An E-BOOK Website (Bookworm)

<u>Using Full Stack Development with C# (.net MVC)</u>

Project Report submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering

of

Maulana Abul Kalam Azad University of Technology

Under the guidance of Anindya Mukherjee

Project Carried Out At

Ardent Computech Pvt Ltd(AnISO9001:2008Certified)

Module-132, SDF Building, Sector-V, Salt Lake City, Kolkata-700091

By

HRICHA CHAKRABORTY(University Roll-34200121031)

Future Institute of Technology

Garia, Kolkata – 700154

Academic Year of Pass out:2021-2025

An E-BOOK Website (Bookworm)

Project Submitted by:

HRICHA CHAKRABORTY (University Roll-34200121031)

Under the guidance of:

Prof. Mr. Anindya Mukherjee

This Project Thesis is submitted in partial fulfilment of the requirement for the Bachelor of Technology in Computer Science & Engineering under Maulana Abul Kalam Azad University of Technology.

Project Submitted By: Hricha Chakraborty

Signature----------------------------------

Under the guidance of: Anindya Mukherjee

signature of the Project guide----------------------------------

Department of Computer Science & Engineering,

Future Institute of Technology

Garia, Kolkata- 700154

CERTIFICATE

This is to certify that this project report titled **An E-book website named Bookworm Using C# Full Stack Development (.net MVC)** Submitted in partial fulfilment of requirements for award of the degree bachelor of Technology in Computer Science & Engineering of Maulana Abul Kalam Azad University of Technology is a faithful record of the original work carried out by -

HRICHA CHAKRABORTY (REGISTRATION NO – 213420100110046 OF 2021-2022)
&
(UNIVERSITY ROLL NUMBER -34200121031)

Under my guidance and supervision. It is further certified that it contains no material, which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except the assistances drawn from other sources, for which due acknowledgement has been made.

(Signature of HOD) *(Signature of project guide)*

……………………………… …………………………………

Anindya Mukherjee

FIT

Garia, Kolkata-700154

DECLARATION

We hereby declare that this project report titled

An E-BOOK Website (Bookworm) Using C# Full Stack Development (.net MVC)

Is our own work carried out as a under graduate student in Future Institute of Technology except to the extent that assistances from other sources are duly acknowledged.

All sources used for this project report have been fully and properly cited. It contains no material which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except where due acknowledgement is made.

Students Name	Signature	Dates
....	....	....
....	....	....
....	....	....
....	....	....

CERTIFICATE OF APPROVAL

We hereby approve this dissertation titled **An E-Book Website (Bookworm) Using C# Full Stack Development (.net MVC)**

Carried out by

Hricha Chakraborty (Reg.No-213420100110046 OF 2021-2022)

(Roll No-34200121031)

Under the guidance of

Anindya Mukherjee

Of Future Institute of Technology, Kolkata in partial fulfillment of requirements for award of the Bachelor of Technology in Computer Science & Engineering of Maulana Abul Kalam Azad University of Technology West Bengal.

Date: ....

Examiner's Signatures:

1. …………………………………………

2. …………………………………………

ACKNOWLEDGEMENTS

The achievement that is associated with the successful Completion of any task would be incomplete without mentioning the name of **Hricha Chakraborty** whose endless cooperation made it possible. We take this opportunity to express our deep gratitude towards our project mentor **Mr. Anindya Mukherjee** for giving such valuable suggestions guidance and encouragement during the development of this project work. Last but not the least we are grateful to all the faculty members of **Engineering Study Centre** for their support.

Dated: ………………...   **Aritra Mondal**

INTRODUCTION

Welcome to Bookworm, your online haven for literary exploration and discovery. In a digital age where screens dominate our lives, Bookworm invites you to rediscover the timeless joy of reading in its purest form: the eBook.

At Bookworm, we believe that every book holds the potential to transport us to new worlds, challenge our perspectives, and inspire our imaginations. That's why we've curated a diverse collection of eBooks spanning genres from classic literature to contemporary fiction, non-fiction, and everything in between. Whether you're seeking thrills, romance, adventure, or knowledge, Bookworm has something to captivate every reader.

But Bookworm is more than just a digital library; it's a community of passionate bibliophiles united by their love for the written word. Join fellow bookworms in spirited discussions, virtual book clubs, and author events, where connections are forged, insights are shared, and friendships are cultivated.

Navigating through our extensive eBook library is effortless with our user-friendly platform, ensuring that your reading experience is seamless and enjoyable. Whether you prefer to cosy up with a tablet at home or read on the go with your smartphone, Bookworm puts your favourite stories at your fingertips whenever and wherever you are.

So come, and embark on a literary journey with us. Let the pages of Bookworm be your guide as you delve into the boundless world of books and let your imagination take flight. Whether you're a seasoned reader or just beginning your literary adventure, Bookworm welcomes you to explore, discover, and get lost in the magic of storytelling.

ABSTRACT

"Bookworm: Nurturing Literary Curiosity in the Digital Age"

In an era characterized by the rapid digitization of information and entertainment, Bookworm emerges as a digital sanctuary, preserving and promoting the timeless art of reading. This abstract delves into the essence of Bookworm, an eBook website designed to ignite and nurture literary curiosity in readers of all ages and backgrounds.

At its core, Bookworm is a platform dedicated to celebrating the written word in its myriad forms. Through a carefully curated selection of eBooks spanning genres, cultures, and periods, Bookworm aims to cater to the diverse tastes and interests of its audience. Whether readers seek to escape into the realms of fantasy, unravel the mysteries of the human condition, or explore the depths of historical narratives, Bookworm offers a rich tapestry of literary experiences waiting to be discovered.

Beyond its extensive eBook library, Bookworm fosters a sense of community among its users. Through interactive features such as discussion forums, virtual book clubs, and author Q&A sessions, readers have the opportunity to engage with one another, share insights, and forge meaningful connections centred around their shared love of literature. This sense of camaraderie serves to enhance the reading experience, transforming solitary acts of consumption into collaborative journeys of exploration and discovery.

With a user-friendly interface and seamless accessibility across digital devices, Bookworm empowers readers to indulge their literary passions anytime, anywhere. Whether curled up on a cosy couch at home or commuting on a bustling train, Bookworm ensures that the magic of storytelling is always within reach.

In essence, Bookworm is more than just an eBook website; it is a gateway to worlds unknown, a beacon of inspiration, and a testament to the enduring power of the written word in an increasingly digital landscape.

HARDWARE AND SOFTWARE REQUIREMENTS

- HARDWARE:-
  - HP 15S LAPTOP
  - PROCESSOR –INTEL I5
  - RAM- 8GM
  - HARDDISK-512 GB

- OPERATING SYSTEM USED: -
  - WINDOWS 11 64 BIT

- TOOLS USED: -
  - VISUAL STUDIO CODE
  - VISUAL STUDIO 2022

- FRAMEWORK USED: -
  - .NET MVC

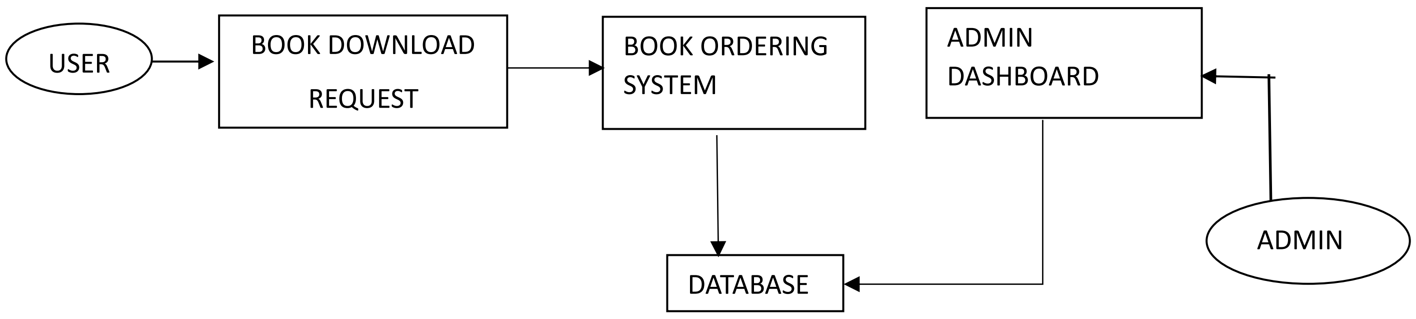
- LANGUAGES USED: -
  - HTML
  - CSS
    BOOTSTRAP
  - JAVASCRIPT

- DATABASE USED: -
  - MICROSFT SQL SERVER
- C# VERSION:-
  - C# 12.0

Requirements Specification: -

System Model –

The structure of the system can be divided into three main logical components. The first component must provide some dropdown form of menu management after logging in, allowing the admin to control what can be ordered by users. The second component is the Book ordering system or the Book Download request which provides the functionality for users to place their order of desired books and supply all necessary details. The third logical component is the order retrieval system. Used by the admin to keep track of all orders that have been placed, this component takes care of retrieving and displaying order information, as well as updating orders that have already been processed.



Functional Requirements –

As can be seen in the system model diagram above, each of the three system components essentially provides a layer of isolation between the end user and the database. The motivation behind this isolation is twofold. Firstly, allowing the end user to interact with the system through a rich interface provides a much more enjoyable user experience, particularly for the non-technical users which will account for the majority of the system's users. In addition, this isolation layer also protects the integrity of the database by preventing users from taking any action outside those that the system is designed to handle. Because of this design pattern, it is essential to enumerate exactly which functions a user will be presented and these functions are outlined below, grouped by component.

The Book Ordering System:-

Users of the Book ordering system must be provided with the following functionalities:

- Create an account.
- Manage their account.
- Log in to the system.
- Navigate the Book's menu.
- Select an item from the menu.
- Customize options for a selected item.
- Add an item to their current order.
- Review their current order.
- Remove an item/remove all items from their current order.
- Provide delivery and payment details.
- Place an order.
- Receive confirmation in the form of an order number.

As the goal of the system is to make the process of placing an order as simple as possible for the customer, the functionality provided through the Book ordering system is restricted to that which is most pertinent to accomplish the desired task. All of the functions outlined above, with the exceptions of account creation and management, will be used every time a user places an order. By not including extraneous functions, I am moving towards my goal of simplifying the ordering process.

Admin Dashboard: -

The Admin Dashboard will be available to the Admin only and will, as the name suggests, allow them to manage the categories that are displayed to users of the book ordering system. The functions afforded by the book management system provide the user with the ability to, use a graphical interface:

- Add a new/update/delete option for a given book.
- Update price for a given book.
- Update default options for a given book.
- Update additional information (description, photo, etc.) for a given book.

An admin dashboard in an online book ordering system serves as the nerve centre for managing various aspects of the platform. It provides administrators with a centralized interface to oversee and control essential functionalities such as inventory management, user accounts, order processing, and analytics. Through the dashboard, administrators can add, update, or remove books from the catalogue, monitor sales and inventory levels in real time, manage user accounts, track orders from placement to delivery, and generate reports to gain insights into customer behaviour and sales trends. Additionally, it may offer tools for customer support, enabling administrators to address inquiries, resolve issues, and communicate with users efficiently. Ultimately, the admin dashboard streamlines operations enhances decision-making, and ensures the smooth functioning of the online book ordering system.

User Interface Specifications -

Each of the system components will have its unique interface. These are described below.

BOOK DOWNLOAD REQUEST

In our online book ordering system, the book download request functionality enables users to access purchased digital books seamlessly. When a user initiates a download request, the system verifies their credentials and validates the availability of the requested book. Upon confirmation, the system facilitates the secure transfer of the digital book file to the user's device, ensuring compatibility and adherence to any digital rights management (DRM) policies in place. Additionally, the system may offer options for users to select preferred formats (e.g., PDF) and provide download progress updates for transparency. By efficiently handling download requests, the system enhances user experience, enabling swift access to desired content while upholding security and copyright standards.

Non-Functional Requirements-

In the development of an online book ordering system using ASP.NET MVC, C#, and SQL Server, several non-functional requirements are crucial to ensure its effectiveness, scalability, and reliability. Firstly, performance requirements dictate the system's ability to handle concurrent user requests efficiently, ensuring swift response times even during peak usage periods. Scalability requirements necessitate the system's capability to accommodate increasing user loads by seamlessly scaling resources such as server capacity and database throughput. Security requirements mandate robust measures for user authentication, data encryption, and protection against common threats like SQL injection and cross-site scripting.

Our web application is cross-compiled using HTML helpers (cshtml) to bind with the models of the database. Along with HTML helpers, there is bootstrap and C# used in the backend all of which are maintained by a reasonably good web server.

All of the application data is stored in a Microsoft SQL Server database, and therefore the Microsoft SQL server must also be installed on the host computer.

The server hardware can be any computer capable of running both the web and database servers and handling the expected traffic. For an online book ordering system, that is not expecting to see much web traffic, or possibly doing only a limited test run, an average personal computer may be appropriate. Once the site starts generating more hits, though, it will likely be necessary to upgrade to a dedicated host to ensure proper performance. The exact cutoffs will need to be determined through a more thorough stress testing of the system.

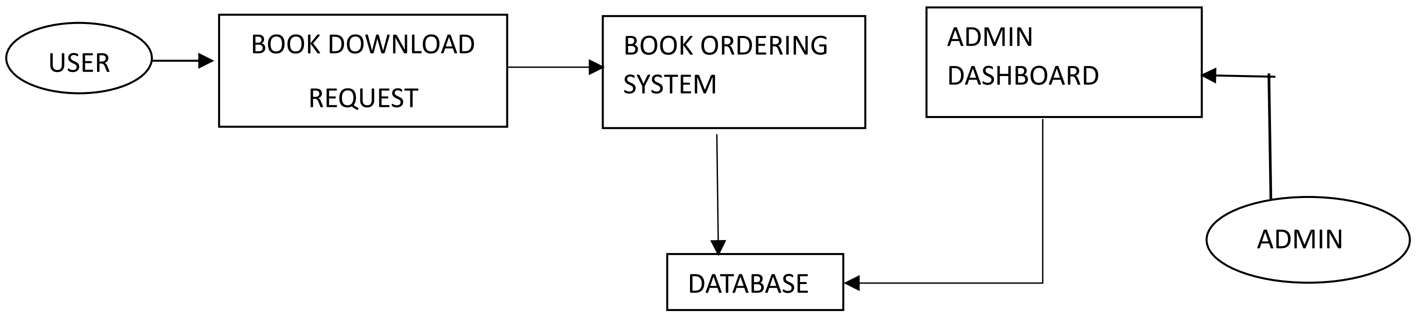
System Evolution-

The system evolution of an online book ordering system developed using ASP.NET MVC, C#, and SQL Server involves iterative enhancements and adaptations to meet changing user needs and technological advancements. Initially, the system may start with basic features such as user registration, book browsing, and simple ordering capabilities. As the system evolves, additional functionalities like personalized recommendations, advanced search filters, and integration with external payment gateways are introduced to enhance user experience and streamline the ordering process. Moreover, continuous optimization for performance, scalability, and security is vital to accommodate growing user traffic and address emerging threats. With each iteration, the system undergoes rigorous testing and refinement to ensure reliability and usability. Additionally, advancements in technologies and industry standards prompt the integration of new tools and methodologies, such as cloud computing and microservices architecture, to enhance flexibility and maintainability. Overall, the system evolution of the online book ordering system is a dynamic process driven by user feedback, technological innovation, and business requirements, aimed at delivering an efficient and feature-rich platform for book enthusiasts.

SYSTEM DESIGN-

Level 1: The Database and the 3 components

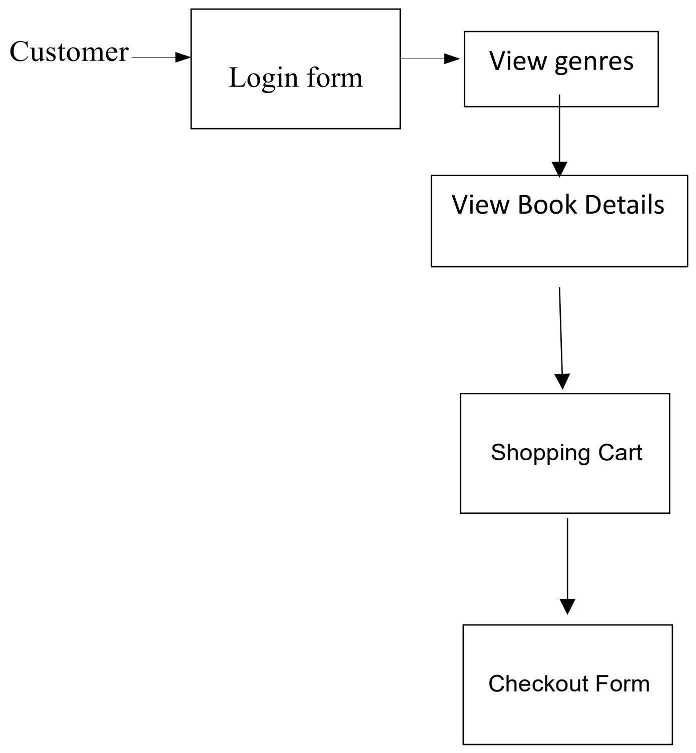
The structure of the system can be divided into three main logical components. The first component must provide some dropdown form of menu management after logging in, allowing the admin to control what can be ordered by users. The second component is the Book ordering system or the Book Download request which provides the functionality for users to place their order of desired books and supply all necessary details. The third logical component is the order retrieval system used by the admin to keep track of all orders that have been placed. This component takes care of retrieving and displaying order information, as well as updating orders that have already been processed.



Level 2: Book Ordering System Components

To order a book from the online book ordering system, begin by logging into your account. Once logged in, navigate to the genres section to explore the available categories. Choose the genre that matches the book you're looking for, then browse through the selection until you find the specific book you want to order. Click on the book to view its details, including

the author, description, and price. If satisfied, locate the option to add the book to your cart and click on it. Ensure that the book is successfully added to your cart before proceeding to checkout. At the checkout page, review your order to confirm the book and its quantity, then proceed to complete the purchase by following the prompts for payment and shipping information. Once the transaction is complete, you'll receive confirmation of your order, and the book will be on its way to you.



Level 3: The Login Form

The login form is standard for a form of this type. It provides text fields for username and password, which the user must enter before signing in. This form also gives the option for a user to register for the site if they have not yet done so.

Level 3: View Genres

The Genres appear on the index page as soon as the user logs in to his/her account. The user can select any genre of his/her choice and then view the books present in that particular genre.

Level 3: View Book Details

After selecting a particular genre, a user, then sees the details of a particular book according to his/her liking.

Level 3: Shopping Cart

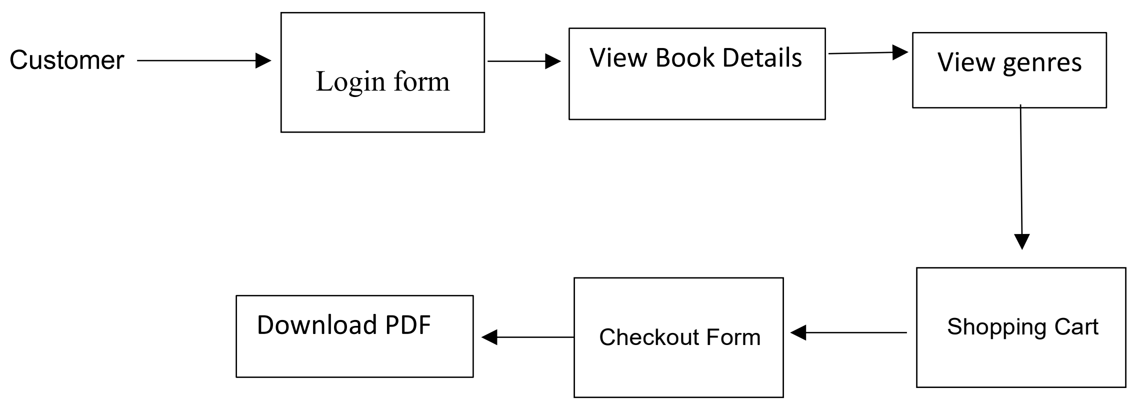
After seeing the details of a book, if the user likes it, he/she may add that book to their cart. The shopping cart performs much like a shopping cart in any other application. After an item is added to the order, it is displayed, along with its price, in the shopping cart. The shopping cart also keeps a running total of the current price of the whole order. By clicking on an item in the shopping cart, the user can review all of the details for that particular item. Finally, the shopping cart contains a button for the user to proceed to checkout.

Level 3: Checkout Form

The checkout form is the user's last chance to verify that the contents of their order are correct before actually placing it. This form also provides fields for the user to supply all of the necessary checkout and delivery details (payment type, delivery address, etc.).

Level 2: Book Download Request Components

To download a PDF of a purchased book from the online book ordering system, first, log in to your account. Once logged in, navigate to the genres section to find the category of the book you've purchased. After selecting the genre, browse through the available books and click on the one you've purchased to view its details. Within the book details page, locate the option to add the book to the cart. Click on it to add the book to your shopping cart. Once the PDF is downloaded, proceed to your cart and ensure the book is added. If not, you can add it from the book details page. Finally, proceed to checkout, where you may be prompted to confirm your purchase and complete the transaction. Once done, you should have successfully downloaded the PDF of the book you purchased.



Level 3: The Login Form

The login form is standard for a form of this type. It provides text fields for username and password, which the user must enter before signing in. This form also gives the option for a user to register for the site if they have not yet done so.

Level 3: View Genres

The Genres appear on the index page as soon as the user logs in to his/her account. The user can select any genre of his/her choice and then view the books present in that particular genre.

Level 3: View Book Details

After selecting a particular genre, a user, then sees the details of a particular book according to his/her liking.

Level 3: Shopping Cart

After seeing the details of a book, if the user likes it, he/she may add that book to their cart. The shopping cart performs much like a shopping cart in any other application. After an item is added to the order, it is displayed, along with its price, in the shopping cart. The shopping cart also keeps a running total of the current price of the whole order. By clicking on an item in the shopping cart, the user can review all of the details for that particular item. Finally, the shopping cart contains a button for the user to proceed to checkout.

Level 3: Checkout Form

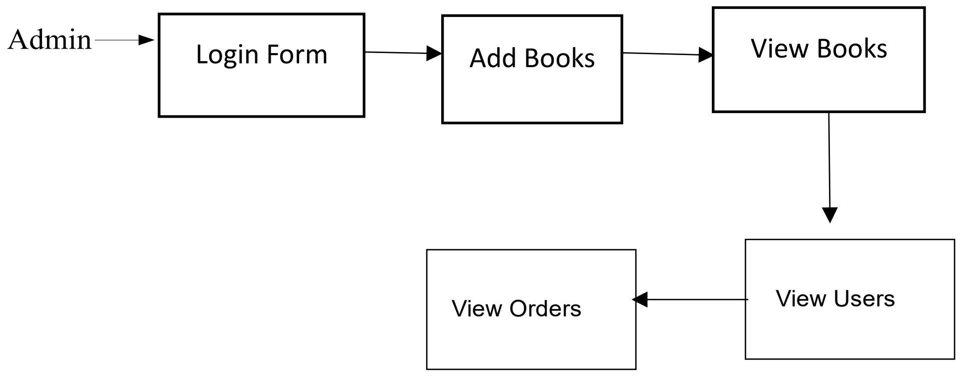
The checkout form is the user's last chance to verify that the contents of their order are correct before actually placing it. This form also provides fields for the user to supply all of the necessary checkout and delivery details (payment type, delivery address, etc.).

Level 3: Download PDF

After successfully checking out, the user can now download the pdf of the book the user has purchased.

Level 2: Admin Dashboard Components

As an admin on the online book ordering system, begin by logging into your account using your credentials. Once logged in, navigate to the admin dashboard, where you'll find options to manage books. To add a new book, select the "Add Book" option and fill in the required details such as title, author, genre, description, price, and available stock. After adding the book, you can view the list of existing books by selecting the "View Books" option. Here, you can see details of all the books in the system, including their current stock levels. If needed, update the stock of any book by selecting the book and adjusting the stock quantity accordingly. Additionally, you can view user information by selecting the "View Users" option, where you can see details about registered users, their activity, and any pertinent information related to their accounts. This comprehensive control allows you to manage books, stocks, and users effectively within the online book ordering system.



Level 3: Login Form

The login form is standard for a form of this type. It provides text fields for username and password, which the admin must enter before signing in. There is no signup page for the admin as the admin will be provided with his/her credentials beforehand.

Level 3: Add Books

After logging in, the admin can add books to a particular genre by adding the necessary details including the price, available stock, etc.

Level 3: View Books

The admin can also view all the books from all the genres present, and check if any updation is required.

Level 3: View Users

The admin can also check the number of users currently logged in.

Level 3: View Orders

The admin can also check the number of orders made.

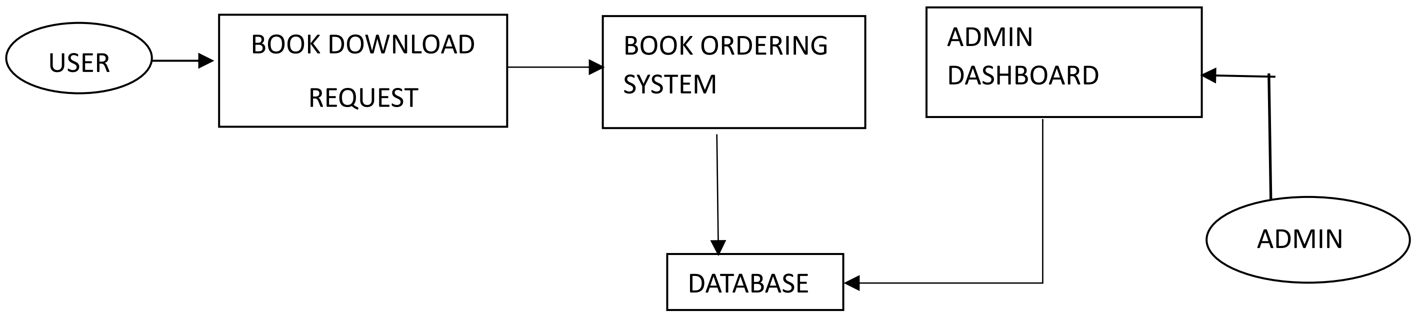
User Interface Design

The user interface design principles can be broken into two groups. The interface in the web application is designed to limit free-form user input, using mostly drop-down menus, radio buttons and checkboxes. This is done for two reasons – to simplify the ordering process as much as possible, and to limit SQL injection attempts. Free-form input is necessary for the book ordering system component, however, as all of the values must be user-supplied. The interface for this component contains traditional forms comprised of text fields and corresponding labels, along with save and discard buttons for each form.

Testing Design –

Testing

Phases - The structure of the system can be divided into three main logical components, plus the database, which is invisible to the end user. Each of these components must be tested individually, and the approaches that will be used for each component are described in the following sections.



Database-

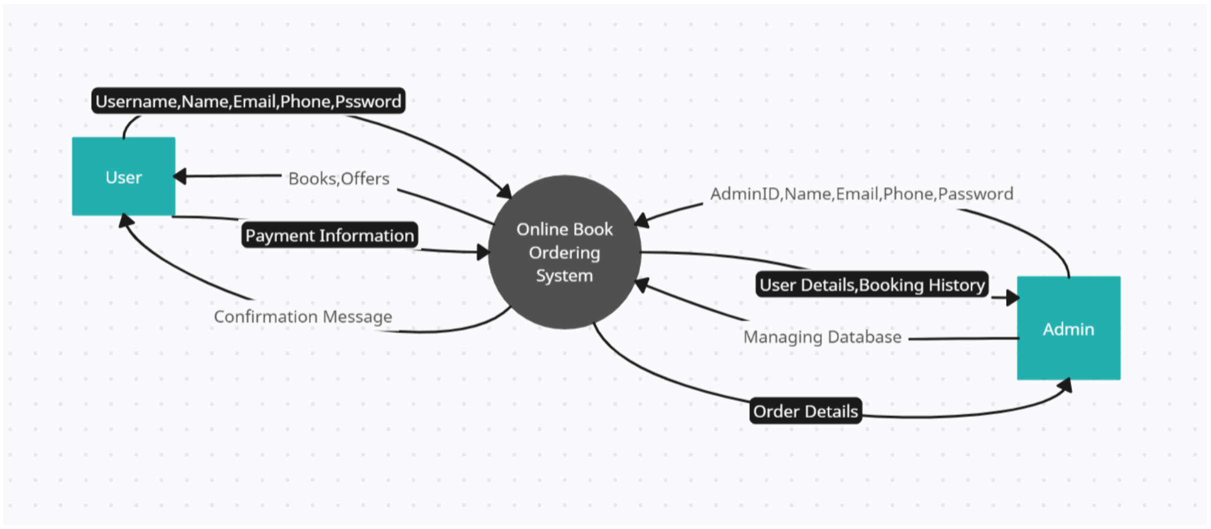
Testing of the database component is very straightforward and has already been mostly completed. The database was the first component designed and before beginning work on any of the applications, all of the SQL statements which are expected to be needed have been written and executed directly, essentially isolating the database. By doing this, we were able to reveal, and promptly fix a large percentage of the errors within the database itself.

DFD (Data Flow Diagram): The Book Ordering System is a type of software that allows the providers to manage and accept the books and placed orders over the website. Let us understand the workings of the book ordering system by using DFD.

Here, different levels of DFD are shown for Book Ordering System such as Level 0 DFD, Level 1 DFD, Level 2 DFD, and Level 3 DFD.

LEVEL 0 DFD: –

At this level, the Input and Output of the system are shown. The system is designed and established across the world with input and output at this level.


Username,Name,Email,Phone,Pssword
User
Books,Offers
Payment Information
Confirmation Message
Online Book Ordering System
AdminID,Name,Email,Phone,Password
User Details,Booking History
Managing Database
Admin
Order Details

The Online Book Ordering System has the following Input :

- User Details
- User's Payment Information
- Admin Details
- Managing Database

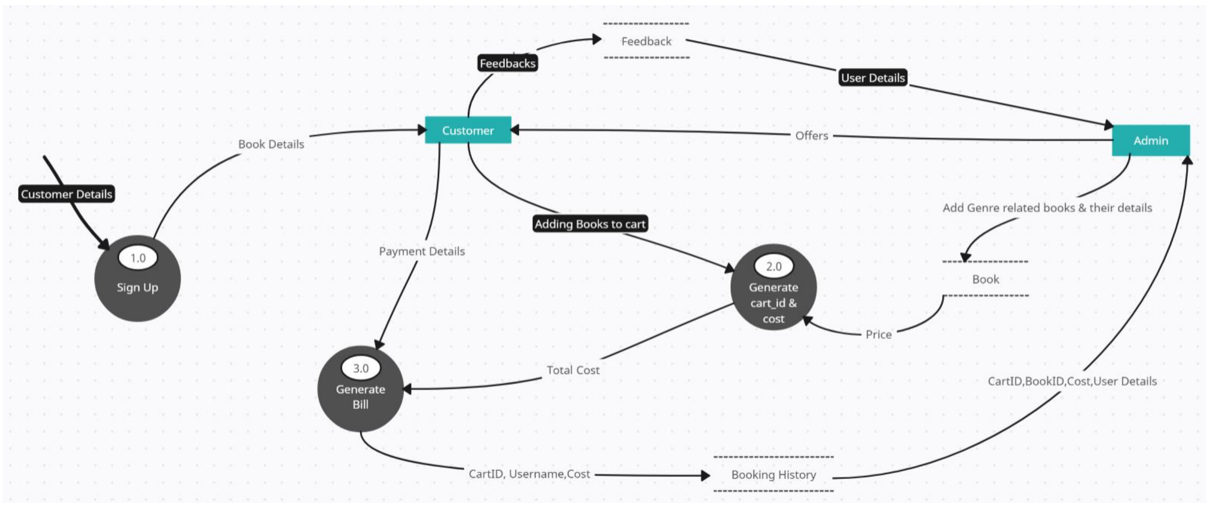
The Online Book Ordering System has the following Output:

- Books and Offers
- User Details and Booking History to Admin
- Confirmation message to User
- Order Details

LEVEL 1 DFD: -

At this level, we will see the different essential processes in the main system and how the data flows through them.

The first process (1.0) is where the customer will sign up, the second process (2.0) is where the Cart_id and cost are generated and the third process (3.0) is where the bill is generated.



1. **Sign-Up Process** (Process 1.0):

   - This process represents the user registration or sign-up phase.

   - Input: User information (e.g., name, email, password).

   - Output: View of the books

2. **Generate Cart ID and Cost** (Process 2.0):

   - This process is responsible for creating a unique Cart ID for each user and calculating the total cost of items added to the cart.

   - Input: User actions (e.g., adding items to the cart, quantity of items).

   - Output: Cart ID assigned to the user's session and total cost of items in the cart.

3. **Generate Bill** (Process 3.0):

   - This process generates a bill for the user based on the items in their cart and their respective costs.

   - Input: Cart ID, Payment details, and total cost.

   - Output: Cart ID along with a few user details and the total cost is updated in the database.

4. **Database** (Data Store):

   - This represents the storage of book information according to the different genres, cart details (including cart ID and items), feedback from users and billing information.

   - Data Store: Book database, Feedback, Booking History.

5. **Customer Interface** (External Entity):

- Represents the customer interacting with the system, such as signing up, viewing the books, adding items to the cart, giving feedback, and giving payment details.

- External Entity: Customer

6. **Admin** (External Entity):

- Represents the Admin managing the database and processing data flows.

- External Entity: Admin.

Here's how the data flows in this system:

- User signs up, providing registration data to the system.

- Upon successful registration, the system generates a view for the user.

- The user interacts with the system to add items to the cart, which updates the cart ID and cost calculations.

- When the user requests a bill, the system uses the cart ID to fetch the relevant item details and prices, then generates and presents the bill to the user.

This DFD outlines the flow of data and processes involved in managing user registration, shopping cart operations, and billing within a system.

LEVEL 2 DFD:-

At this level, we elaborate on the Generate cart_id and Cost process (2.0), to see the different processes that take place here.

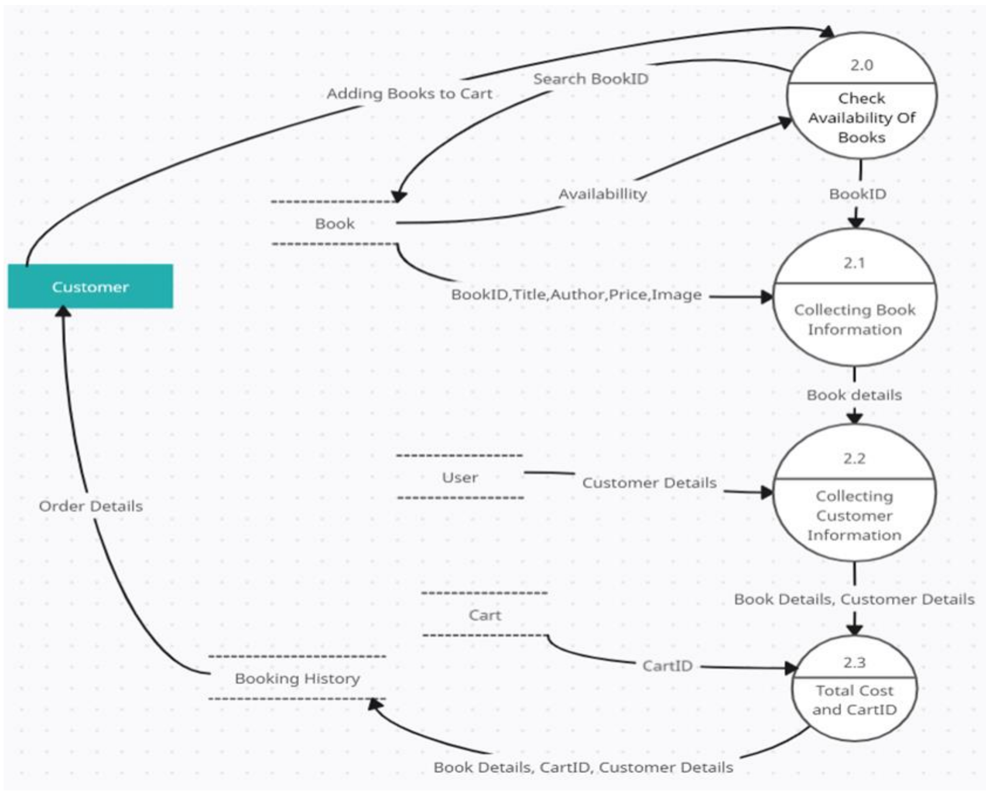
The sub-processes are checking book availability (2.0), collecting book information(2.1), collecting customer information(2. 2) and finally total cost and Cart ID(2.3)

1. **Generate Cart ID and Cost (Process 2):**

   - **Sub-Process 2.0: Check Availability of Book**
     - This sub-process checks the availability of a book in the book database before adding it to the cart.
     - So it searches the BookID in the database, takes the status as input and provides the BookID to the next process.
   - **Sub-Process 2.1: Collect Book Information**
     - If the book is available, this sub-process collects information about the book such as title, author, price, etc.
     - Input: Book ID.

    - Output: Book details (title, author, price).

- **Sub-Process 2.2: Collect Customer Information**
    - This sub-process collects customer information from the user database based on the user's login.
    - Input: Customer Details.
    - Output: Customer information to the final process of generating total cost and Cart ID.
- **Sub-Process 2.3: Generate Total Cost**
    - Once the book and customer information are gathered, this sub-process calculates the total cost of the items in the cart and generates the Cart ID.
    - Input: Book details, quantity, customer information.
    - Output: Total cost of items in the cart and Cart ID, updated Booking History.
- **Data Stores:**
    - Book Database: Stores book information including availability status, title, author, price, etc.
    - User Database: Contains customer information such as name, contact details, etc.
    - Cart Database: Stores cart-related data including cart ID, items in the cart, total cost, etc.
    - Booking History: Keeps a record of transactions including cart IDs, customer information, and total cost.
- **External Entities:**
    - Customer: Interacts with the system to add books to the cart and make purchases.



2.0
Check Availability Of Books
Search BookID
Adding Books to Cart
Availabillity
BookID
Book
2.1
Collecting Book Information
BookID,Title,Author,Price,Image
Customer
Book details
2.2
Collecting Customer Information
User
Customer Details
Order Details
Book Details, Customer Details
Cart
2.3
Total Cost and CartID
CartID
Booking History
Book Details, CartID, Customer Details

In the Level 2 Data Flow Diagram (DFD) for Process 2, data flows follow a clear sequence. First, user input (Book ID or details) is sent to check the book's availability in the Book database, which returns an availability status. This status then triggers the collection of book information (title, author, price) from the Book Database. Simultaneously, user information (Username, name, contact details) is retrieved from the User Database. These data sets, along with quantity information, converge to generate the total cost of the items in the cart. Once the total cost is calculated, it is displayed to the user. Additionally, the system fetches the Cart ID from the Cart Database and updates the Booking History with transaction details, completing the data flow loop.

ER Diagram for an e-Book Website

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. The entity is represented by a rectangle shape. The entity will be our database table of the E-book website later on. The oval shape represents an attribute.

We will follow the 3 basic rules in creating the ER Diagram.

1. Identify all the entities.

2. Identify the relationship between entities and

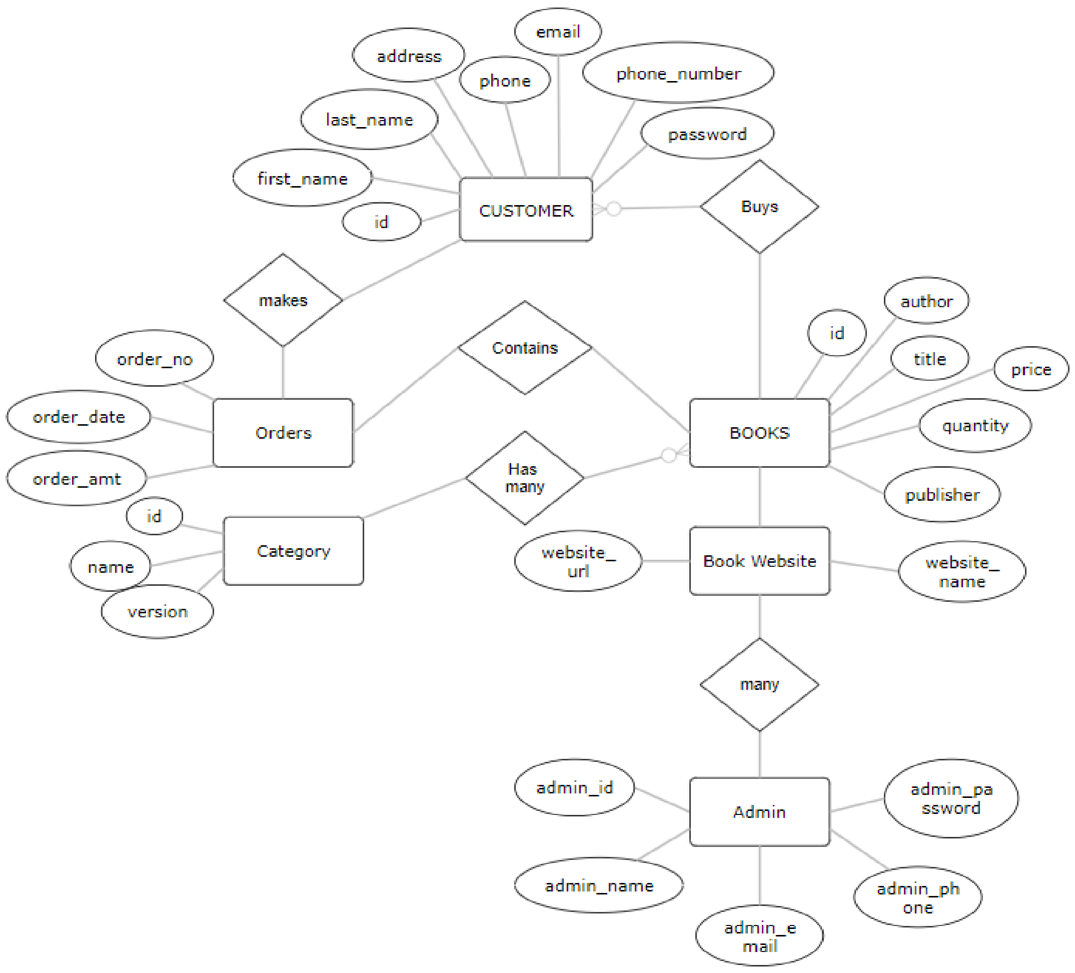
3. Add meaningful attributes to our entities.

On the E-book website, we have the following entities

- Admin
- User
- Customer
- Order
- Books
- Category
- Book website

Our design of the E-book website system consists of 6 entities; the specified entities will be our database tables in the design and implementation of our E-book website. We will now draw the entities of the E-book website named Bookworm specified above and a rectangle shape will represent it.

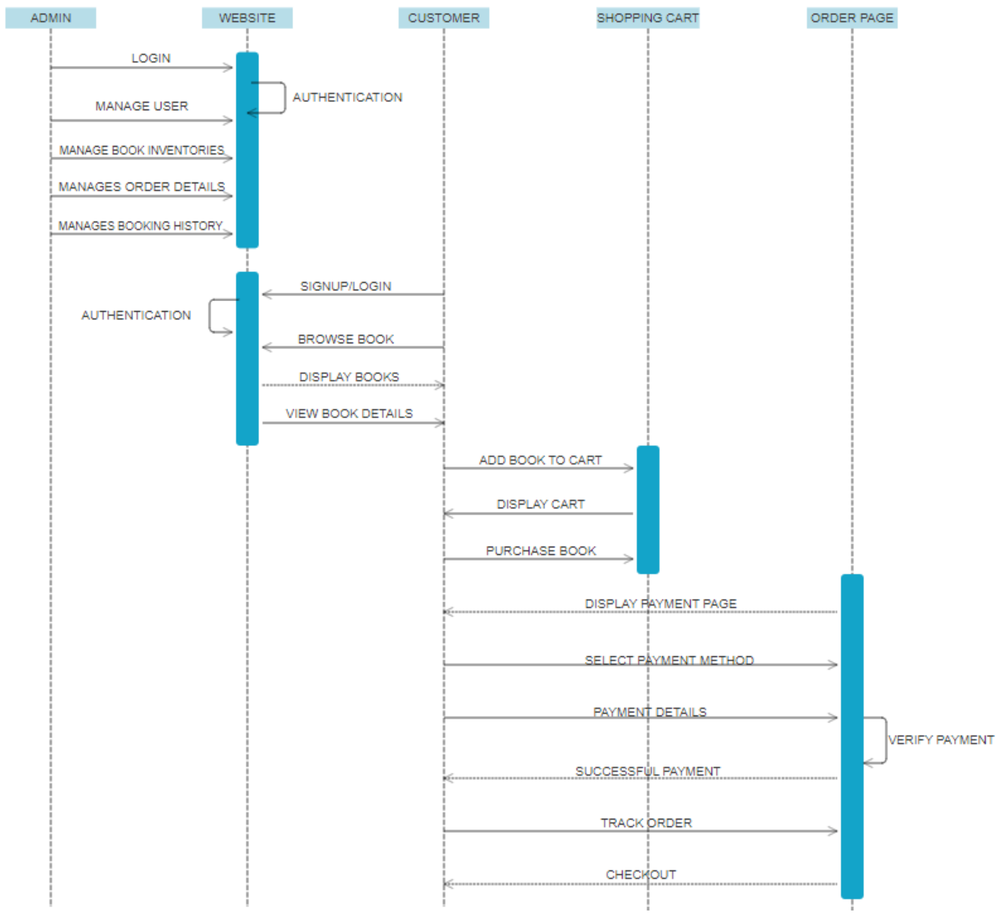
The image given below is the ER Diagram of the e-book website:



Sequence Diagram for E-Book Website

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. This is the UML sequence diagram of the E-book Website which shows the interaction between the objects of Book, Customer, Admin, Order and Cart.

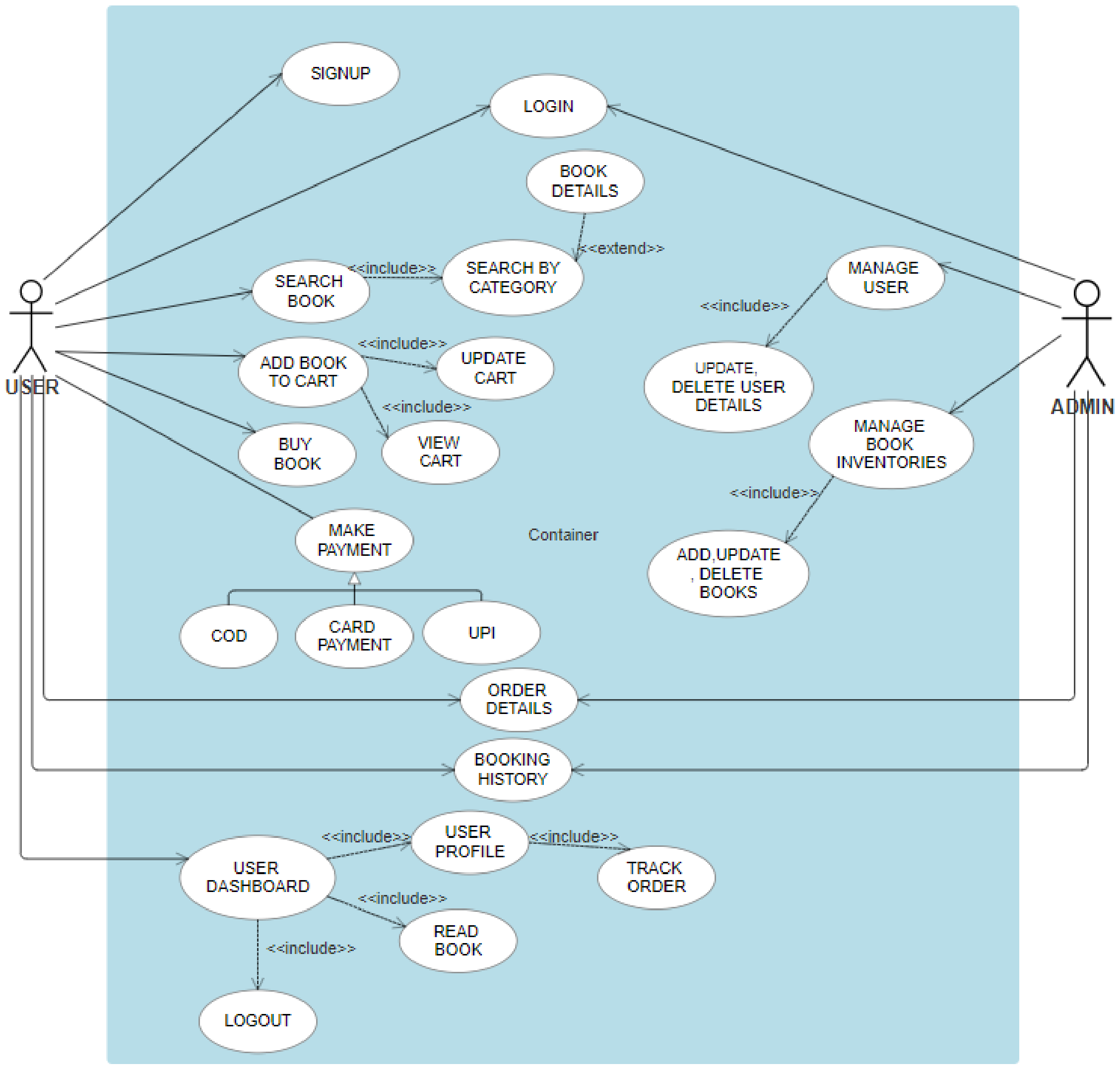
The image given below is the Sequence Diagram of the E-book website:



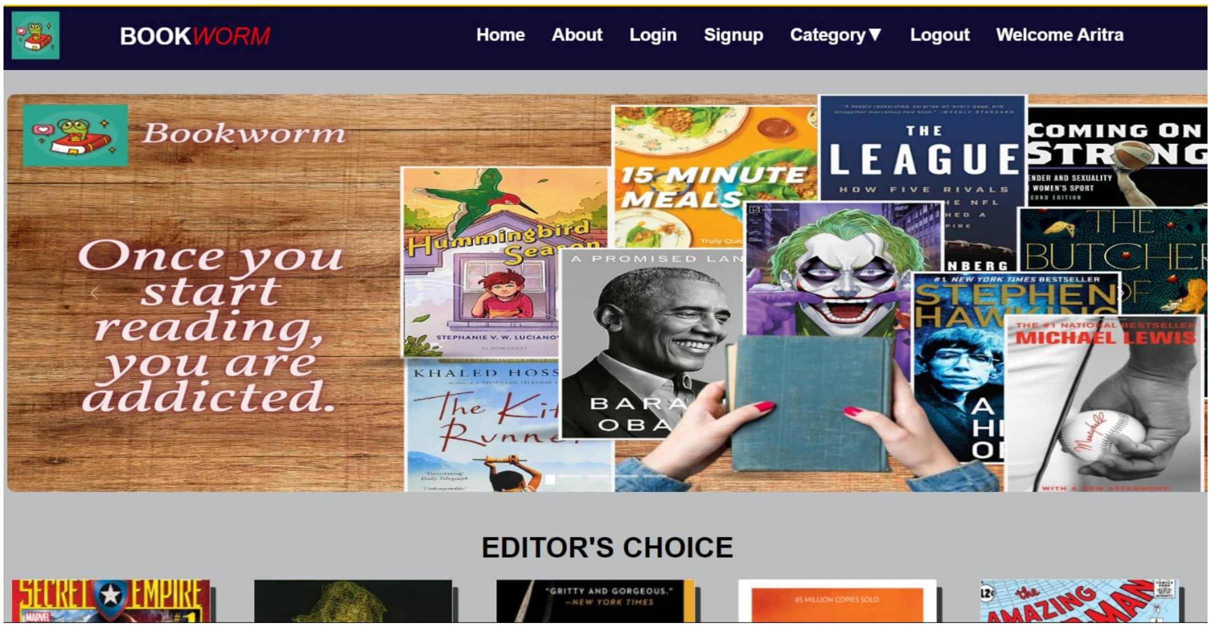
Use-Case Diagram for E-Book website:

This Use Case Diagram is a graphic depiction of the interactions among the elements of an e-book website. It represents the methodology used in system analysis to identify, clarify, and organise system requirements. In this Use Case Diagram, the main actors of an e-book website are the Admin and the User i.e., the Customer. The admin here manages the Customer details i.e., add, delete and update user accounts, book inventories i.e., add, delete and update the book, order details, and booking details. Whereas the User will register on the website. They can search, buy and read books.

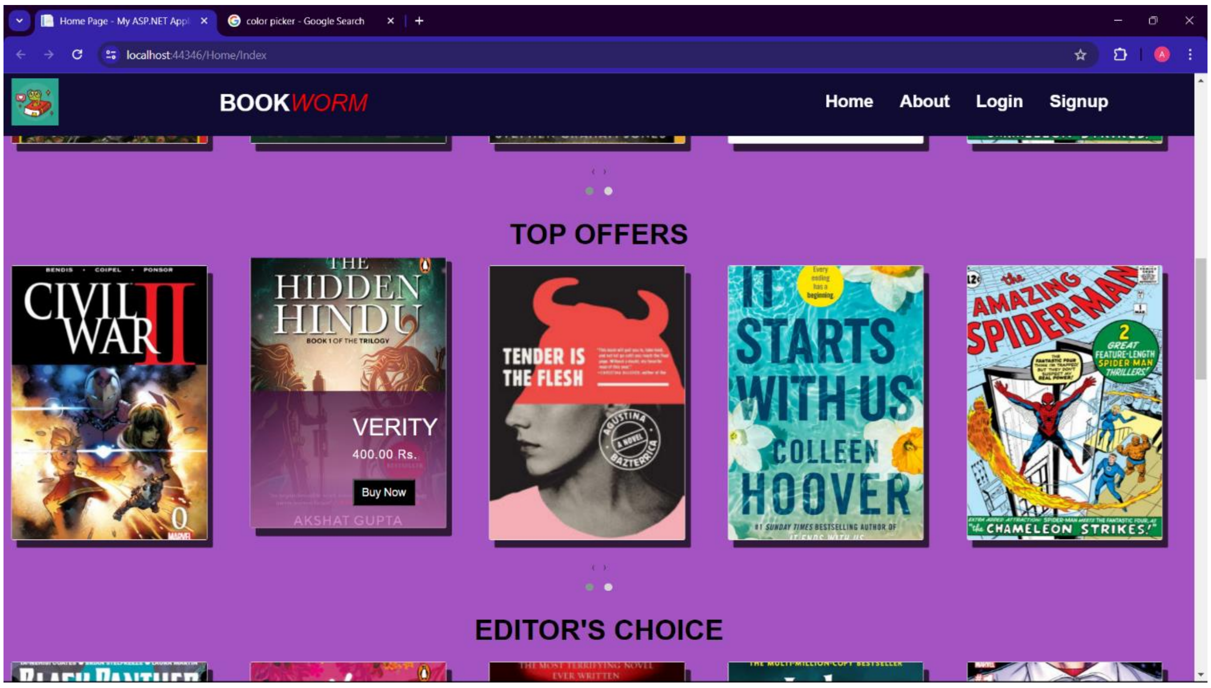
The major elements of the UML use case diagram of the e-book website are shown in the picture below.



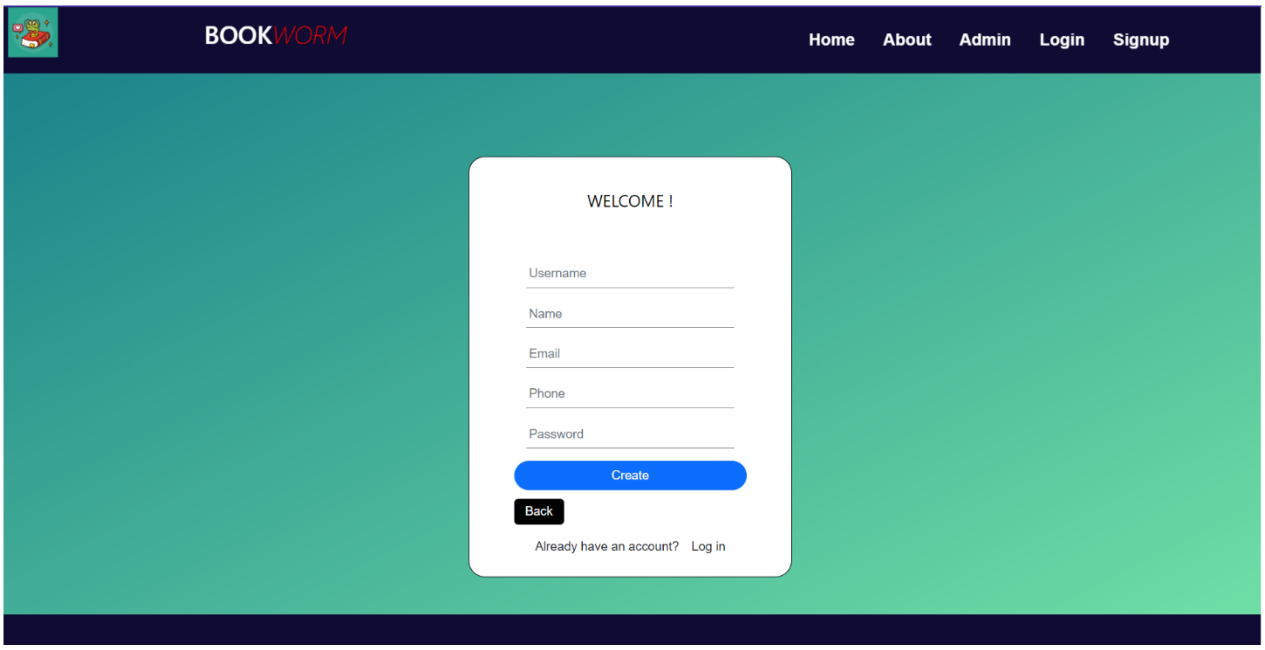
SCREENSHOTS OF PROJECT VIEWS:



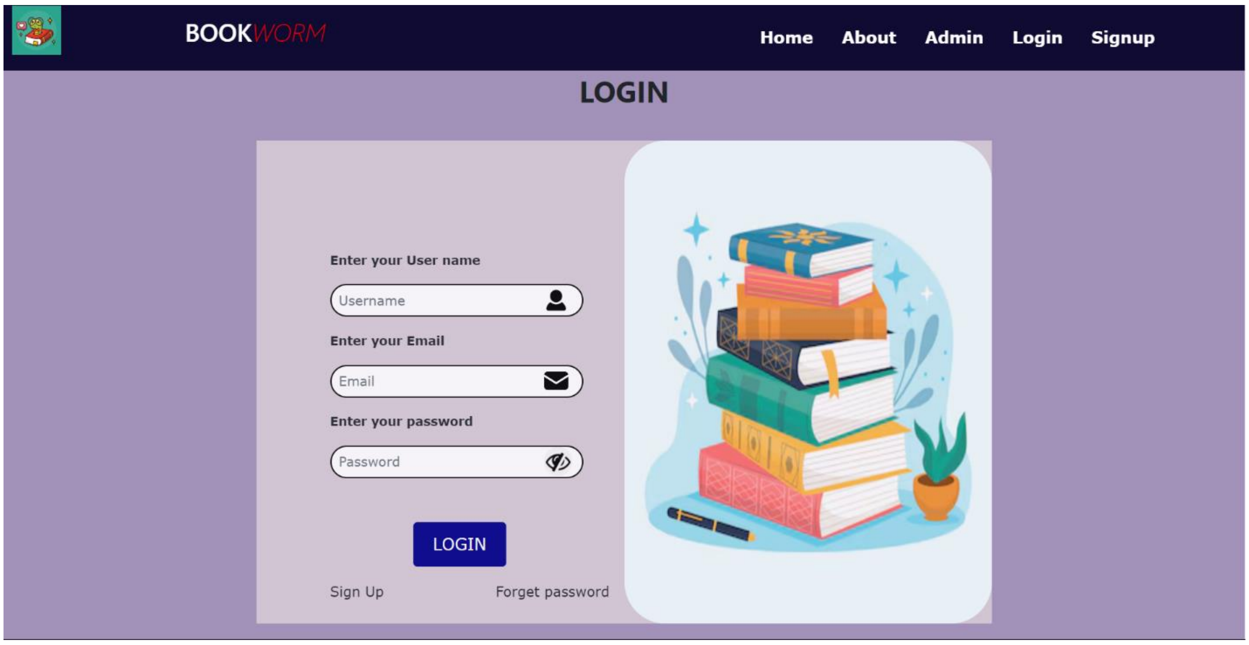
HOME PAGE


Home Page - My ASP.NET Appl
color picker - Google Search
localhost:44346/Home/Index
BOOKWORM
Home
About
Login
Signup
TOP OFFERS
VERITY
400.00 Rs.
Buy Now
EDITOR'S CHOICE

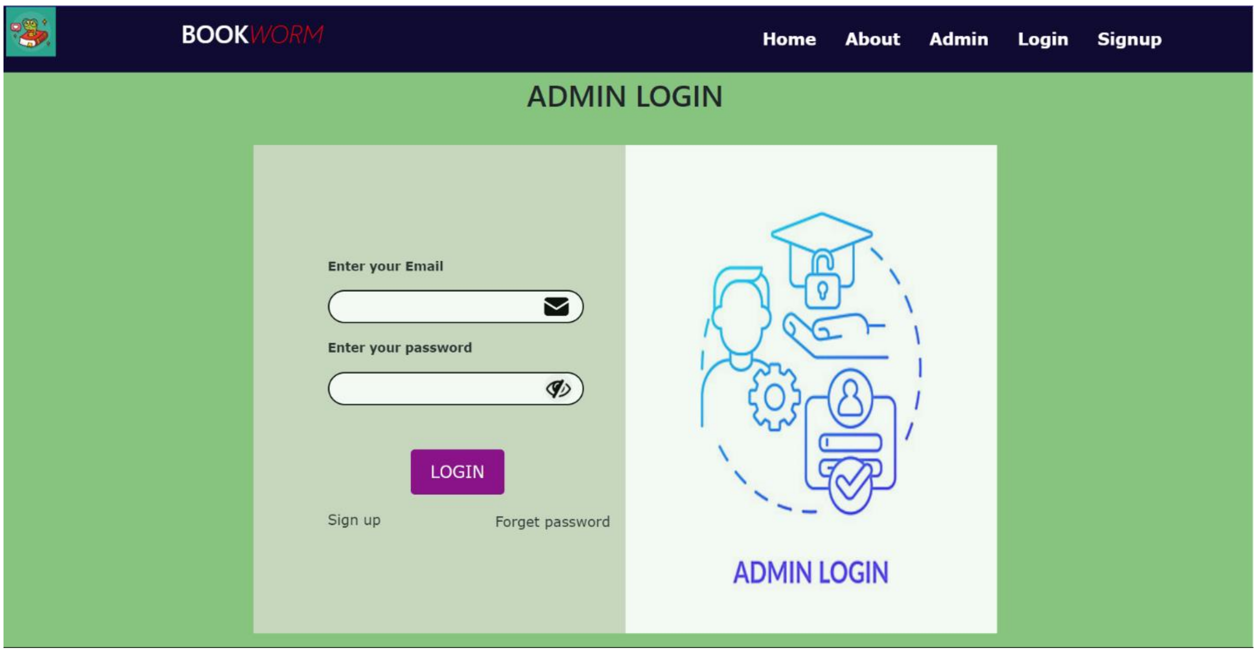
BOOKS ON OFFER



USER SIGNUP PAGE



USER LOGIN PAGE


BOOKWORM Home About Admin Login Signup

ADMIN LOGIN

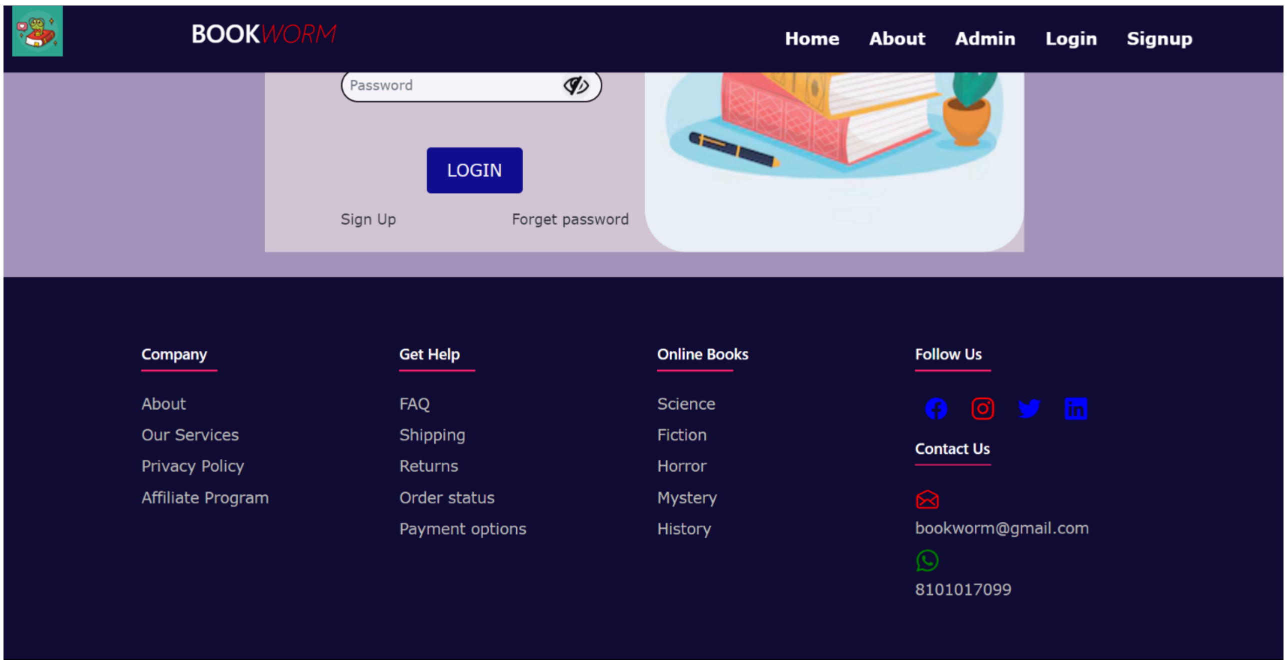
Enter your Email

Enter your password

LOGIN

Sign up Forget password

ADMIN LOGIN PAGE


BOOKWORM Home About Admin Login Signup

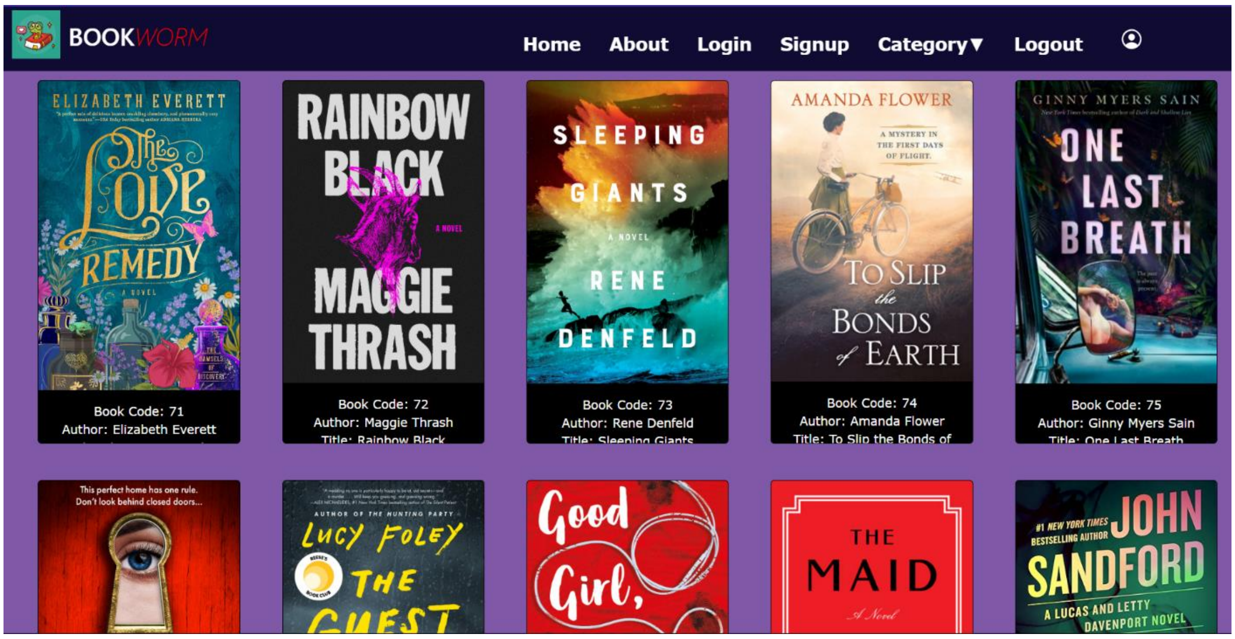
Password

LOGIN

Sign Up Forget password

Company	Get Help	Online Books	Follow Us
About	FAQ	Science	
Our Services	Shipping	Fiction	Contact Us
Privacy Policy	Returns	Horror	bookworm@gmail.com
Affiliate Program	Order status	Mystery	8101017099
	Payment options	History	

FOOTER PART



Book Code: 71
Author: Elizabeth Everett

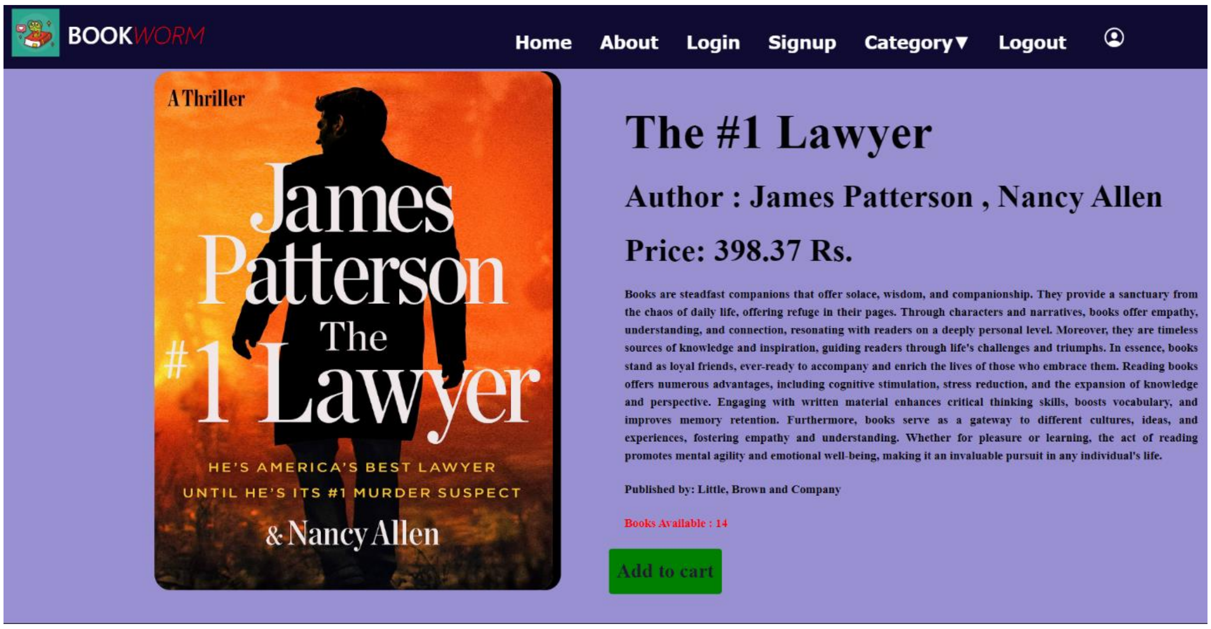
Book Code: 72
Author: Maggie Thrash
Title: Rainbow Black

Book Code: 73
Author: Rene Denfeld
Title: Sleeping Giants

Book Code: 74
Author: Amanda Flower
Title: To Slip the Bonds of

Book Code: 75
Author: Ginny Myers Sain
Title: One Last Breath

BOOKLIST FOR USERS



The #1 Lawyer

Author : James Patterson , Nancy Allen

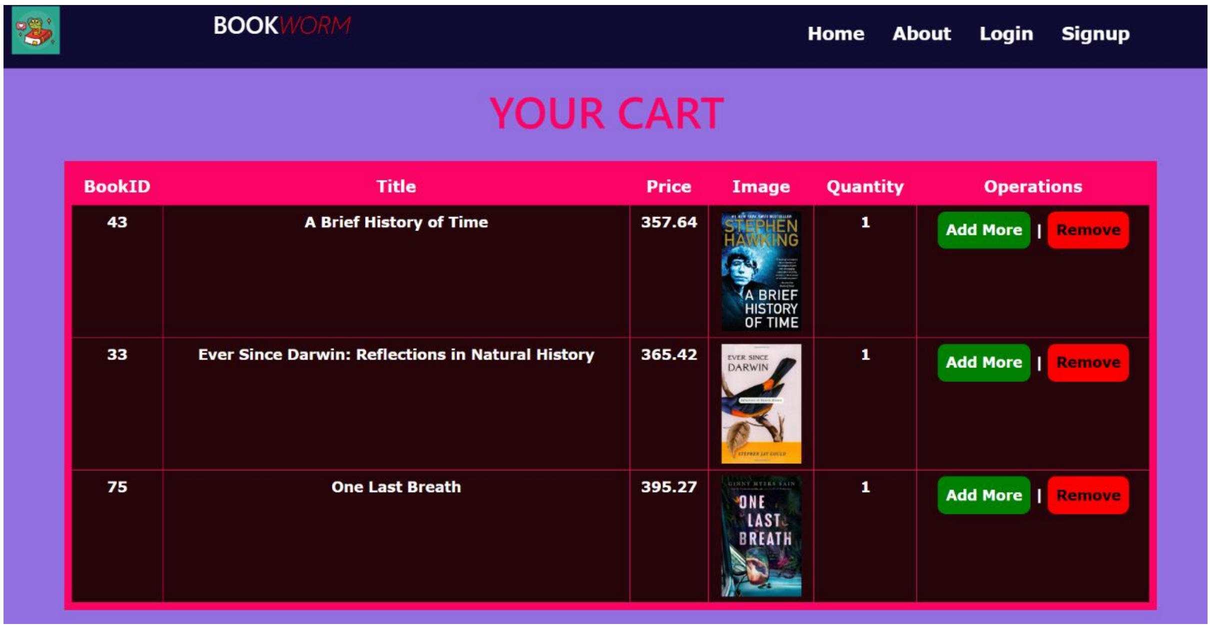
Price: 398.37 Rs.

Books are steadfast companions that offer solace, wisdom, and companionship. They provide a sanctuary from the chaos of daily life, offering refuge in their pages. Through characters and narratives, books offer empathy, understanding, and connection, resonating with readers on a deeply personal level. Moreover, they are timeless sources of knowledge and inspiration, guiding readers through life's challenges and triumphs. In essence, books stand as loyal friends, ever-ready to accompany and enrich the lives of those who embrace them. Reading books offers numerous advantages, including cognitive stimulation, stress reduction, and the expansion of knowledge and perspective. Engaging with written material enhances critical thinking skills, boosts vocabulary, and improves memory retention. Furthermore, books serve as a gateway to different cultures, ideas, and experiences, fostering empathy and understanding. Whether for pleasure or learning, the act of reading promotes mental agility and emotional well-being, making it an invaluable pursuit in any individual's life.

Published by: Little, Brown and Company

Books Available : 14

BOOK DETAILS PAGE

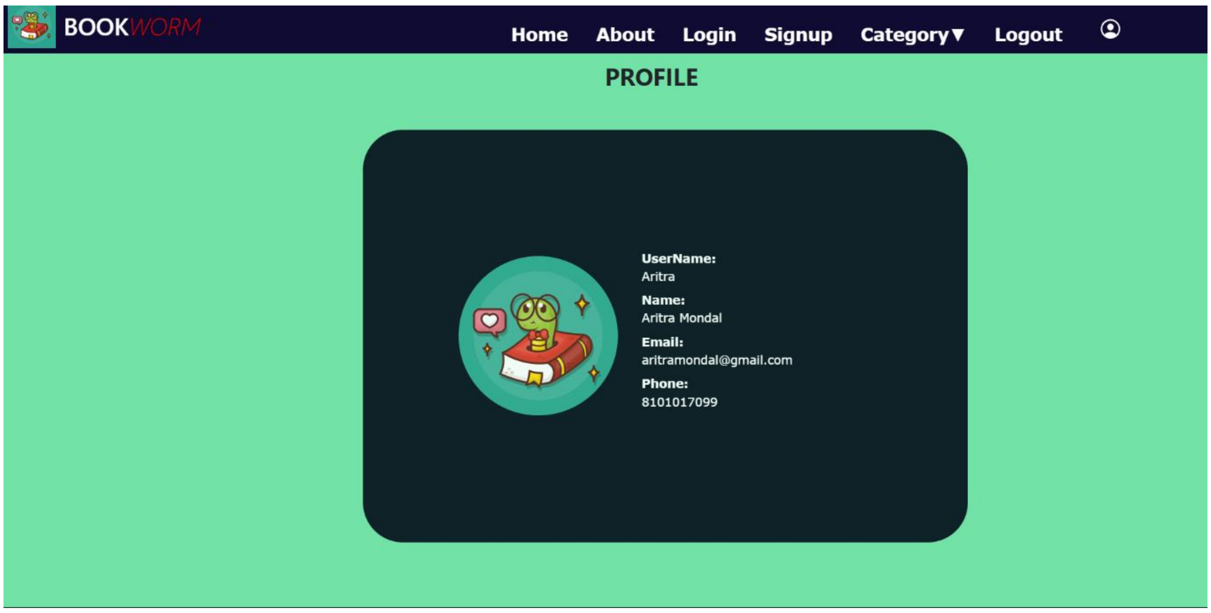

BOOKWORM

Home About Login Signup

YOUR CART

BookID	Title	Price	Image	Quantity	Operations
43	A Brief History of Time	357.64		1	Add More | Remove
33	Ever Since Darwin: Reflections in Natural History	365.42		1	Add More | Remove
75	One Last Breath	395.27		1	Add More | Remove

CART SECTION


BOOKWORM

Home About Login Signup Category▼ Logout

PROFILE

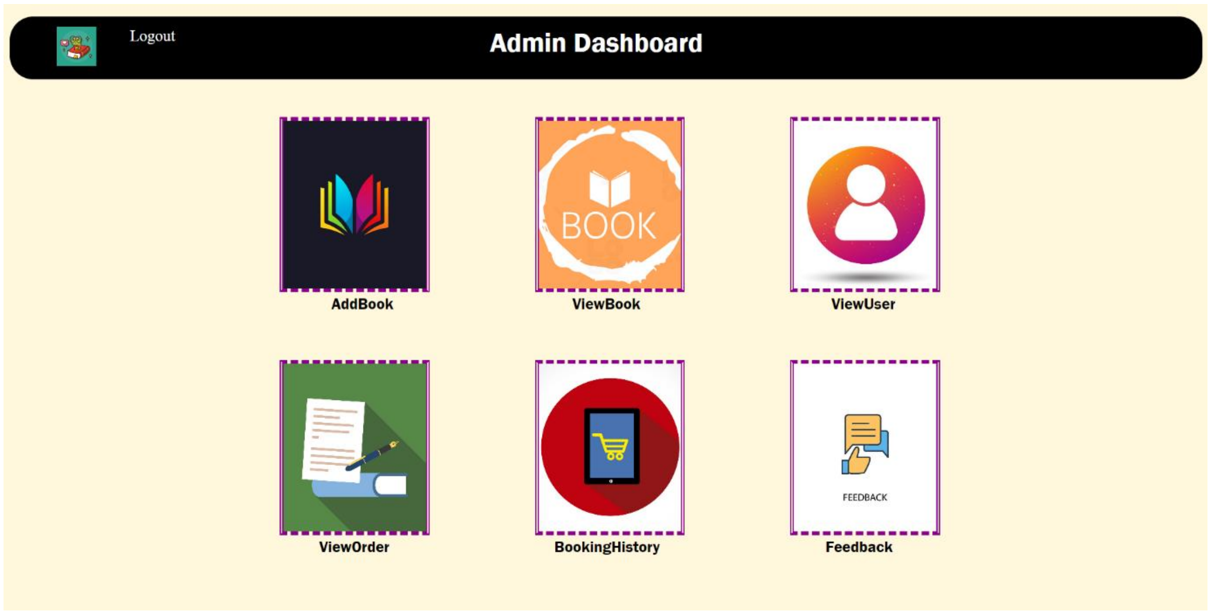
UserName:
Aritra

Name:
Aritra Mondal

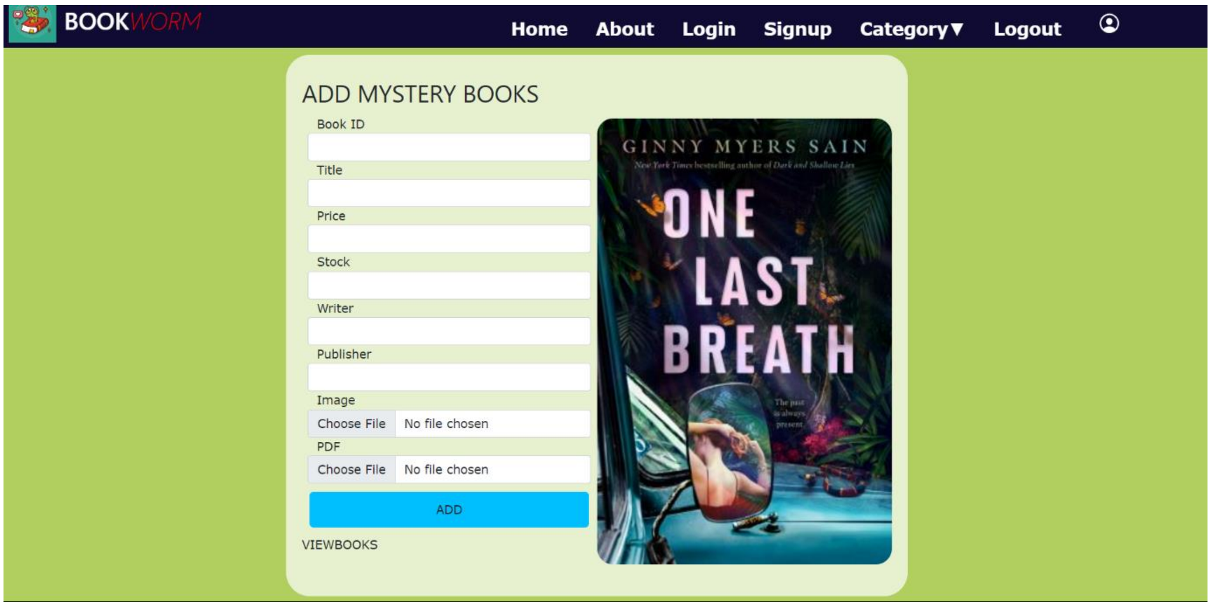
Email:
aritramondal@gmail.com

Phone:
8101017099

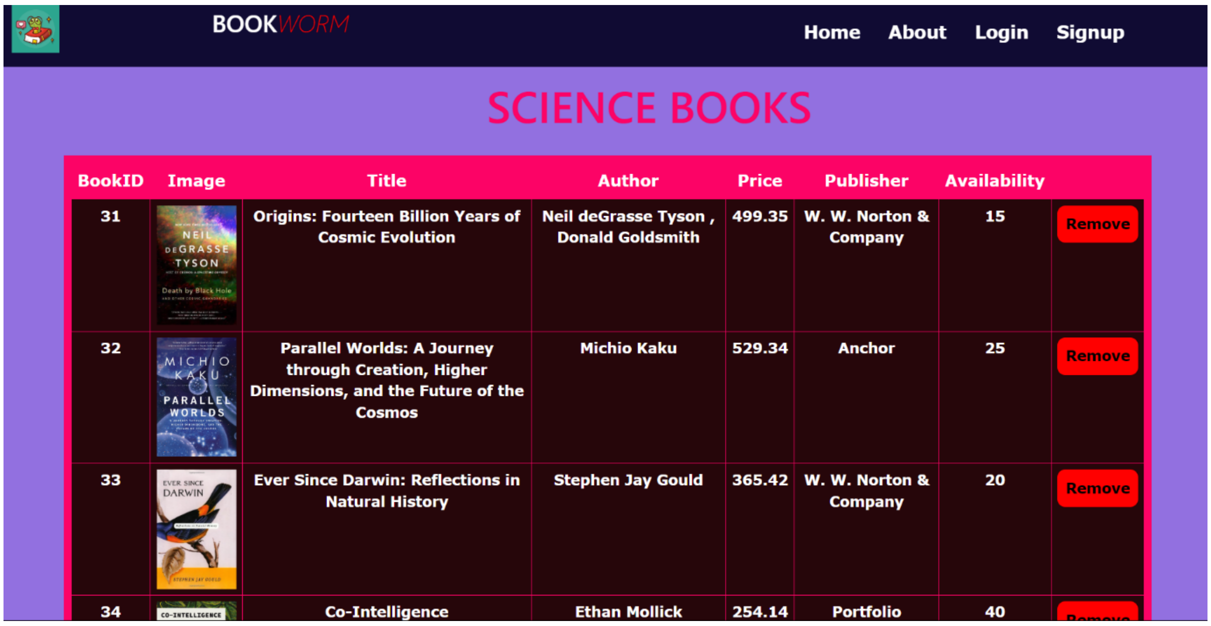
USER PROFILE DETAILS



ADMIN DASHBOARD



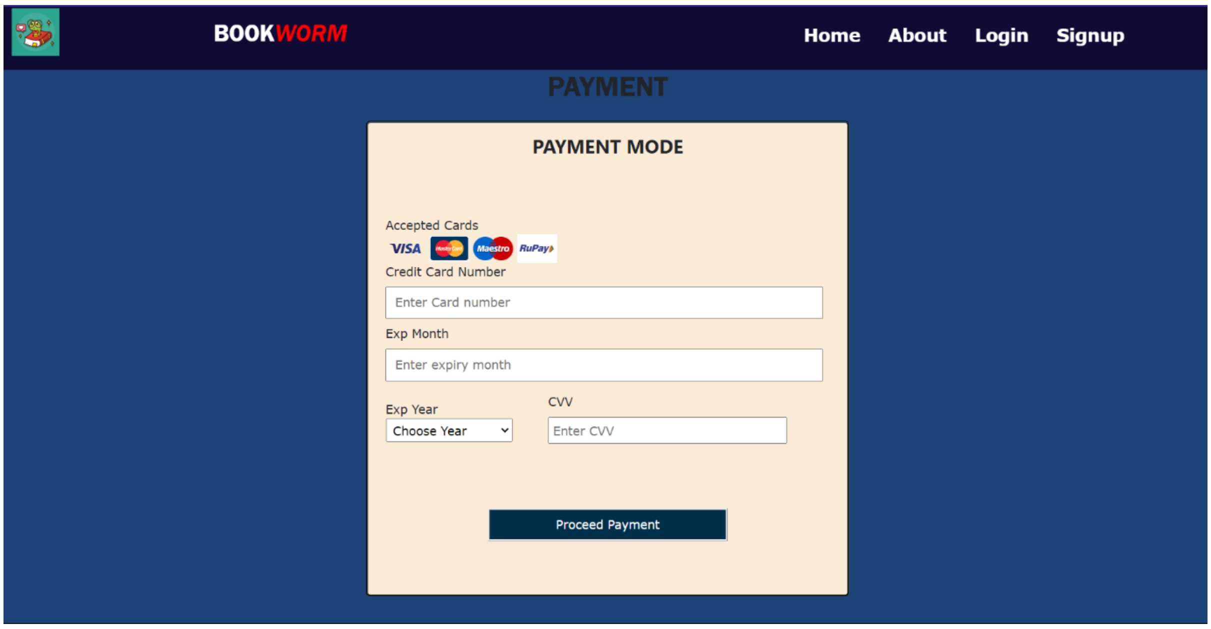
ADD BOOKS SECTION



SCIENCE BOOKS

BookID	Image	Title	Author	Price	Publisher	Availability	
31		Origins: Fourteen Billion Years of Cosmic Evolution	Neil deGrasse Tyson , Donald Goldsmith	499.35	W. W. Norton & Company	15	Remove
32		Parallel Worlds: A Journey through Creation, Higher Dimensions, and the Future of the Cosmos	Michio Kaku	529.34	Anchor	25	Remove
33		Ever Since Darwin: Reflections in Natural History	Stephen Jay Gould	365.42	W. W. Norton & Company	20	Remove
34		Co-Intelligence	Ethan Mollick	254.14	Portfolio	40	

VIEW BOOKS SECTION FOR ADMIN



PAYMENT

PAYMENT MODE

Accepted Cards

Credit Card Number

Enter Card number

Exp Month

Enter expiry month

Exp Year

Choose Year

CVV

Enter CVV

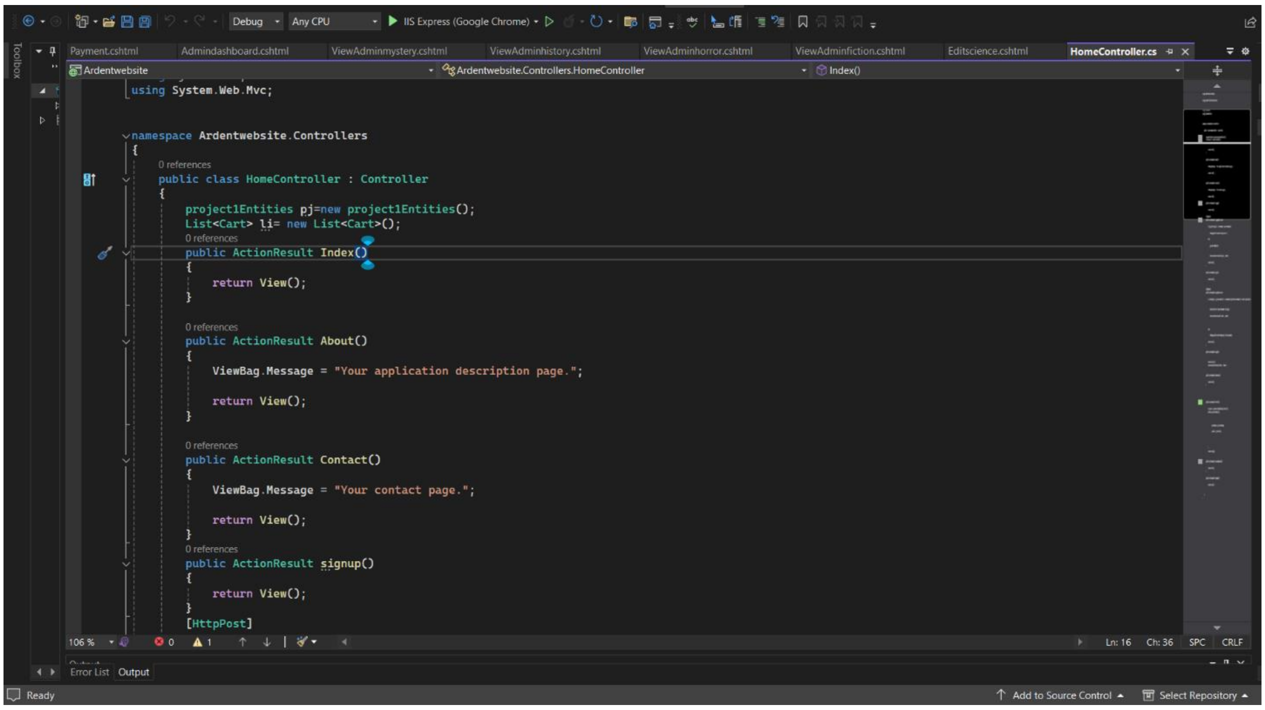
Proceed Payment

PAYMENT PAGE

SOURCE CODE OF PROJECT:

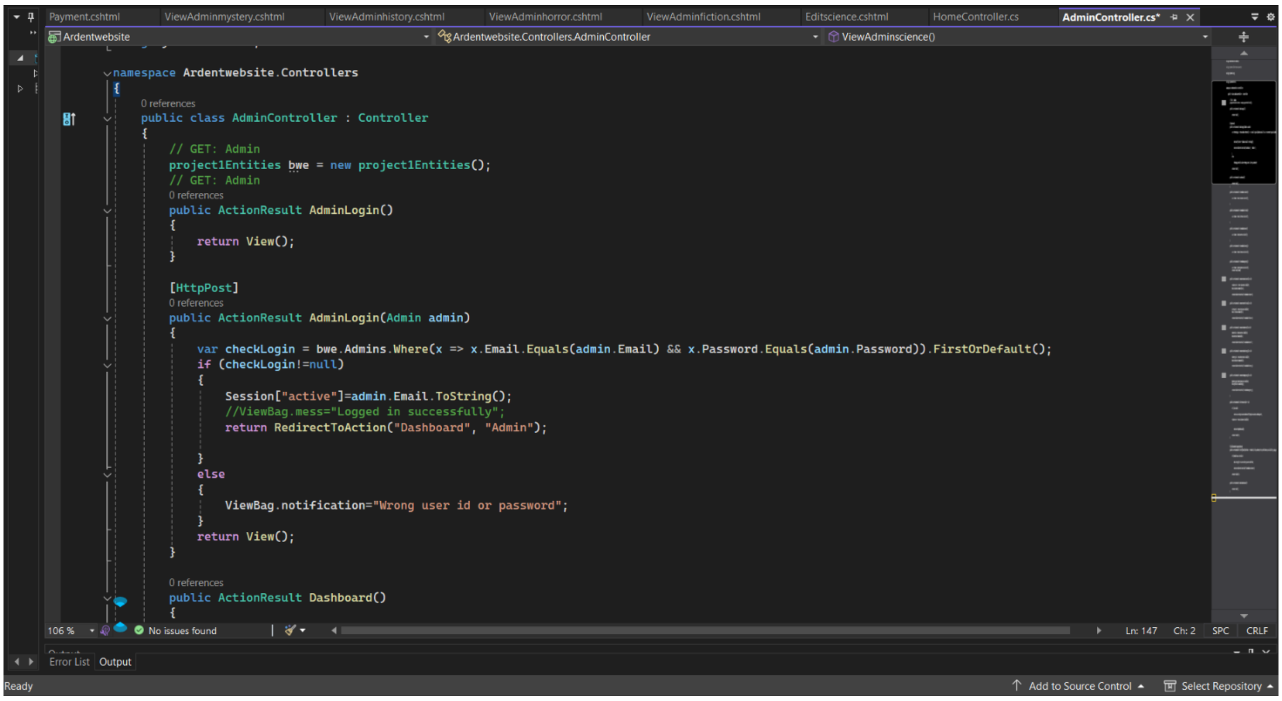
CONTROLLERS:

HomeController.cs



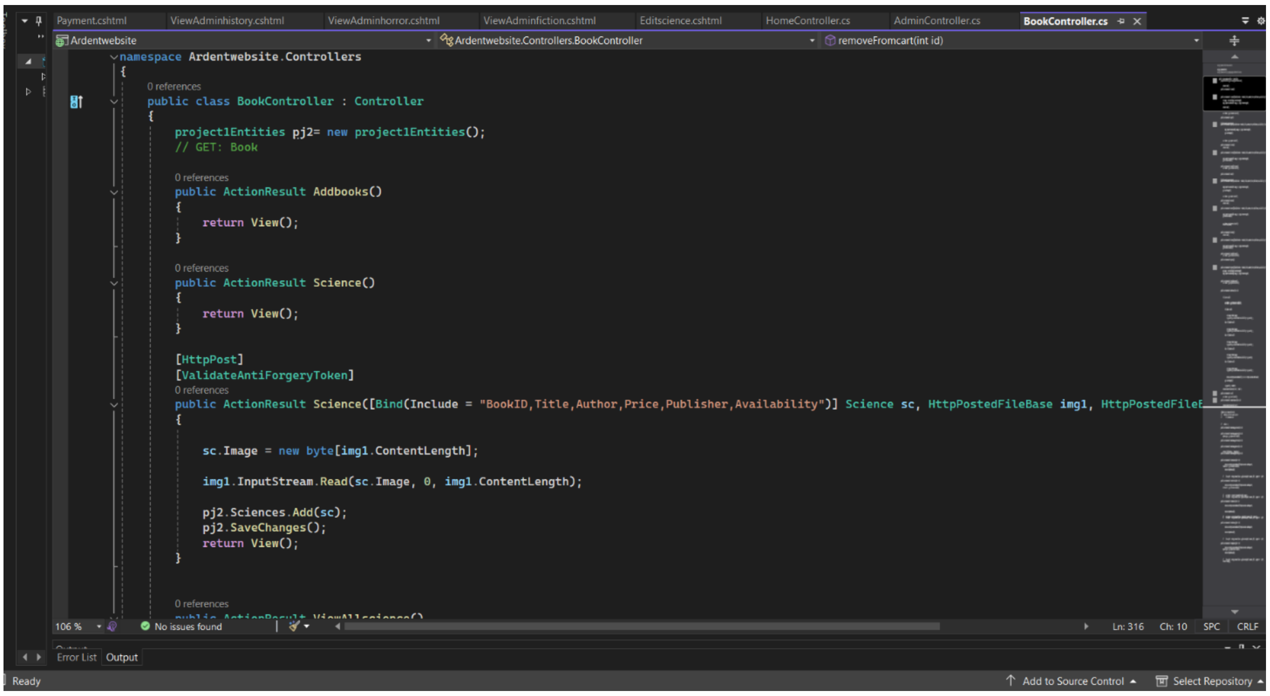
```
using System.Web.Mvc;


namespace Ardentwebsite.Controllers
{
    0 references
    public class HomeController : Controller
    {
        project1Entities pj=new project1Entities();
        List<Cart> li= new List<Cart>();
        0 references
        public ActionResult Index()
        {
            return View();
        }

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

            return View();
        }

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

            return View();
        }
        0 references
        public ActionResult signup()
        {
            return View();
        }
        [HttpPost]
```

AdminController.cs



```
namespace Ardentwebsite.Controllers
{
    0 references
    public class AdminController : Controller
    {
        // GET: Admin
        project1Entities bwe = new project1Entities();
        // GET: Admin
        0 references
        public ActionResult AdminLogin()
        {
            return View();
        }

        [HttpPost]
        0 references
        public ActionResult AdminLogin(Admin admin)
        {
            var checkLogin = bwe.Admins.Where(x => x.Email.Equals(admin.Email) && x.Password.Equals(admin.Password)).FirstOrDefault();
            if (checkLogin!=null)
            {
                Session["active"]=admin.Email.ToString();
                //ViewBag.mess="Logged in successfully";
                return RedirectToAction("Dashboard", "Admin");

            }
            else
            {
                ViewBag.notification="Wrong user id or password";
            }
            return View();
        }

        0 references
        public ActionResult Dashboard()
        {
```

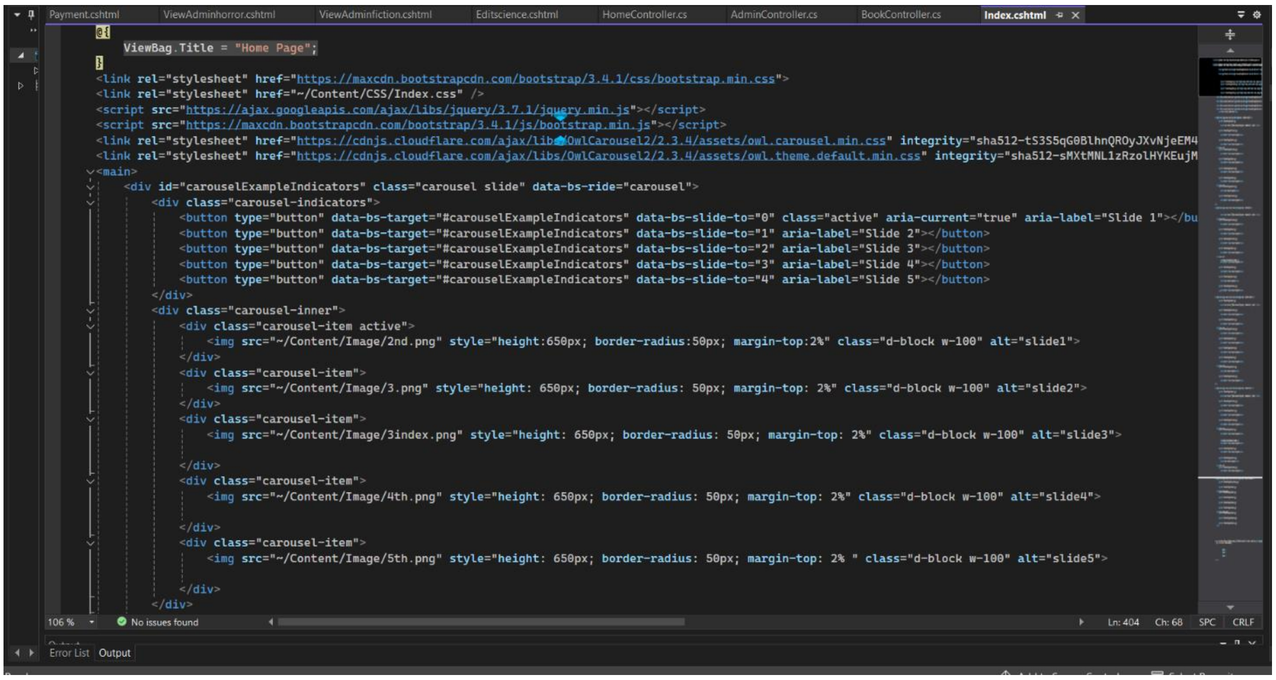
BookController.cs



```
namespace Ardentwebsite.Controllers
{
    0 references
    public class BookController : Controller
    {
        project1Entities pj2= new project1Entities();
        // GET: Book

        0 references
        public ActionResult Addbooks()
        {
            return View();
        }

        0 references
        public ActionResult Science()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        0 references
        public ActionResult Science([Bind(Include = "BookID,Title,Author,Price,Publisher,Availability")] Science sc, HttpPostedFileBase img1, HttpPostedFileE
        {

            sc.Image = new byte[img1.ContentLength];

            img1.InputStream.Read(sc.Image, 0, img1.ContentLength);

            pj2.Sciences.Add(sc);
            pj2.SaveChanges();
            return View();
        }


        0 references
```

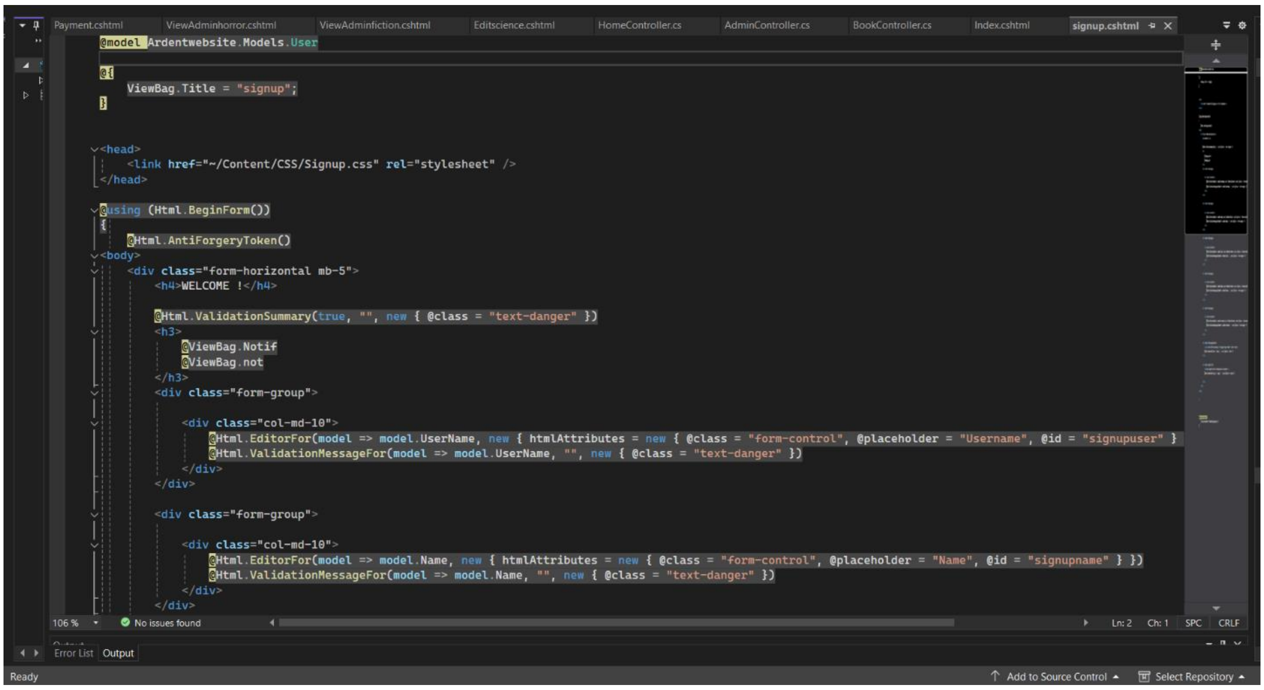
VIEWS

INDEX.cshtml



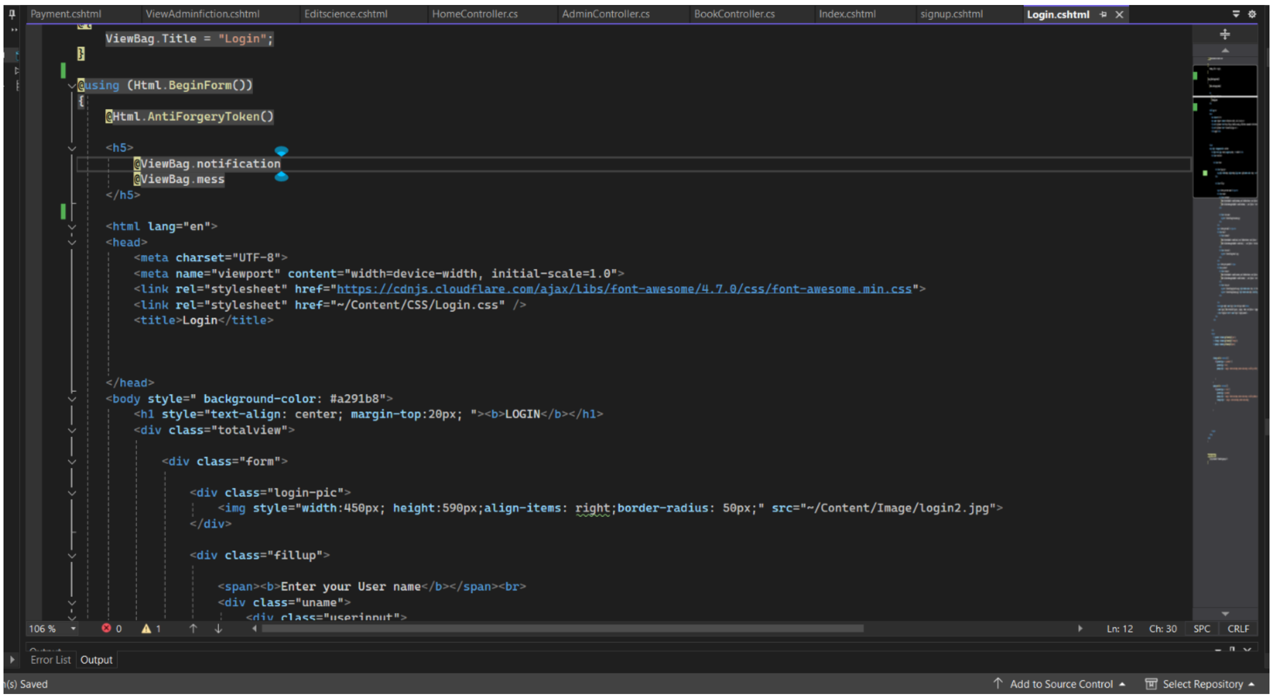
```
@{
    ViewBag.Title = "Home Page";
}
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<link rel="stylesheet" href="~/Content/CSS/Index.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.min.css" integrity="sha512-tS3S5qG0BlhnQROyJXvNjeEM4
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.theme.default.min.css" integrity="sha512-sMXtMNL1zRzolHYKEujM
<main>
    <div id="carouselExampleIndicators" class="carousel slide" data-bs-ride="carousel">
        <div class="carousel-indicators">
            <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></bu
            <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1" aria-label="Slide 2"></button>
            <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="2" aria-label="Slide 3"></button>
            <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="3" aria-label="Slide 4"></button>
            <button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="4" aria-label="Slide 5"></button>
        </div>
        <div class="carousel-inner">
            <div class="carousel-item active">
                <img src="~/Content/Image/2nd.png" style="height:650px; border-radius:50px; margin-top:2%" class="d-block w-100" alt="slide1">
            </div>
            <div class="carousel-item">
                <img src="~/Content/Image/3.png" style="height: 650px; border-radius: 50px; margin-top: 2%" class="d-block w-100" alt="slide2">
            </div>
            <div class="carousel-item">
                <img src="~/Content/Image/3index.png" style="height: 650px; border-radius: 50px; margin-top: 2%" class="d-block w-100" alt="slide3">

            </div>
            <div class="carousel-item">
                <img src="~/Content/Image/4th.png" style="height: 650px; border-radius: 50px; margin-top: 2%" class="d-block w-100" alt="slide4">

            </div>
            <div class="carousel-item">
                <img src="~/Content/Image/5th.png" style="height: 650px; border-radius: 50px; margin-top: 2% " class="d-block w-100" alt="slide5">

            </div>
        </div>
```

SIGNUP.cshtml



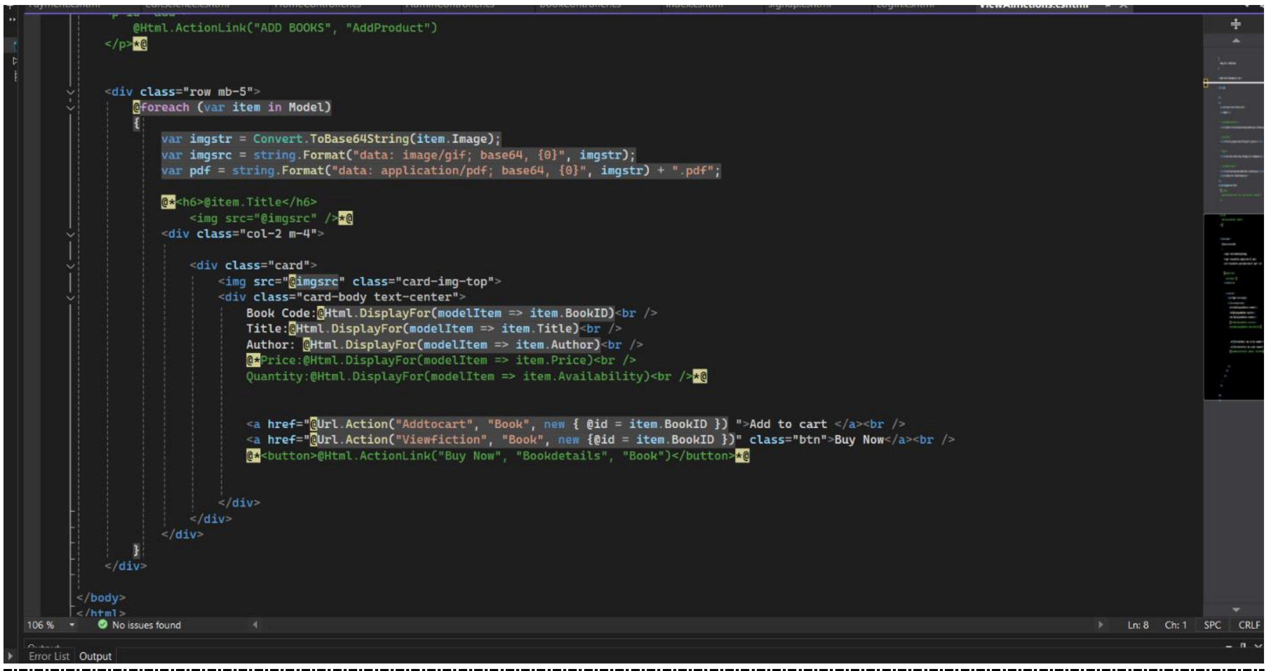
```
@model Ardentwebsite.Models.User

@{
    ViewBag.Title = "signup";
}


<head>
    <link href="~/Content/CSS/Signup.css" rel="stylesheet" />
</head>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
<body>
    <div class="form-horizontal mb-5">
        <h4>WELCOME !</h4>

        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <h3>
            @ViewBag.Notif
            @ViewBag.not
        </h3>
        <div class="form-group">

            <div class="col-md-10">
                @Html.EditorFor(model => model.UserName, new { htmlAttributes = new { @class = "form-control", @placeholder = "Username", @id = "signupuser" }
                @Html.ValidationMessageFor(model => model.UserName, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">

            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control", @placeholder = "Name", @id = "signupname" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>
```

LOGIN.cshtml



```
    ViewBag.Title = "Login";
}

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

    <h5>
        @ViewBag.notification
        @ViewBag.mess
    </h5>

    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
        <link rel="stylesheet" href="~/Content/CSS/Login.css" />
        <title>Login</title>



    </head>
    <body style=" background-color: #a291b8">
        <h1 style="text-align: center; margin-top:20px; "><b>LOGIN</b></h1>
        <div class="totalview">

            <div class="form">

                <div class="login-pic">
                    <img style="width:450px; height:590px;align-items: right;border-radius: 50px;" src="~/Content/Image/login2.jpg">
                </div>

                <div class="fillup">

                    <span><b>Enter your User name</b></span><br>
                    <div class="uname">
                        <div class="userinput">
```

VIEWALLBOOKS.cshtml

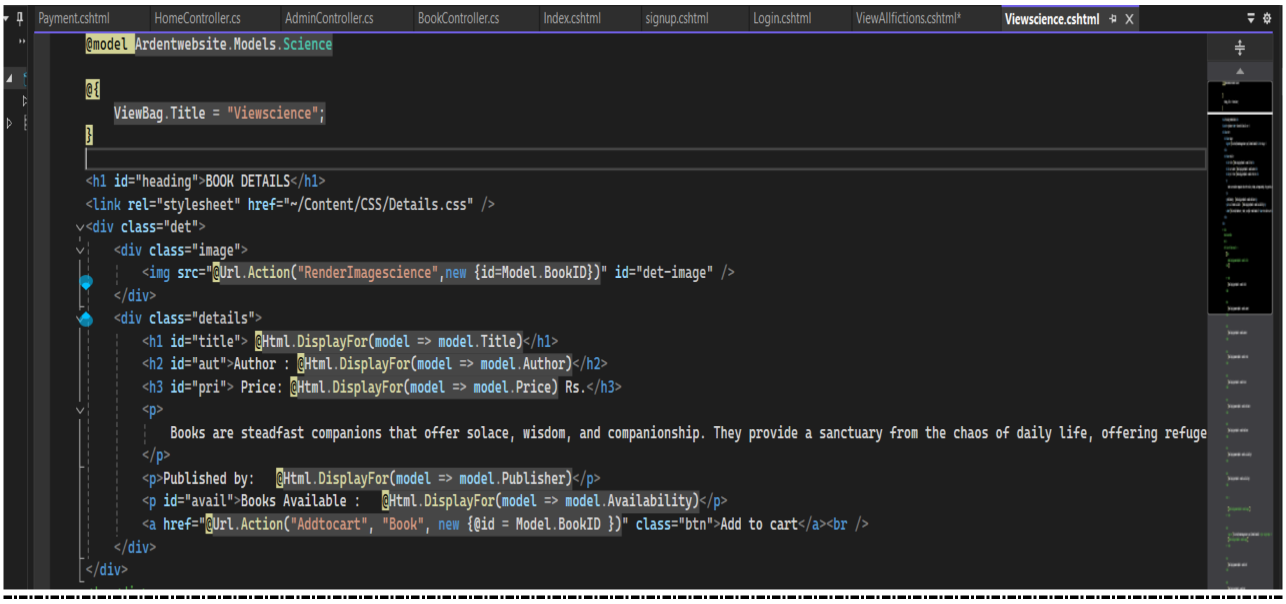


```
        @Html.ActionLink("ADD BOOKS", "AddProduct")
    </p>*@


    <div class="row mb-5">
        @foreach (var item in Model)
        {
            var imgstr = Convert.ToBase64String(item.Image);
            var imgsrc = string.Format("data: image/gif; base64, {0}", imgstr);
            var pdf = string.Format("data: application/pdf; base64, {0}", imgstr) + ".pdf";

            @*<h6>@item.Title</h6>
                <img src="@imgsrc" />*@
            <div class="col-2 m-4">

                <div class="card">
                    <img src="@imgsrc" class="card-img-top">
                    <div class="card-body text-center">
                        Book Code:@Html.DisplayFor(modelItem => item.BookID)<br />
                        Title:@Html.DisplayFor(modelItem => item.Title)<br />
                        Author: @Html.DisplayFor(modelItem => item.Author)<br />
                        @*Price:@Html.DisplayFor(modelItem => item.Price)<br />
                        Quantity:@Html.DisplayFor(modelItem => item.Availability)<br />*@


                        <a href="@Url.Action("Addtocart", "Book", new { @id = item.BookID }) ">Add to cart </a><br />
                        <a href="@Url.Action("Viewfiction", "Book", new {@id = item.BookID })" class="btn">Buy Now</a><br />
                        @*<button>@Html.ActionLink("Buy Now", "Bookdetails", "Book")</button>*@


                    </div>
                </div>
            </div>
        }
    </div>

</body>
</html>
```

106 % No issues found Ln: 8 Ch: 1 SPC CRLF

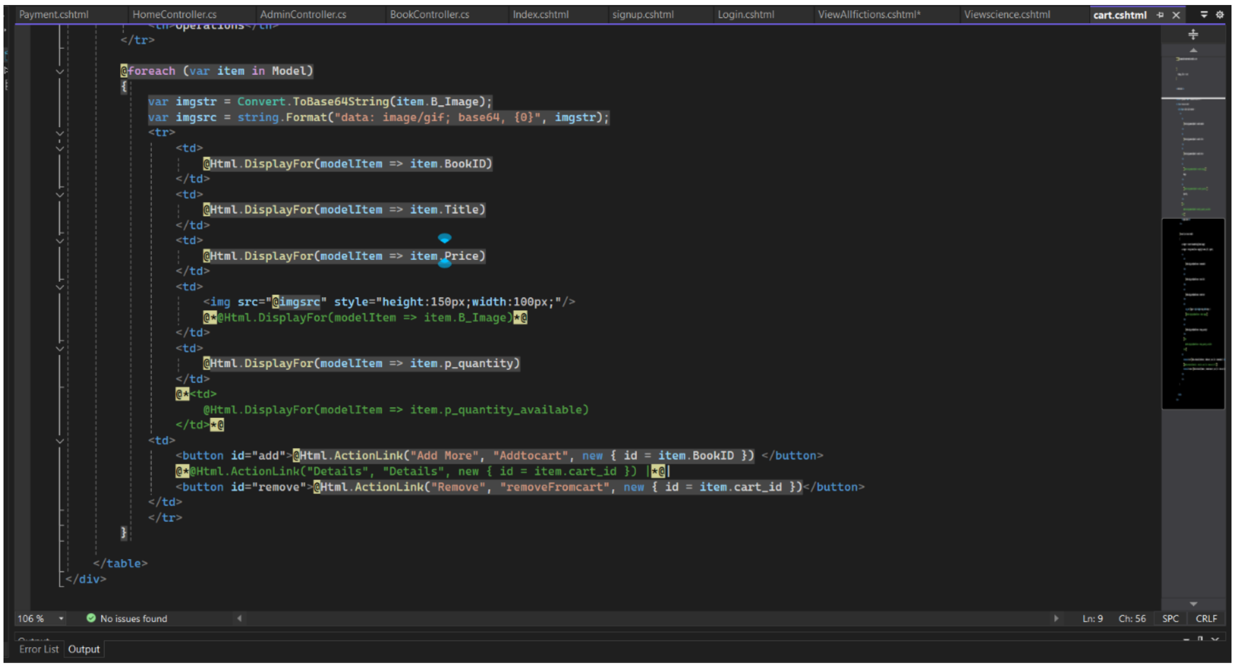
Error List Output

VIEWSCIENCE.cshtml



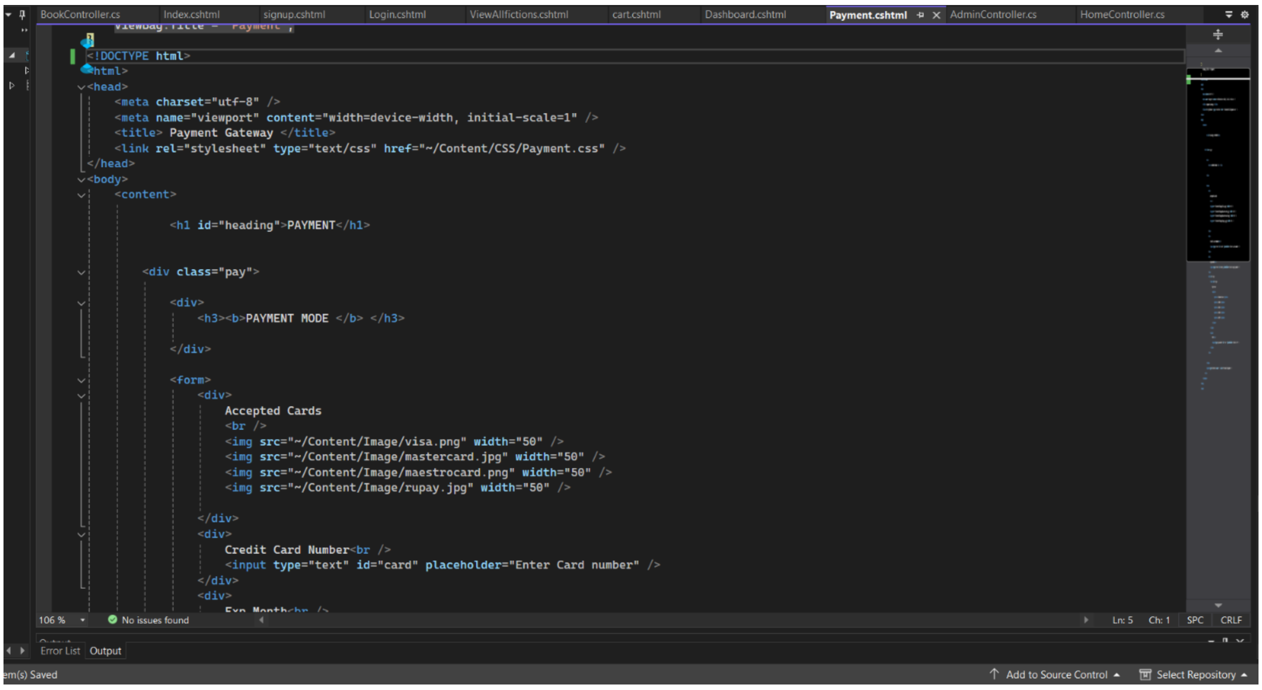
```
@model Ardentwebsite.Models.Science

@{
    ViewBag.Title = "Viewscience";
}

<h1 id="heading">BOOK DETAILS</h1>
<link rel="stylesheet" href="~/Content/CSS/Details.css" />
<div class="det">
    <div class="image">
        <img src="@Url.Action("RenderImagescience",new {id=Model.BookID})" id="det-image" />
    </div>
    <div class="details">
        <h1 id="title"> @Html.DisplayFor(model => model.Title)</h1>
        <h2 id="aut">Author : @Html.DisplayFor(model => model.Author)</h2>
        <h3 id="pri"> Price: @Html.DisplayFor(model => model.Price) Rs.</h3>
        <p>
            Books are steadfast companions that offer solace, wisdom, and companionship. They provide a sanctuary from the chaos of daily life, offering refuge
        </p>
        <p>Published by:   @Html.DisplayFor(model => model.Publisher)</p>
        <p id="avail">Books Available :   @Html.DisplayFor(model => model.Availability)</p>
        <a href="@Url.Action("Addtocart", "Book", new {@id = Model.BookID })" class="btn">Add to cart</a><br />
    </div>
</div>
```

CART.cshtml



```
            <th>Operations</th>
        </tr>

        @foreach (var item in Model)
        {
            var imgstr = Convert.ToBase64String(item.B_Image);
            var imgsrc = string.Format("data: image/gif; base64, {0}", imgstr);
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.BookID)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Title)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Price)
                </td>
                <td>
                    <img src="@imgsrc" style="height:150px;width:100px;"/>
                    @*@Html.DisplayFor(modelItem => item.B_Image)*@
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.p_quantity)
                </td>
                @*<td>
                    @Html.DisplayFor(modelItem => item.p_quantity_available)
                </td>*@
            <td>
                <button id="add">@Html.ActionLink("Add More", "Addtocart", new { id = item.BookID }) </button>
                @*@Html.ActionLink("Details", "Details", new { id = item.cart_id }) |*@
                <button id="remove">@Html.ActionLink("Remove", "removeFromcart", new { id = item.cart_id })</button>
            </td>
            </tr>
        }

    </table>
</div>
```

PAYMENT.cshtml



```
    ViewBag.Title = "Payment";
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title> Payment Gateway </title>
    <link rel="stylesheet" type="text/css" href="~/Content/CSS/Payment.css" />
</head>
<body>
    <content>

            <h1 id="heading">PAYMENT</h1>


        <div class="pay">

            <div>
                <h3><b>PAYMENT MODE </b> </h3>

            </div>

            <form>
                <div>
                    Accepted Cards
                    <br />
                    <img src="~/Content/Image/visa.png" width="50" />
                    <img src="~/Content/Image/mastercard.jpg" width="50" />
                    <img src="~/Content/Image/maestrocard.png" width="50" />
                    <img src="~/Content/Image/rupay.jpg" width="50" />

                </div>
                <div>
                    Credit Card Number<br />
                    <input type="text" id="card" placeholder="Enter Card number" />
                </div>
                <div>
                    Exp Month<br />
```

WHITE BOX TESTING

Whitebox testing of an online book ordering system involves examining the internal structure and logic of the system to ensure its functionality, security, and efficiency. This method scrutinizes the code, databases, and architecture of the system, aiming to identify any flaws or vulnerabilities that may exist within. Testers assess individual components, algorithms, and data flows, verifying that each aspect operates as intended and conforms to specifications. Through techniques such as code reviews, static analysis, and path coverage testing, Whitebox testing seeks to uncover issues such as incorrect calculations, database errors, or unauthorized access points. By thoroughly assessing the system's internal workings, Whitebox testing enhances the system's reliability and robustness, ultimately contributing to a seamless and secure online book ordering experience for users.

BLACK BOX TESTING

Blackbox testing of an online book ordering system involves evaluating its functionality without delving into the internal code or system architecture. Testers approach the system from an external perspective, focusing on its inputs, outputs, and user interactions. They create test scenarios based on user stories, requirements, and typical usage patterns, simulating various scenarios to assess the system's behavior. By executing tests such as boundary analysis, equivalence partitioning, and usability testing, Blackbox testing aims to uncover issues related to functionality, usability, and compatibility across different platforms and devices. This method ensures that the system behaves as expected from the user's standpoint, identifying any inconsistencies, errors, or unexpected behaviors that may impact the user experience. Through Blackbox testing, the online book ordering system can be validated comprehensively, providing users with a reliable and intuitive platform for browsing, selecting, and purchasing books.

OUTPUT TESTING

Output testing of an online book ordering system involves verifying the accuracy, completeness, and format of the information presented to users and other system stakeholders. This testing phase focuses on examining the system's outputs, such as web pages, emails, notifications, and reports, ensuring they align with expected requirements and standards. Testers assess whether the displayed information matches the input data, whether it's correctly formatted, and whether it's presented in a user-friendly manner. Additionally, output testing evaluates how the system handles various scenarios, such as error messages or edge cases, to guarantee that users receive clear and actionable feedback. Through meticulous examination of the system's output mechanisms, output testing enhances the overall usability, reliability, and satisfaction of the online book ordering experience.

User Acceptance Testing:

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done in regard to the following point:

a) Input Screen Design
b) Output Screen Design
c) Format of reports and other outputs.

GOAL OF TESTING

"Program testing can be used to slow the presence of bug, but never to slow their absence." If the results delivered by the system are different from the excepted ones then the system is incorrect and these bugs should be fixed.

INTEGRATION TEST REPORTS

Integration test reports for an online book ordering system provide a comprehensive overview of how different components and modules of the system interact and function together. These reports detail the outcomes of integration testing, which involves testing the interfaces and interactions between various system elements to ensure seamless operation. Integration test reports typically include information on the integration test plan, test cases executed, observed behaviours, identified defects, and any remedial actions taken. They document the integration points tested, data exchanged between modules, and the overall integration strategy employed. By summarizing the integration testing process and its results, these reports offer insights into the system's interoperability, reliability, and readiness for deployment, enabling stakeholders to make informed decisions regarding system integration and functionality.

FUNCTIONAL TESTING

Functional testing of an online book ordering system involves verifying that each functional aspect of the system performs as expected according to predefined requirements. Testers evaluate the system's features, such as user authentication, search functionality, book browsing, ordering process, payment processing, and order management, ensuring they meet user expectations and business objectives. Functional testing encompasses various techniques, including black-box testing, where testers assess the system's behaviour without knowledge of its internal workings, and white-box testing, which examines the system's internal logic and code structure. Test cases are designed to cover typical user scenarios as well as edge cases to validate the system's robustness and reliability. By rigorously testing the system's functionality, functional testing ensures that users can seamlessly navigate the platform, find desired books, place orders, and complete transactions, thereby enhancing user satisfaction and driving the success of the online book ordering system.

Testing Method Used

We have adopted a testing method which is a mix of both (structural) and black box (functional) testing. For modules we have adopted white box testing. Then we integrated the module into sub - systems and further into the system. These we adopted black box testing for checking the correctness of the system.

Requirements Validated and Verified:

- The data is getting entered properly into database.
- The Screens are being loaded correctly
- The Various functions specified are being performed completely.

DATABASE SECURITY

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So, you should follow some security measures. We have used security levels in database level at system level.

SYSTEM SECURITY

If we talk about the system security in our proposed system, we have implemented with the help of maintain the session throughout the system’s use. Once a user has logged out than he/she will not be able to perform any task before signing back again.

A high level of authentic login is given to the system so this is a very tedious task to enter without authorization and authentication.

LIMITATIONS

There are some limitations of using this software:-

1) Since it is an online project, customers need internet connection to use it.
2) People who are not familiar with computers can’t use this software.
3) Customer must have debit card or credit card to book tickets.

CONCLUSION

In conclusion, this documentation provides a comprehensive understanding of our online book ordering system, detailing its features, functionality, architecture, and testing processes. Through meticulous design and development, we have created a platform that offers users a seamless and convenient way to browse, select, and purchase books online. Our system not only prioritizes user experience but also incorporates robust security measures to safeguard user data and transactions. Extensive testing, including functional, integration, output, and white box testing, ensures the reliability, performance, and interoperability of our system. As we continue to evolve and enhance our platform, this documentation serves as a valuable resource for stakeholders, guiding them in leveraging the full potential of our online book ordering system to meet the needs and expectations of our users and stakeholders alike.

FUTURE SCOPE AND FUTURE ENHANCEMENTS

In future we would like to keep working on this project and make new additions to provide users with more advanced features and more detailed information. We have set our sights on the following additions in future:-

1) Implementing a Payment API.
2) Adding the pdf for the users to download the softcopy of the book they have purchased.
3) Adding a wishlist feature for the users.
4) Select a quantity for the books the users have added in their cart for ordering.

BIBLIOGRAPHY

- https://www.w3schools.com
- https://www.tutorialspoint.com
- https://www.youtube.com
- https://getbootstrap.com/
- https://fontawesome.com/

THANK YOU