



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

TEAM MEMBERS.....	2
DECLARATION.....	2
ACKNOWLEDGEMENT.....	3
TABLE OF CONTENTS.....	4
TABLE OF FIGURES.....	6
TABLE OF TABLE.....	8
01) ABSTRACT	10
02) REQURIMENT SPECIFICATION	11
2.1 The Functional Requirements of The E-book PVT Ltd.....	11
2.2 The Non-functional Requirements of The E-book PVT Ltd.....	12
03) DESIGN.....	13
3.1 Software Architecture Diagram	13
3.2 Use Case Diagram.....	14
3.3 Class Diagram	15
3.4 ER Diagram.....	16
3.5 Database	17
3.5.1 Book_Profile Table.....	17
3.5.2 Category_Profile Table.....	17
3.5.3 Customer_Profile Table.....	18
3.6 USER-INTERFACE	20
3.7 HARDWARE INTERFACE.....	20
3.8 SOFTWARE INTERFACE	21
04) IMPLEMENTATION	22
05) TESTING.....	42
5.1 Test Cases for Customer End (Ecommerce Site).....	43
5.1 Test Cases for Admin End (Backend System).....	45
06) INSTALLATION GUIDE AND CONFIGURATION AND MANUAL OF SYSTEM ..	47
6.1 Customer End (Ecommerce Store).....	47
6.1.2 Login & Register	49
6.1.3 Cart	50
6.1.4 Checkout.....	50

6.1.5 My Account	52
6.1.6 Review	53
6.1.7 Steps to Order Books and Give Review.	54
6.2 Admin End (Backend of The System).	55
6.2.1 Login.....	55
6.2.2 Dashboard.....	55
6.2.3 User Management.....	56
6.2.4 Customer Management.....	58
6.2.5 Book Management.....	59
6.2.6 Order Management.	61
6.2.7 Reports.....	62
6.2.8 Reports Printing Process.....	65
07. CODING BEST PRACTICES FOLLOWED IN THIS SYSTEM DEVELOPMENT	66
7.1 Use of CamelCase.....	66
7.2 Use of Comments.....	67
7.3 Use of Exception Handling.	68
08) Reflection of Own Experience.	69
8.1 Group Member: Raman Lakshan / 21038992 (E016540).	69
8.2 Group Member: Sivayogalingam Satheeskaran / 21038989 (E016551).....	69
09) INDIVIDUAL TASK.....	70
9.1 Individual task by Sivayogalingam Satheeskaran.....	70
9.2 Individual task by Raman Lakshan.	70
10) REFERENCES	71

TABLE OF FIGURES.

Figure 1 Software Architecture Diagram	13
Figure 2 Use case diagram	14
Figure 3 Class diagram.....	15
Figure 4 ER diagram	16
Figure 5 Book_Profile_Table	17
Figure 6 Category_Profile Table	17
Figure 7 Customer_Profile Table	18
Figure 8 Employee_Profile Table	18
Figure 9 Feedback_Profile Table	19
Figure 10 Order_Detail_Profile Table	19
Figure 11 Order_Header_Profile Table.....	19
Figure 12 User_Profile Table.....	20
Figure 13 Ebook solution structure.....	22
Figure 14 BookModel	23
Figure 15 CartModel	24
Figure 16 CategoryModel.....	24
Figure 17 CustomerModel	25
Figure 18 EmployeeModel	26
Figure 19 FeedbackModel	26
Figure 20 OrderDetailModel	27
Figure 21 OrderHeaderModel	28
Figure 22 UserModel	28
Figure 23 About.....	29
Figure 24 Account.....	29
Figure 25 Admin – Dashboard.....	30
Figure 26 Admin Login	30
Figure 27 Book	31
Figure 28 Cart	31
Figure 29 Checkout.....	32
Figure 30 Customer	32
Figure 31 Home	33
Figure 32 Login	33
Figure 33 Order.....	34
Figure 34 Register	34
Figure 35 Report	35
Figure 36 Shared	35
Figure 37 User	36
Figure 38 AccountController	36
Figure 39 AdminLoginController	37
Figure 40 BookController	37
Figure 41 CartController	38
Figure 42 CustomerController	38
Figure 43 HomeController	39
Figure 44 OrderController.....	39

Figure 45 RegisterController	40
Figure 46 ReportsController.....	40
Figure 47 UserController	41
Figure 48 Home-1.....	47
Figure 49 Home-2.....	47
Figure 50 Home-3.....	48
Figure 51 Login & Register.....	49
Figure 52 Cart	50
Figure 53 Checkout - Step1.....	50
Figure 54 Checkout - Step2.....	51
Figure 55 Checkout - Step3.....	51
Figure 56 Order Success Message	52
Figure 57 My Account	52
Figure 58 Review.....	53
Figure 59 Review Error Message	53
Figure 60 Admin Login	55
Figure 61 Dashboard	55
Figure 62 User Management - View	56
Figure 63 User Management - Edit.....	57
Figure 64 Customer Management.....	58
Figure 65 Book Management – View	59
Figure 66 Book Management – Edit.....	60
Figure 67 Order Management – View	61
Figure 68 Order Management – Edit.....	61
Figure 69 Reports Section	62
Figure 70 Customer Order Report.....	63
Figure 71 Order Cancel Report.....	63
Figure 72 Book Stock Report.....	64
Figure 73 Customer Report	64
Figure 74 Employee Report	65
Figure 75 Report Printing.....	65
Figure 76 CamelCase	66
Figure 77 Comments.....	67
Figure 78 Exception Handling	68

TABLE OF TABLE.

Table 1 - Hardware Configuration.....	21
Table 2 - Software Configuration.....	21
Table 3 Test Cases for Customer End.....	44
Table 4 Test Cases for Admin.....	46
Table 5 - INDIVIDUAL TASK BY – Sivayogalingam Satheeskaran	70
Table 6 - INDIVIDUAL TASK BY – Raman Lakshan	70



“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) REQUIREMENT 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).

- **Usability:** 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.

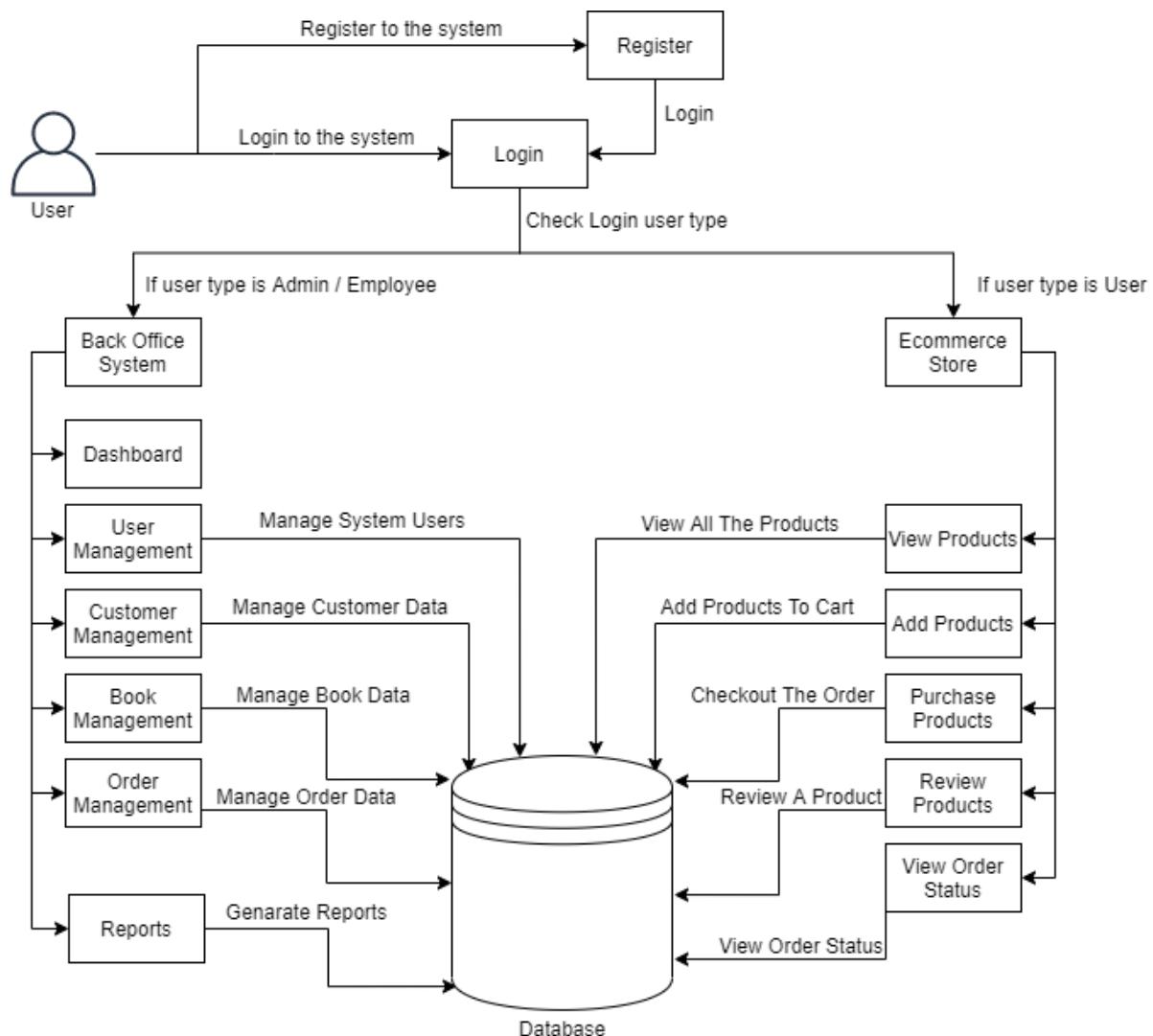


Figure 1 Software Architecture Diagram

3.2 Use Case Diagram.

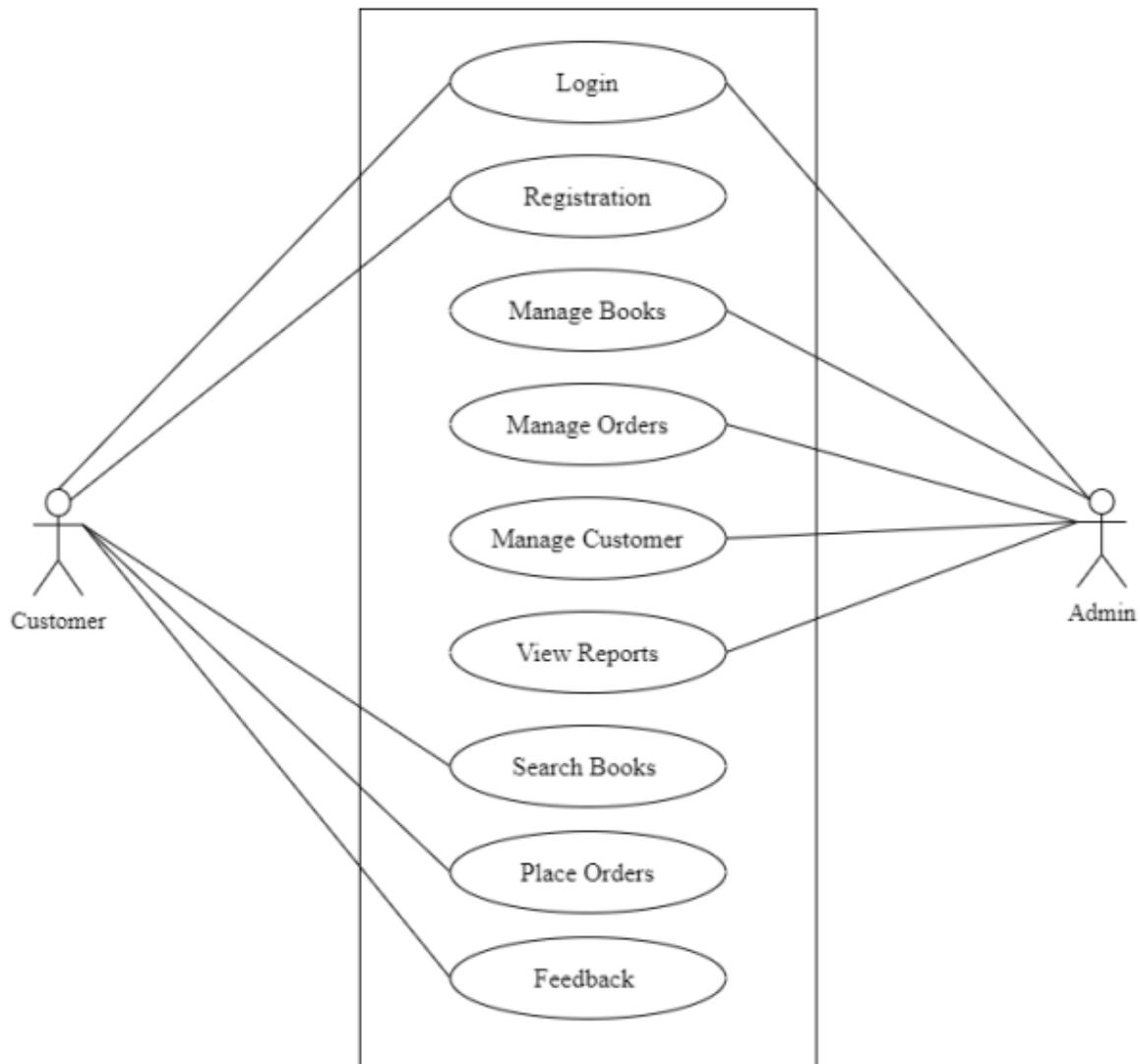


Figure 2 Use case diagram

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

3.3 Class Diagram.

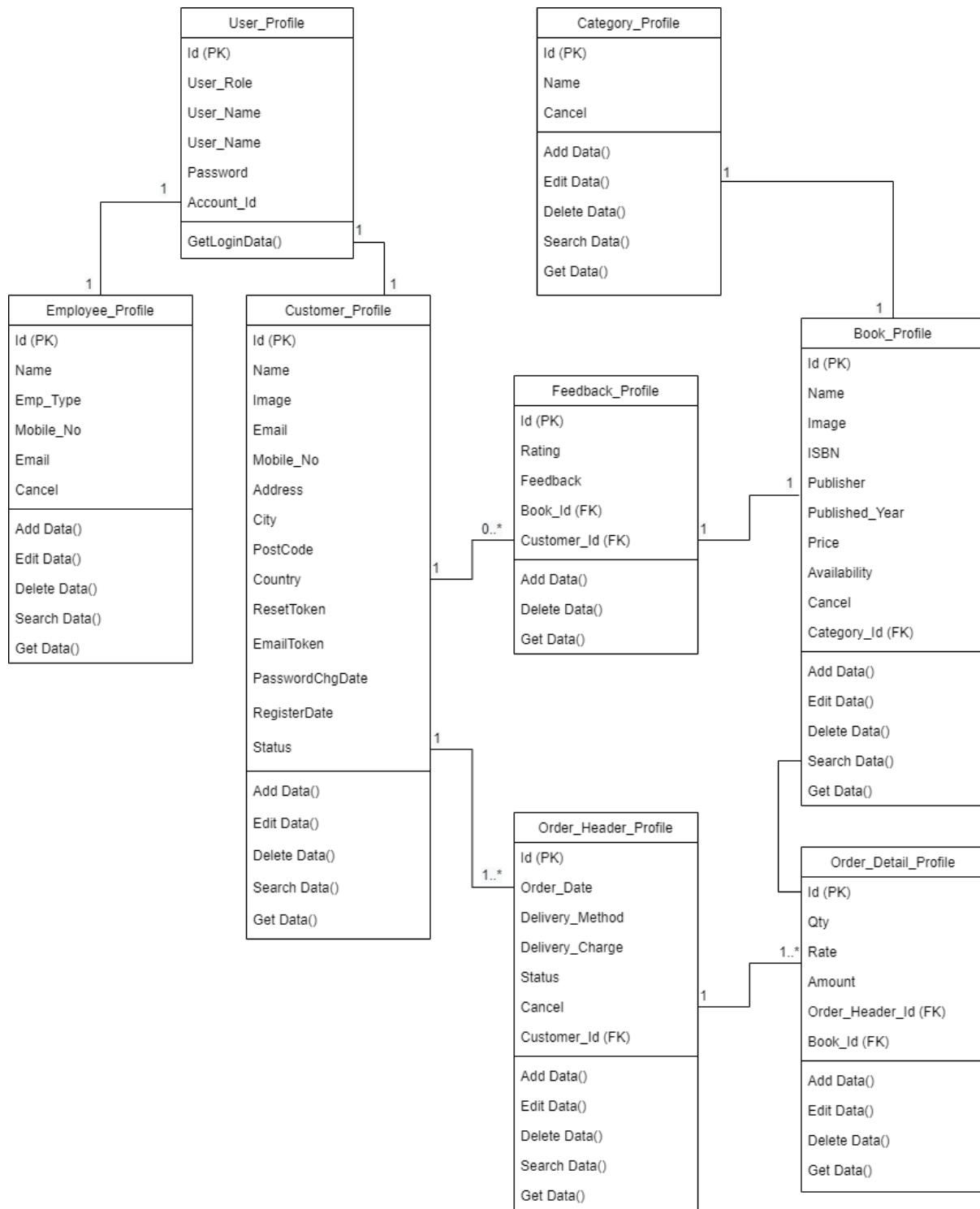


Figure 3 Class diagram

3.4 ER Diagram.

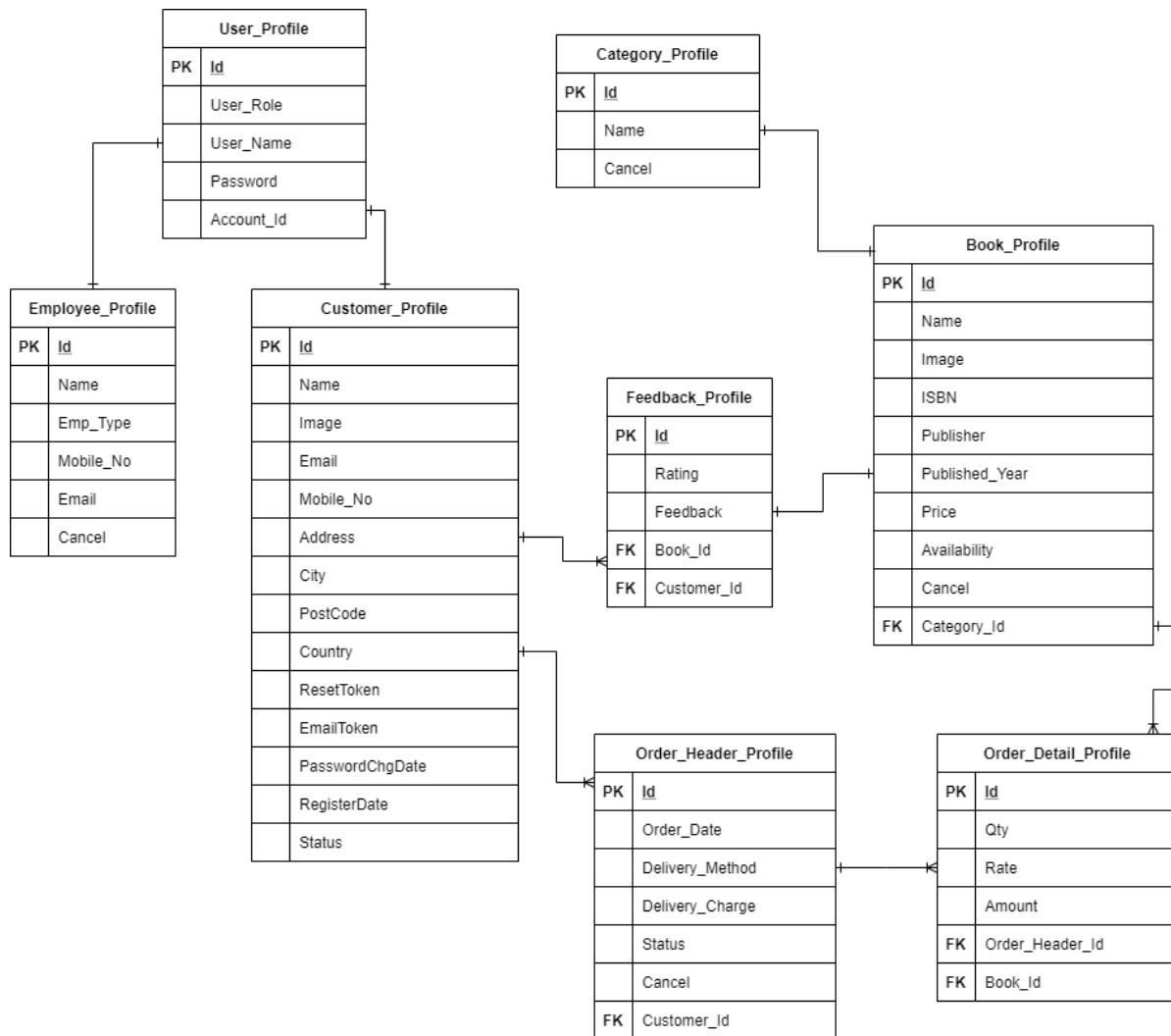


Figure 4 ER diagram

[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

INFODATASYS-PC2...dbo.Book_Profile			Summary
Column Name	Data Type	Allow Nulls	
Id	bigint	<input type="checkbox"/>	
Name	varchar(255)	<input checked="" type="checkbox"/>	
Image	nvarchar(255)	<input checked="" type="checkbox"/>	
ISBN	nvarchar(255)	<input checked="" type="checkbox"/>	
Publisher	varchar(255)	<input checked="" type="checkbox"/>	
Published_Year	datetime	<input checked="" type="checkbox"/>	
Price	decimal(18, 2)	<input checked="" type="checkbox"/>	
Availability	int	<input checked="" type="checkbox"/>	
Category_Id	bigint	<input checked="" type="checkbox"/>	
Cancel	int	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

Figure 5 Book_Profile Table

3.5.2 Category_Profile Table

INFODATASYS-PC2...Category_Profile		
Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
Name	varchar(255)	<input checked="" type="checkbox"/>
Cancel	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Figure 6 Category_Profile Table

3.5.3 Customer_Profile Table

Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
Name	varchar(255)	<input checked="" type="checkbox"/>
Image	nvarchar(255)	<input checked="" type="checkbox"/>
Email	varchar(50)	<input checked="" type="checkbox"/>
Mobile_No	varchar(50)	<input checked="" type="checkbox"/>
Address	varchar(255)	<input checked="" type="checkbox"/>
City	varchar(255)	<input checked="" type="checkbox"/>
PostCode	varchar(20)	<input checked="" type="checkbox"/>
Country	varchar(255)	<input checked="" type="checkbox"/>
ResetToken	varchar(50)	<input checked="" type="checkbox"/>
EmailToken	varchar(50)	<input checked="" type="checkbox"/>
PasswordChgDate	datetime	<input checked="" type="checkbox"/>
RegisterDate	datetime	<input checked="" type="checkbox"/>
Status	int	<input type="checkbox"/>

Figure 7 Customer_Profile Table

3.5.4 Employee_Profile Table

Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
Name	varchar(255)	<input checked="" type="checkbox"/>
Emp_Type	nvarchar(255)	<input checked="" type="checkbox"/>
Mobile_No	nvarchar(15)	<input checked="" type="checkbox"/>
Email	nvarchar(255)	<input checked="" type="checkbox"/>
Cancel	int	<input checked="" type="checkbox"/>

Figure 8 Employee_Profile Table

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

3.5.5 Feedback_Profile Table

Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
Rating	int	<input checked="" type="checkbox"/>
Feedback	varchar(255)	<input checked="" type="checkbox"/>
Book_Id	bigint	<input checked="" type="checkbox"/>
Customer_Id	bigint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Figure 9 Feedback_Profile Table

3.5.6 Order_Detail_Profile Table

Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
Qty	decimal(18, 3)	<input checked="" type="checkbox"/>
Rate	decimal(18, 3)	<input checked="" type="checkbox"/>
Amount	decimal(18, 3)	<input checked="" type="checkbox"/>
Order_Header_Id	bigint	<input checked="" type="checkbox"/>
Book_Id	bigint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Figure 10 Order_Detail_Profile Table

3.5.7 Order_Header_Profile Table

Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
Order_Date	datetime	<input checked="" type="checkbox"/>
Delivery_Method	varchar(50)	<input checked="" type="checkbox"/>
Delivery_Charge	decimal(18, 2)	<input checked="" type="checkbox"/>
Status	char(50)	<input checked="" type="checkbox"/>
Cancel	int	<input checked="" type="checkbox"/>
Customer_Id	bigint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Figure 11 Order_Header_Profile Table

3.5.8 User_Profile Table

Column Name	Data Type	Allow Nulls
Id	bigint	<input type="checkbox"/>
User_Role	varchar(255)	<input checked="" type="checkbox"/>
User_Name	varchar(255)	<input checked="" type="checkbox"/>
Password	nvarchar(50)	<input checked="" type="checkbox"/>
Account_Id	bigint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

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.

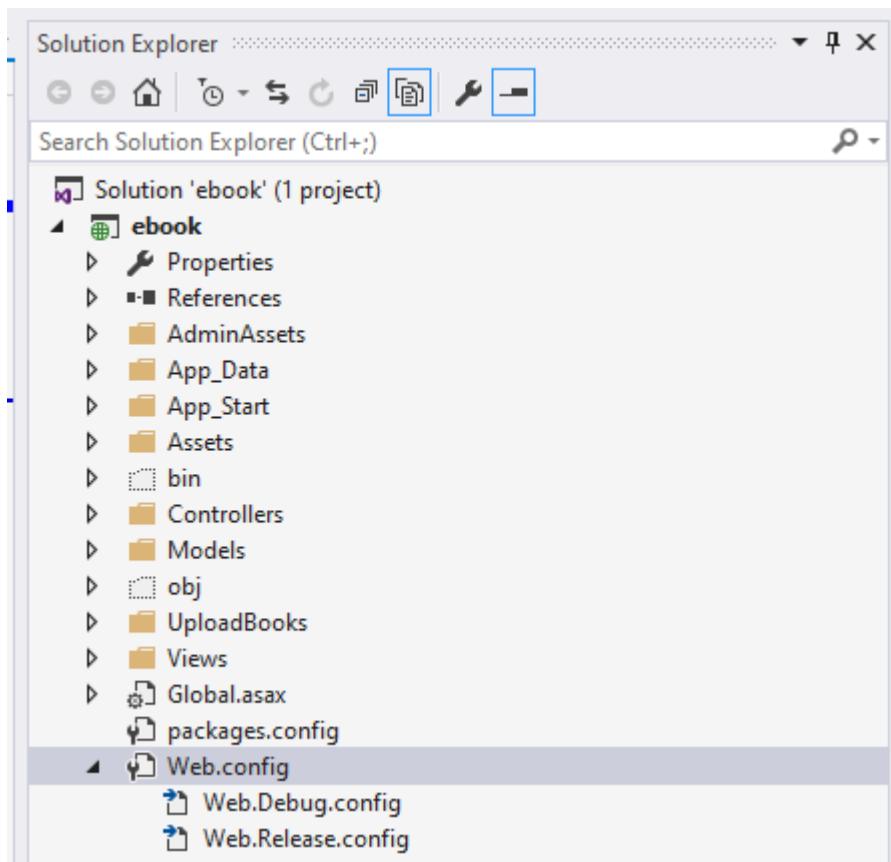


Figure 13 Ebook solution structure

[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
{
    public class BookModel
    {
        public string Id { get; set; }
        public string Name { get; set; }
        public string Image { get; set; }
        public string ISBN { get; set; }
        public string Publisher { get; set; }
        public string Published_Year { get; set; }
        public string Price { get; set; }
        public string Availability { get; set; }
        public string Category_Id { get; set; }
        public string Category { get; set; }
        public string Cancel { get; set; }
    }
}
```

Figure 14 BookModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

CartModel

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

Figure 15 CartModel

CategoryModel

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

Figure 16 CategoryModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

CustomerModel

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

Figure 17 CustomerModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

EmployeeModel

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

Figure 18 EmployeeModel

FeedbackModel

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

Figure 19 FeedbackModel

OrderDetailModel

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

Figure 20 OrderDetailModel

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

OrderHeaderModel

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

Figure 21 OrderHeaderModel

UserModel

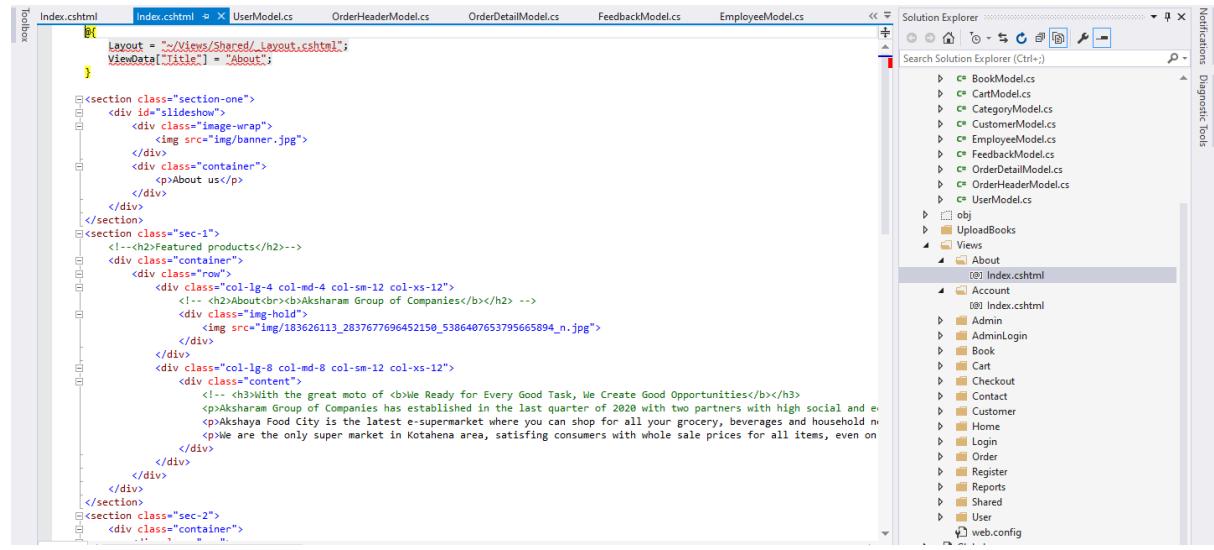
```
namespace ebook.Models
{
    public class UserModel
    {
        public string Name { get; set; }
        public string UserName { get; set; }
        public string UserRole { get; set; }
        public string Password { get; set; }
        public string AccountId { get; set; }
        public string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



The screenshot shows the Visual Studio IDE with the 'About' view code open in the editor. The code uses Razor syntax to render HTML and CSS. The Solution Explorer on the right shows the project structure, including files like BookModel.cs, CartModel.cs, CategoryModel.cs, CustomerModel.cs, EmployeeModel.cs, FeedbackModel.cs, OrderDetailModel.cs, OrderHeaderModel.cs, and UserModel.cs. The 'Views' folder contains 'About' and 'Account' subfolders, each with its own Index.cshtml file.

```
Index.cshtml
Index.cshtml.cs
UserModel.cs
OrderHeaderModel.cs
OrderDetailModel.cs
FeedbackModel.cs
EmployeeModel.cs
```

```

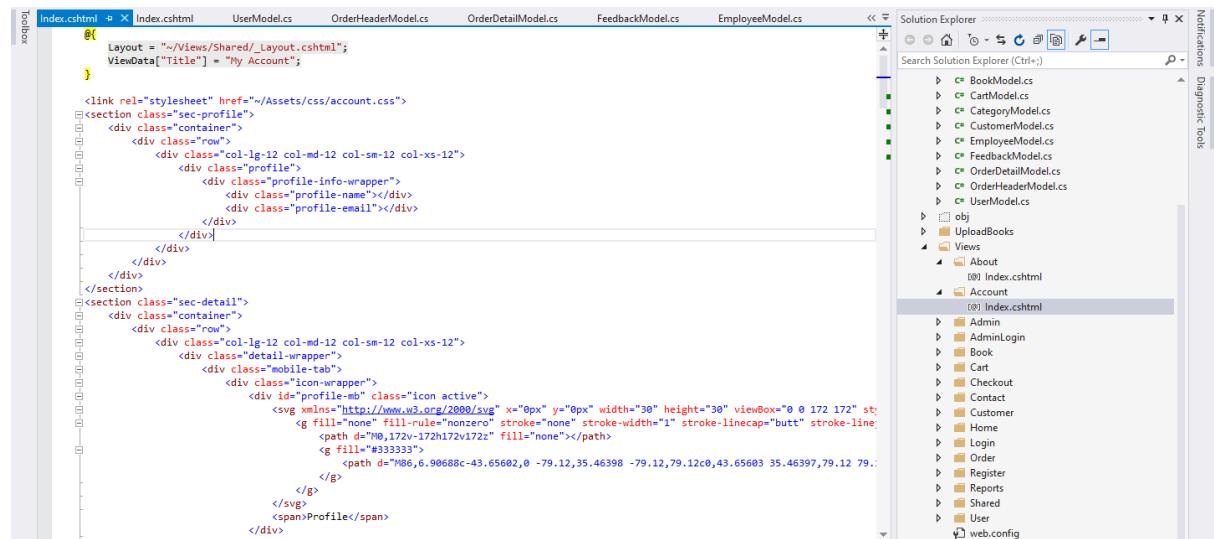
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
    ViewData["Title"] = "About";
}

<section class="section-one">
    <div id="slideshow">
        <div class="image-wrap">
            
        </div>
        <div class="container">
            <p>About us</p>
        </div>
    </div>
</section>
<section class="sec-1">
    <!--<h2>Featured products</h2>-->
    <div class="container">
        <div class="row">
            <div class="col-lg-4 col-md-4 col-sm-12 col-xs-12">
                <!--<h2>About<br/><b>Aksharam Group of Companies</b></h2>-->
                <div class="img-hold">
                    
                </div>
            </div>
            <div class="col-lg-8 col-md-8 col-sm-12 col-xs-12">
                <div class="content">
                    <!--<h3>With the great moto of <b>We Ready for Every Good Task, We Create Good Opportunities</b></h3>
                    <p>Aksharam Group of Companies has established in the last quarter of 2020 with two partners with high social and e-<br/><p>Akshaya Food City is the latest e-supermarket where you can shop for all your grocery, beverages and household ne-<br/><p>are the only super market in Kotahena area, satisfying consumers with whole sale prices for all items, even on
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<section class="sec-2">
    <div class="container">
        ...
    </div>

```

Figure 23 About

Account



The screenshot shows the Visual Studio IDE with the 'Account' view code open in the editor. The code uses Razor syntax to render HTML and CSS, including an SVG icon. The Solution Explorer on the right shows the project structure, including files like BookModel.cs, CartModel.cs, CategoryModel.cs, CustomerModel.cs, EmployeeModel.cs, FeedbackModel.cs, OrderDetailModel.cs, OrderHeaderModel.cs, and UserModel.cs. The 'Views' folder contains 'About' and 'Account' subfolders, each with its own Index.cshtml file.

```
Index.cshtml
Index.cshtml.cs
UserModel.cs
OrderHeaderModel.cs
OrderDetailModel.cs
FeedbackModel.cs
EmployeeModel.cs
```

```

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

<link rel="stylesheet" href="~/Assets/css/account.css">
<section class="sec-profile">
    <div class="container">
        <div class="row">
            <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
                <div class="profile">
                    <div class="profile-info-wrapper">
                        <div class="profile-name"></div>
                        <div class="profile-email"></div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<section class="sec-detail">
    <div class="container">
        <div class="row">
            <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
                <div class="detail-wrapper">
                    <div class="mobile-tab">
                        <div class="icon-wrapper">
                            <div id="profileTab" class="icon active">
                                <img alt="Profile icon" data-bbox="114 788 154 821" />
                            </div>
                            <span>Profile</span>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
```

Figure 24 Account

Admin - Dashboard

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for 'Dashboard.cshtml'. The right pane shows the 'Solution Explorer' with project files like BookModel.cs, CartModel.cs, etc., and a folder structure for 'Views' containing 'About', 'Account', 'Admin', and other subfolders. The 'Admin' folder contains files such as 'Index.cshtml' and 'Dashboard.cshtml', which is highlighted.

```
<div class="main_content_inner overely_inner">
    <div class="container-fluid p-0">
        <div class="row">
            <div class="col-12">
                <div class="page_title_box d-flex flex-wrap align-items-center justify-content-between">
                    <div class="page_title_left d-flex align-items-center">
                        <h3 class="f_s_25 f_w_700 dark_text mr_30">Dashboard</h3>
                        <ol class="breadcrumb page_breadcrumb mb-0">
                            <li class="breadcrumb-item"><a href="javascript:void(0);">Home</a></li>
                            <li class="breadcrumb-item active">Analytic</li>
                        </ol>
                    </div>
                    <div class="page_title_right">
                        <div class="page_date_button d-flex align-items-center">
                            
                            August 1, 2020 - August 31, 2020
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-8">
                <div class="white_card mb_30 card_height_100">
                    <div class="white_card_header">
                        <div class="row align-items-center justify-content-between flex-wrap">
                            <div class="col-4">
                                <div class="main-title">
                                    <h3 class="m-0">Stoke Details</h3>
                                </div>
                            </div>
                            <div class="col-1g-4 text-right d-flex justify-content-end">
                                <select class="nice_Select2 max-width-220">
                                    <option value="1">Show by month</option>
                                    <option value="2">Show by day</option>
                                </select>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

Figure 25 Admin – Dashboard

Admin Login

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for 'Index.cshtml'. The right pane shows the 'Solution Explorer' with project files and a folder structure for 'Views' containing 'About', 'Account', 'Admin', and other subfolders. The 'Admin' folder contains files such as 'Index.cshtml' and 'AdminLogin.cshtml', which is highlighted.

```
<div class="main_content_inner overely_inner">
    <div class="container-fluid p-0">
        <div class="row">
            <div class="col-12">
                <div class="page_title_box d-flex flex-wrap align-items-center justify-content-between">
                    <div class="page_title_left d-flex align-items-center">
                        <h3 class="f_s_25 f_w_700 dark_text mr_30">Dashboard</h3>
                        <ol class="breadcrumb page_breadcrumb mb-0">
                            <li class="breadcrumb-item"><a href="javascript:void(0);">Home</a></li>
                            <li class="breadcrumb-item active">Analytic</li>
                        </ol>
                    </div>
                    <div class="page_title_right">
                        <div class="page_date_button d-flex align-items-center">
                            
                            August 1, 2020 - August 31, 2020
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-8">
                <div class="white_card mb_30 card_height_100">
                    <div class="white_card_header">
                        <div class="row align-items-center justify-content-between flex-wrap">
                            <div class="col-4">
                                <div class="main-title">
                                    <h3 class="m-0">Stoke Details</h3>
                                </div>
                            </div>
                            <div class="col-1g-4 text-right d-flex justify-content-end">
                                <select class="nice_Select2 max-width-220">
                                    <option value="1">Show by month</option>
                                    <option value="2">Show by day</option>
                                </select>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

Figure 26 Admin Login

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Book

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `EditBook.cshtml` file, which contains HTML and C# code for a book editing page. On the right, the Solution Explorer pane shows the project structure, including files like `BookModel.cs`, `CartModel.cs`, and various views such as `About`, `Account`, `Admin`, `Book`, `Cart`, `Checkout`, `Contact`, `Customer`, `Home`, `Login`, `Order`, and `UploadBooks`.

```
using ebook.Models;
using System.Linq;
@{
    Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
}



<div class="container-fluid p-0">
        <div class="row">
            <div class="col-12">
                <div class="white_card card_height_100 mb_30">
                    <div class="white_card_header">
                        <div class="box_header m-0">
                            <div class="main-title">
                                <h3 class="m-0">Edit Book </h3>
                            </div>
                        </div>
                    </div>
                    <div class="white_card_body">
                        <div class="row">
                            <div class="col-lg-6">
                                <div class="mb_10">
                                    <div class="box col-6">
                                        <div class="card custom-card">
                                            
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div class="col-lg-12">
                                <div class="mb_15">
                                    <input accept="image/png, image/gif, image/jpeg" type="file" class="form-control-file" id="UploadImage" />
                                </div>
                            </div>
                            <div class="col-lg-6">
                                <div class="common_input mb_15">
                                    <input id="id" type="text" placeholder="Id" value="@ViewBag.Id" disabled>
                                </div>
                            <div class="mb_15">
                                <input type="button" value="Edit" class="button" />
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>


```

Figure 27 Book

Cart

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `Index.cshtml` file, which contains HTML and C# code for a shopping cart page. On the right, the Solution Explorer pane shows the project structure, including files like `BookModel.cs`, `CartModel.cs`, and various views such as `About`, `Account`, `Admin`, `Book`, `Cart`, `Checkout`, `Contact`, `Customer`, `Home`, `Login`, and `UploadBooks`.

```
Layout = "~/Views/Shared/_Layout.cshtml";
 ViewData["Title"] = "Cart";

<link rel="stylesheet" href="/Assets/css/cart.css">
<div class="info_top_bar"></div>
<section class="productsec-1">
    <div class="overlay"></div>
    <div class="container">
        <div class="row">
            <div class="col-lg-8 col-md-8 col-sm-12 col-xs-12">
                <div class="info_left">
                    <div class="out-stock-wrap">
                        <div class="empty-cart">
                            
                        </div>
                        <div class="empty-text">
                            <p>Cart is empty!</p>
                            <span>Looks like you have no items in your shopping cart. Click <a href="/Home">here</a> to continue shopping!</span>
                        </div>
                    </div>
                    <div id="fetch-items"></div>
                </div>
            </div>
            <div class="col-lg-4 col-md-4 col-sm-12 col-xs-12">
                <div class="info_right">
                    <div class="box sa-box">
                        <div class="price-detail">
                            <span class="title">Price Details</span>
                            <hr>
                            <table>
                                <tr>
                                    <td>Cart Total</td>
                                    <td id="cart-total" class="fl-right">Rs. 00.00</td>
                                </tr>
                                <tr>
                                    <td>Cart Discount</td>
                                    <td>0.00</td>
                                </tr>
                            </table>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

Figure 28 Cart

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Checkout

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for a C# ASP.NET MVC view named 'Index.cshtml'. The code includes a navigation bar, a tabbed section with three tabs, and a form for entering address information. The right pane shows the 'Solution Explorer' with a tree view of the project structure, including 'Views/Checkout/Index.cshtml' selected.

```
<link rel="stylesheet" href="~/Assets/css/checkout.css">


</div>
<section class="productsec-1">
    <div class="container">
        <div class="row">
            <div class="col-lg-8 col-md-8 col-sm-12 col-xs-12">
                <div class="info_left">
                    <ul class="nav nav-pills mb-3" id="pills-tab" role="tablist">
                        <li class="col-4 nav-item">
                            <a class="nav-link active disabled" id="pills-1-tab" data-toggle="pill" href="#pills-1" role="tab" aria-controls="pills-1">Address</a>
                        </li>
                        <li class="col-4 nav-item">
                            <a class="nav-link disabled" id="pills-2-tab" data-toggle="pill" href="#pills-2" role="tab" aria-controls="pills-2">Old Address</a>
                        </li>
                        <li class="col-4 nav-item">
                            <a class="nav-link disabled" id="pills-3-tab" data-toggle="pill" href="#pills-3" role="tab" aria-controls="pills-3">New Address</a>
                        </li>
                    </ul>
                <hr>
                <div id="danger" class="alert alert-danger" role="alert"></div>
                <div class="tab-content" id="pills-tabContent">
                    <div class="tab-pane fade show active" id="pills-1" role="tabpanel" aria-labelledby="pills-1-tab">
                        <div>
                            <h6>Address</h6>
                            <div class="custom-control custom-radio">
                                <input type="radio" class="custom-control-input" id="existaddress" name="defaultExampleRadios1" checked="">
                                <label class="custom-control-label" for="existaddress">I want to use existing address</label>
                            </div>
                            <fieldset disabled>
                                <div class="form-group">
                                    <input type="text" id="oldAddress" class="form-control" placeholder="">
                                </div>
                            </fieldset>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>


```

Figure 29 Checkout

Customer

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for a C# ASP.NET MVC view named 'ViewCustomer.cshtml'. The code includes a search bar and a table for displaying customer data. The right pane shows the 'Solution Explorer' with a tree view of the project structure, including 'Views/Customer/Index.cshtml' selected.

```
using ebook.Models;
@{
    Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
}
<div class="main_content_iner overly_inner ">
    <div class="container-fluid p-0">
        <div class="row">
            <div class="col-lg-12">
                <div class="white_card card_height_100 mb_30 pt-4">
                    <div class="white_card_body">
                        <div class="position-relative">
                            <div class="white_box_tittle list_header">
                                <h4>Customer List </h4>
                            </div>
                            <div class="box_right d-flex lms_block">
                                <div class="search_field_2">
                                    <div class="search_inner">
                                        <form Active="#">
                                            <div class="search_field">
                                                <input id="term" type="text" placeholder="Search content here..." onkeyup="myFunction()">
                                            </div>
                                            <button type="button"> <i class="ti-search"></i> </button>
                                        </form>
                                    </div>
                                </div>
                                <div class="add_button ml-10">
                                    <a id="btnSearch" href="#" data-toggle="modal" data-target="#addcategory" class="bta_1">search</a>
                                </div>
                            </div>
                        </div>
                        <div class="QA_table mb_30">
                            <table id="DataTables_Table_0" class="table lms_table_active ">
                                <thead>
                                    <tr>
                                        <th scope="col">id</th>
                                        <th scope="col">Username</th>
                                        <th scope="col">Email Address</th>
                                        <th scope="col">Mobile No</th>
                                        ... <th scope="col">...</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    ... <tr> ... </tr>
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Figure 30 Customer

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Home

```
Index.cshtml
Layout = "~/Views/Shared/_Layout.cshtml";
 ViewData["Title"] = "Home Page";
}

<section class="sec-slider">
<div class="owl-carousel owl-theme">
<div class="item">

</div>
</div>
</section>
<section class="sec-product">
<h2>Featured Books</h2>
<img alt="A grid of book covers with one book highlighted in red." data-bbox="106 450 890 886"/>

```

Figure 31 Home

Login

```
Index.cshtml # X Index.cshtml ViewCustomer.cshtml Index.cshtml Index.cshtml EditBook.cshtml Index.cshtml << Solution Explorer
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
    ViewData["Title"] = "Login";
}

<link rel="stylesheet" href="~/Assets/css/sign.css">
<section class="info_sign">
    <div class="container">
        <div class="row">
            <div class="col-lg-6 info_line">
                <div class="wrap col-lg-12 col-md-12 col-sm-12 col-xs-12">
                    <div class="form-row">
                        <div style="text-align: left;" class="form-group col-md-12">
                            <h2>Login</h2>
                            <p>Welcome Back!, To keep connected with us please login with your personal info.</p>
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="col-md-12">
                            <div id="danger" class="alert alert-danger" role="alert">Please enter a email.</div>
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="form-group col-md-12">
                            <label for="logEmail">Email<sup>x</sup></label>
                            <input type="email" class="form-control" id="logEmail">
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="form-group col-md-12">
                            <label for="logPass">Password<sup>x</sup></label>
                            <div class="input-group" id="show_hide_password">
                                <input type="password" class="form-control" id="logPass">
                                <div class="input-group-append">
                                    <a class="input-group-text" href="#">
```

Figure 32 Login

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Order

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `ViewOrder.cshtml` file, which contains HTML and C# code for a search and list view. On the right, the Solution Explorer window shows the project structure with several files under the `Order` folder, including `ViewOrder.cshtml` and `ViewOrderDetail.cshtml`. The status bar at the bottom indicates "THIS SPACE IS INTENTIONALLY LEFT BLANK".

```
ViewOrder.cshtml
...
<div class="main_content_inner">
    Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
}
</div>
<div class="main_content_inner">
    <div class="container-fluid p-0">
        <div class="row">
            <div class="col-lg-12">
                <div class="white_card card_height_100 mb_30 pt-4">
                    <div class="white_card_body">
                        <div class="QA_section">
                            <div class="white_box_tittle list_header">
                                <h4>Book Order List </h4>
                            <div class="box_right d-flex lms_block">
                                <div class="search_inner">
                                    <form Active="#">
                                        <div class="search_field">
                                            <input id="term" type="text" placeholder="Search content here..." onkeyup="myFunction()" name="term">
                                        </div>
                                        <button type="button"> <i class="ti-search"></i> </button>
                                    </form>
                                </div>
                                <div class="add_button ml-10">
                                    <a href="#" data-toggle="modal" data-target="#addcategory" class="btn_1">search</a>
                                </div>
                            </div>
                        </div>
                        <div class="QA_table mb_30">
                            <table id="DataTables_Table_0" class="table lms_table_active">
                                <thead>
                                    <tr>
                                        <th scope="col">id</th>
                                        <th scope="col">OrderDate</th>
                                        <th scope="col">Customer</th>
                                        <th scope="col">DeliveryMethod</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    <tr>
                                        <td>1</td>
                                        <td>2023-10-01</td>
                                        <td>John Doe</td>
                                        <td>Standard</td>
                                    </tr>
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

Figure 33 Order

Register

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `Index.cshtml` file, which contains HTML and C# code for a registration form. On the right, the Solution Explorer window shows the project structure with several files under the `Register` folder, including `Index.cshtml`. The status bar at the bottom indicates "THIS SPACE IS INTENTIONALLY LEFT BLANK".

```
Index.cshtml
...
<div class="info_line">
    Layout = "~/Views/Shared/_Layout.cshtml";
    ViewData["Title"] = "Register";
}
</div>
<link rel="stylesheet" href("~/Assets/css/sign.css")>
<section class="info_sign">
    <div class="container">
        <div class="row">
            <div class="col-lg-6 info_line">
                <div class="wrap col-lg-12 col-md-12 col-sm-12 col-xs-12">
                    <div class="form-row">
                        <div style="text-align: left;" class="form-group col-md-12">
                            <h2>Login</h2>
                            <p>Welcome Back!, To keep connected with us please login with your personal info.</p>
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="col-md-12">
                            <div id="danger" class="alert alert-danger" role="alert">Please enter a email.</div>
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="form-group col-md-12">
                            <label for="logEmail">Email<sup>x</sup></label>
                            <input type="email" class="form-control" id="logEmail">
                        </div>
                    </div>
                    <div class="form-row">
                        <div class="form-group col-md-12">
                            <label for="logPass">Password<sup>x</sup></label>
                            <div class="input-group">
                                <input type="password" class="form-control" id="logPass">
                                <div class="input-group-append">
                                    <a class="input-group-text" href="#"><i class="fa fa-eye-slash" aria-hidden="true"></i></a>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

Figure 34 Register

Report

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `BookStockReport.cshtml` file, which contains C# code and HTML for generating a book stock report. The code includes a `Layout` reference to a shared layout file. In the center, the Solution Explorer window lists various files and folders related to the project, such as `Cart`, `Checkout`, `Contact`, `Customer`, `Home`, `Login`, `Order`, `Register`, and `Reports`. The `Reports` folder is currently selected, and the `BookStockReport.cshtml` file is highlighted.

Figure 35 Report

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

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `_Layout.cshtml` file from the `Shared` folder. This file serves as a master layout for both the admin and customer sites, containing boilerplate HTML, CSS links, and JavaScript imports. In the center, the Solution Explorer window lists the same project structure as Figure 35, with the `Shared` folder expanded to show the `_Layout.cshtml` file.

Figure 36 Shared

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

User

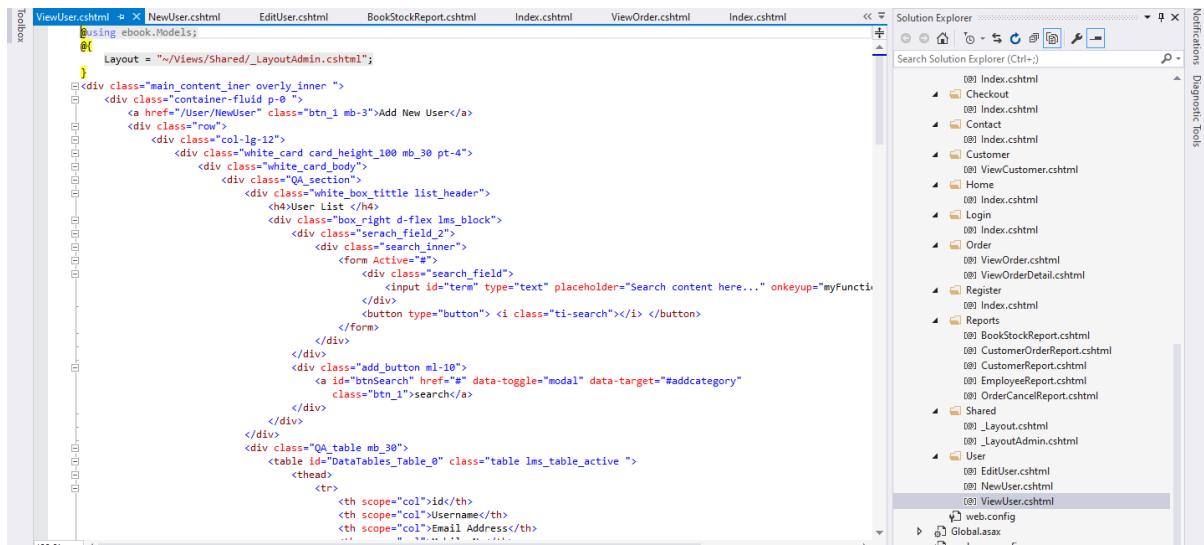


Figure 37 User

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

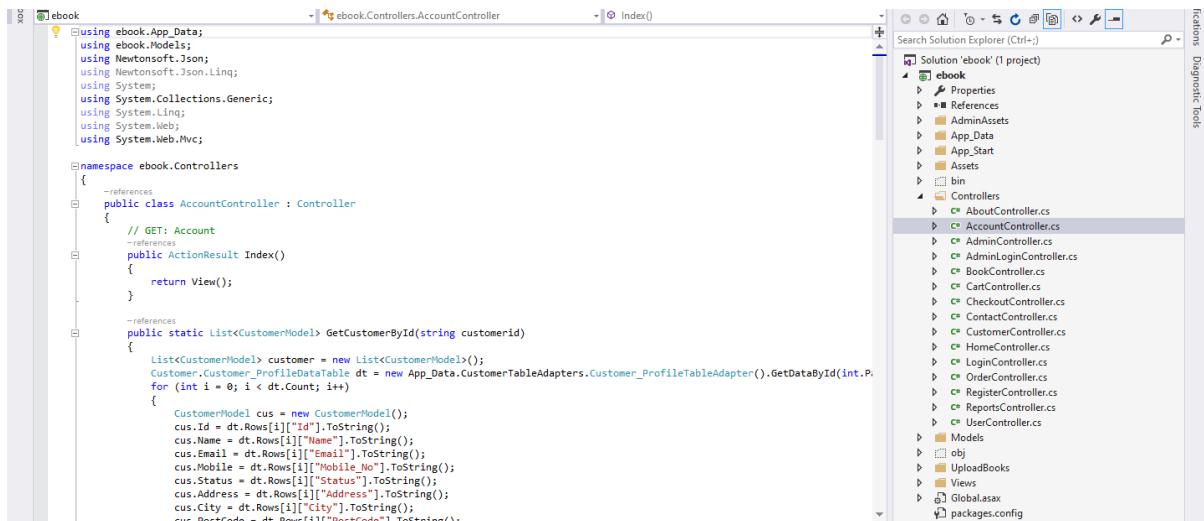


Figure 38 AccountController

AdminLoginController

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for AdminLoginController.cs, which contains C# code for handling user login. The right pane shows the Solution Explorer with the project structure for 'ebook' containing various controllers like AboutController.cs, AccountController.cs, AdminController.cs, AdminLoginController.cs, BookController.cs, CartController.cs, CheckoutController.cs, ContactController.cs, CustomerController.cs, HomeController.cs, LoginController.cs, OrderController.cs, RegisterController.cs, ReportsController.cs, and UserController.cs. The AdminLoginController.cs file is currently selected.

```
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", email);
                if (dt.Count == 0)
                {
                    msg = "false";
                }
                else
                {
                    msg = "true";
                }
            }
        }
    }
}
```

Figure 39 AdminLoginController

BookController

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for BookController.cs, which handles book details. The right pane shows the Solution Explorer with the project structure for 'ebook' containing various controllers like AboutController.cs, AccountController.cs, AdminController.cs, AdminLoginController.cs, BookController.cs, CartController.cs, CheckoutController.cs, ContactController.cs, CustomerController.cs, HomeController.cs, LoginController.cs, OrderController.cs, RegisterController.cs, ReportsController.cs, and UserController.cs. The BookController.cs file is currently selected.

```
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"] != 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

The screenshot shows the Visual Studio IDE with the CartController.cs file open in the code editor. The code implements a controller for managing a shopping cart. It includes methods for displaying the index of items in the cart and confirming an order. The code uses the NewtonSoft.Json library for JSON serialization and the System.Web.Mvc framework.

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

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

        // POST: Cart/ConfirmOrder
        public ActionResult ConfirmOrder(string deliverymed, string deliverycharge, string customerid)
        {
            //Fetch the Cookie using its Key.
            HttpCookie nameCookie = Request.Cookies["_inok"];

            //If Cookie exists fetch its value.
            string s = nameCookie != null ? nameCookie.Value : "undefined";
            DateTime date = DateTime.Now;
            int order_id = Convert.ToInt32(new App_Data.OrderTableAdapters.Order_Header_ProfileTableAdapter().InsertOrderHeader(date, date, deliverymed, deliverycharge, customerid));
            string[] strArr = s.Split('/');
            for (int i = 0; i < strArr.Length; i++)
            {
                ...
            }
        }
    }
}
```

Figure 41 CartController

CustomerController

The screenshot shows the Visual Studio IDE with the CustomerController.cs file open in the code editor. This controller handles customer-related operations, including viewing all customers and updating their status. It uses the App_Data.CustomerTableAdapters.Customer_ProfileTableAdapter to interact with the database.

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

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

        // GET: Customer/ViewCustomer
        public ActionResult ViewCustomer()
        {
            List<CustomerModel> customer = GetAllCustomers();
            ViewBag.Customers = customer.ToList();
            return View();
        }

        [HttpPost]
        // Post: Customer/UpdateStatus
        public ActionResult UpdateStatus(string Id, string status)
        {
            new App_Data.CustomerTableAdapters.Customer_ProfileTableAdapter().UpdateCustomerStatus(int.Parse(status), int.Parse(Id));
            List<CustomerModel> customer = GetAllCustomers();
            ViewBag.Customers = customer.ToList();
        }
    }
}
```

Figure 42 CustomerController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

HomeController

The screenshot shows the Visual Studio IDE with the HomeController.cs file open in the code editor. The code implements the Index() action method, which retrieves all books from the database and returns them as a view. The Solution Explorer on the right shows the project structure, including other controllers like AboutController, AccountController, AdminController, AdminLoginController, BookController, CartController, CheckoutController, ContactController, CustomerController, LoginController, OrderController, RegisterController, ReportsController, and UserController.

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

namespace ebook.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            List<BookModel> book = GetAllBooks();
            ViewBag.FeaturedBooks = book.ToList();

            ViewBag.Books = book.Take(8).ToList();
            return View();
        }

        public static List<BookModel> GetAllBooks()
        {
            List<BookModel> books = new List<BookModel>();
            Book_Book_ProfileDataTable dt = new App_Data.BookTableAdapters.Book_ProfileTableAdapter().GetData();
            for (int i = 0; i < dt.Count; i++)
            {
                BookModel book = new BookModel();
                book.Id = dt.Rows[i]["Id"].ToString();
                book.Name = dt.Rows[i]["Name"].ToString();
                book.ISBN = dt.Rows[i]["ISBN"].ToString();

                string img = dt.Rows[i]["Image"].ToString();
                string[] imgPath = img.Split('/');
                string imgPath2 = imgPath[imgPath.Length - 1];
            }
        }
    }
}
```

Figure 43 HomeController

OrderController

The screenshot shows the Visual Studio IDE with the OrderController.cs file open in the code editor. The controller handles three actions: Index(), ViewOrder(), and ViewOrderDetail(). The Index() method retrieves all order headers and returns them as a view. The ViewOrder() method retrieves a specific order header by ID and returns it as a view. The ViewOrderDetail() method retrieves a specific order header by ID and returns its details as a view. The Solution Explorer on the right shows the project structure, including other controllers like AboutController, AccountController, AdminController, AdminLoginController, BookController, CartController, CheckoutController, ContactController, CustomerController, LoginController, HomeController, OrderController, RegisterController, ReportsController, and UserController.

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

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

        // GET: Order/ViewOrder
        public ActionResult ViewOrder()
        {
            List<OrderHeaderModel> order = GetAllOrderHeaders();
            ViewBag.Orders = order.ToList();
            return View();
        }

        // GET: Order/ViewOrderDetail
        public ActionResult ViewOrderDetail()
        {
            string OrderId = Request.QueryString["id"].ToString();
            if(OrderId != null)
            {
                OrderHeaderModel orderHeader = GetAllOrderHeaders().Find(x => x.Id == int.Parse(OrderId));
                ViewBag.OrderHeader = orderHeader;
            }
            return View();
        }
    }
}
```

Figure 44 OrderController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

RegisterController

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for `RegisterController.cs`, which contains C# code for a controller. The right pane shows the `Solution Explorer` with the project structure for 'ebook'.

```
RegisterController.cs // Active tab
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 RegisterController : Controller
    {
        // GET: Register
        public ActionResult Index()
        {
            return View();
        }

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

`Solution Explorer` (Ctrl+Shift+S)

- Project: ebook (1 project)
 - ebook
 - Properties
 - References
 - AdminAssets
 - App_Data
 - App_Start
 - Assets
 - bin
 - Controllers
 - AboutController.cs
 - AccountController.cs
 - AdminController.cs
 - AdminLoginController.cs
 - BookController.cs
 - CartController.cs
 - CheckoutController.cs
 - ContactController.cs
 - CustomerController.cs
 - HomeController.cs
 - LoginController.cs
 - OrderController.cs
 - RegisterController.cs
 - ReportController.cs
 - ReportController.cs
 - Models
 - UploadBooks
 - Views
 - Global.asax
 - packages.config

Figure 45 RegisterController

ReportsController

The screenshot shows the Visual Studio IDE interface. The code editor on the left displays the `ReportsController.cs` file, which contains C# code for a controller. The Solution Explorer on the right shows the project structure for a solution named 'ebook'. The project includes various controllers like AboutController, AccountController, AdminController, AdminLoginController, BookController, CartController, CheckoutController, ContactController, CustomerController, HomeController, LoginController, OrderController, RegisterController, ReportsController, and UserController. It also includes Models, App_Start, and Global.asax files, along with AdminAssets, Assets, bin, and obj folders.

```
ReportsController.cs  ✘ RegisterController.cs  OrderController.cs  LoginController.cs  HomeController.cs  CustomerController.cs  <>  Solution Explorer  Search Solution Explorer (Ctrl+.)  Solution 'ebook' (1 project)  ebook  Properties  References  AdminAssets  App_Data  App_Start  Assets  bin  Controllers  AboutController.cs  AccountController.cs  AdminController.cs  AdminLoginController.cs  BookController.cs  CartController.cs  CheckoutController.cs  ContactController.cs  CustomerController.cs  HomeController.cs  LoginController.cs  OrderController.cs  RegisterController.cs  ReportsController.cs  UserController.cs  Models  UploadBooks  Views  Global.asax  packages.config
```

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

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

        public ActionResult CustomerReport()
        {
            List<CustomerModel> customer = GetAllCustomers();
            ViewBag.Customers = customer.ToList();
            return View();
        }

        public static List<CustomerModel> GetAllCustomers()
        {
            List<CustomerModel> customer = new List<CustomerModel>();
            Customer.Customer_ProfileDataTable dt = new App_Data.CustomerTableAdapters.Customer_ProfileTableAdapter().GetDataTable();
            for (int i = 0; i < dt.Count; i++)
            {
                CustomerModel cus = new CustomerModel();
                cus.Id = dt.Rows[i]["Id"].ToString();
                cus.
```

Figure 46 ReportsController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

UserController

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes 'File', 'Edit', 'View', 'Project', 'Server Explorer', 'Toolbox', 'Solution Explorer', 'Task List', 'Properties', 'Toolbox', 'Help', and 'Exit'. The 'Toolbox' tab is selected. Below the menu is a toolbar with icons for 'New', 'Open', 'Save', 'Print', 'Build', 'Run', 'Find', 'Replace', and 'Format'. The main window has tabs for 'UserController.cs', 'ReportsController.cs', 'RegisterController.cs', 'OrderController.cs', 'LoginController.cs', and 'HomeController.cs'. The 'UserController.cs' tab is active, displaying the following C# code:

```
using ebook.App_Data;
using ebook.Models;
using System;
using System.Collections.Generic;
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()
        {
            ListEmployeeModel 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));
        }
    }
}
```

The 'Solution Explorer' pane on the right shows the project structure for 'ebook' (1 project). It includes the 'ebook' folder containing 'Properties', 'References', 'App_Data', 'App_Start', 'Assets', 'bin', 'Controllers' (containing 'AboutController.cs', 'AccountController.cs', 'AdminController.cs', 'AdminLoginController.cs', 'BookController.cs', 'CartController.cs', 'CheckoutController.cs', 'ContactController.cs', 'CustomerController.cs', 'HomeController.cs', 'LoginController.cs', 'OrderController.cs', 'RegisterController.cs', 'UserController.cs'), 'Models', 'obj', 'UploadBooks', 'Views', 'Global.asax', and 'packages.config'.

Figure 47 UserController

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

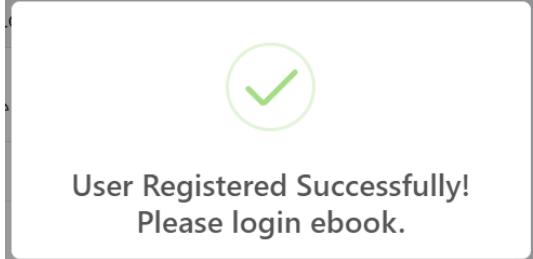
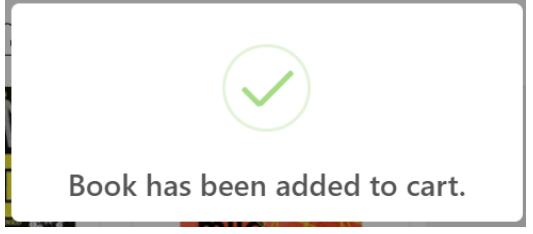
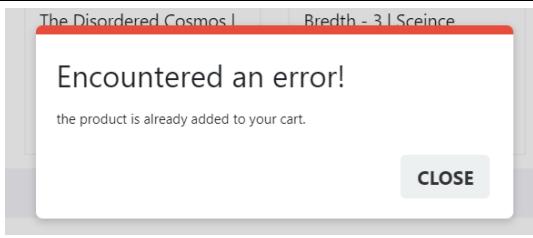
05) TESTING.

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

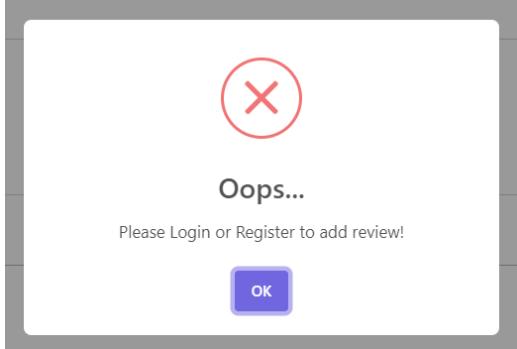
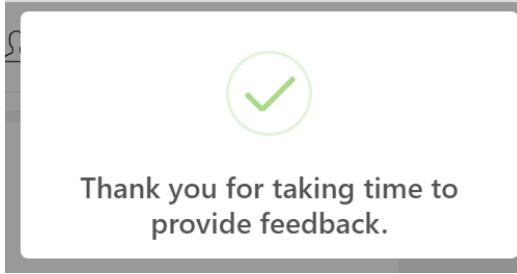
4	Customer giving book feedback without login to the system.	Should show alert saying should login to the system to provide review.		Pass
5	Customer giving book feedback after login to the system.	Should show feedback submitted successfully message.		Pass

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.	 Customer has been activated successfully.	Pass
2	User filling all the details of book and clicking add button.	Should show book successfully added message.	 Book Added Succefully!	Pass
3	Admin filling all the details of new system user and clicking add button.	Should show user successfully added message.	 User Added Succefully!	Pass

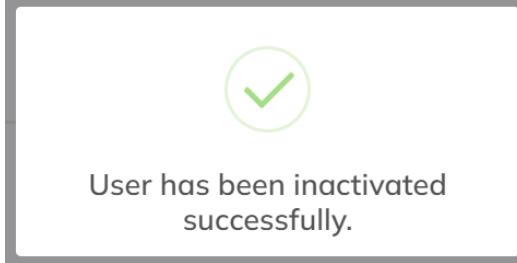
4	Admin clicking deactivate button for a particular user in user management.	Should show user deactivated successfully message.		Pass																																																																		
5	User clicking generate report button in reports pages.	Should generate the selected report in pdf format.	 <p>Book Stock Report</p> <table border="1"><thead><tr><th>Id</th><th>Name</th><th>ISBN</th><th>Price</th><th>Availability</th><th>Category</th></tr></thead><tbody><tr><td>14</td><td>Breath - 1</td><td>978-3-16-148410-0</td><td>1200.00</td><td>Available</td><td>Science</td></tr><tr><td>15</td><td>Breath - 2</td><td>978-3-16-148410-1</td><td>1300.00</td><td>Available</td><td>Science</td></tr><tr><td>16</td><td>The Disordered Cosmos</td><td>978-3-16-148415-4</td><td>1600.00</td><td>Available</td><td>Science</td></tr><tr><td>17</td><td>Breath - 3</td><td>978-3-16-148410-8</td><td>1300.00</td><td>Available</td><td>Science</td></tr><tr><td>18</td><td>The Science Book</td><td>978-3-16-148413-9</td><td>1800.00</td><td>Available</td><td>Science</td></tr><tr><td>19</td><td>Math without numbers</td><td>978-3-16-148410-0</td><td>1750.00</td><td>Available</td><td>Maths</td></tr><tr><td>20</td><td>Math Without Numbers 2</td><td>978-3-16-148415-7</td><td>1800.00</td><td>Available</td><td>Maths</td></tr><tr><td>21</td><td>Beyond Infinity</td><td>978-3-16-148416-9</td><td>1850.00</td><td>Available</td><td>Maths</td></tr><tr><td>22</td><td>Numbers Dont Lie</td><td>978-3-16-148418-8</td><td>1300.00</td><td>Available</td><td>Maths</td></tr><tr><td>23</td><td>Humble Pi</td><td>978-3-16-148415-7</td><td>1600.00</td><td>Available</td><td>Maths</td></tr></tbody></table>	Id	Name	ISBN	Price	Availability	Category	14	Breath - 1	978-3-16-148410-0	1200.00	Available	Science	15	Breath - 2	978-3-16-148410-1	1300.00	Available	Science	16	The Disordered Cosmos	978-3-16-148415-4	1600.00	Available	Science	17	Breath - 3	978-3-16-148410-8	1300.00	Available	Science	18	The Science Book	978-3-16-148413-9	1800.00	Available	Science	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	Pass
Id	Name	ISBN	Price	Availability	Category																																																																	
14	Breath - 1	978-3-16-148410-0	1200.00	Available	Science																																																																	
15	Breath - 2	978-3-16-148410-1	1300.00	Available	Science																																																																	
16	The Disordered Cosmos	978-3-16-148415-4	1600.00	Available	Science																																																																	
17	Breath - 3	978-3-16-148410-8	1300.00	Available	Science																																																																	
18	The Science Book	978-3-16-148413-9	1800.00	Available	Science																																																																	
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																																																																	

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.

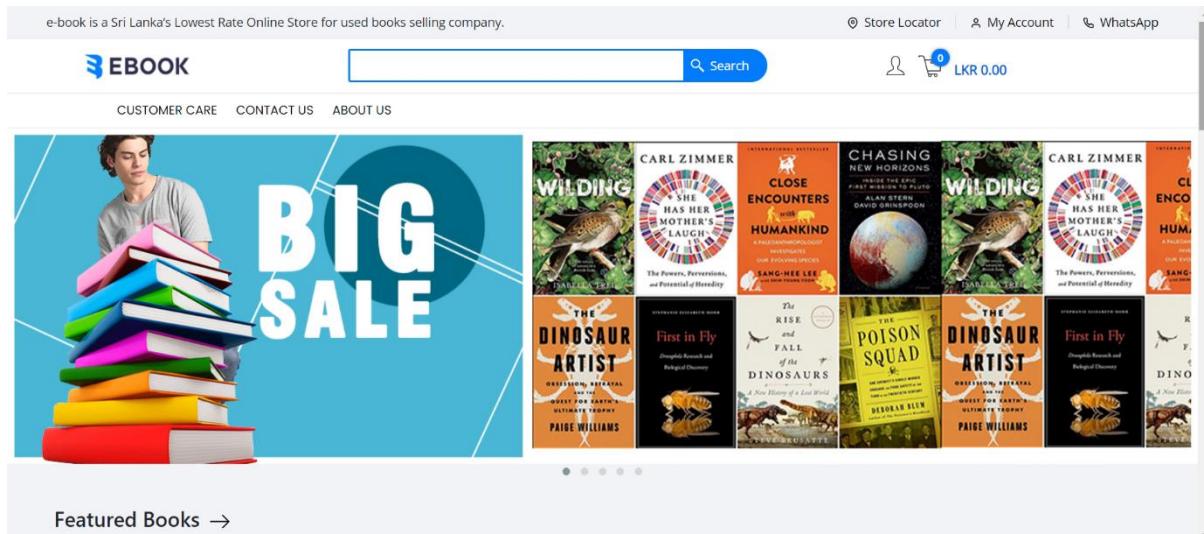


Figure 48 Home-1

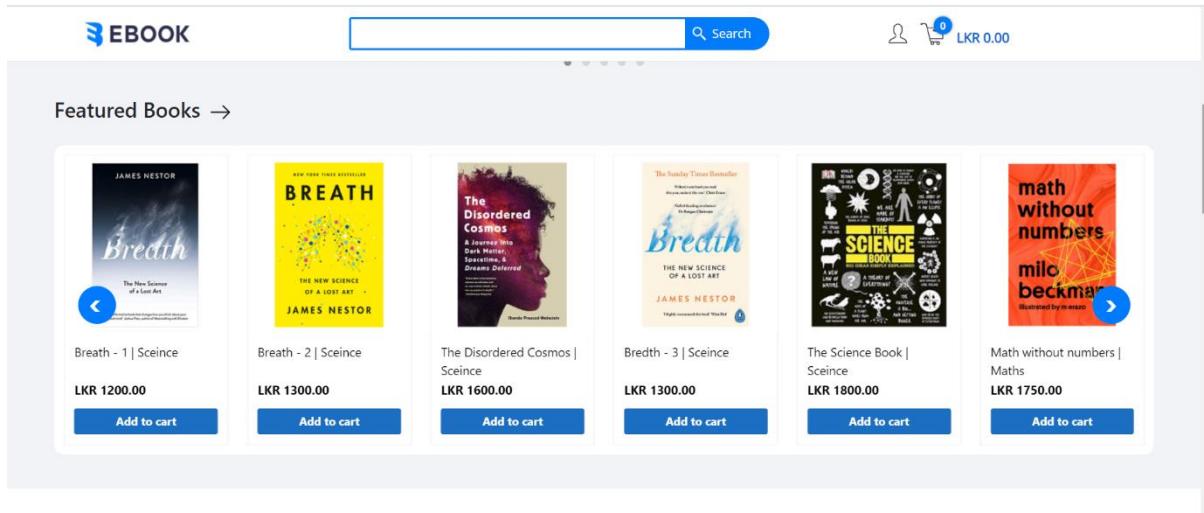


Figure 49 Home-2

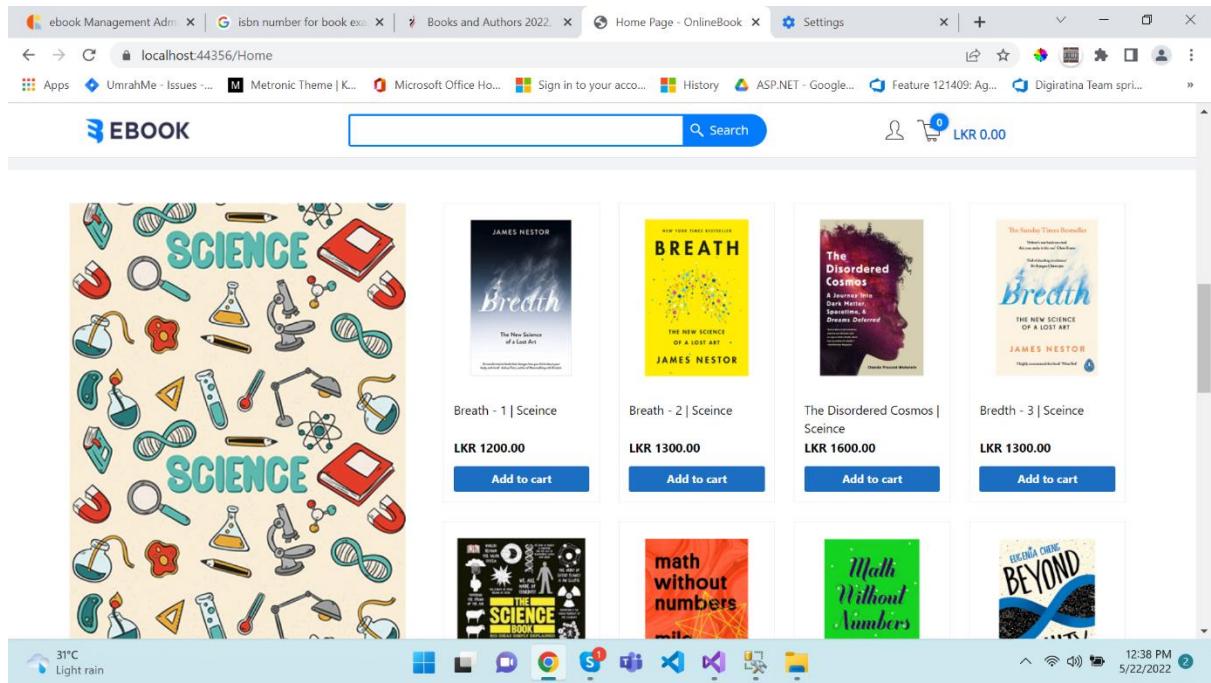


Figure 50 Home-3

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

6.1.2 Login & Register.

The screenshot shows the EBOOK website's customer login and registration interface. At the top, there is a navigation bar with links for CUSTOMER CARE, CONTACT US, ABOUT US, SIGNUP, and LOGIN. A search bar is also present. On the right side of the header, there is a user icon, a shopping cart icon with a count of 0, and a balance of LKR 0.00.

Login
Welcome Back!, To keep connected with us please login with your personal info.

Email X
Password
 Remember Me

Register
Please fill up the personal informations to serve you well. If you have an account!, please login

Full Name
E-mail Mobile Number
Address1 Address2
Country City
Password Confirm Password

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.

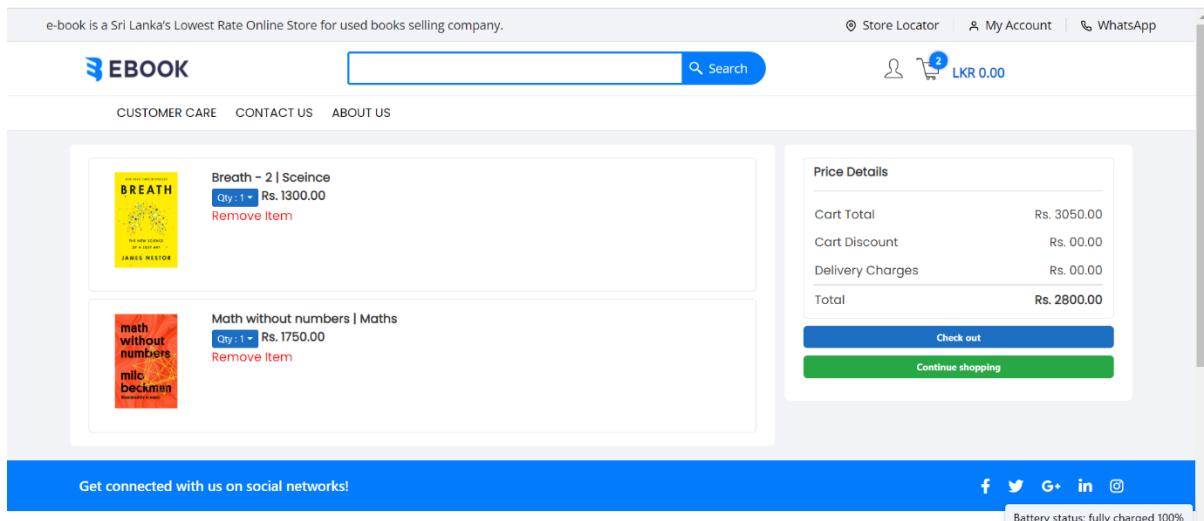


Figure 52 Cart

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:

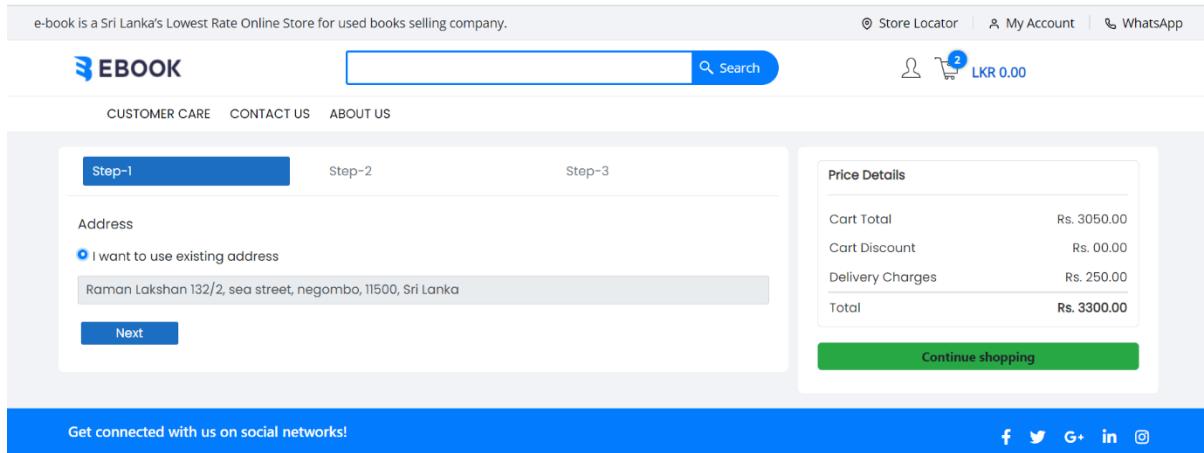


Figure 53 Checkout - Step1

In First step customer delivery address will be shown.

Step 2:

The screenshot shows the EBOOK website's checkout process. At the top, there is a navigation bar with links for Store Locator, My Account, and WhatsApp. The main content area has tabs for Step-1, Step-2 (which is active), and Step-3. Under Step-2, there is a section for 'Delivery Method' with two radio button options: 'Home Delivery (Cash On Delivery)' (selected) and 'Pick up at store'. Below this is a text input field for 'Add Commands About Your Orders'. To the right, a 'Price Details' box shows the following breakdown:

Cart Total	Rs. 3050.00
Cart Discount	Rs. 0.00
Delivery Charges	Rs. 250.00
Total	Rs. 3300.00

At the bottom of the page are 'Back' and 'Next' buttons, and a green 'Continue shopping' button.

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:

The screenshot shows the EBOOK website's checkout process. At the top, there is a navigation bar with links for Store Locator, My Account, and WhatsApp. The main content area has tabs for Step-1, Step-2 (which is active), and Step-3. Under Step-3, there is a table of ordered products:

#	Product Name	Qty	Unit Price	Total
1	BREATH - 2 Sceince	1	1300.00	1300.00
2	Math without numbers Maths	1	1750.00	1750.00

Below the table, there is a summary of the total amount:

Sub Total	Rs. 3050.00
Delivery Charge	Rs. 250.00
Total	Rs. 3300.00

At the bottom of the page are 'Back' and 'Confirm' buttons, and a green 'Continue shopping' button.

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

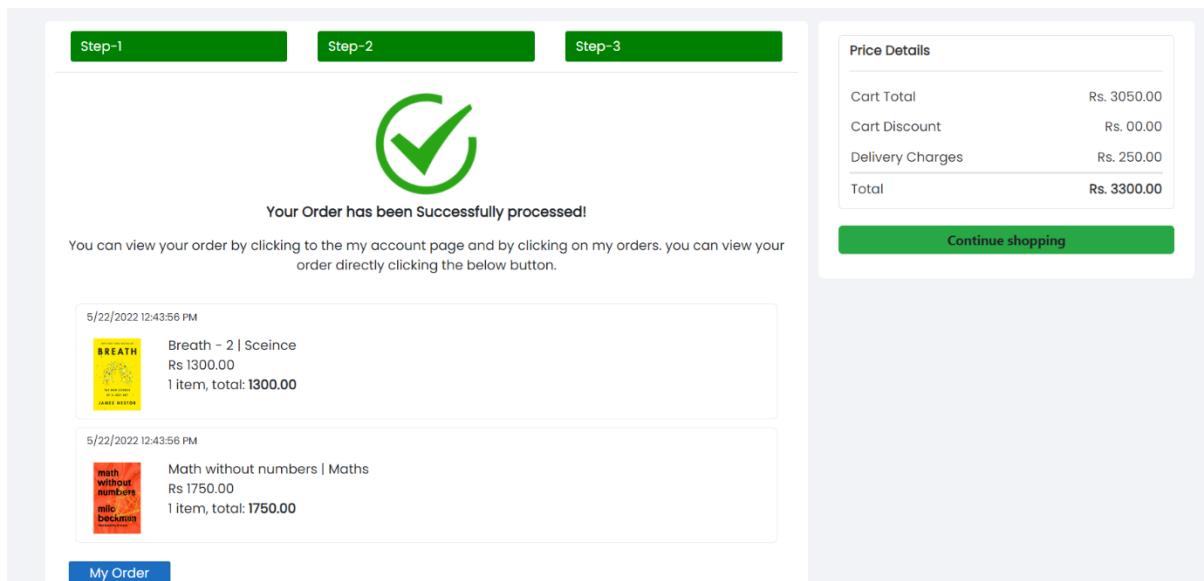


Figure 56 Order Success Message

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

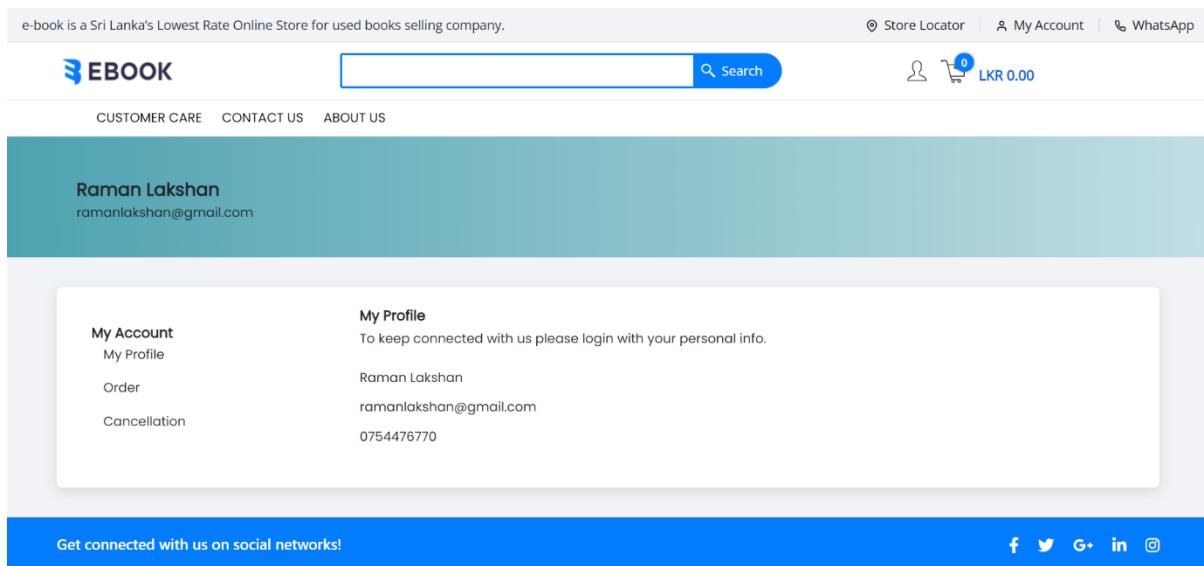


Figure 57 My Account

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

6.1.6 Review

The screenshot shows a web-based application for reviewing books. At the top, there's a header with a logo labeled 'EBOOK', a search bar with a 'Search' button, and user icons. Below the header, a section titled 'Reviews' displays two existing reviews and a form for adding a new one.

Review 1: Posted by 'satheeskaran' (sathees@gmail.com) with a 5-star rating. The review text: "The strength of this excellent textbook is that it overviews the broad field of deterministic and stochastic operations research in a clear, easy to follow way with a rich collection of examples, applications and exercises".

Review 2: Posted by 'Raman Lakshan' (ramanlakshan@gmail.com) with a 5-star rating. The review text: "I am a mathematician with an interest in the history of mathematics and of mathematicians. As such, I loved this book. It is a fascinating tour of the life and work of Norbert Wiener."

Add Review Form: A text input field with a placeholder 'Add Review About book' and a 'Search' button. Below it is a 'Submit' button labeled 'Add 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.

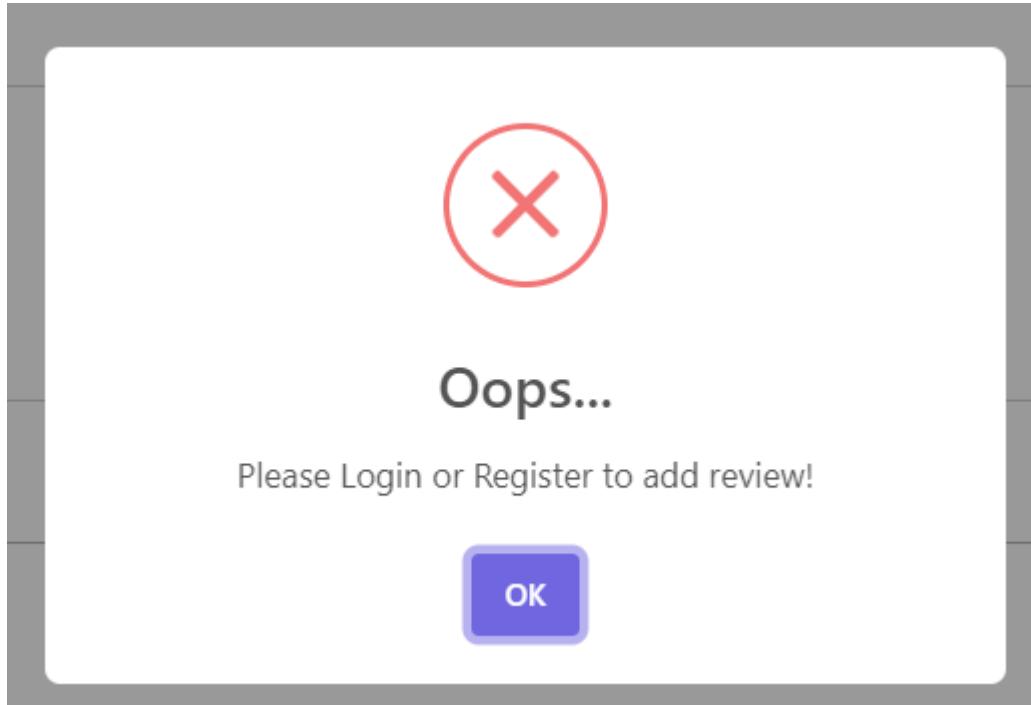


Figure 59 Review Error Message

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

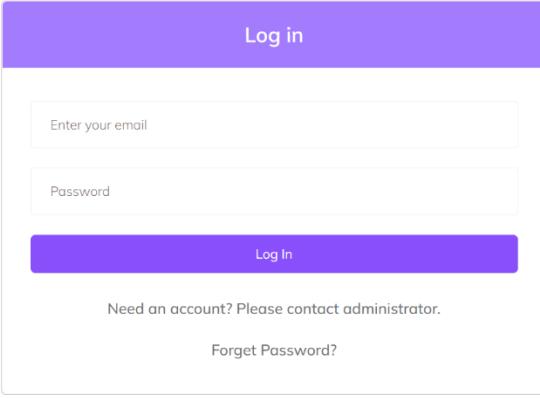
6.1.7 Steps to Order Books and Give Review.

- 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.



The image shows a login form titled "Log in". It has two text input fields: "Enter your email" and "Password". Below the fields is a purple "Log In" button. Underneath the button, there is a link "Need an account? Please contact administrator." and another link "Forgot Password?".

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.

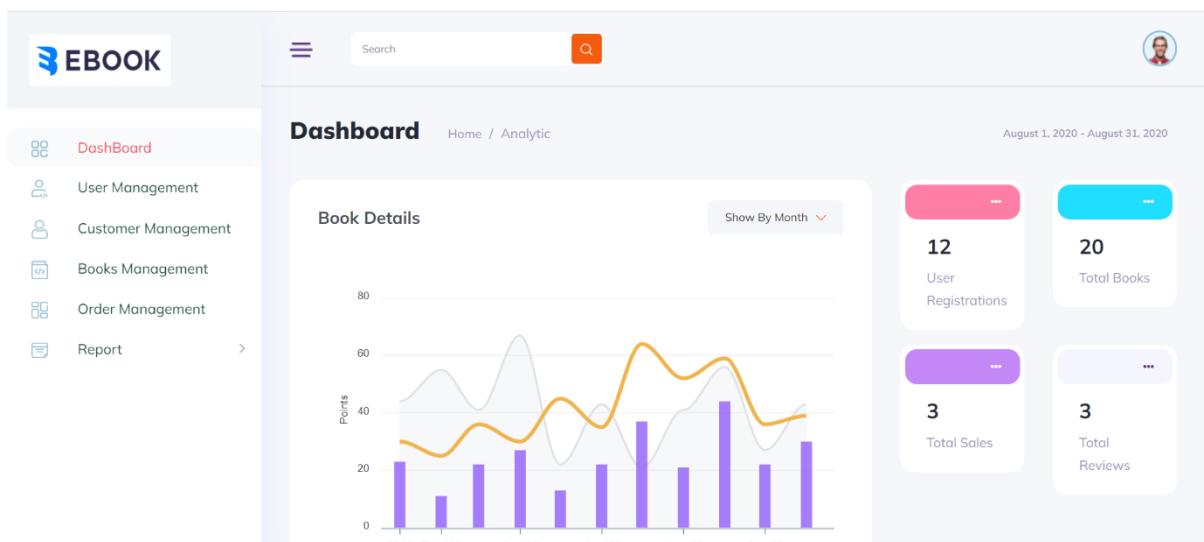


Figure 61 Dashboard

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

6.2.3 User Management.

This process consists of two pages.

View Page:

The screenshot shows a user management interface for an 'EBOOK' application. On the left, a sidebar menu includes 'DashBoard', 'User Management' (which is highlighted in red), 'Customer Management', 'Books Management', 'Order Management', and 'Report'. A purple 'Add New User' button is located at the top right of the sidebar. The main area is titled 'User List' and contains a table with columns: Id, Username, Email Address, Mobile No, Role, Status, and Action. The table has 5 entries. The 'Status' column uses green and red buttons labeled 'Active' and 'Inactive' respectively. The 'Action' column contains blue buttons. A search bar with placeholder 'Search content here...' and a 'Search' button are at the top right of the table area. At the bottom, it says 'Showing 1 to 5 of 5 entries' and has navigation buttons for page 1.

ID	Username	Email Address	Mobile No	Role	Status	Action
1	sathees	satheeskaran1995@gmail.com	0767050242	Admin	Active	Inactive
2	karan	karan1995@gmail.com	0767050242	Admin	Inactive	Active
3	siva	siva1995@gmail.com	0786169566	Admin	Active	Inactive
4	siva	siva@gmail.com	0786169566	Admin	Active	Inactive
5	sasasasd			Admin	Active	Inactive

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:

The screenshot shows the EBOOK application interface. At the top left is the logo 'EBOOK'. To its right is a search bar with a magnifying glass icon. On the far right is a user profile icon. The left sidebar contains navigation links: 'DashBoard', 'User Management', 'Customer Management', 'Books Management', 'Order Management', and 'Report'. The main content area is titled 'Add New User'. It contains four input fields: 'Name' and 'Email Address' (both with placeholder 'Name'), 'Mobile No' and 'Password' (both with placeholder 'Mobile No'). Below these is a dropdown menu labeled 'Role' with the option 'Select Role'. At the bottom is a large blue button labeled 'Add User'.

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.

Customer List							Search
Search content here...						Search	
	ID	Username	Email Address	Mobile No	Address	Status	Action
1	sadasda	siva@gmail.com	0786169566	saasas	Inactive	Active	
2	jhgkg	siva@gmail.com	07861789677	jgkg	Inactive	Active	
3	kkkj	siva@gmail.com	0786456789	hjfh	Active	Inactive	
4	kkkj	siva@gmail.com	0786456789	hjfh	Inactive	Active	
5	kjhjh	siva@gmail.com	07861669566	hjff	Inactive	Active	
6	satheeskaran	satheeskaran1995@gmail.com	0786169566	sadasfdkjg	Active	Inactive	
7	satheeskaran	sathees@gmail.com	0786169677	jktiuyr	Active	Inactive	
8	gfds	sdfghjk@gmail.com	0786169566	ghjkkjhgf	Active	Inactive	
9	lakshan	lakshan@gmail.com	0876543455	dscfadf	Active	Inactive	

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:

The screenshot shows the 'Books List' page of the EBOOK application. The left sidebar has a navigation menu with items: Dashboard, User Management, Customer Management, Books Management (which is highlighted in red), Order Management, and Report. The main area is titled 'Books List' and contains a table with the following data:

ID	Name	ISBN	Price	Availability	Category	Action
4	Breath - 1	978-3-16-148410-0	1200.00	Available	Science	
5	Breath - 2	978-3-16-148410-1	1300.00	Available	Science	
6	The Disordered Cosmos	978-3-16-148415-4	1600.00	Available	Science	
7	Breath - 3	978-3-16-148410-8	1300.00	Available	Science	
8	The Science Book	978-3-16-148413-9	1800.00	Available	Science	

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:

The screenshot shows a web-based application interface for managing books. At the top left is a logo for 'EBOOK'. On the left side, there is a sidebar menu with the following items: Dashboard (selected), User Management, Customer Management, Books Management, Order Management, and Report. Below the sidebar is a URL bar containing 'https://localhost:44356/Admin/Dashboard'. The main content area is titled 'Edit' and contains a form for adding a new book. The form fields include:

- Image: A placeholder for a book cover image.
- Choose File: A button to select a file, with a message 'No file chosen'.
- Book Name: An input field labeled 'Name'.
- ISBN NO: An input field labeled 'ISBN'.
- Publisher: An input field labeled 'Publisher'.
- Published Year: A date input field labeled 'mm/dd/yyyy'.
- Category: A dropdown menu labeled 'Select Category'.
- Availability: A dropdown menu labeled 'Select Availability'.
- Price: An input field labeled 'Price'.

Figure 66 Book Management – Edit

This interface works 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:

The screenshot shows the 'Book Order List' page. At the top left is the 'EBOOK' logo. To its right is a search bar with placeholder text 'Search content here...' and a purple 'Search' button. On the far left is a sidebar with navigation links: Dashboard, User Management, Customer Management, Books Management, Order Management (which is highlighted in red), and Report. The main area contains a table titled 'Book Order List' with the following columns: Id, OrderDate, Customer, DeliveryMethod, DeliveryCharge, Status, and Action. The table displays 7 entries. The first entry has an Id of 1, OrderDate of 5/21/2022 5:29:00 PM, Customer as satheeskaran, DeliveryMethod as HOME DELIVERY, DeliveryCharge as 250.00, Status as Pending, and Action buttons for edit and delete. The last entry has an Id of 7, OrderDate of 5/22/2022 12:43:56 PM, Customer as Raman Lakshan, DeliveryMethod as HOME DELIVERY, DeliveryCharge as 250.00, Status as Pending, and Action buttons for edit and delete. Below the table, a message says 'Showing 1 to 6 of 6 entries'. At the bottom right are navigation arrows for the table.

ID	Order Date	Customer	Delivery Method	Delivery Charge	Status	Action
1	5/21/2022 5:29:00 PM	satheeskaran	HOME DELIVERY	250.00	Pending	
3	5/21/2022 5:37:08 PM	satheeskaran	HOME DELIVERY	250.00	Pending	
4	5/21/2022 5:47:32 PM	satheeskaran	HOME DELIVERY	250.00	Pending	
5	5/22/2022 1:12:44 AM	satheeskaran	PICK UP	250.00	Pending	
6	5/22/2022 12:36:04 PM	Raman Lakshan	HOME DELIVERY	250.00	Pending	
7	5/22/2022 12:43:56 PM	Raman Lakshan	HOME DELIVERY	250.00	Pending	

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:

The screenshot shows the 'Order Detail List' page. At the top left is the 'EBOOK' logo. To its right is a search bar with placeholder text 'Search' and a magnifying glass icon. On the far left is a sidebar with navigation links: Dashboard, User Management, Customer Management, Books Management, Order Management (which is highlighted in red), and Report. The main area contains a table titled 'Order Detail List' with the following columns: Id, Book, Category, Qty, Rate, and Amount. The table displays 2 entries. The first entry has an Id of 8, Book as Breath - 1, Category as Sceince, Qty as 1.000, Rate as 1200.000, and Amount as 1200.000. The second entry has an Id of 9, Book as The Science Book, Category as Sceince, Qty as 1.000, Rate as 1800.000, and Amount as 1800.000. Below the table, a message says 'Showing 1 to 2 of 2 entries'. At the bottom right are navigation arrows for the table.

ID	Book	Category	Qty	Rate	Amount
8	Breath - 1	Sceince	1.000	1200.000	1200.000
9	The Science Book	Sceince	1.000	1800.000	1800.000

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

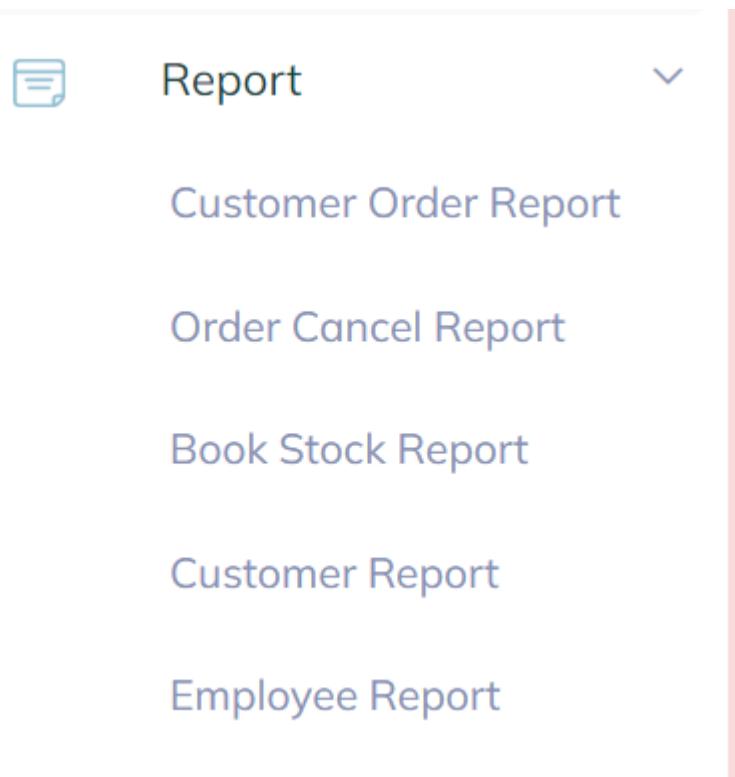


Figure 69 Reports Section

[THIS SPACE IS INTENTIONALLY LEFT BLANK]

Customer Order Report:

This report will show all the customers active orders.

The screenshot shows the 'Customer Order Report' section of the EBOOK application. On the left, there is a sidebar with navigation links: Dashboard, User Management, Customer Management, Books Management, Order Management, Report (with sub-links: Customer Order Report, Order Cancel Report, Book Stock Report, Customer Report, Employee Report), and a user profile icon. The main content area has a search bar at the top right. Below it is a table titled 'Customer Order Report' with columns: Id, OrderDate, Customer, DeliveryMethod, DeliveryCharge, and Status. Two entries are listed: Order 6 (Id 6) and Order 7 (Id 7). Both orders are from 'Raman Lakshan' via 'HOME DELIVERY' for 250.00, with a status of 'Pending'. At the bottom, it says 'Showing 1 to 2 of 2 entries' and has navigation buttons for pages 1 and 2.

ID	Order Date	Customer	Delivery Method	Delivery Charge	Status
6	5/22/2022 12:36:04 PM	Raman Lakshan	HOME DELIVERY	250.00	Pending
7	5/22/2022 12:43:56 PM	Raman Lakshan	HOME DELIVERY	250.00	Pending

Figure 70 Customer Order Report

Order Cancel Report:

This report will show all the cancelled order details.

The screenshot shows the 'Customer Order Cancel Report' section of the EBOOK application. The sidebar and layout are identical to Figure 70. The main content area has a table titled 'Customer Order Cancel Report' with columns: Id, OrderDate, Customer, DeliveryMethod, DeliveryCharge, and Status. One entry is listed: Order 8 (Id 8) from 'Raman Lakshan' via 'HOME DELIVERY' for 250.00, with a status of 'Cancel'. At the bottom, it says 'Showing 1 to 1 of 1 entries' and has navigation buttons for pages 1 and 2.

ID	Order Date	Customer	Delivery Method	Delivery Charge	Status
8	5/22/2022 4:13:41 PM	Raman Lakshan	HOME DELIVERY	250.00	Cancel

Figure 71 Order Cancel Report

Book Stock Report:

This report will show all the book details.

The screenshot shows a web application interface for managing books. On the left is a sidebar with icons and labels for Dashboard, User Management, Customer Management, Books Management, Order Management, Report (with sub-options: Customer Order Report, Order Cancel Report, Book Stock Report, Customer Report, Employee Report), and Ebook. The main content area has a header "Book Stock Report" with a search bar and a "Print Book Stock Report" button. Below is a table with the following data:

ID	Name	ISBN	Price	Availability	Category
4	Breath - 1	978-3-16-148410-0	1200.00	Available	Scince
5	Breath - 2	978-3-16-148410-1	1300.00	Available	Scince
6	The Disordered Cosmos	978-3-16-148415-4	1600.00	Available	Scince
7	Bredth - 3	978-3-16-148410-8	1300.00	Available	Scince
8	The Science Book	978-3-16-148413-9	1800.00	Available	Scince
9	Math without numbers	978-3-16-148410-0	1750.00	Available	Maths
10	Math Without Numbers 2	978-3-16-148415-7	1800.00	Available	Maths
11	Beyond Infinity	978-3-16-148416-9	1850.00	Available	Maths

Figure 72 Book Stock Report

Customer Report:

This report will show all the customer details.

The screenshot shows a web application interface for managing customers. The sidebar and main content area are similar to the Book Stock Report page, with the "Report" section expanded to show "Customer Order Report", "Order Cancel Report", "Book Stock Report", "Customer Report" (which is highlighted in red), and "Employee Report". The main content area has a header "Customer Report" with a search bar and a "Print Customer Report" button. Below is a table with the following data:

ID	Username	Email Address	Mobile No	Address	Status
1	sadasda	siva@gmail.com	0786169566	saasas	Inactive
2	jhghg	siva@gmail.com	07861789677	jgkg	Inactive
3	kkkj	siva@gmail.com	0786456789	hjfh	Active
4	kkkj	siva@gmail.com	0786456789	hjfh	Inactive
5	kjhjhjh	siva@gmail.com	07861669566	hjff	Inactive
6	satheeskaran	satheeskaran1995@gmail.com	0786169566	sadasfdkjg	Active
7	satheeskaran	sathees@gmail.com	0786169677	jktiuyr	Active

Figure 73 Customer Report

Employee Report

This report will show all the employee details.

ID	Username	Email Address	Mobile No	Role	Status
1	sathees	satheeskaran1995@gmail.com	0767050242	Admin	Active
2	karan	karan1995@gmail.com	0767050242	Admin	Inactive
3	siva	siva1995@gmail.com	0786169566	Admin	Active
4	siva	siva@gmail.com	0786169566	Admin	Active
5	sasasasd			Admin	Active

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(ms,
            {
                cs.Write(cipherBytes, 0, cipherBytes.Length);
                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 time 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