

## Assignment # 3

### **Deliverables for assignment three:**

In this assignment, you are requested to implement the following components for the project you created in assignment one and two:

- Controllers
- Models
- Modify the views in order to take care of session variables (log in, maintaining session and log out).

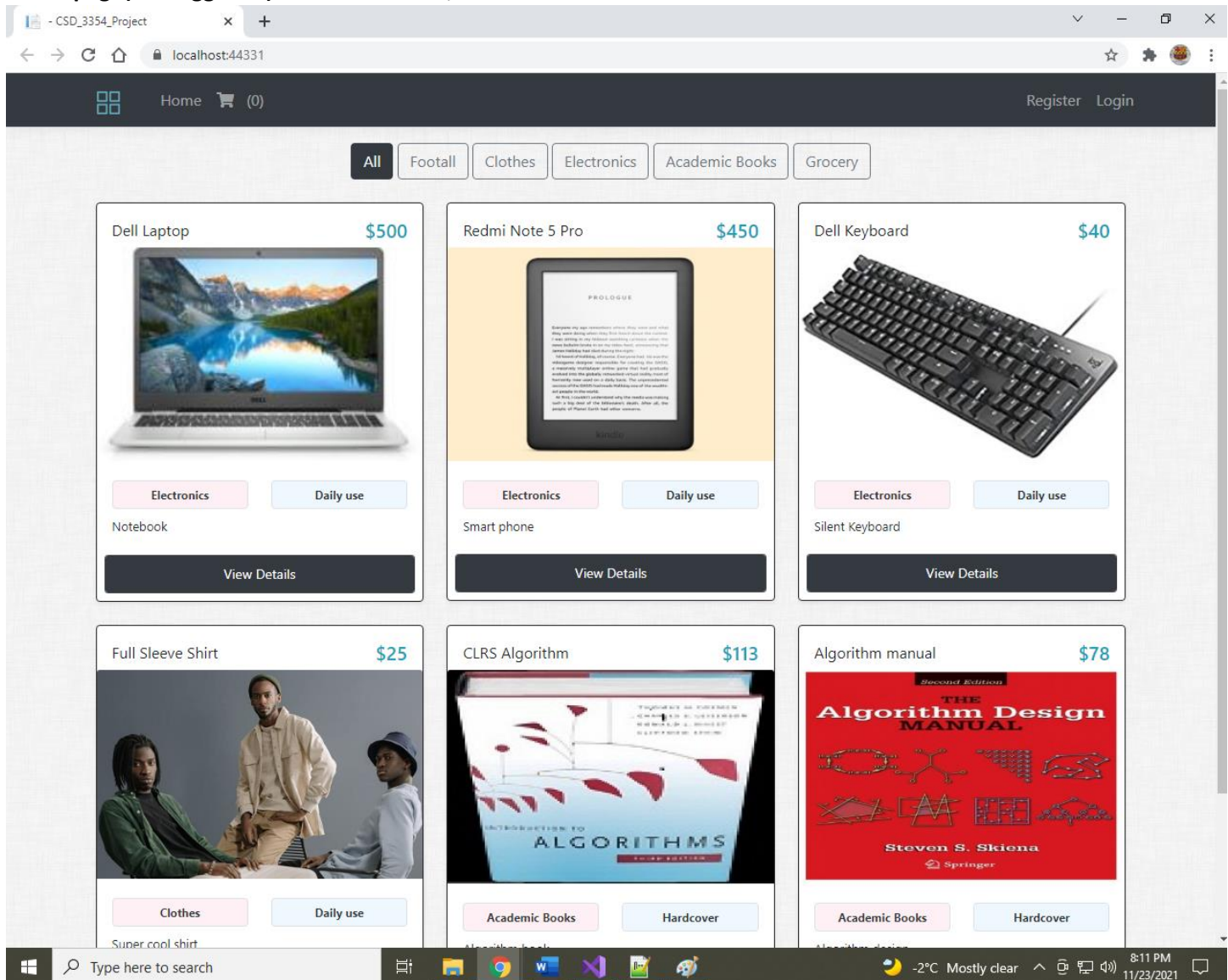
### **To submit:**

Create a document with the screen shot of all the view pages output in browser; for each view describe how the view is generated based on the action and model. (if your files size is too big convert it to pdf that will shrink it down)

Copy your Model, View and Controller folders into one single folder. Copy the documentation in the same folder. Zip the folder and submit through Moodle.

Documentation starts here:

Home page(not logged in) : Action: Index , Controller: Home



Explanation:

When we launch this web application, index view of the home controller gets executed.

\_Layout.cshtml is the wrapper cshtml file that gets called for every view in the project/solution, and almost every other meaningful view uses \_Layout.cshtml as their page for common styles and structures.

As you can tell from the screenshot, this page has categories at the top and below that are the products with their image displayed.

To display the categories and products, I have used view model called **HomeVM** to store the list of categories and products.

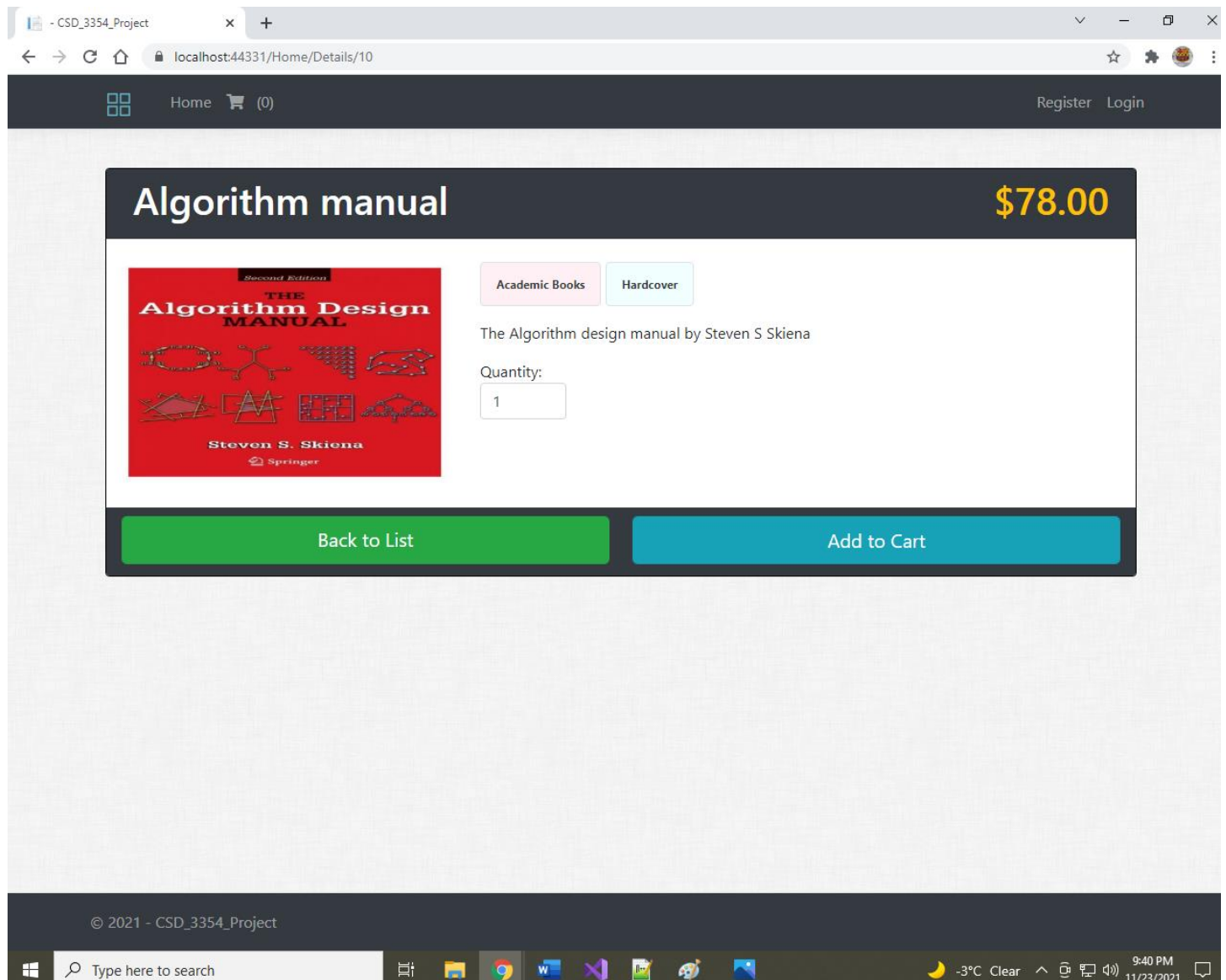
This **HomeVM** is passed to the Index view where both the Categories and Products list are iterated through for displaying the data using foreach loop.

I use GetAll() of an interface IRepository to fetch the data from the database.

We can pass filter conditions, order by clause, join table/object names etc to generic GetAll function.

Each product card on the page has a View Details button.

Clicking this button redirects to Details page.



View model called DetailsVM with two properties: Product and ExistsnCart is used here.

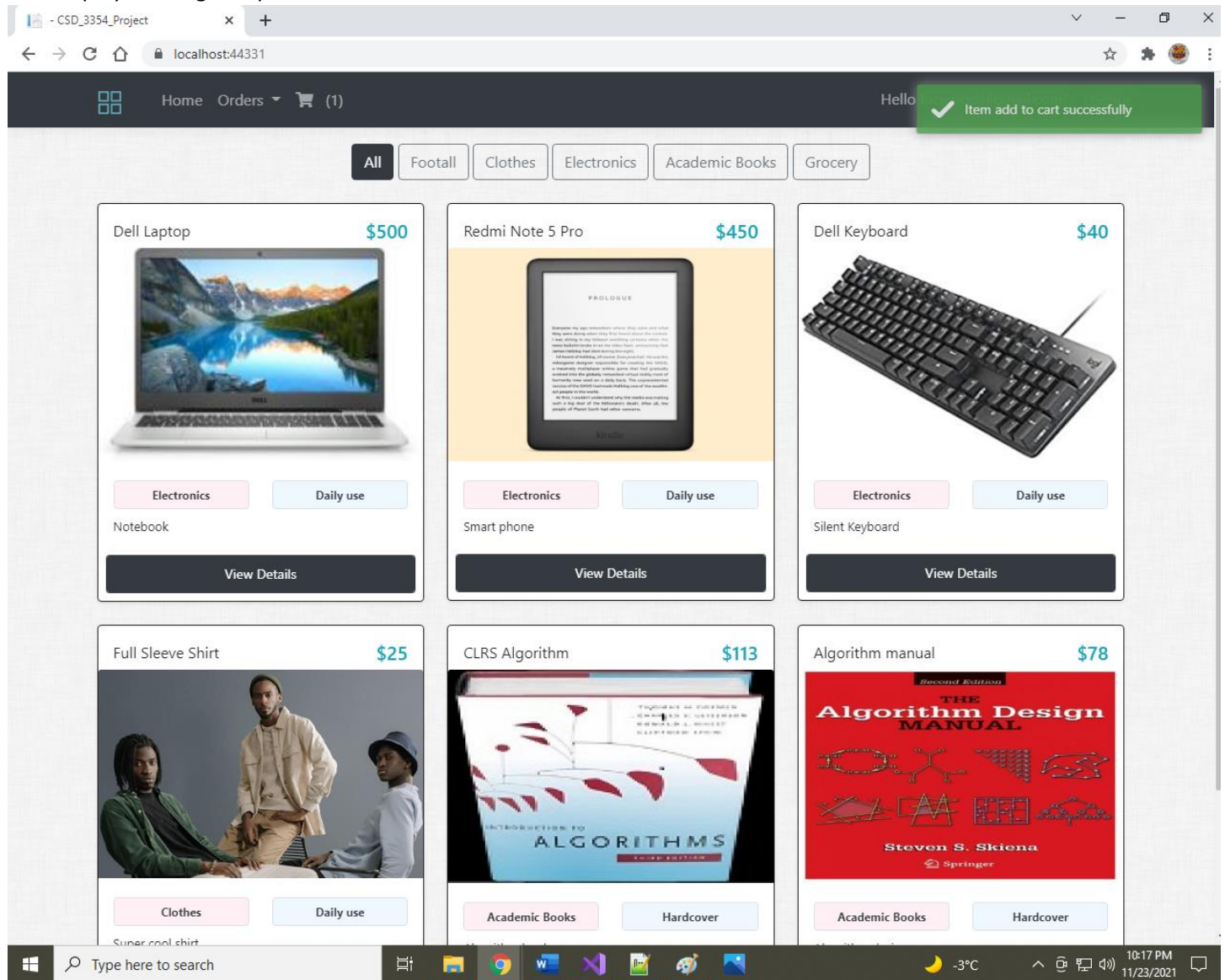
First all the data(product records) is loaded into DetailsVM from the database and ExistsnCart is set to false.

If we are logged in and we already have this product in our cart then the product details will be present in our session as per our logic. Using the session data, we set the ExistsnCart of only matching product in DetailsVM to true for displaying either RemoveFromCart or AddToCart button.

If it is already added to our cart then we display a remove from cart button to remove the product else Add to Cart button is displayed.

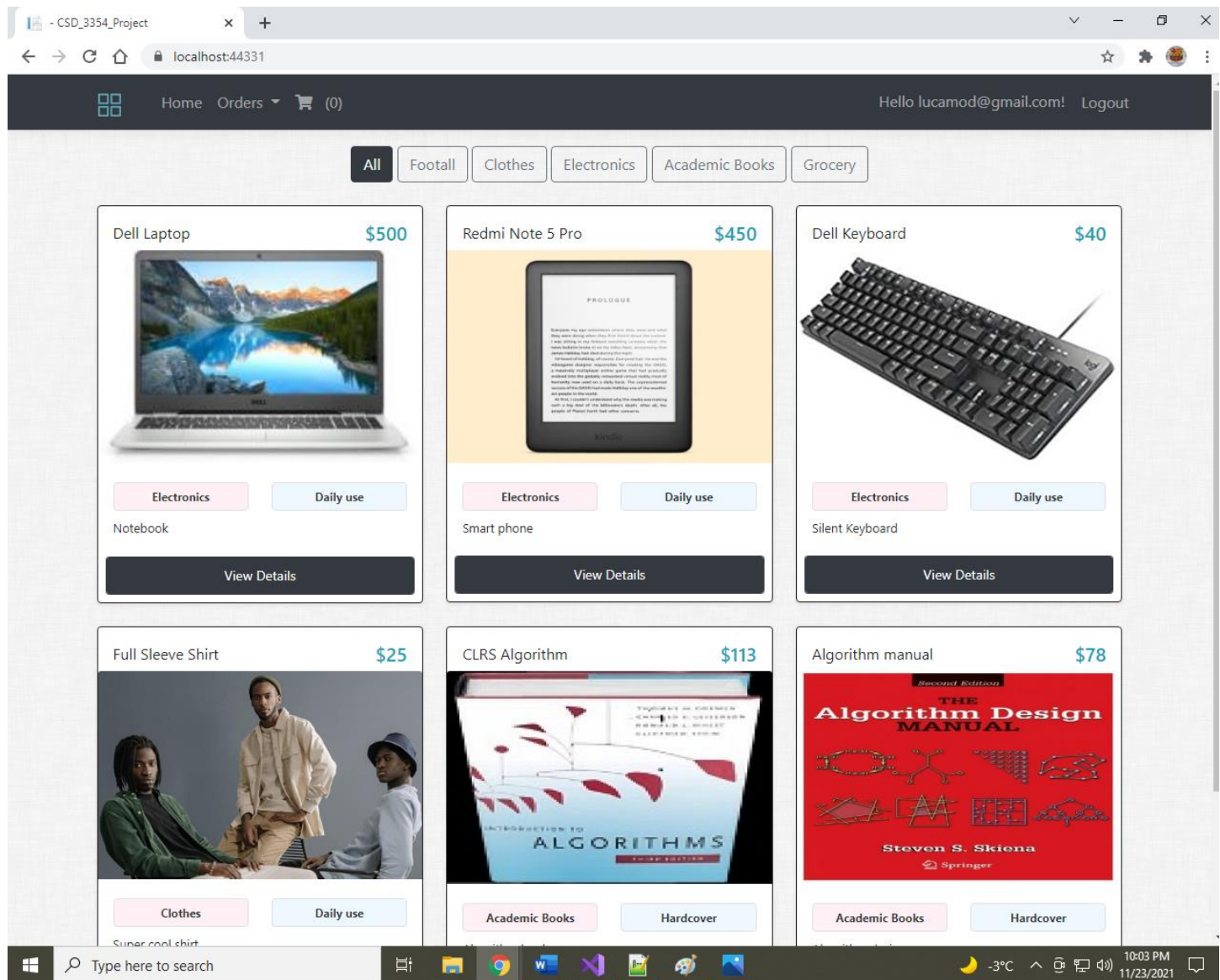
When you click on Add to Cart button for a product, then the product gets added to the Session and at the top after the cart symbol you can see the change in the number of items in the cart. Notification is

also displayed using TempData.



This page also has a BackToList button to go back to the index page.

## Home Page(Logged In): Action: Index, Controller: Home



When someone is logged in, then at the top right you can see a “Hello Username”.

We have a partial view called `_LoginPartial` which takes care of changing the username as per the logged in user.

We include this partial view in the `_Layout` file so that it works for every view.

We have used a built-in functionality using `Microsoft.AspNetCore.Identity` package, `AddEntityFrameworkStores` service for admin, customer users registration and login.


Rest everything is same as the home page when no one is logged in.



## Cart Page: Action:Index, Controller: Cart

Shopping Cart

Clear Cart Continue Shopping

Product Details	Price per unit	Unit	Total Price
 CLRS Algorithm Algorithm book	\$113/unit	1	\$113

Order Total : \$113.00

Update Cart Continue

© 2021 - CSD\_3354\_Project

In this page, we are showing a list of products present in the cart using session.

You can see two buttons here: one for updating and the other for proceeding ahead.

Update button can be used to delete or update the quantity of a product.

Clicking Continue button takes us to Summary page as follows.

Summary page: Action: Summary, Controller: Cart

Summary page: Action: Summary, Controller: Cart

Browser: CSD\_3354\_Project x +

Address: localhost:44331/Cart/Summary

Header: Home Orders (1) Hello lucamod@gmail.com! Logout

Order Summary [Back to Cart](#)

**User Details:**

Name: Luca Modric

Phone: 33333334344

Email: lucamod@gmail.com

Address: Jackson Street

City: Menlo Park

State: California

Postal Code: M1B1K9

**Summary:**

CLRS Algorithm \$113  
Unit: 1

Total (CAD) \$113

Pay with card

VISA Mastercard AMEX JCB DISCOVER

Card Number: 4111 1111 1111 1111 VISA

Expiration Date (MM/YY): 04 / 22

[Submit Order](#)

© 2021 - CSD\_3354\_Project

Windows taskbar: Type here to search, -3°C Clear, 10:30 PM 11/23/2021

This page contains the summary details of the order we are placing.

Hitting the submit order places the order. In other words, we are trying to connect to the payment gateway and complete the transaction and update our database.

However, I am facing a couple of technical issues as I do not have a domain to complete a fake payment transaction.

I am referring this website for payment: <https://developer.paypal.com/braintree/docs/start/overview>

I will try to fix it and update my code.

Orders page: Action: GetMyOrders, Controller: Order

My Orders

Search filters: Name..., Email..., Phone..., --Order Status--

Search

ID	Name	Email	Phone	Status	Order Date	
1	Cristiano Ronaldo	customer@customer...	4379285366	Pending	11/01/2021	[Details]
2	Cristiano Ronaldo	customer@customer...	4379285366	Cancelled	11/01/2021	[Details]
3	Cristiano Ronaldo	customer@customer...	4379285366	Pending	11/01/2021	[Details]
4	Cristiano Ronaldo	customer@customer...	4379285366	Pending	11/01/2021	[Details]
5	Cristiano Ronaldo	customer@customer...	4379285366	Approved	11/01/2021	[Details]

1 of 1 pages (5 items)

© 2021 - CSD\_3354\_Project

This page shows a list of all the orders placed by the logged in user.

This is done using OrderListVM viewmodel which has 3 properties: OrderHeader list, StatusList and Status.

OrderHeader contains the high level data OrderDetail contains more granular data.

For example: I buy a book and a laptop. So now only one order header record is inserted in the database but two records in OrderDetail table.

Firstly, we are getting all the orders from the database and checking their users against the currently logged in user and displaying the orders of the matching user.

We are using the below code to check the currently logged in user against the users of all the orders.

```
ApplicationUser.FirstOrDefault(u => u.Id ==
(ClaimsIdentity)User.Identity.FindFirst(ClaimTypes.NameIdentifier).Value)
```

Note: ApplicationUser is an IdentityUser

The user can modify(cancel, update for ex: delivery address) all the orders that are not completed yet.

The user can modify or cancel an order by clicking on the details button present at the end of every row.



The screenshot shows a web browser window with the URL `localhost:44331/Order/GetMyOrderDetails/9`. The page has a dark header with a shopping cart icon, navigation links (Home, Orders, (0)), and user information (Hello thirumeni.m07@gmail.com! Logout). The main content area is divided into two sections: 'Order Details' and 'Order Summary'.

**Order Details:**

Name	Ousmane Dembele
Phone	7777777777
Address	sion koliwada
City	Mumbai
State	Maharashtra
Zip Code	456887
Email	thirumeni.m07@gmail.com
Order Date	11/23/2021
Shipping Date	
Transaction ID	ew1v4h7v
Order Status	Processing

[Update Order Details](#)

**Order Summary**

CLRS Algorithm	\$113
Price : 113	
Quantity : 1	
<b>TOTAL</b>	<b>\$113</b>

[Cancel Order](#)

[Back to Orders](#)

© 2021 - CSD\_3354\_Project

We are using OrdeVM viewmodel with two properties OrderHeader and OrderDetail to display the order details like Product Name, Quantity etc.

We are using the orderheader and orderdetail ids to display their data on summary page.

We have cancel button to cancel the order and change the order status to cancelled.

Cancelling the order will cause change in the database table as well.


## Admin page: Action: Index, Controller: Home

Browser: CSD\_3354\_Project | localhost:44331 | Hello admins@admins.com! Logout

Navigation: Home | Stocks | Orders | (0)

Categories: All | Footall | Clothes | Electronics | Academic Books | Grocery

**Dell Laptop** \$500




Electronics | Daily use

Notebook

View Details

**Redmi Note 5 Pro** \$450




Electronics | Daily use

Smart phone

View Details

**Dell Keyboard** \$40




Electronics | Daily use

Silent Keyboard

View Details

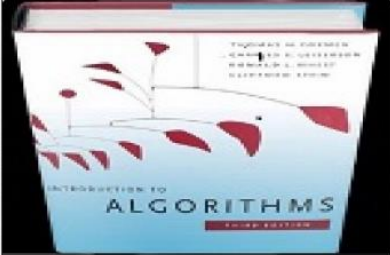
**Full Sleeve Shirt** \$25



Clothes | Daily use

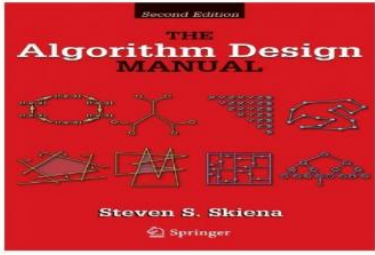
Super cool shirt

**CLRS Algorithm** \$113



Academic Books | Hardcover

**Algorithm manual** \$78

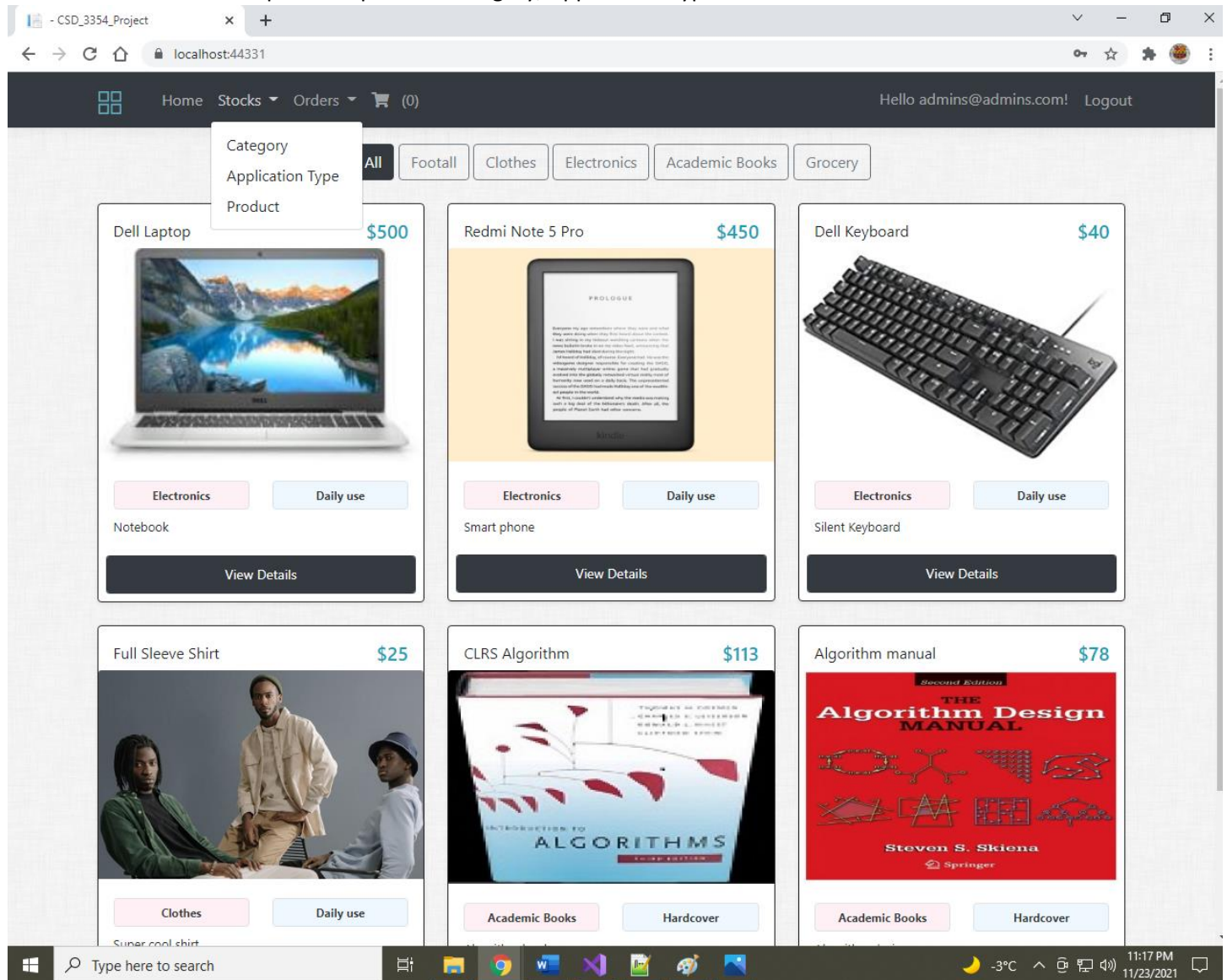


Academic Books | Hardcover

When an admin or manager logs in, then Stock menu becomes enabled and visible at the top.

We have reused the index action of Home Controller by just checking in the \_Layout file if the logged in user is Admin then display "STOCK" option also else do not show it for other users or when no one is logged in.

## Stock menu has three drop down options – Category, Application Type and Product



Admin can perform CRUD operations on Category, Application Type and Product as shown below.

## Category admin page: Action: Index, Controller: Category

The screenshot displays a web application interface for managing categories. The header includes navigation links and a shopping cart icon. The main content area features a table listing categories with their display orders and edit/delete actions. The footer contains copyright information and a Windows taskbar.

Category Name	Display Order	
Footall	1	
Clothes	2	
Electronics	3	
Academic Books	4	
Grocery	5	

For reusability, we have an interface called IRepository for performing adding, reading, removing items from database.

For performing update, we have a separate child class of IRepository because different tables or entity has different fields or attributes.

The list of data shown is retrieved using GetAll method.

I have added comments in Controller files for better understanding and to be honest comments help me remember things when I forget why I wrote a piece of code.

## Category Create page: Action: Create, Controller: Category

The screenshot displays a web application interface for creating a new category. The browser's address bar shows the URL `localhost:44331/Category/Create`. The page features a dark navigation bar at the top with a logo, links for 'Home', 'Stocks', and 'Orders', and a shopping cart icon with '(0)' items. On the right side of the navigation bar, it says 'Hello admins@admins.com!' and provides a 'Logout' link. The main content area is a white box titled 'Add Category' in blue. It contains two text input fields: one for 'Name' and one for 'Display Order'. Below these fields are two buttons: a blue 'Create' button and a green 'Back' button with a left-pointing arrow. The footer of the page is dark and contains the text '© 2021 - CSD\_3354\_Project'. The Windows taskbar at the bottom shows the search bar, task view icon, and several application icons, along with the system clock indicating 11:28 PM on 11/23/2021.

Here we accept two inputs – Name and Display Order.

These inputs are mapped to Category fields using asp-for.

Hitting the create button will save the data on database.



## Category Create page: Action: Edit, Controller: Category

The screenshot displays a web application interface for editing a category. The browser window shows the URL `localhost:44331/Category/Edit/1`. The page header includes a navigation menu with 'Home', 'Stocks', and 'Orders', along with a shopping cart icon showing '(0)'. The user is logged in as 'admins@admins.com!'. The main content area features a form titled 'Edit Category' with two input fields: 'Name' (containing 'Footall') and 'Display Order' (containing '1'). Below the fields are two buttons: 'Update' and 'Back'. The footer of the page indicates the copyright year as 2021 for 'CSD\_3354\_Project'.

Here we accept two inputs for update – Name and Display Order.

These inputs are mapped to Category fields using asp-for.

Hitting the update button will save the data on database using Update method and back button is for going back to the index page of the Category.

## Category Create page: Action: Delete, Controller: Category

The screenshot displays a web application interface for deleting a category. The browser's address bar shows the URL `localhost:44331/Category/Delete/3`. The page header includes a navigation menu with 'Home', 'Stocks', and 'Orders', along with a shopping cart icon showing '(0)' items. A user is logged in as 'admins@admins.com!'. The main content area is titled 'Delete Category' and contains two input fields: 'Name' with the value 'Electronics' and 'Display Order' with the value '3'. Below these fields are two buttons: a red 'Delete' button and a green 'Back' button. The footer shows the copyright notice '© 2021 - CSD\_3354\_Project'.

Here we have two buttons – one for deleting and other going back to the index page.

When we click on delete, the id of that category is passed to the delete post method and from there that data is deleted using remove method.

**We have similar methods for Product and Application Type entities.**









CSD\_3354\_Project x +

localhost:44331/ApplicationType

Home Stocks Orders (0) Hello admins@admins.com! Logout

## Application Type List

[+ Create New Application Type](#)

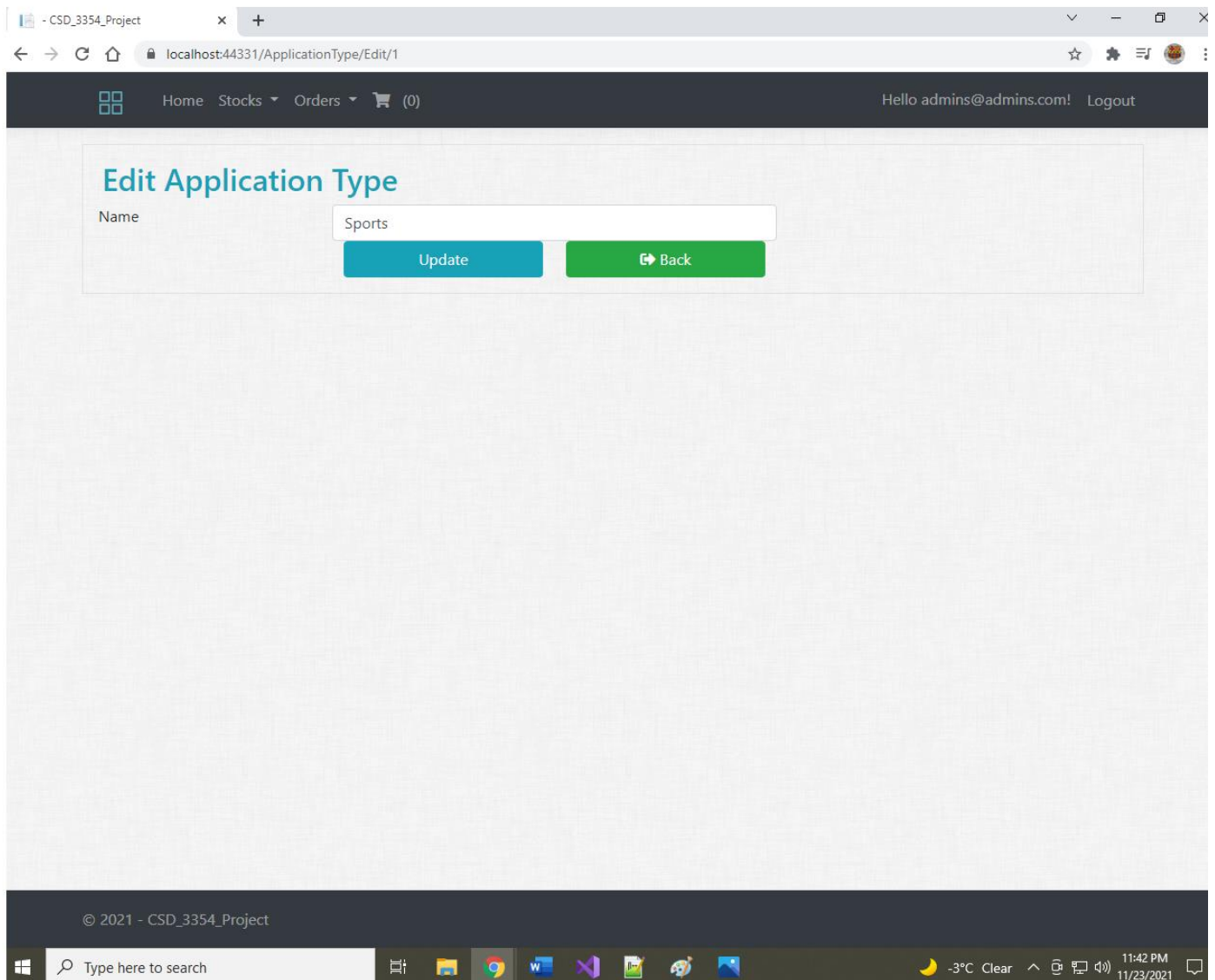
Name		
Sports		
Hardcover		
Daily use		
Junk		

© 2021 - CSD\_3354\_Project

Type here to search

-3°C Clear 11:41 PM 11/23/2021

ApplicationType edit page: Action: Edit, Controller: ApplicationType



## ApplicationType delete page: Action: Delete, Controller: ApplicationType

CSD\_3354\_Project x +

localhost:44331/ApplicationType/Delete/2

Home Stocks Orders (0) Hello admins@admins.com! Logout

### Delete Application Type

Name Hardcover

Delete Back

© 2021 - CSD\_3354\_Project

Type here to search

-3°C Clear 11:43 PM 11/23/2021



## ApplicationType Create page: Action: Create, Controller: ApplicationType

Create - CSD\_3354\_Project x +

localhost:44331/ApplicationType/Create

Home Stocks Orders (0) Hello admins@admins.com! Logout

### Add Application Type

Name

Create Back

© 2021 - CSD\_3354\_Project

Type here to search

-4°C Clear 11:44 PM 11/23/2021

## Product index page: Action: Index, Controller: Product













CSD\_3354\_Project x +

localhost:44331/Product

Home Stocks Orders (0) Hello admins@admins.com! Logout

### Product List

[+ Create New Product](#)

Product Name	Price	Category	Application Type		
Dell Laptop	500	Electronics	Daily use		
Redmi Note 5 Pro	450	Electronics	Daily use		
Dell Keyboard	40	Electronics	Daily use		
Full Sleeve Shirt	25	Clothes	Daily use		
CLRS Algorithm	113	Academic Books	Hardcover		
Algorithm manual	78	Academic Books	Hardcover		

© 2021 - CSD\_3354\_Project

Type here to search

11:45 PM 11/23/2021

Product index page: Action: Upsert, Controller: Product

Note: We have same method for both create and update.

Based on product id we are finding out if it is a new record or update record

CSD\_3354\_Project x +

localhost:44331/Product/Upsert/3

Home Stocks Orders (0) Hello admins@admins.com! Logout

## Edit Product

Name: Dell Laptop

Price: 500

ShortDesc: Notebook


Description: Intel powered laptop for best performance

Image: Choose Files No file chosen

Category Type: Electronics

Application Type: Daily use

Update Back



Windows Taskbar: Type here to search, -4°C Clear, 11:45 PM 11/23/2021

**Product index page: Action: Delete, Controller: Product**

A screenshot of a web browser displaying a 'Delete Product' form. The browser's address bar shows 'localhost:44331/Product/Delete/3'. The page has a dark header with a shopping cart icon and the text 'Hello admins@admins.com! Logout'. The form itself is light gray and contains the following fields: 'Name' with the value 'Dell Laptop', 'Price' with the value '500', and 'Description' with the value 'Intel powered laptop for best performance'. The description field has a rich text editor toolbar above it. To the right of the form is a placeholder image of a laptop. Below the form, there are two buttons: a red 'Delete' button and a green 'Back' button. The footer of the browser window shows '© 2021 - CSD\_3354\_Project'.

The screenshot displays a web application interface for managing orders. At the top, there is a navigation bar with a logo, links for Home, Stocks, Orders, and a shopping cart icon showing 0 items. The user is logged in as 'admins@admins.com!' with a Logout button. Below the navigation bar, the main heading is 'Orders Management'. A search form is located below the heading, featuring input fields for Name, Email, and Phone, a dropdown for Order Status, and a Search button. Below the search form, a table lists the orders. The table has columns for ID, Name, Email, Phone, Status, and Order Date. There are 5 orders listed, all for 'Cristiano Ronaldo' with the email 'customer@customer....' and phone '4379285366'. The statuses are Pending, Cancelled, Pending, Pending, and Approved. The order date for all is 11/01/2021. Each row has a menu icon. At the bottom of the table, there is a pagination control showing '1 of 2 pages (9 items)' and a page number '1' highlighted in a red circle. The footer of the page shows the copyright '© 2021 - CSD\_3354\_Project'.

© 2021 - CSD\_3354\_Project

Here, we are using OrderListVM view model to store list of order headers, list of status and status. The view model is returned to the view where we use javascript/ejs to display the data in a tabular format.



## Admin details page: Action: Details, Controller: Order

The screenshot displays a web application interface for an admin user. The browser address bar shows the URL `localhost:44331/Order/Details/5`. The page header includes navigation links (Home, Stocks, Orders) and a user greeting (Hello admins@admins.com! Logout). The main content area is titled "Order Summary" and contains two sections: "Order Details" and "Order Summary".

**Order Details:**

Name	Cristiano Ronaldo
Phone	4379285366
Address	107 Coxwell Drive
City	NewYork
State	Texas
Zip Code	M1B1K6
Email	customer@customer.com
Order Date	11/1/2021
Shipping Date	
Order Status	Approved

**Order Summary:**

TOTAL	\$1349
-------	--------

Buttons: Start Processing, Cancel Order, Update Order Details

This page shows the order summary of the selected order same as users order summary.

So the functionality is same summary page of normal user.

This page has three buttons.

**StartProcessing:** This button processes the order , meaning update the order record in the database and in the user interface as well.

**cancelOrder:** This button cancels the order and the status column gets changed in the database.

**UpdateOrdeDetails:** We can update the details using this button. This is an update operation.

## Admin Order Details: Action: Details, Controller: Order

Order Summary

Back to Orders

### Order Details:

Name	Cristiano Ronaldo
Phone	4379285366
Address	107 Coxwell Drive
City	NewYork
State	Texas
Zip Code	M1B1K6
Email	customer@customer.com
Order Date	11/1/2021
Shipping Date	
Order Status	Approved

Update Order Details

### Order Summary

TOTAL	\$1349
-------	--------

Start Processing

Cancel Order

© 2021 - CSD\_3354\_Project

### Registration page:

We are using the built in functionality for registration, login and user roles.

AddEntityFrameworkStores creates identity user table for IdentityUser in our sql server database using ApplicationDbContext. After adding this line and app.UseAuthentication() run add-migration and update-database commands to create built-in identity related tables provided by asp.net. Then add endpoints.MapRazorPages(); to endpoints at the bottom here for web pages for login/registration stuff. And add <partial name="\_LoginPartial" /> to \_Layout.cshtml. Note: \_LoginPartial.cshtml gets added automatically once identity related packages are installed, scaffold items are created and commands are run.

We are using Razor pages for login registration using endpoints.MapRazorPages(); //Let asp know we have Razor pages for identityUser login/registration related stuff

