LONDON METROPOLITAN UNIVERSITY

CS6004ES

Application Development Group Coursework - 2

Name: Raman Lakshan

LMU ID: 21038992

TEAM MEMBERS:

LMU ID	STUDENT NAME
21038992	Raman Lakshan
21038989	Sivayogalingam Satheeskaran

DECLARATION:

We declare that all materials included in this report is the end result of our own work and that due acknowledgement have been given in the bibliography and references to all sources be they printed, electronic or personal.

Name: Raman Lakshan
Student ID: 21038992 (E016540)
Signature: ramanlakshan1995@gmail.com

Name: Sivayogalingam Satheeskaran
Student ID: 21038989 (E016551)
Signature: satheeskaran1995@gmail.com

ACKNOWLEDGEMENT.

Praise and thanks to Almighty God for the fulfillment of this assignment work. It is essential at this point to acknowledge with thanks those who have made useful contributions to facilitate the successful accomplishment of this academic task.

We desire to express or reserve gratitude to our supervisor, Ms. Malsha for her comments and suggestions for expanding our knowledge on our group project and assignment.

We are also grateful to other academic and non-academic staff members of Esoft Metro Campus Negombo for their ideas and knowledge.

Thank you.

TABLE OF CONTENTS.

Table of Contents

TABLE OF FIGURES.

TABLE OF TABLE.

“E-Book (PVT) Ltd”

01) ABSTRACT.

It refers to the purchasing and selling of products and services through the Internet, as well as the money and data transfers required to complete these transactions. E-commerce is most commonly associated with the selling of actual things over the Internet, although it can also apply to any sort of online business. E-book PVT Ltd is one of Sri Lanka's most well-known secondhand bookshops.

The administration has chosen to create an online store under an online shopping marketplace to supply consumers with a vast assortment of books used in Sri Lanka and overseas covering a wide range of disciplines. E-book management has opted to grow its company in order to meet the different demands of its diversified readership, which ranges from young children to highly educated individuals.

The report first details the functional and non-functional requirements addressed by the system, and then Diagrams. Diagram section contain system use case diagrams, class diagrams, ER diagrams, and architectural diagram. Lastly, installation guide and configuration and manual of system.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

02) REQURIMENT SPECIFICATION.

2.1 The Functional Requirements of The E-book PVT Ltd.

Product characteristics that developers must incorporate in order for users to execute tasks are known as functional requirements. As a result, communicating them to both the development team and stakeholders is crucial. In general, functional requirements outline how a system behaves in specific situations (Functional and Nonfunctional Requirements, 2022).

According to E-book PVT Ltd requirements this system will have two user points.

- **Admin:** This solution will act has the back-office system (BOS) for E-book PVT Ltd website.
- **Customer:** This solution will act has ecommerce system where the E-book PVT Ltd customers can view their products and purchase it.

Admin Functional Requirements

- There should be an admin login for the admin to log in to the system.
- Admin should be able to manage books.
- Admin should be able to manage book categories.
- Admin should be able to manage customers.
- Admin should be able to manage orders.
- Admin should be able to view reports.
- Admin should be able to review inquiries.

Customer Functional Requirements

- Customer should be able to register to the system.
- Customer should be able to login to the system by using their username and password.
- The customer should be able to search book.
- The customer should be able to place orders and do the payments.
- The customer should be able track their orders.
- The customers should be able to provide feedback about books.

2.2 The Non-functional Requirements of The E-book PVT Ltd.

Non-functional requirements determine how a system operates and the functionality constraints that must be met. This type of requirement is also known as a system quality trait (Functional and Nonfunctional Requirements, 2022).

- **Useability:** A website's visitors may easily engage with it. Visually appealing, straightforward navigation, readability, and responsiveness are just a few examples.
- **Accessibility:** Throughout, E-Book PVT (Ltd) verified that the platform met basic accessibility criteria.
- **Scalability and performance:** During regular and peak hours, the system was able to grow to meet predicted traffic and order quantities.
- **Security:** Any sort of information that must be safeguarded from unwanted access to preserve a customer's privacy or security is classified as confidential data. E-Book PVT (Ltd) verified that sensitive data would not be accessed by unauthorized individuals.
- **Quality:** Even the greatest e-commerce platforms might be misused, but E-Book PVT(Ltd) insists on high-quality code development.
- **Extensibility:** Ensured that the platform is expanded in a way that allows for future development.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

03) DESIGN.

The design process for the project is documented in this section of the document. The application summary with UML diagrams and the hardware and software requirements for running this system are briefly addressed here.

3.1 Software Architecture Diagram.

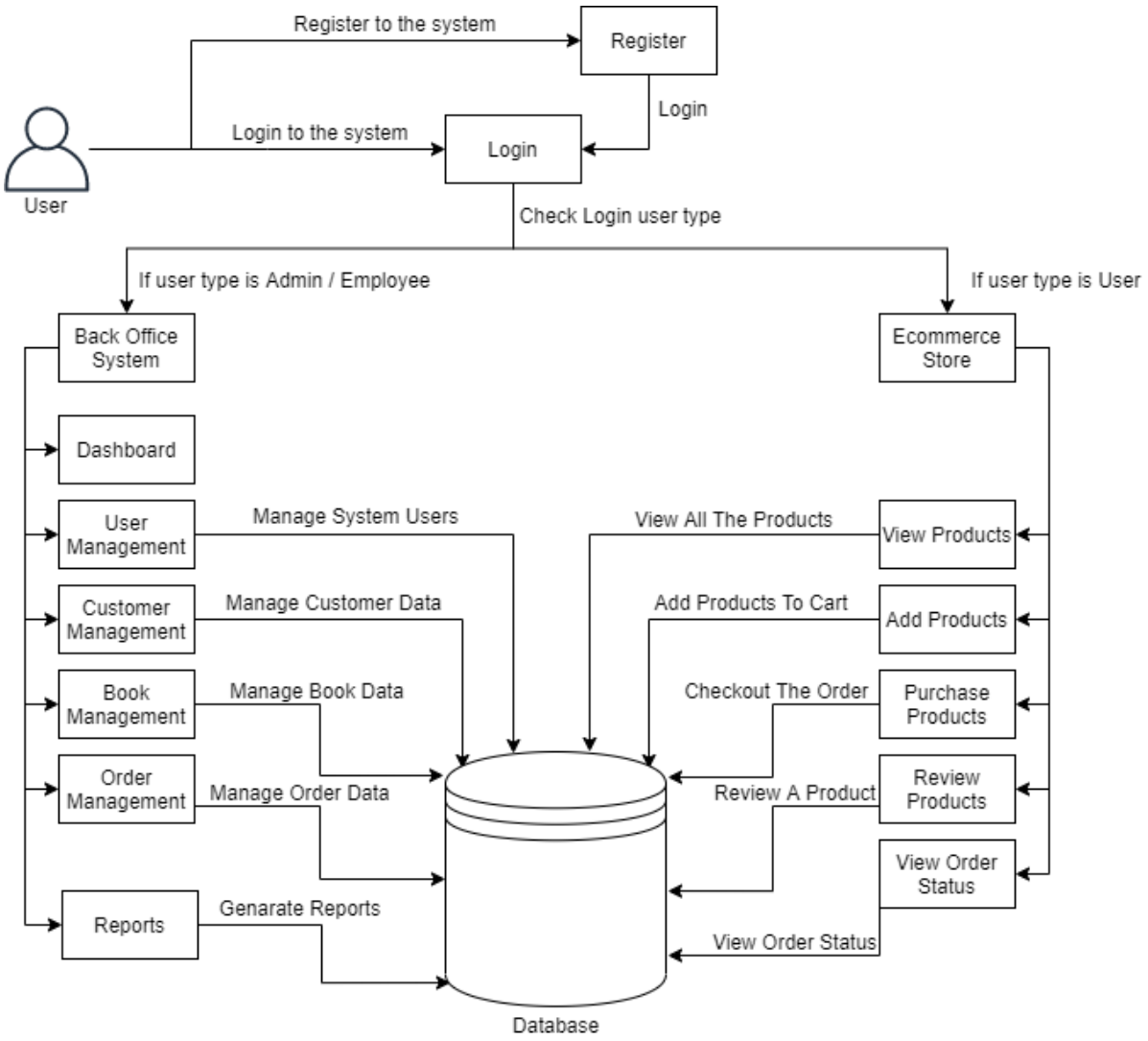


Figure 1 Software Architecture Diagram

3.2 Use Case Diagram.

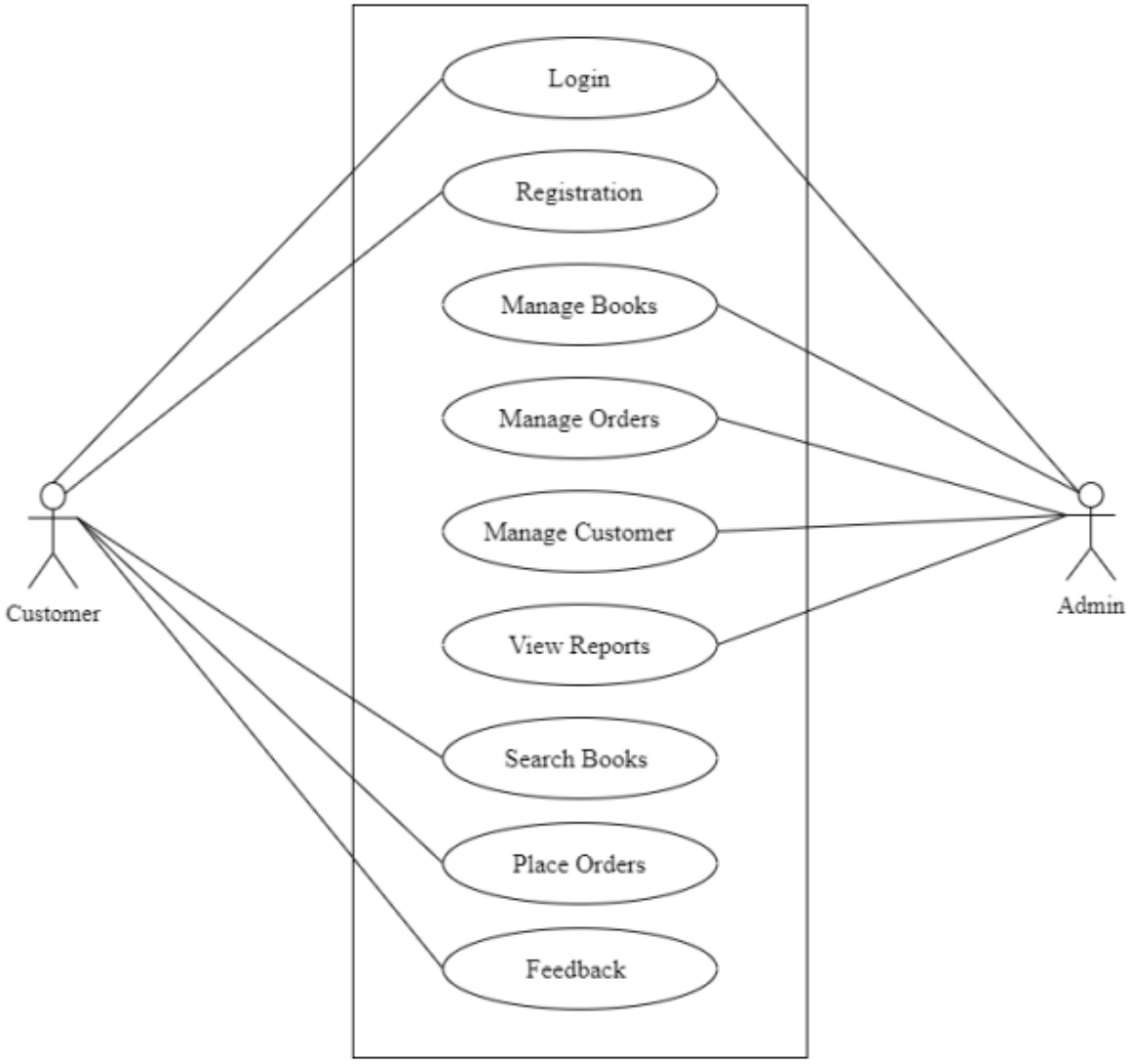


Figure 2 Use case diagram

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

3.3 Class Diagram.

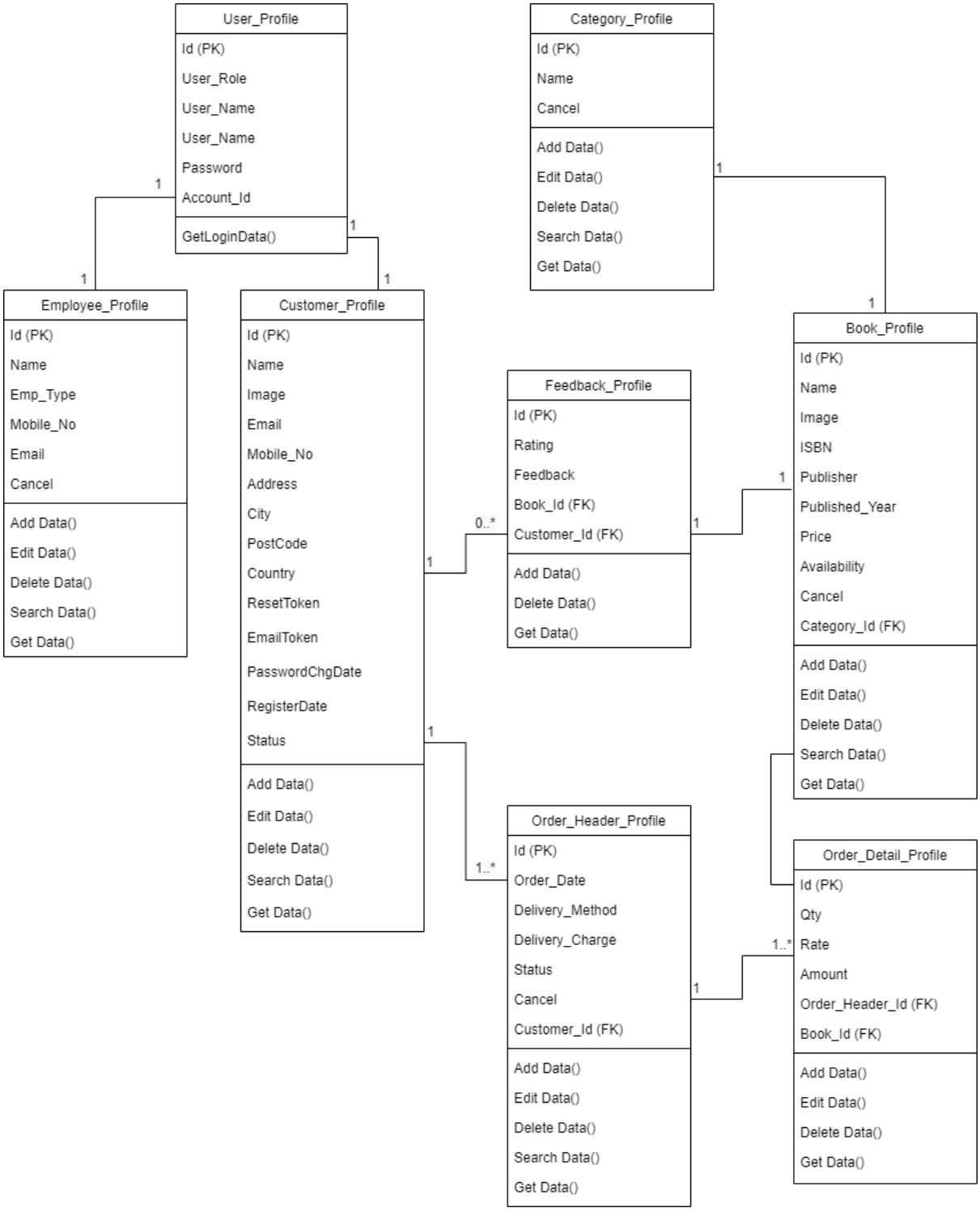


Figure 3 Class diagram

3.4 ER Diagram.

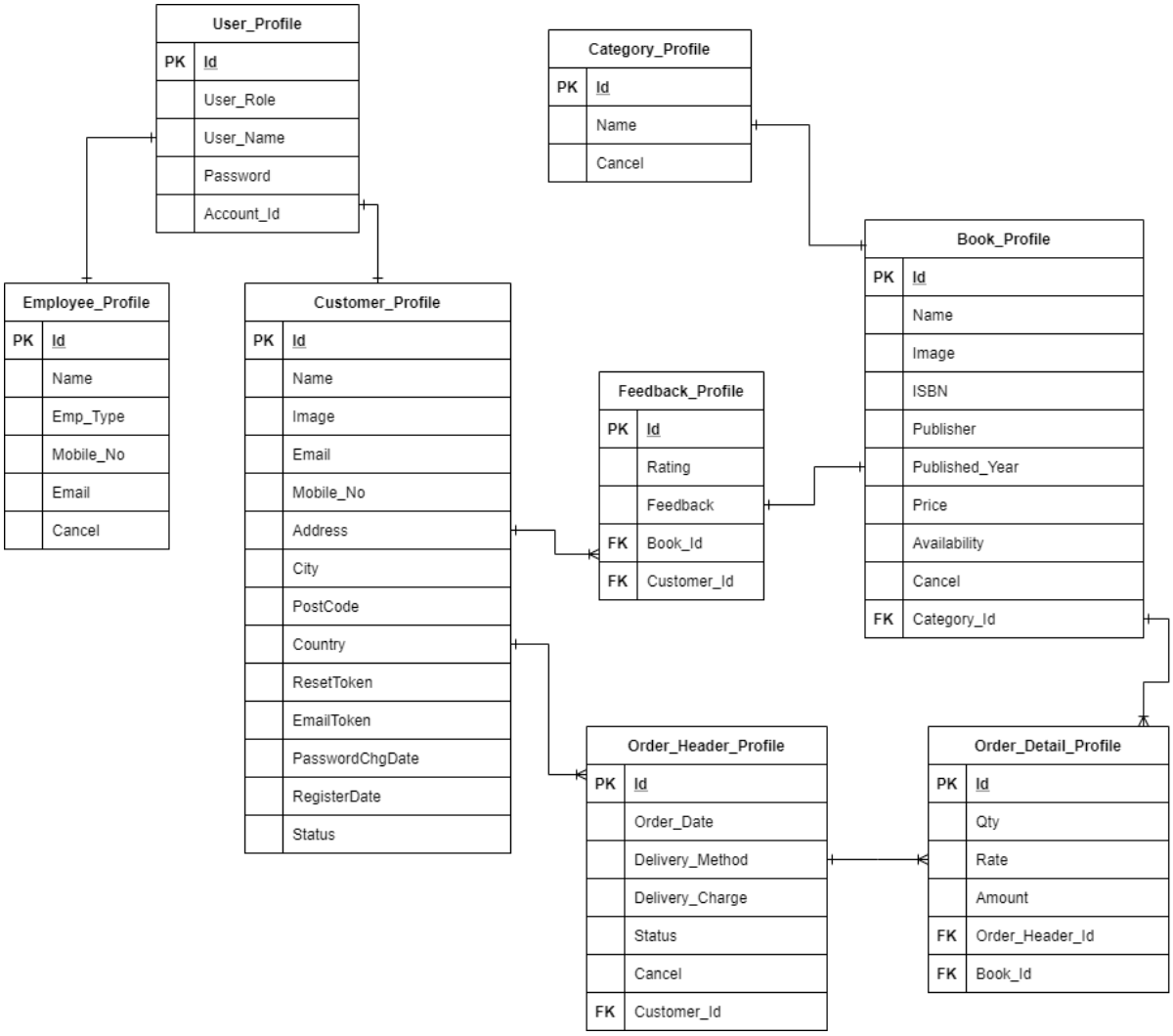


Figure 4 ER diagram

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

3.5 Database

SQL Server Management Studio 2005 was utilized in this project for database creation and design. From SQL Server to Azure Database, this is an integrated platform for administering any SQL infrastructure. SQL Server Management Studio 2005 is particularly useful for querying, designing, and managing databases and data warehouses, whether on a local computer or in the cloud. In the database system utilized in the project backend, the following tables were constructed. In SQL Server Management Studio 2005, this is how the database architecture looks:

3.5.1 Book_Profile Table

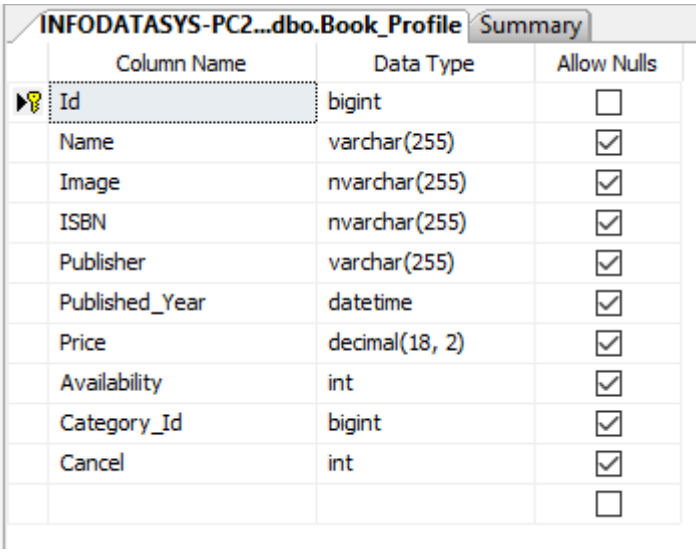


Figure 5 Book_Profile_Table

3.5.2 Category_Profile Table

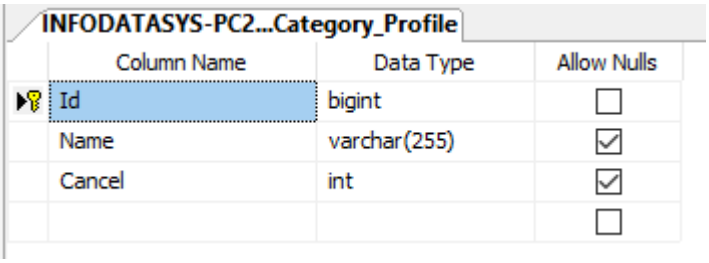


Figure 6 Category_Profile Table

3.5.3 Customer_Profile Table

INFODATASYS-PC2...Customer_Profile

Column Name	Data Type	Allow Nulls
Id	bigint	☐
Name	varchar(255)	☑
Image	nvarchar(255)	☑
Email	varchar(50)	☑
Mobile_No	varchar(50)	☑
Address	varchar(255)	☑
City	varchar(255)	☑
PostCode	varchar(20)	☑
Country	varchar(255)	☑
ResetToken	varchar(50)	☑
EmailToken	varchar(50)	☑
PasswordChgDate	datetime	☑
RegisterDate	datetime	☑
Status	int	☐

Figure 7 Customer_Profile Table

3.5.4 Employee_Profile Table

INFODATASYS-PC2...Employee_Profile

Column Name	Data Type	Allow Nulls
Id	bigint	☐
Name	varchar(255)	☑
Emp_Type	nvarchar(255)	☑
Mobile_No	nvarchar(15)	☑
Email	nvarchar(255)	☑
Cancel	int	☑
		☐

Figure 8 Employee_Profile Table

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

3.5.5 Feedback_Profile Table

INFODATASYS-PC2...Feedback_Profile

Column Name	Data Type	Allow Nulls
Id	bigint	☐
Rating	int	☑
Feedback	varchar(255)	☑
Book_Id	bigint	☑
Customer_Id	bigint	☑
		☐

Figure 9 Feedback_Profile Table

3.5.6 Order_Detail_Profile Table

INFODATASYS-PC2\...er_Detail_Profile

Column Name	Data Type	Allow Nulls
Id	bigint	☐
Qty	decimal(18, 3)	☑
Rate	decimal(18, 3)	☑
Amount	decimal(18, 3)	☑
Order_Header_Id	bigint	☑
Book_Id	bigint	☑
		☐

Figure 10 Order_Detail_Profile Table

3.5.7 Order_Header_Profile Table

INFODATASYS-PC2...r_Header_Profile

Column Name	Data Type	Allow Nulls
Id	bigint	☐
Order_Date	datetime	☑
Delivery_Method	varchar(50)	☑
Delivery_Charge	decimal(18, 2)	☑
Status	char(50)	☑
Cancel	int	☑
Customer_Id	bigint	☑
		☐

Figure 11 Order_Header_Profile Table

3.5.8 User_Profile Table

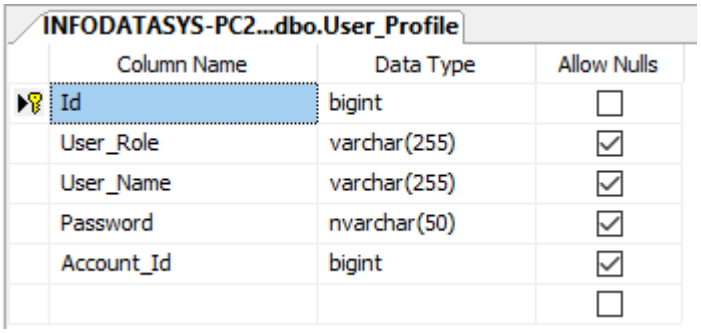


Figure 12 User_Profile Table

3.6 USER-INTERFACE.

This system's user interface must be simple and straightforward. Most essential, the ages must be simple to read, comprehend, and use. There should be no contrast difficulties and the color palette should be adequate to establish familiarity with the Company.

3.7 HARDWARE INTERFACE.

- There are no additional hardware interfaces required.
- Standard hardware and data connectivity resources will be used in the system.
- This includes, but is not limited to, the server/hosting site's general network connection, network server, and network management tools.

HARDWARE CONFIGURATION	
Processor	Intel Core i7 (3.2Mhz)
Ram	4 GB | 8 GB
Hard Disk	1 TB
LCD Monitor	15” Color Monitor
Keyboard	QWERTY

Mouse	PS/2 | USB
CD | DVD | Blu-Ray Combo Drive	Optional (Recommended)
Printer	Ink-jet | Laser

Table 1 - Hardware Configuration

3.4 SOFTWARE INTERFACE.

SOFTWARE CONFIGURATION	
Operating System	Windows 10 | Windows 11
Language	High Level Programming Language ASP.Net
Database	Microsoft SQL server

Table 2 - Software Configuration

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

04) IMPLEMENTATION.

The coding takes place in this chapter. Some of the coding samples are added below:

This system is developed by using ASP.NET MVC architecture. MVC stands for Model, View, and Controller. MVC separates this Ebook application into three components - Model, View, and Controller.

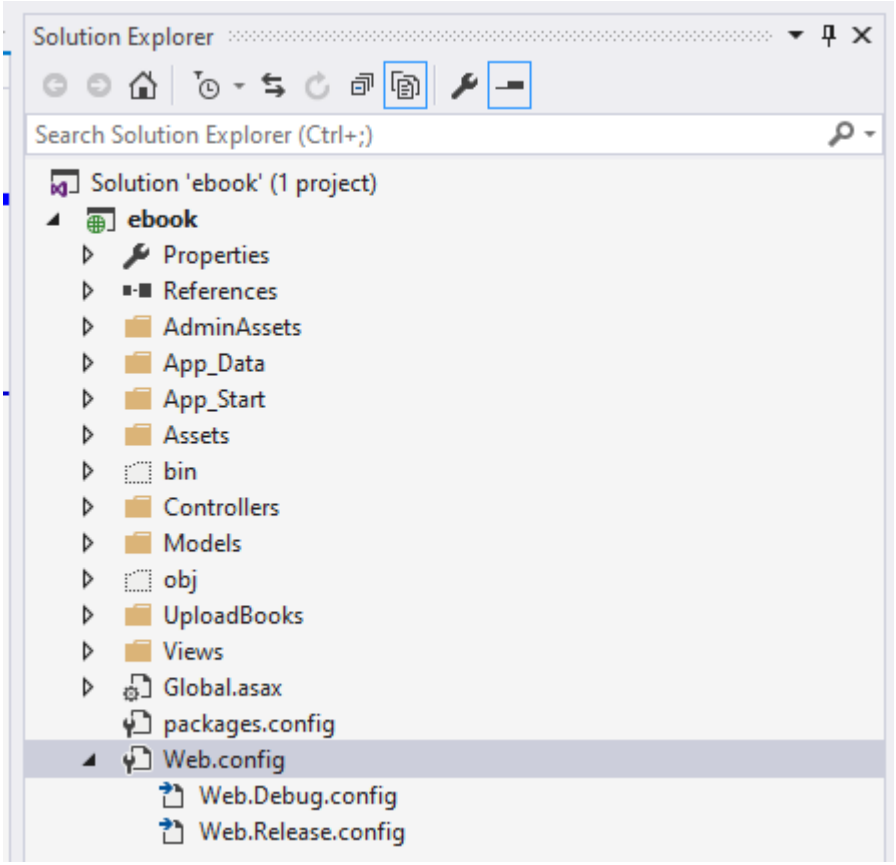


Figure 13 Ebook solution structure

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

4.1 Model Coding Samples.

The data's shape is represented by the model. A model is described using a class in C#. Data from the database is stored in model objects.

BookModel

```
namespace ebook.Models
{
    -references
    public class BookModel
    {
        -references
        public string Id { get; set; }
        -references
        public string Name { get; set; }
        -references
        public string Image { get; set; }
        -references
        public string ISBN { get; set; }
        -references
        public string Publisher { get; set; }
        -references
        public string Published_Year { get; set; }
        -references
        public string Price { get; set; }
        -references
        public string Availability { get; set; }
        -references
        public string Category_Id { get; set; }
        -references
        public string Category { get; set; }
        -references
        public string Cancel { get; set; }
    }
}
```

Figure 14 BookModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

CartModel

```
namespace ebook.Models
{
    0 references
    public class CartModel
    {
        0 references
        public string BookId { get; set; }
        0 references
        public string BookName { get; set; }
        0 references
        public string Image { get; set; }
        0 references
        public string Category { get; set; }
        0 references
        public string Price { get; set; }
        0 references
        public string Qty { get; set; }
        0 references
        public string Amount { get; set; }
    }
}
```

Figure 15 CartModel

CategoryModel

```
namespace ebook.Models
{
    -references
    public class CategoryModel
    {
        -references
        public string Id { get; set; }
        -references
        public string Name { get; set; }
    }
}
```

Figure 16 CategoryModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

CustomerModel

```
namespace ebook.Models
{
    -references
    public class CustomerModel
    {
        -references
        public string Id { get; set; }
        -references
        public string Name { get; set; }
        -references
        public string Image { get; set; }
        -references
        public string Email { get; set; }
        -references
        public string Mobile { get; set; }
        -references
        public string Address { get; set; }
        -references
        public string City { get; set; }
        -references
        public string PostCode { get; set; }
        -references
        public string Country { get; set; }
        -references
        public string ResetToken { get; set; }
        -references
        public string EmailToken { get; set; }
        -references
        public string PasswordChgDate { get; set; }
        -references
        public string RegisterDate { get; set; }
        -references
        public string Status { get; set; }
    }
}
```

Figure 17 CustomerModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

EmployeeModel

```
namespace ebook.Models
{
    -references
    public class EmployeeModel
    {
        -references
        public string Id { get; set; }
        -references
        public string Name { get; set; }
        -references
        public string Mobile { get; set; }
        -references
        public string Email { get; set; }
        -references
        public string Cancel { get; set; }
    }
}
```

Figure 18 EmployeeModel

FeedbackModel

```
namespace ebook.Models
{
    -references
    public class FeedbackModel
    {
        -references
        public string Id { get; set; }
        -references
        public string Rating { get; set; }
        -references
        public string Feedback { get; set; }
        -references
        public string Book_Id { get; set; }
        -references
        public string Customer_Id { get; set; }
        -references
        public string Customer { get; set; }
        -references
        public string Email { get; set; }
        -references
        public string Date { get; set; }
    }
}
```

Figure 19 FeedbackModel

OrderDetailModel

```
namespace ebook.Models
{
    0 references
    public class OrderDetailModel
    {
        0 references
        public string Id { get; set; }
        0 references
        public string Qty { get; set; }
        0 references
        public string Rate { get; set; }
        0 references
        public string Image { get; set; }
        0 references
        public string Amount { get; set; }
        0 references
        public string Order_Header_Id { get; set; }
        0 references
        public string Book_Id { get; set; }
        0 references
        public string Book { get; set; }
        0 references
        public string Category { get; set; }
        0 references
        public string OrderDate { get; set; }

    }
}
```

Figure 20 OrderDetailModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

OrderHeaderModel

```
namespace ebook.Models
{
    0 references
    public class OrderHeaderModel
    {
        0 references
        public string Id { get; set; }
        0 references
        public string Order_Date { get; set; }
        0 references
        public string Delivery_Method { get; set; }
        0 references
        public string Delivery_Charge { get; set; }
        0 references
        public string Status { get; set; }
        0 references
        public string Customer_Id { get; set; }
        0 references
        public string Customer { get; set; }
        0 references
        public string Cancel { get; set; }
    }
}
```

Figure 21 OrderHeaderModel

UserModel

```
namespace ebook.Models
{
    0 references
    public class UserModel
    {
        0 references
        public string Name { get; set; }
        0 references
        public string UserName { get; set; }
        0 references
        public string UserRole { get; set; }
        0 references
        public string Password { get; set; }
        0 references
        public string AccountId { get; set; }
        0 references
        public string Status { get; set; }
    }
}
```

Figure 22 UserModel

4.2 View Coding Samples.

In MVC, a view is a user interface. View gives the user access to model data and allows them to edit it. In ASP.NET MVC, the view is composed of HTML, CSS, and a specific syntax (Razor syntax) that facilitates communication with the model and controller.

About

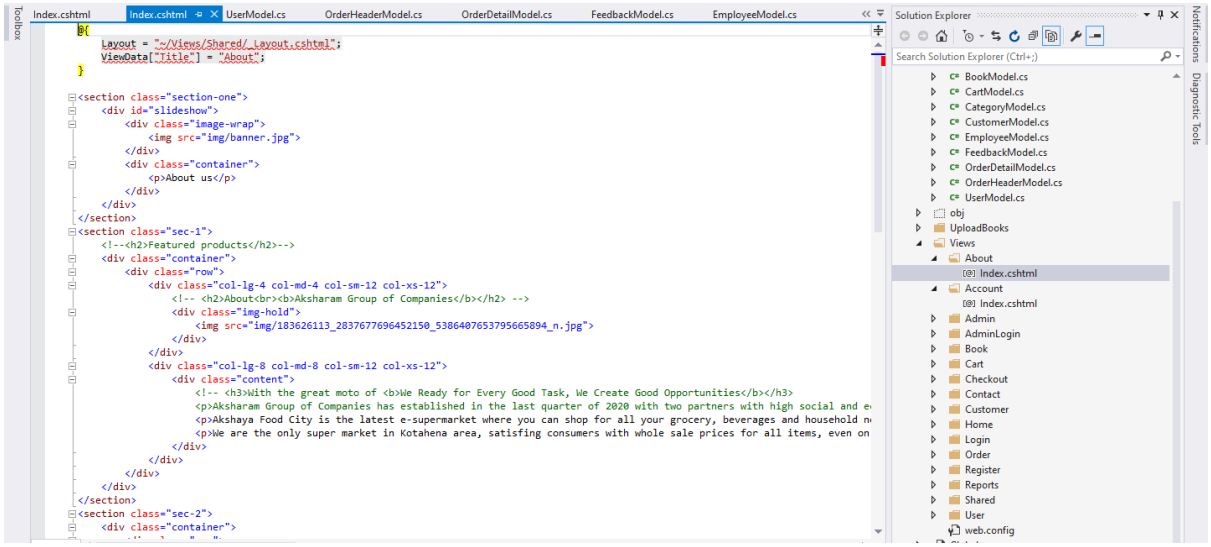


Figure 23 About

Account

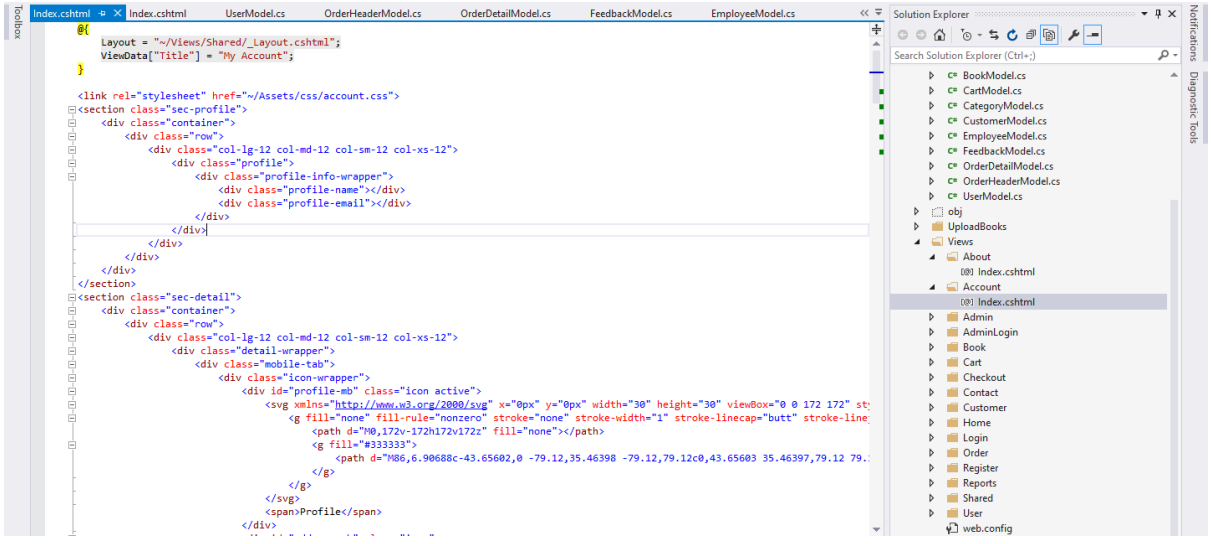


Figure 24 Account

Admin - Dashboard

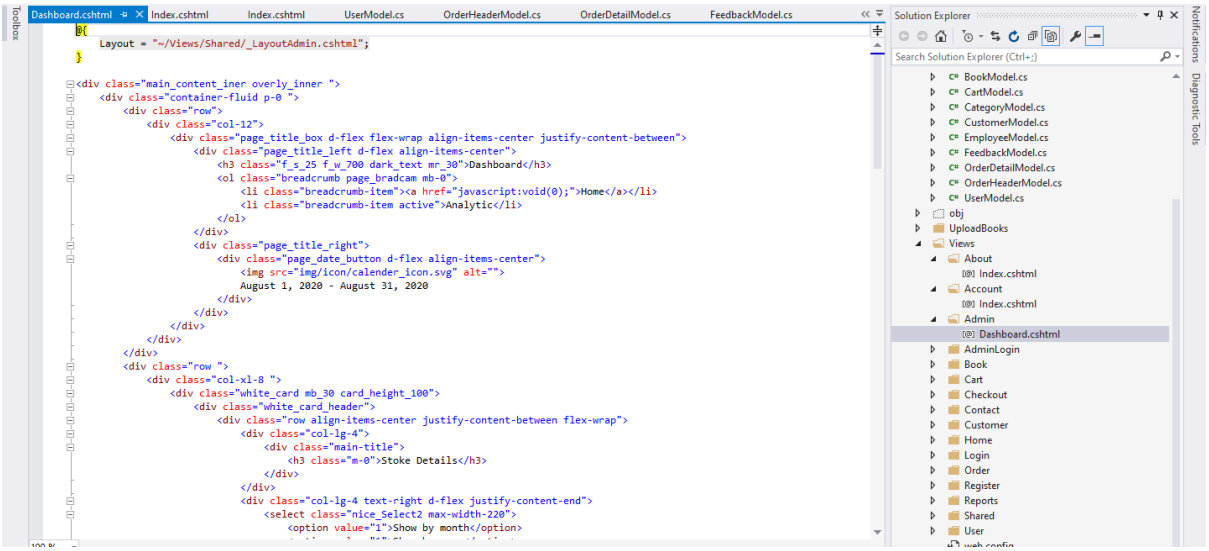


Figure 25 Admin – Dashboard

Admin Login

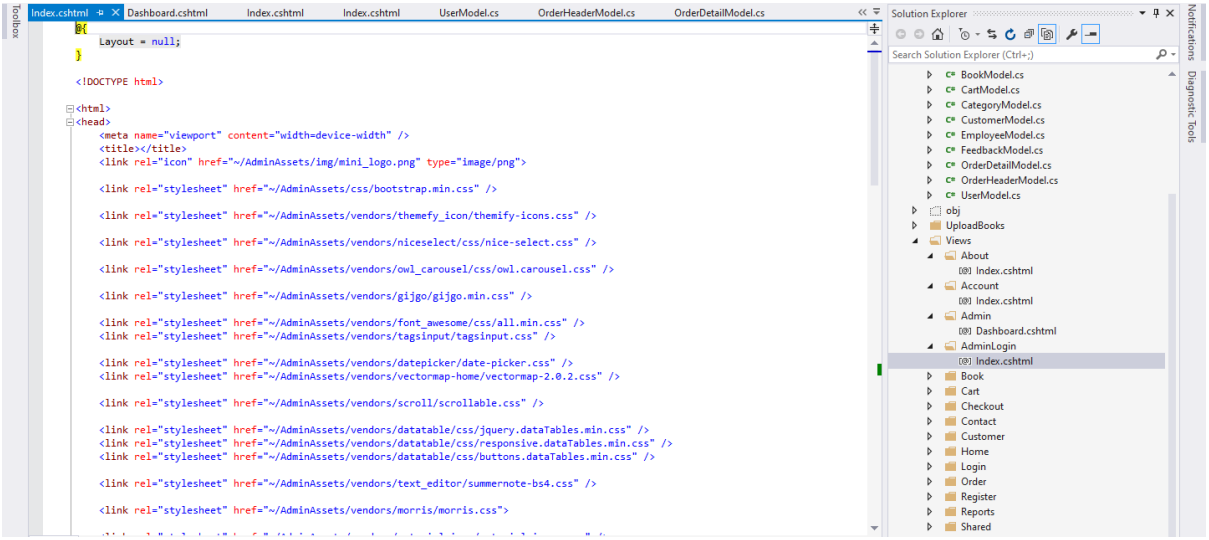


Figure 26 Admin Login

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Book

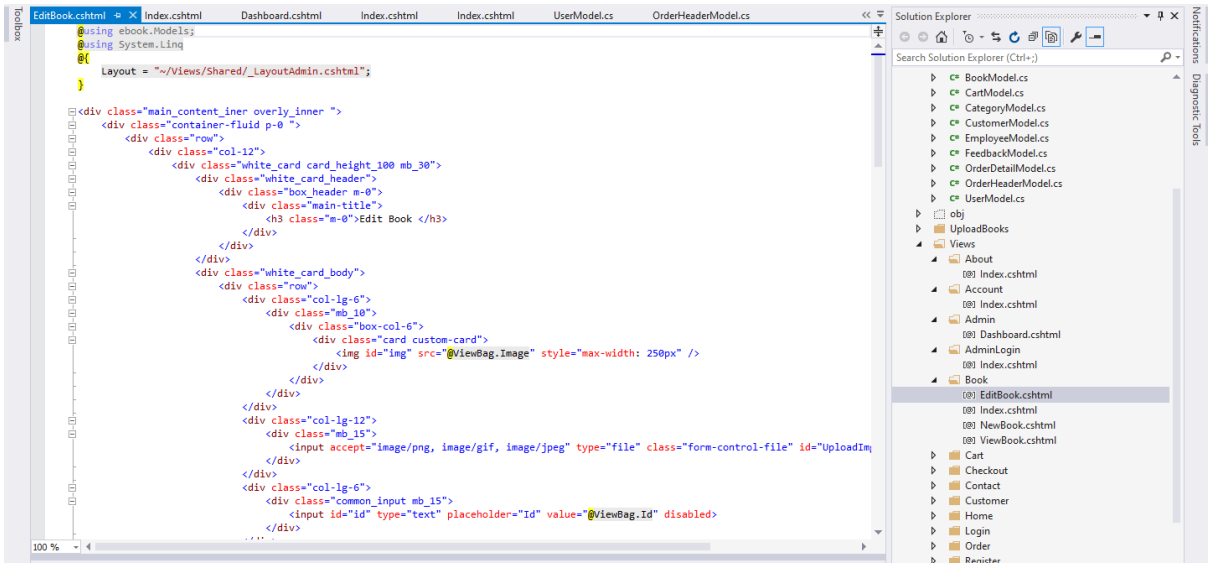


Figure 27 Book

Cart

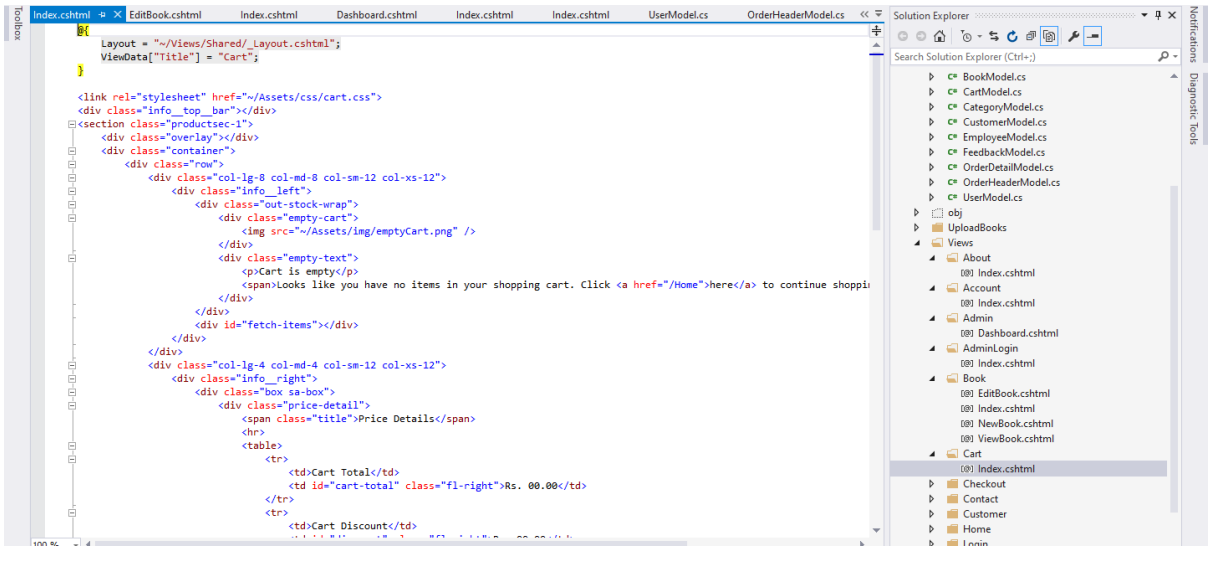


Figure 28 Cart

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Checkout

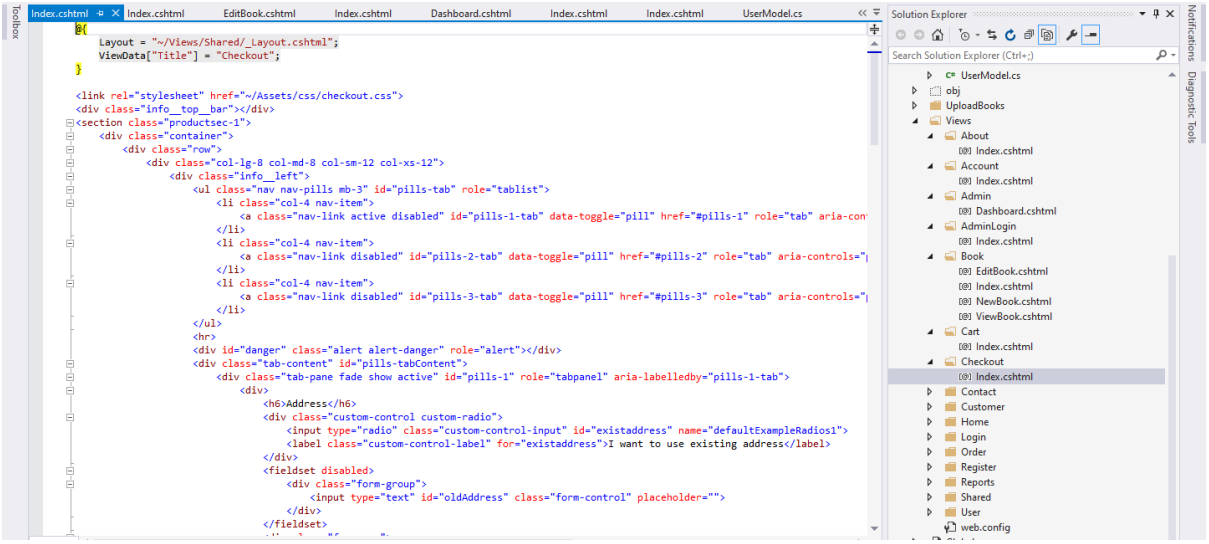


Figure 29 Checkout

Customer

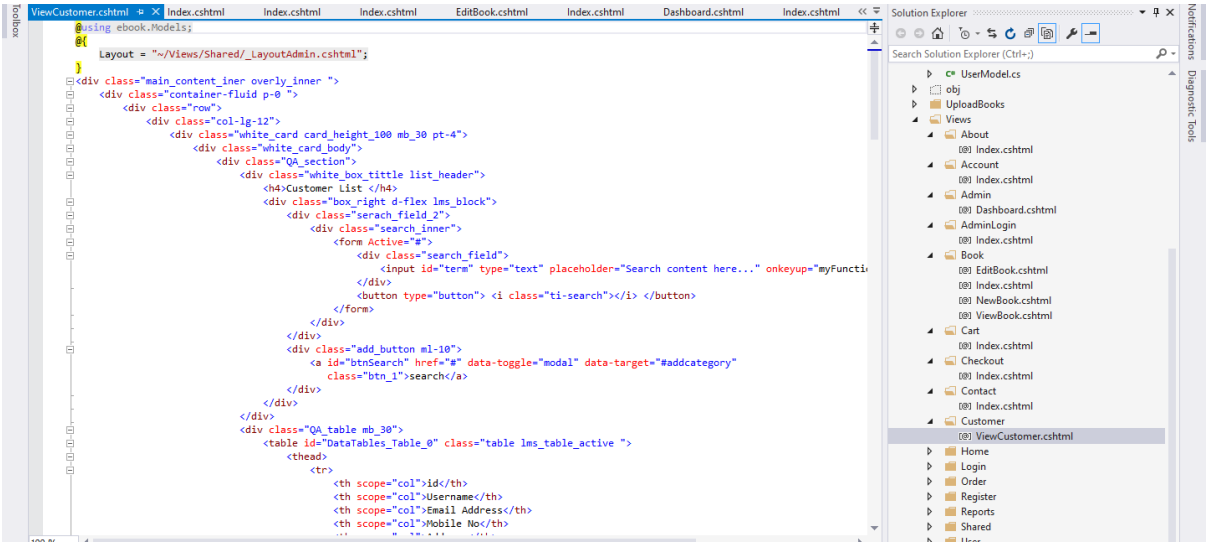


Figure 30 Customer

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Home

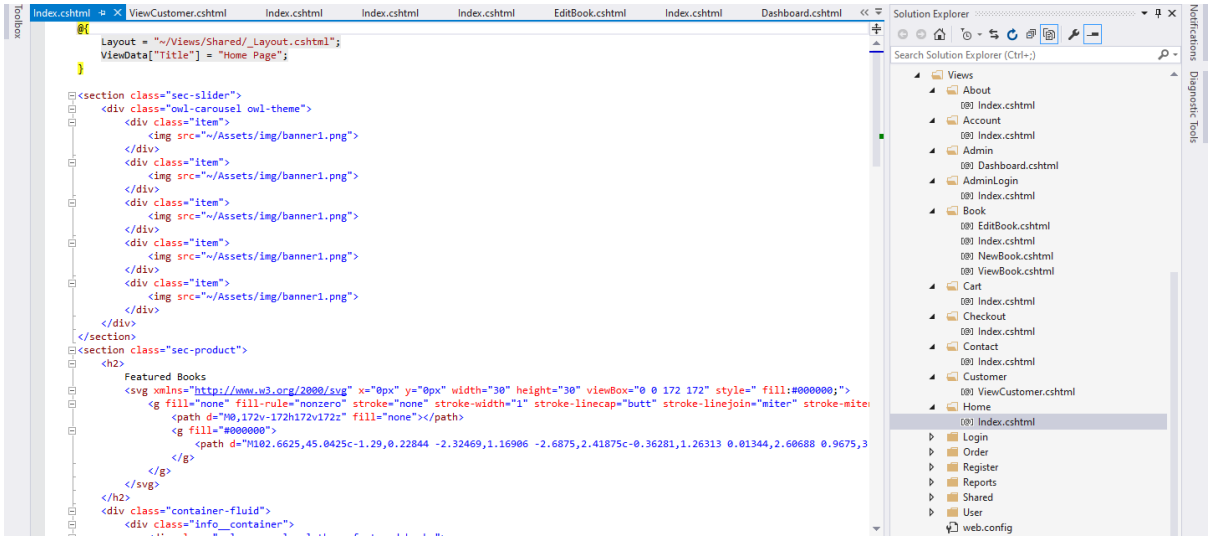


Figure 31 Home

Login

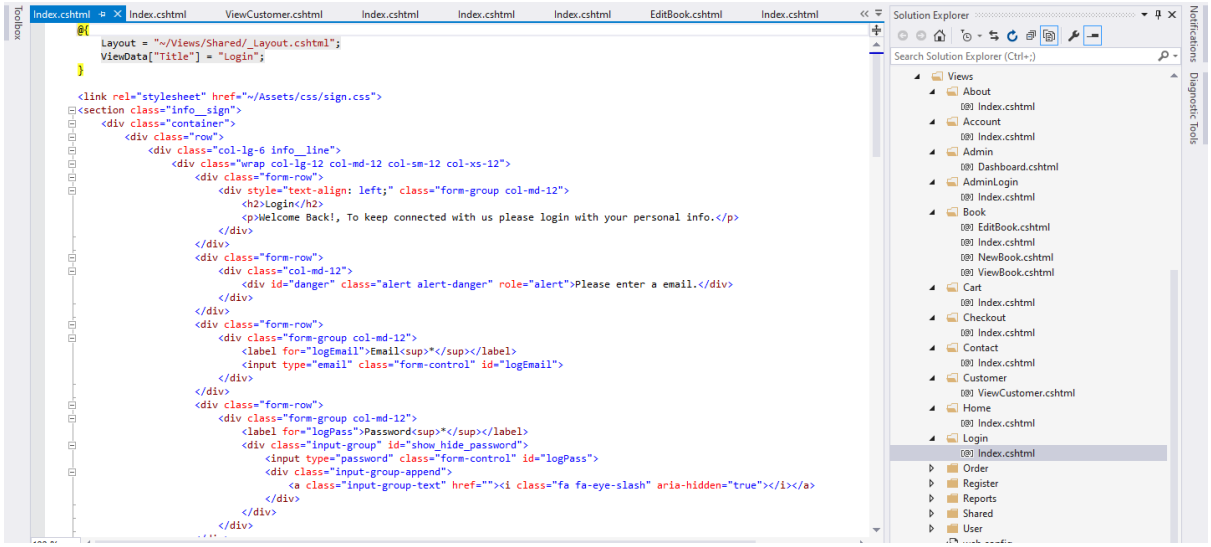


Figure 32 Login

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Order

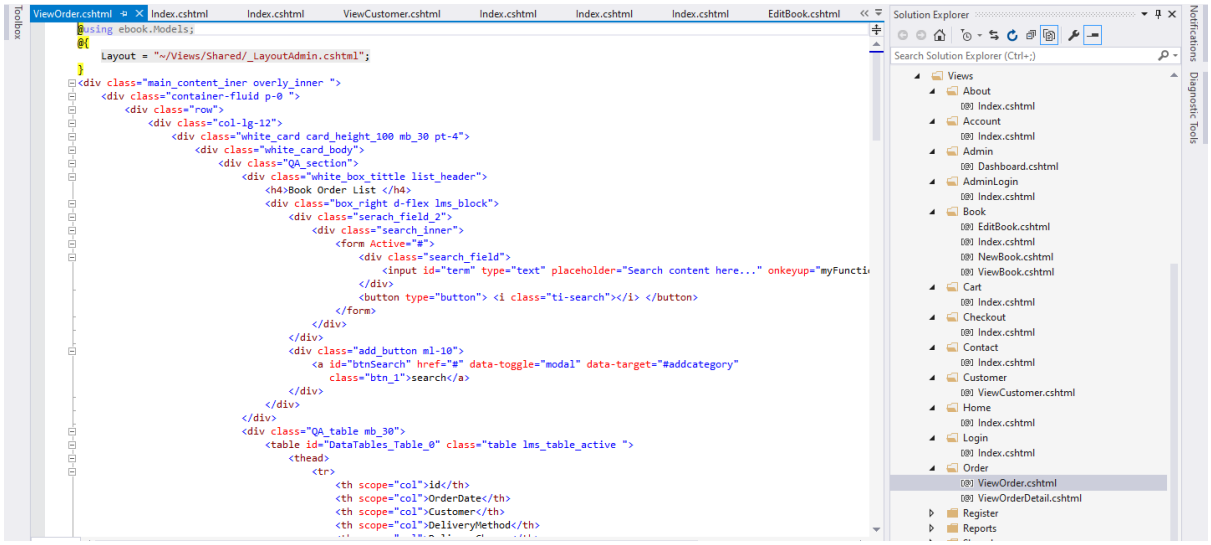


Figure 33 Order

Register

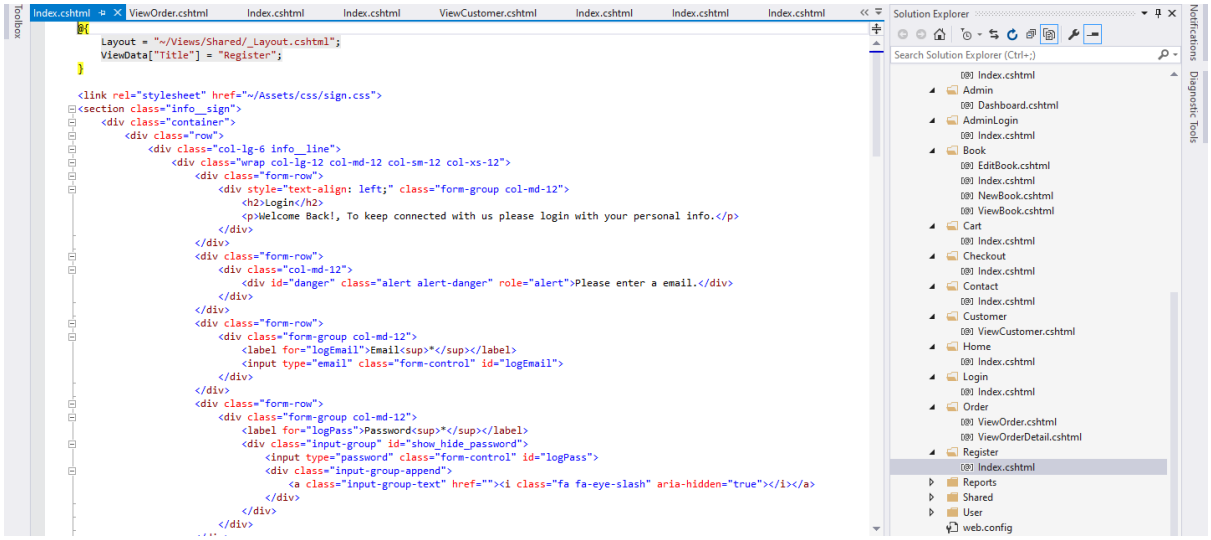


Figure 34 Register

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Report

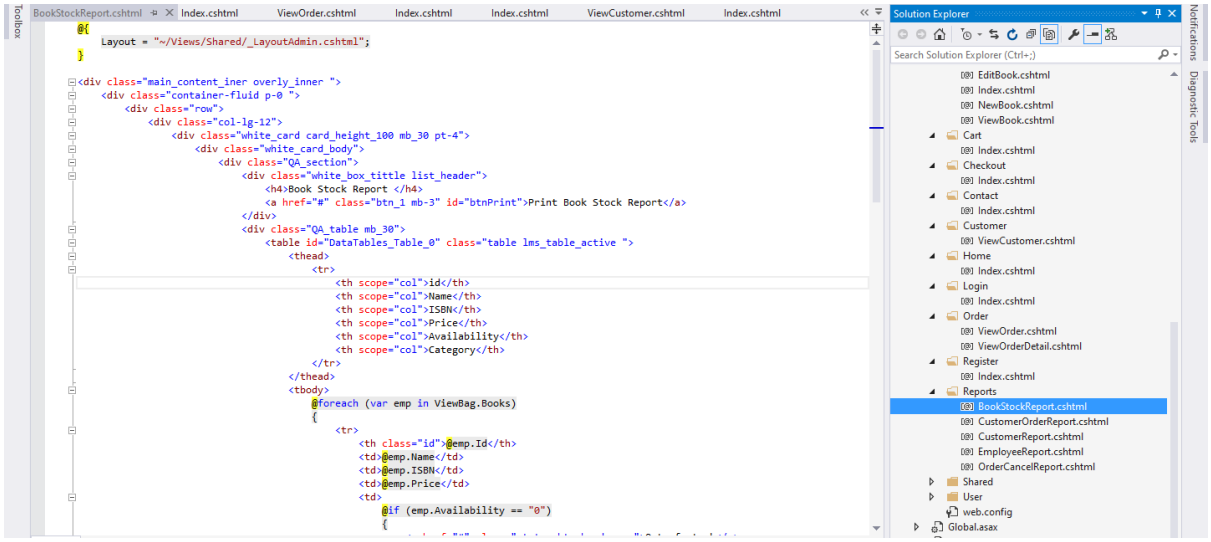


Figure 35 Report

Shared (This folder contains the master view file for admin and customer sites)

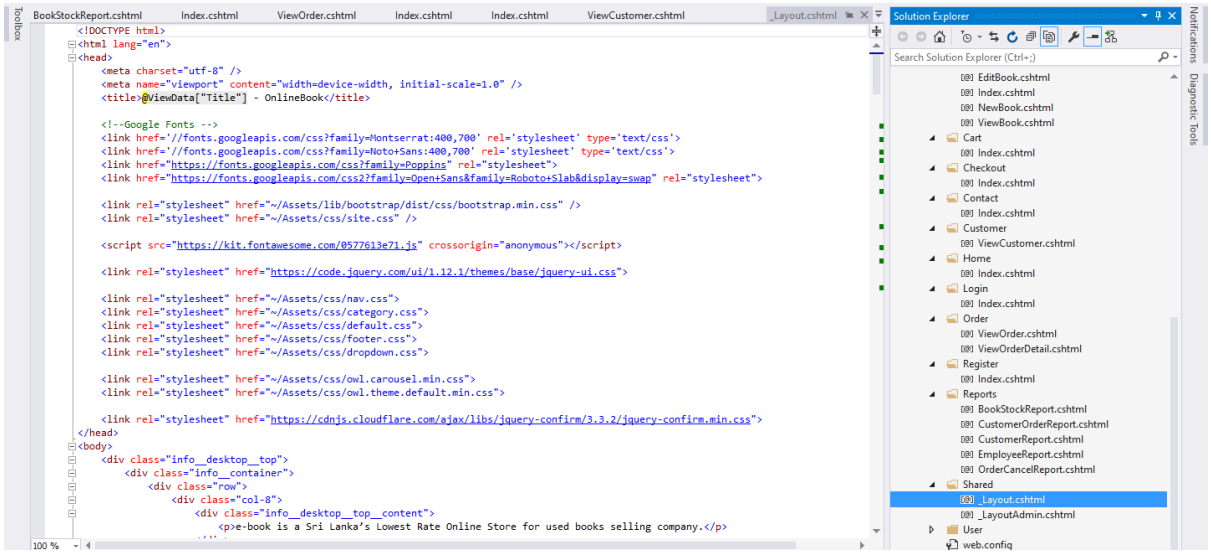


Figure 36 Shared

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

User

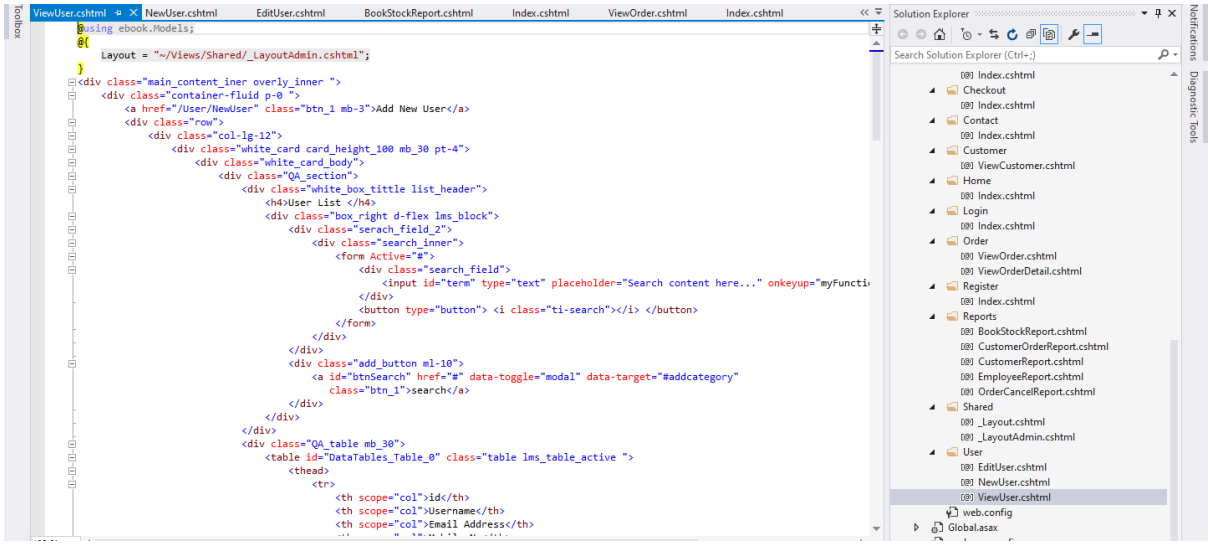


Figure 37 User

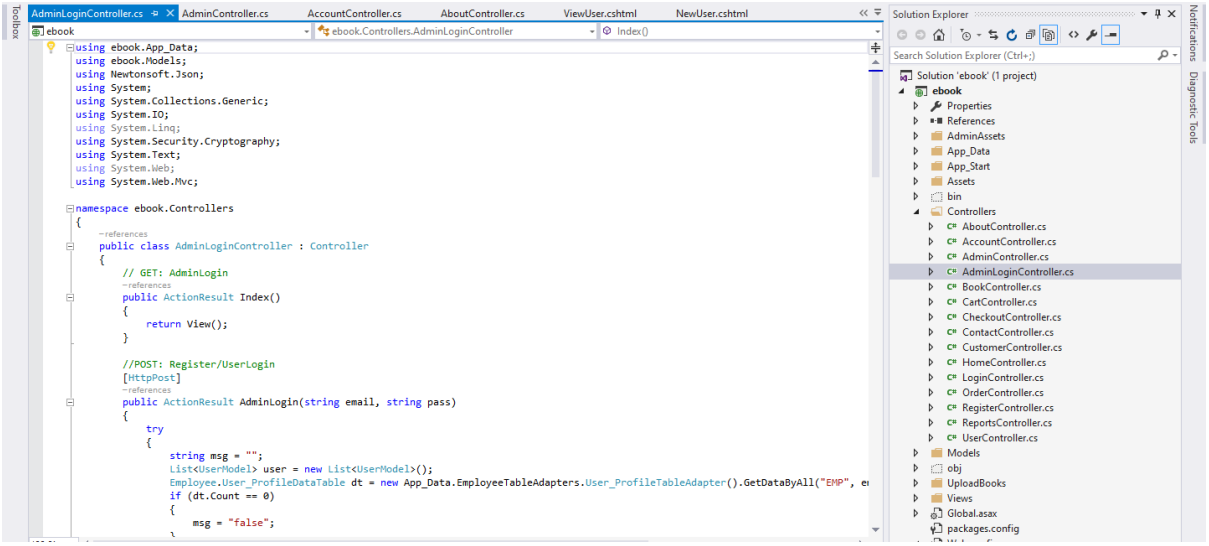
4.3 Controller Coding Samples.

The user's request is handled by the controller. The user typically uses the view to issue an HTTP request, which is processed by the controller. As a response, the controller processes the request and returns the appropriate view.

AccountController

Figure 38 AccountController

AdminLoginController



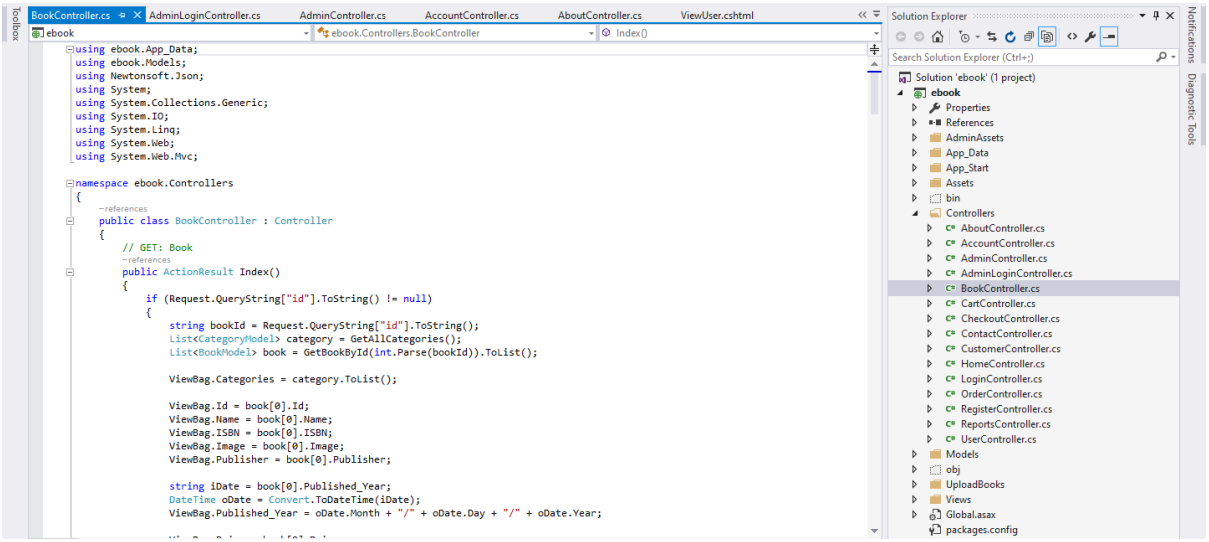

```
using ebook.App_Data;
using ebook.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Web.Mvc;

namespace ebook.Controllers
{
    public class AdminLoginController : Controller
    {
        // GET: AdminLogin
        public ActionResult Index()
        {
            return View();
        }

        //POST: Register/UserLogin
        [HttpPost]
        public ActionResult AdminLogin(string email, string pass)
        {
            try
            {
                string msg = "";
                List<UserModel> user = new List<UserModel>();
                Employee.User_ProfileDataTable dt = new App_Data.EmployeeTableAdapters.User_ProfileTableAdapter().GetDataByAll("EMP", e
                if (dt.Count == 0)
                {
                    msg = "false";
```

Figure 39 AdminLoginController

BookController




```
using ebook.App_Data;
using ebook.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace ebook.Controllers
{
    public class BookController : Controller
    {
        // GET: Book
        public ActionResult Index()
        {
            if (Request.QueryString["id"].ToString() != null)
            {
                string bookId = Request.QueryString["id"].ToString();
                List<CategoryModel> category = GetAllCategories();
                List<BookModel> book = GetBookById(int.Parse(bookId)).ToList();

                ViewBag.Categories = category.ToList();

                ViewBag.Id = book[0].Id;
                ViewBag.Name = book[0].Name;
                ViewBag.ISBN = book[0].ISBN;
                ViewBag.Image = book[0].Image;
                ViewBag.Publisher = book[0].Publisher;

                string iDate = book[0].Published_Year;
                DateTime oDate = Convert.ToDateTime(iDate);
                ViewBag.Published_Year = oDate.Month + "/" + oDate.Day + "/" + oDate.Year;
```

Figure 40 BookController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

CartController

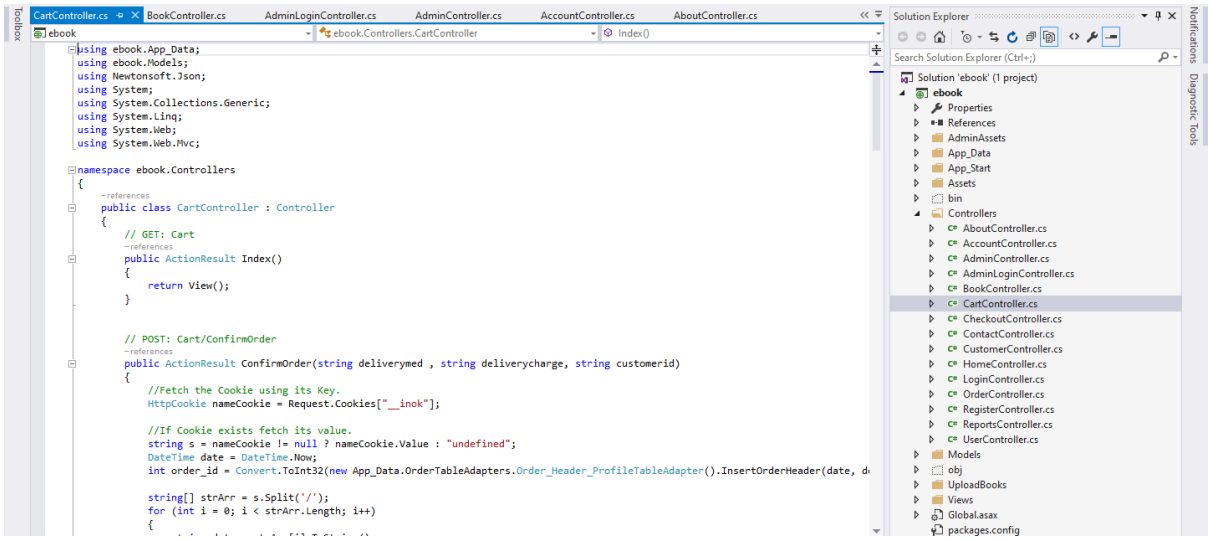


Figure 41 CartController

CustomerController

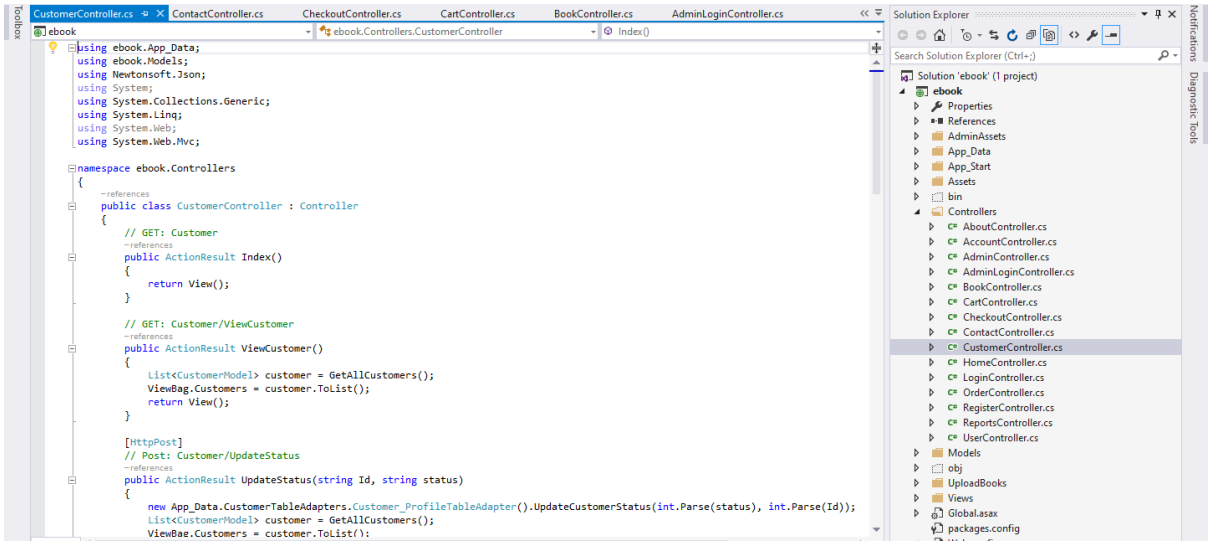


Figure 42 CustomerController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

HomeController

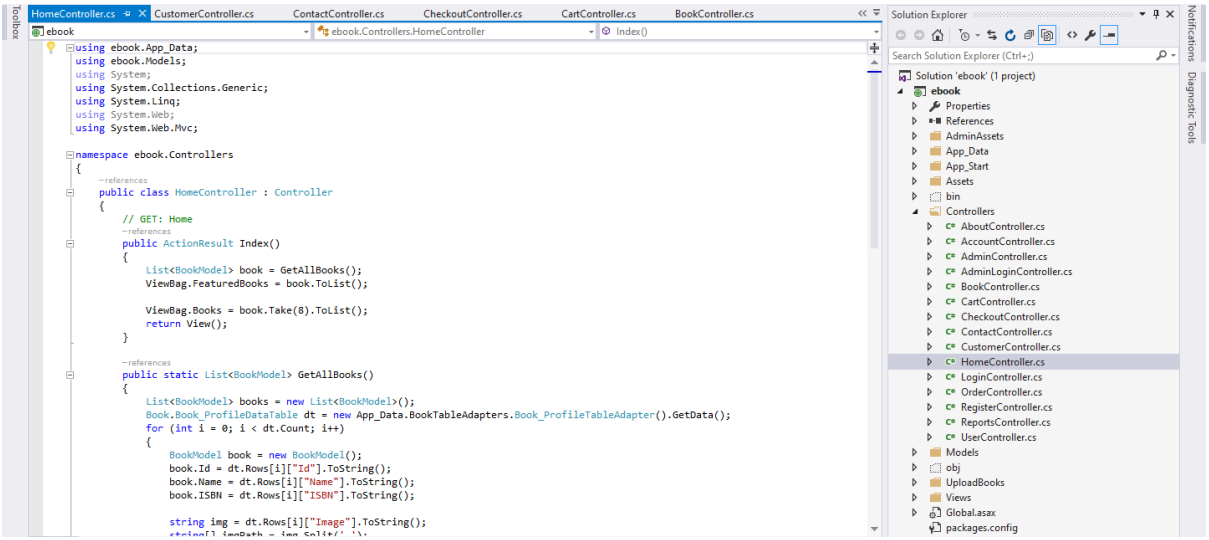


Figure 43 HomeController

OrderController

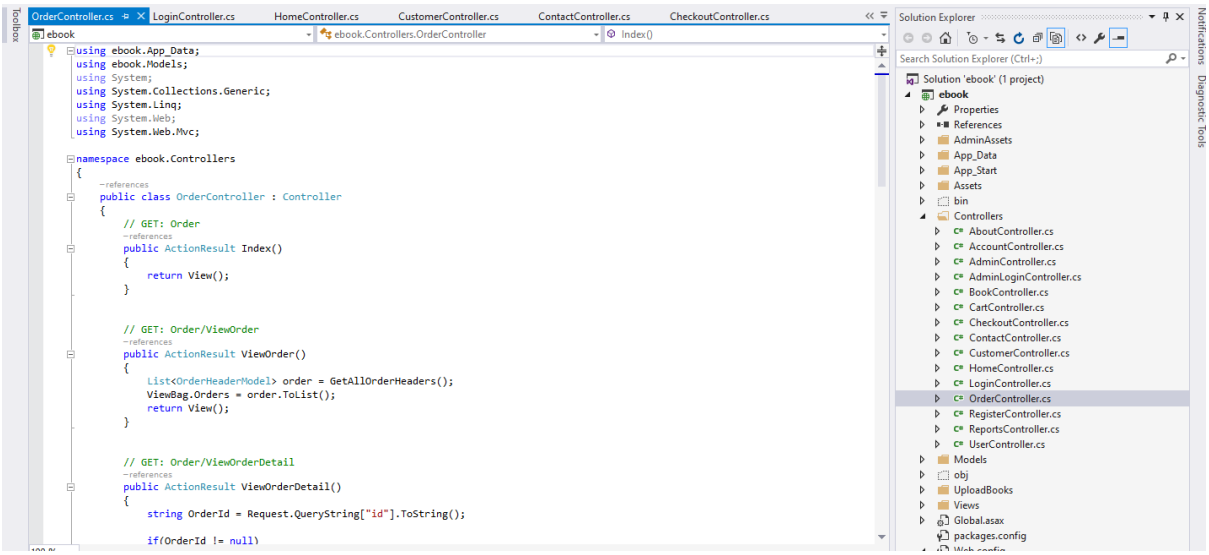


Figure 44 OrderController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

RegisterController

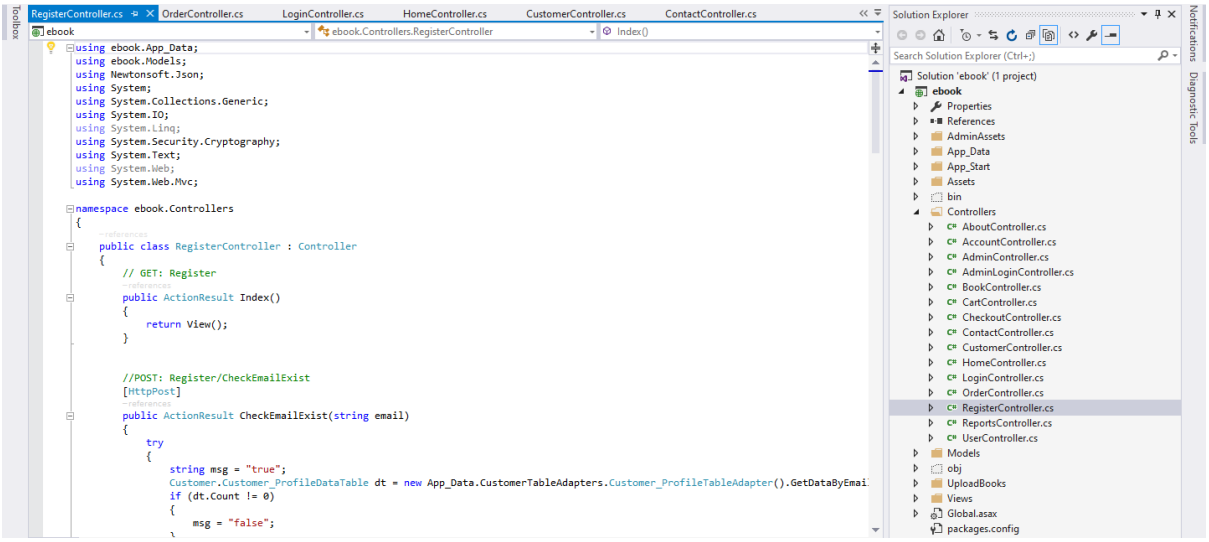


Figure 45 RegisterController

ReportsController

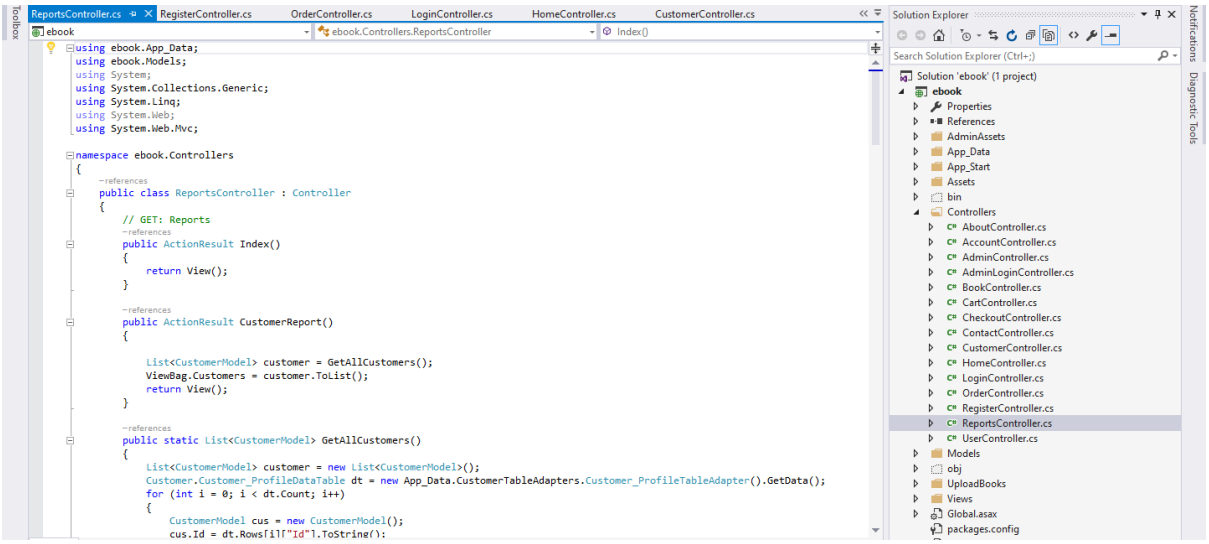
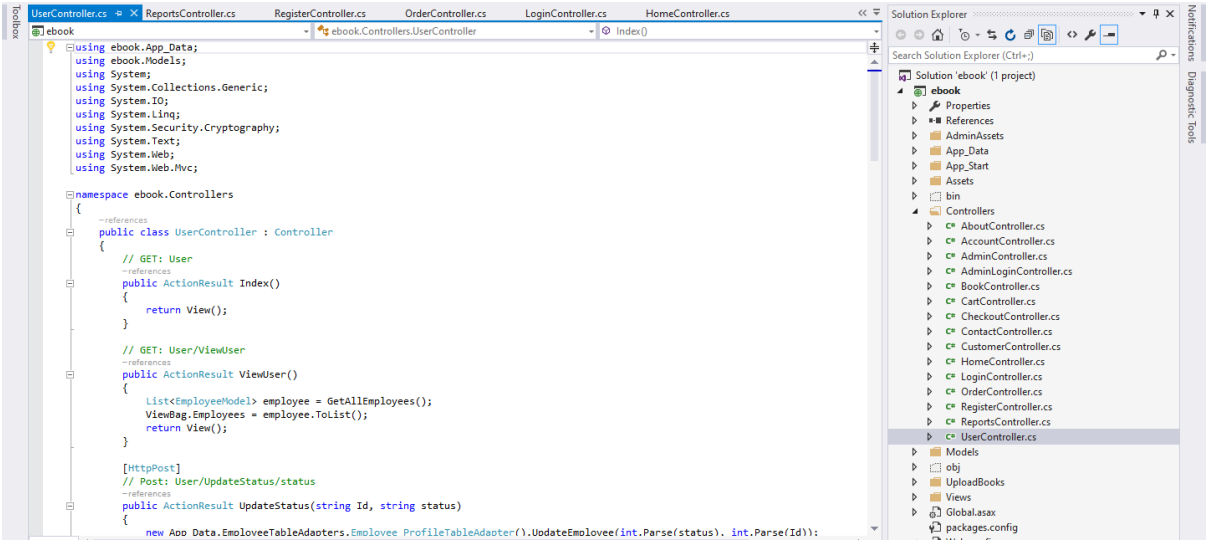


Figure 46 ReportsController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

UserController




```
using ebook.App_Data;
using ebook.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Web.Mvc;

namespace ebook.Controllers
{
    public class UserController : Controller
    {
        // GET: User
        public ActionResult Index()
        {
            return View();
        }

        // GET: User/ViewUser
        public ActionResult ViewUser()
        {
            List<EmployeeModel> employee = GetAllEmployees();
            ViewBag.Employees = employee.ToList();
            return View();
        }

        [HttpPost]
        // Post: User/UpdateStatus/status
        public ActionResult UpdateStatus(string Id, string status)
        {
            new App_Data.EmployeeTableAdapters.Employee_ProfileTableAdapter().UpdateEmployee(int.Parse(status), int.Parse(Id));
```

Figure 47 UserController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

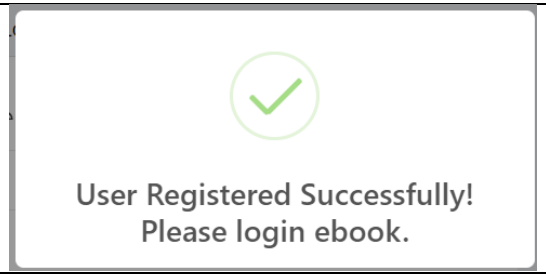
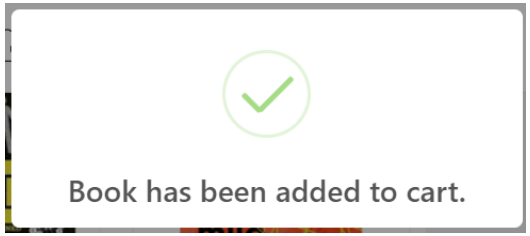
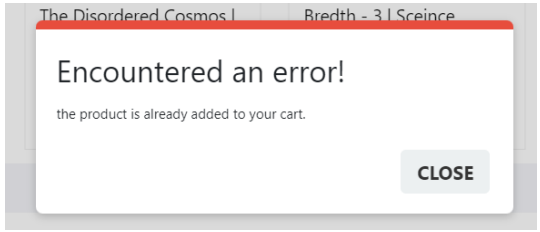
05) TESTING.

System testing is described as the process of determining whether or not a complete and fully integrated software product meets the required standards. The system is put to the test using the procedures listed below:

- **Black Box Testing:** The testing team does black box testing, which means that no knowledge of the code's underlying design is necessary, and each function must be tested.
- **White-box Testing:** White Box Testing is a software testing method that examines the product's fundamental structure, design, and code to check the input-output flow and improve planning, usability, and security.
- **User acceptance Testing:** User Acceptance Testing is a type of testing in which the software system is verified and accepted by the end-user or customer. The criteria are then created in order to receive feedback on the solution, and suitable revisions are made in response to the comments.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

5.1 Test Cases for Customer End (Ecommerce Site).

ID	Test Case	Expected Out Come	Actual Results	Status
1	Customer registering by filling all the required filled.	Should show register successfully message.		Pass
2	Customer adding a product to cart.	Should show book added to cart message.		Pass
3	Customer adding a product which already added to the cart.	Should show a alert saying the item already added to the cart.		Pass

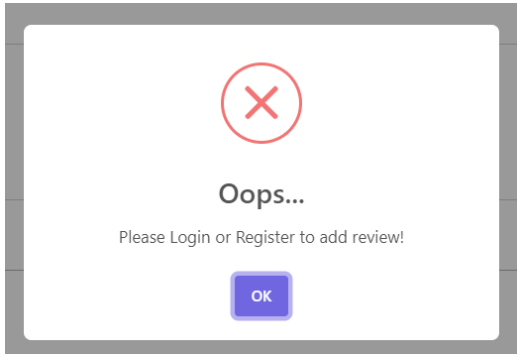
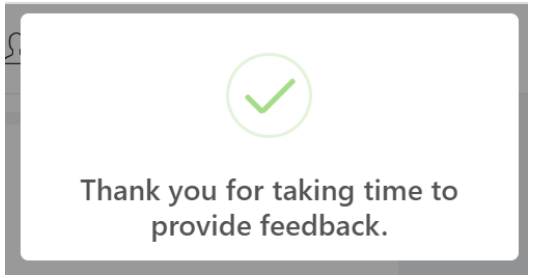
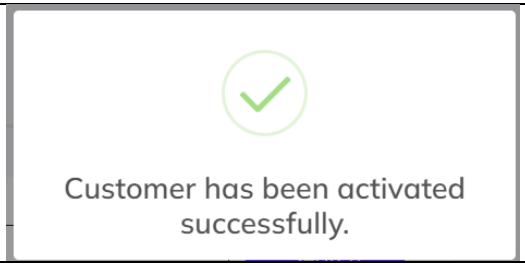
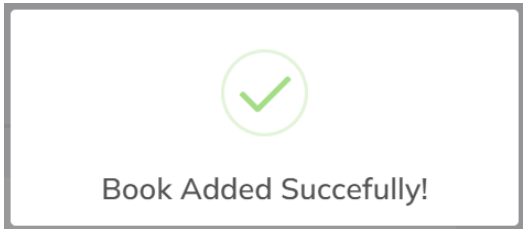
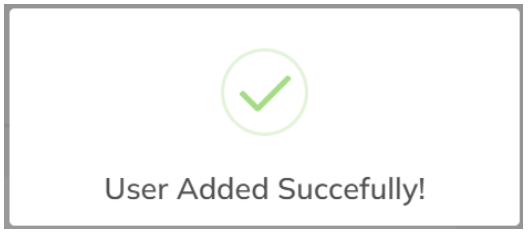
4	Customer giving book feedback without login to the system.	Should show alert saying should login to the system to provide review.		Pass
5	Customer giving book feedback after login to the system.	Should show feedback submitted successfully message.		Pass

Table 3 Test Cases for Customer End

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

5.1 Test Cases for Admin End (Backend System).

ID	Test Case	Expected Out Come	Actual Results	Status
1	User activating a customer account.	Should show customer activated successfully message.		Pass
2	User filling all the details of book and clicking add button.	Should show book successfully added message.		Pass
3	Admin filling all the details of new system user and clicking add button.	Should show user successfully added message.		Pass

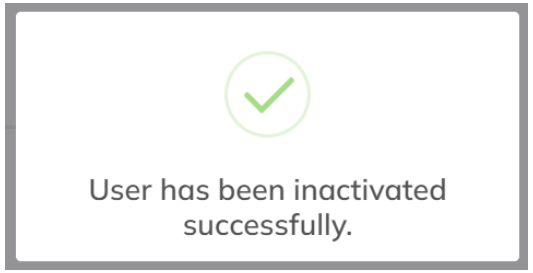
4	Admin clicking deactivate button for a particular user in user management.	Should show user deactivated successfully message.		Pass
5	User clicking generate report button in reports pages.	Should generate the selected report in pdf format.		Pass

Table 4 Test Cases for Admin

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

06) INSTALLATION GUIDE AND CONFIGURATION AND MANUAL OF SYSTEM.

6.1 Customer End (Ecommerce Store).

6.1.1 Home Page.

This is the page which user see first when they visit Ebook website. In this page the overall process of the Ebook ecommerce site is processed. All the new offers and book marketing details will be shown in banners. The home page contains the books and user can search the book in search bar.

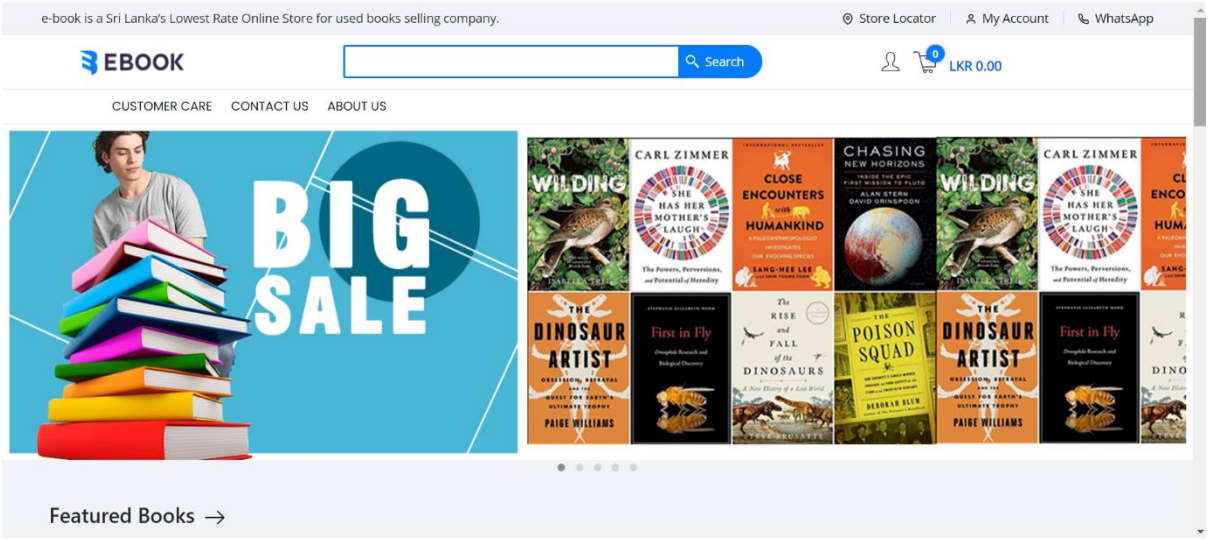


Figure 48 Home-1

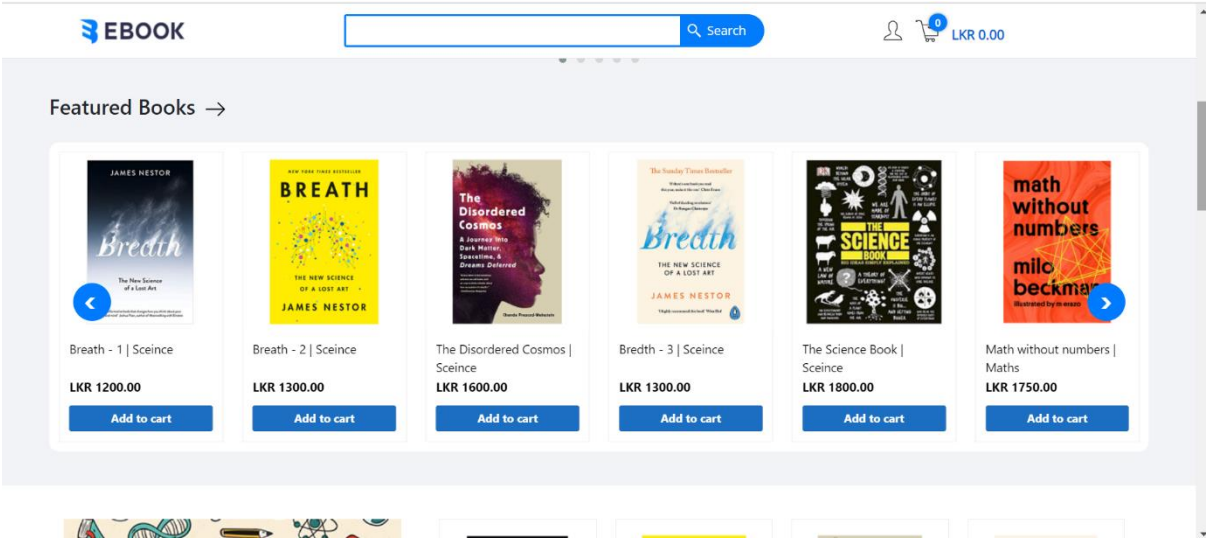


Figure 49 Home-2

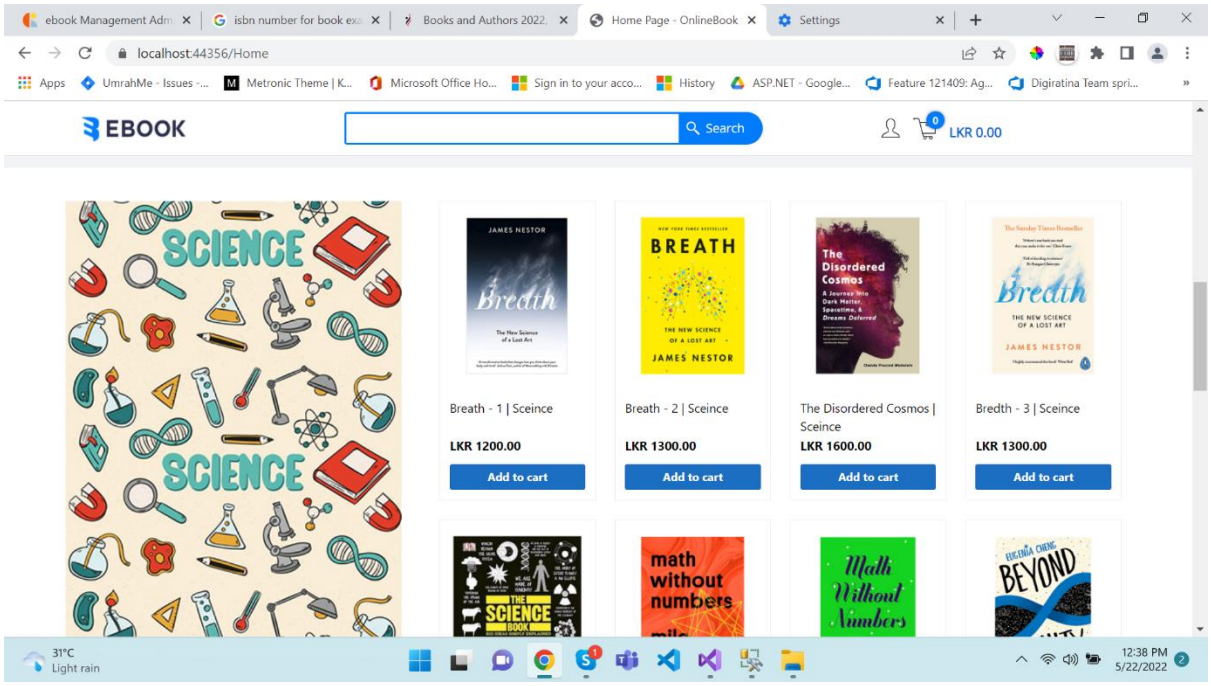


Figure 50 Home-3

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.1.2 Login & Register.

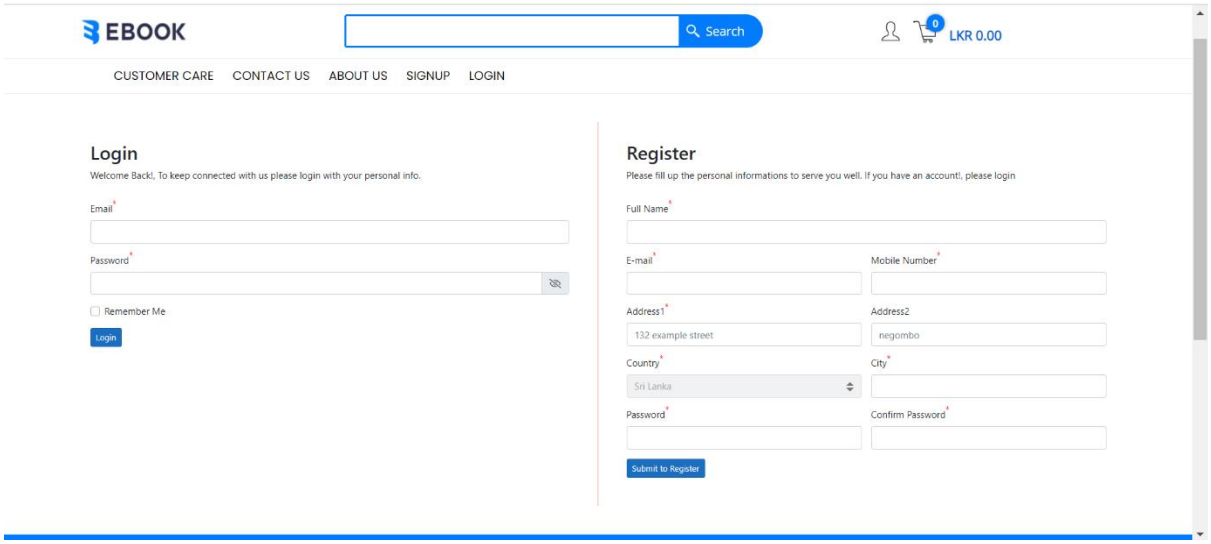


Figure 51 Login & Register

This interface displays the Customer Login section and Customer Registration section. To log in, customers must enter their Username and Password into the text fields and then click the "Login" button. If the client does not have a login, they need to register to access place orders in the system. To Register, customers must enter all the details need in the required field and click "Submit to Register" button. The password which customer entered will be encrypted and saved to the database.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.1.3 Cart.

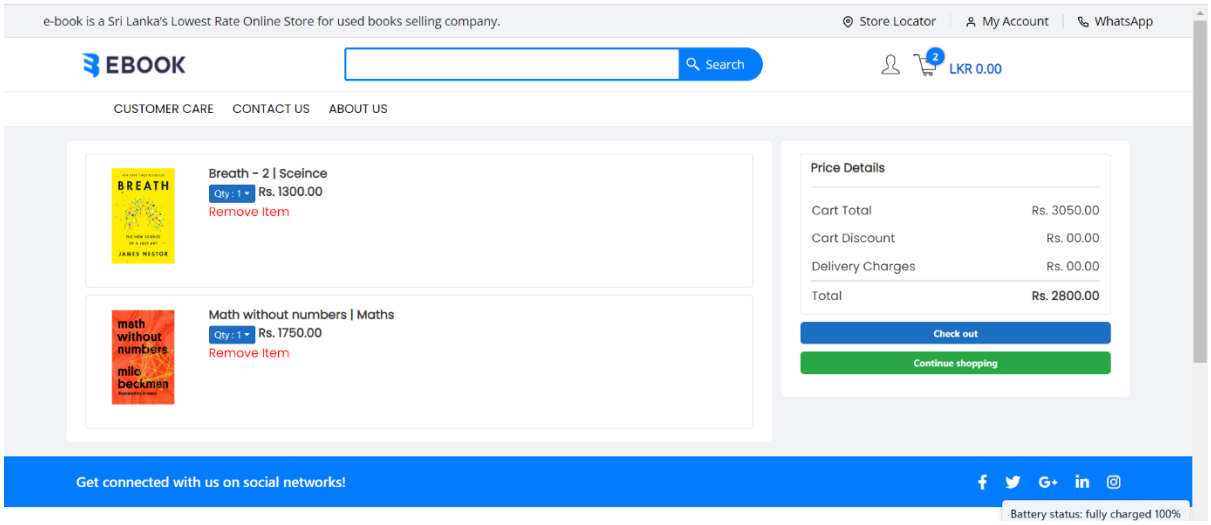


Figure 52 Cart

This interface displays all the items which customer added. Customer can remove a product from the added list by clicking "Remove Item" button. The right corner box will show the total amount of the product added to the cart.

6.1.4 Checkout

This interface displays the Checkout Page. All the item and the total cost will be displayed in this screen and once the customer click proceed button his order will be successfully added. Checkout process consist of three steps:

Step 1:

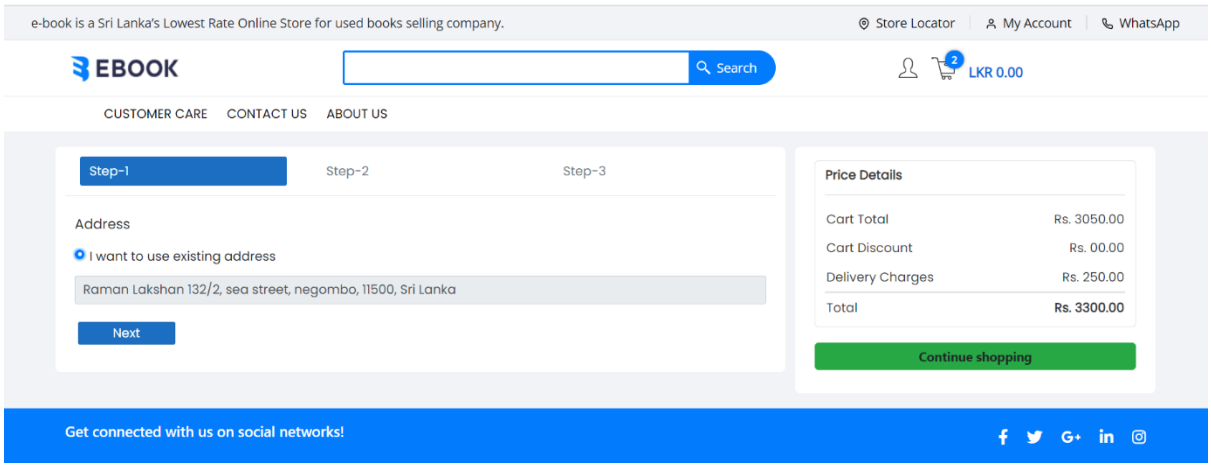


Figure 53 Checkout - Step1

In First step customer delivery address will be shown.

Step 2:

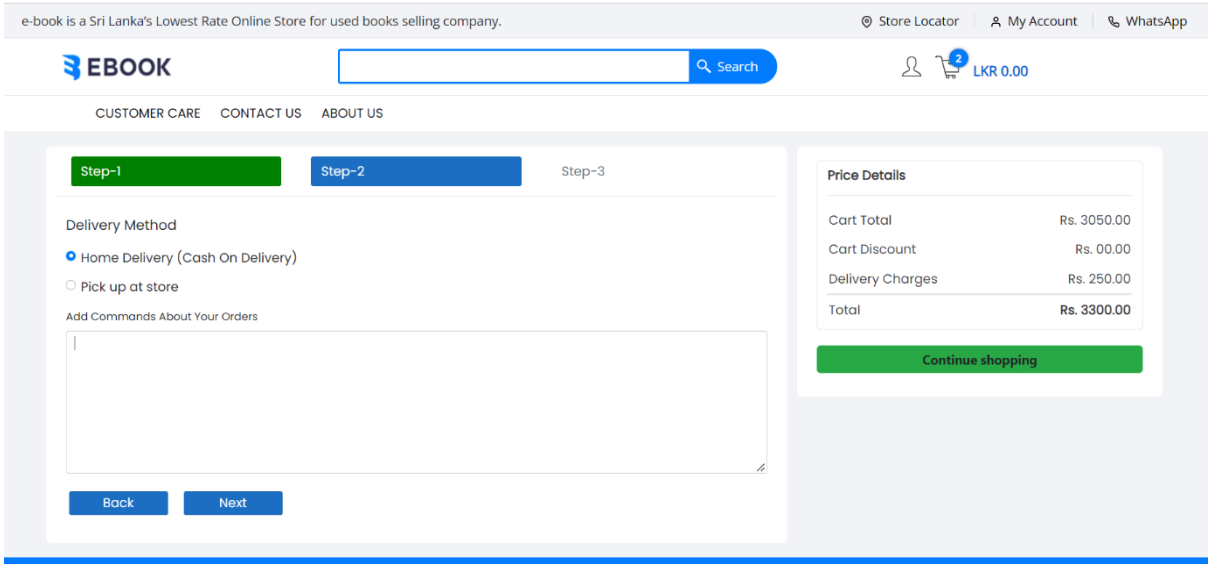


Figure 54 Checkout - Step2

In second step customer can select option to get the product deliver to them or pick up the product at store. And in here customer have given option to set a comment for the order if they wish.

Step 3:

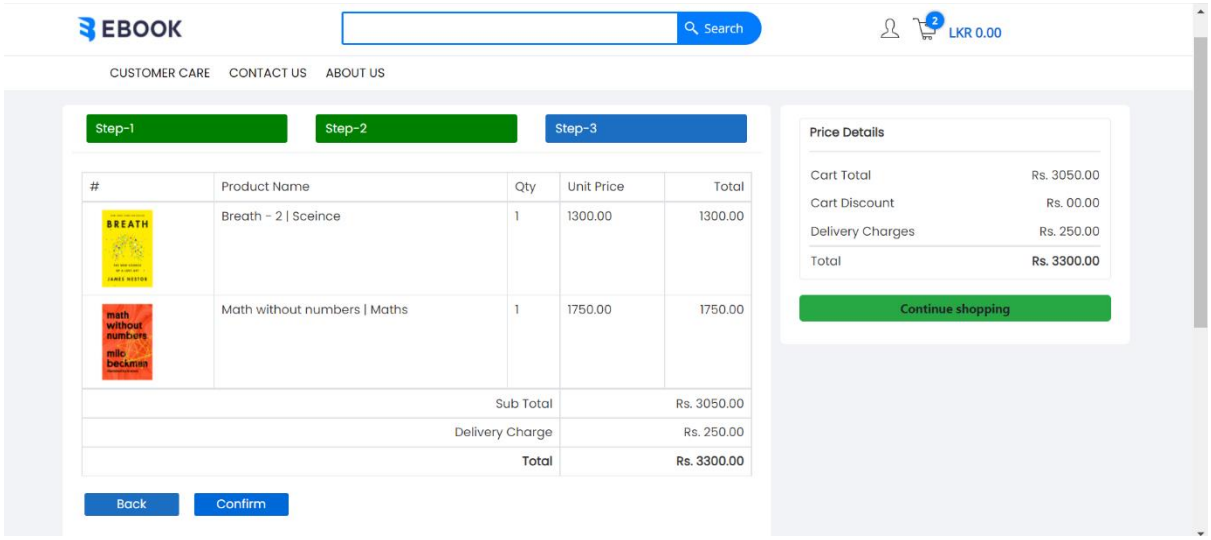


Figure 55 Checkout - Step3

In third step the ordered products will be shown in a table with the total amount. To complete the order customer must click "Confirm" button.

Order Success Message

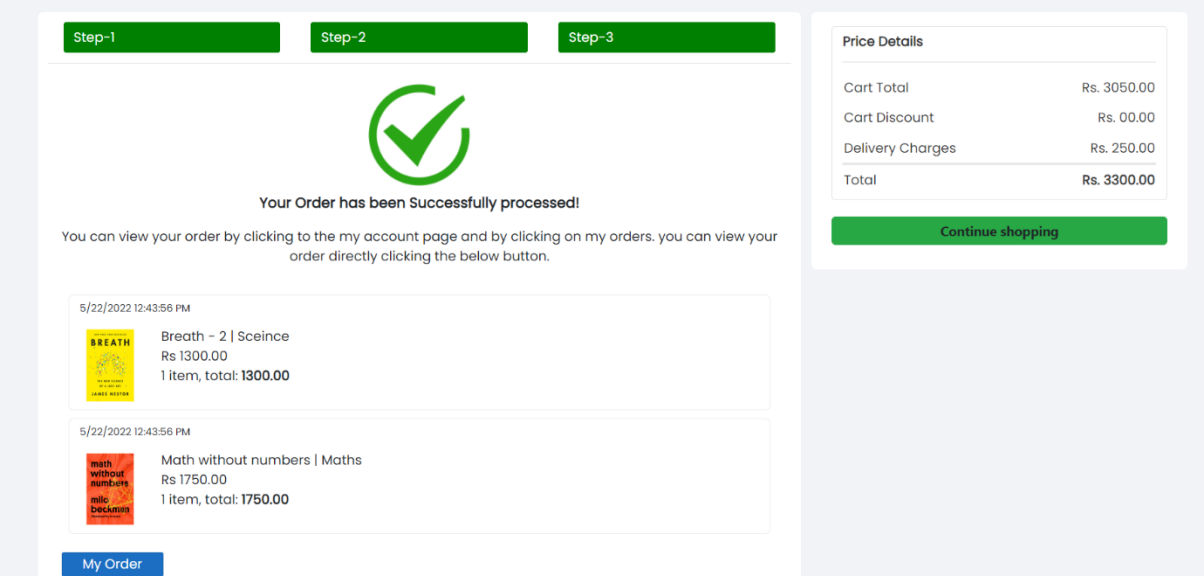


Figure 56 Order Success Message

After confirming the order, a success message will be showed, and customer can click "My Order" button to visit the customer account page to view his order details.

6.1.5 My Account

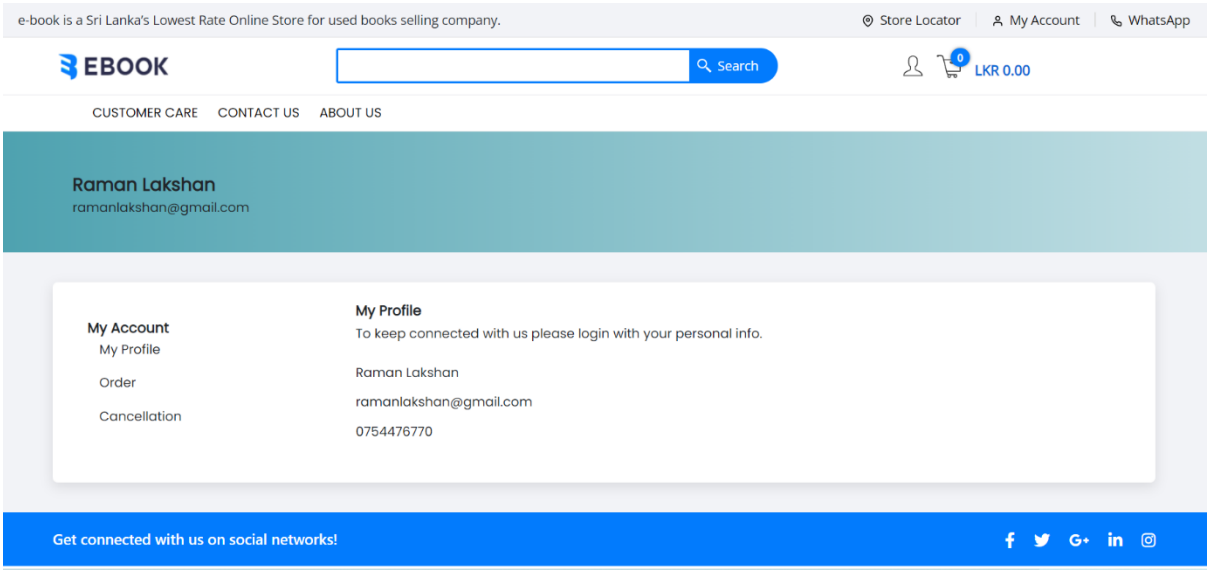


Figure 57 My Account

This interface displays customer account details, and they order details. Customer can click "Order" button to see their orders.

6.1.6 Review

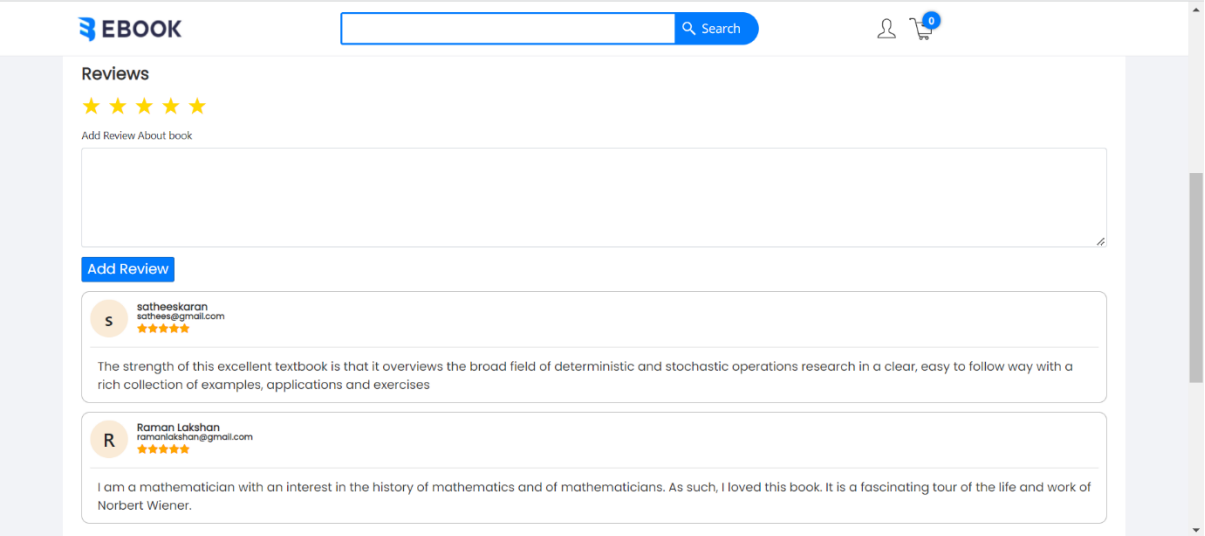


Figure 58 Review

This section shows the review of the customers for the particular book. Customer can give review for the book by entering the review on the text box and clicking “Add Review” button. The review for the book only can given by the registered customer.

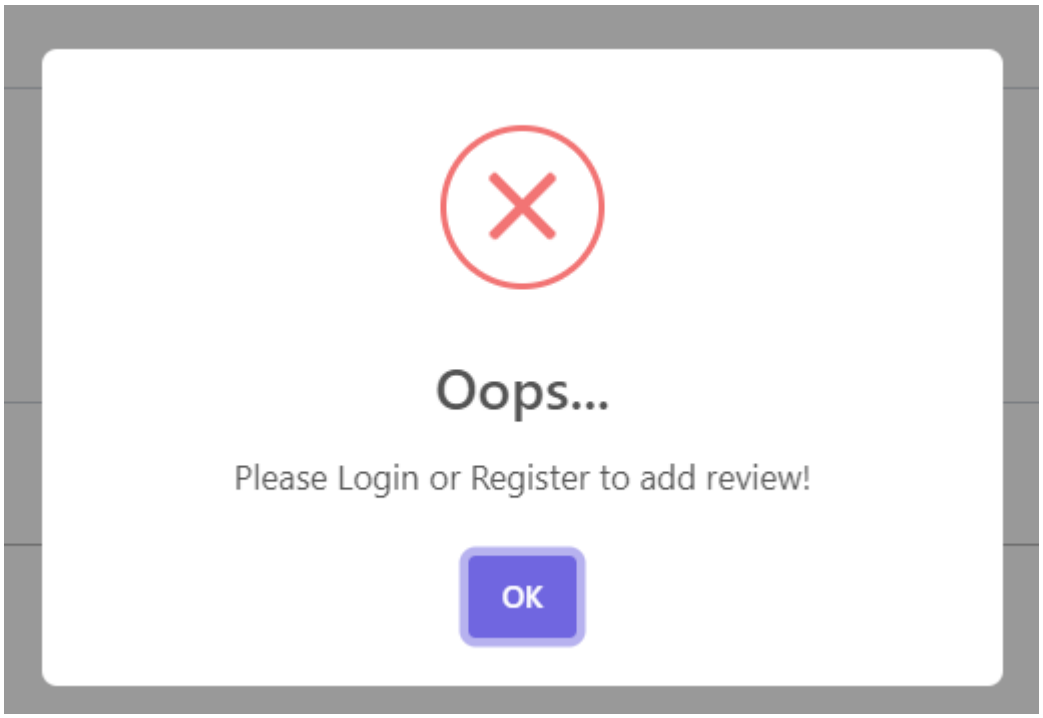


Figure 59 Review Error Message

This is the error alert which customer get when they try to give review for book without registering to the system.

6.1.7 Steps to Order Books and Give Review.

- Customers first need to register to Ebook store to order books. (Anyone can visit the site and successfully add the product to the cart, but they need to login to the system before going to the checkout option system won't allow to proceed order without login to the system).
- Then customer need to select the books and click add to cart button to add the book to the cart. Customer can search the book by category and book name.
- Once the customer added the required books to the cart next, they need to click checkout button to proceed to checkout page. In here every book that added to the cart and the total amount will be showed. To complete the order customer must click "Confirm" button.
- Once the order is completed, a success message will be showed, and customer can click "My Order" button to visit the customer account page to view his order details.
- Customer can provide review by visiting the purchasing product page and in book review section there will be text box given which will allow the user to add book review.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.2 Admin End (Backend of The System).

6.2.1 Login.

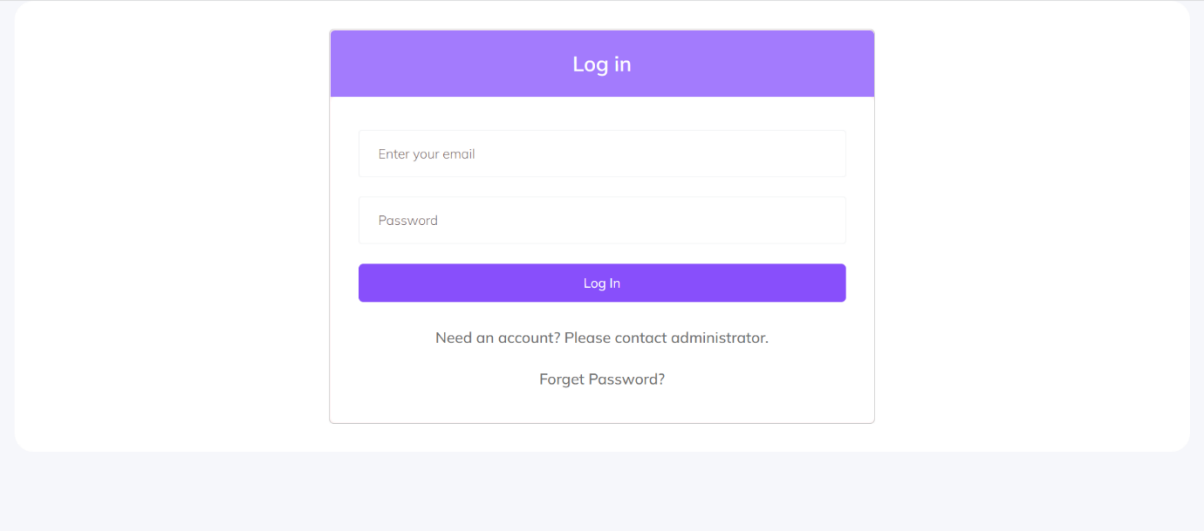


Figure 60 Admin Login

This interface displays the backend system Login page. To log in, employee must enter their User Email and Password into the text fields and then click the Login button. In here employees cannot register to the system. Employee logins will be created by the admin of the system in User Management page.

6.2.2 Dashboard.

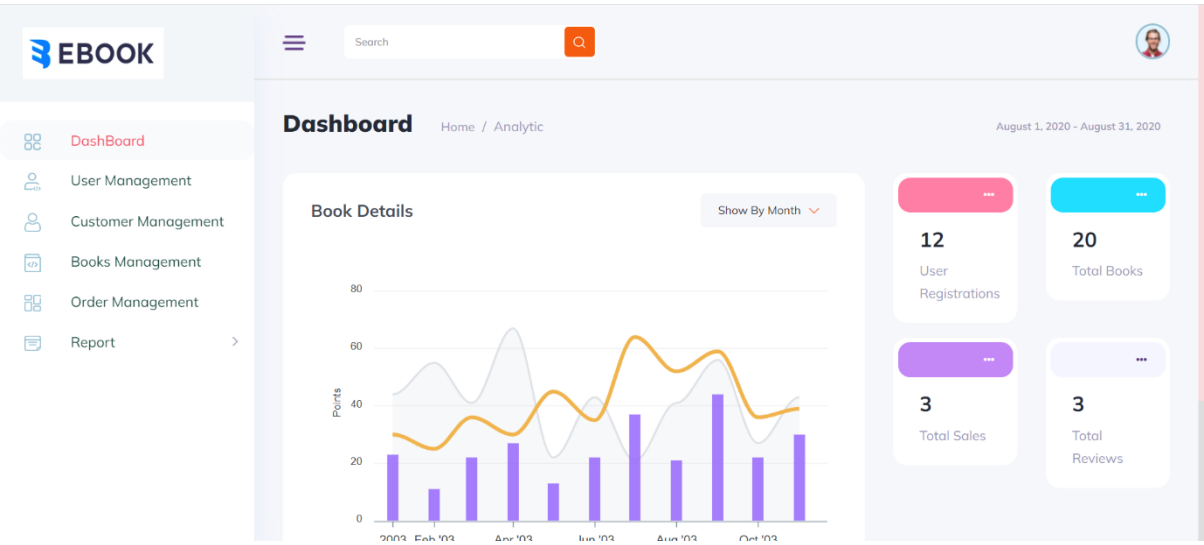


Figure 61 Dashboard

This interface displays the overall details of the Ebook company performance.

6.2.3 User Management.

This process consists of two pages.

View Page:

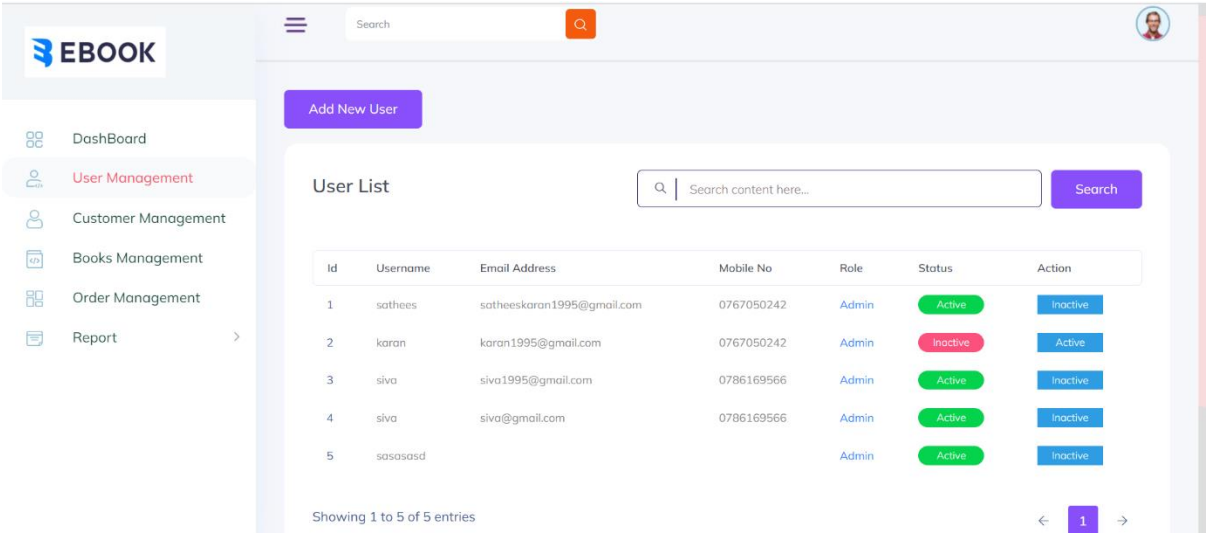


Figure 62 User Management - View

This interface displays all the user details which added to the system in a data table and the action column in the data table have button like “Active” and “Inactive” and if the user profile is active then only “Inactive” button will be shown so if Admin wanted to inactive a user he can click “Inactive” button which will set that user status to inactive so that user cannot login to the system likewise Admin can activate the inactive user by clicking “Active” button. The “Add New User” button will allow the user to add the new user by redirecting to the edit page.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Edit Page:

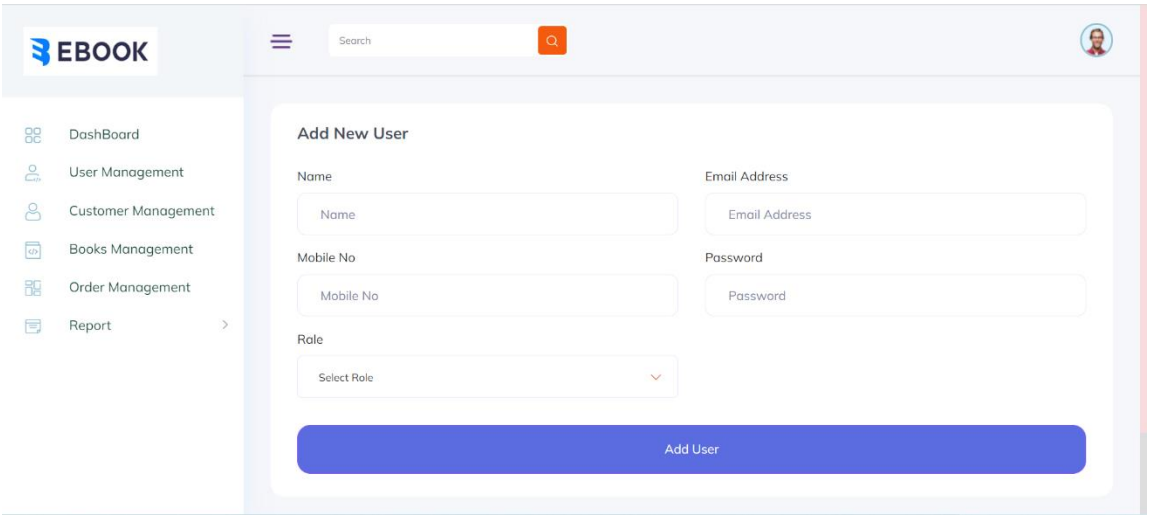


Figure 63 User Management - Edit

If the customer clicks “Add New Book” button on view page this page will allow the user to create a new user. User should fill all the details and click “Add User” button to successfully add the new user to the system.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.2.4 Customer Management.

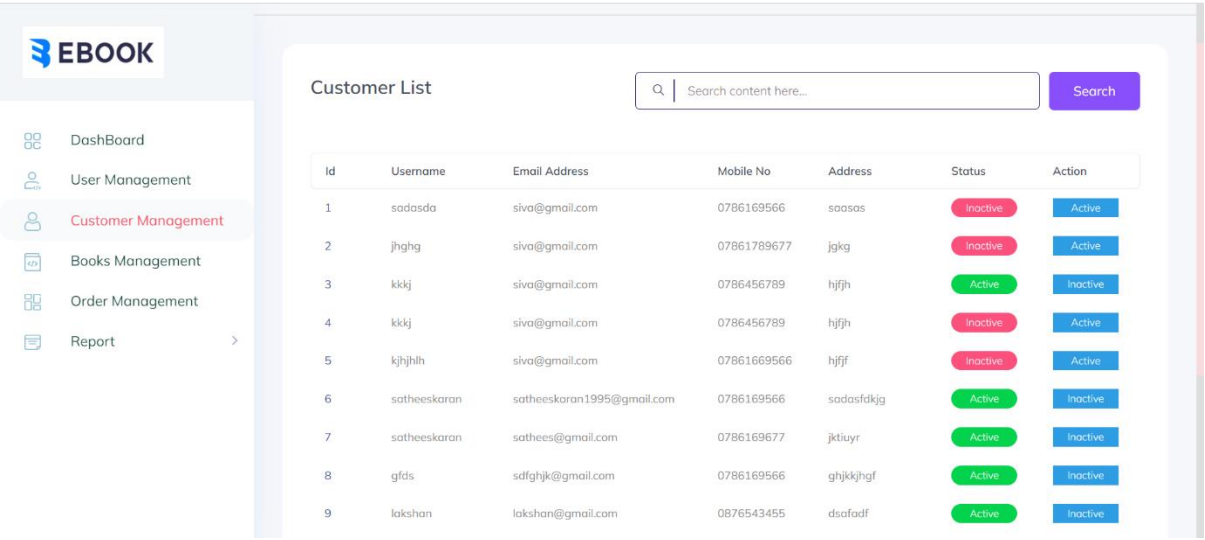


Figure 64 Customer Management

This interface displays all the customer details in a data table and the action column in the data table have button like “Active” and “Inactive” and if the customer profile is active then only “Inactive” button will be shown so if user wanted to inactive a customer they can click “Inactive” button which will set that customer status to inactive so that customer cannot login to the system likewise user can activate the inactive customer account by clicking “Active” button.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.2.5 Book Management.

This process consists of two pages.

View Page:

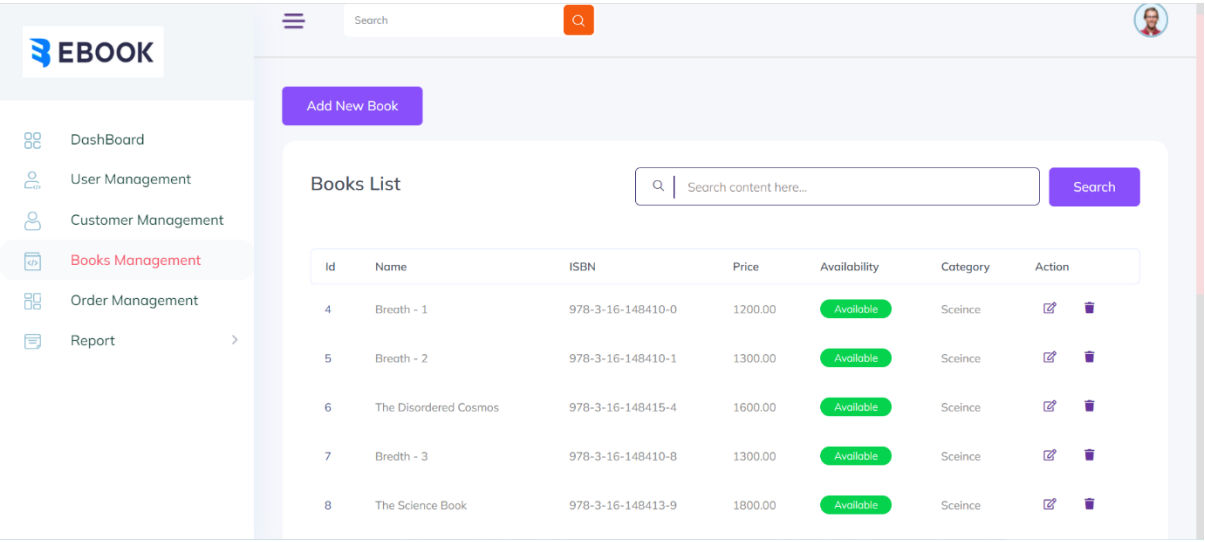


Figure 65 Book Management – View

This interface displays all the books details which added to the system in a data table and the action column in the data table have icon buttons like “edit” and “delete”. The “Add New Book” button will allow the user to add the book by redirecting to the edit page. User can change the details of a book by click “edit” button it will redirect the user to edit page and user can change the details of the book at that page. The “delete” button will allow user to delete the book.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Edit Page:

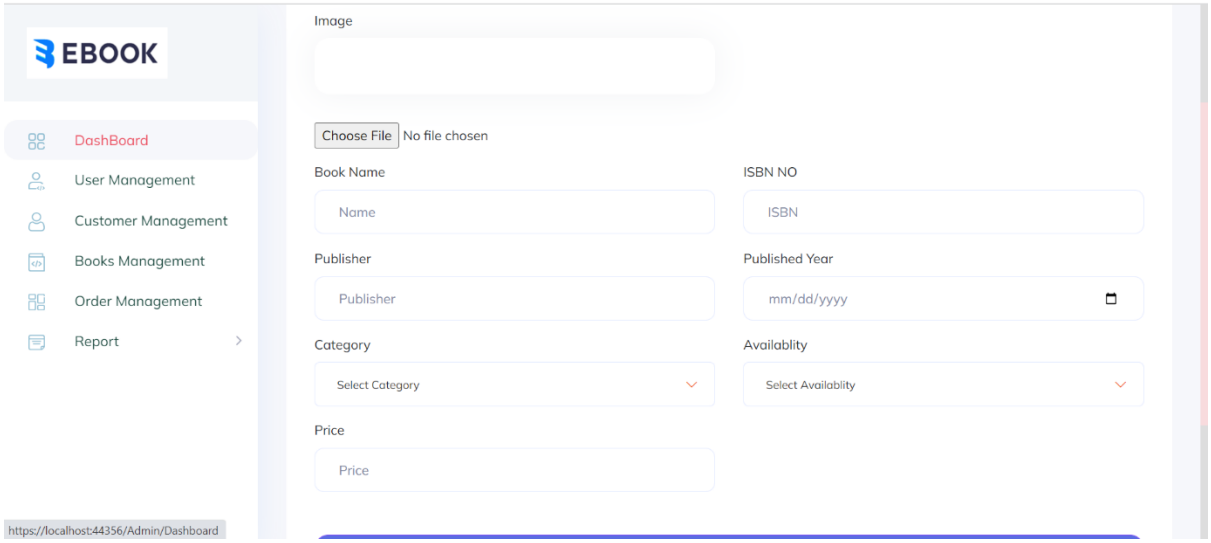


Figure 66 Book Management – Edit

This interface woks in two ways:

- If the user clicks “Add New Book” button on view page this page will allow the user to enter a new book. User should fill all the details and click “Add” button to successfully add the book to the system.
- If user clicks “edit” button on view page data table, then this page will show the details of the book which they clicked. So, user can change the existing data of the book here and click “Add Book” button to successfully update that book data.

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.2.6 Order Management.

This process consists of two pages.

View Page:

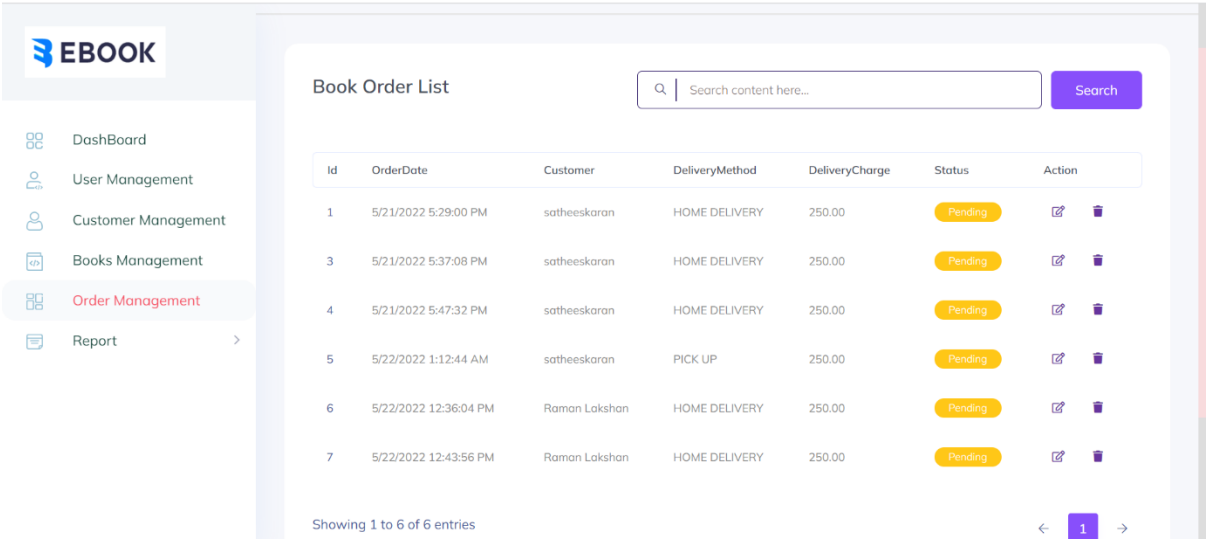


Figure 67 Order Management – View

This interface displays all the order details in a data table and the action column in the data table have icon buttons like “edit” and “delete”. User can see the details of an order by clicking “edit” button it will redirect the user to edit page and user can see the details of the ordered books at that page. The “delete” button will allow user to delete the order.

Edit Page:

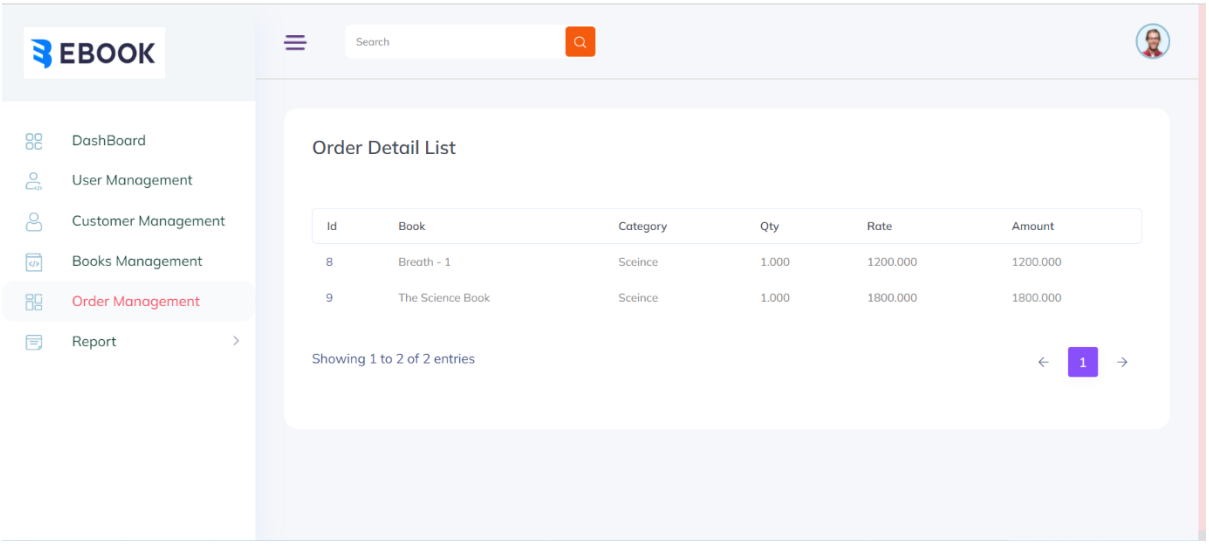


Figure 68 Order Management – Edit

This interface displays the selected order details.

6.2.7 Reports.

User can create five reports in this system:

- Customer Order Report
- Order Cancel Report
- Book Stock Report
- Customer Report
- Employee Report

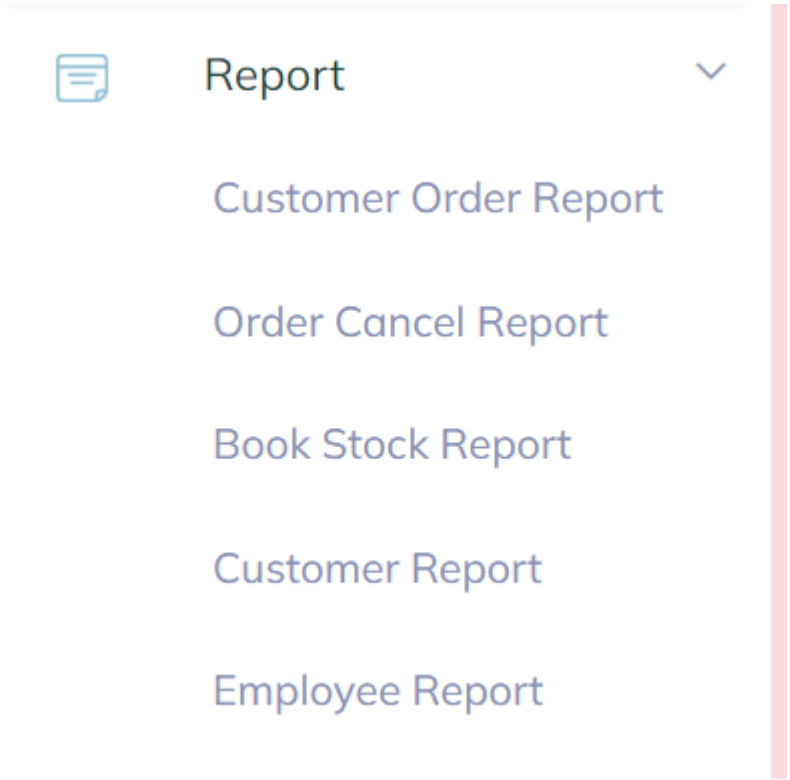


Figure 69 Reports Section

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Customer Order Report:

This report will show all the customers active orders.

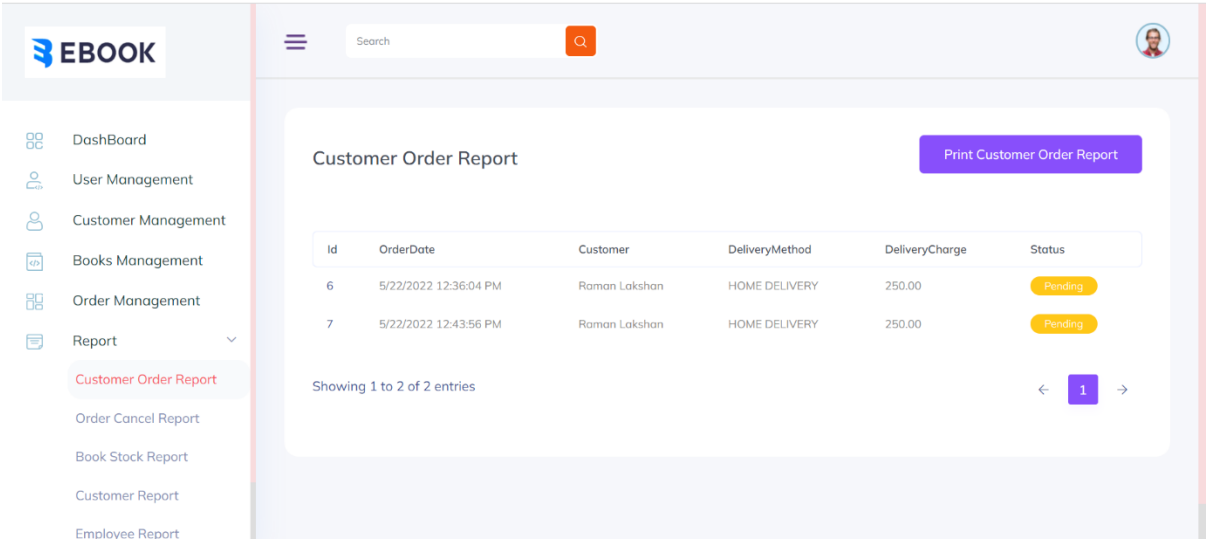


Figure 70 Customer Order Report

Order Cancel Report:

This report will show all the cancelled order details.

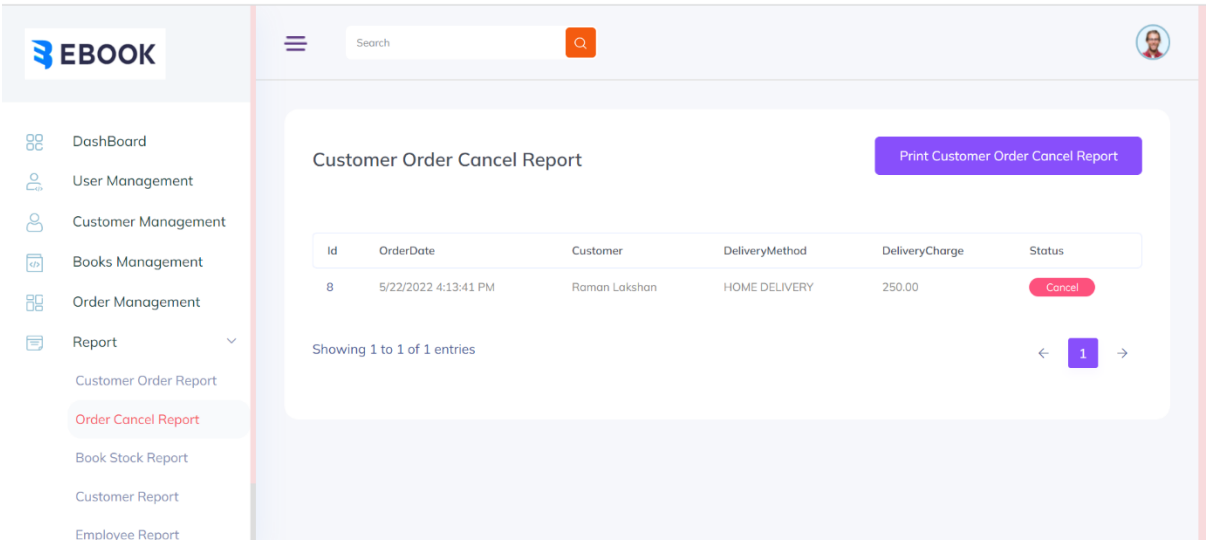


Figure 71 Order Cancel Report

Book Stock Report:

This report will show all the book details.

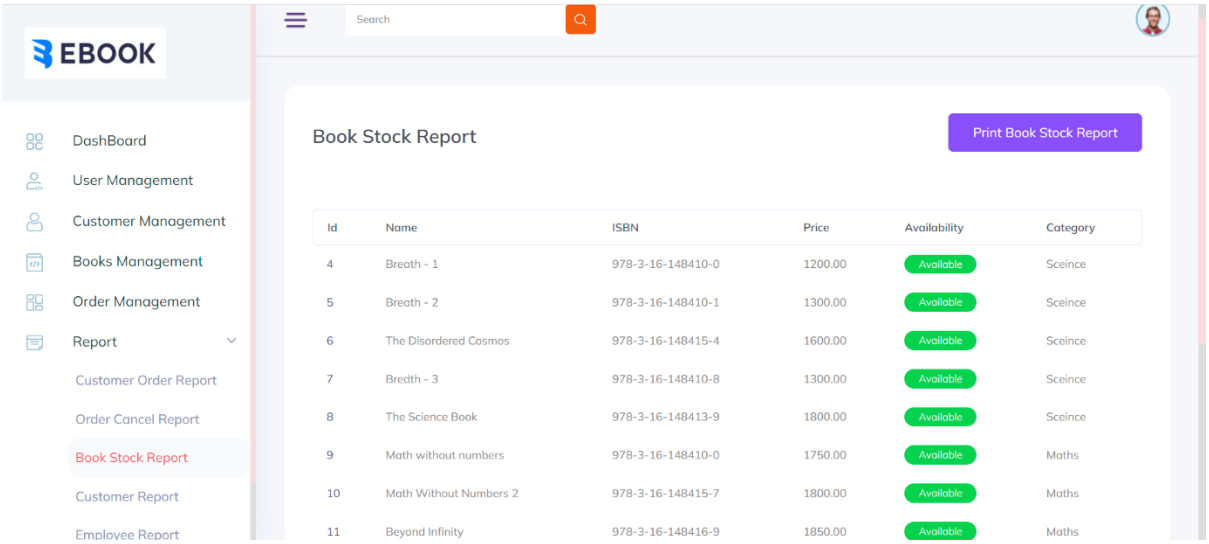


Figure 72 Book Stock Report

Customer Report:

This report will show all the customer details.

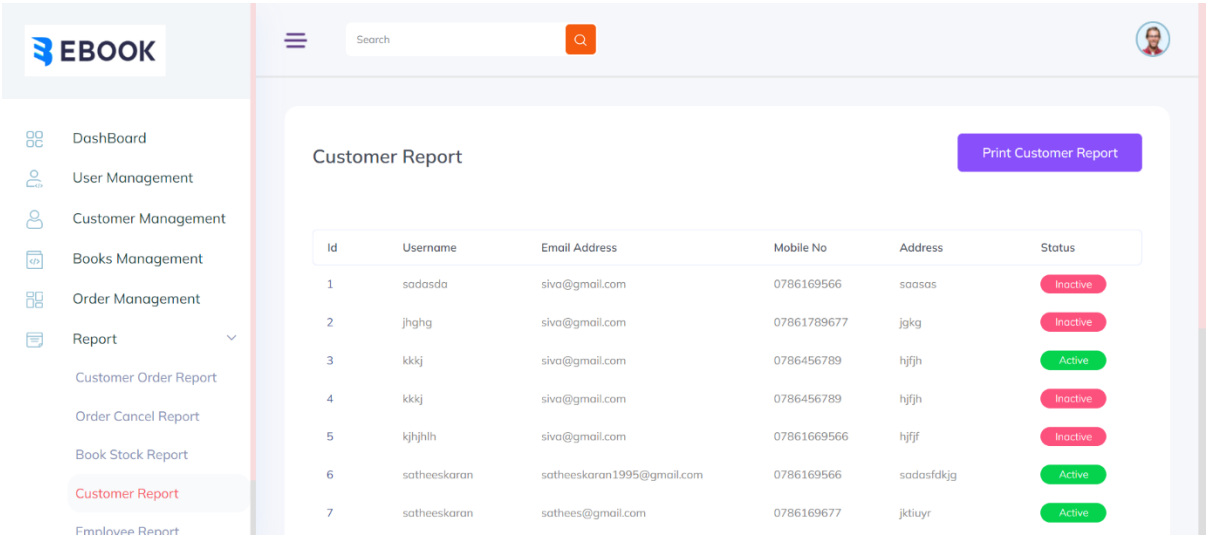


Figure 73 Customer Report

Employee Report

This report will show all the employee details.

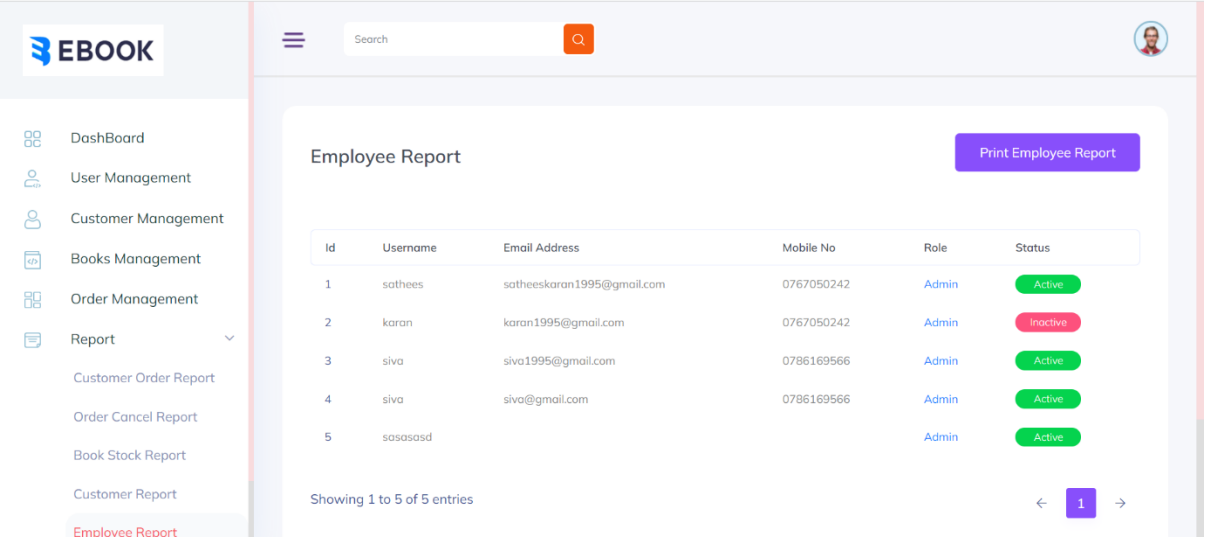


Figure 74 Employee Report

6.2.8 Reports Printing Process.

User can print the reports has pdf file by clicking “Print Report” Button on top right corner of the system. The below image shows the Book Stock Report pdf print file.

Book Stock Report

id	Name	ISBN	Price	Availability	Category
14	Breath - 1	978-3-16-148410-0	1200.00	Available	Sceince
15	Breath - 2	978-3-16-148410-1	1300.00	Available	Sceince
16	The Disordered Cosmos	978-3-16-148415-4	1600.00	Available	Sceince
17	Bredth - 3	978-3-16-148410-8	1300.00	Available	Sceince
18	The Science Book	978-3-16-148413-9	1800.00	Available	Sceince
19	Math without numbers	978-3-16-148410-0	1750.00	Available	Maths
20	Math Without Numbers 2	978-3-16-148415-7	1800.00	Available	Maths
21	Beyond Infinity	978-3-16-148416-9	1850.00	Available	Maths
22	Numbers Dont Lie	978-3-16-148418-8	1300.00	Available	Maths
23	Humble Pi	978-3-16-148415-7	1600.00	Available	Maths

Figure 75 Report Printing

07. CODING BEST PRACTICES FOLLOWED IN THIS SYSTEM DEVELOPMENT.

7.1 Use of CamelCase.

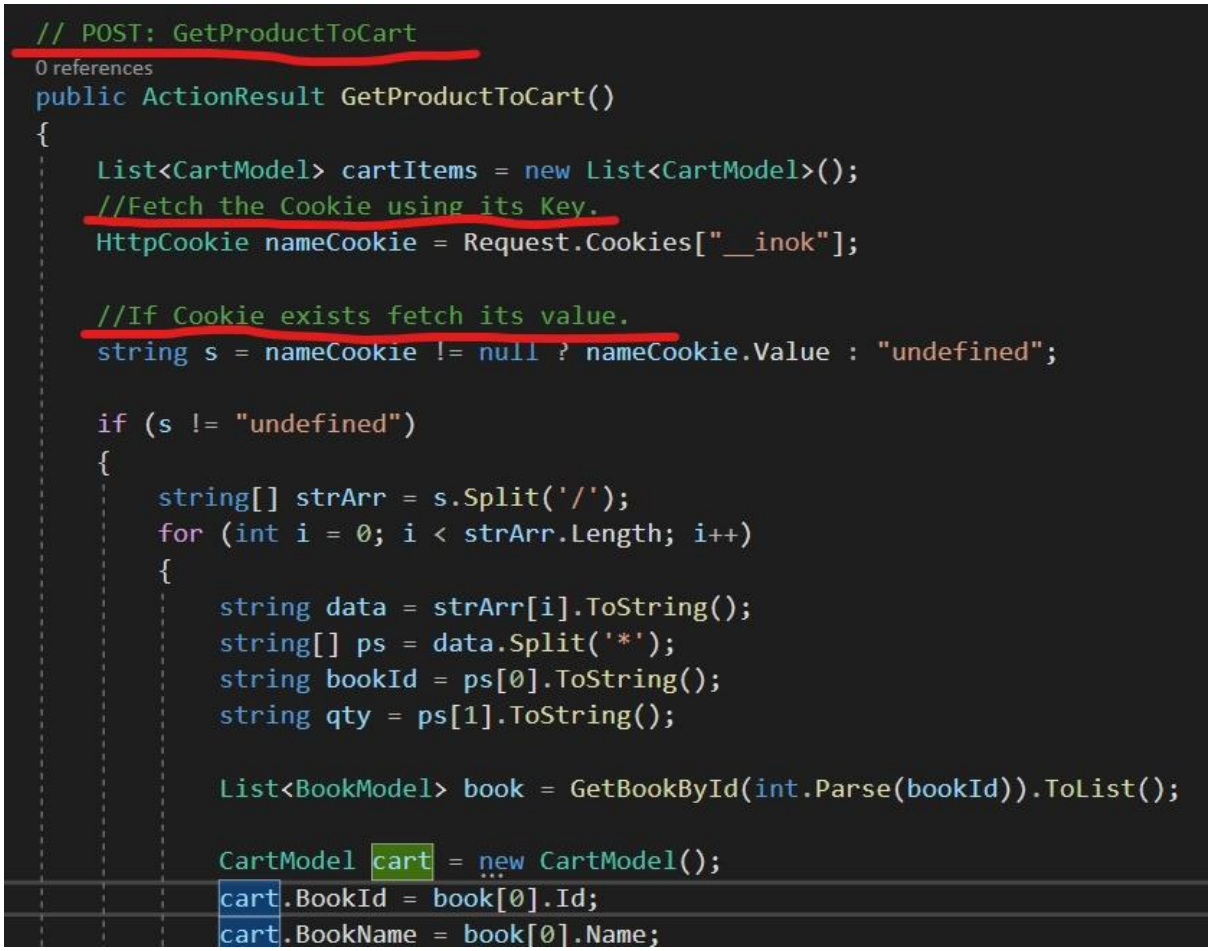
Variable naming is an important aspect in making your code readable.

```
//Decrypt function
0 references
public static string Decrypt(string cipherText)
{
    string EncryptionKey = "MAKV2SPBNI99212";
    byte[] cipherBytes = Convert.FromBase64String(ciph
    using (Aes encryptor = Aes.Create())
    {
        Rfc2898DeriveBytes pdb = new Rfc2898DeriveByte
        encryptor.Key = pdb.GetBytes(32);
        encryptor.IV = pdb.GetBytes(16);
        using (MemoryStream ms = new MemoryStream())
        {
            using (CryptoStream cs = new CryptoStream(
            {
                cs.Write(cipherBytes, 0, cipherBytes.L
                cs.Close();
            }
            cipherText = Encoding.Unicode.GetString(ms
        }
    }
    return cipherText;
}
```

Figure 76 CamelCase

7.2 Use of Comments.

Commenting may be the most important way to organize and segment code.



```
// POST: GetProductToCart
0 references
public ActionResult GetProductToCart()
{
    List<CartModel> cartItems = new List<CartModel>();
    //Fetch the Cookie using its Key.
    HttpCookie nameCookie = Request.Cookies["__inok"];

    //If Cookie exists fetch its value.
    string s = nameCookie != null ? nameCookie.Value : "undefined";

    if (s != "undefined")
    {
        string[] strArr = s.Split('/');
        for (int i = 0; i < strArr.Length; i++)
        {
            string data = strArr[i].ToString();
            string[] ps = data.Split('*');
            string bookId = ps[0].ToString();
            string qty = ps[1].ToString();

            List<BookModel> book = GetBookById(int.Parse(bookId)).ToList();

            CartModel cart = new CartModel();
            cart.BookId = book[0].Id;
            cart.BookName = book[0].Name;
```

Figure 77 Comments

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

7.3 Use of Exception Handling.

Exception handling can control run tune errors that occur in the program.

```
//POST: Register/CheckEmailExist
[HttpPost]
0 references
public ActionResult CheckEmailExist(string email)
{
    try
    {
        string msg = "true";
        Customer.Customer_ProfileDataTable dt = new App_Data.
        if (dt.Count != 0)
        {
            msg = "false";
        }

        return Json(new { result = msg });
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Figure 78 Exception Handling

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

08) Reflection of Own Experience.

8.1 Group Member: Raman Lakshan / 21038992 (E016540).

My experience with this program has been substantially different from my past projects because it addresses a wide range of functional and non-functional criteria. This is the first time that I am working with Model-View-Controller (MVC) in ASP.NET application so it was bit difficult to understand all the process of merging backend coding and front end but with the help of my lecturer and you tube videos we were able to us to finish this e-book system without sacrificing any functionality.

Furthermore, I have greater experience with the database portion, so we were able to plan the correct database structure and implemented it correctly to suit the need of this e-book system.

C# is a more straightforward language to work with and it's easy to achieve the desired outcome quickly. In addition, for this project, I worked on the customer functions. From the register page to checkout page, it's a long path for me with all the essential validations and features need in the customer functions. To wrap up this chapter, I'd like to mention that my overall experience with this implementation was great and unique, and that I learnt a lot of new things that will certainly help me in the future.

8.2 Group Member: Sivayogalingam Satheeskaran / 21038989 (E016551).

I am usual with working on .Net frameworks but the Model-View-Controller (MVC) implementation is totally different from my usual way of coding in a number of ways. Because I used to code on .Net frameworks in previous projects, it was bit easy for me to understand the Model-View-Controller (MVC) process. However, its development is distinguished by the wide range of capabilities and real-world collaboration.

When it came to this system development, I designed the material for the admin functions. Coding was a little difficult for me because of the approaches to the functionalities with Model-View-Controller (MVC. Because of my previous experiences, I was able to complete the assignment on time and without rushing. My experience with the C# language has been favorable because it allows us to complete any task without sacrificing functionality. Finally, my overall experience with this system implementation has been excellent and helpful in preparing me for new challenges.

09) INDIVIDUAL TASK.

9.1 Individual task by Sivayogalingam Satheeskaran.

INDIVIDUAL TASK BY – Sivayogalingam Satheeskaran
1. Document
✓ Functional requirements for admin module.
✓ Non – Functional Requirements
✓ Use case diagram
✓ Class diagram
✓ Test Cases related to admin module.
✓ Testing related to admin module.
2. System
✓ Admin module designing and backend coding (Backend System).

Table 5 - INDIVIDUAL TASK BY – Sivayogalingam Satheeskaran

9.2 Individual task by Raman Lakshan.

INDIVIDUAL TASK BY – Raman Lakshan
1. Document
✓ Functional requirements for customer module.
✓ Database of the system.
✓ Software architecture diagram.
✓ ER diagram.
✓ Test Cases related to customer module.
✓ Testing related to customer module.
2. System
✓ Customer module designing and backend coding (Ecommerce Site).

Table 6 - INDIVIDUAL TASK BY – Raman Lakshan

10) REFERENCES.

- AltexSoft. 2022. *Functional and Nonfunctional Requirements*. [online] Available at: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/#> [Accessed 2 May 2022].
- Troelsen, A. and Japikse, P., 2017. Introducing ASP.NET MVC. Pro C# 7, pp.1179-1221.
- Tutorialsteacher.com. 2022. ASP.NET MVC Tutorials. [online] Available at: <https://www.tutorialsteacher.com/mvc> [Accessed 10 May 2022].

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

THANK YOU