshtml copied and placed under the View of Customer and Admin Views

Note: It is not under the Home View, it is under the Views only, the root view in view folder

5. Now we need to tell the Home Controller which we moved to Customer about the Area. Hence do it as below: Add [Area("Customer")]

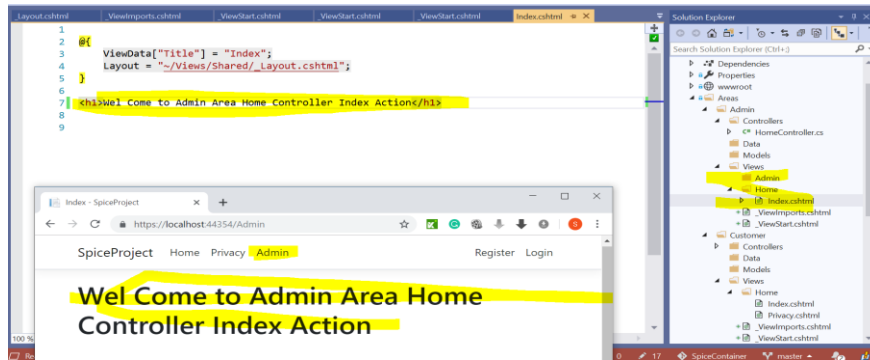


```

4 using System.Linq;
5 using System.Threading.Tasks;
6 using Microsoft.AspNetCore.Mvc;
7 using SpiceProject.Models;
8
9 namespace SpiceProject.Controllers
10 {
11     [Area("Customer")]
12     public class HomeController : Controller
13     {
14         public IActionResult Index()
15         {
16             return View();
17         }
18     }

```

- 6.
7. Below is a screen shot that shown the admin controller as well.



The complete code is on github with commits [Customer&AdminAreaWithReadMeFile](#)

Migration/Model/DBSet.

To create the Category model do the following:

1. create the Category Model Class in Model folder.
2. In your ApplicationDbContext.cs add : `public DbSet<Category> Categories { get; set; }`
3. At Nuget Console type: `add-migration MainCategoryAddedFirstMigration`
4. Type: `Update-database`

To add a foreign Key in the Code first approach, see example below how the parent class is referenced in the sub class.

```
public class SubCategory
{
    public int Id { get; set; }

    public string Name { get; set; }

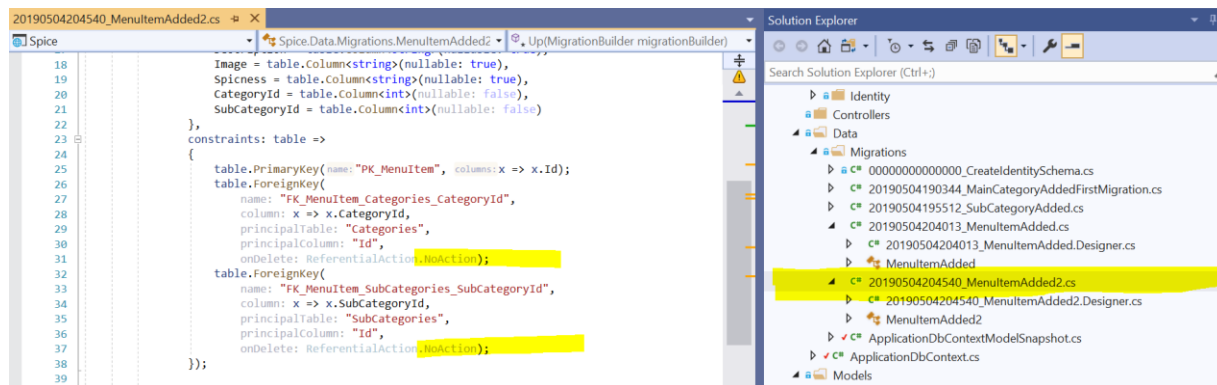
    [Required]
    [Display(Name = "Category")]
    public int CategoryId { get; set; }

    [ForeignKey("CategoryId")] // Category is refenced to the Id of Category in //Category
                              // class, each sub category can belong to a category, the Category class
                              // is referenced after the virtual word.
    public virtual Category Category { get; set; }
}

//The Category Class: Note the Id in Category class is just Id
public class Category
{
    [Key]
    public int Id { get; set; }
    [Required]
    public string Name { get; set; }
}
```

**Adding the Enum for MenuItem and its detail members:**

you can get an error when updating the database at Nuget Console , so go to the context detail file and change the casecad to NoAction, then rerun the update-database command. See example below



The error is because of Foreign Key constraint, if parent deleted what will the child do.

For Adding Enum: The `Html.getEnumList` must be used in the view.

```
<div class="form-group"><label asp-for="Spycness" class="control-label"></label>
<select asp-for="Spycness" class="form-control" asp-items="Html.GetEnumSelectList<MenuItem.Espicy>()"></select></div>
```

Below is a complete class of MenuItem.

```
public class MenuItem
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public string Image { get; set; }

    //Enum Start
    public string Spycness { get; set; } // We will use this as Input in View & this field is saved in
    DATABASE
    public enum Espicy { Na=0, NotSpicy=1, Spicy=2, HotSpicy=3} //This will be Html.get Elements

    //End,

    public decimal Price { get; set; }

    //referencing Category
    public int CategoryId { get; set; }

    [ForeignKey("CategoryId")]
    public Category Category { get; set; }

    //referencing Sub Category
    public int SubCategoryId { get; set; }

    [ForeignKey("SubCategoryId")]
    public SubCategory SubCategory { get; set; }
}
```

## Enum on View...

```
<div class="form-group">
    <label asp-for="Spycness" class="control-label"></label>
    <select asp-for="Spycness" class="form-control" asp-
items="Html.GetEnumSelectList<MenuItem.Espicy>()"></select>
</div>
```

## Enum in DataBase

[illegible]

## Rich Text Editor in the Menu Page:

On the page it self such as on the Menu Create Action Page make sure to include this.

@section Scripts

```
{
    <script>tinymce.init({
        selector:'textarea', plugins: "lists", menubar: 'file edit format'}};</script>}
//Note Don't include only text area, you have to include plugins, and menubar
```

In the Main

In the \_Layout.cshtml page, In Include Enviroment add below, **note** it has API Key for localhost, You will need another key for domain name from the tiny.

```
<script
src="https://cloud.tinymce.com/5/tinymce.min.js?apiKey=koc6wzmfpb6p8drpyd2uoymbfpgbyw
p5ysjd75wr8buorhut"> </script>
```

## Just Viewing the Rich text

```
<dd class="col-sm-10">
    @Html.Raw(Model.Description)
</dd>
```

## Changing the SubCategory Select Option when Main Category is changed

1. If you are expecting JSON data then your controller should have an action which returns data in json format. In this example see the controller action below:

```
public/* JsonResult*/ async Task<IActionResult> GetSubCategories(int?
CategoryIdInController)
{
    if (CategoryIdInController == null)
    {
        return NotFound();
    }

    var subCategory = await _context.SubCategories
        .Include(s => s.Category).Where(c=>c.CategoryId==
CategoryIdInController).ToListAsync();
    if (subCategory == null)
    {
        return NotFound();
    }

    return Json(new SelectList(subCategory, "Id", "Name"));
}
```

2. In View you will basically have 2 select lists, One main which will be populated by default and the other one will be papulated based on the selction of main Category.

Code for main Category:

```
<div class="form-group">
    <label asp-for="CategoryId" class="control-label"></label>
    <select id="CategoryId" asp-for="CategoryId" class="form-control" asp-items="ViewBag.CategoryId"
onchange="GetSubCategory()">
        <option>Please Select One</option>
    </select>
</div>
```

### 3. Code for subcategory

```
<div class="form-group">
    <label asp-for="SubCategoryId" class="control-label"></label>
    <select id="SubCategoryId" asp-for="SubCategoryId" class="form-control">
        <option >Please select one</option>
    </select>
</div>
```

The AJAX, JQUERY AND JS Code to make it work.

@\*Really important to understand how you can get different select list using, jquery, js and ajax.\*@

```
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-
ui.min.js"></script>
```

@\*Ajax plugin is needed, it will work even without adding the above because it is in the\_layout page\*@

@\*This is added to check the wiring of JQuery , if it works.\*@

```
<script type="text/javascript">
    $(document).ready(function(){
        $("#msgid").html("This is Hello World from JQuery");
    });
```

@\* When the Category changes, we want to get the list of subcategory hence using java script for this purpose\*@

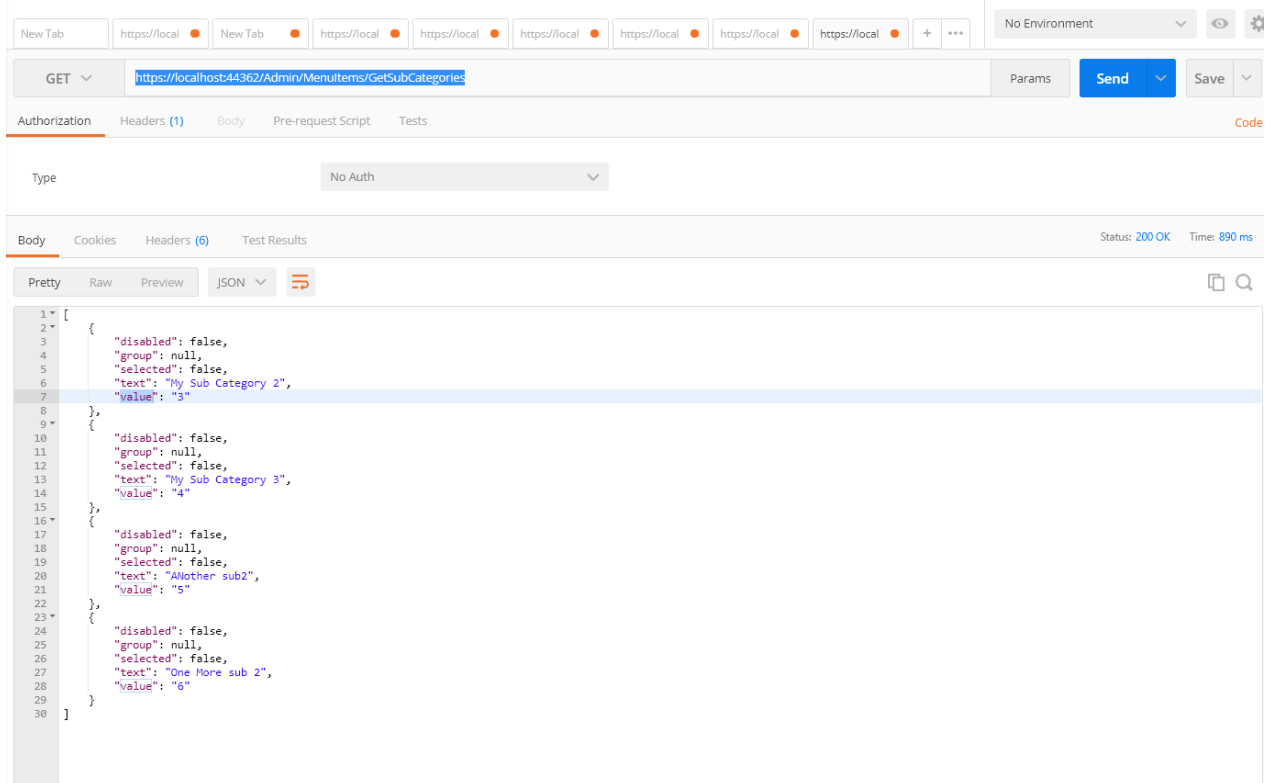
```
function GetSubCategory() {
    //alert("main category selected"); /*again to check wiring that we can
    successfully call the function when category is changed.*/
    console.log("Sub category function called on Category change");
    $.ajax({
        url: '@Url.Action("GetSubCategories", "MenuItems")', /*In the url.action
        we write the action and then controller*/
        contentType: "application/json; charset=utf-8", //the type of content
        datatype: "Json", // Make sure you know you are getting the data in return in
        the json format, try to check it by postman
        //Gethttps://localhost:44362/Admin/MenuItems/GetSubCategories the data
        format will be like [{ value:1, text}, {value:2, text: cat name}, {etc}]
        // You can also send/receive data in other formats
        data: { CategoryIdInController: $(CategoryId).val() },// Here we write
        the Id in the controller action and the one we are sending.I have selected the Id from
        DOM using JQuery
        success: function (data) {
            $('#SubCategoryId').empty();/* To make the sub category list empty, make
            sure you call this method in correct syntax way*/
            $('#SubCategoryId').append("<option value= 0>Please select one
            </option>");
            var k = 0;
            while (k < data.length) {
                $('#SubCategoryId').append("<option value=" +data[k].value+">"
                +data[k].text+"</option>"); // This syntax
```

```

        k++;
        //The syntax above is extremely crucial , make sure you write it
in the correct way, otherwise will go crazy.
        console.log("value" + data[k].value);
        console.log("text" + data[k].value);
    }
},
error: function(reponse) {
    alert("error : No Category found " + response);
}
});
}
</script>

```

The Screen shot of postman, for this to work you action controller should not require Id, so remove the where condition in the GetSubCategories method in Menu item Controller.



## Image/Picture Saving/displaying from/to Database:

```

<form method="post" enctype="multipart/form-data">
  <div asp-validation-summary="All" class="text-danger"></div>
  <div class="form-group">
    <label asp-for="Name" class="control-label"></label>
    <input asp-for="Name" class="form-control" />
    <span asp-validation-for="Name" class="text-danger"></span>
  </div>
  <div class="form-group row">
    <div class="col-2">
      <label asp-for="Picture" class="col-form-label"></label>
    </div>
    <div class="col-5">
      <input type="file" id="projectImage" name="files" multiple class="form-control" />
    </div>
  </div>

```

```
</div>
</form>
```

Note: Pay attention to the yellow highlighted requirements. Above is for Coupon Create Page for saving image into Database.

## Creating /Saving into Database

```
[HttpPost]
[ValidateAntiForgeryToken]
//Receiving of the method is default such as Bind, Model etc
public async Task<IActionResult>
Create([Bind("Id,Name,CouponType,Discount,MinimumAmount,Picture,IsActive")] Coupon coupon)
{
    //If Model state is Valid
    if (ModelState.IsValid)
    {
        var files = HttpContext.Request.Form.Files;
        if (files.Count > 0)
        {
            byte[] p1 = null;
            using (var fs1 = files[0].OpenReadStream())
            {
                using (var ms1 = new MemoryStream())
                {
                    fs1.CopyTo(ms1);
                    p1 = ms1.ToArray();
                }
                coupon.Picture = p1;
            }
            _context.Add(coupon);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
    }

    //If model state is not valid.
    return View(coupon);
}
```

Note: The Yellow Marked section needs to be noted

## Displaying Image/ Detail Page

```
// GET: Admin/Coupons/Details/5 Nothing special
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var coupon = await _context.Coupon
        .FirstOrDefaultAsync(m => m.Id == id);
    if (coupon == null)
    {
        return NotFound();
    }

    return View(coupon);
}
```

View/ Detail View Below will display the image:



```

@if (Model.Picture.Count() > 0)
{
<div class=" col-12 border">
    @{
        var base64 = Convert.ToBase64String(Model.Picture);
        var imgsrc = string.Format("data:image/jpg;base64,{0}", base64);
    }
    
</div>
}

```

**// Rest of the action would be like this**

```

<div class="col-12 pt-4">
    <fieldset disabled>
        <div class="form-group row">
            <div class="col-2">
                <label asp-for="Name" class="custom-label"></label>
            </div>
            <div class="col-5">
                <input asp-for="Name" class="form-control" />
            </div>
        </div>
    </fieldset>
</div>

```

**Note: Other fields as may be needed**

```

</fieldset>

```

```

</div>

```

Displaying different coupons Images every 2 seconds

@\*Note you are calling here the count Method, so make sure you type it as ..Count()>0, with braces\*@

```

@if (Model.Coupon.ToList().Count() > 0)
{
    <div class="border">
        @*the data interval determine how quickly the copon should be displayed.*@
        <div class="carousel" data-ride="carousel" data-interval="1000">
            @for (int i = 0; i < Model.Coupon.Count(); i++)
            {
                if (i == 0)
                {
                    <div class="carousel-item active">
                        @{
var base64 = Convert.ToBase64String(Model.Coupon.ToList()[0].Picture);
var imgsrc = string.Format("data:image/jpg;base64,{0}", base64);
                        }
                        @*Below we are now actually display image.*@
                        
                    </div>
                }
                else
                {
                    @* it will displayed when not active.
                    <div class="carousel-item">
                        @{
var base64 = Convert.ToBase64String(Model.Coupon.ToList()[i].Picture);
var imgsrc = string.Format("data:image/jpg;base64,{0}", base64);
                        }
                        
                    </div>
                }
            }
        </div>
    </div>
}

```

```

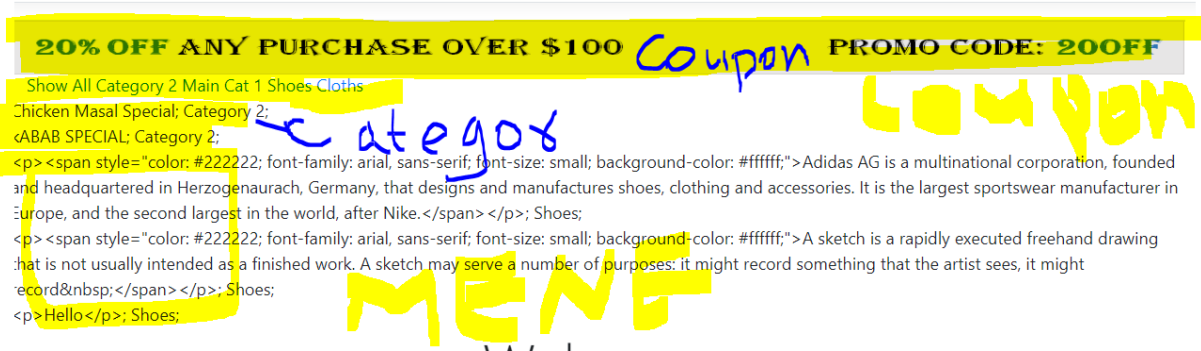
    </div>
  </div>
}

```

## ViewModel

If you are creating you may need to have IEnumerable/List of the various model that you want to combine in one model.

See the landing page example, where want to have Categories, MenuItem, and Coupons, hence



```

namespace Spice.ViewModel
{
    public class LandingPageViewModel
    {
        public IEnumerable<MenuItem> MenuItem { get; set; }
        public IEnumerable<Coupon> Coupon { get; set; }
        public IEnumerable<Category> Category { get; set; }
    }
}

```

Using it in Controller action,

```

//Receiving Category Id, Optional
public async Task<IActionResult> Index( int ? Id)
{
    LandingPageViewModel vm = new LandingPageViewModel();
    MenuItem menu = new MenuItem();
    //Id is received when a specific Category is selected.Its the category Id
    if (Id != null)
    {
        // Checking first if the Id is a genuine
        var menuItems = await _context.MenuItem.Include(s =>
            s.SubCategory).Include(c => c.Category).Where(c => c.CategoryId == Id).ToListAsync();

        if (menuItems == null)
        {
            return NotFound();
        }
        else // If a specific category has been selected
        {
            vm.Category = await _context.Categories.ToListAsync();
            //Will display coupons/offers which currently active, administrator
            //will the opportunity to activate/deactivate coupons
            vm.Coupon = await _context.Coupon.Where(a => a.IsActive == true).ToListAsync();
            vm.MenuItem = menuItems;
            return View(vm);
        }
    }
}

```

```

    }
    else// If No Id has been received(no specific selection), all menuitems are
displayed
    {
        //Retrieving list of only those coupons which are active.
        vm.Category = await _context.Categories.ToListAsync();
        vm.Coupon = await _context.Coupon.Where(a => a.IsActive == true).ToListAsync();
        //Note use the include where there is a relationship involve.
        vm.MenuItem = await _context.MenuItem.Include(c => c.Category).Include(s =>
s.SubCategory).ToListAsync();
    }
    return View(vm);
}

```

In View/Page

Note: Below is for Category and menu selection, the coupon image code is up under images.

@model Spice.ViewModel.LandingPageViewModel

```

@{
    ViewData["Title"] = "Home Page";
}
<div class="container">
    <a asp-action="Index" asp-controller="Home" >Show All</a>
    @foreach (var item in Model.Category)
    {
        <a asp-action="Index" asp-route-id="@item.Id">@item.Name</a>
    }
</div>

<div class=" container">
    @foreach (var menu in Model.MenuItem)
    {
        <div class=" row">
            @menu.Description;
            @menu.Category.Name;
        </div>
    }
</div>

```

## Image/Picture Saving on server & path in Database:

```

private readonly ApplicationDbContext _context;

//Step1 in Controller dependency injection, include the ASP.NETCore hosting
environment.
private readonly IHostingEnvironment _hostingEnvironment;
public MenuItemsController(ApplicationDbContext context, IHostingEnvironment
hostingEnvironment)
{
    _context = context;
    //Step1 in Controller, include the ASP.NETCore hosting environment.
    _hostingEnvironment = hostingEnvironment;
}

```

```

[HttpPost]
[ValidateAntiForgeryToken]

//Here we are saving file on server not on into DB, th ereceiving is as usual
public async Task<IActionResult>
Create([Bind("Id,Name,Description,Image,Spycness,Price,CategoryId,SubCategoryId")]
MenuItem menu)
{
    // If Modle is valid
    if (ModelState.IsValid)
    {
        // lets save the model first to the database.
        //We made a copy of the object, so when it is saved into DB, it will have
the Id assign to it.
        var newItem = menu;
        _context.Add(newMenuItem);
        await _context.SaveChangesAsync();

        // declare the root path ,, wich is www
        string webRootPath = _hostingEnvironment.WebRootPath;
        // files variable
        var files = HttpContext.Request.Form.Files;

        //if user has uploaded file
        if (files.Count > 0)
        {
            // Creating Path where you want the image to be saved, it will be to
root folder www
            var path = Path.Combine(webRootPath, @"images\Products\");
            // Get the extension of the uploaded file i.e Jpg
            var extension = Path.GetExtension(files[0].FileName);

            // Creating the actual file in the specified directory/path with
Model.Id & its extension, you may choose it with diffrent id as you plan
            using (var fileStream = new FileStream(Path.Combine(path,
newMenuItem.Id + extension), FileMode.Create))
            {
                //Copying single file
                files[0].CopyTo(fileStream);
            }
            //Saving Path of the Image into Database
            newMenuItem.Image = @"\images\Products\" + newMenuItem.Id +
extension;
        }
        //no file was uploaded, so use default image which is on the server, note
the sc is just a path.
        else
        {
            // This will copy the default file and save it with Id name
            //var uploads = Path.Combine(webRootPath, @"images\" +
SC.DefaultFoodImage);
            //System.IO.File.Copy(uploads, webRootPath + @"\images\Products\" +
newMenuItem.Id + ".png");

            ////Saving Path of the Image into Database
            //newMenuItem.Image = @"\images\Products\" + newMenuItem.Id + ".png";

```

```

        // Instead of copying the default file again and again, you could
        point to the default file once only
        // Hence no need to copy
        newItem.Image = @"\images\" + SC.DefaultFoodImage;
    }
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

// If Model is not Valid
ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name",
menu.CategoryId);
ViewData["SubCategoryId"] = new SelectList(_context.SubCategories, "Name",
"Name", menu.SubCategoryId);
return View(menu);
}

```

### In View Create Action:

@model Spice.Models.MenuItem

```

@{
    ViewData["Title"] = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Create</h1>

<h4>MenuItem</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        @*change the form method to include support for sending files as per below line*@
        <form method="post" asp-action="Create" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            .....
            .....etc.....
            <div class="form-group">
                <label asp-for="Image" class="control-label"></label>
                @*make sure to add the line as below to pass images espially type="file"
name="files"*@
                <input type="file" name="files" asp-for="Image" class="form-control" />
                <span asp-validation-for="Image" class="text-danger"></span>
            </div>
        </form>

```

### On Detail View/ Just Viewing of Image

```

<dd class="col-sm-10">
    
    @*@Html.DisplayFor(model => model.Image)*@
</dd>

```

In controller for delete action.

```
// POST: Admin/MenuItems/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var menuItem = await _context.MenuItem.FindAsync(id);
    string webRootPath = _hostingEnvironment.WebRootPath;
    if (menuItem != null)
    {
        //Checking if the MenuItem has a default image/picture or one was
        uploaded by user.
        //We do not want to delete the default image on server as this will be
        used by menuitem
        //as default when user does't upload one
        //Path of the item in Db
        var menuItemPathInDb = menuItem.Image.ToString().TrimStart('\\');
        // The default image path on server
        var pathOnServer = (@"images\" + SC.DefaultFoodImage);
        // If there both path are not the same then we want to delete the image
        // so we only delete the user uploaded image.
        if (menuItemPathInDb != pathOnServer)
        {
            var imagePath = Path.Combine(webRootPath, menuItemPathInDb);

            if (System.IO.File.Exists(imagePath))
            {
                System.IO.File.Delete(imagePath);
            }
        }
        _context.MenuItem.Remove(menuItem);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return RedirectToAction(nameof(Index));
}
```

**Adding the Unlock and Locking functionality for users:** We need to create method in user controller for locking and unlocking the user and then do the necessary things in View:

```
// For locking the user
public async Task<IActionResult> Lock(string Id)
{
    if (Id == null)
    {
        return NotFound();
    }
    var UserInDb = await _context.ApplicationUser.FirstOrDefault(u => u.Id == Id);
    if (UserInDb == null)
    {
        return NotFound();
    }
    // Note that we adding year and assigning it to it property ,
    the lockoutEnd is not a method but rather a property
    UserInDb.LockoutEnd = DateTime.Now.AddYears(1000);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

```

public async Task<IActionResult> Unlock(string Id)
{
    if (Id == null)
    {
        return NotFound();
    }
    var UserInDb = await _context.ApplicationUser.FirstOrDefaultAsync(u => u.Id == Id);
    if (UserInDb == null)
    {
        return NotFound();
    }
    // Just setting the value to current time
    UserInDb.LockoutEnd = DateTime.Now;
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

```

In View since , it is mostly in Idex page with in foreach

```

<th>
    User Status
</th>
@foreach (var item in Model){

    @*Note: We are refrencing ITEM below not USER*@
    <td>
        @if (item.LockoutEnd == null || item.LockoutEnd < DateTime.Now)
        {
            <a asp-action="Lock" asp-route-id="@item.Id" class="btn btn-
success"><i class="fas fa-lock-open"></i></a>
        }
        else
        {
            <a asp-action="Unlock" asp-route-id="@item.Id" class="btn btn-danger
text-white"><i class="fas fa-lock"></i></a>
        }
    </td>
}

```

**Authorization, Authentication:** Authentication is identifying the user by login in , authorization to see if the user has access to those resources.

To add authorization on top of the controller write

**[Authorize(Roles = SC.ManagerRole)]** we added to all the controller in Authorization.

Showing or not Show nav-link to user.

Just start by **@if (User.IsInRole(SC.Manager)){**

i.e put the Whole Li or Div within it.

**}**

You can user OR condition extra, In this project we put the Whole Administration Link in it.

## Shopping Cart

We need to have shopping cart icon where the number of item in the cart will appear, so In the view we should add an icon from font awesome and add anchor tag (a) to Li item which could be as below:

```
<li style="color:white">
  <a href="#" class="nav-link">
    <i class="fas fa-shopping-cart"></i> &nbsp; (0)
  </a>
</li> Note: (0) shows number of item in Cart
```

We should also use the font awesome css

```
<link rel="stylesheet" href="~/lib/fontawesome-free-5.8.2-web/css/all.css" />
```

In the above case font awesome is under www/lib folder.

Next We need to add a class/model and add it into DB. So the model would include:

```
public class ShoppingCart
{
    //We want the count of an item to be shown as 1 by default
    public ShoppingCart()
    {
        Count = 1;
    }
    public int Id { get; set; }

    public string AppliUserId { get; set; }
    //We do not need to map, but yes we want to store the user Id ,
    //map means likes refrencing the Foreign key and linking it.
    [NotMapped]
    [ForeignKey("AppliUserId")]
    public virtual ApplicationUser ApplicationUser { get; set; }

    public int MenuItemId { get; set; }
    [NotMapped]
    [ForeignKey("MenuItemId")]
    public virtual MenuItem menuItem { get; set; }

    [Range(1,int.MaxValue, ErrorMessage ="Please enter value greater than or equal to 1")]
    public int Count { get; set; }
}
```

Next we need to create the Cart controller and setup session variable, we will use session variable so that the number of item in the Cart could displayed from any page, and we donot need to access the DB again and again. Like from \_layoutout when each times it is loaded , it will access the session variable count data only.

```
[Authorize]
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AddToCart(ShoppingCart CartObject)
{
    CartObject.Id = 0;
    if (ModelState.IsValid)
    {
        // Getting the user Identiyy to know its User Id
        var claimsIdentity = (ClaimsIdentity)this.User.Identity;
        var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);
        CartObject.AppliUserId = claim.Value;
        // Retreiving cartfrom Db which is added by this user and has this menu Item
```



```

ShoppingCart cartFromDb = await _context.shoppingCart.Where(c => c.AppliUserId ==
CartObject.AppliUserId && c.MenuItemId == CartObject.MenuItemId).FirstOrDefaultAsync();
    // No cart data found than we add the user selection i.e Cart object with
count field
    if (cartFromDb == null)
    {
        await _context.shoppingCart.AddAsync(CartObject);
    }
    // In case where user has already entered the cart data and it is in DB, we
want to just changed the Quantity
    else
    {
        cartFromDb.Count = cartFromDb.Count + CartObject.Count;
    }
    // Saving changes
    await _context.SaveChangesAsync();
    // Saving the quatity in a session variable
    var count = _context.shoppingCart.Where(c => c.AppliUserId ==
CartObject.AppliUserId).ToList().Count();
    HttpContext.Session.SetInt32(SC.sSsCartCount, count);
    return RedirectToAction("Index", "Home");
}

//If Model is not Valid we need to send back the model to user
else
{
    var menuItemFromDb = await _context.MenuItem.Include(m =>
m.Category).Include(m => m.SubCategory).Where(m => m.Id ==
CartObject.MenuItemId).FirstOrDefaultAsync();

    ShoppingCart cartObj = new ShoppingCart()
    {
        menuItem = menuItemFromDb,
        MenuItemId = menuItemFromDb.Id
    };
    return View(cartObj);
}
}

```

Note :We should also setup the session variable in Startp.cs file .

In the configurationService method modify the appropriate serve to the following, **note this service has Bootstrap and EF.**

```

services.AddIdentity<IdentityUser, IdentityRole>()
    .AddDefaultTokenProviders()
    .AddDefaultUI(UIFramework.Bootstrap4)
    .AddEntityFrameworkStores<ApplicationDbContext>();

```

Ensure to add use session. Add the below just above MVC.

```
app.UseSession();
```

Next we need access or setup the session variable from \_layoutout page(only getting its value so that it is displayed in the cart, On the login page we need to read the db and see if there is anything in cart of this user after login, if yes we want set the value in session variable, so other pages can access it, on the log out page, we need to set the session variable value to zero, so the cart would not display anything. Also note each time an item is added to cart shown in above Code AddtoCart Task, we need to set the session variable value.

First lets do it on \_LayoutOut Page:

Ensure to add fonts awesome.

@\*Checking if there is any value in session variable so that value will be displayed in the Cart.\*@

```
@if
(HttpContextAccessor.HttpContext.Session.GetInt32(SC.sSsCartCount) != null)
{
    <li style="color:white">
        <a asp-area="Customer" asp-controller="ShoppingCarts"
asp-action="Index" class="nav-link">
@{ var count = HttpContextAccessor.HttpContext.Session.GetInt32(SC.sSsCartCount); }
        <i class="fas fa-shopping-cart"></i> &nbsp; (@count)
        </a>
    </li>
}
@*If the session variable has nothing we want to show zero items in the Cart*@
else
{
    <li style="color:white">
        <a href="#" class="nav-link">
        <i class="fas fa-shopping-cart"></i> &nbsp; (0)
        </a>
    </li>
}
```

Next we need to setup the value in Login page so as the user log in we will setup the value in session variable. In Identity/Pages/ Login.cshtml.cs a razor page.

Lets add database dependency injection.

In the constructor add DbContext.

```
private readonly ApplicationDbContext _context;
// Added Db Context so we can query database for user identity
public LoginModel(SignInManager<IdentityUser> signInManager, ILogger<LoginModel> logger, ApplicationDbContext context)
{
    _signInManager = signInManager;
    _logger = logger;
    _context = context;
}
```

Now after user succesfully login we need to setup value in session variable as below.

```
if (result.Succeeded)
{
    // Once the user login successfully we want to set the value of count
in session in the shopping cart.
    //we need to read the database to know this.
    //Make sure to use entity framework otherwise asych would not work
var user = await _context.Users.Where(u => u.Email ==
Input.Email).FirstOrDefaultAsync();
    // List<ShoppingCart> lstShoppingCart = await
_context.shoppingCart.Where(u => u.AppliUserId == user.Id).ToListAsync();
    var count = await _context.shoppingCart.Where(u => u.AppliUserId ==
user.Id).CountAsync();
    HttpContext.Session.SetInt32(SC.sSsCartCount, count);
    _logger.LogInformation("User logged in.");
    return LocalRedirect(returnUrl);
}
```

In the logout page we need to set the session variable data to zero.

So on the page Identity/Page/Logout.cshtml.cs

```
await _signInManager.SignOutAsync();
```

```

//Below set the session to 0- zero of the shopping cart, at the logout
HttpContext.Session.SetInt32(SC.sSsCartCount, 0);
_logger.LogInformation("User logged out.");
....-----.....,,,,, etc.

```

## Order- Cart

We need to make to Models/Class one would be for Order Header that would contain key information of Order such as total, original total(without coupon), coupon status, Order number. While the order details would includes only all items of the order and its order Id that belong. However we need to show them together hence we will make ViewModel as.

```

public class OrderHeader
{
    //Primary Key
    [Key]
    public int Id { get; set; }

    //Required
    public string UserId { get; set; }

    // Note this require different DataAnnotationn than the regular one, make
    sure to write it correctly.
    [ForeignKey("UserId")]
    public virtual ApplicationUser applicationUser { get; set; }

    public decimal OrignalTotal { get; set; }

    [DisplayFormat(DataFormatString = "{0:C}")]
    [Display(Name = "Order Total")]
    [Required]
    public double OrderTotal { get; set; }

    [Required]
    [Display(Name = "PickUp Time")]
    public DateTime PickUpTime { get; set; }

    // The Following property is not mapped because we will use the date from
    here and use it in Pickup Time
    [Required]
    [Display(Name = "Pickup Date")]
    [NotMapped]
    public DateTime PickUpDate { get; set; }

    //Checking/ Using Coupon, Not using the Coupon Object
    //for Security reason,User Will enter the coupon details.

    public string CouponCode { get; set; }
    public string CouponCodeDiscount { get; set; }
    public string CoupnStatus { get; set; }

    public string PaymentStatus { get; set; }
}

```

```

        public string Comments { get; set; }

        [Display(Name = "Pick Up by :")]
        public string PickupName { get; set; }

        [Display(Name = "Phone Number")]
        public string PhoneNumber { get; set; }
        public string TransactionId { get; set; }
    }

    public class OrderDetail
    {
        [Key]
        public int Id { get; set; }

        public int OrderId { get; set; }

        [ForeignKey("OrderId")]
        public virtual OrderHeader OrderHeader { get; set; }

        public int MenuItemId { get; set; }

        [ForeignKey("MenuItemId")]
        public virtual MenuItem MenuItem { get; set; }

        // Adding detail of the Each MenuItem
        public string Name { get; set; }
        public int Count { get; set; }
        public string Discription { get; set; }
        public double Price { get; set; }
    }
}

```

Now we need to create View Model that would have list of all the items in shopping cart, in other list of menu item in shopping cart- Hence I had to map the shopping cart Menu Item so that I can navigate to also view menu item. Previously it was not mapped.

Another item in the View Model will be Order Header which has detail of Order total.

See below the **ViewModel**.

```

//We are creating this view model , so to get the data for the Index page of Cart
// The Cart Index page has a list of menuitem and its quantity+ Order Total and Order coupon
// In shopping Cart the the menuitem is not mapped hence we cannot not navigate through it.
// so we are using list of shopping cart instead of just using .inculde in the controller to get the cart data
// I HAD TO MAP IT.
public class OrderHeaderShoppingCartViewModel
{
    public IEnumerable<ShoppingCart> ShoppingCart { get; set; }
    public OrderHeader OrderHeader { get; set; }
}

```

Now Index View when a user click the shopping icon we have to display list of all those items and implement the ViewModel.

See below

... THIS INDEX is modified

```
// GET: Customer/ShoppingCarts
// Getting the shopping cart items.
public async Task<IActionResult> Index()
{
    // order header object is created, we need to iterate through the list of
menuItem and add the prices to orderHeader.
    OrderHeader orderHeader = new OrderHeader();
    // Initialized order total to Zero.
    orderHeader.OrderTotal = 0;

    // Getting the user Identity to know its User Id
    var claimsIdentity = (ClaimsIdentity)this.User.Identity;
    var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);

    // Retrieiving cart from Db which is added by this user
    var cart = await _context.shoppingCart.Include(m=>m.menuItem).Where(c =>
c.AppliUserId == claim.Value).ToListAsync();

    if (cart != null)
    {
        // Adding the Order Total
        foreach (var item in cart)
        {
            orderHeader.OrderTotal = orderHeader.OrderTotal + item.menuItem.Price
* item.Count;
        }

        OrderHeaderShoppingCartViewModel detailCart = new
OrderHeaderShoppingCartViewModel()
        {
            ShoppingCart = cart,
            OrderHeader = orderHeader
        };

        // At this time coupon code is not yet applied
        return View(detailCart);
    }

    return View();
}
```

/// SEE BELOW THE CORRECT WAY OF INDEX PAGE

```
// GET: Customer/ShoppingCarts
// Getting the shopping cart items.
public async Task<IActionResult> Index()
{
    // Creating an object of Orderheader and initializing it to zero, becouse
// we will be adding OrderHeader.OrderTotal previous value to a new value. so
if no previous value then it will give error
    OrderHeader orderHeader = new OrderHeader();
    // Initialized to zero, to avoid error
```

```

        orderHeader.OrderTotal = 0;
        OrderHeaderShoppingCartViewModel detailCart = new
OrderHeaderShoppingCartViewModel()
        {
            OrderHeader= orderHeader
        };
        // Initialized order total to Zero.
        orderHeader.OrderTotal = 0;

        // Getting the user Identity to know its User Id
        var claimsIdentity = (ClaimsIdentity)this.User.Identity;
        var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);

        // Retrieving cart from Db which is added by this user
        // and assigning it to the detailCard Shopping cart List
        detailCart.ShoppingCart = await _context.shoppingCart.Where(c =>
c.AppliUserId == claim.Value).ToListAsync();
        if (detailCart.ShoppingCart != null)
        {
            // Adding the Order Total
            // And since the Menu Item is not mapped we can not use the Include
statement above and will not be able to navigate to
            // Include Menu Item so we have to assign manually MenuItem object to the
detailCart.Shopping Cart List
            // We know we have the menu Item Id so we will get Menu Item Object for
each of that Id and we also have MenuItem Object
            // but we cannot not navigate through it.
            foreach (var item in detailCart.ShoppingCart)
            {
                //since we have not mapped the menuItem we have to manually add each
MenuItem details to the Shopping cart list object
                // Creating the Menu Item object
                var menuItem = await _context.MenuItem.Where(m => m.Id ==
item.MenuItemId).FirstAsync();
                // Important thing to note here is that we are change the menuItem
itself through for each loop.
                // Assigning and changing the MenuItem
                item.menuItem = menuItem;
                // now adding details of menu Item.
                detailCart.OrderHeader.OrderTotal =
detailCart.OrderHeader.OrderTotal + item.menuItem.Price * item.Count;
            }
            // At this time coupon code is not yet applied
            return View(detailCart);
        }

        return View();

        // You may be thinking why we are not mapping the MenuItem to avoid the manual
addition.
        // You can do that but then when you want to add menu Item to the List, it will
give error since the incoming Object will
        // not have Id of Category or subcategory. If you map it, then the only solution
may be to retrieve the MenuItem Object for that Id
        // and then add that object to Database.
    }

```