Content

Abstraction

About the proposed solution The online web application for E-book Pvt. Ltd shall be a versatile application to cater to the new capabilities that the growing company must meet in order to accommodate the demands of its broad client base and administrative systems. In lay terms, customers will be able to perform operations such as user registration, search for books, ordering and managing their accounts, and fill out feedback forms. Store administrators will be able to use various reliable tools for organizing book stocks, customers' information, orders and receiving useful and accessible client reports through the administrators' page. Leveraging ASP. Despite that net with C# guarantees the application will work with high security standard, and will have capacity to hold larger amount of data and string up the performance level of the users' interface, all of which E-book Pvt. Ltd. is focused on due to the provision of quality services and innovation.

Introduction

e-Book Pvt. Ltd is a well-known used book selling company in Sri Lanka fully equipped with a large number of bookshops and holding a wide range of both Sri Lankan and international used books. The organization targets all ages starting from childhood up to college level users and has ranked high in the industry for the past three years. For enhanced customer reach and better customer satisfaction, the company aims at developing an online web application to contain all necessary activities for the customers as well as the administrators for e-Book Pvt. Ltd.

❖ Aim

The major focus of this project is to design and implement an enhanced web based application that could be used by e-Book Pvt. Ltd for the purpose of selling and buying second hand books and for managing its customer relationship as well. With the use of an online platform, this company will be able to expand its market base, increase productivity, and provide a smooth interaction between the users as well as the administrators of the site.

❖ Objectives

To achieve the aim, the project has set the following specific objectives:To achieve the aim, the project has set the following specific objectives:

- Customer Registration and Authentication:

For existing customers, allow them to purchase and access the features of the web application as well as give new customers that haven't register the ability to do that.

Customers' accounts should be protected by safe and better authentication processes.

- Order Management:

Provide the customer with an option to order and track books they would like to purchase while being logged into the site.

Develop tools that enable customers to manage their orders, including tracking and canceling.

- Book Search and Feedback:

Briefly enable a Search function for all users allowing them to explore book details.

Let the customers privy to express their feedback and reviews for books when they are logged in.

- Administrative Control:

Create a secure Stack administrators' access where Stack administrators can register and login.

Introduce the functionalities that allow editing information about books, customers, and orders.

Supply reports that would help in getting key business information.

Set up an admin panel that will act as an interface where the user is informed of the various processes in the business.

❖ Scope

The scope of the project encompasses the design, development, and deployment of the web application with the following functionalities:

- Customer Portal:

The user registration and the login form.

Protect measures regarding customer operations, including authentication and authorization.

Updated and easily navigable book library collection with keyword search and narrowing down options.

Navigation and Observation of orders, and Other functionalities of placing an order.

This is a system that allows users to rate books or post commentaries about books they have read or even preview new books to read.

- Admin Portal:

Main site with admin registration and admin login mostly.

Executive summary: A concise list of figures that are important for the overall understanding of business operations.

Functions for specifying and modifying book information such as book details for the library database.

Customer information overview and managing current customers that involve customer data viewing and editing.

Tasks include functionalities for tracking and modifying the order status.

Sales and customer reporting and analysis tools to produce reports on sales and customer trends, inventory availability, etc.

- Technology and Tools:

Utilization of ASP. NET for the web application framework Occasionally.

Primarily C# as the main language.

Compatibility with a high quality data base management system for data storing and retrieval.

These features can help e-Book Pvt. Ltd in delivering a superior experience for the new-age customers it targets while optimizing its corporate core processes that would help in improving its revenue and operational effectiveness in the highly competitive bookselling industry.

System Analysis

Another activity of systems analysis is the gathering of facts and information as well as appreciation of the functions which characterize a system, assessment of existing difficulties and feasible recommendation as regards enhancement of the functioning of the system. A part of this includes mapping out the business processes carried out, collecting operational data and analyzing the entire information flow in the organization, identifying areas of the system that are weak or slow and continually improving the system to work around these deficiencies for the improvement of organizational objectives. There are other aspects of system analysis, which include the ability to dissect larger processes involving the whole system, creation of documentations on existing and proposed data stores, and documentation of certain manually driven processes.

The major purposes of systems analysis for e-Book Pvt. Ltd. Online Web Application are to find answers for each business process:The major purposes of systems analysis for e-Book Pvt. Ltd. Online Web Application are to find answers for each business process:

- What is being done? Reselling the used books through physical book shops.
- The following is an illustration of how it is being done: Mainly through swapping of goods for cash transacted within its store and record of its store stocks.
- Who is doing it? Store staff and management.
- How: When is it being done? While store operation is ongoing.
- However, every action or plan being implemented must therefore answer the following questions; However, in the case of this concept, to offer a platform where used books can be sold and different readership needs are met.

Its social value is apparent; how can it be enhanced? because it is crucial to probe into other ways of reaching out to customers, improving efficiency and the operations of the company as well as creating an appealing web application all with the aim of providing excellent service to the customers.

This involves analysis of the current state of e-Book Pvt. Ltd with focus on the current business mode and assessing problem areas and come up with better solutions as to how the business can transform into an online oriented company. The goal is to develop a plan for creating a coherent and logical structure, which will serve the functions required by users at the current stage of development, as well as in the future with regard to the organizational structure.

Systems analysis as quite a fluid method for the e-Book Pvt. Ltd. Online Web Application since the process keeps on running in cycles until the stakeholders arrive at the right and satisfactory solution to provide a refined and efficient Online Web Application.

- **Architecture and Technologies**

  Framework: ASP.Net with C#

  Database: SQL Server

  Frontend: HTML, CSS, JavaScript

  Authentication: ASP.Net Identity

Customer Module

- **Registration and Login:**
  - New users can register by providing personal details.
  - Registered users can log in to access additional functionalities.
- **Book Searching:**
  - Users can search for books using keywords, categories, or other filters.
- **Order Management:**
  - Logged-in users can place orders for selected books.
  - Users can view and delete their orders through their profile.
- **Feedback:**
  - Logged-in users can provide feedback on purchased books, enhancing the community feel and helping other buyers.

Admin Module

- **Admin Authentication:**
  - Admins must log in to access the dashboard.
- **Book Management:**
  - Admins can add, update, and delete book details, ensuring the catalog is up to date.

- **Customer Management:**
    - Admins can view and manage customer information, facilitating better customer service.
- **Order Management:**
    - Admins can track and manage orders, ensuring timely processing and delivery.
- **Report Generation:**
    - Admins can generate reports on sales, inventory, customer activity, and other metrics to make informed business decisions.
- **Dashboard:**
    - A centralized dashboard provides a quick overview of key metrics and system status, aiding in efficient management.

Security and Performance

- **Authentication & Authorization:** Ensuring secure access to both customer and admin functionalities.
- **Data Integrity:** Protecting data accuracy and consistency through validation and secure database transactions.
- **Scalability:** Designing the system to handle growing numbers of users and transactions efficiently.

System Flow

- **User Interaction:**
    - Users access the site, browse books, register/log in, and manage their orders.
- **Admin Interaction:**
    - Admins log in, update book inventories, manage users and orders, and generate reports.

- **Backend Processes:**
    - Handling database transactions, ensuring data security, and maintaining high performance.

By implementing these functionalities, E-book Pvt. Ltd will be able to offer a seamless and engaging online shopping experience for its customers, while also ensuring efficient management and growth of its business operations.

System Overview

e-book pvt. Ltd is a leading used book dealer in Sri Lanka. In the last three years it has emerged as one of the most prestigious and leading chain of bookstores in Sri Lanka with the rapid development and expansion of its business activities both qualitatively and quantitatively. It offers a large collection of Sri Lankan and foreign reference books covering a wide variety of subjects to cater to the diverse needs of their wide readership, from young children to the highly educated. To expand their business activities, they plan to develop an online web application.

Present System

In Current system e-book pvt. Ltd is a leading used books selling is handled manually. Also, customers will find it difficult to get the required information about the event schedule like where exactly the particular event is going on.

Proposed System

- **Architecture and Technologies**

  Framework: ASP.Net with C#

  Database: SQL Server

  Frontend: HTML, CSS, JavaScript

  Authentication: ASP.Net Identity

Customer Module

- **Registration and Login:**
  - New users can register by providing personal details.
  - Registered users can log in to access additional functionalities.
- **Book Searching:**
  - Users can search for books using keywords, categories, or other filters.
- **Order Management:**
  - Logged-in users can place orders for selected books.
  - Users can view and delete their orders through their profile.
- **Feedback:**
  - Logged-in users can provide feedback on purchased books, enhancing the community feel and helping other buyers.

Admin Module

- **Admin Authentication:**
  - Admins must log in to access the dashboard.
- **Book Management:**
  - Admins can add, update, and delete book details, ensuring the catalog is up to date.
- **Customer Management:**
  - Admins can view and manage customer information, facilitating better customer service.
- **Order Management:**
  - Admins can track and manage orders, ensuring timely processing and delivery.
- **Report Generation:**
  - Admins can generate reports on sales, inventory, customer activity, and other metrics to make informed business decisions.
- **Dashboard:**
  - A centralized dashboard provides a quick overview of key metrics and system status, aiding in efficient management.

Security and Performance

- **Authentication & Authorization:** Ensuring secure access to both customer and admin functionalities.
- **Data Integrity:** Protecting data accuracy and consistency through validation and secure database transactions.
- **Scalability:** Designing the system to handle growing numbers of users and transactions efficiently.

System Flow

- **User Interaction:**
    - Users access the site, browse books, register/log in, and manage their orders.
- **Admin Interaction:**
    - Admins log in, update book inventories, manage users and orders, and generate reports.
- **Backend Processes:**
    - Handling database transactions, ensuring data security, and maintaining high performance.

By implementing these functionalities, E-book Pvt. Ltd will be able to offer a seamless and engaging online shopping experience for its customers, while also ensuring efficient management and growth of its business operations.

Software Requirements Specification

- **Software Requirements**

❖ **Functional Requirements**

Customer Functionalities:

Registration:

New customers should most likely be registered by entering their information: name, e-mail, password, address, and the phone number.

Login/Logout:

The registered customers have to be able to sign in with the following credentials, the email and the password.

Secure logout functionality.

Book Search:

Each user has the ability to perform a basic search by title, author and publisher, isbn/issn, category or any keyword of his or her choice.

Order Books:

The buying of books can only be done by clients who sign in to the online store.

Customers are they are able to select and put books into the cart,view the cart and check out.

Order Management:

Registered users only can track the history of orders they placed and the orders in progress.

Several customers may cancel or delete their orders before shipment.

Feedback:

Registered customers can offer feedback or review reports of books that they have bought.

Admin Functionalities:

Admin Registration/Login:

Managers ought to be capable of registering and logging in without being exposed to security risks.

Manage Books:

Using the Pro-Code, admins can Insert, Update, and delete details such as book title, author, category, price, and whether the book is available or not.

Manage Customers:

Concerning the changes and amendments possible on customer information by the admins, they are as follows:

Manage Orders:

It only allows admins to view the orders and furthermore make any changes concerning the status of the order or even delete the order completely.

Reports:

There are many activities that can be conducted by admins; sales reporting, inventory reporting, customer activity reporting, and customer feedback reporting.

Dashboard:

A complex, all-encompassing client reporting and analytics solution which can offer information such as current and future sales projections, activity and engagement of active consumers, inventory and stock quality, and many other valuable tidings.

❖ **Non-Functional Requirements**

Performance:

The use of the system should support up to 1000 concurrent users primarily.

The page load time should include the time taken for various components to load and should not exceed 2 seconds.

Security:

Any passwords that belong to the system's users should be stored safely using the hash techniques.

It also means that the system has to be safeguarded from such attacks as SQL injection, Cross-site scripting, and cross-site forgery.

As an important prerequisite for the target application to run securely, HTTPS has to be used for the communication.

Usability:

The UI should be clean and clear with the understanding of infavorites and each individual feature.

The application should therefore be usable by disabled or physically challenged persons.

Scalability:

When choosing the instances of the system, the potential for scalability of the system for future increases in the number of users and the amount of data has to be taken under consideration.

Reliability:

There is also a need to enhance the available system uptime to 99%. 9%.

Maintainability:

It should also be clean and documentation standard in a manner that allows easy modification and in the future when bug fixing.

- **Hardware Requirements**

Client-Side Requirements

Minimum:

Processor: For example, we could use Intel Core i3 or any other equivalent processors.

RAM: 4GB

Storage: 2GB free space

Browser: Google Chrome v84.0.4147.105, Mozilla Firefox v 74.0.1 (64-bit), Microsoft Edge v 88.0.705.56, Apple Safari v13.1

Internet: Had reliable connectivity of at least 5 Mbps download speed.

Server-Side Requirements

Minimum:

Processor: Intel Xeon E3 CPU for business class or equivalent

RAM: 16GB

Storage: 500GB SSD

Network: 100 Mbps Ethernet

Operating System: The Guest operating system can be any of the following: Windows Server 2019 or Linux Ubuntu 20.

Web Server: IIS 10 (in case with a Windows-based OS) while Nginx/Apache is preferable for hosting a Linux-based OS.

Database: For pairing, it is suggested to use SQL Server 2019 or MySQL 8. 0

- **System Methodology**

Agile Methodology:

Agile processes will be adopted as this type of process ensures that the development is done in stages and allows for flexibility.

To meet minor goals, two-week sprints will be utilized to produce incremental value.

Here it will be possible to incorporate the feedback from stakeholders which will enable the project to achieve the business requirements.

Methodology for System Design - Object Oriented Design Methodology

1. Encapsulation:

It is therefore similar to the process of placing and object into a box, and from there regulating when and how it can be manipulated. In the aspect of the program, Encapsulation would mean that we have to group the data the program uses, like the customer data (customer name, password, email) and the functions that will be working on it like registering or logging in functions in classes like Customer and Admin classes respectively. When declaring data and methods within these classes, we create the scope that corresponds to the encapsulation – no outside entity can modify what goes on within the classes. This not only ensures data accuracy but also improves maintainability as operation changes are only made in the class they are relevant to.

2. Abstraction:

Abstraction therefore entails the step of conversion of the real complicated system into a simple form where some specific features are emphasized and more complex details are obscured. In fact in our application, abstraction can be depicted by creating abstract classes or interfaces that describe general interfaces that shall be implemented by customers and admins since they share some of the similar functionalities. Hence, an example of an abstract class or interface could be 'User' that defines methods such as register(), login(), and submitFeedback(). Some concrete embodiments of such approaches would include the provision of detail by the 'Customer' and 'Admin' classes which are sub-classes or implementing classes of the 'User' abstract class. By making certain characteristics of objects, like functionality, belong to interfaces or abstract classes, we set a framework that defines how different parts of the application should behave.

3. Inheritance:

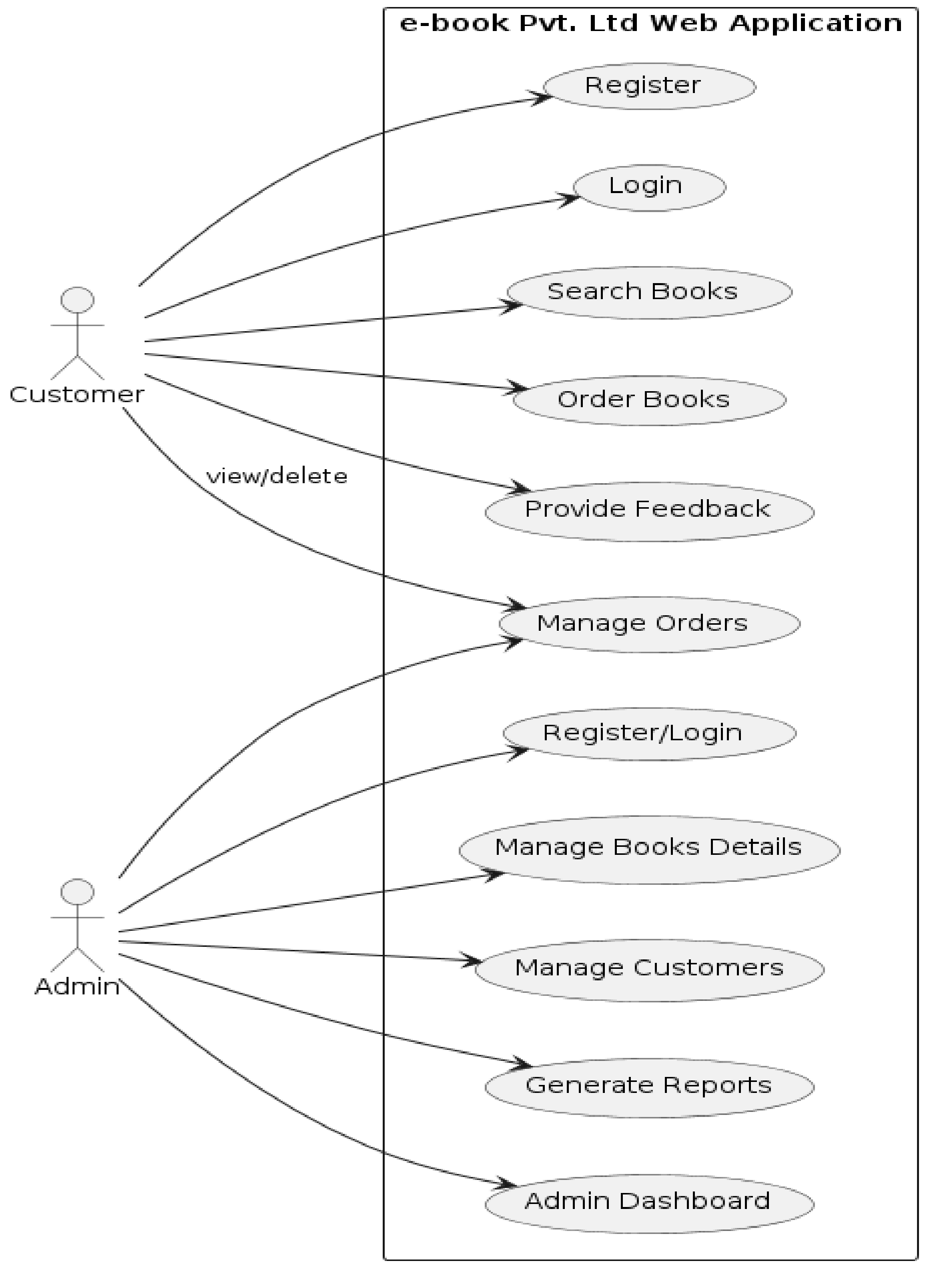
Another feature of object-oriented programming refers to the Inheritance that is the possibility of new classes to be derived from the existing ones while having similarities in terms of properties and behaviors, but at the same time being able to contain specific features of their own. When using classes within our application, inheritance can then be used to create a relationship between classes such as the subclass and superclass relationship. For example, the Customer and the Admin classes could further inherit from a base-class or interface called ‘User’. Apart from reusing code through elimination of repeated code as a form of inheritance it enhances polymorphism by enabling objects that belong to different classes to be used in a form of interchangeability. Subclasses should inherit from the general classes in a manner that enhances arrangement and orderly presentation of classes to meet the demands of learners.

4. High Cohesion and Loose Coupling:High Cohesion and Loose Coupling:

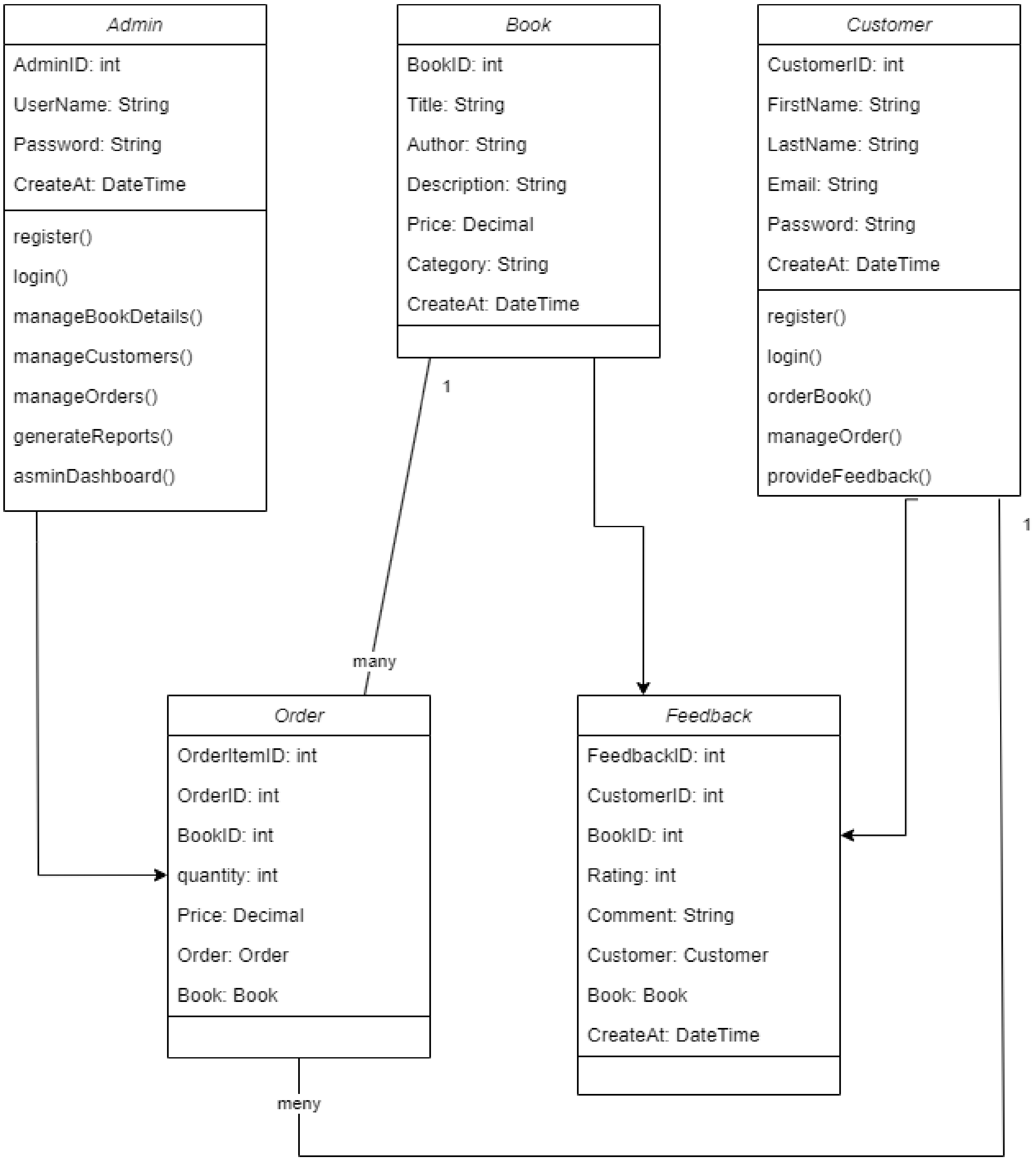
High cohesiveness focus with the elements within the module (e. g. , class) to ensure that they are closely related to one part of the system and serve a well-defined objective. For its part, loose coupling means the degree by which the different modules within a system are dependent on each other; the lower the level of interconnection, the better. All these have to be done to ensure that the ASP has high cohesion and low coupling. In relations to NET MVC architecture it is necessary to create classes that clearly define responsibilities of tasks. For instance, one can have distinct manager or service classes specific to tasks like order, book and customer management. These classes would include all related functionalities into the same module, thus, the levels of internal coupling would be high. Furthermore, by using dependency injection as a way to construct objects, classes can be separated and this can lead to even greater flexibility since dependencies are not hard coded into the class. This design approach helps improve modularity, maintainability as well as the scalability of growth within the application context.

Methodology for System Design

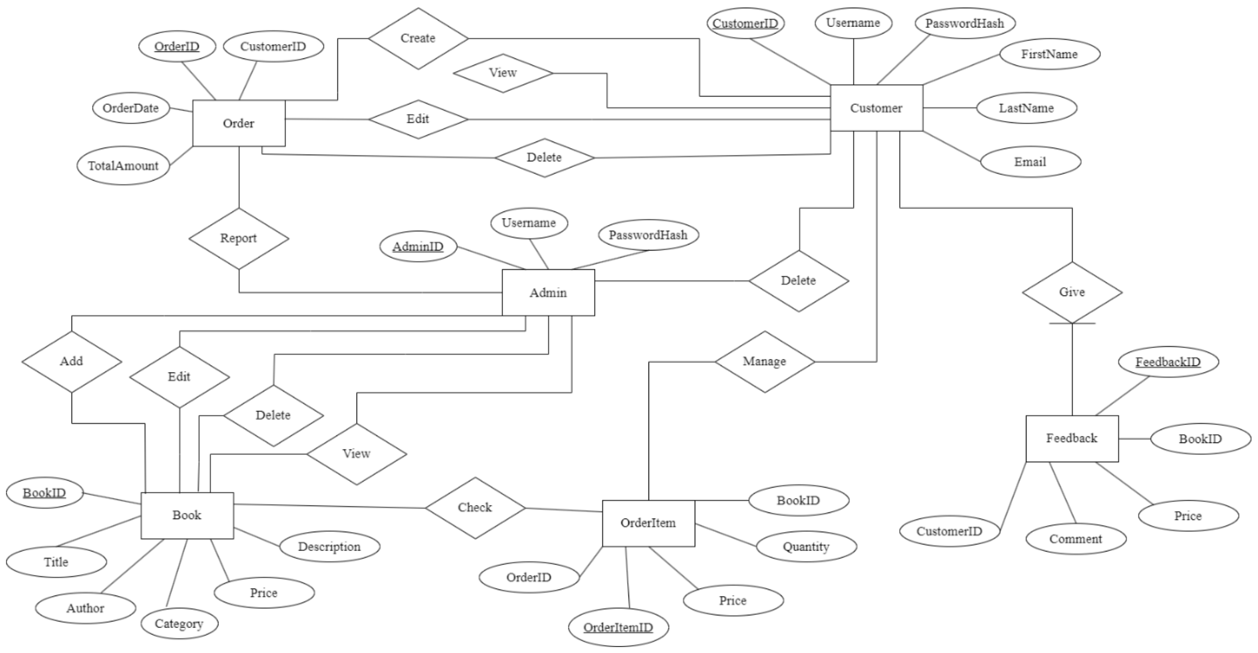
User Case Diagram



Class Diagram

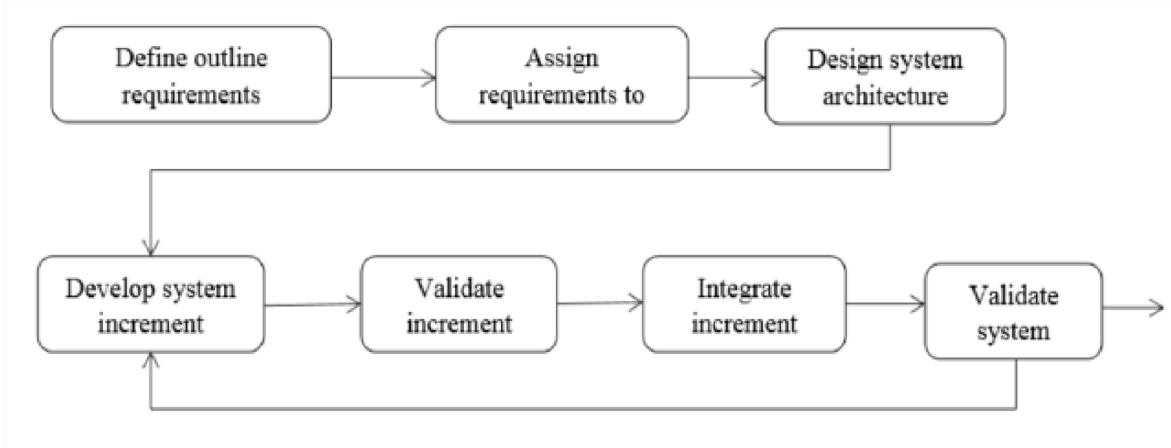


ER Diagram



Preliminary Analysis

Preliminary analysis aims at the following goals: define user requirements, assess system concept for its practical applicability, perform an economic and technical analysis, CBA, and define the contours of any subsequent practical work on the system. It must be sufficient capital of expertise for the computer hardware and software required for analysis. The following questions are asked when it comes to performing analysis. Which aspects can be associated with the given activity and how much time is required for its implementation? Accordingly, there cannot be any prescriptions to determine this, or mathematical formulae on which this can be worked out. Thus it should be decided on size, difficulty, application field, end-use, and contractual obligation are few of the classifications in which it is decided. One more crucial issue arises in the context of this discussion and it regards to deciding on who should perform the task. Well, anybody with a fair level of experience and well-trained analyst should be able to do it. In large project there can be an investigation team to work for the project and to investigate for the discrepancies if there is any prior to the project. Software development life cycle is a method pursuing the process of forming a good software and its life cycle stage offer quality and correctness of good software. In this regard there are numerous types of models that have been developed to initiate formation of sd life cycle. However, they differ from one another and y have got different effects/advantages as well as teachers. The basic models of SDLC are waterfall model, incremental model, prototype model, and a spiral model but the incremental model is a combination of one waterfall model. Several development cycles recur endlessly before getting to the right combination for success. They can be described as multi-watershed cycle, to know their workings it's possible to look at familiar to us water commitments. Cycles are separated up there into much smaller, simply handle iterations. It is used in the requisites, structuring, instantiation and examination phase of the project. The initial working of a software is designed through the first iteration meaning that the software application is operational during the initial stages of the software life cycle. Subsequent iterations are based upon the first software which was developed by the first iteration.



It has been only very well designed for house creation that will develop over time. When an incremental model is followed, the first increment is often a center creation for the company involved. This coincides with the utilization of center creation by the client or the experience of a full analysis.

As a result of applying and estimating an agreement is urbanized for the next development the same of an. The preparation involves shifting the center creation toward a healthier amount of meeting the customer requirements and then unleashing of additional features and functionality. There are no big jumps in the development process of a software product. However, coursing a software creation is being created, every pace put in to what has previously been finished.

Database Diagram

Table name: Admin

Filed	Data type	Constraints	Description
AdminID	Number	Primary Key	To store the ID of the admin.
Username	Varchar(20)	Not Null	To store admin user name.
PasswordHash	Varchar(20)	Not Null	To store password with hashing.

Table name: Customer

Filed	Data type	Constraints	Description
CustomerID	Number	Primary Key	To store the ID of the customer.
FirstName	Varchar(20)	Not Null	To store customer first name.
LastName	Varchar(20)	Not Null	To store customer last name.
Email	Varchar(50)	Not Null	To store customer email.
PasswordHash	Varchar(20)	Not Null	To store password with hashing.

Table name: Book

Filed	Data type	Constraints	Description
BookID	Number	Primary Key	To store the ID of the book.
Title	Varchar(50)	Not Null	To store book title.
Author	Varchar(50)	Not Null	To store author.
Description	Varchar()	Not Null	To store description of the book.
Price	Number	Not Null	To store price of the book.
Category	Varchar(50)	Not Null	To store category of the book.

Table name: Order

Filed	Data type	Constraints	Description
OrderID	Number	Primary Key	To store the ID of the order.
CustomerID	Number	Foreign Key	To store the ID of the customer.
OrderDate	DateTime	Not Null	To store the date of the order.
TotalAmount	Number	Not Null	To store the amount of the order.

Table name: OrderItem

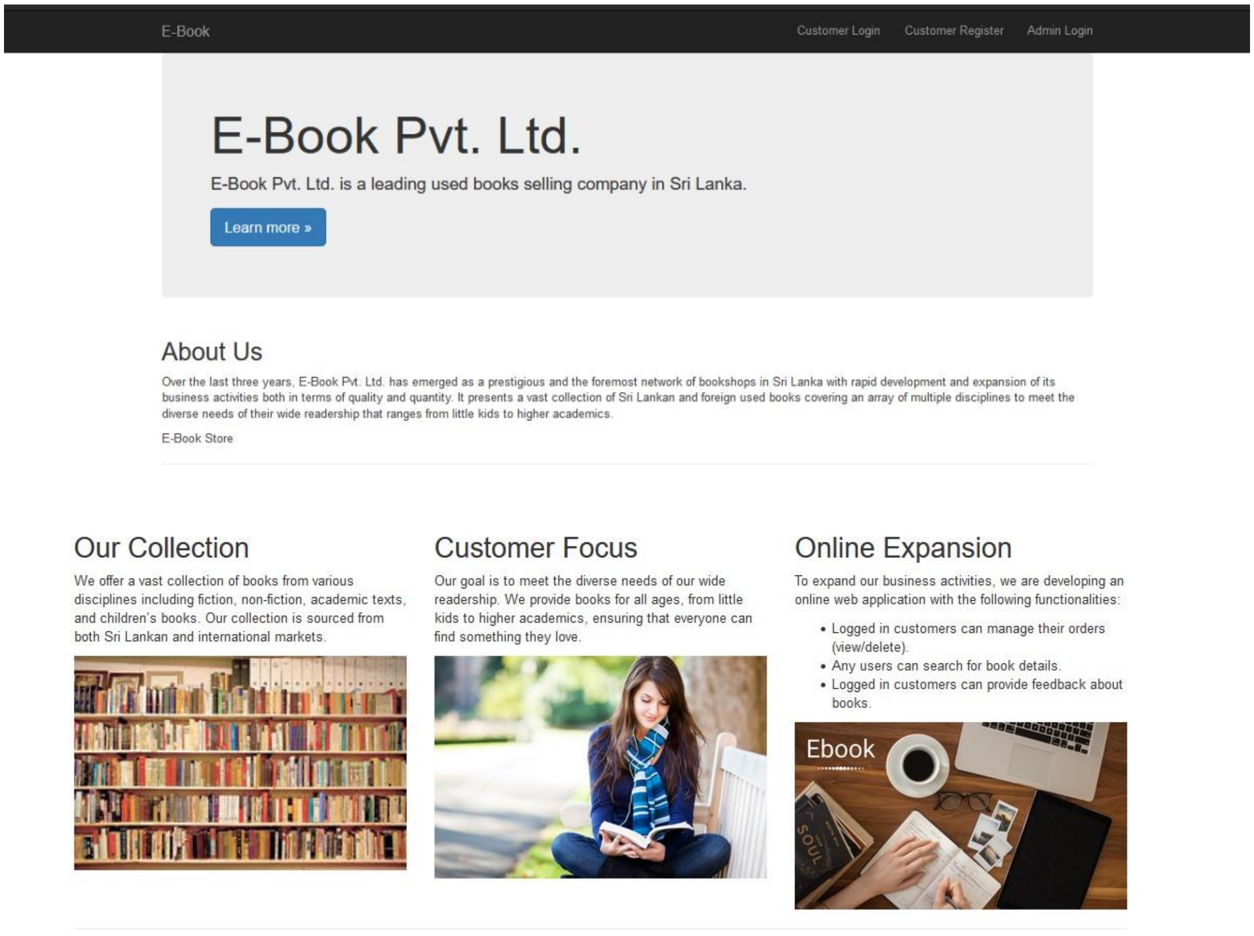
Filed	Data type	Constraints	Description
OrderItemID	Number	Primary Key	To store the ID of the order item.
OrderID	Number	Foreign Key	To store the ID of the customer.
BookID	Number	Foreign Key	To store the ID of the book.
Quantity	Number	Not Null	To store the number of the order item.
Price	Number	Not Null	To store the price of the order item.

Table name: Feedback

Filed	Data type	Constraints	Description
FeedbackID	Number	Primary Key	To store the ID of the feedback.
CustomerID	Number	Foreign Key	To store the ID of the customer.
BookID	Number	Foreign Key	To store the ID of the book.
Rating	Number	Not Null	To store the rating.
Comment	Varchar()	Not Null	To store the comment.

Interfaces

Landing page


E-Book Customer Login Customer Register Admin Login

E-Book Pvt. Ltd.

E-Book Pvt. Ltd. is a leading used books selling company in Sri Lanka.

Learn more »

About Us

Over the last three years, E-Book Pvt. Ltd. has emerged as a prestigious and the foremost network of bookshops in Sri Lanka with rapid development and expansion of its business activities both in terms of quality and quantity. It presents a vast collection of Sri Lankan and foreign used books covering an array of multiple disciplines to meet the diverse needs of their wide readership that ranges from little kids to higher academics.

E-Book Store

Our Collection

We offer a vast collection of books from various disciplines including fiction, non-fiction, academic texts, and children's books. Our collection is sourced from both Sri Lankan and international markets.

Customer Focus

Our goal is to meet the diverse needs of our wide readership. We provide books for all ages, from little kids to higher academics, ensuring that everyone can find something they love.

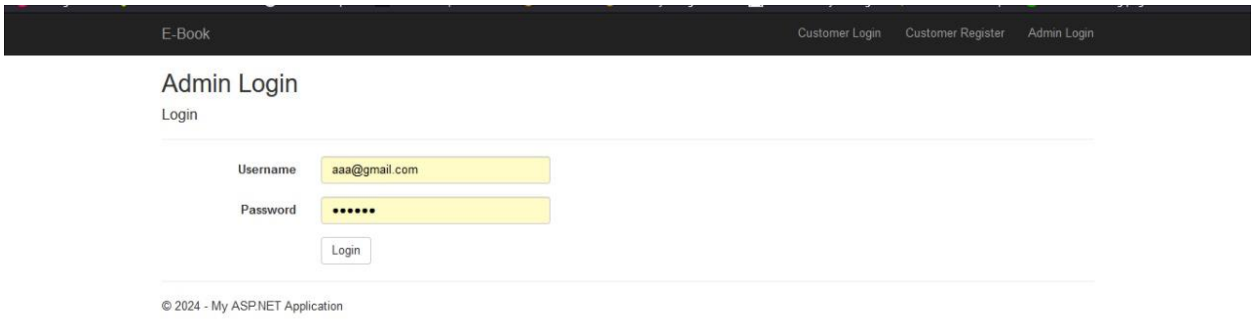
Online Expansion

To expand our business activities, we are developing an online web application with the following functionalities:

- Logged in customers can manage their orders (view/delete).
- Any users can search for book details.
- Logged in customers can provide feedback about books.

Admin Section

❖ **Admin login**


E-Book Customer Login Customer Register Admin Login

Admin Login

Login

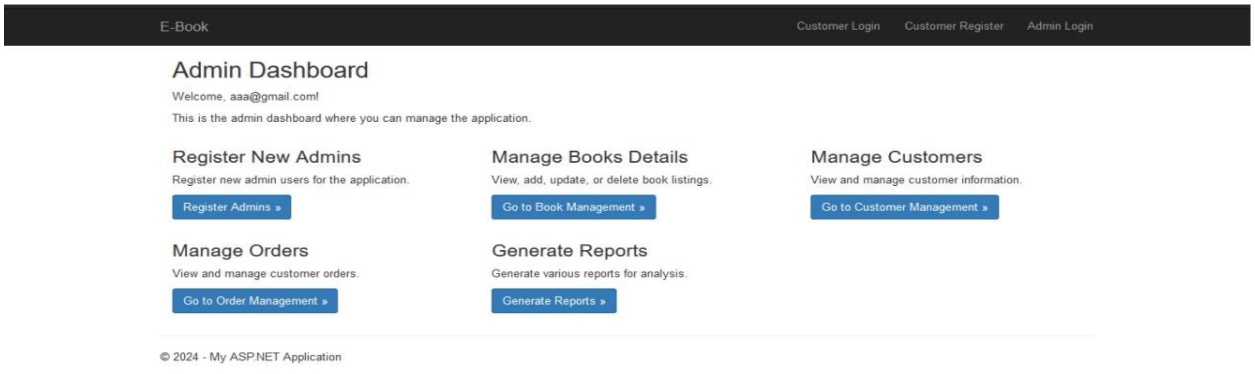
Username: aaa@gmail.com

Password: ••••••

Login

© 2024 - My ASP.NET Application

❖ Admin dashboard



E-Book | Customer Login | Customer Register | Admin Login

Admin Dashboard

Welcome, aaa@gmail.com!

This is the admin dashboard where you can manage the application.

Register New Admins
Register new admin users for the application.
Register Admins »

Manage Books Details
View, add, update, or delete book listings.
Go to Book Management »

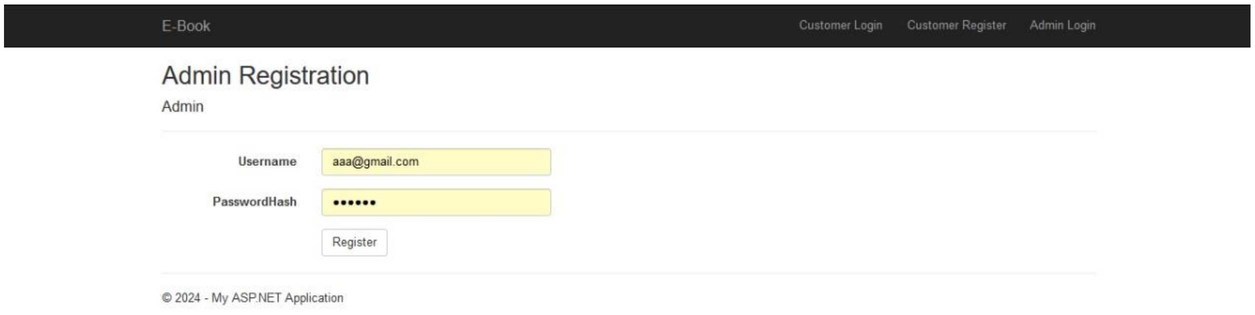
Manage Customers
View and manage customer information.
Go to Customer Management »

Manage Orders
View and manage customer orders.
Go to Order Management »

Generate Reports
Generate various reports for analysis.
Generate Reports »

© 2024 - My ASP.NET Application

❖ Admin registration



E-Book | Customer Login | Customer Register | Admin Login

Admin Registration

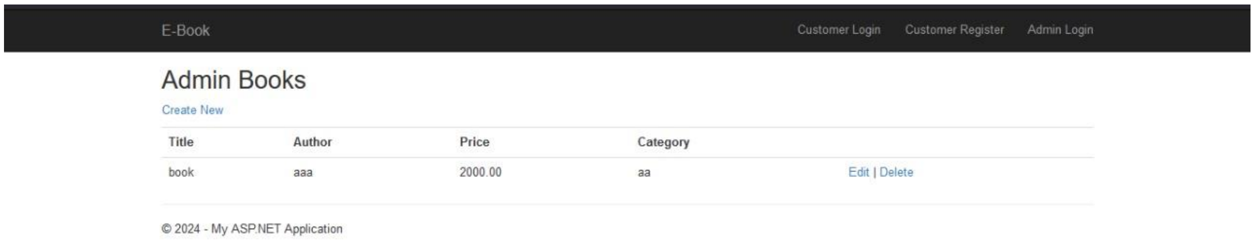
Admin

Username	aaa@gmail.com
PasswordHash	••••••

Register

© 2024 - My ASP.NET Application

❖ Add book – Admin



E-Book | Customer Login | Customer Register | Admin Login

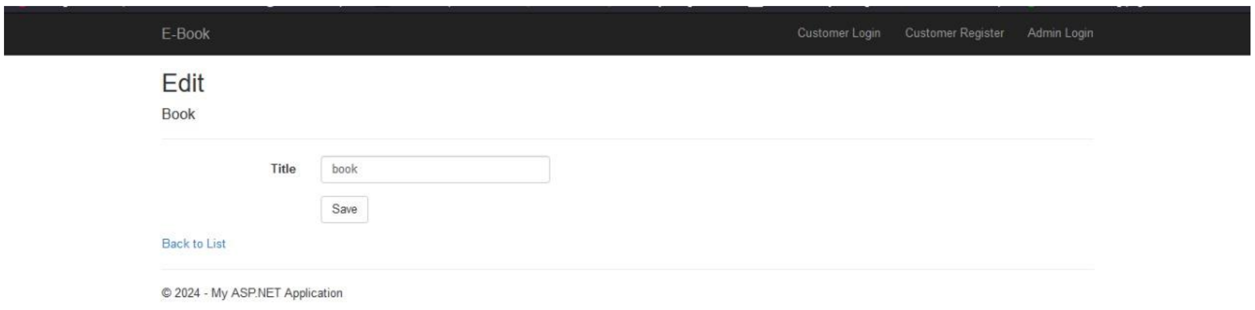
Admin Books

Create New

Title	Author	Price	Category	
book	aaa	2000.00	aa	Edit | Delete

© 2024 - My ASP.NET Application

❖ Edit book - Admin



E-Book | Customer Login | Customer Register | Admin Login

Edit

Book

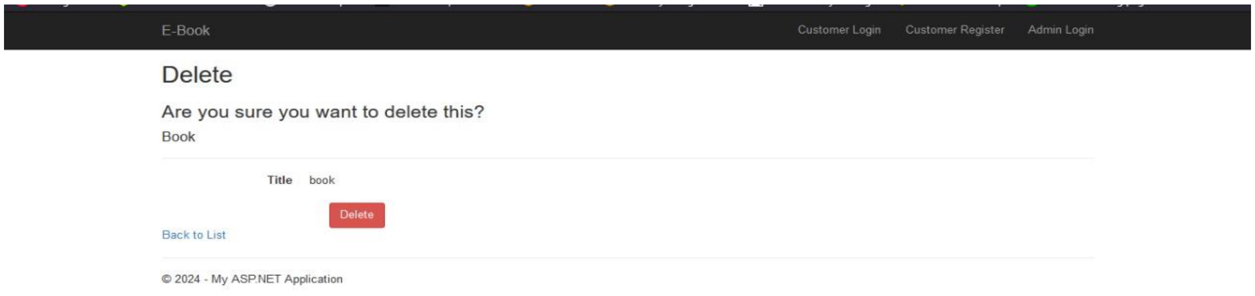
Title: book

Save

Back to List

© 2024 - My ASP.NET Application

❖ Delete book - Admin



Delete

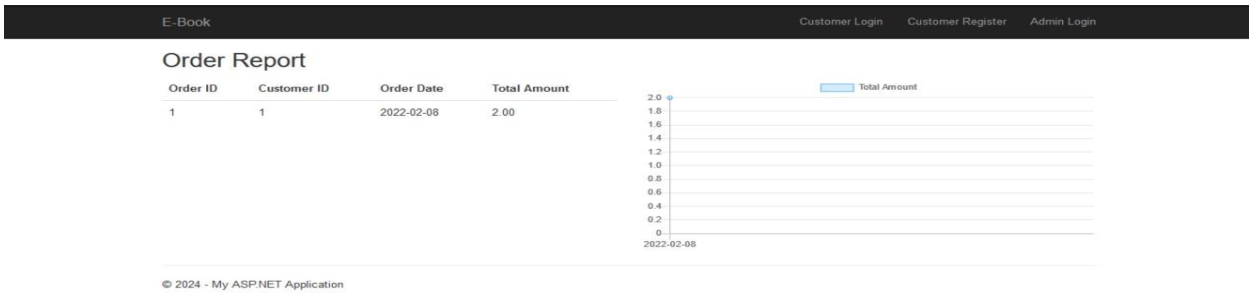
Are you sure you want to delete this?

Book

Title	book

© 2024 - My ASP.NET Application

❖ Generate report

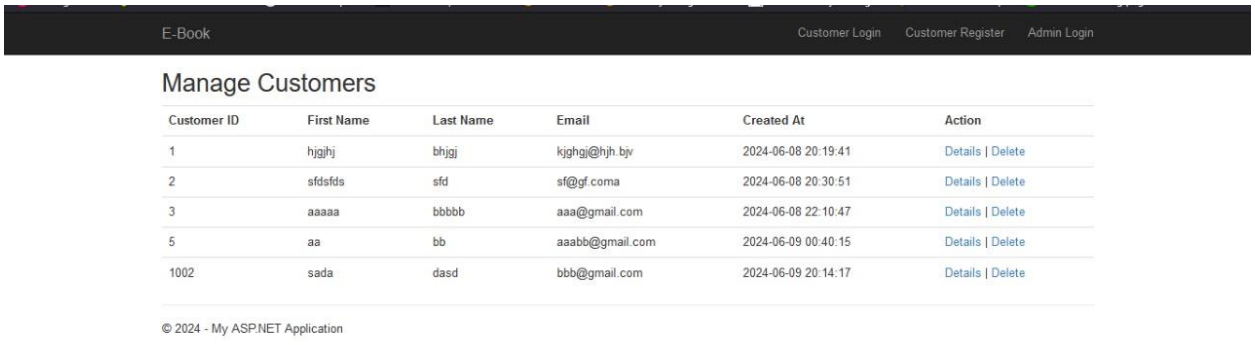


Order Report

Order ID	Customer ID	Order Date	Total Amount
1	1	2022-02-08	2.00

© 2024 - My ASP.NET Application

❖ Manage customer - Admin

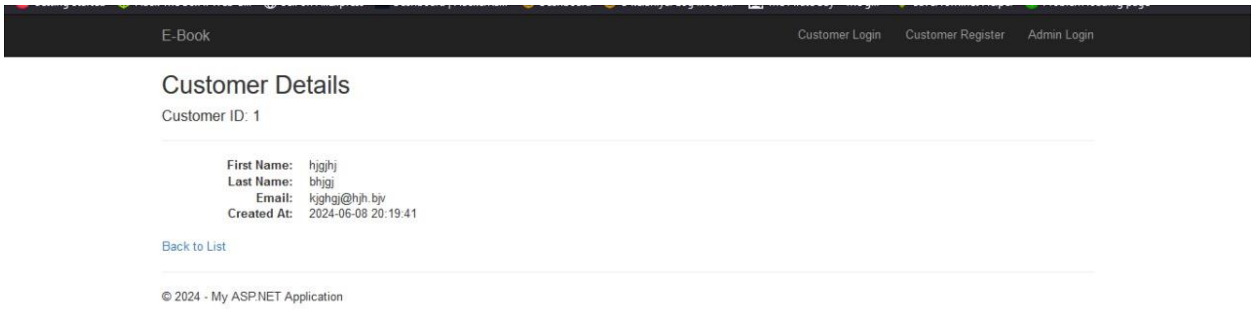


Manage Customers

Customer ID	First Name	Last Name	Email	Created At	Action
1	hjgjhj	bhjgj	kjghgj@hjh.bjv	2024-06-08 20:19:41	Details | Delete
2	sfdsfds	sfd	sf@gf.coma	2024-06-08 20:30:51	Details | Delete
3	aaaaa	bbbbb	aaa@gmail.com	2024-06-08 22:10:47	Details | Delete
5	aa	bb	aaabb@gmail.com	2024-06-09 00:40:15	Details | Delete
1002	sada	dasd	bbb@gmail.com	2024-06-09 20:14:17	Details | Delete

© 2024 - My ASP.NET Application

❖ View customer details – Admin



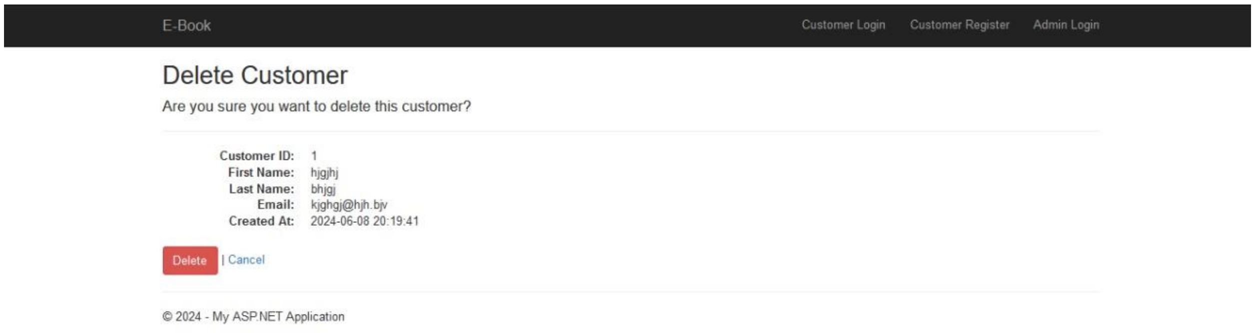
Customer Details

Customer ID: 1

First Name:	hjgjhj
Last Name:	bhjgj
Email:	kjghgj@hjh.bjv
Created At:	2024-06-08 20:19:41

© 2024 - My ASP.NET Application

❖ Delete customer - Admin



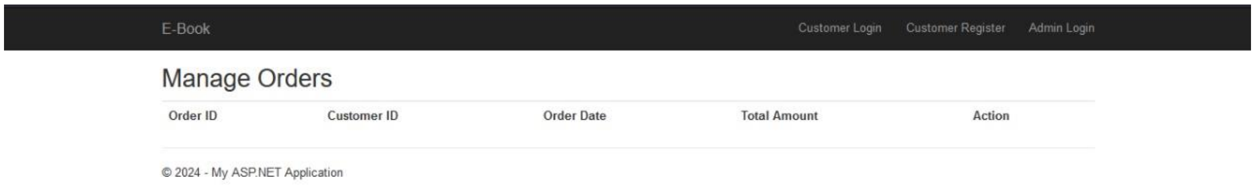
Delete Customer

Are you sure you want to delete this customer?

Customer ID:	1
First Name:	hjgjhj
Last Name:	bhjgj
Email:	kjghgj@hjh.bjv
Created At:	2024-06-08 20:19:41

© 2024 - My ASP.NET Application

❖ Manage order - Admin



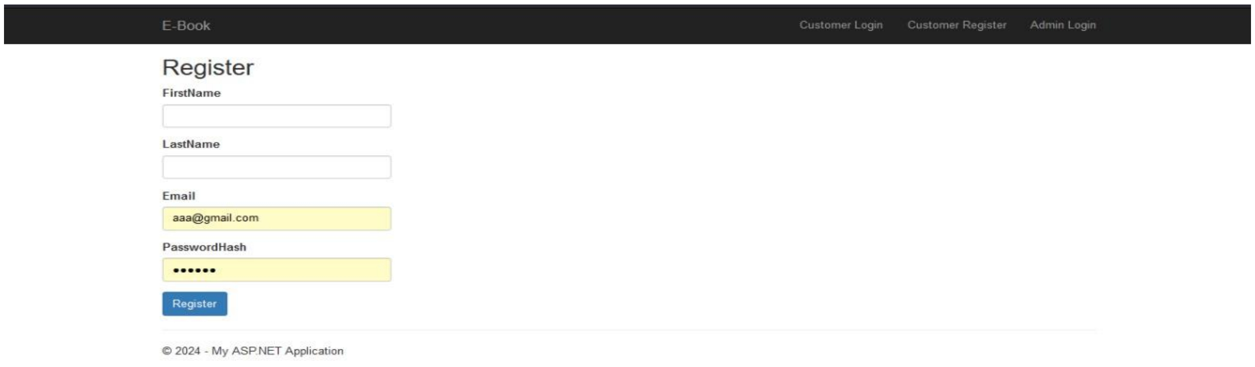
Manage Orders

Order ID	Customer ID	Order Date	Total Amount	Action

© 2024 - My ASP.NET Application

Customer Section

❖ Customer registration



Register

FirstName

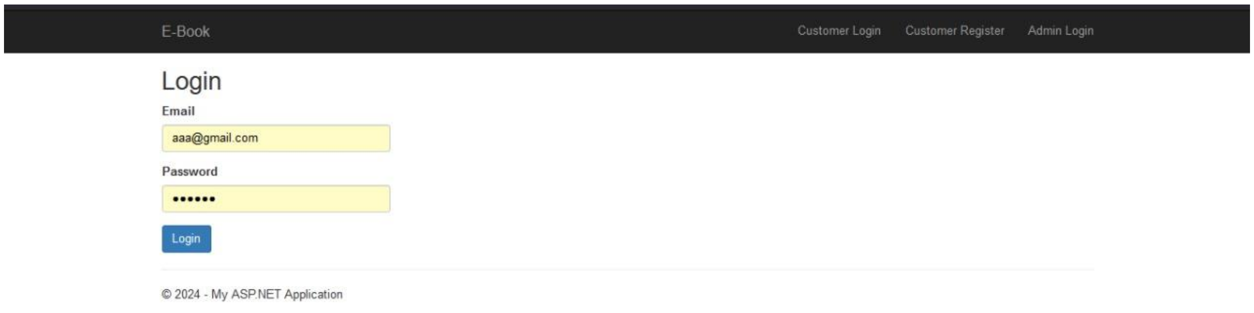
LastName

Email: aaa@gmail.com

PasswordHash: ••••••

© 2024 - My ASP.NET Application

❖ Customer login



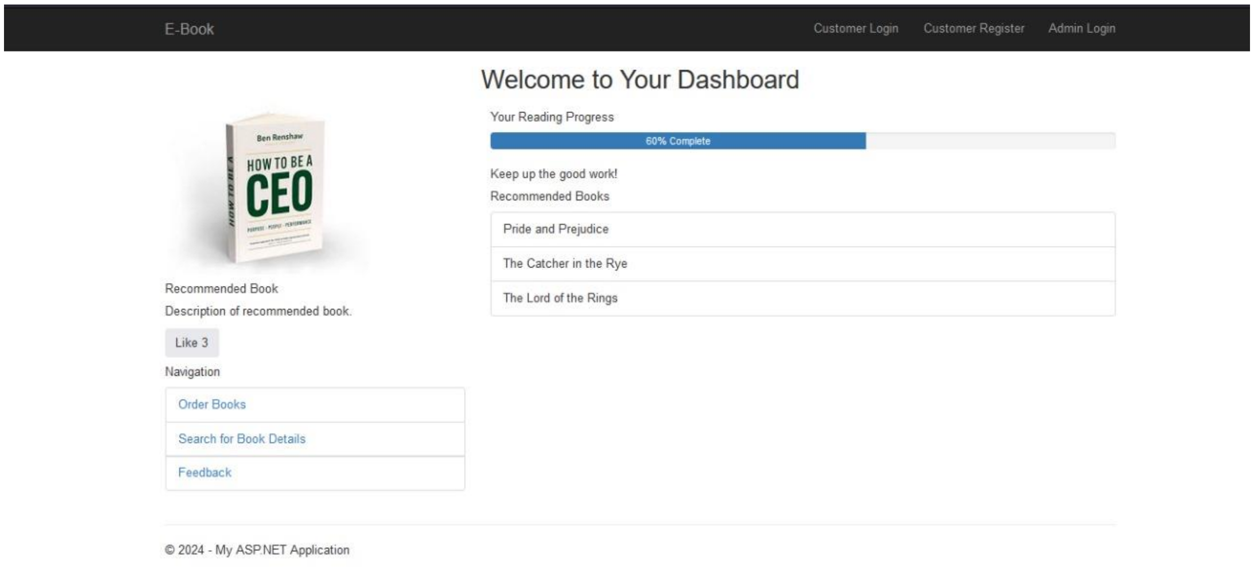
Login

Email: aaa@gmail.com

Password: ••••••

© 2024 - My ASP.NET Application

❖ **Customer dashboard**


E-Book   Customer Login   Customer Register   Admin Login

Welcome to Your Dashboard

Your Reading Progress

60% Complete

Keep up the good work!

Recommended Books

- Pride and Prejudice
- The Catcher in the Rye
- The Lord of the Rings

Recommended Book

Description of recommended book.

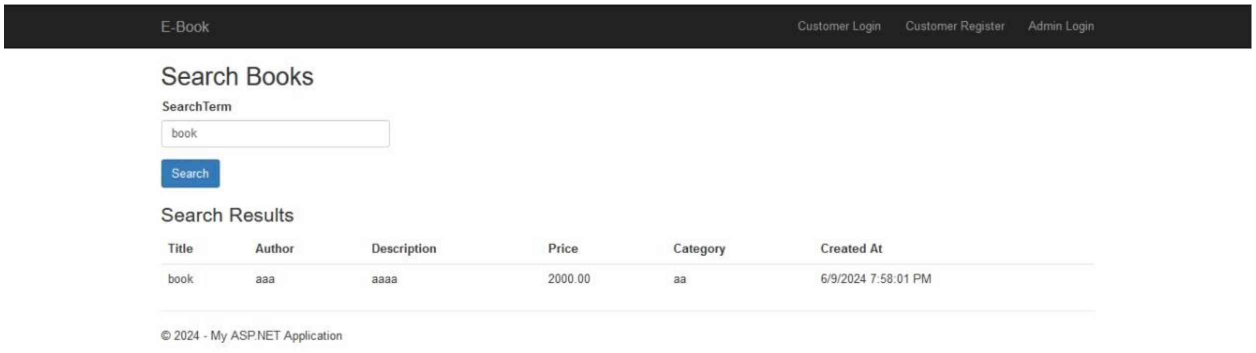
Like 3

Navigation

- Order Books
- Search for Book Details
- Feedback

© 2024 - My ASP.NET Application

❖ **Search book details – Customer**


E-Book   Customer Login   Customer Register   Admin Login

Search Books

SearchTerm

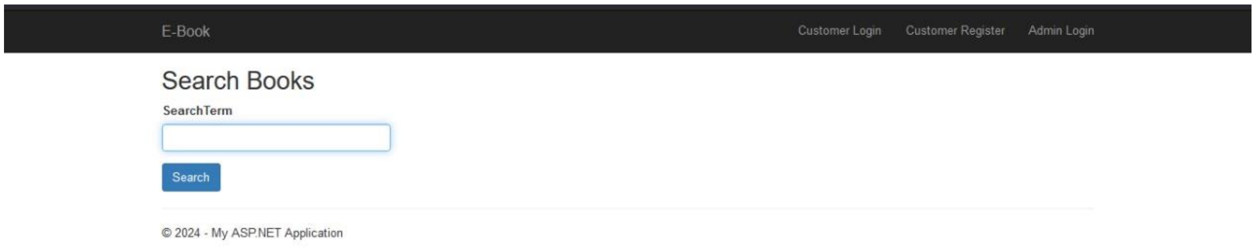
book

Search

Search Results

Title	Author	Description	Price	Category	Created At
book	aaa	aaaa	2000.00	aa	6/9/2024 7:58:01 PM

© 2024 - My ASP.NET Application


E-Book   Customer Login   Customer Register   Admin Login

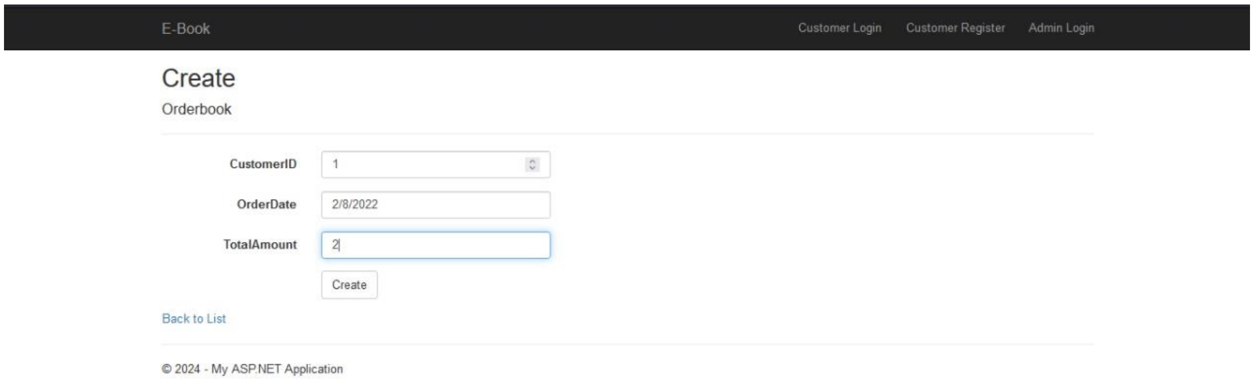
Search Books

SearchTerm

Search

© 2024 - My ASP.NET Application

❖ Order book – Customer



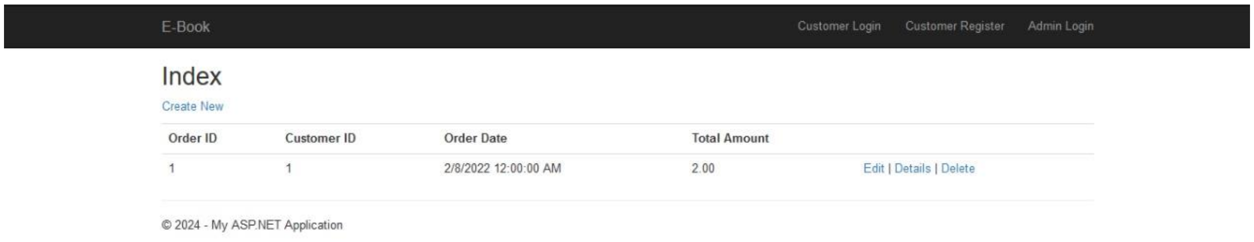
Create

Orderbook

CustomerID	1
OrderDate	2/8/2022
TotalAmount	2

© 2024 - My ASP.NET Application

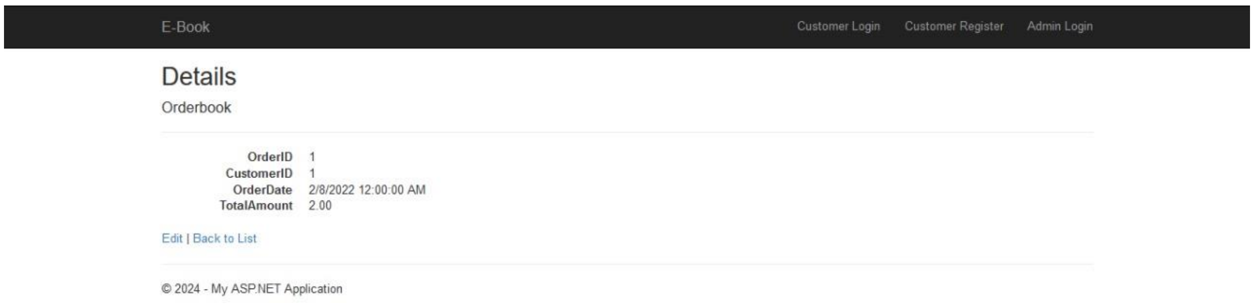
❖ View Order details – Customer



Index

Order ID	Customer ID	Order Date	Total Amount	
1	1	2/8/2022 12:00:00 AM	2.00	Edit | Details | Delete

© 2024 - My ASP.NET Application



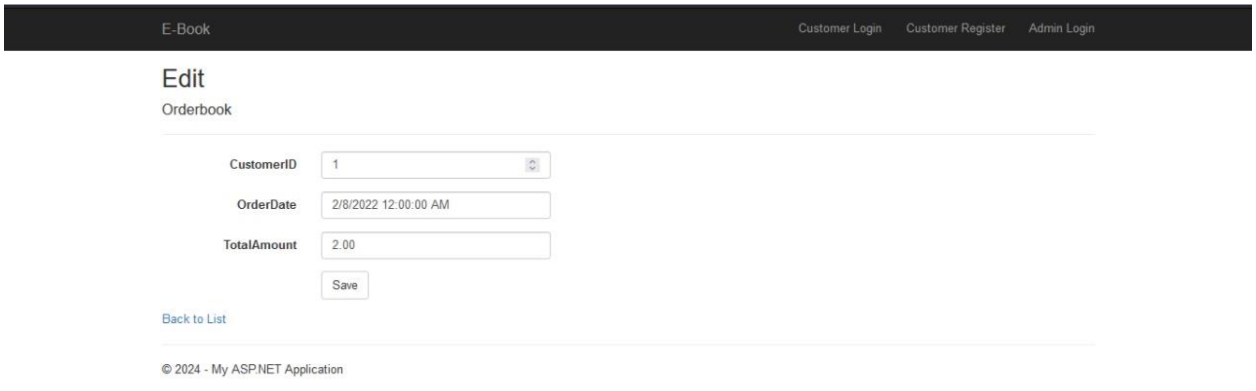
Details

Orderbook

OrderID	1
CustomerID	1
OrderDate	2/8/2022 12:00:00 AM
TotalAmount	2.00

© 2024 - My ASP.NET Application

❖ Edit Order details – Customer



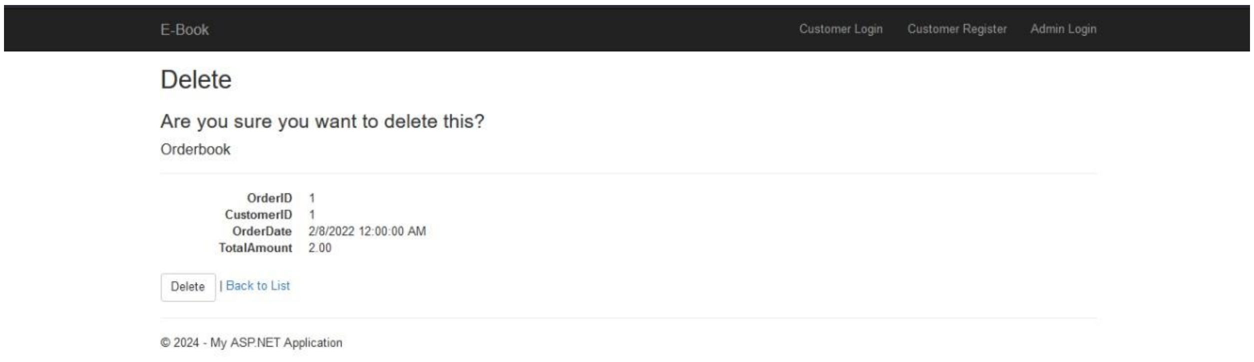
Edit

Orderbook

CustomerID	1
OrderDate	2/8/2022 12:00:00 AM
TotalAmount	2.00

© 2024 - My ASP.NET Application

❖ Delete Order – Customer



Delete

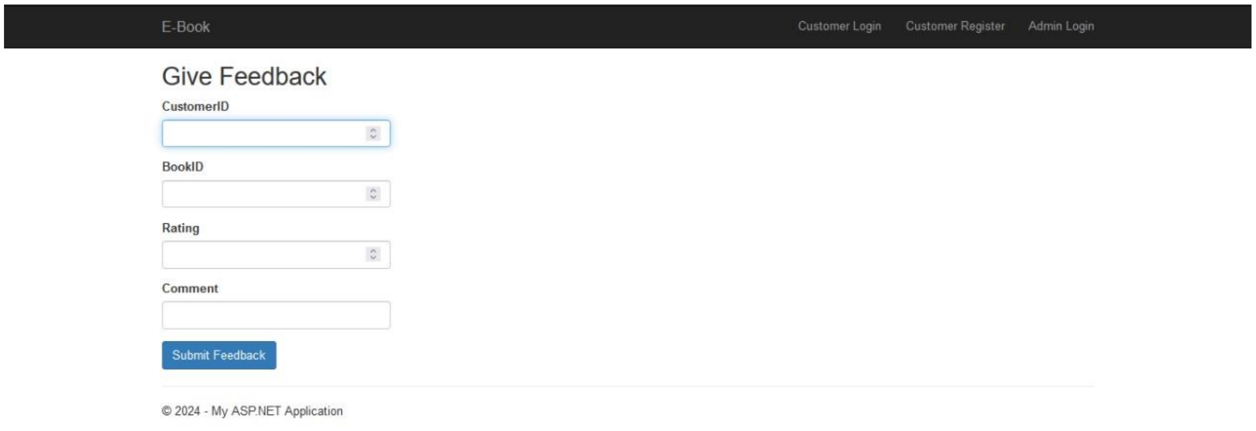
Are you sure you want to delete this?

Orderbook

OrderID	1
CustomerID	1
OrderDate	2/8/2022 12:00:00 AM
TotalAmount	2.00

© 2024 - My ASP.NET Application

❖ Feedback – Customer



Give Feedback

© 2024 - My ASP.NET Application

Coding and implementation

Using **MVC** Architecture.

- ✓ **ASP C# code base**

- ➢ **Model**

Admin.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebApplication1.Models
{
    public class Admin
    {
        public int AdminID { get; set; }
        public string Username { get; set; }
        public string PasswordHash { get; set; }
        public DateTime CreatedAt { get; set; }
    }

}
```

ApplicationDbContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace WebApplication1.Models
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext() : base("DefaultConnection")
        {
        }

        public DbSet<Customer> Customers { get; set; }
    }
}
```

LoginViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace WebApplication1.Models
{
    public class LoginViewModel
    {
        [Required]
        [EmailAddress]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
}
```

Book.cs

```
using System;
using System.Collections.Generic;

namespace WebApplication1.Models
{
    public class Book
    {
        public int BookID { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public string Description { get; set; }
        public decimal Price { get; set; }
        public string Category { get; set; }
        public DateTime CreatedAt { get; set; }

        public ICollection<OrderItembook> OrderItems { get; set; }
        public ICollection<Feedback> Feedbacks { get; set; }
    }

    public class BookSearchViewModel
    {
        public string SearchTerm { get; set; }
        public List<Book> Results { get; set; }
    }
}
```

Customer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;


namespace WebApplication1.Models
{
    public class Customer
    {
        public int CustomerID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public string PasswordHash { get; set; }
        public DateTime CreatedAt { get; set; }

        public ICollection<Orderbook> Orders { get; set; }
        public ICollection<Feedback> Feedbacks { get; set; }
    }
}
```

CustomerModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace WebApplication1.Models
{
    public class CustomerModel
    {
        [Key]
        public int CustomerID { get; set; }

        [Required]
        [StringLength(50)]
        public string FirstName { get; set; }

        [Required]
        [StringLength(50)]
        public string LastName { get; set; }

        [Required]
        [StringLength(100)]
        [EmailAddress]
        public string Email { get; set; }

        [Required]
        [StringLength(255)]
        public string PasswordHash { get; set; }

        public DateTime CreatedAt { get; set; } = DateTime.Now;

        public ICollection<Orderbook> Orders { get; set; }
        public ICollection<Feedback> Feedbacks { get; set; }
    }
}
```

Orderbook.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebApplication1.Models
{
    public class Orderbook
    {
        public int OrderID { get; set; }
        public int CustomerID { get; set; }
        public DateTime OrderDate { get; set; }
        public decimal TotalAmount { get; set; }

    }
}
```

OrderItemBook.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebApplication1.Models
{
    public class OrderItembook
    {
        public int OrderItemID { get; set; }
        public int OrderID { get; set; }
        public int BookID { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }



    }
}
```

Feedback.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebApplication1.Models
{
    public class Feedback
    {
        public int FeedbackID { get; set; }
        public int CustomerID { get; set; }
        public int BookID { get; set; }
        public int Rating { get; set; }
        public string Comment { get; set; }
        public DateTime CreatedAt { get; set; }

        public Customer Customer { get; set; }
        public Book Book { get; set; }
    }
}
```

YourDbContext.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace WebApplication1.Models
{
    public class YourDbContext : DbContext
    {
        public DbSet<CustomerModel> Customers { get; set; }
    }
}
```

➢ **View**

Admin

Dashboard.cshtml

```
@{ ViewBag.Title = "Admin Dashboard";
                Layout = "~/Views/Shared/_Layout.cshtml"; }

<div class="container">
    <h2>Admin Dashboard</h2>
    <p>Welcome, @Session["Username"]!</p>
    <p>This is the admin dashboard where you can manage the application.</p>

    <div class="row">
        <div class="col-md-4">
            <h3>Register New Admins</h3>
            <p>Register new admin users for the application.</p>
            <p><a class="btn btn-primary" href="@Url.Action("Register",
"Admin")">Register Admins &raquo;</a></p>
        </div>
        <div class="col-md-4">
            <h3>Manage Books Details</h3>
            <p>View, add, update, or delete book listings.</p>
            <p><a class="btn btn-primary" href="@Url.Action("Index",
"AdminBook")">Go to Book Management &raquo;</a></p>
        </div>
        <div class="col-md-4">
            <h3>Manage Customers</h3>
            <p>View and manage customer information.</p>
            <p><a class="btn btn-primary" href="@Url.Action("Index",
"AdminCustomer")">Go to Customer Management &raquo;</a></p>
        </div>
    </div>
    <div class="row">
        <div class="col-md-4">
            <h3>Manage Orders</h3>
            <p>View and manage customer orders.</p>
            <p><a class="btn btn-primary" href="@Url.Action("Index",
"AdminOrder")">Go to Order Management &raquo;</a></p>
        </div>
        <div class="col-md-4">
            <h3>Generate Reports</h3>
            <p>Generate various reports for analysis.</p>
            <p><a class="btn btn-primary" href="@Url.Action("Index",
"AdminCustomer")">Generate Reports &raquo;</a></p>
        </div>
    </div>
</div>
```

Register.cshtml

```
@model WebApplication1.Models.Admin

@{ ViewBag.Title = "Admin Registration"; }

<h2>Admin Registration</h2>

@using (Html.BeginForm("Register", "Admin", FormMethod.Post))
{
@Html.AntiForgeryToken()

                <div class="form-horizontal">
                    <h4>Admin</h4>
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class = "text-danger"
})

                    <div class="form-group">
                        @Html.LabelFor(model => model.Username, htmlAttributes: new
{ @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.Username, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Username, "",
new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.PasswordHash, htmlAttributes:
new { @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.PasswordFor(model => model.PasswordHash, new {
@class = "form-control" })
                            @Html.ValidationMessageFor(model => model.PasswordHash,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        <div class="col-md-offset-2 col-md-10">
                            <input type="submit" value="Register" class="btn btn-
default" />
                        </div>
                    </div>
                </div>}
```

Login.cshtml

```
@{ ViewBag.Title = "Admin Login"; }

<h2>Admin Login</h2>

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

                <div class="form-horizontal">
                    <h4>Login</h4>
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class = "text-danger"
})

                    <div class="form-group">
                        @Html.Label("Username", htmlAttributes: new { @class =
"control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.TextBox("Username", null, new { @class = "form-
control" })

                            @Html.ValidationMessage("Username", "", new { @class =
"text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.Label("Password", htmlAttributes: new { @class =
"control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.Password("Password", null, new { @class = "form-
control" })

                            @Html.ValidationMessage("Password", "", new { @class =
"text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        <div class="col-md-offset-2 col-md-10">
                            <input type="submit" value="Login" class="btn btn-
default" />
                        </div>
                    </div>
                </div>}
```

Admin – Manage Book

Index.cshtml

```
@model IEnumerable<WebApplication1.Models.Book>

@{ ViewBag.Title = "Admin Books"; }

<h2>Admin Books</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Title)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Author)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Price)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Category)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.Title)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Author)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Price)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Category)
    </td>
    <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.BookID }) |

        @Html.ActionLink("Delete", "Delete", new { id = item.BookID })
    </td>
</tr>}
</table>
```

Create.cshtml

```
@model WebApplication1.Models.Book

@{ ViewBag.Title = "Create Book"; }

<h2>Create Book</h2>

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

                <div class="form-horizontal">
                    <h4>Book</h4>
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class = "text-danger"
})

                    <div class="form-group">
                        @Html.LabelFor(model => model.Title, htmlAttributes: new {
@class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.Title, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Title, "", new
{ @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.Author, htmlAttributes: new {
@class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.Author, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Author, "",
new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.Description, htmlAttributes:
new { @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.Description, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Description,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.Price, htmlAttributes: new {
@class = "control-label col-md-2" })
                        <div class="col-md-10">
```

```
                            @Html.EditorFor(model => model.Price, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Price, "", new
{ @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.Category, htmlAttributes: new
{ @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.Category, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Category, "",
new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        <div class="col-md-offset-2 col-md-10">
                            <input type="submit" value="Create" class="btn btn-
default" />
                        </div>
                    </div>
                </div>}
```

Delete.cshtml

```
@model WebApplication1.Models.Book

@{ ViewBag.Title = "Delete"; }

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Book</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Title)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Title)
        </dd>

    </dl>

    @using (Html.BeginForm())
    {
@Html.AntiForgeryToken()

                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Delete" class="btn btn-danger"
/>
                    </div>
                </div>}
</div>

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Edit.cshtml

```
@model WebApplication1.Models.Book

@{ ViewBag.Title = "Edit"; }

<h2>Edit</h2>

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

                <div class="form-horizontal">
                    <h4>Book</h4>
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class = "text-danger"
})

                    <div class="form-group">
                        @Html.LabelFor(model => model.Title, htmlAttributes: new {
@class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.Title, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.Title, "", new
{ @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        <div class="col-md-offset-2 col-md-10">
                            <input type="submit" value="Save" class="btn btn-
default" />
                        </div>
                    </div>
                </div>}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Admin – Manage Customer

Index.cshtml

```
@model List<WebApplication1.Models.Customer>

@{ ViewBag.Title = "Manage Customers"; }

<h2>Manage Customers</h2>

<table class="table">
    <tr>
        <th>Customer ID</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Email</th>
        <th>Created At</th>
        <th>Action</th>
    </tr>
    @foreach (var customer in Model)
    {
<tr>
    <td>@customer.CustomerID</td>
    <td>@customer.FirstName</td>
    <td>@customer.LastName</td>
    <td>@customer.Email</td>
    <td>@customer.CreatedAt.ToString("yyyy-MM-dd HH:mm:ss")</td>
    <td>
        @Html.ActionLink("Details", "Details", new { id = customer.CustomerID }) |
        @Html.ActionLink("Delete", "Delete", new { id = customer.CustomerID })
    </td>
</tr>}
</table>
```

Details.cshtml

```
@model WebApplication1.Models.Customer

@{ ViewBag.Title = "Customer Details"; }

<h2>Customer Details</h2>
<div>
    <h4>Customer ID: @Model.CustomerID</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>First Name:</dt>
        <dd>@Model.FirstName</dd>
        <dt>Last Name:</dt>
        <dd>@Model.LastName</dd>
        <dt>Email:</dt>
        <dd>@Model.Email</dd>
        <dt>Created At:</dt>
        <dd>@Model.CreatedAt.ToString("yyyy-MM-dd HH:mm:ss")</dd>
    </dl>
</div>
<p>
    @Html.ActionLink("Back to List", "Index")
</p>
```

Delete.cshtml

```
@model WebApplication1.Models.Customer

@{ ViewBag.Title = "Delete Customer"; }

<h2>Delete Customer</h2>
<div>
    <h4>Are you sure you want to delete this customer?</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>Customer ID:</dt>
        <dd>@Model.CustomerID</dd>
        <dt>First Name:</dt>
        <dd>@Model.FirstName</dd>
        <dt>Last Name:</dt>
        <dd>@Model.LastName</dd>
        <dt>Email:</dt>
        <dd>@Model.Email</dd>
        <dt>Created At:</dt>
        <dd>@Model.CreatedAt.ToString("yyyy-MM-dd HH:mm:ss")</dd>
    </dl>
    @using (Html.BeginForm())
    {
@Html.AntiForgeryToken()
                <div class="form-actions no-color">
                    <input type="submit" value="Delete" class="btn btn-danger" /> |
                    @Html.ActionLink("Cancel", "Index")
                </div>}
</div>
```

Admin – Manage Order

Index.cshtml

```
@model List<WebApplication1.Models.Orderbook>

@{ ViewBag.Title = "Manage Orders"; }

<h2>Manage Orders</h2>

<table class="table">
    <tr>
        <th>Order ID</th>
        <th>Customer ID</th>
        <th>Order Date</th>
        <th>Total Amount</th>
        <th>Action</th>
    </tr>
    @foreach (var order in Model)
    {
<tr>
    <td>@order.OrderID</td>
    <td>@order.CustomerID</td>
    <td>@order.OrderDate.ToShortDateString()</td>
    <td>@order.TotalAmount.ToString("C")</td>
    <td>
        @Html.ActionLink("Details", "Details", new { id = order.OrderID }) |
        @Html.ActionLink("Delete", "Delete", new { id = order.OrderID })
    </td>
</tr>}
</table>
```

Details.cshtml

```
@model WebApplication1.Models.Orderbook

@{ ViewBag.Title = "Order Details"; }

<h2>Order Details</h2>

<table class="table">
    <tr>
        <td><strong>Order ID:</strong></td>
        <td>@Model.OrderID</td>
    </tr>
    <tr>
        <td><strong>Customer ID:</strong></td>
        <td>@Model.CustomerID</td>
    </tr>
    <tr>
        <td><strong>Order Date:</strong></td>
        <td>@Model.OrderDate.ToShortDateString()</td>
    </tr>
    <tr>
        <td><strong>Total Amount:</strong></td>
        <td>@Model.TotalAmount.ToString("C")</td>
    </tr>
</table>

@Html.ActionLink("Back to List", "Index")
```

Delete.cshtml

```
@model WebApplication1.Models.Orderbook

@{ ViewBag.Title = "Delete Order"; }

<h2>Delete Order</h2>

<p>Are you sure you want to delete this order?</p>

<table class="table">
    <tr>
        <td><strong>Order ID:</strong></td>
        <td>@Model.OrderID</td>
    </tr>
    <tr>
        <td><strong>Customer ID:</strong></td>
        <td>@Model.CustomerID</td>
    </tr>
    <tr>
        <td><strong>Order Date:</strong></td>
        <td>@Model.OrderDate.ToShortDateString()</td>
    </tr>
    <tr>
        <td><strong>Total Amount:</strong></td>
        <td>@Model.TotalAmount.ToString("C")</td>
    </tr>
</table>

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

                <div class="form-actions no-color">
                    <input type="submit" value="Delete" class="btn btn-danger" /> |
                    @Html.ActionLink("Back to List", "Index")
                </div>}
```

Admin – Report generator

OrderReport.cshtml

```
@model List<WebApplication1.Models.Orderbook>

@{ ViewBag.Title = "Order Report"; }

<h2>Order Report</h2>

<div class="row">
    <div class="col-md-6">
        <table class="table">
            <thead>
                <tr>
                    <th>Order ID</th>
                    <th>Customer ID</th>
                    <th>Order Date</th>
                    <th>Total Amount</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var order in Model)
                {
                    <tr>
                        <td>@order.OrderID</td>
                        <td>@order.CustomerID</td>
                        <td>@order.OrderDate.ToString("yyyy-MM-dd")</td>
                        <td>@order.TotalAmount</td>
                    </tr>
}
            </tbody>
        </table>
    </div>
    <div class="col-md-6">
        <canvas id="orderChart" width="400" height="200"></canvas>
    </div>
</div>

@section scripts {
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.min.js"></script>
    <script>
        var orderDates = [];
        var totalAmounts = [];
        @foreach (var order in Model)
        {
            <text>orderDates.push('@order.OrderDate.ToString("yyyy-MM-dd")');</text>
            <text>totalAmounts.push('@order.TotalAmount');</text>
        }

        var ctx = document.getElementById('orderChart').getContext('2d');
        var orderChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: orderDates,
                datasets: [{
                    label: 'Total Amount',
                    data: totalAmounts,
                    backgroundColor: 'rgba(54, 162, 235, 0.2)',
                    borderColor: 'rgba(54, 162, 235, 1)',
                    borderWidth: 1,
                    lineTension: 0.1
                }]
            },
            options: {
                scales: {
                    yAxes: [{
                        ticks: {
                            beginAtZero: true
                        }
                    }]
                }
            }
        });
    </script>
}
```

Book

Search.cshtml

```
@model WebApplication1.Models.BookSearchViewModel

@{ ViewBag.Title = "Search Books"; }

<h2>Search Books</h2>

@using (Html.BeginForm("Search", "Book", FormMethod.Get))
{
<div class="form-group">
    @Html.LabelFor(m => m.SearchTerm)
    @Html.TextBoxFor(m => m.SearchTerm, new { @class = "form-control" })
</div>
                <button type="submit" class="btn btn-primary">Search</button>}

@if (Model.Results != null && Model.Results.Count > 0)
{
<h3>Search Results</h3>
                <table class="table">
                    <thead>
                        <tr>
                            <th>Title</th>
                            <th>Author</th>
                            <th>Description</th>
                            <th>Price</th>
                            <th>Category</th>
                            <th>Created At</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var book in Model.Results)
                        {
        <tr>
            <td>@book.Title</td>
            <td>@book.Author</td>
            <td>@book.Description</td>
            <td>@book.Price</td>
            <td>@book.Category</td>
            <td>@book.CreatedAt</td>
        </tr>}
                    </tbody>
                </table> }
                            else if (Model.Results != null)
                            {
                <p>No books found matching your search criteria.</p>}
```

Customer

Dashbord.cshtml

```
@{ ViewBag.Title = "Dashboard"; }

<h2 class="text-center mb-4">Welcome to Your Dashboard</h2>

<div class="row">
    <div class="col-md-4">
        <div class="card">
            <img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSxH2E_OUJnUmO00wfsDeF6kC0eSKNBSTNKbw&s"
class="card-img-top" alt="Book Image">
            <div class="card-body">
                <h5 class="card-title">Recommended Book</h5>
                <p class="card-text">Description of recommended book.</p>
                <button class="btn btn-outline-danger mt-2 like-btn"><i class="bi
bi-heart"></i> Like <span class="like-count">0</span></button>
            </div>
        </div>
    </div>
    <div class="col-md-8">
        <div class="card">
```

```
            <div class="card-body">
                <h5 class="card-title">Your Reading Progress</h5>
                <div class="progress">
                    <div class="progress-bar bg-success" role="progressbar"
style="width: 60%;" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100">60%
Complete</div>
                </div>
                <p class="card-text mt-2">Keep up the good work!</p>
            </div>
        </div>
        <div class="card mt-3">
            <div class="card-body">
                <h5 class="card-title">Recommended Books</h5>
                <ul class="list-group">
                    <li class="list-group-item">Pride and Prejudice</li>
                    <li class="list-group-item">The Catcher in the Rye</li>
                    <li class="list-group-item">The Lord of the Rings</li>
                </ul>
            </div>
        </div>
    </div>
</div>

<div class="row mt-4">
    <div class="col-md-4">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Navigation</h5>
                <ul class="list-group">
                    <li class="list-group-item"><a href="@Url.Action("Index",
"OrderBook")" class="text-dark">Order Books</a></li>
                    <li class="list-group-item"><a href="@Url.Action("Search",
"Book")" class="text-dark">Search for Book Details</a></li>
                    <li class="list-group-item"><a href="@Url.Action("Feedback",
"Feedback")" class="text-dark">Feedback</a></li>
                </ul>
            </div>
        </div>
    </div>
</div>

@section scripts {
    <script>
        $('.like-btn').click(function () {
            var count = parseInt($(this).find('.like-count').text());
            $(this).find('.like-count').text(count + 1);
        });
    </script>
}
```

Register.cshtml

```
@model WebApplication1.Models.CustomerModel

@{ ViewBag.Title = "Register"; }

<h2>Register</h2>

@using (Html.BeginForm("Register", "Customer", FormMethod.Post))
{
@Html.AntiForgeryToken()

                <div class="form-group">
                    @Html.LabelFor(model => model.FirstName)
                    @Html.TextBoxFor(model => model.FirstName, new { @class = "form-
control" })
                    @Html.ValidationMessageFor(model => model.FirstName)
                </div>

                                <div class="form-group">
                                    @Html.LabelFor(model => model.LastName)
                                    @Html.TextBoxFor(model => model.LastName, new {
@class = "form-control" })
                                    @Html.ValidationMessageFor(model =>
model.LastName)
                                </div>

                                                <div class="form-group">
                                                    @Html.LabelFor(model =>
model.Email)
                                                    @Html.TextBoxFor(model =>
model.Email, new { @class = "form-control" })
                                                    @Html.ValidationMessageFor(model
=> model.Email)
                                                </div>

                                                                <div class="form-
group">
@Html.LabelFor(model => model.PasswordHash)
@Html.PasswordFor(model => model.PasswordHash, new { @class = "form-control" })
@Html.ValidationMessageFor(model => model.PasswordHash)
                                                                </div>


<button type="submit" class="btn btn-primary">Register</button>}
```

RegistrationSuccess.cshtml

```
@{ ViewBag.Title = "Registration Success"; }

<h2>Registration Successful</h2>
<p>Your registration was successful. You can now login with your credentials.</p>
```

Login.cshtml

```
@model WebApplication1.Models.LoginViewModel

@{ ViewBag.Title = "Login"; }

<h2>Login</h2>

@using (Html.BeginForm("Login", "Customer", FormMethod.Post))
{
@Html.AntiForgeryToken()

                <div class="form-group">
                    @Html.LabelFor(model => model.Email)
                    @Html.TextBoxFor(model => model.Email, new { @class = "form-
control" })
                    @Html.ValidationMessageFor(model => model.Email, "", new {
@class = "text-danger" })
                </div>

                                <div class="form-group">
                                    @Html.LabelFor(model => model.Password)
                                    @Html.PasswordFor(model => model.Password, new {
@class = "form-control" })
                                    @Html.ValidationMessageFor(model =>
model.Password, "", new { @class = "text-danger" })
                                </div>

                                                <button type="submit" class="btn
btn-primary">Login</button>}
```

Feedback

Feedback.cshtml

```
@model WebApplication1.Models.Feedback

@{ ViewBag.Title = "Give Feedback"; }

<h2>Give Feedback</h2>

@using (Html.BeginForm("Feedback", "Feedback", FormMethod.Post))
{
@Html.AntiForgeryToken()

                <div class="form-group">
                    @Html.LabelFor(model => model.CustomerID)
                    @Html.EditorFor(model => model.CustomerID, new { htmlAttributes
= new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.CustomerID)
                </div>

                                <div class="form-group">
                                    @Html.LabelFor(model => model.BookID)
                                    @Html.EditorFor(model => model.BookID, new {
htmlAttributes = new { @class = "form-control" } })
                                    @Html.ValidationMessageFor(model =>
model.BookID)
                                </div>

                                                <div class="form-group">
                                                    @Html.LabelFor(model =>
model.Rating)
                                                    @Html.EditorFor(model =>
model.Rating, new { htmlAttributes = new { @class = "form-control" } })
                                                    @Html.ValidationMessageFor(model
=> model.Rating)
                                                </div>

                                                                <div class="form-
group">

@Html.LabelFor(model => model.Comment)

@Html.EditorFor(model => model.Comment, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Comment)
                                                                </div>


<button type="submit" class="btn btn-primary">Submit Feedback</button>}
```

OrderBook

Index.cshtml

```
@model IEnumerable<WebApplication1.Models.Orderbook>

@{ ViewBag.Title = "Index"; }

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

<table class="table">
    <tr>
        <th>Order ID</th>
        <th>Customer ID</th>
        <th>Order Date</th>
        <th>Total Amount</th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>@Html.DisplayFor(modelItem => item.OrderID)</td>
    <td>@Html.DisplayFor(modelItem => item.CustomerID)</td>
    <td>@Html.DisplayFor(modelItem => item.OrderDate)</td>
    <td>@Html.DisplayFor(modelItem => item.TotalAmount)</td>
    <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.OrderID }) |
        @Html.ActionLink("Details", "Details", new { id = item.OrderID }) |
        @Html.ActionLink("Delete", "Delete", new { id = item.OrderID })
    </td>
</tr>}

</table>
```

Create.cshtml

```
@model WebApplication1.Models.Orderbook

@{ ViewBag.Title = "Create"; }

<h2>Create</h2>

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

                <div class="form-horizontal">
                    <h4>Orderbook</h4>
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class = "text-danger"
})
                    <div class="form-group">
                        @Html.LabelFor(model => model.CustomerID, htmlAttributes:
new { @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.CustomerID, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.CustomerID,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.OrderDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.OrderDate, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.OrderDate, "",
new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.TotalAmount, htmlAttributes:
new { @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.TotalAmount, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.TotalAmount,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        <div class="col-md-offset-2 col-md-10">
                            <input type="submit" value="Create" class="btn btn-
default" />
                        </div>
                    </div>
                </div>}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Details.cshtml

```
@model WebApplication1.Models.Orderbook

@{ ViewBag.Title = "Details"; }

<h2>Details</h2>

<div>
    <h4>Orderbook</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.OrderID)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.OrderID)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.CustomerID)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.CustomerID)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.OrderDate)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.OrderDate)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.TotalAmount)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.TotalAmount)
        </dd>
    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.OrderID }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model WebApplication1.Models.Orderbook

@{ ViewBag.Title = "Edit"; }

<h2>Edit</h2>

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

                <div class="form-horizontal">
                    <h4>Orderbook</h4>
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class = "text-danger"
})
                    @Html.HiddenFor(model => model.OrderID)

                    <div class="form-group">
                        @Html.LabelFor(model => model.CustomerID, htmlAttributes:
new { @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.CustomerID, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.CustomerID,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.OrderDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.OrderDate, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.OrderDate, "",
new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        @Html.LabelFor(model => model.TotalAmount, htmlAttributes:
new { @class = "control-label col-md-2" })
                        <div class="col-md-10">
                            @Html.EditorFor(model => model.TotalAmount, new {
htmlAttributes = new { @class = "form-control" } })
                            @Html.ValidationMessageFor(model => model.TotalAmount,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-group">
                        <div class="col-md-offset-2 col-md-10">
```

```
                            <input type="submit" value="Save" class="btn btn-
default" />
                        </div>
                    </div>
                </div>}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model WebApplication1.Models.Orderbook

@{ ViewBag.Title = "Delete"; }

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Orderbook</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.OrderID)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.OrderID)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.CustomerID)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.CustomerID)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.OrderDate)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.OrderDate)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.TotalAmount)
        </dt>
        <dd>
```

```
            @Html.DisplayFor(model => model.TotalAmount)
        </dd>
    </dl>

    @using (Html.BeginForm())
    {
@Html.AntiForgeryToken()
                <div class="form-actions no-color">
                    <input type="submit" value="Delete" class="btn btn-default" /> |
                    @Html.ActionLink("Back to List", "Index")
                </div>}
</div>
```

Order

OrderSubmitted.cshtml

```
@{ ViewBag.Title = "Order Submitted"; }

<h2>Order Submitted</h2>

<p>Your order has been submitted successfully!</p>
```

- **Controller**

AccountController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class AccountController : Controller
    {
        private YourDbContext db = new YourDbContext();

        [HttpGet]
        public ActionResult Registration()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Register(CustomerModel model)
        {
            if (ModelState.IsValid)
            {
                db.Customers.Add(model);
                db.SaveChanges();
                return RedirectToAction("RegistrationSuccess");
            }
            return View(model);
        }

        public ActionResult RegistrationSuccess()
        {
            return View();
        }
    }
}
```

AdminController.cs

```
using System;
using System.Data.SqlClient;
using System.Security.Cryptography;
using System.Text;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class AdminController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";

        public ActionResult Register()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Register(Admin model)
        {
            if (ModelState.IsValid)
            {
                if (!string.IsNullOrEmpty(model.PasswordHash))
                {
                    model.PasswordHash = ComputeSha256Hash(model.PasswordHash);
                }
                model.CreatedAt = DateTime.Now;

                using (SqlConnection connection = new
SqlConnection(connectionString))
                {
                    string sql = "INSERT INTO Admins (Username, PasswordHash,
CreatedAt) VALUES (@Username, @PasswordHash, @CreatedAt)";
                    SqlCommand command = new SqlCommand(sql, connection);
                    command.Parameters.AddWithValue("@Username", model.Username);


                    if (!string.IsNullOrEmpty(model.PasswordHash))
                    {
                        command.Parameters.AddWithValue("@PasswordHash",
model.PasswordHash);
                    }
                    else
                    {

                        ModelState.AddModelError("", "Password cannot be null or
empty.");
                        return View(model);
                    }

                    command.Parameters.AddWithValue("@CreatedAt", model.CreatedAt);
                    connection.Open();
                    command.ExecuteNonQuery();
                }

                return RedirectToAction("Login");
            }

            return View(model);
        }


        public ActionResult Login()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(string username, string password)
        {
            if (string.IsNullOrEmpty(password))
            {
                ModelState.AddModelError("", "Password cannot be empty.");
                return View();
            }

            string passwordHash = ComputeSha256Hash(password);

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT AdminID, Username FROM Admins WHERE Username =
@Username AND PasswordHash = @PasswordHash";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@Username", username);
                command.Parameters.AddWithValue("@PasswordHash", passwordHash);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    Session["AdminID"] = reader.GetInt32(0);
                    Session["Username"] = reader.GetString(1);
                    return RedirectToAction("Dashboard");
                }
                else
                {
                    ModelState.AddModelError("", "Invalid login attempt.");
                }
            }

            return View();
        }


        public ActionResult Dashboard()
        {
            if (Session["AdminID"] == null)
            {
                return RedirectToAction("Login");
            }

            return View();
        }


        public ActionResult ManageBooks()
        {
            return View();
        }

        public ActionResult ManageCustomers()
        {
            return View();
        }

        public ActionResult ManageOrders()
        {
            return View();
        }

        public ActionResult GenerateReports()
        {
            return View();
        }

        private static string ComputeSha256Hash(string rawData)
        {
            using (SHA256 sha256Hash = SHA256.Create())
            {
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }
    }
}
```

AdminBookController.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class AdminBookController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";


        public ActionResult Index()
        {
            List<Book> books = new List<Book>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Books";
                SqlCommand command = new SqlCommand(sql, connection);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    books.Add(new Book
                    {
                        BookID = reader.GetInt32(0),
                        Title = reader.GetString(1),
                        Author = reader.GetString(2),
                        Description = reader.GetString(3),
                        Price = reader.GetDecimal(4),
                        Category = reader.GetString(5),
                        CreatedAt = reader.GetDateTime(6)
                    });
                }
            }
            return View(books);
        }


        public ActionResult Create()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Book book)
        {
            if (ModelState.IsValid)
            {
                if (!string.IsNullOrEmpty(book.Author))
                {
                    book.CreatedAt = DateTime.Now;
                    using (SqlConnection connection = new
SqlConnection(connectionString))
                    {
                        string sql = "INSERT INTO Books (Title, Author, Description,
Price, Category, CreatedAt) VALUES (@Title, @Author, @Description, @Price,
@Category, @CreatedAt)";
                        SqlCommand command = new SqlCommand(sql, connection);
                        command.Parameters.AddWithValue("@Title", book.Title);
                        command.Parameters.AddWithValue("@Author", book.Author);
                        command.Parameters.AddWithValue("@Description",
book.Description);
                        command.Parameters.AddWithValue("@Price", book.Price);
                        command.Parameters.AddWithValue("@Category", book.Category);
                        command.Parameters.AddWithValue("@CreatedAt",
book.CreatedAt);

                        connection.Open();
                        command.ExecuteNonQuery();
                    }
                    return RedirectToAction("Index");
                }
                else
                {
                    ModelState.AddModelError("", "Author cannot be null or empty.");
                }
            }

            return View(book);
        }


        public ActionResult Edit(int id)
        {
            Book book = null;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Books WHERE BookID = @BookID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@BookID", id);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    book = new Book
                    {
                        BookID = reader.GetInt32(0),
                        Title = reader.GetString(1),
                        Author = reader.GetString(2),
                        Description = reader.GetString(3),
                        Price = reader.GetDecimal(4),
                        Category = reader.GetString(5),
                        CreatedAt = reader.GetDateTime(6)
                    };
                }
            }
            if (book == null)
            {
                return HttpNotFound();
            }
            return View(book);
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Book book)
        {
            if (ModelState.IsValid)
            {
                using (SqlConnection connection = new
SqlConnection(connectionString))
                {
                    string sql = "UPDATE Books SET Title = @Title, Author = @Author,
Description = @Description, Price = @Price, Category = @Category WHERE BookID =
@BookID";
                    SqlCommand command = new SqlCommand(sql, connection);
                    command.Parameters.AddWithValue("@Title", book.Title);
                    command.Parameters.AddWithValue("@Author", book.Author);
                    command.Parameters.AddWithValue("@Description",
book.Description);
                    command.Parameters.AddWithValue("@Price", book.Price);
                    command.Parameters.AddWithValue("@Category", book.Category);
                    command.Parameters.AddWithValue("@BookID", book.BookID);

                    connection.Open();
                    command.ExecuteNonQuery();
                }
                return RedirectToAction("Index");
            }
            return View(book);
        }


        public ActionResult Delete(int id)
        {
            Book book = null;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Books WHERE BookID = @BookID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@BookID", id);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    book = new Book
                    {
                        BookID = reader.GetInt32(0),
                        Title = reader.GetString(1),
                        Author = reader.GetString(2),
                        Description = reader.GetString(3),
                        Price = reader.GetDecimal(4),
                        Category = reader.GetString(5),
                        CreatedAt = reader.GetDateTime(6)
                    };
                }
            }
            if (book == null)
            {
                return HttpNotFound();
            }
            return View(book);
        }


        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "DELETE FROM Books WHERE BookID = @BookID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@BookID", id);

                connection.Open();
                command.ExecuteNonQuery();
            }
            return RedirectToAction("Index");
        }
    }
}
```

AdminCustomerController.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class AdminCustomerController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";


        public ActionResult Index()
        {
            List<Customer> customers = new List<Customer>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Customers";
                SqlCommand command = new SqlCommand(sql, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    customers.Add(new Customer
                    {
                        CustomerID = reader.GetInt32(0),
                        FirstName = reader.GetString(1),
                        LastName = reader.GetString(2),
                        Email = reader.GetString(3),
                        PasswordHash = reader.GetString(4),
                        CreatedAt = reader.GetDateTime(5)
                    });
                }
            }

            return View(customers);
        }


        public ActionResult Details(int id)
        {
            Customer customer = GetCustomerById(id);
            if (customer == null)
            {
                return HttpNotFound();
            }
            return View(customer);
        }


        public ActionResult Delete(int id)
        {
            Customer customer = GetCustomerById(id);
            if (customer == null)
            {
                return HttpNotFound();
            }
            return View(customer);
        }


        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "DELETE FROM Customers WHERE CustomerID = @CustomerID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@CustomerID", id);
```

```
                connection.Open();
                command.ExecuteNonQuery();
            }
            return RedirectToAction("Index");
        }


        private Customer GetCustomerById(int id)
        {
            Customer customer = null;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Customers WHERE CustomerID =
@CustomerID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@CustomerID", id);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    customer = new Customer
                    {
                        CustomerID = reader.GetInt32(0),
                        FirstName = reader.GetString(1),
                        LastName = reader.GetString(2),
                        Email = reader.GetString(3),
                        PasswordHash = reader.GetString(4),
                        CreatedAt = reader.GetDateTime(5)
                    };
                }
            }
            return customer;
        }
    }
}
```

AdminOrderController.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class AdminOrderController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";


        public ActionResult Index()
        {
            List<Orderbook> orders = new List<Orderbook>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Orders";
                SqlCommand command = new SqlCommand(sql, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    orders.Add(new Orderbook
                    {
                        OrderID = reader.GetInt32(0),
                        CustomerID = reader.GetInt32(1),
                        OrderDate = reader.GetDateTime(2),
                        TotalAmount = reader.GetDecimal(3)
                    });
                }
            }

            return View(orders);
        }


        public ActionResult Details(int id)
        {
            Orderbook order = GetOrderById(id);
            if (order == null)
            {
                return HttpNotFound();
            }
            return View(order);
        }


        public ActionResult Delete(int id)
        {
            Orderbook order = GetOrderById(id);
            if (order == null)
            {
                return HttpNotFound();
            }
            return View(order);
        }


        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "DELETE FROM Orders WHERE OrderID = @OrderID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@OrderID", id);

                connection.Open();
                command.ExecuteNonQuery();
            }
            return RedirectToAction("Index");
        }

        private Orderbook GetOrderById(int id)
        {
            Orderbook order = null;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Orders WHERE OrderID = @OrderID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@OrderID", id);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    order = new Orderbook
                    {
                        OrderID = reader.GetInt32(0),
                        CustomerID = reader.GetInt32(1),
                        OrderDate = reader.GetDateTime(2),
                        TotalAmount = reader.GetDecimal(3)
                    };
                }
            }
            return order;
        }
    }
}
```

AdminReportController.cs

```
using System;
using System.Collections.Generic;
using System.Web.Mvc;
using WebApplication1.Models;
using System.Data.SqlClient;

namespace WebApplication1.Controllers
{
    public class AdminReportController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";


        public ActionResult OrderReport()
        {
            List<Orderbook> orders = GetOrderDataFromDatabase();
            return View(orders);
        }


        private List<Orderbook> GetOrderDataFromDatabase()
        {
            List<Orderbook> orders = new List<Orderbook>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Orderbook";
                SqlCommand command = new SqlCommand(sql, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    orders.Add(new Orderbook
                    {
                        OrderID = Convert.ToInt32(reader["OrderID"]),
                        CustomerID = Convert.ToInt32(reader["CustomerID"]),
                        OrderDate = Convert.ToDateTime(reader["OrderDate"]),
                        TotalAmount = Convert.ToDecimal(reader["TotalAmount"])
                    });
                }
            }

            return orders;
        }
    }
}
```

BookController.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class BookController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";

        // GET: Book/Search
        public ActionResult Search(string searchTerm)
        {
            var model = new BookSearchViewModel();

            if (!string.IsNullOrEmpty(searchTerm))
            {
                model.SearchTerm = searchTerm;
                model.Results = SearchBooks(searchTerm);
            }

            return View(model);
        }

        private List<Book> SearchBooks(string searchTerm)
        {
            var books = new List<Book>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string query = "SELECT * FROM Books WHERE Title LIKE @SearchTerm OR 
Author LIKE @SearchTerm";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@SearchTerm", "%" + searchTerm 
+ "%");

                    connection.Open();
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            var book = new Book
                            {
                                BookID = Convert.ToInt32(reader["BookID"]),
                                Title = reader["Title"].ToString(),
                                Author = reader["Author"].ToString(),
                                Description = reader["Description"].ToString(),
                                Price = Convert.ToDecimal(reader["Price"]),
                                Category = reader["Category"].ToString(),
                                CreatedAt = Convert.ToDateTime(reader["CreatedAt"])
                            };

                            books.Add(book);
                        }
                    }
                }
            }

            return books;
        }
    }
}
```

CustomerController.cs

```
using System;
using System.Data.SqlClient;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class CustomerController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";


        public ActionResult Register()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Register(CustomerModel model)
        {
            if (ModelState.IsValid)
            {
                SaveCustomerToDatabase(model);
                return RedirectToAction("RegistrationSuccess");
            }

            return View(model);
        }
        private void SaveCustomerToDatabase(CustomerModel model)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string sql = "INSERT INTO Customers (FirstName, LastName, Email, 
PasswordHash, CreatedAt) " +
                             "VALUES (@FirstName, @LastName, @Email, @PasswordHash, 
@CreatedAt)";
                using (SqlCommand command = new SqlCommand(sql, connection))
                {
                    command.Parameters.AddWithValue("@FirstName", model.FirstName);
                    command.Parameters.AddWithValue("@LastName", model.LastName);
                    command.Parameters.AddWithValue("@Email", model.Email);
                    command.Parameters.AddWithValue("@PasswordHash", 
model.PasswordHash);
                    command.Parameters.AddWithValue("@CreatedAt", DateTime.Now);
                    command.ExecuteNonQuery();
                }
            }
        }


        public ActionResult RegistrationSuccess()
        {
            return View();
        }


        [HttpGet]
        public ActionResult Login()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(LoginViewModel model)
        {
            if (ModelState.IsValid)
            {
                var customer = AuthenticateUser(model.Email, model.Password);
                if (customer != null)
                {
                    return RedirectToAction("Dashboard");
                }

                ModelState.AddModelError("", "Invalid email or password.");
            }
            return View(model);
        }

        private CustomerModel AuthenticateUser(string email, string password)
        {
            CustomerModel customer = null;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string sql = "SELECT * FROM Customers WHERE Email = @Email AND
PasswordHash = @PasswordHash";
                using (SqlCommand command = new SqlCommand(sql, connection))
                {
                    command.Parameters.AddWithValue("@Email", email);
                    command.Parameters.AddWithValue("@PasswordHash", password);

                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            customer = new CustomerModel
                            {
                                CustomerID = (int)reader["CustomerID"],
                                FirstName = reader["FirstName"].ToString(),
                                LastName = reader["LastName"].ToString(),
                                Email = reader["Email"].ToString(),
                                PasswordHash = reader["PasswordHash"].ToString(),
                                CreatedAt = (DateTime)reader["CreatedAt"]
                            };
                        }
                    }
                }
            }

            return customer;
        }


        public ActionResult Dashboard()
        {
            return View();
        }
    }
}
```

FeedbackController.cs

```
using System;
using System.Data.SqlClient;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class FeedbackController : Controller
    {

        [HttpGet]
        public ActionResult Feedback()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Feedback(Feedback model)
        {
            if (ModelState.IsValid)
            {
                SaveFeedbackToDatabase(model);
                return RedirectToAction("FeedbackSubmitted");
            }
            return View(model);
        }


        private void SaveFeedbackToDatabase(Feedback model)
        {
            string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";
            try
            {
                using (SqlConnection connection = new
SqlConnection(connectionString))
                {
                    connection.Open();

                    string sql = "INSERT INTO Feedback (CustomerID, BookID, Rating,
Comment, CreatedAt) " +
                                 "VALUES (@CustomerID, @BookID, @Rating, @Comment,
@CreatedAt)";
                    using (SqlCommand command = new SqlCommand(sql, connection))
                    {
                        command.Parameters.AddWithValue("@CustomerID",
model.CustomerID);
                        command.Parameters.AddWithValue("@BookID", model.BookID);
                        command.Parameters.AddWithValue("@Rating", model.Rating);
                        command.Parameters.AddWithValue("@Comment", model.Comment);
                        command.Parameters.AddWithValue("@CreatedAt", DateTime.Now);

                        command.ExecuteNonQuery();
                    }
                }

            }
            catch
            {

            }
        }


        public ActionResult FeedbackSubmitted()
        {
            return View();
        }
    }
}
```

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

MainController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebApplication1.Controllers
{
    public class MainController : Controller
    {


        public ActionResult Index()
        {
            return View();
        }
    }
}
```

OrderController.cs

```
using System;
using System.Web.Mvc;
using System.Data.SqlClient;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class OrderController : Controller
    {

        [HttpGet]
        public ActionResult Create()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Orderbook model)
        {
            if (ModelState.IsValid)
            {

                SaveOrderToDatabase(model);


                return RedirectToAction("OrderSuccess");
            }


            return View(model);
        }


        private void SaveOrderToDatabase(Orderbook model)
        {

            string connectionString = "YourConnectionStringHere";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();

                string sql = "INSERT INTO Orders (CustomerID, OrderDate, 
TotalAmount) " +
                             "VALUES (@CustomerID, @OrderDate, @TotalAmount); SELECT 
SCOPE_IDENTITY();";
                using (SqlCommand command = new SqlCommand(sql, connection))
                {
                    command.Parameters.AddWithValue("@CustomerID", 
model.CustomerID);
                    command.Parameters.AddWithValue("@OrderDate", model.OrderDate);
                    command.Parameters.AddWithValue("@TotalAmount", 
model.TotalAmount);

                    model.OrderID = Convert.ToInt32(command.ExecuteScalar());
                }
            }
        }


        public ActionResult OrderSuccess()
        {
            return View();
        }
    }
}
```

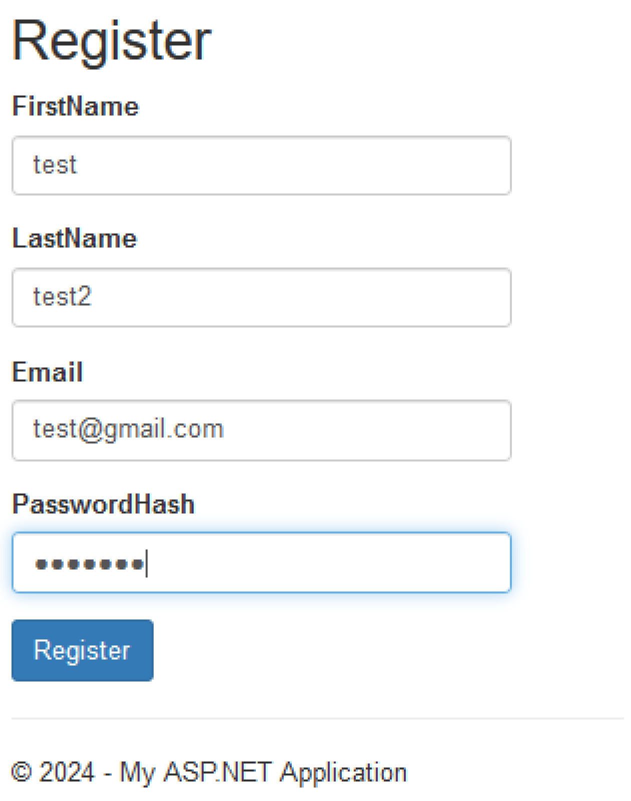
OrderbookController.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class OrderbookController : Controller
    {
        private string connectionString = "Data Source=DESKTOP-
CJT3444\\SQLEXPRESS;Initial Catalog=ebook;Integrated Security=True";


        public ActionResult Index()
        {
            List<Orderbook> orderbooks = new List<Orderbook>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT OrderID, CustomerID, OrderDate, TotalAmount 
FROM Orderbook";
                SqlCommand command = new SqlCommand(sql, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    Orderbook orderbook = new Orderbook
                    {
                        OrderID = reader.GetInt32(0),
                        CustomerID = reader.GetInt32(1),
                        OrderDate = reader.GetDateTime(2),
                        TotalAmount = reader.GetDecimal(3)
                    };
                    orderbooks.Add(orderbook);
                }
            }

            return View(orderbooks);
        }


        public ActionResult Details(int id)
        {
            Orderbook orderbook = null;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT OrderID, CustomerID, OrderDate, TotalAmount 
FROM Orderbook WHERE OrderID = @OrderID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@OrderID", id);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    orderbook = new Orderbook
                    {
                        OrderID = reader.GetInt32(0),
                        CustomerID = reader.GetInt32(1),
                        OrderDate = reader.GetDateTime(2),
                        TotalAmount = reader.GetDecimal(3)
                    };
                }
            }

            if (orderbook == null)
            {
                return HttpNotFound();
            }

            return View(orderbook);
        }


        public ActionResult Create()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Orderbook orderbook)
        {
            if (ModelState.IsValid)
            {
                using (SqlConnection connection = new 
SqlConnection(connectionString))
                {
                    string sql = "INSERT INTO Orderbook (CustomerID, OrderDate, 
TotalAmount) VALUES (@CustomerID, @OrderDate, @TotalAmount)";
                    SqlCommand command = new SqlCommand(sql, connection);
                    command.Parameters.AddWithValue("@CustomerID", 
orderbook.CustomerID);
                    command.Parameters.AddWithValue("@OrderDate", 
orderbook.OrderDate);
                    command.Parameters.AddWithValue("@TotalAmount", 
orderbook.TotalAmount);

                    connection.Open();
                    command.ExecuteNonQuery();
                }

                return RedirectToAction("Index");
            }

            return View(orderbook);
        }


        public ActionResult Edit(int id)
        {
            Orderbook orderbook = null;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT OrderID, CustomerID, OrderDate, TotalAmount 
FROM Orderbook WHERE OrderID = @OrderID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@OrderID", id);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    orderbook = new Orderbook
                    {
                        OrderID = reader.GetInt32(0),
                        CustomerID = reader.GetInt32(1),
                        OrderDate = reader.GetDateTime(2),
                        TotalAmount = reader.GetDecimal(3)
                    };
                }
            }

            if (orderbook == null)
            {
                return HttpNotFound();
            }

            return View(orderbook);
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Orderbook orderbook)
        {
            if (ModelState.IsValid)
            {
                using (SqlConnection connection = new 
SqlConnection(connectionString))
                {
                    string sql = "UPDATE Orderbook SET CustomerID = @CustomerID, 
OrderDate = @OrderDate, TotalAmount = @TotalAmount WHERE OrderID = @OrderID";
                    SqlCommand command = new SqlCommand(sql, connection);
                    command.Parameters.AddWithValue("@OrderID", orderbook.OrderID);
                    command.Parameters.AddWithValue("@CustomerID",
orderbook.CustomerID);
                    command.Parameters.AddWithValue("@OrderDate",
orderbook.OrderDate);
                    command.Parameters.AddWithValue("@TotalAmount",
orderbook.TotalAmount);

                    connection.Open();
                    command.ExecuteNonQuery();
                }

                return RedirectToAction("Index");
            }

            return View(orderbook);
        }


        public ActionResult Delete(int id)
        {
            Orderbook orderbook = null;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "SELECT OrderID, CustomerID, OrderDate, TotalAmount
FROM Orderbook WHERE OrderID = @OrderID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@OrderID", id);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.Read())
                {
                    orderbook = new Orderbook
                    {
                        OrderID = reader.GetInt32(0),
                        CustomerID = reader.GetInt32(1),
                        OrderDate = reader.GetDateTime(2),
                        TotalAmount = reader.GetDecimal(3)
                    };
                }
            }

            if (orderbook == null)
            {
                return HttpNotFound();
            }

            return View(orderbook);
        }


        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sql = "DELETE FROM Orderbook WHERE OrderID = @OrderID";
                SqlCommand command = new SqlCommand(sql, connection);
                command.Parameters.AddWithValue("@OrderID", id);

                connection.Open();
                command.ExecuteNonQuery();
            }

            return RedirectToAction("Index");
        }
    }
}
```

Test plan and test cases

Customer Registration

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC001	New Customer Registration	1. Navigate to registration page 2. Enter valid details 3. Submit form	New account is created, user redirected to login page	New account is created, user redirected to login page	Pass	Black Box	3 min
TC002	Registration with missing fields	1. Navigate to registration page 2. Leave required fields blank 3. Submit form	Error messages displayed for missing fields	Error messages displayed for missing fields	Pass	Black Box	2 min
TC003	Registration with invalid email	1. Navigate to registration page 2. Enter invalid email 3. Submit form	Error message displayed for invalid email format	Error message displayed for invalid email format	Pass	Black Box	2 min


Register

FirstName

test

LastName

test2

Email

test@gmail.com

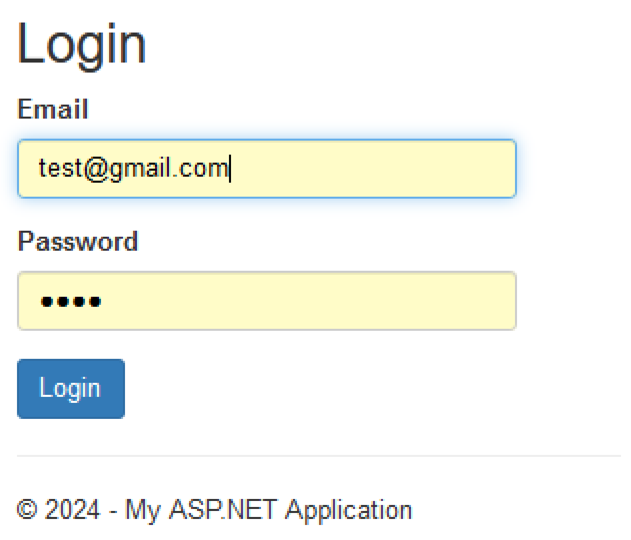
PasswordHash

•••••••

© 2024 - My ASP.NET Application

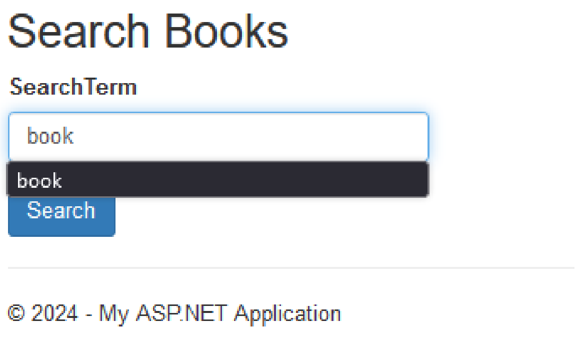
Customer Login

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC004	Customer Login with valid credentials	1. Navigate to login page 2. Enter valid credentials 3. Submit form	User is logged in, redirected to homepage	User is logged in, redirected to homepage	Pass	Black Box	2 min
TC005	Customer Login with invalid credentials	1. Navigate to login page 2. Enter invalid credentials 3. Submit form	Error message displayed for invalid credentials	Error message displayed for invalid credentials	Pass	Black Box	2 min



Book Search

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC006	Search for books as guest	1. Navigate to homepage 2. Enter search query 3. Submit search	List of books matching search query is displayed	List of books matching search query is displayed	Pass	Black Box	3 min
TC007	Search for books as logged in customer	1. Login as customer 2. Enter search query 3. Submit search	List of books matching search query is displayed	List of books matching search query is displayed	Pass	Black Box	3 min



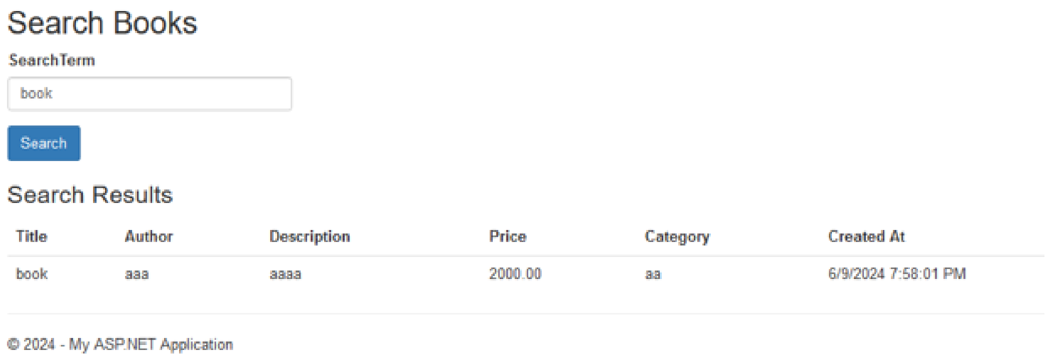
Search Books

SearchTerm

book

book

© 2024 - My ASP.NET Application



Search Books

SearchTerm

book

Search Results

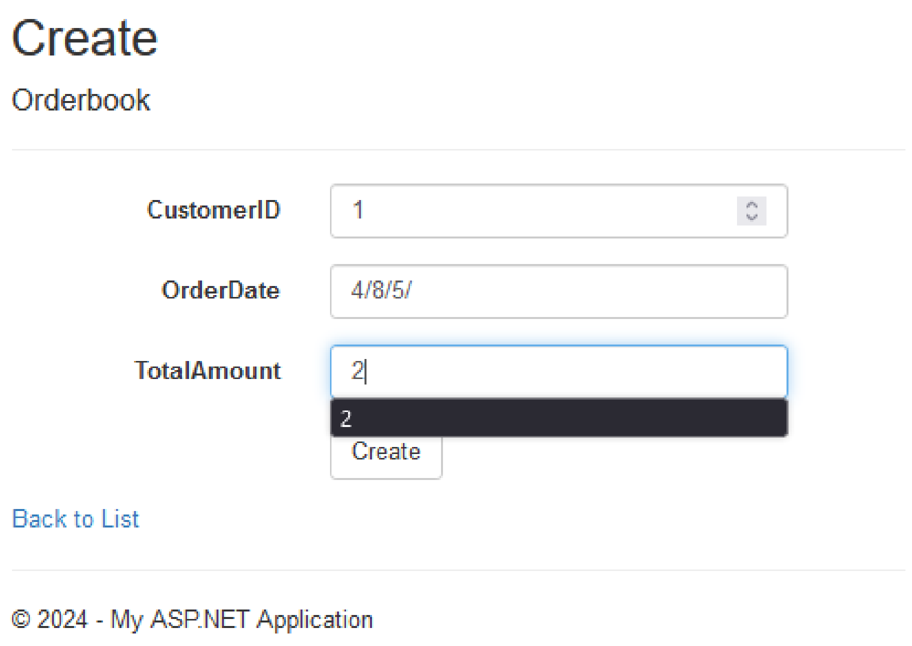
Title	Author	Description	Price	Category	Created At
book	aaa	aaaa	2000.00	aa	6/9/2024 7:58:01 PM

© 2024 - My ASP.NET Application

Book Ordering

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC008	Order a book as logged in customer	1. Login as customer 2. Search for a book 3. Add book to cart 4. Proceed to checkout 5.	Order is placed, confirmation message displayed	Order is placed, confirmation message displayed	Pass	Black Box	4 min

		Complet e order process					
TC00 9	Order a book as guest	1. Search for a book 2. Attempt to add book to cart	Redirected to login page with prompt to login/registe r	Redirected to login page with prompt to login/registe r	Pass	Blac k Box	2 min



Create

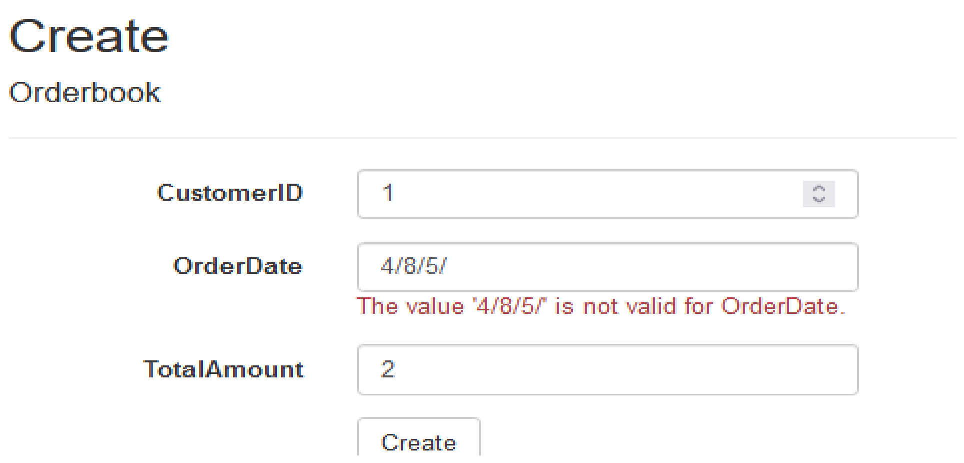
Orderbook

CustomerID 1

OrderDate 4/8/5/

TotalAmount 2

© 2024 - My ASP.NET Application



Create

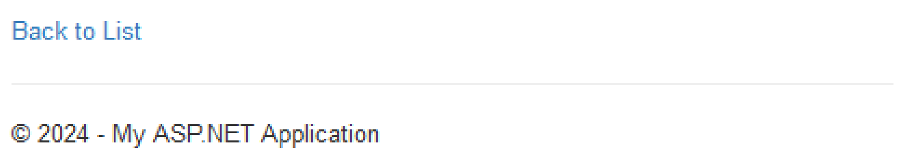
Orderbook

CustomerID 1

OrderDate 4/8/5/

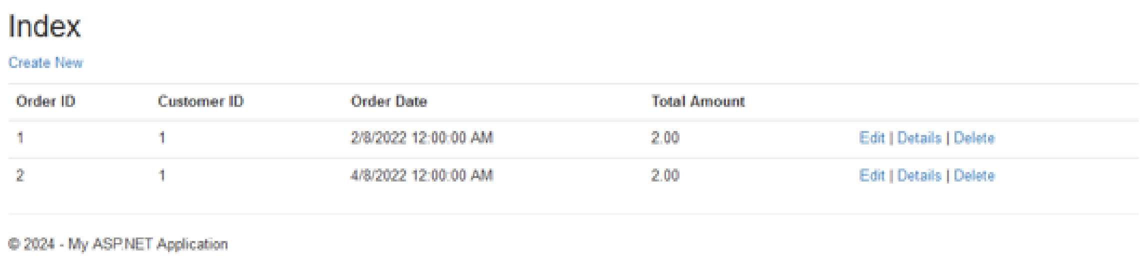
The value '4/8/5/' is not valid for OrderDate.

TotalAmount 2



Back to List

© 2024 - My ASP.NET Application

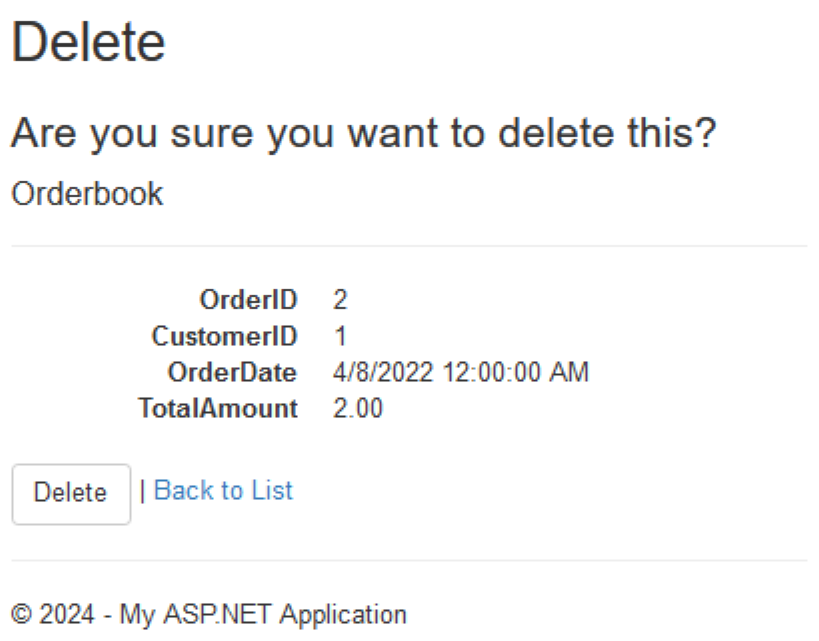


Index

Create New

Order ID	Customer ID	Order Date	Total Amount	
1	1	2/8/2022 12:00:00 AM	2.00	Edit | Details | Delete
2	1	4/8/2022 12:00:00 AM	2.00	Edit | Details | Delete

© 2024 - My ASP.NET Application



Delete

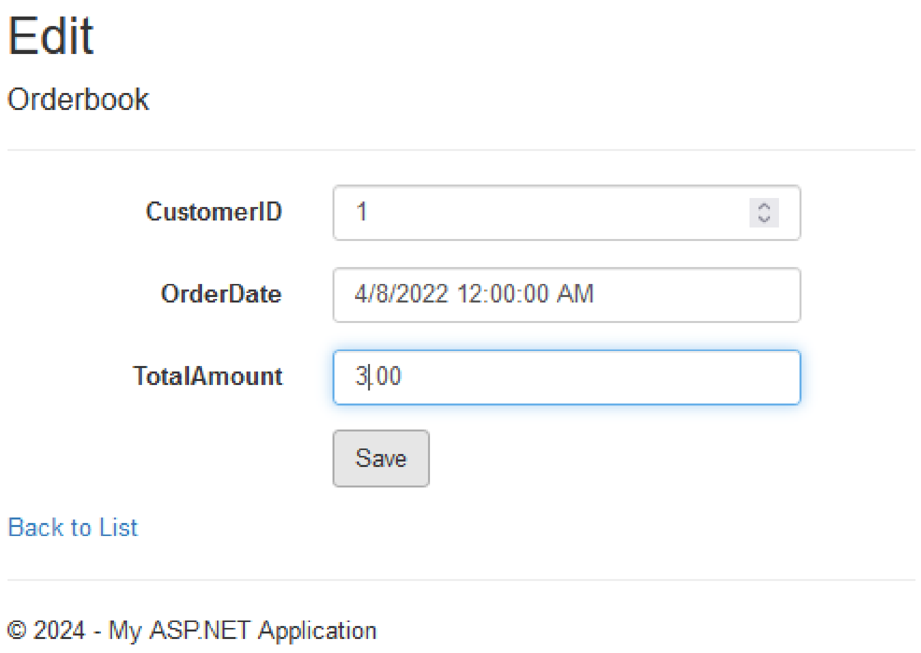
Are you sure you want to delete this?

Orderbook

OrderID	2
CustomerID	1
OrderDate	4/8/2022 12:00:00 AM
TotalAmount	2.00

Delete | Back to List

© 2024 - My ASP.NET Application


Edit

Orderbook

CustomerID 1

OrderDate 4/8/2022 12:00:00 AM

TotalAmount 3.00

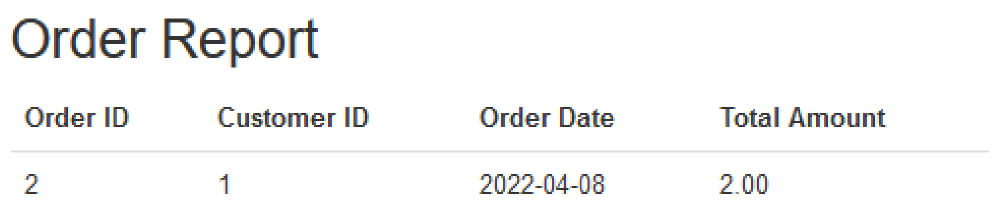
Save

Back to List

© 2024 - My ASP.NET Application

Order Management

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC010	View orders as logged in customer	1. Login as customer 2. Navigate to "My Orders"	List of customer's orders is displayed	List of customer's orders is displayed	Pass	Black Box	3 min
TC011	Delete an order	1. Login as customer 2. Navigate to "My Orders" 3. Select an order to delete 4. Confirm deletion	Order is deleted, confirmation message displayed	Order is deleted, confirmation message displayed	Pass	Black Box	3 min

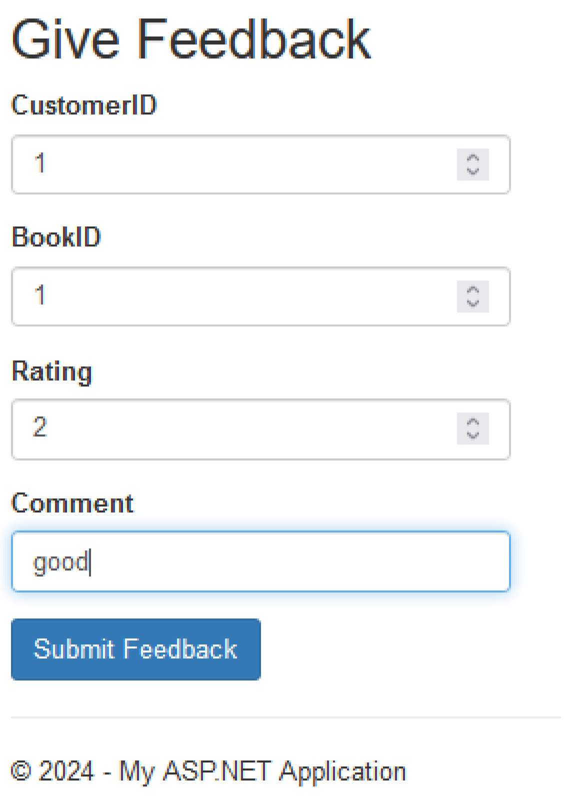


Order Report

Order ID	Customer ID	Order Date	Total Amount
2	1	2022-04-08	2.00

Feedback Submission

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC012	Submit feedback as logged in customer	1. Login as customer 2. Navigate to book details 3. Submit feedback	Feedback is submitted, confirmation message displayed	Feedback is submitted, confirmation message displayed	Pass	Black Box	2 min
TC013	Submit feedback as guest	1. Navigate to book details 2. Attempt to submit feedback	Redirected to login page with prompt to login/register	Redirected to login page with prompt to login/register	Pass	Black Box	2 min


Give Feedback

CustomerID

1

BookID

1

Rating

2

Comment

good

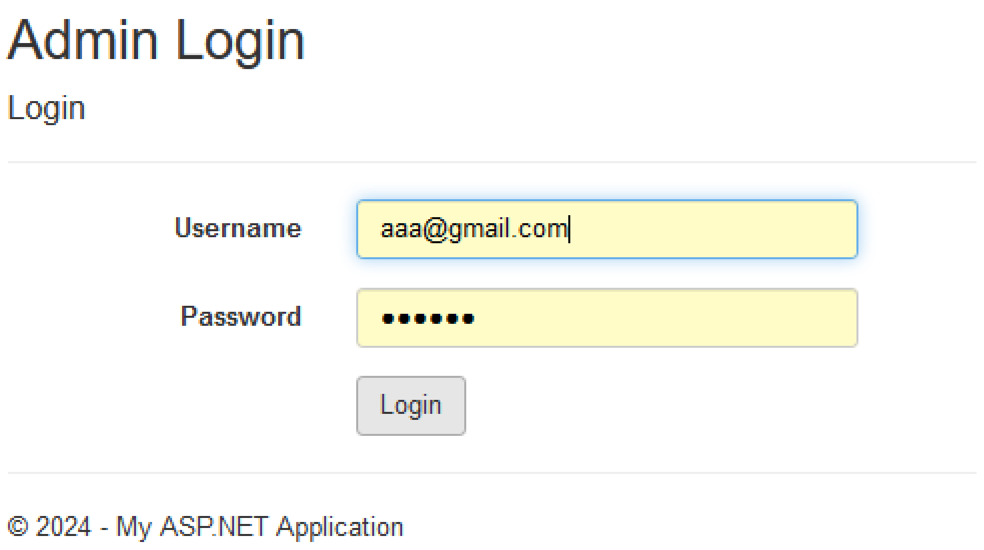
Submit Feedback

© 2024 - My ASP.NET Application

Admin Functionalities

Test Case ID	Description	Steps	Expected Result	Actual Result	Pass/Fail	Type	Duration
TC014	Admin Login	1. Navigate to admin login page 2. Enter valid credentials 3. Submit form	Admin is logged in, redirected to admin dashboard	Admin is logged in, redirected to admin dashboard	Pass	Black Box	2 min
TC015	Manage Books	1. Login as admin 2. Navigate to manage books 3. Add/Edit/Delete book details	Book details are managed successfully	Book details are managed successfully	Pass	White Box	3 min

TC016	Manage Customers	1. Login as admin 2. Navigate to manage customers 3. View/Edit/Delete customer details	Customer details are managed successfully	Customer details are managed successfully	Pass	White Box	3 min
TC017	Manage Orders	1. Login as admin 2. Navigate to manage orders 3. View/Edit/Delete order details	Order details are managed successfully	Order details are managed successfully	Pass	White Box	3 min
TC018	Generate Reports	1. Login as admin 2. Navigate to generate reports 3. Select report criteria 4. Generate report	Report is generated and displayed	Report is generated and displayed	Pass	White Box	4 min

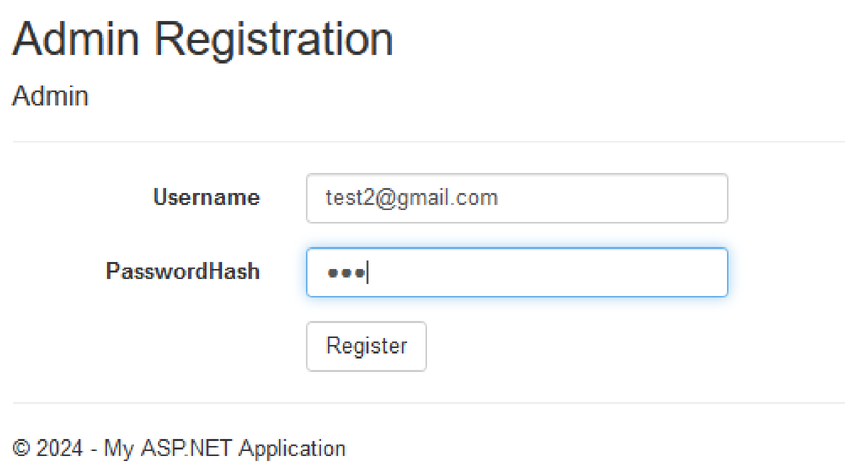

Admin Login

Login

Username: aaa@gmail.com

Password: ••••••

© 2024 - My ASP.NET Application



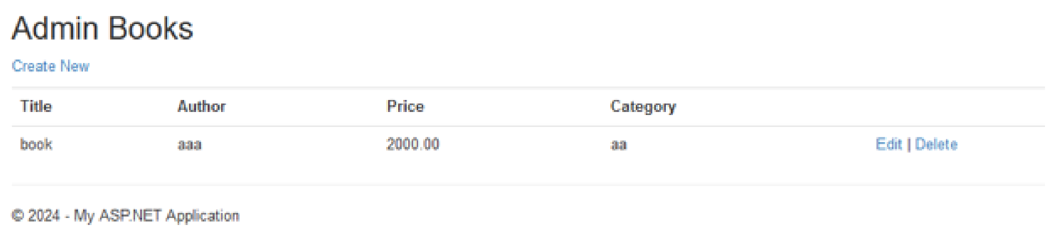
Admin Registration

Admin

Username	test2@gmail.com
PasswordHash	•••

Register

© 2024 - My ASP.NET Application

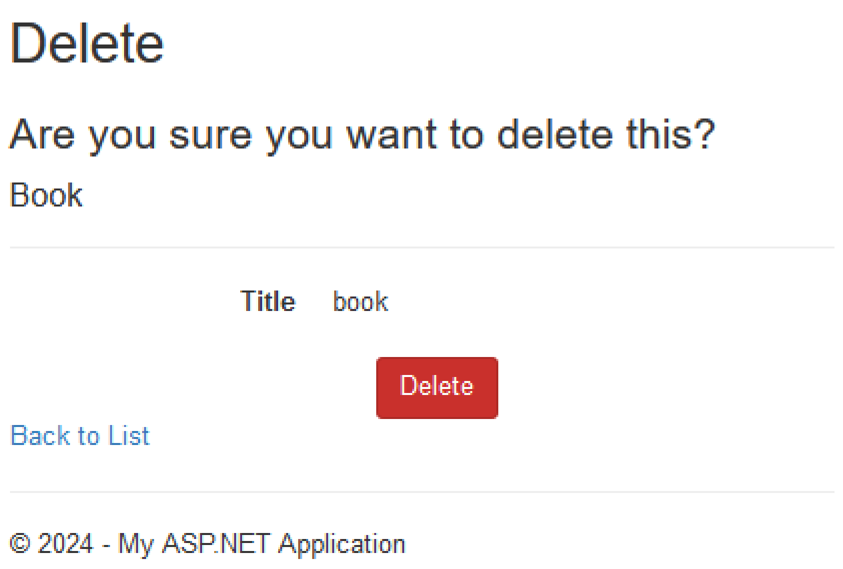


Admin Books

Create New

Title	Author	Price	Category	
book	aaa	2000.00	aa	Edit | Delete

© 2024 - My ASP.NET Application



Delete

Are you sure you want to delete this?

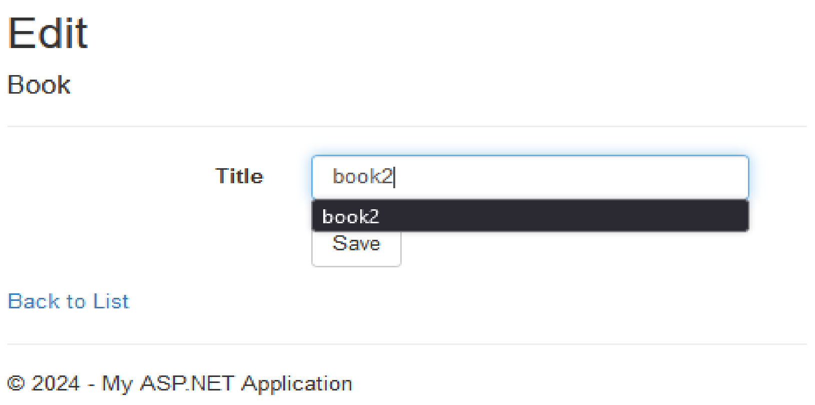
Book

Title	book

Delete

Back to List

© 2024 - My ASP.NET Application



Edit

Book

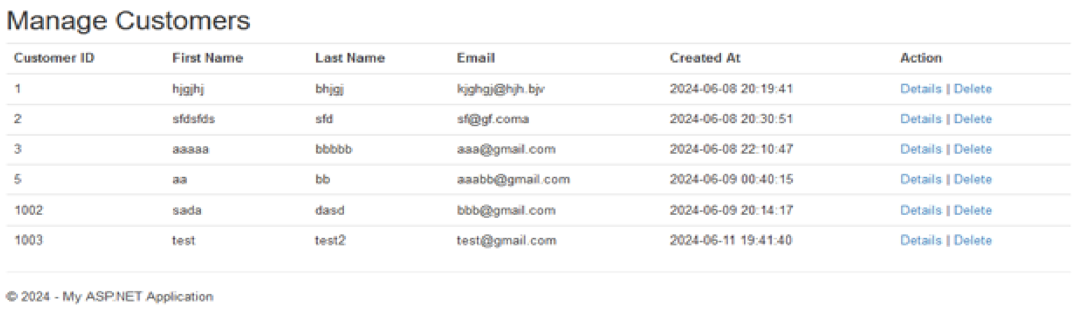
Title: book2

book2

Save

Back to List

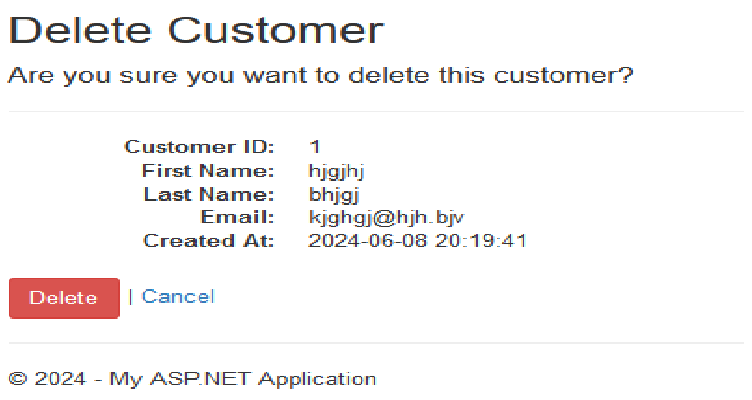
© 2024 - My ASP.NET Application



Manage Customers

Customer ID	First Name	Last Name	Email	Created At	Action
1	hjgjhj	bhjgj	kjghgj@hjh.bjv	2024-06-08 20:19:41	Details | Delete
2	sfdsfds	sfd	sf@gf.coma	2024-06-08 20:30:51	Details | Delete
3	aaaaa	bbbbb	aaa@gmail.com	2024-06-08 22:10:47	Details | Delete
5	aa	bb	aaabb@gmail.com	2024-06-09 00:40:15	Details | Delete
1002	sada	dasd	bbb@gmail.com	2024-06-09 20:14:17	Details | Delete
1003	test	test2	test@gmail.com	2024-06-11 19:41:40	Details | Delete

© 2024 - My ASP.NET Application



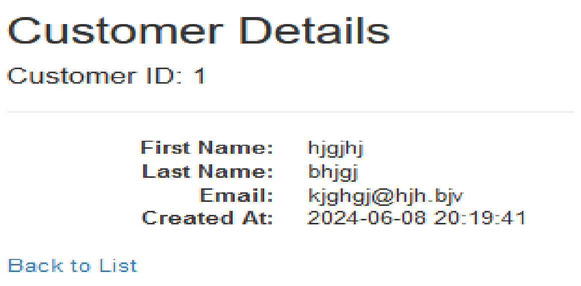
Delete Customer

Are you sure you want to delete this customer?

Customer ID:	1
First Name:	hjgjhj
Last Name:	bhjgj
Email:	kjghgj@hjh.bjv
Created At:	2024-06-08 20:19:41

Delete | Cancel

© 2024 - My ASP.NET Application

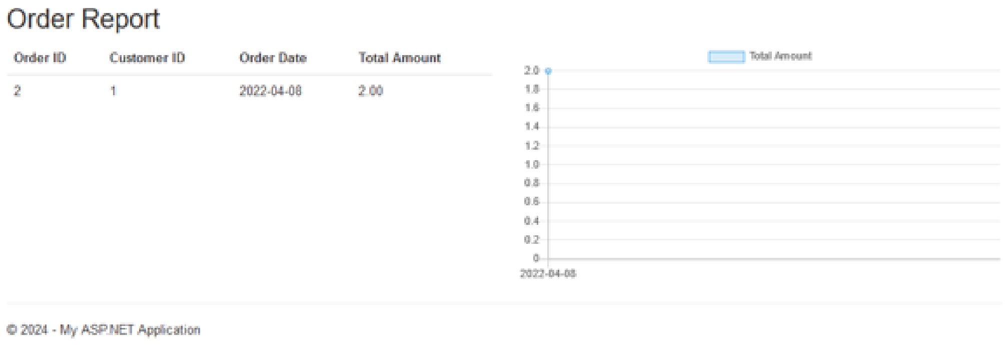


Customer Details

Customer ID: 1

First Name:	hjgjhj
Last Name:	bhjgj
Email:	kjghgj@hjh.bjv
Created At:	2024-06-08 20:19:41

Back to List



Order Report

Order ID	Customer ID	Order Date	Total Amount
2	1	2022-04-08	2.00

© 2024 - My ASP.NET Application

Limitations

Continuous from this, the first time, the group have created a web application for E-Book Pvt. Ltd using ASP. One should be aware that there could be some problems which NET MVC faced or which are being faced. One of the potential challenges is the Availability aspect; it means that one may not access data whenever one wants because increasing the number of customers or the amount of data can cause slow execution and therefore delayed responses. The security concern is another important factor because an organization may employ the basic security measures and it may not be adequate to prevent sophisticated hackers. However, the convenience to the users at the same time might not be quite evident at first attempts of interaction and using the app, for example, an inconvenient or even rather irrelevant to the touch icon might be observed on some devices. It may also provide a low or no interaction with other services, which will create issues with payment processors and shipping integration as seen in the first edition.

The reporting features might be easier to introduce during the first stages and would not include the complete data analysis and data visualization capabilities. The same is also valid in the reverse order; the application can only be developed in English, or it may lack adequate provisions for local languages, making it hard to penetrate the international market. These are as follows These are the limitation They will help when it comes to making adjustment as since the specification of the of the Application is done based on the nature of companies more complex business operation and more diversity readership.

Scope for Future Enhancements

However, there is efficacy to the improvements should be made in the future to get the better of E-Book Pvt. Ltd's web application. Still, there are some lines of improvement concerning the functionality at the site: a possibility to enter the required field with the necessary key by means of filters or voice control and a number of the offered recommendations based on the artificial intelligence system. It is possible to note that presence in both iOS and Android can create a better Mobile hit along with the Webcl/Web application that will complement each other.

For the more specific options it might have been relevant to certain clients or consumers the application of artificial intelligence and machine learning improvements could be called to mind. It is also possible to integrate social supplements, which include such components as book clubs, user groups, and discussion boards, where it is possible to expect rather a high level of activity among users. By enhancing the feedback system to allow multimedia feedback, and the use of such features as badges, leaderboards, and other incentives to make the users want to stay more active in the site, more possibilities can be made.

Taking into account the free trial, the implementation of the subscription will result in the delivery of even more unique content, or closer proximity to new products, and the occasional special discounts to customers, which associates the product with a more premium experience. Several of these could be made more elaborate so that the admin tools would better fit at quite another level where it would meet the work that concerns stock, customers or analysis. Enhancing third-party integration could likely contribute positively to payment processes, delivery instruments, and sharing across channels for a variety of social interlocutors. Perhaps, it is worthwhile to implement the automation of the forwarding of the order, changes in the stock, and interacting with customers last.

Conclusion

The issue that arose from the competition isibility of designing a web application online for E-Book Pvt. Ltd. using ASP. NET MVC gives a great foundation for building effectiveness in the satisfaction of customers and also eradicating most of the bureaucratic tasks. However, specific issues can be addressed immediately, whereas issues with the scale of application, its speed, dependability in terms of security or enhancing the level of user satisfaction may be problematical; however, the potential for growth is enormous. It also aids the application to develop as it matures due to specific user necessities and business demands by merely adding up the already established, advance technologies. The same process will be continued time to time as it is one of the efficient ways to continue to assert the competitiveness of E-Book Pvt. Ltd over the used book market Sri Lanka and to meet the expectations of readers of used books.

End.