**ASP.NET CORE Project**

A diagram of a diagram

Description automatically generated

* [**Login page**](#login)  (admin, customer) 
* **2 Layouts,** [**one**](#custnav)for **customers pages** and [another Layout](#adminnav)  for **admin** pages 
* In the Layouts include the **NavBar** 
* [3 Tables](#tables) (users, items, orders) 

**For Customers Add following pages**:

* + Only Customers can access these pages
  + [**Customer Home page**](#customerhome) that shows a Welcome title\_\_\_\_\_ , discounted items 
  + [**My Purchase page**](#mypurchase) that shows: all bought items title, price \_\_\_\_
  + [**Items image slider page**](#imagesilder) where **Customer** Get to see all items and its prices 
    - ADD **buy link** to the Items where **Customer** can click to make the buy at the [**buy page**](#buy)  
  + Create [buy **page**](#buy)
    - Show the item detail note that the price with will have 10% discount if it is on a discount (javascript) 
    - If the buy is submitted then it saves these details (customer name, item id , date of buy ) in the order table \_\_\_\_\_
  + Create a [**register page**](#customerregistration):A user can register himself as a customer from the **customer registration page**  \_
    - For this register page include proper validation \_\_\_\_
* Mobile app to allow customer to login and buy from shopping cart \_\_\_\_

**For Admin Add following pages:**

* + Only Admin can access these pages 
  + [**Admin Home page**](#adminhome) that shows a Welcome title, \_\_\_\_\_
  + [**Manage Items**](#itemmanage) **page where admin can manage any item** \_\_\_\_\_
  + Send [**email page**](#emai) add a welcome email to a customer \_\_\_\_\_
  + [**Items List page**](#itemlist) within 
  + [**Insert a new admin page**](#addadmin). validate all inputs
  + [**User Search page** to search](#customersearch) a user based on his role \_\_\_
  + [**Manage users page where**](#usermanagement) **admin can manage any item** \_\_\_\_\_

* + **[Purchase report page](#purchaseReport)** that will show all customers total purchase amount of in a table, When clicking customer order detail will be displayed ordered by the purchase date \_\_\_\_
  + Dashboard where customer can see the statistics category items and the sum of bought items quantity 

**So finally the Project will have 4 Controllers with the Following actions**

1. **Usersall Controller (Entity Framework)**
   * [Index action](#usermanagement) (default) \_
   * Details action (default) 
   * Edit action (default) 
   * Delete action (default) 
   * [Create action](#usercreate) (default) 
   * [Users\_search](#customersearch) action 
   * [addadmin](#addadmin) action
   * [registration](#customerregistration) action
   * [customer\_home](#customerhome) action 
   * [admin\_home](#adminhome) action 
   * [email](#emai) action
   * [login action](#login) 
2. **Items Controller (Entity Framework)**
   * Index action (default)
     + [For admin](#itemindex) it shows all controls that manages items
     + [For customer](#itemindexcustomer) only detail button is available 
   * [Details](#itemDetail) action (default) 
   * Edit action (default) 
   * Delete action (default) 
   * Create action (default) 
   * [list](#itemlist) action 
   * [image\_slider](#imagesilder) 
   * Dashboard action 
3. **Orders Controller**
   * **create\_order action** \_\_\_\_\_\_\_\_
   * [order\_detail](#orderdetail) action \_\_\_\_\_\_\_\_
   * my\_[purchase](#mypurchase) action \_\_\_\_\_\_\_\_

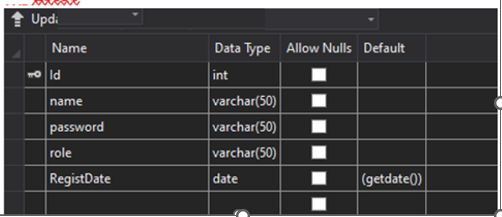
* + [purchase\_report](#purchaseReport) action \_\_\_\_\_\_\_\_
  + [buy](#buy) action

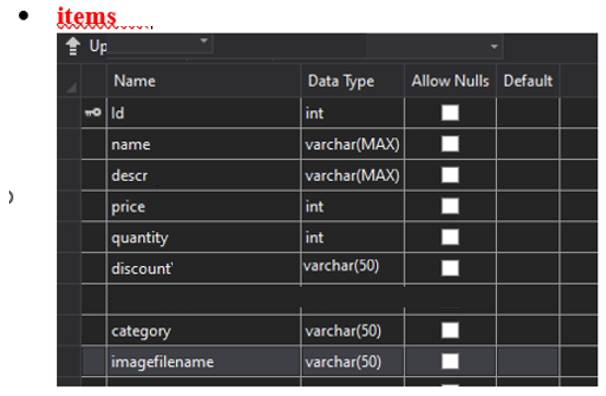
1. **getnameAPI Controller**
   * [**get action**](#apinames) **(returns all users name based on role value)** 

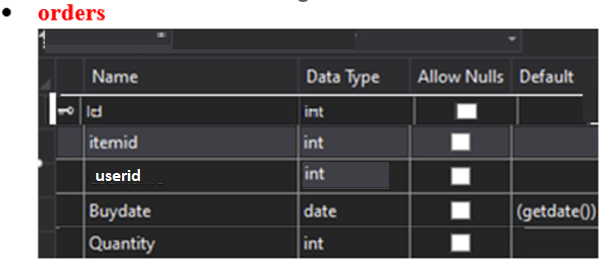
**Project specification and GUI**

1. Create 3 tables (you can change table and attributes names to select proper naming)

(all Ids should be auto increment) usersall items, orders







1. Create 2 Layouts ( layoutMaster , layoutCustomer) to be **included** in all pages
2. Inside the layouts include **NavBar**
   * For the admin **NavBar** (home, Manage Items, Send email, Customer Search, Report Page, dashboard, Add Admin, User Management,Logout button )
   * For the customer **NavBar** (home, Items Slider, My Purchase, items list)
   * Also include the Logo, Website Title, Logout button)
3. **The Project will have 3 Models**

* Usersall
* **Items**
* **Orders**

1. **For these models create 3 controllers Entities (Usersall, Items, Orders)**
2. **Login page** (which will redirect users to maser Home page or customer Home Page)

(Enable Auto Login checkbox)

(at **Usersall** controller – login)

* use Null layout and use CSS class at the head section

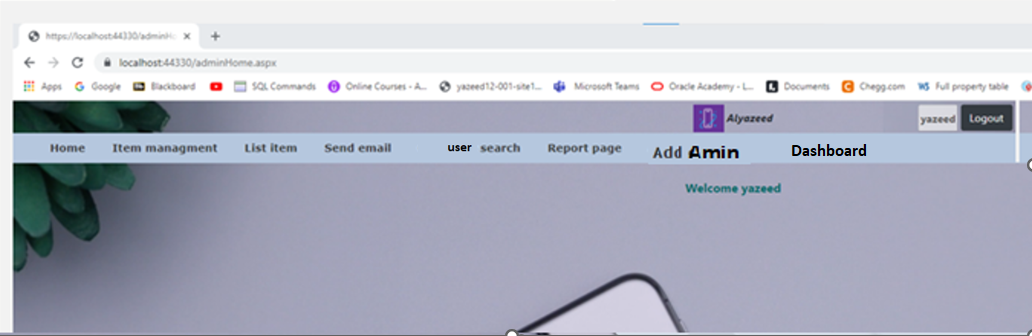
(Use div, table, and background color, image for proper design as shown)

Graphical user interface, text, application, chat or text message

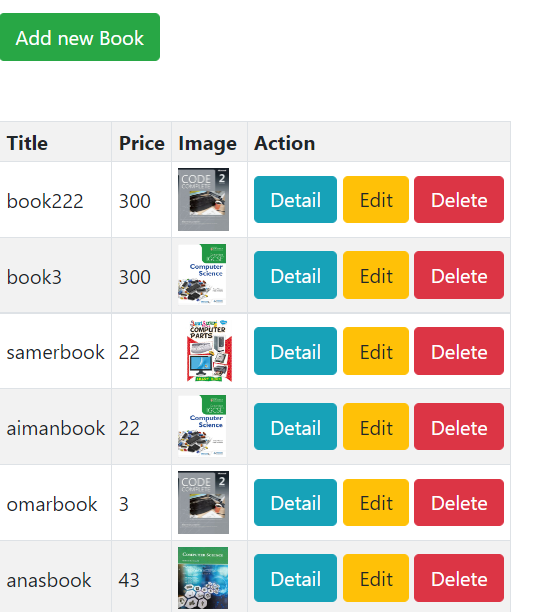
Description automatically generated

**For admin implement these pages** (check for correct role session variable used at view to redirect unauthorized users to login page)

1. AdminHome Page (at **Usersall** controller - admin\_home )



1. Item manage page: make sure to add the picture (at item control – index) Also Make sure to use partial view with modal effect for at least (Add, Detail, Edit, Delete)



1. **List items page: item table ordered by Category**
   * **(at item controller** list action **)**

**Text

Description automatically generated with low confidence**

1. **email page**

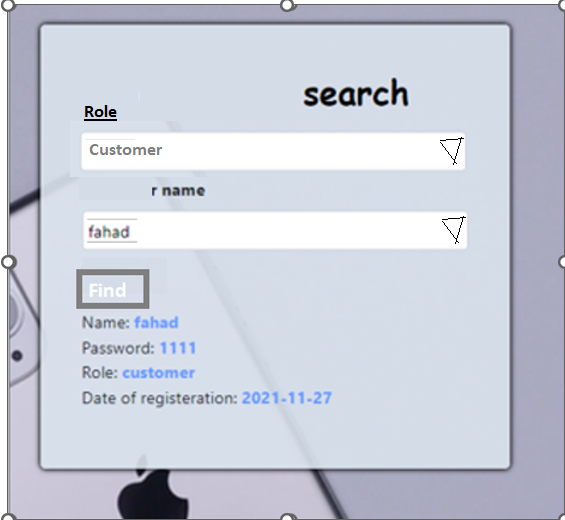
**(at Usersall controller** email action **)**

Graphical user interface, text, application, chat or text message

Description automatically generated

1. **User search page to search a users based on his role then his name and then shows his detail (at usersall controller** customer\_search action **)**

* Search Role List
* Search Name List
* when click the find button it will show user name, id , etc......
* use cascading lists for this search and also use API for getting names based on selected role at the role list



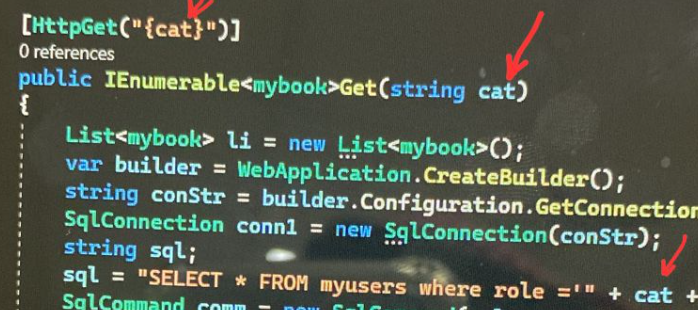
<select class="form-control" name="cata" id="ca" onchange= "**gettitle()">**

<**option** **value**="0">Please select</**option**>

<**option** **value**="admin">Admin</**option**>

<**option** **value**="customer">Customer</**option**>

</select>

****

1. **User management page: show all users index page with all management functions(detail, delete, update)**
2. **Purchase report page: that will show all customers total purchase amount of in a** **as shown (at order controller - purchase\_report action)**

* First Need to define a report class at the model

{

public int Id { get; set; }

public string customername { get; set; }

public int total { get; set; }

}

* Check the Data Folder (at ApplicationContext.cs file) **that** report context is added:
  + public DbSet<WebApplicationxx.Models.report > itemsall { get; set; }
* add this entity code at the action

var orItems = await \_context.report.FromSqlRaw(“select

usersall.id as Id,

usersall**.name** as customername,

sum (quantity \* price)  as total

from itemsall, orders , usersall

where itemid= itemsall.Id ,and custid= usersall.Id

group by usersall.id, usersall**.name** “).ToListAsync();

* make sure that the orders report is **order by date**



* add the Detail link (sending customer id)

<a asp-action="ordersdetail" asp-route-id="@item.Id ">Details</a>

When click at View that customer **orders detail** will be shown at customer **order**

1. **detail page**

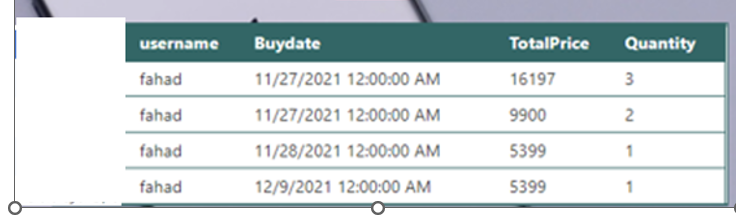
* Add orderdetail class(id, username, buydate, totalprice, quantity) at the model folder
* Check the Data Folder (at ApplicationContext.cs file) **that the** orderdetail context is added:
  + public DbSet<WebApplicationxx.Models. orderdetail> itemsall { get; set; }
* (at order controller – orderdetail action)

public async Task<IActionResult> ordersdetail (int? idd)

var orItems = await \_context.details.FromSqlRaw("select usersalls.id, usersalls.name as username, orders.Buydate as BuyDate, items.price\*orders.Quantity as totalprice ,orders.Quantity as quantity from orders, usersalls,items where userid= '" + idd + "' and usersalls.Id = order.userid and orders.itemid = items.id and ")

.ToListAsync();

return View(orItems);



1. **Add admin Page** (at usersall controller – add admin action)

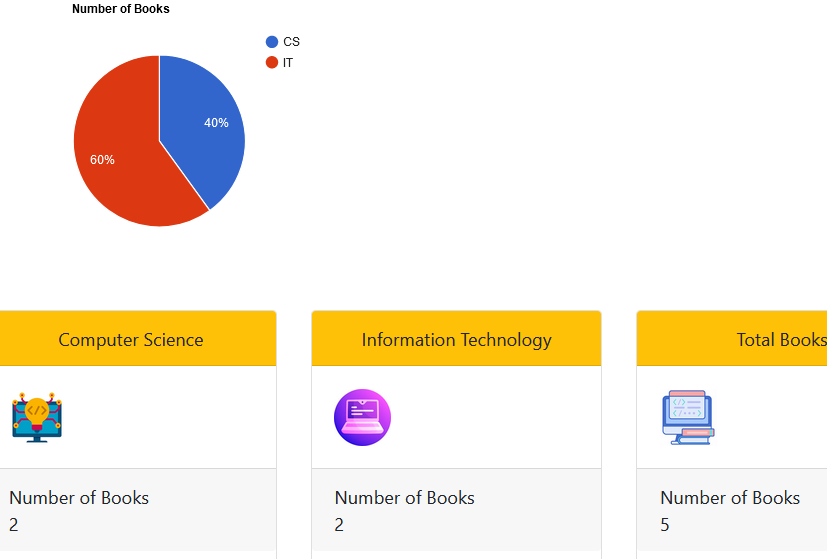
Graphical user interface, application

Description automatically generated

1. Dashboard page:

(item controller - dashboard action )

* **One pie chart showing the categories item counts**
* **Card showing total items, card for sum quantity from item orders table**

****

1. **customerHome page** ( welcome: customer name + description , only Discounted Items , Footer: contact detail: email,, Tel ,etc)

(at **Usersall** controller- customer\_home )

Graphical user interface, application

Description automatically generated



1. When the customer click the **more** it shows the item info at the product **item Detail page**

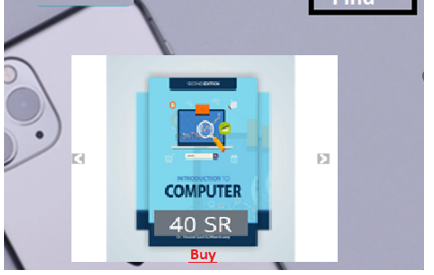
**(****at item controller -** Details action (default)**)**

A screenshot of a computer

Description automatically generated with medium confidence

1. Item image slider (**Carousel**) page:

(item controller - image\_slider action )

****

1. When customer click on the item name it will redirect to the **Buy page**

(at order controller – buy action)

**Graphical user interface, application

Description automatically generated**

****

* + Use a java script code When the quantity is inserted the input onchange event
    - get the price and Quantity from the inputs and then a div message that shows how much you need to pay (price \* quantity \* discount)
  + only if the item has a discount, then subtract 10% from the price

………

1. when buy button is clicked make sure that the quantity is more than the quantity in store otherwise show an error message
2. if there is enough quantity then the buy record should be inserted in the **order table and** the quantity – input text value and also show a successful message
3. **create page** for Customer (use required field and compare password validation)

**(****at** **user controller – create action)**

Graphical user interface, text, application, chat or text message

Description automatically generated

* Make sure that the user name does not exist in the database. If ok then register method else show an error message
* If sign up correctly then direct to customer home page and save customer name, Id, Role at sessions

1. **My Purchase page** that shows: all bought items title, price

**(at** **order controller -** my\_purchase action**)**

Table

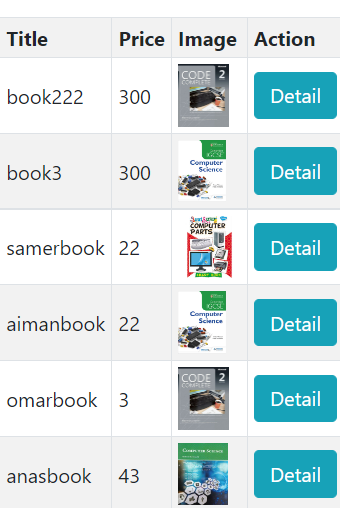
Description automatically generated

int userid = Convert.ToInt32(HttpContext.Session.GetString("userid"));

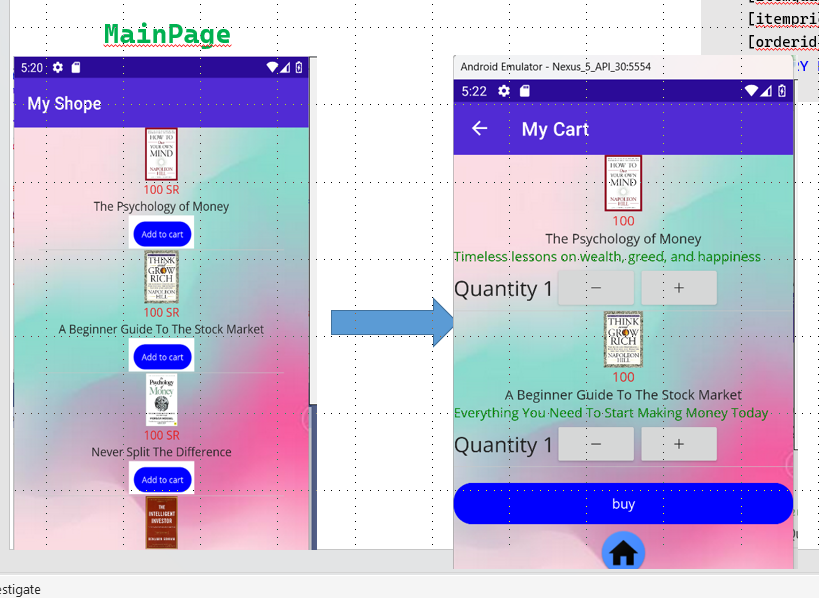
var orItems = await \_context.orders.FromSqlRaw("select \*  from orders where  userid = '" + userid + "'  ").ToListAsync();

item index page:

1. -(item controller -index action)



1. **Shopping Cart Mobile App (Bonus 2 marks)**

****