ASSIGNMENT 2 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Business**		
Unit number and title	**Unit 30: Application Development**		
Submission date	2/11/2002	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Mai The Duc	**Student ID**	GCH0907
Class	GCH0907	**Assessor name**	Nguyen Dinh Tran Long

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	[illegible]

Grading grid

P4	P5	P6	M3	M4	M5	D2	D3

❐ Summative Feedback:		❐ Resubmission Feedback:
Grade:	Assessor Signature:	Date:
Internal Verifier's Comments:		
Signature & Date:		

TABLE OF CONTENTS

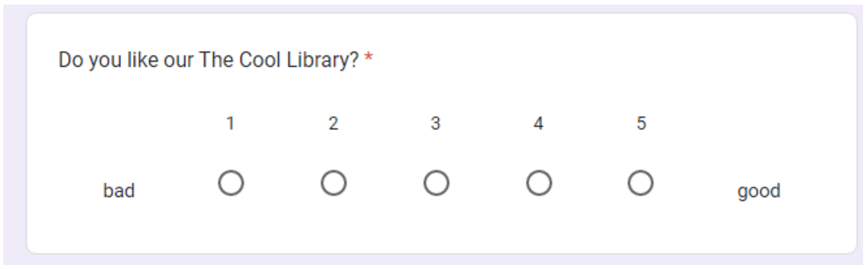
INTRODUCTION

In this part 2 of the project, the author will introduce and explain the system and collect data from survey. Our "The Cool Library" is an internet library, where we sale book for profit. Our team have 3 members, and we work together to develop this website.

I. PEER REVIEW AND FEEDBACK ANALYSIS

A) List of questions

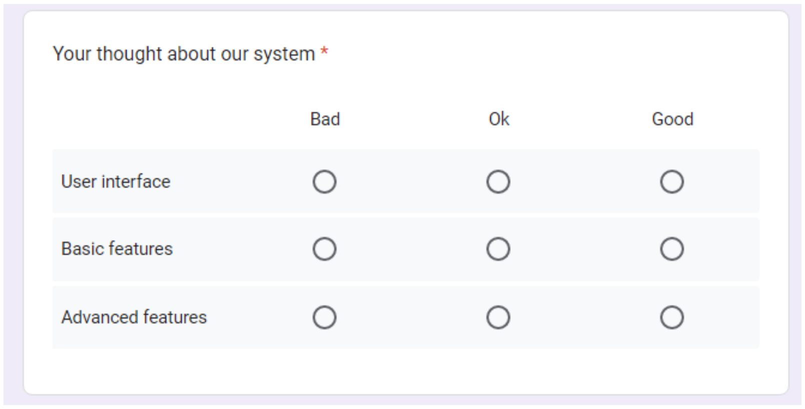
1. Do you like our The Cool Library



Do you like our The Cool Library? *

	1	2	3	4	5	
bad	○	○	○	○	○	good

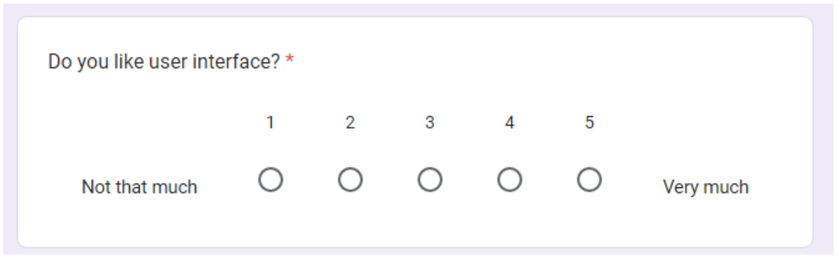
2. Your thought about our system



Your thought about our system *

	Bad	Ok	Good
User interface	○	○	○
Basic features	○	○	○
Advanced features	○	○	○

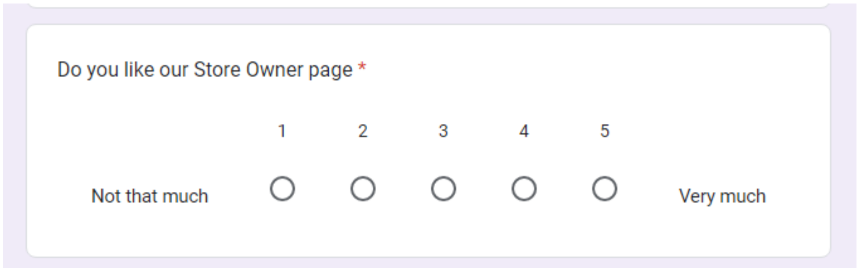
3. Do you like the interface?



Do you like user interface? *

	1	2	3	4	5	
Not that much	○	○	○	○	○	Very much

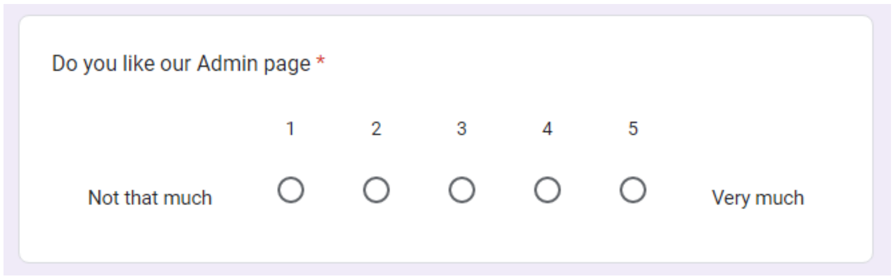
4. Do you like our Store Owner page?



Do you like our Store Owner page *

	1	2	3	4	5	
Not that much	○	○	○	○	○	Very much

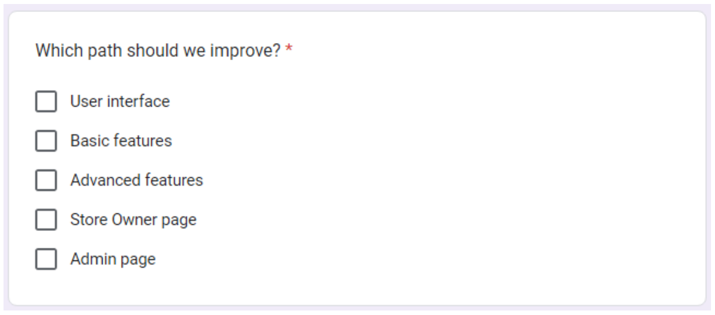
5. Do you like our Admin page?



Do you like our Admin page *

	1	2	3	4	5	
Not that much	○	○	○	○	○	Very much

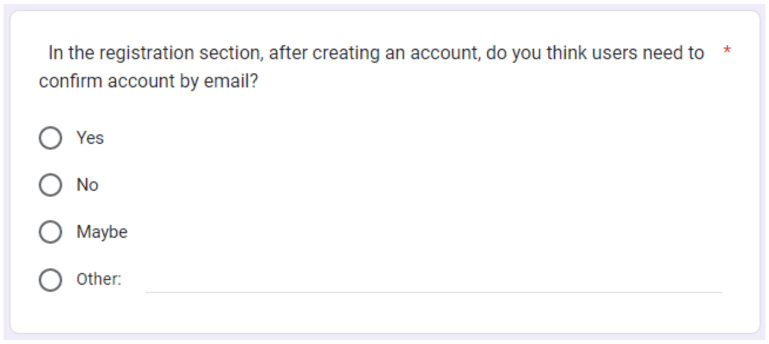
6. Which path should we improve?



Which path should we improve? *

☐ User interface
☐ Basic features
☐ Advanced features
☐ Store Owner page
☐ Admin page

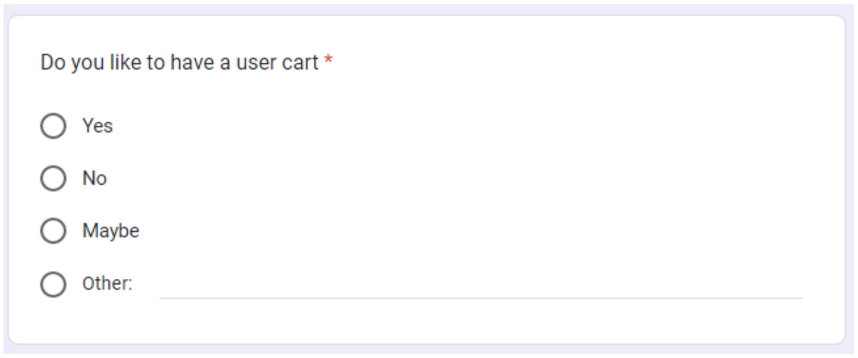
7. In the registration section, after creating an account, do you think users need to confirm account by email?



In the registration section, after creating an account, do you think users need to confirm account by email? *

○ Yes
○ No
○ Maybe
○ Other:

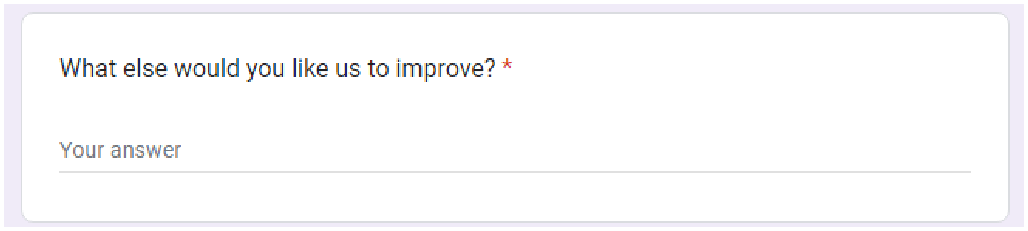
8. Do you like to have a user cart?



Do you like to have a user cart *

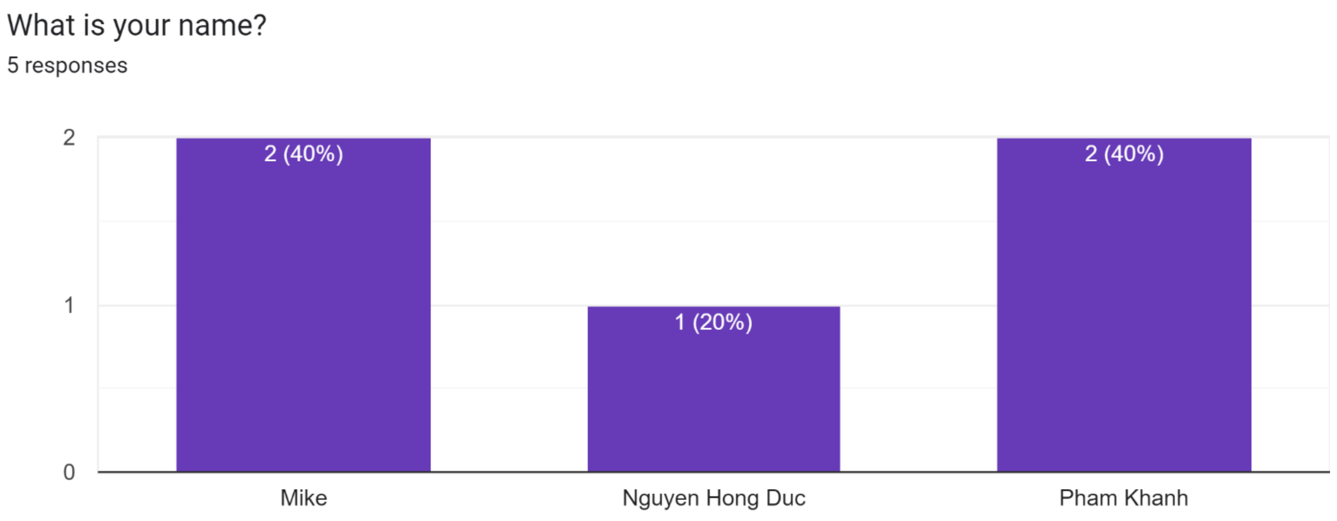
○ Yes
○ No
○ Maybe
○ Other:

9. What else would you like us to improve?

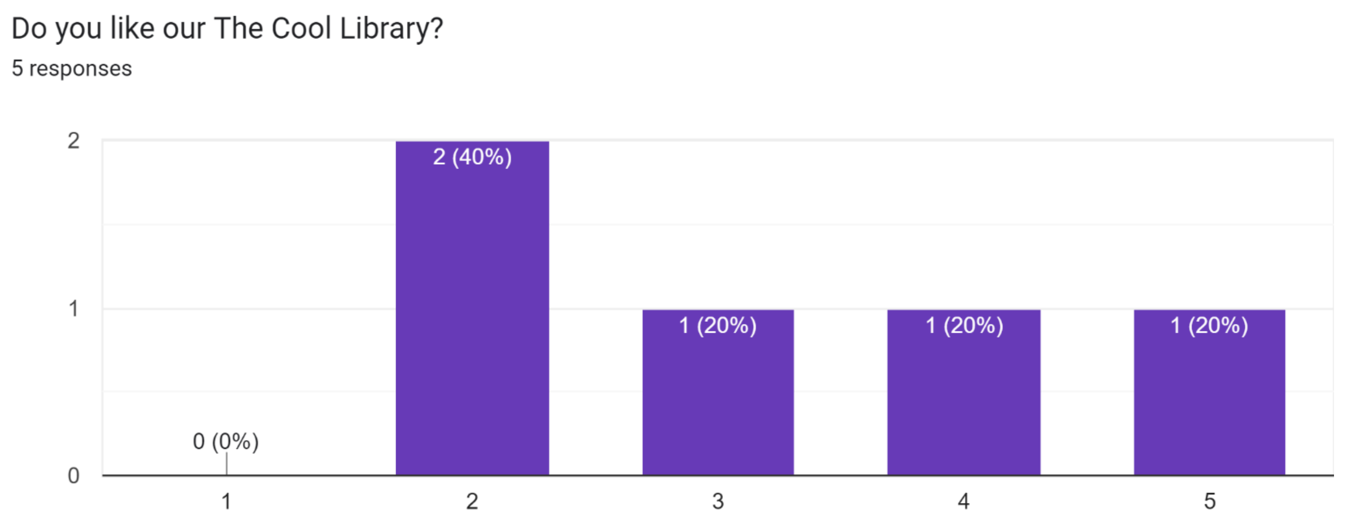

What else would you like us to improve? *

Your answer

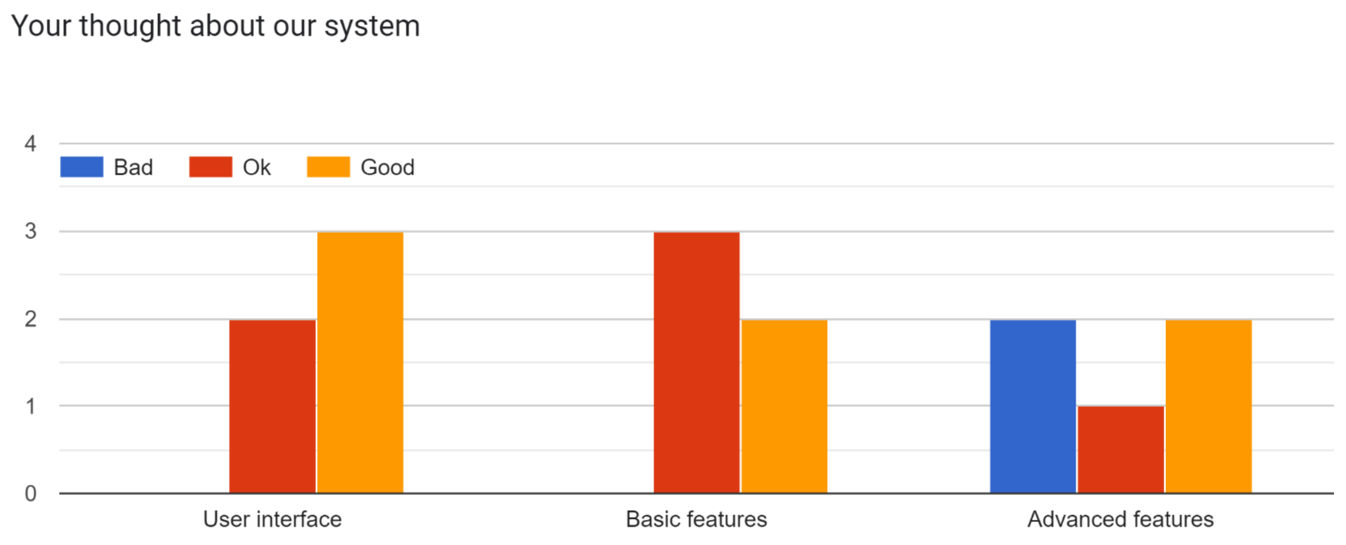
B) Review feedbacks



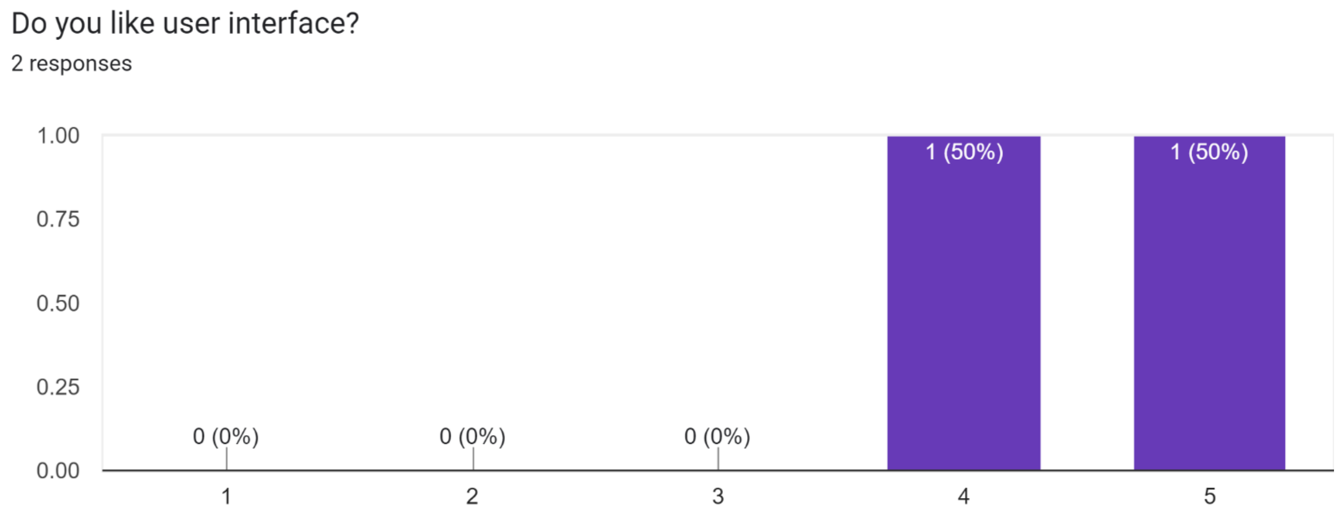
According to the survey this is a section to know who join the survey.



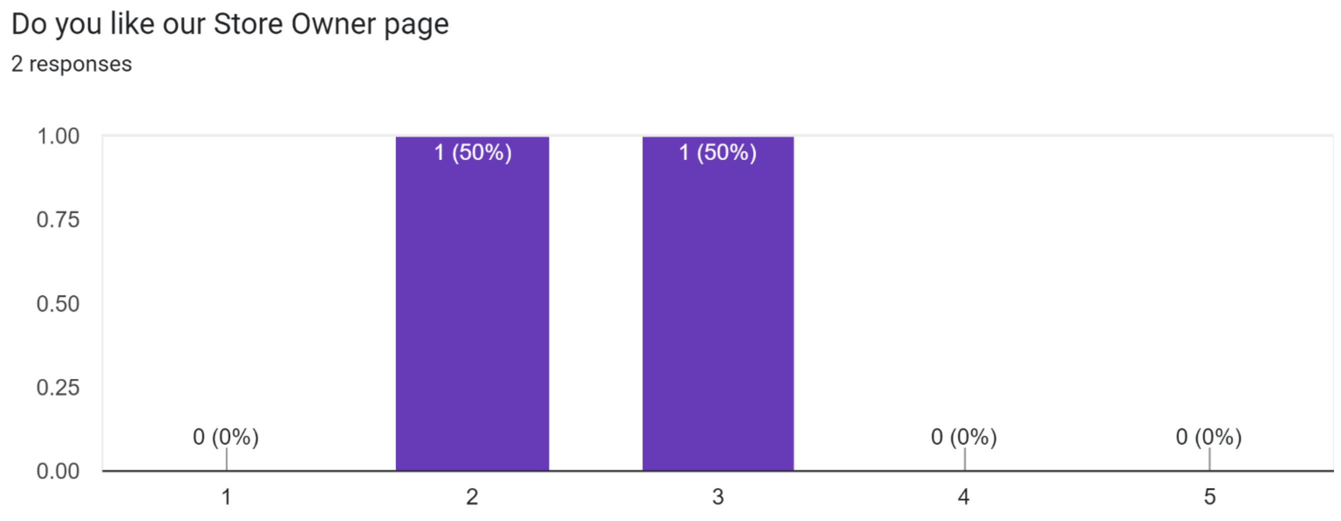
The result show 60% people like our project in the range of 3 – 5, other 40% are in range 2. So it is mean that, our system have satisfied almost everyone.



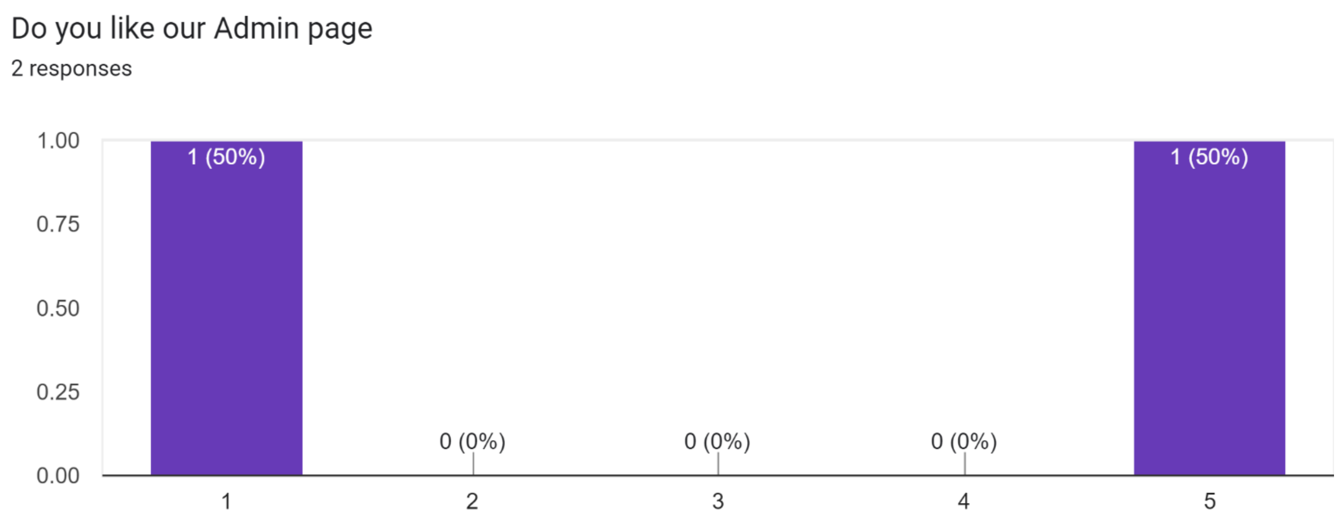
In this question, the answers tell us that the system only have satisfied almost everyone. The result of user interface are 60% good and 40% ok. The basic feature have 40% and 60% ok. Lastly, the advanced have 40% both good and bad, also 20% ok.



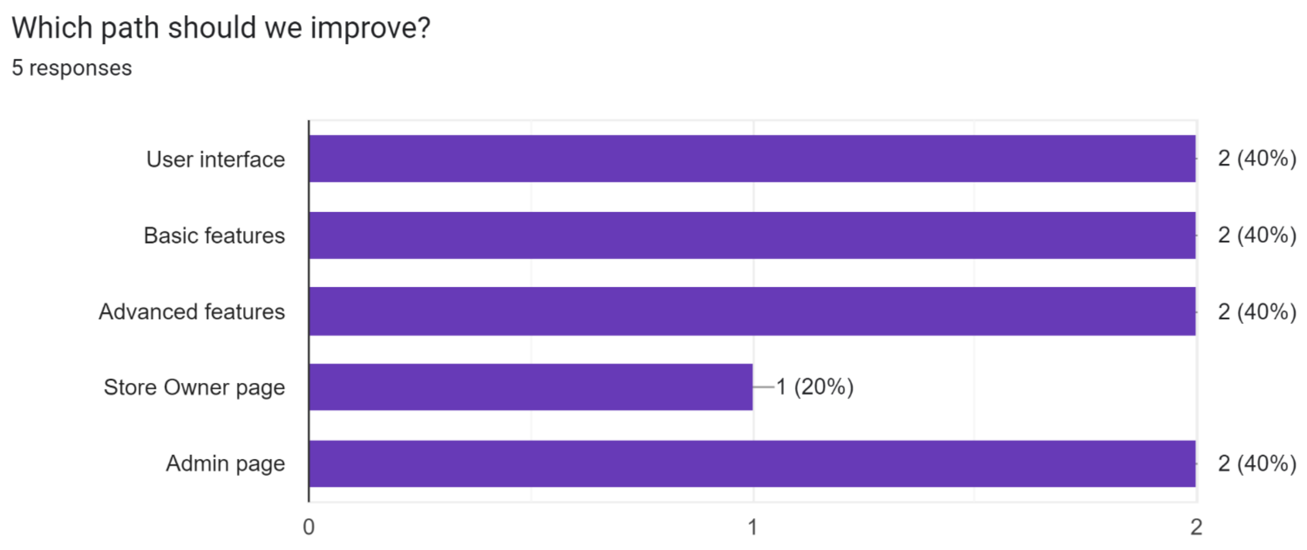
We proudly that our system have good user interface with 50% on both scale 4 and 5. That mean our page is pretty and good-looking.



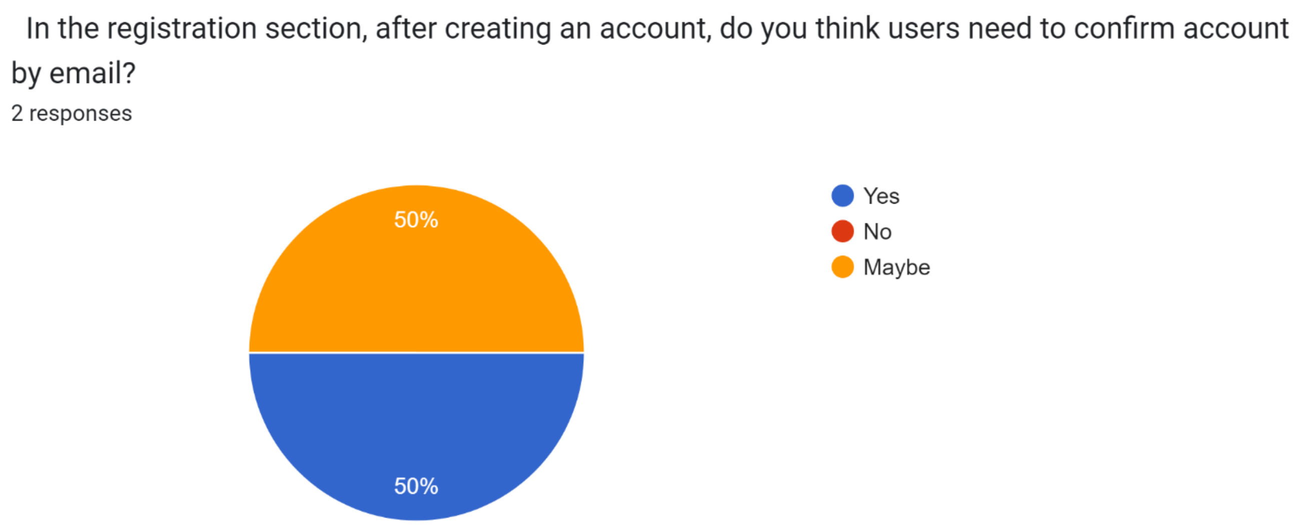
On the other side, Store Owner page have 50% on scale 2 and 3. We should improve more in this Store Owner page.



The Admin page is angrily 50% of save 1 and 5. This is strange so we will keep the design for now.



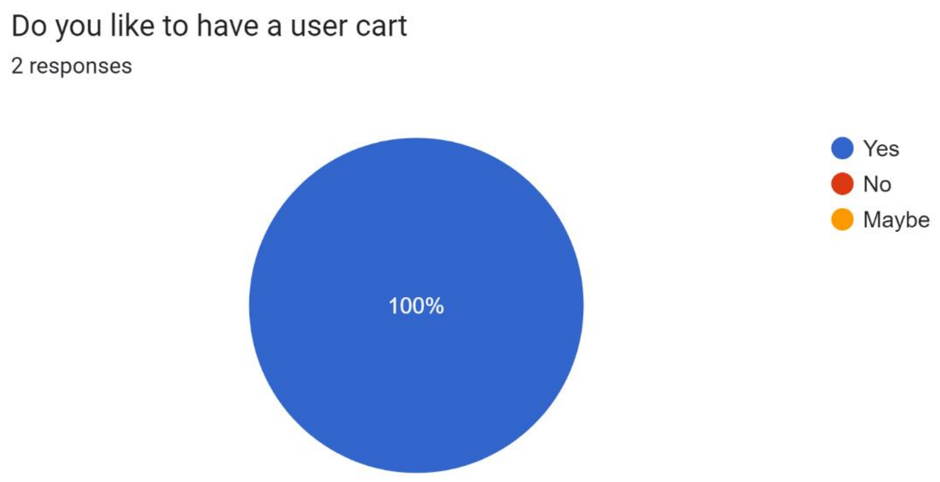
We should improve all the page and feature base on the result of the survey. The Store Owner page maybe can stay at that way for a while.



In the registration section, after creating an account, do you think users need to confirm account by email?

2 responses

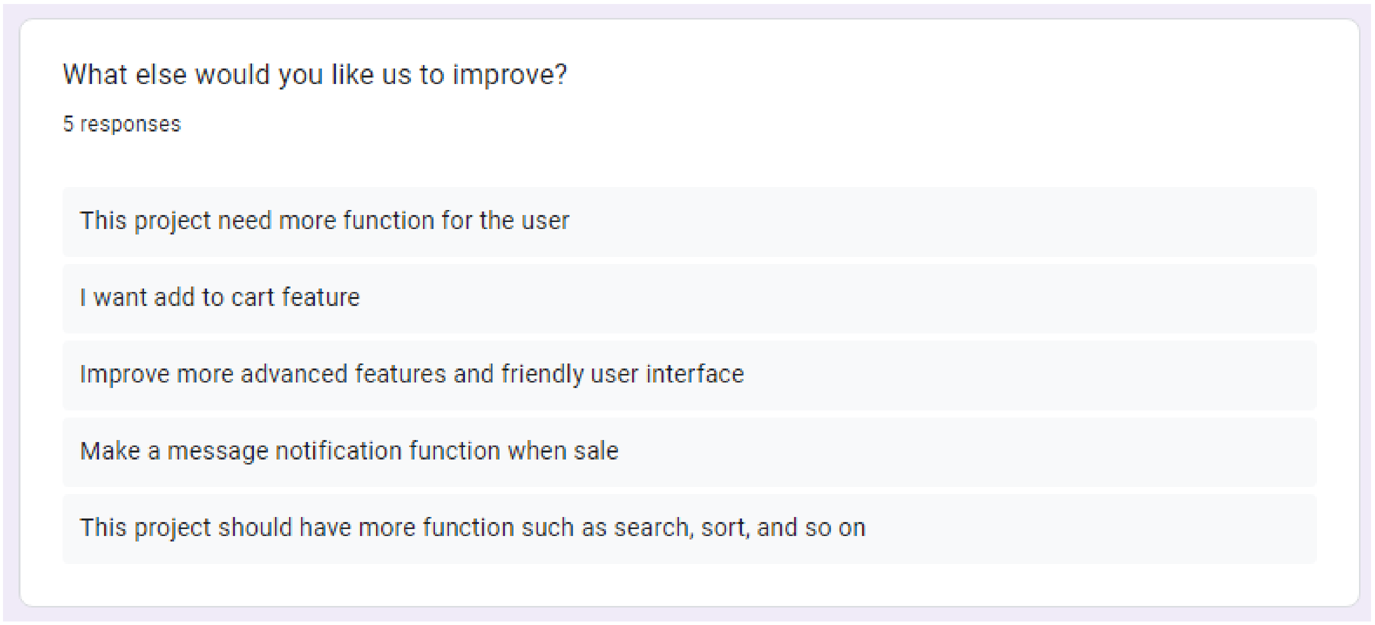
In this question about option step of register, 50% want to add it and 50% confuse and not really want it. We still agreed that maybe add this function is a good idea.



Do you like to have a user cart

2 responses

We aware that 100% user want to have a good user cart. Every system need a cart feature, that is a good point. We plan on update that feature really soon.



What else would you like us to improve?

5 responses

This project need more function for the user

I want add to cart feature

Improve more advanced features and friendly user interface

Make a message notification function when sale

This project should have more function such as search, sort, and so on

Some ideas from the survey result are good like:

- Add to cart feature.
- Message notification.

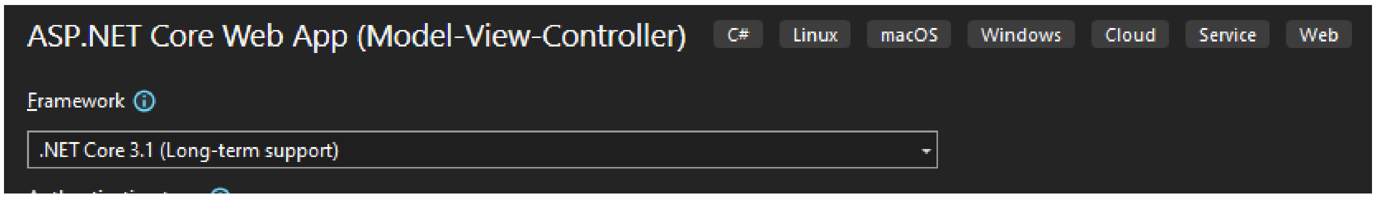
C) Evaluate

Everyone else like our website, and they thought it can be so much more. Almost everyone like to have a cart function. And also, user need more function to do with like a auto message when sale or advanced search & sort.

In my opinion, our system done great and through the survey, we learned a lot about real user experiences. We need to improve like a lot but also can be proud of this project in the same time.

II. APPLICATIONDEVELOPMENT

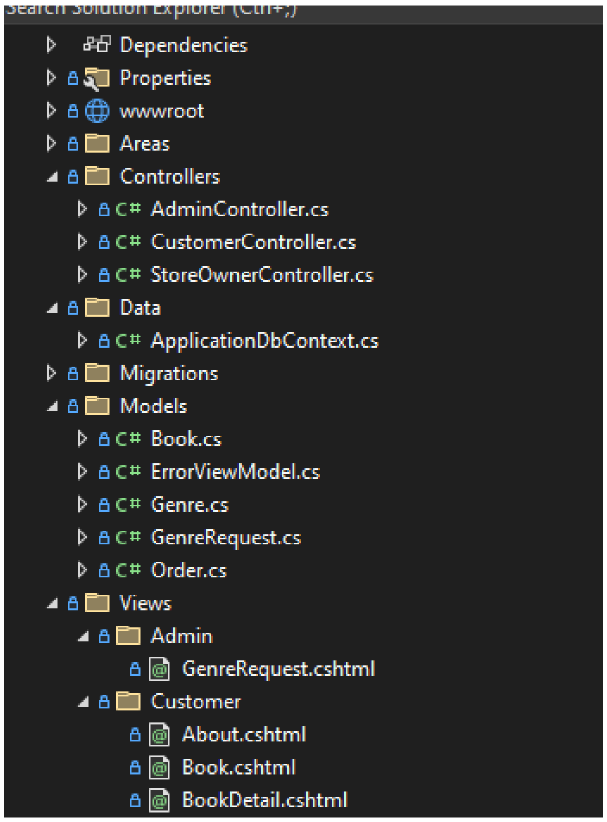
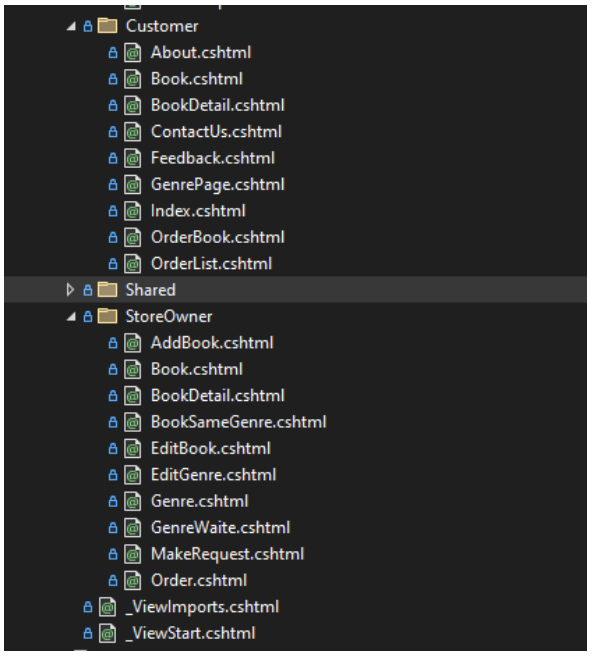
For this project, we using Visual Studio 2022 IDE for coding. And for framework, we use ASP.NET Core 3.1 to develop the web app. This is an open-source framework for developing web application and can be run on Windows, Mac, or Linux. With the help of SQL Server 2019 by Windows, we start develop the app.


ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service Web

Framework

.NET Core 3.1 (Long-term support)

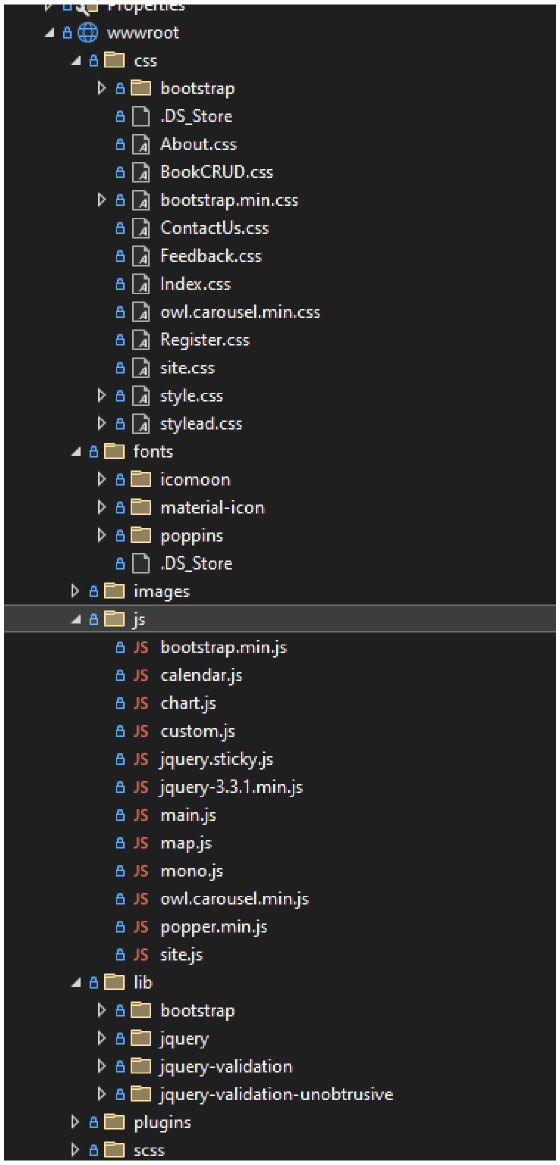
1) Folder structure of the application

```
Dependencies
Properties
wwwroot
Areas
Controllers
    AdminController.cs
    CustomerController.cs
    StoreOwnerController.cs
Data
    ApplicationDbContext.cs
Migrations
Models
    Book.cs
    ErrorViewModel.cs
    Genre.cs
    GenreRequest.cs
    Order.cs
Views
    Admin
        GenreRequest.cshtml
    Customer
        About.cshtml
        Book.cshtml
        BookDetail.cshtml
```

```
Customer
    About.cshtml
    Book.cshtml
    BookDetail.cshtml
    ContactUs.cshtml
    Feedback.cshtml
    GenrePage.cshtml
    Index.cshtml
    OrderBook.cshtml
    OrderList.cshtml
Shared
StoreOwner
    AddBook.cshtml
    Book.cshtml
    BookDetail.cshtml
    BookSameGenre.cshtml
    EditBook.cshtml
    EditGenre.cshtml
    Genre.cshtml
    GenreWaite.cshtml
    MakeRequest.cshtml
    Order.cshtml
_ViewImports.cshtml
_ViewStart.cshtml
```

The folder have a easy understand structure, each controller have a view folder.



```
▷ Properties
◢ wwwroot
  ◢ css
    ▷ bootstrap
      .DS_Store
      About.css
      BookCRUD.css
    ▷ bootstrap.min.css
      ContactUs.css
      Feedback.css
      Index.css
      owl.carousel.min.css
      Register.css
      site.css
    ▷ style.css
    ▷ stylead.css
  ◢ fonts
    ▷ icomoon
    ▷ material-icon
    ▷ poppins
      .DS_Store
  ▷ images
  ◢ js
      JS bootstrap.min.js
      JS calendar.js
      JS chart.js
      JS custom.js
      JS jquery.sticky.js
      JS jquery-3.3.1.min.js
      JS main.js
      JS map.js
      JS mono.js
      JS owl.carousel.min.js
      JS popper.min.js
      JS site.js
  ◢ lib
    ▷ bootstrap
    ▷ jquery
    ▷ jquery-validation
    ▷ jquery-validation-unobtrusive
  ▷ plugins
  ▷ scss
```

Everything else like bootstrap, front, and JavaScript are in ‘wwwroot’ folder.

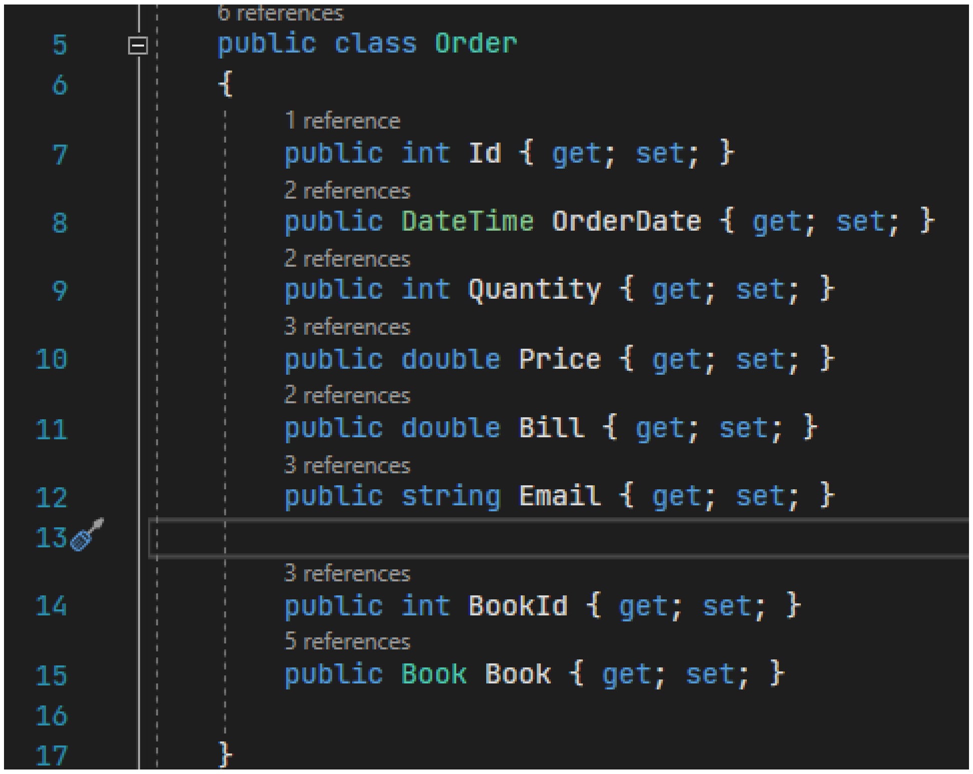
2) Source code samples of the application

Example code: **Order feature**

This feature allow customer to order book with quantity and add it to a order list. In Store Owner page, he or she can see all of the order from difference account. If I am a customer I can only see the order list of my self account.

+) Model:

Order model

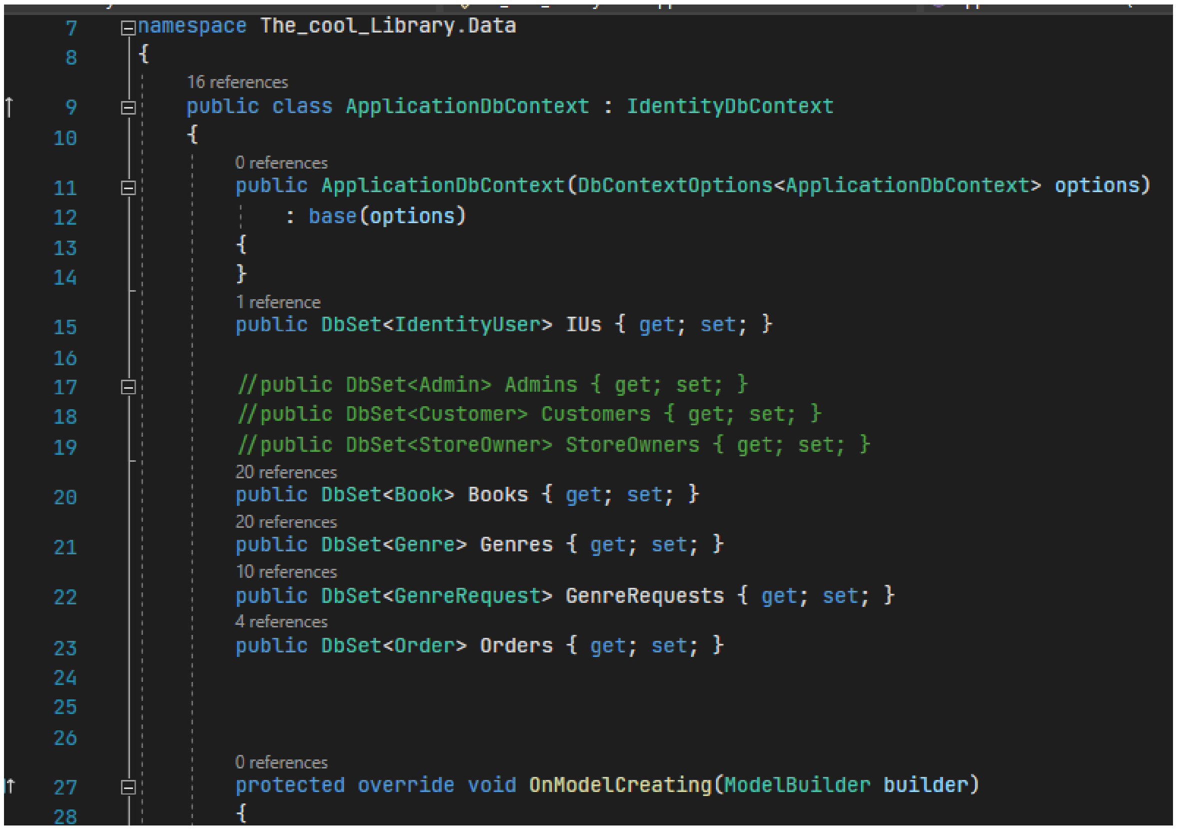


```
6 references
public class Order
{
    1 reference
    public int Id { get; set; }
    2 references
    public DateTime OrderDate { get; set; }
    2 references
    public int Quantity { get; set; }
    3 references
    public double Price { get; set; }
    2 references
    public double Bill { get; set; }
    3 references
    public string Email { get; set; }

    3 references
    public int BookId { get; set; }
    5 references
    public Book Book { get; set; }

}
```

This is model of order table.



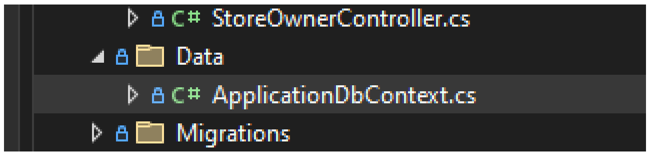
```
namespace The_cool_Library.Data
{
    16 references
    public class ApplicationDbContext : IdentityDbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
        1 reference
        public DbSet<IdentityUser> IUs { get; set; }

        //public DbSet<Admin> Admins { get; set; }
        //public DbSet<Customer> Customers { get; set; }
        //public DbSet<StoreOwner> StoreOwners { get; set; }
        20 references
        public DbSet<Book> Books { get; set; }
        20 references
        public DbSet<Genre> Genres { get; set; }
        10 references
        public DbSet<GenreRequest> GenreRequests { get; set; }
        4 references
        public DbSet<Order> Orders { get; set; }



        0 references
        protected override void OnModelCreating(ModelBuilder builder)
        {
```

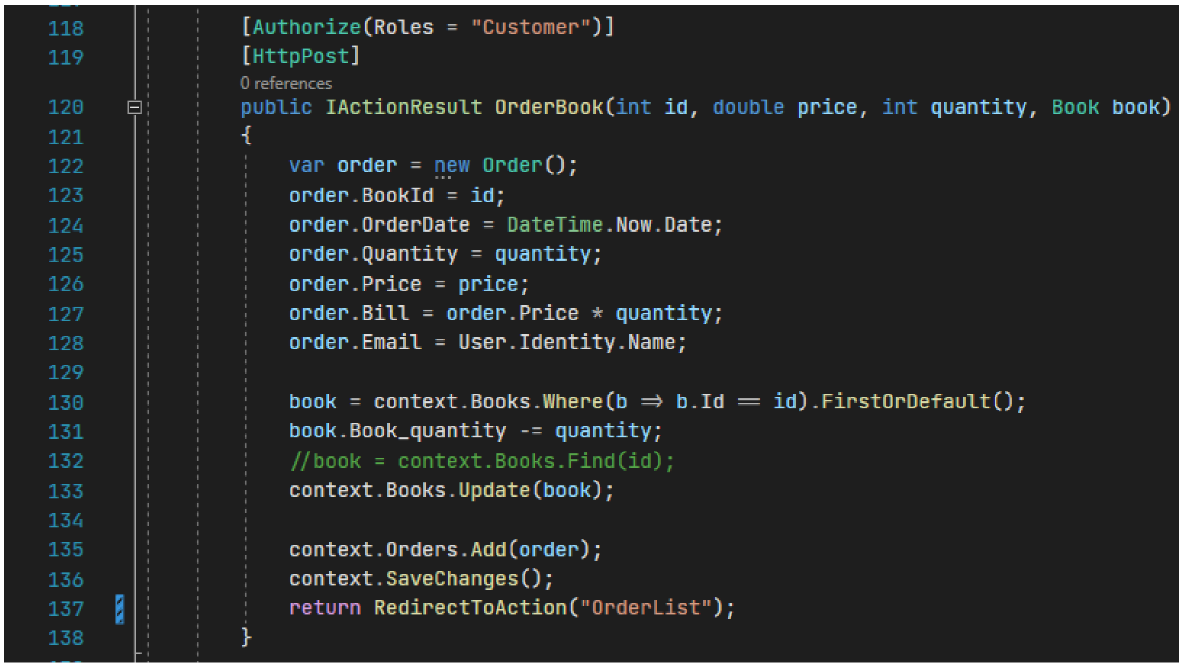
By calling the model in ApplicationDBContext we can create a table in the database.



```
StoreOwnerController.cs
Data
    ApplicationDbContext.cs
Migrations
```

+) Customer's Controller:

Action OrderBook



```
118         [Authorize(Roles = "Customer")]
119         [HttpPost]
            0 references
120         public IActionResult OrderBook(int id, double price, int quantity, Book book)
121         {
122             var order = new Order();
123             order.BookId = id;
124             order.OrderDate = DateTime.Now.Date;
125             order.Quantity = quantity;
126             order.Price = price;
127             order.Bill = order.Price * quantity;
128             order.Email = User.Identity.Name;
129 
130             book = context.Books.Where(b => b.Id == id).FirstOrDefault();
131             book.Book_quantity -= quantity;
132             //book = context.Books.Find(id);
133             context.Books.Update(book);
134 
135             context.Orders.Add(order);
136             context.SaveChanges();
137             return RedirectToAction("OrderList");
138         }
```

This action allow customer to order book that he or she want then redirect to order list page.

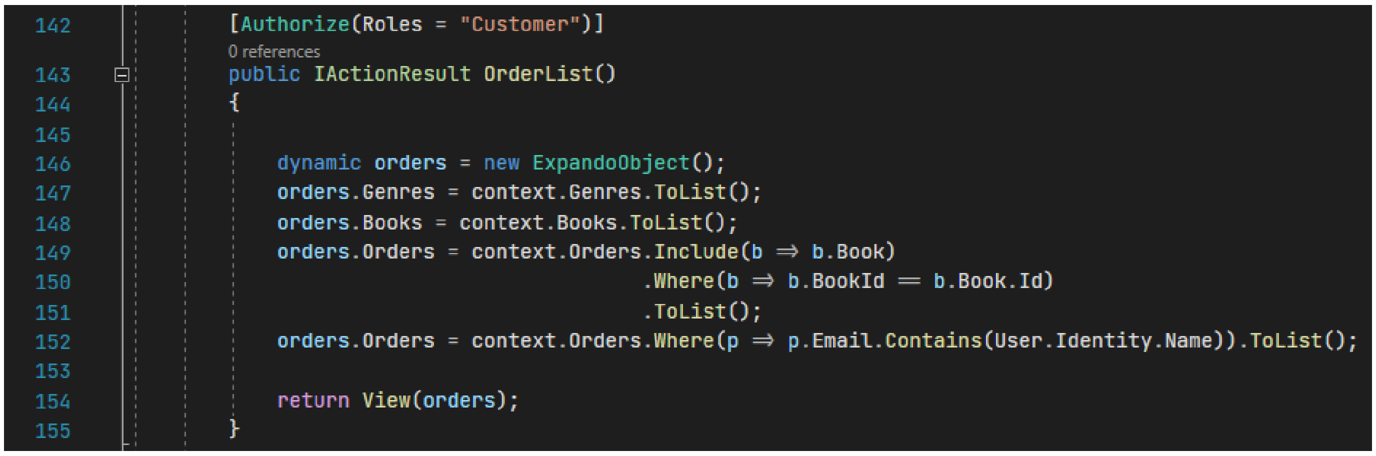
From line 122 – 128, we create a new order into table order.

With the id we get from the submit form, also with price and quantity of the target book, we can make a order to add in the order table.

Line 130 – 133, this is the code to update the quantity of book in the book table.

Line 135 – 137, this is the code to add order to table then save it. Lastly redirect to the user order list.

Action OrderList



```
142        [Authorize(Roles = "Customer")]
           0 references
143        public IActionResult OrderList()
144        {
145
146            dynamic orders = new ExpandoObject();
147            orders.Genres = context.Genres.ToList();
148            orders.Books = context.Books.ToList();
149            orders.Orders = context.Orders.Include(b => b.Book)
150                                          .Where(b => b.BookId == b.Book.Id)
151                                          .ToList();
152            orders.Orders = context.Orders.Where(p => p.Email.Contains(User.Identity.Name)).ToList();
153
154            return View(orders);
155        }
```

We use the dynamic model to generate a genre list on the nav go with the layout. This is a sub function.

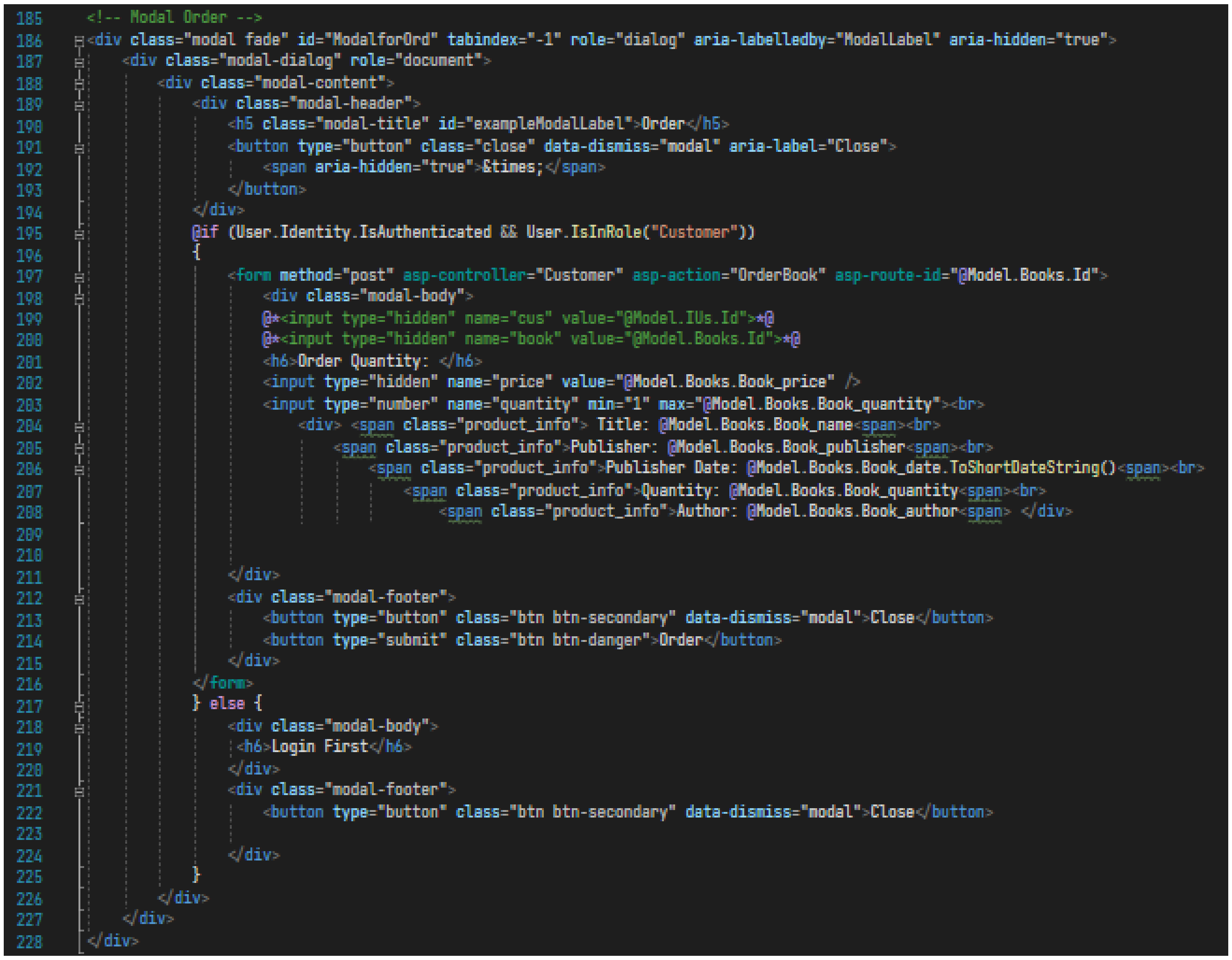
But mainly, line 149 – 152, these are the codes to help create the current user account order list.

Line 149 – 151, we use this to find the name of the book that current customer have order.

Line 152, this line is to distinction the order list of different users. To be most clear, I want to see only my account list of order, not with other orders of other account.

+) Customer's View:

View BookDetail



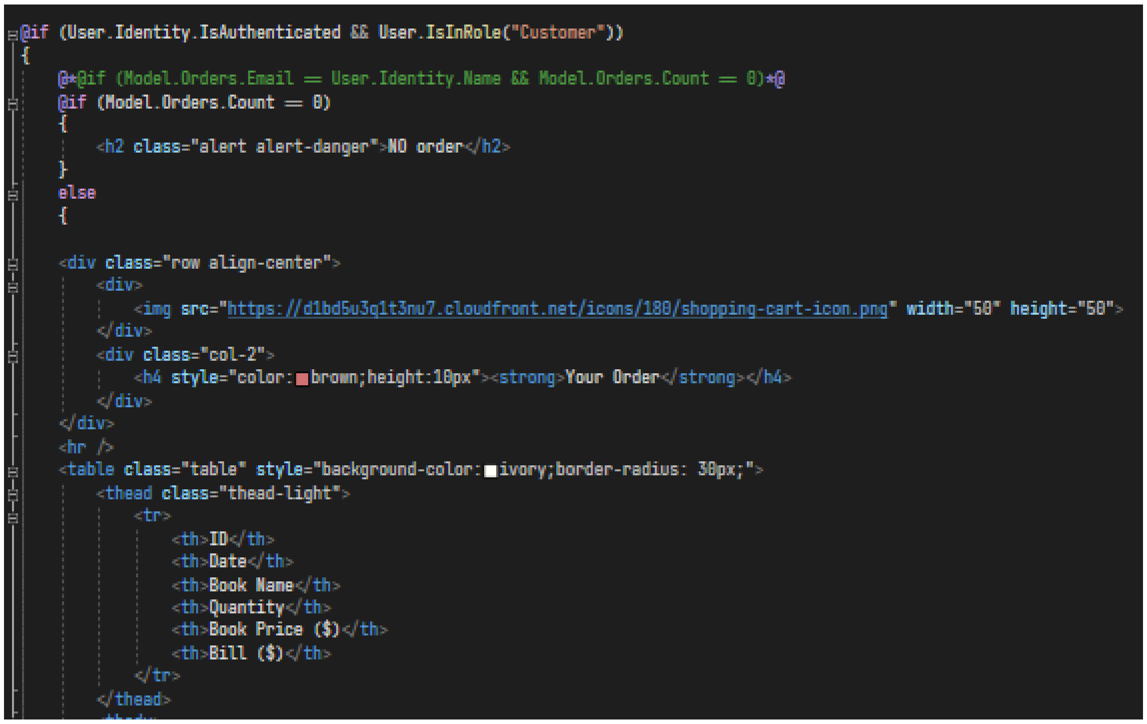
```
185     <!-- Modal Order -->
186     <div class="modal fade" id="ModalforOrd" tabindex="-1" role="dialog" aria-labelledby="ModalLabel" aria-hidden="true">
187         <div class="modal-dialog" role="document">
188             <div class="modal-content">
189                 <div class="modal-header">
190                     <h5 class="modal-title" id="exampleModalLabel">Order</h5>
191                     <button type="button" class="close" data-dismiss="modal" aria-label="Close">
192                         <span aria-hidden="true">&times;</span>
193                     </button>
194                 </div>
195                 @if (User.Identity.IsAuthenticated && User.IsInRole("Customer"))
196                 {
197                     <form method="post" asp-controller="Customer" asp-action="OrderBook" asp-route-id="@Model.Books.Id">
198                         <div class="modal-body">
199                         @*<input type="hidden" name="cus" value="@Model.IUs.Id">*@
200                         @*<input type="hidden" name="book" value="@Model.Books.Id">*@
201                         <h6>Order Quantity: </h6>
202                         <input type="hidden" name="price" value="@Model.Books.Book_price" />
203                         <input type="number" name="quantity" min="1" max="@Model.Books.Book_quantity"><br>
204                             <div> <span class="product_info"> Title: @Model.Books.Book_name<span><br>
205                                 <span class="product_info">Publisher: @Model.Books.Book_publisher<span><br>
206                                     <span class="product_info">Publisher Date: @Model.Books.Book_date.ToShortDateString()<span><br>
207                                         <span class="product_info">Quantity: @Model.Books.Book_quantity<span><br>
208                                             <span class="product_info">Author: @Model.Books.Book_author<span> </div>
209 
210 
211                     </div>
212                     <div class="modal-footer">
213                         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
214                         <button type="submit" class="btn btn-danger">Order</button>
215                     </div>
216                 </form>
217                 } else {
218                     <div class="modal-body">
219                      <h6>Login First</h6>
220                     </div>
221                     <div class="modal-footer">
222                         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
223 
224                     </div>
225                 }
226             </div>
227         </div>
228     </div>
```

We use a modal to appear when customer want to order.

The minimum quantity is 1 and the maximum is depend on the quantity of the target book.

This function is only works if the customer login with an account, if not line 128 – 224 will remind them.

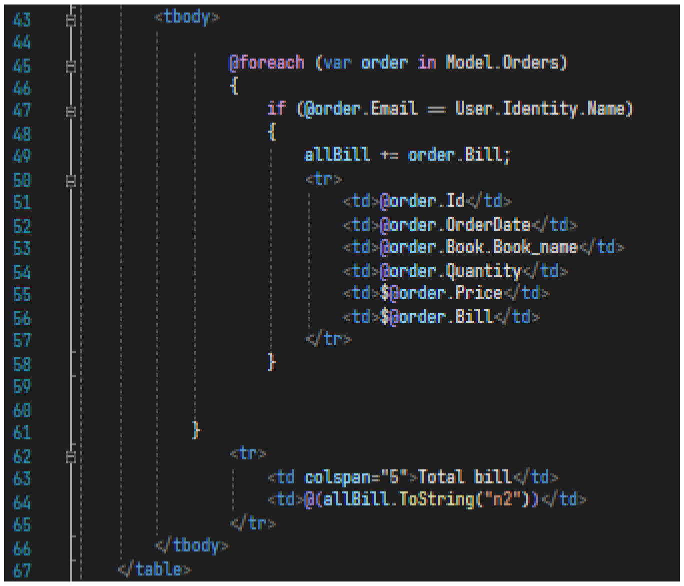
View OrderList



```
@if (User.Identity.IsAuthenticated && User.IsInRole("Customer"))
{
    @*@if (Model.Orders.Email == User.Identity.Name && Model.Orders.Count == 0)*@
    @if (Model.Orders.Count == 0)
    {
        <h2 class="alert alert-danger">NO order</h2>
    }
    else
    {

    <div class="row align-center">
        <div>
            <img src="https://d1bd5u3q1t3nu7.cloudfront.net/icons/180/shopping-cart-icon.png" width="50" height="50">
        </div>
        <div class="col-2">
            <h4 style="color:brown;height:10px"><strong>Your Order</strong></h4>
        </div>
    </div>
    <hr />
    <table class="table" style="background-color:ivory;border-radius: 30px;">
        <thead class="thead-light">
            <tr>
                <th>ID</th>
                <th>Date</th>
                <th>Book Name</th>
                <th>Quantity</th>
                <th>Book Price ($)</th>
                <th>Bill ($)</th>
            </tr>
        </thead>
```

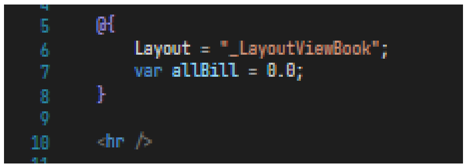
Firstly, we create a table form to see all of the current user table.



```
        <tbody>

                @foreach (var order in Model.Orders)
                {
                    if (@order.Email == User.Identity.Name)
                    {
                        allBill += order.Bill;
                        <tr>
                            <td>@order.Id</td>
                            <td>@order.OrderDate</td>
                            <td>@order.Book.Book_name</td>
                            <td>@order.Quantity</td>
                            <td>$@order.Price</td>
                            <td>$@order.Bill</td>
                        </tr>
                    }


            }
                <tr>
                    <td colspan="5">Total bill</td>
                    <td>@(allBill.ToString("n2"))</td>
                </tr>
        </tbody>
    </table>
```

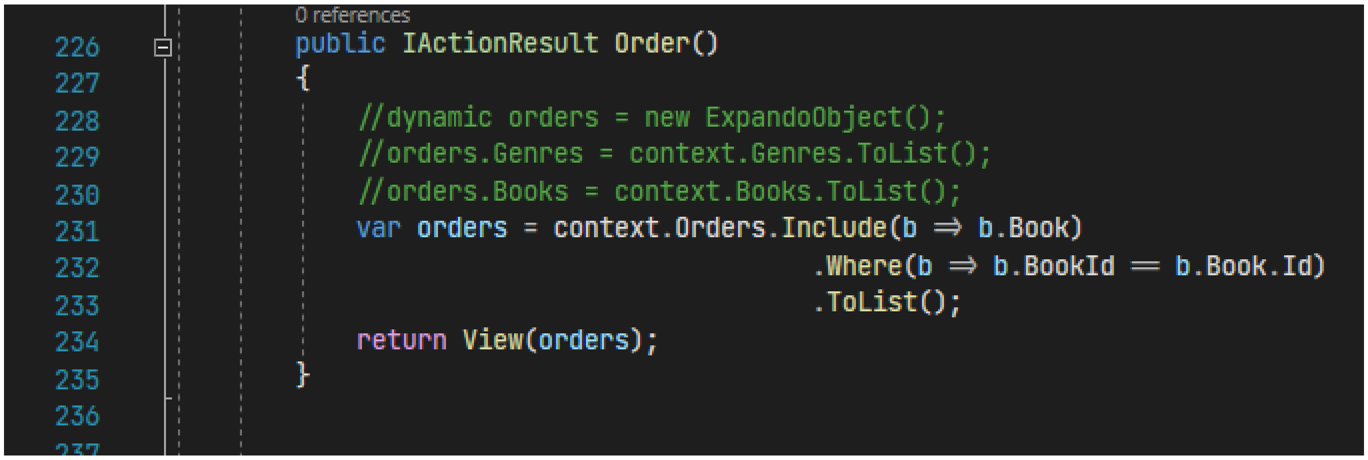


```
@{
    Layout = "_LayoutViewBook";
    var allBill = 0.0;
}

<hr />
```

After that, we create a variable to sum the total money that the current user has spent on books.

+) Store Owner's Controller:

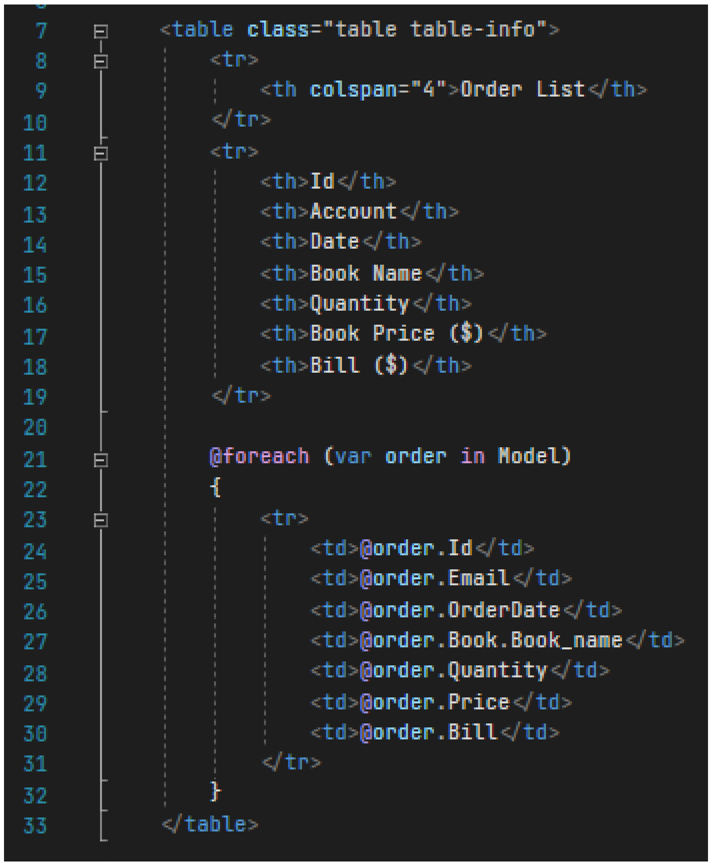


```
            0 references
226         public IActionResult Order()
227         {
228             //dynamic orders = new ExpandoObject();
229             //orders.Genres = context.Genres.ToList();
230             //orders.Books = context.Books.ToList();
231             var orders = context.Orders.Include(b => b.Book)
232                                         .Where(b => b.BookId == b.Book.Id)
233                                         .ToList();
234             return View(orders);
235         }
236
```

We need line 231 – 233 to help reveal all of the orders of all account.

+) Store Owner's View:

View Order



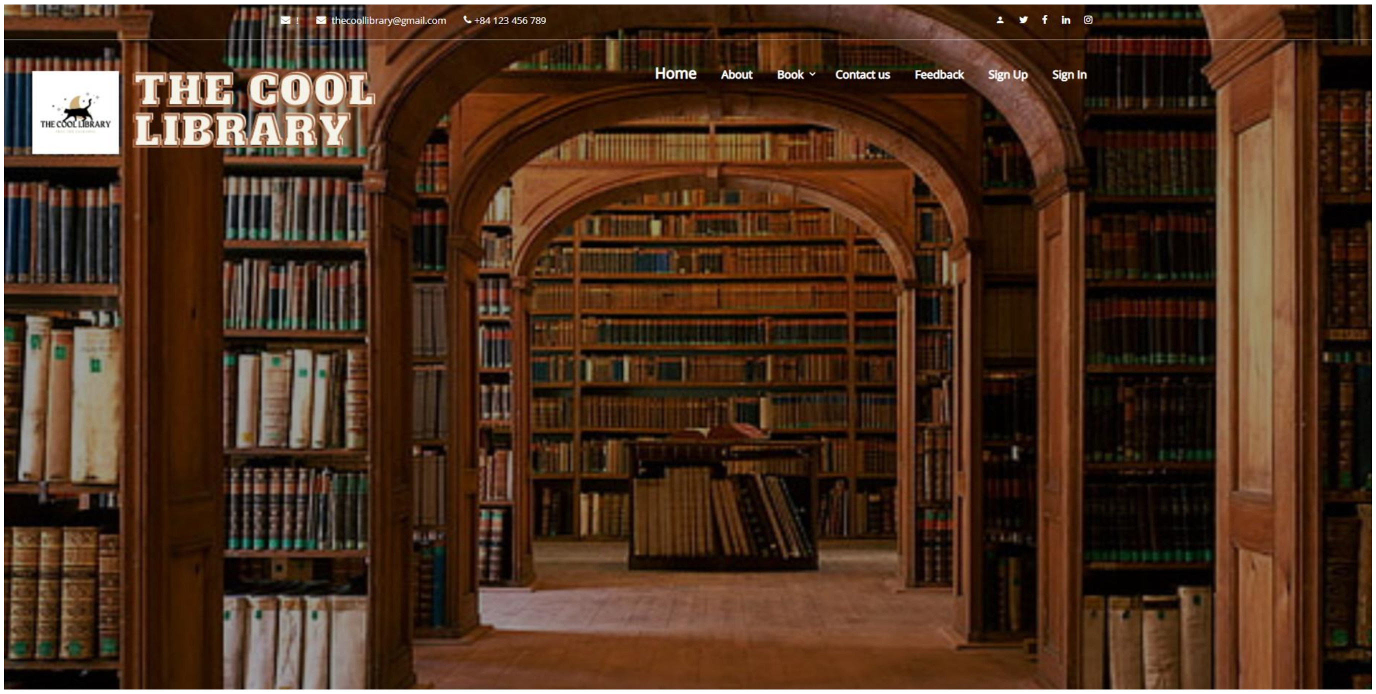
```
<table class="table table-info">
    <tr>
        <th colspan="4">Order List</th>
    </tr>
    <tr>
        <th>Id</th>
        <th>Account</th>
        <th>Date</th>
        <th>Book Name</th>
        <th>Quantity</th>
        <th>Book Price ($)</th>
        <th>Bill ($)</th>
    </tr>

    @foreach (var order in Model)
    {
        <tr>
            <td>@order.Id</td>
            <td>@order.Email</td>
            <td>@order.OrderDate</td>
            <td>@order.Book.Book_name</td>
            <td>@order.Quantity</td>
            <td>@order.Price</td>
            <td>@order.Bill</td>
        </tr>
    }
</table>
```

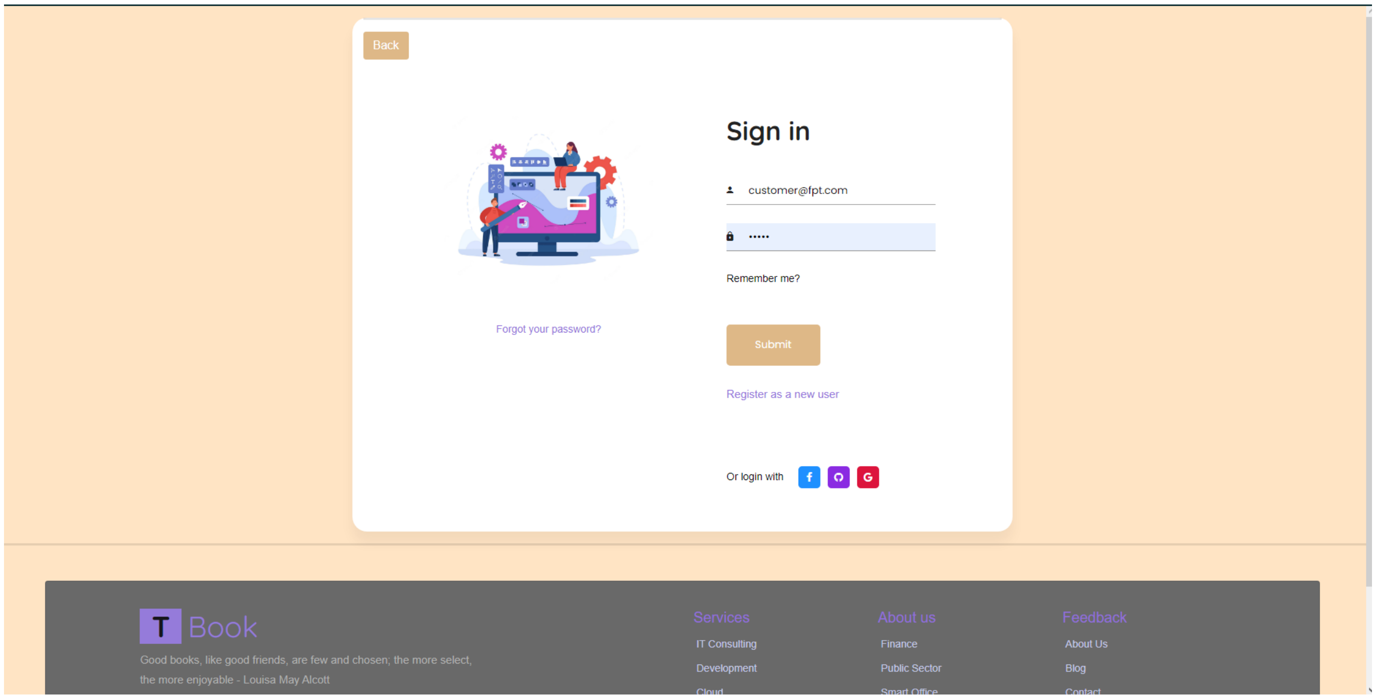
Here are the table of all order. In other word, this is the all orders list.

3) Final screen shot

The Index page

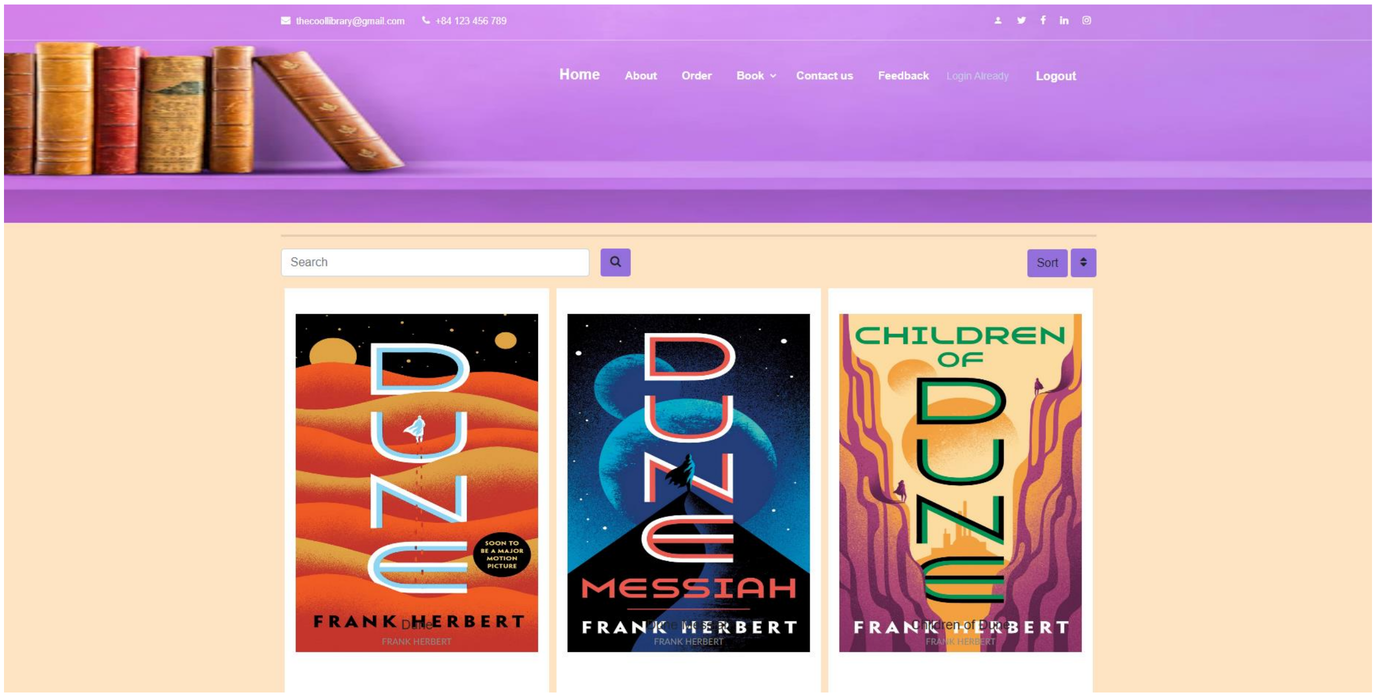


Login page

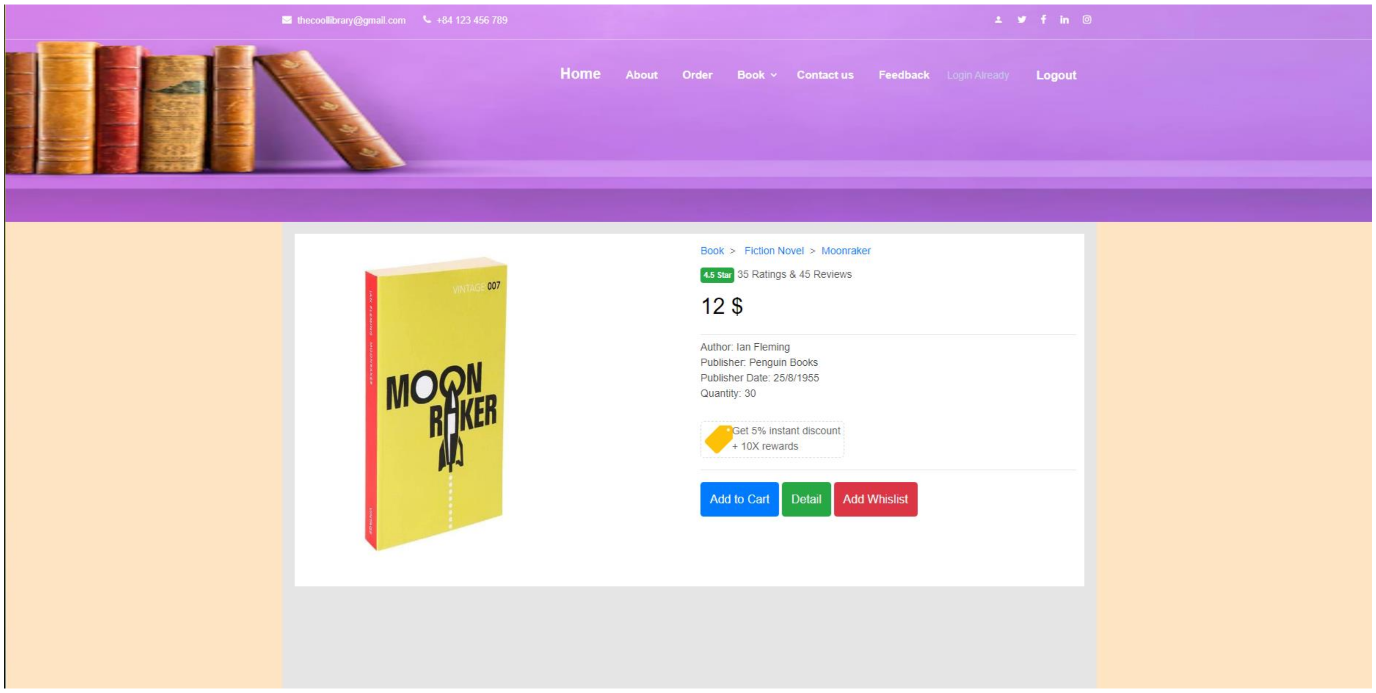


Good books, like good friends, are few and chosen; the more select, the more enjoyable - Louisa May Alcott

All Book page



Book Detail page



Book > Fiction Novel > Moonraker

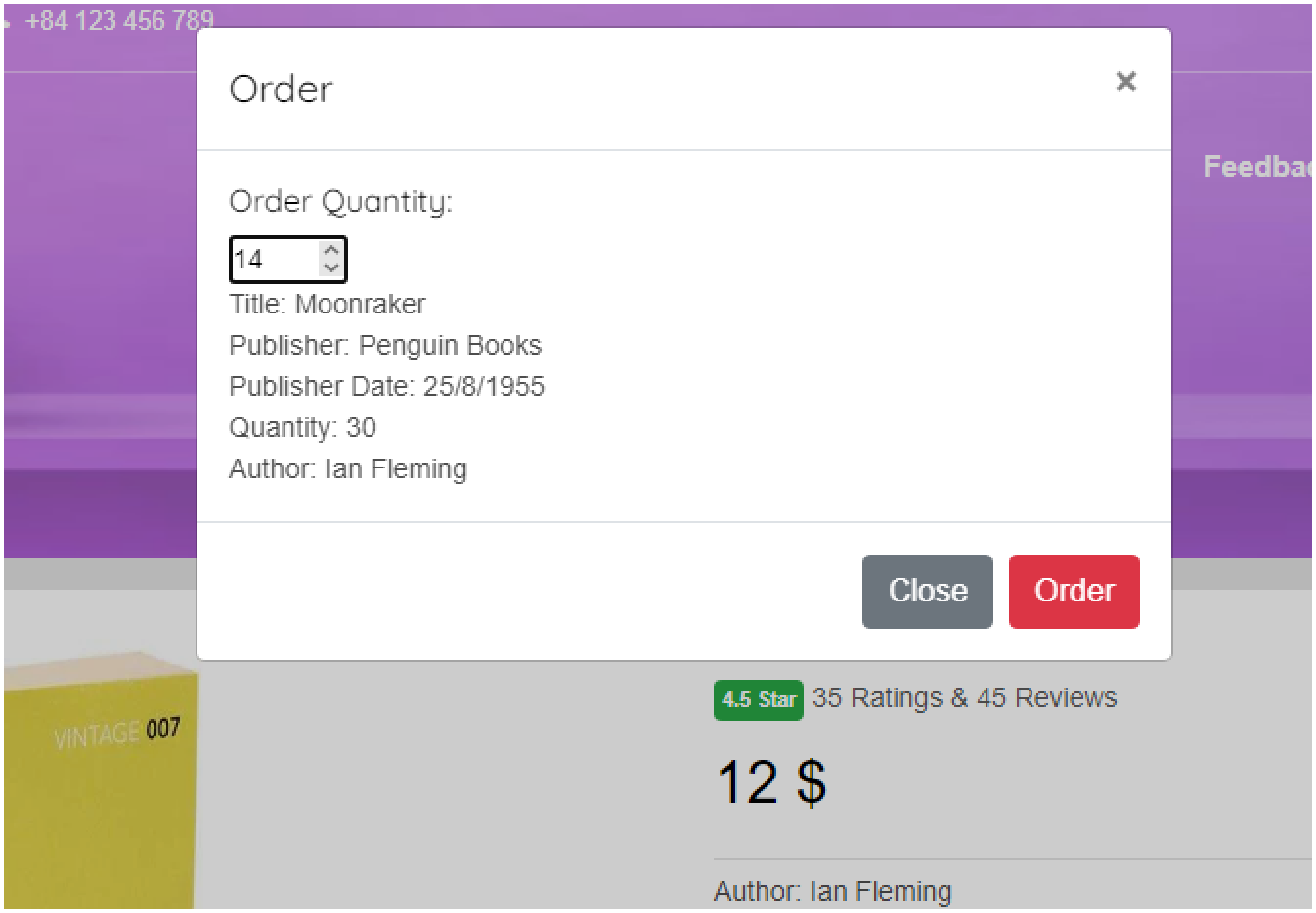
4.5 Star 35 Ratings & 45 Reviews

12 $

Author: Ian Fleming
Publisher: Penguin Books
Publisher Date: 25/8/1955
Quantity: 30

Get 5% instant discount
+ 10X rewards

Order modal



+84 123 456 789

Order

Order Quantity:

14

Title: Moonraker

Publisher: Penguin Books

Publisher Date: 25/8/1955

Quantity: 30

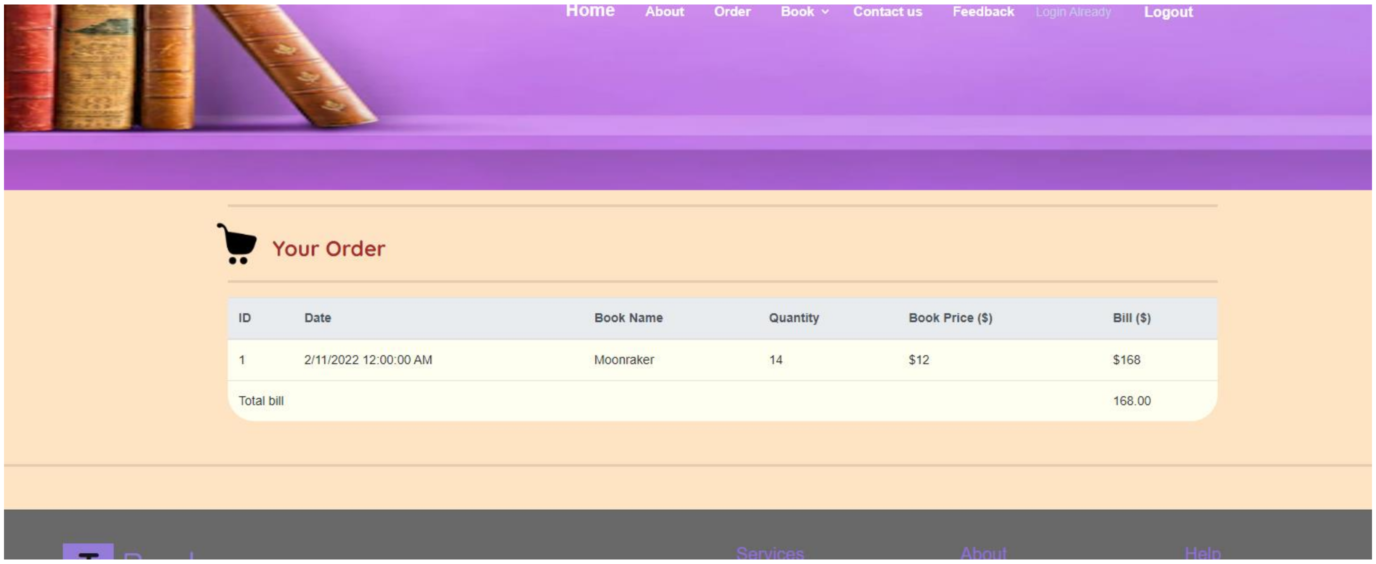
Author: Ian Fleming

35 Ratings & 45 Reviews

12 $

Author: Ian Fleming

Order list

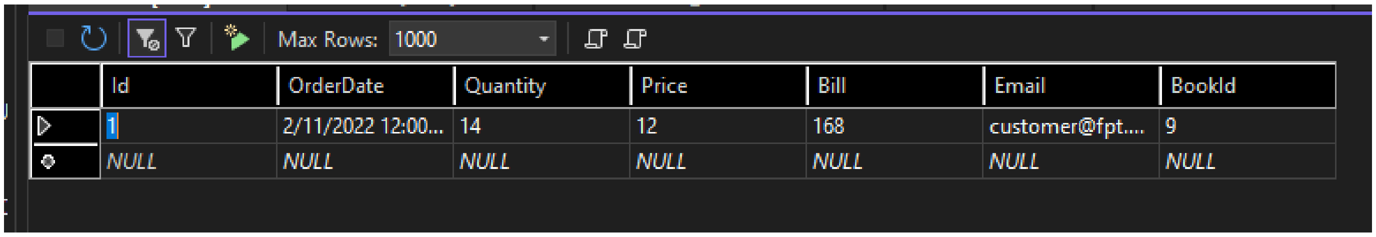


Your Order

ID	Date	Book Name	Quantity	Book Price ($)	Bill ($)
1	2/11/2022 12:00:00 AM	Moonraker	14	$12	$168
Total bill					168.00

Inside of the SQL server:

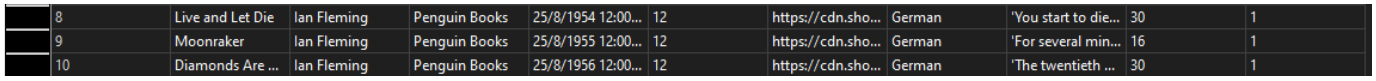
Order table



Max Rows: 1000

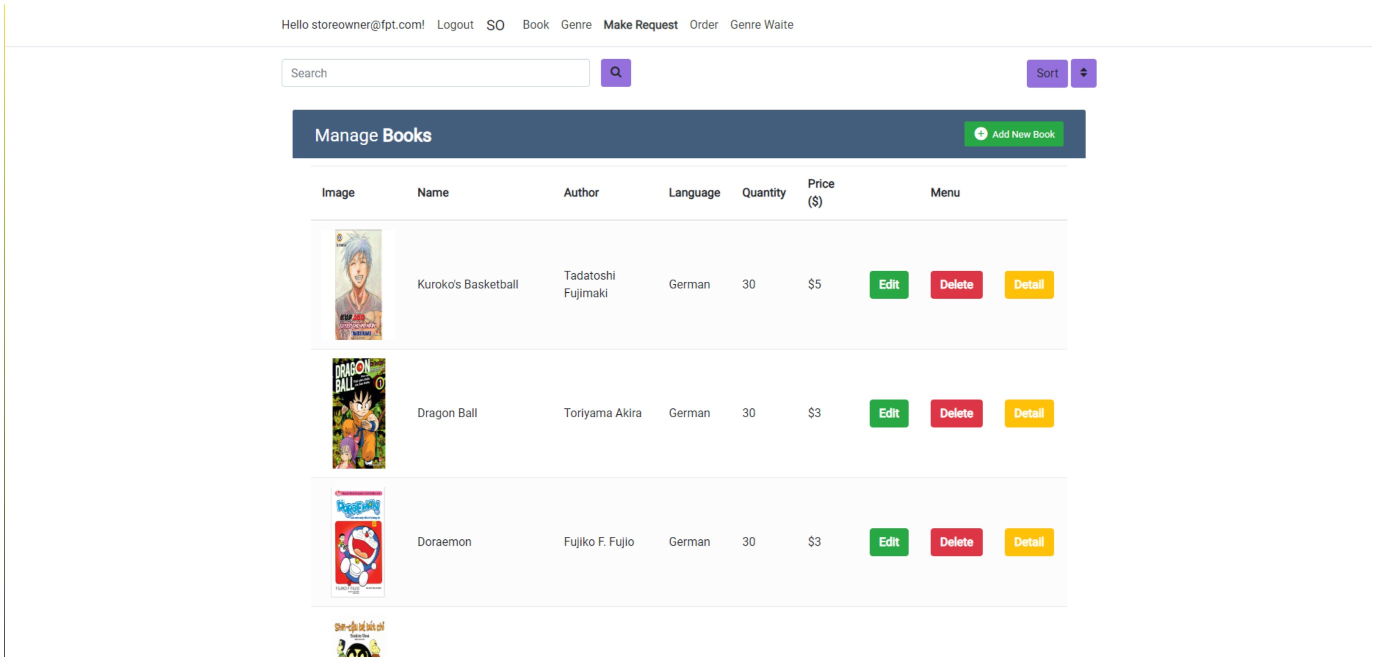
	Id	OrderDate	Quantity	Price	Bill	Email	BookId
	1	2/11/2022 12:00...	14	12	168	customer@fpt....	9
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Book table



8	Live and Let Die	Ian Fleming	Penguin Books	25/8/1954 12:00...	12	https://cdn.sho...	German	'You start to die...	30	1
9	Moonraker	Ian Fleming	Penguin Books	25/8/1955 12:00...	12	https://cdn.sho...	German	'For several min...	16	1
10	Diamonds Are ...	Ian Fleming	Penguin Books	25/8/1956 12:00...	12	https://cdn.sho...	German	'The twentieth ...	30	1

Store Owner site

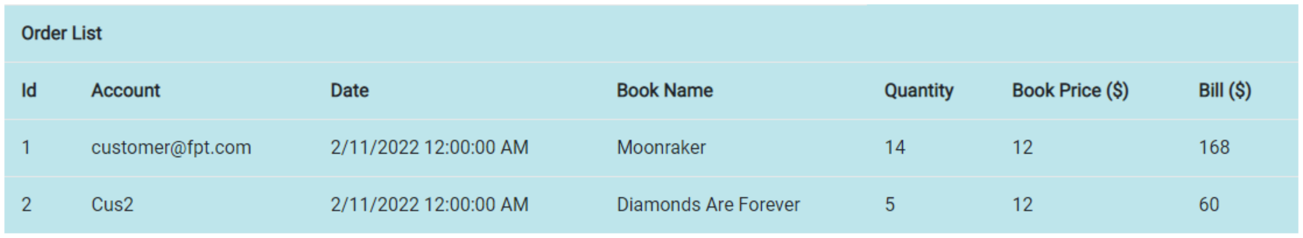

Hello storeowner@fpt.com! Logout SO Book Genre **Make Request** Order Genre Waite

Search | Sort

Manage Books | Add New Book

Image	Name	Author	Language	Quantity	Price ($)	Menu		
	Kuroko's Basketball	Tadatoshi Fujimaki	German	30	$5	Edit	Delete	Detail
	Dragon Ball	Toriyama Akira	German	30	$3	Edit	Delete	Detail
	Doraemon	Fujiko F. Fujio	German	30	$3	Edit	Delete	Detail

Order page of store owner


Order List

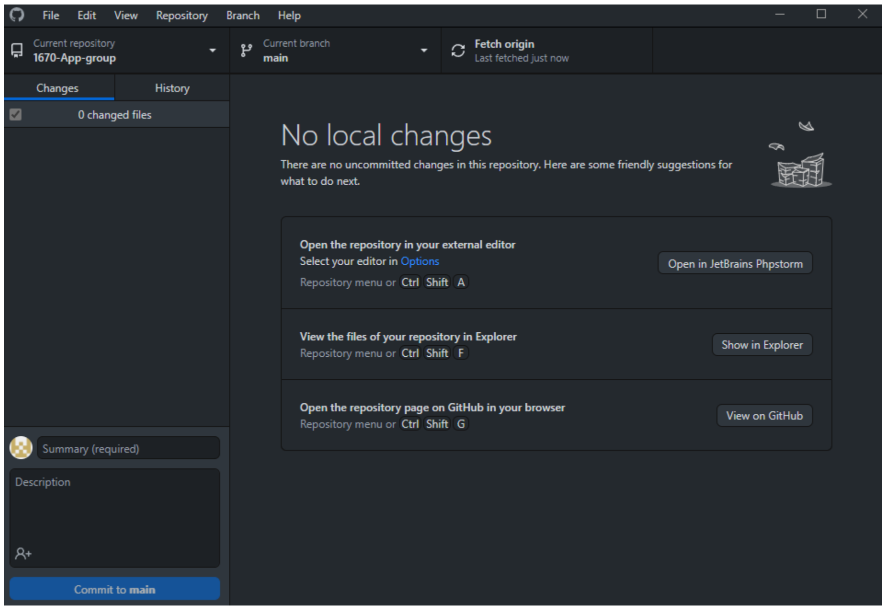
Id	Account	Date	Book Name	Quantity	Book Price ($)	Bill ($)
1	customer@fpt.com	2/11/2022 12:00:00 AM	Moonraker	14	12	168
2	Cus2	2/11/2022 12:00:00 AM	Diamonds Are Forever	5	12	60

This table will have all of the orders from every account. Store owner can see it all.

4) Screenshots of using GitHub to manage the source code

Link: https://github.com/ducmtgch200681/1670-App-group

GitHub Desktop:



No local changes

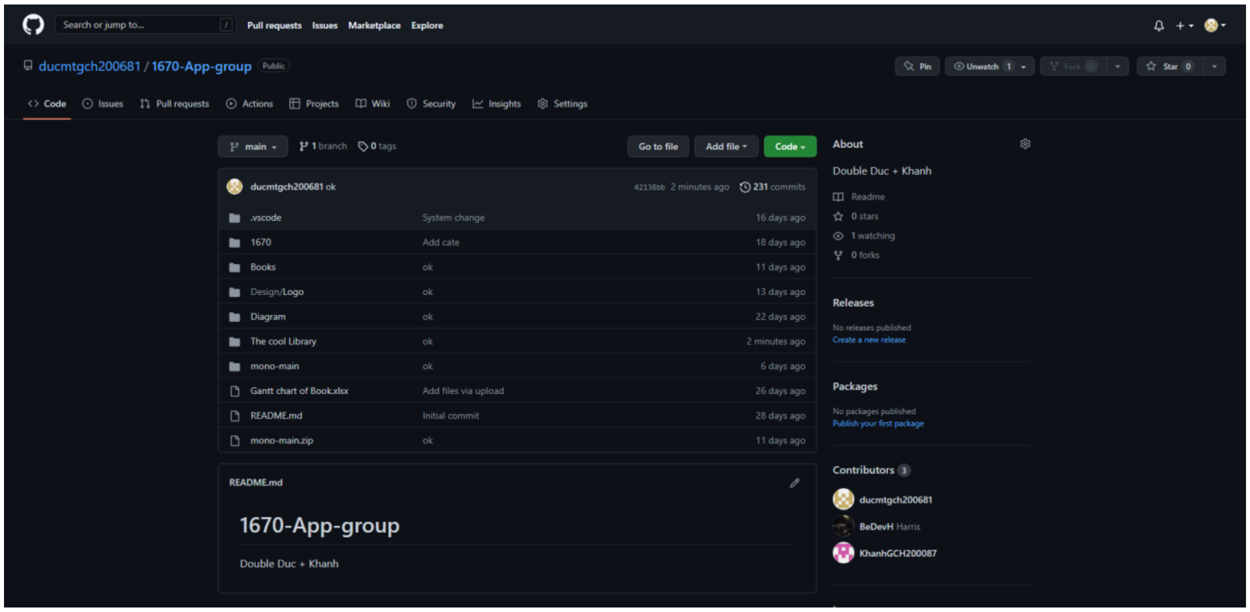
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A
Open in JetBrains Phpstorm

View the files of your repository in Explorer
Repository menu or Ctrl Shift F
Show in Explorer

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G
View on GitHub

GitHub on website:



ducmtgch200681 ok 42138bb 2 minutes ago 231 commits

.vscode	System change	16 days ago
1670	Add cate	18 days ago
Books	ok	11 days ago
Design/Logo	ok	13 days ago
Diagram	ok	22 days ago
The cool Library	ok	2 minutes ago
mono-main	ok	6 days ago
Gantt chart of Book.xlsx	Add files via upload	26 days ago
README.md	Initial commit	28 days ago
mono-main.zip	ok	11 days ago

README.md

1670-App-group

Double Duc + Khanh

About

Double Duc + Khanh

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

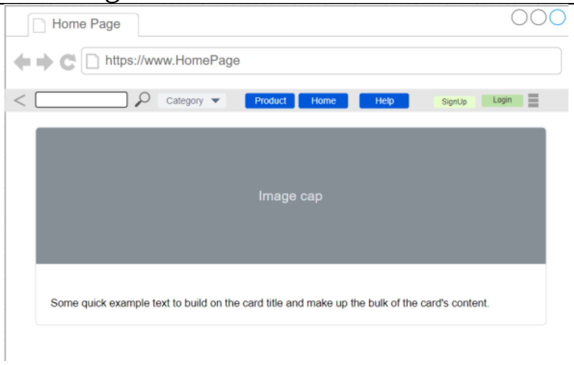
ducmtgch200681
BeDevH Harris
KhanhGCH200087

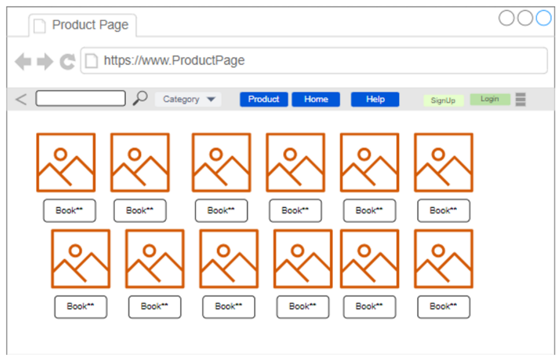
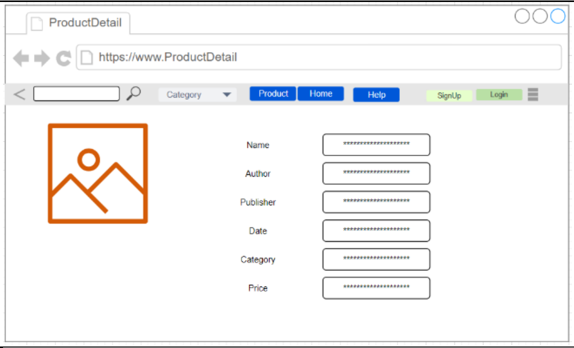
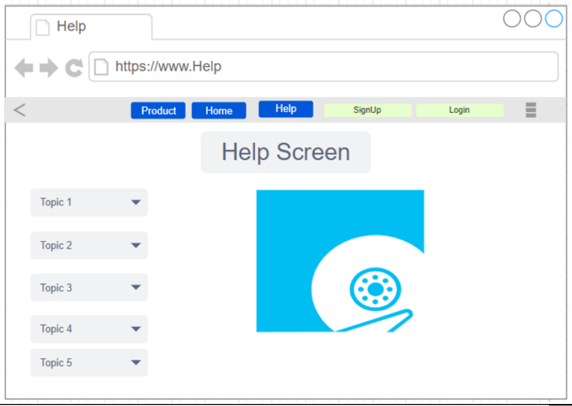
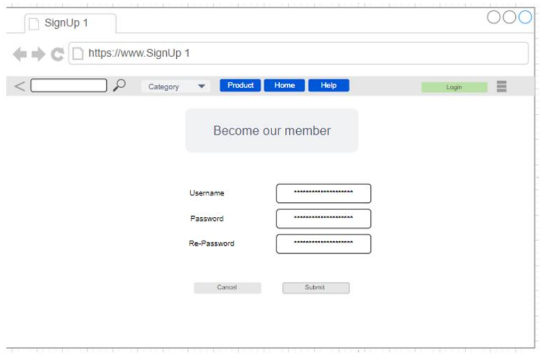
III. APPLICATION EVALUATION

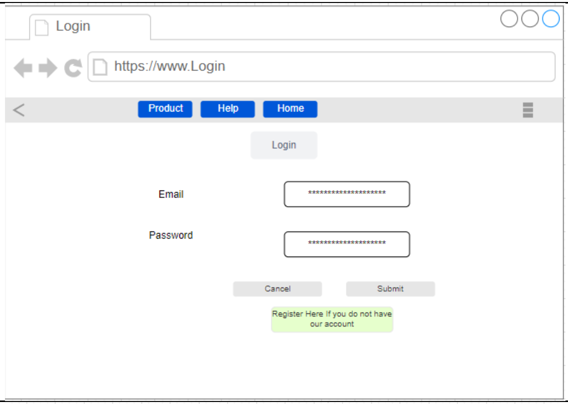
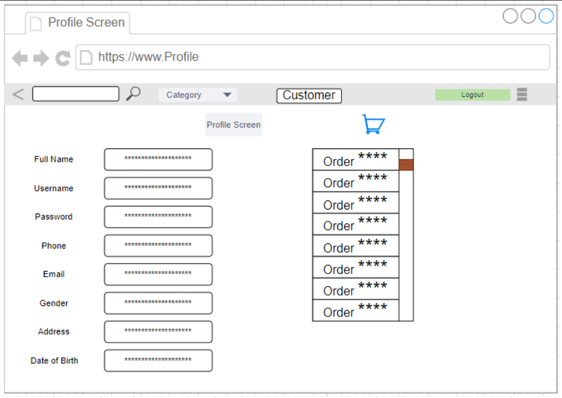
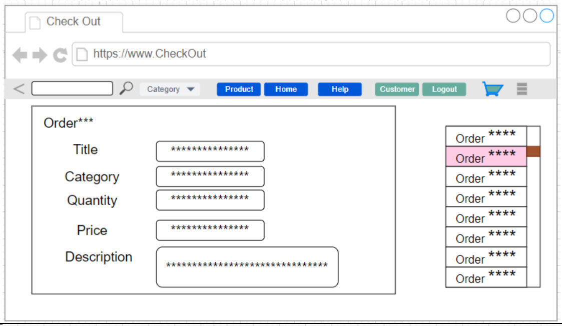
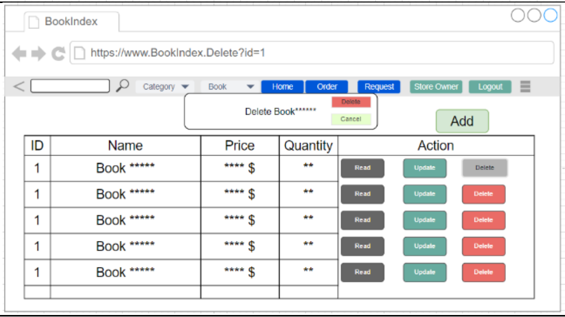
1) Performance of the application

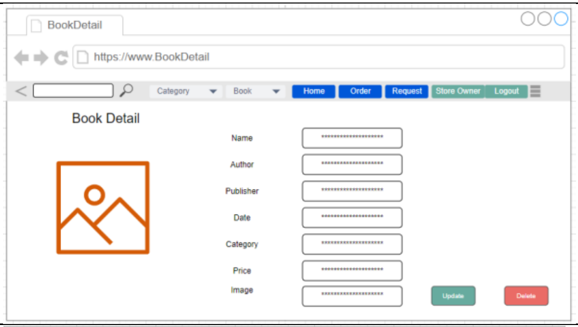
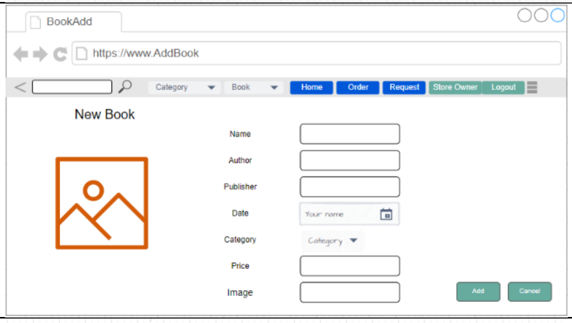
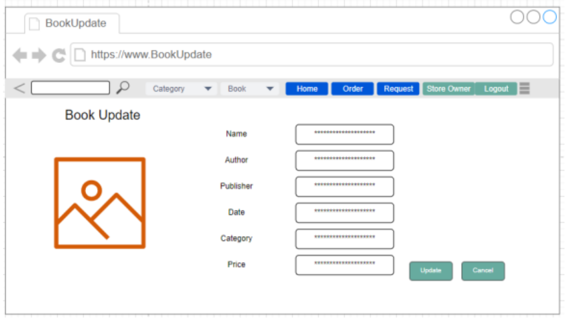
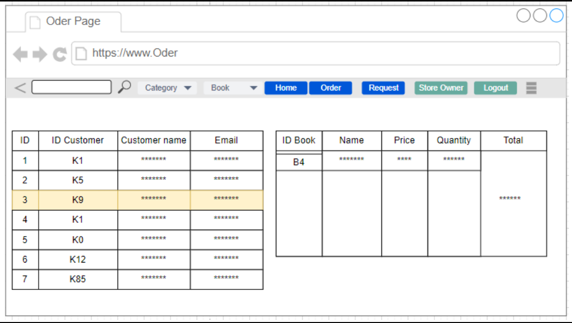
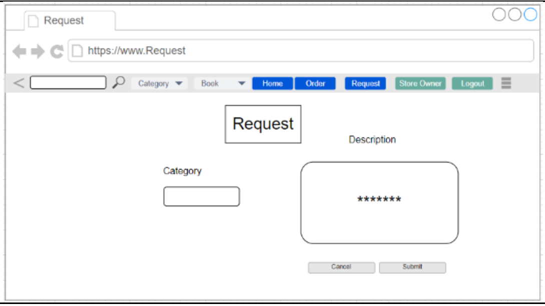
With good design and well equip function, we have created a nicely done web application. The website is fast and there are no sign of bugs or errors.

But as a developer and also a user, I think our project can get better with some more function like a cart for user, a profile management, delete account… This project have so many potential and next time when I have a chance to do some thing like this, I will definitely add them features.

Role	Name	UI Design	Note
Customer	Index	 Home Page https://www.HomePage Category Product Home Help SignUp Login Image cap Some quick example text to build on the card title and make up the bulk of the card's content.	The Index page has a better interface and detail. It has more function than UI design. Navbar of the Index page have more function than UI design so user can have a better experience. →Index home page is way more better than UI design page

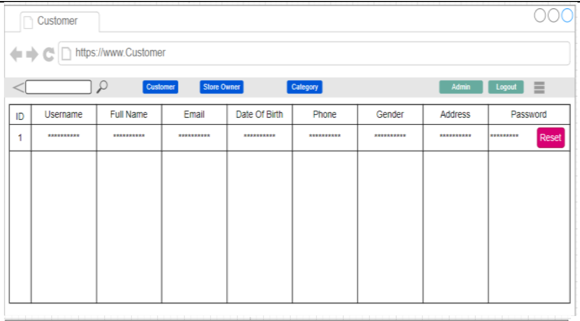
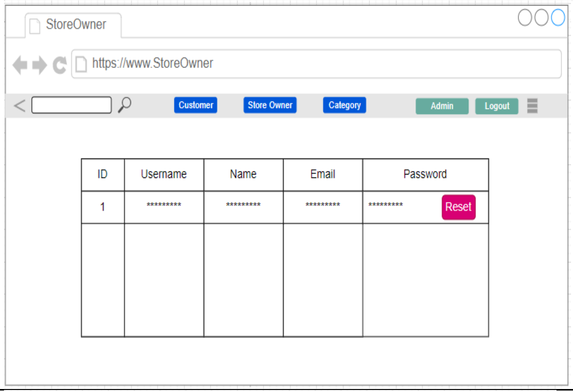
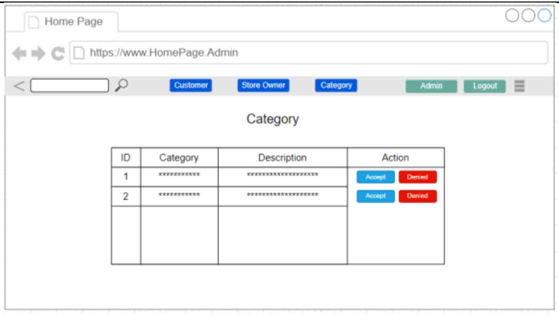
	All Book Page		The all book page have a better design, it has more color, detail and have a sort function by name and price, which UI design do not have. Search bar in the all book page is different than UI design, one is in the navbar, another is in the page. ➔ Actual page the all book page is better than UI design page
	Book Detail Page		Book detail page of UI design is simple and do not have much function like actual page. It does not include path for Product page, Book genre, and book name. UI design do not have function Order or read book detail. →Book detail page is better than UI design page.
	Other page		We also have about page, contact us page, and feedback page,…
	Sign Up page		In UI design page, it has a nav bar but in factor page, we decide to remove the nav bar and change it to a button, which user will be returned to home page if is pressed. →Sign up page has a better layout, it has more color, image than UI design page. In UI design, after user enter Username, password, co-password, it will redirect user to customer detail page, which actual page do not have. ➔ UI design page is better because it is more detail than actual page

	Sign In page	 Login https://www.Login Product Help Home Login Email Password Cancel Submit Register Here if you do not have our account	In UI design, Login page has a nav bar, which factor page do not have. Factor page have a better layout than UI design page because it has more icon, color, image and reasonable button with link. →Sign In page is better than UI design page
Customer	Profile page, Checkout page, Order page	 Profile Screen https://www.Profile Category Customer Logout Profile Screen Full Name Username Password Phone Email Gender Address Date of Birth Order ****  Check Out https://www.CheckOut Category Product Home Help Customer Logout Order*** Title Category Quantity Price Description Order ****	In UI design, user have a profile screen, where they can view their data and their cart. But in actual page, user only have an order page, where they can view their order In UI design, user have a Checkout page, where they can check their order, which in actual page, order page have a same function but it has a better layout. →UI design is better than page design because it has more detail and function.
Store Owner	Delete a book in book list	 BookIndex https://www.BookIndex.Delete?id=1 Category Book Home Order Request Store Owner Logout Delete Book***** Delete Cancel Add ID Name Price Quantity Action 1 Book ***** **** $ ** Read Update Delete	Delete function in UI design and factor design look the same.

	Book Detail	 BookDetail / https://www.BookDetail / Category / Book / Home / Order / Request / Store Owner / Logout / Book Detail / Name / Author / Publisher / Date / Category / Price / Image / Update / Delete	Book detail page of UI design is simple and do not have much function like factor page. It does not include path for Product page, Book genre, and book name. →The book detail page is better than UI design page.
	Add new book	 BookAdd / https://www.AddBook / Category / Book / Home / Order / Request / Store Owner / Logout / New Book / Name / Author / Publisher / Date / Your name / Category / Category / Price / Image / Add / Cancel	Add new book page of UI design is simple and do not have much function and data field like the actual page. →The add new book page is better than UI design page.
	Update book page	 BookUpdate / https://www.BookUpdate / Category / Book / Home / Order / Request / Store Owner / Logout / Book Update / Name / Author / Publisher / Date / Category / Price / Update / Cancel	Update book page of UI design is simple and do not have much function and data field like factor page. →The update book page is better than UI design page.
	Customer order page	 Oder Page / https://www.Oder / Category / Book / Home / Order / Request / Store Owner / Logout	Customer order book page of UI design is better than order book design because it has a smart and scientific layout. Customer order book of UI design is better than factor design
	Make request genre page	 Request / https://www.Request / Category / Book / Home / Order / Request / Store Owner / Logout / Request / Description / Category / ******* / Cancel / Submit	Make request genre page design is has a same function but a better layout than UI design.
	Genre list page		→ Genre list design is where people can view available genre in the database

ID	ID Customer	Customer name	Email
1	K1	*******	*******
2	K5	*******	*******
3	K9	*******	*******
4	K1	*******	*******
5	K0	*******	*******
6	K12	*******	*******
7	K85	*******	*******

ID Book	Name	Price	Quantity	Total
B4	*******	****	******	******

			and check if admin accept their request genre or not
Admin	Detail page for customer and store Owner	 Customer https://www.Customer Customer Store Owner Category Admin Logout ID Username Full Name Email Date Of Birth Phone Gender Address Password Reset  StoreOwner https://www.StoreOwner Customer Store Owner Category Admin Logout ID Username Name Email Password Reset	UI design page have a detail page, where they can view data of customer and storeowner, and reset their account password if needed, which factor design page do not have.
	Request genre page	 Home Page https://www.HomePage.Admin Customer Store Owner Category Admin Logout Category ID Category Description Action Accept Denied	Request genre page design have a same function but a better layout than UI design page.

2) Strengths and Weaknesses of the application

Strengths:

- The system is more practical than it is detailed, containing fields like publisher, date, and author as well as a search feature, sort by name, and book price.
- In comparison to the specifications in the SRS document, the system has achieved 80% of the functionalities.
- The system's design is superior to UI design.
- Comparing the system's upgraded pages to the suggested design.

Weaknesses:

- The system contains numerous minor typographical and syntax issues.
- The system is still in operation, and features like the shopping cart and the ability to access user information should not be disabled when the account or password is incorrect.