C#-1

Islington College,Nepal

Document Details

Submission ID	trn:oid:::3618:95810752
Submission Date	May 14, 2025, 12:46 PM GMT+5:45
Download Date	May 14, 2025, 12:51 PM GMT+5:45
File Name	C#-1
File Size	89.7 KB

121 Pages

13,652 Words

74,688 Characters

12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

174 Not Cited or Quoted 12%
Matches with neither in-text citation nor quotation marks

6 Missing Quotations 1%
Matches that are still very similar to source material

1 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

2%	Internet sources
1%	Publications
12%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

174	Not Cited or Quoted 12% Matches with neither in-text citation nor quotation marks
6	Missing Quotations 1% Matches that are still very similar to source material
1	Missing Citation 0% Matches that have quotation marks, but no in-text citation
0	Cited and Quoted 0% Matches with in-text citation present, but no quotation marks

Top Sources

2%	Internet sources
1%	Publications
12%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Illinois Institute of Technology on 2024-11-21	1%
2	Submitted works	Ho Chi Minh University of Technology and Education on 2023-12-13	<1%
3	Internet	www.coursehero.com	<1%
4	Submitted works	Daffodil International University on 2025-01-12	<1%
5	Submitted works	Colorado Technical University Online on 2024-08-12	<1%
6	Submitted works	University of Wales Institute, Cardiff on 2024-10-02	<1%
7	Submitted works	University of Wales Institute, Cardiff on 2025-05-12	<1%
8	Internet	interviewprep.org	<1%
9	Submitted works	Middlesex University on 2023-07-04	<1%
10	Submitted works	Illinois Institute of Technology on 2024-11-16	<1%

11	Submitted works	
ESoft Metro Campus, Sri Lanka on 2025-04-26		<1%
12	Submitted works	
Melbourne Institute of Technology on 2024-05-08		<1%
13	Submitted works	
University of East London on 2025-05-09		<1%
14	Submitted works	
Montclair State University on 2025-03-28		<1%
15	Submitted works	
NCC Education on 2025-01-04		<1%
16	Submitted works	
University of Plymouth on 2024-01-08		<1%
17	Internet	
brainyboy.com		<1%
18	Submitted works	
islingtoncollege on 2025-05-14		<1%
19	Submitted works	
American Intercontinental University Online on 2025-04-29		<1%
20	Submitted works	
Sri Lanka Institute of Information Technology on 2022-04-05		<1%
21	Submitted works	
University of Ghana on 2023-11-02		<1%
22	Submitted works	
Asia Pacific Instutute of Information Technology on 2021-06-03		<1%
23	Submitted works	
London Metropolitan University on 2024-01-08		<1%
24	Submitted works	
University of Greenwich on 2025-04-30		<1%

25	Internet	
vdoc.pub		<1%
26	Submitted works	
Bradford College, West Yorkshire on 2017-05-05		<1%
27	Submitted works	
Monash University on 2023-10-08		<1%
28	Submitted works	
University of Wales Institute, Cardiff on 2024-01-30		<1%
29	Internet	
dotnettutorials.net		<1%
30	Submitted works	
American Intercontinental University Online on 2024-10-24		<1%
31	Submitted works	
Asia Pacific University College of Technology and Innovation (UCTI) on 2023-02-06		<1%
32	Submitted works	
Asia Pacific University College of Technology and Innovation (UCTI) on 2024-07-31		<1%
33	Submitted works	
Cavan and Monaghan Education and Training Board on 2025-05-12		<1%
34	Submitted works	
City University on 2025-05-13		<1%
35	Submitted works	
Colorado Technical University Online on 2024-10-20		<1%
36	Internet	
ambaland.com		<1%
37	Submitted works	
Asia Pacific University College of Technology and Innovation (UCTI) on 2021-02-07		<1%
38	Submitted works	
Middlesex University on 2022-07-01		<1%

39	Submitted works	
	HELP UNIVERSITY on 2023-04-09	<1%
40	Submitted works	
	University of Wales Institute, Cardiff on 2024-11-13	<1%
41	Submitted works	
	Whitireia Community Polytechnic on 2018-11-26	<1%
42	Internet	
	free2live.eu	<1%
43	Submitted works	
	islingtoncollege on 2025-04-08	<1%
44	Submitted works	
	Illinois Institute of Technology on 2024-09-27	<1%
45	Submitted works	
	Postgraduate Institute of Management on 2023-08-18	<1%
46	Submitted works	
	University of Ghana on 2023-11-10	<1%
47	Submitted works	
	Far Eastern University on 2016-11-10	<1%
48	Submitted works	
	The University of Wolverhampton on 2022-05-01	<1%
49	Internet	
	www.scribd.com	<1%
50	Submitted works	
	HELP UNIVERSITY on 2019-10-30	<1%
51	Submitted works	
	Help University College on 2010-10-03	<1%
52	Internet	
	www.thevoicebw.com	<1%

53	Submitted works	
	Asia Pacific University College of Technology and Innovation (UCTI) on 2024-03-22	<1%
54	Submitted works	
	Asia Pacific University College of Technology and Innovation (UCTI) on 2025-04-13	<1%
55	Submitted works	
	CITY College, Affiliated Institute of the University of Sheffield on 2018-03-12	<1%
56	Submitted works	
	Gulf College Oman on 2020-12-24	<1%
57	Submitted works	
	Illinois Institute of Technology on 2024-11-21	<1%
58	Submitted works	
	Military Technological College on 2023-12-03	<1%
59	Submitted works	
	RMIT University on 2024-08-18	<1%
60	Submitted works	
	University of Bahrain on 2025-05-05	<1%
61	Submitted works	
	University of London External System on 2019-04-02	<1%
62	Submitted works	
	University of Technology, Sydney on 2024-05-12	<1%
63	Internet	
	akit.cyber.ee	<1%
64	Submitted works	
	islingtoncollege on 2025-01-23	<1%
65	Submitted works	
	Asia Pacific University College of Technology and Innovation (UCTI) on 2018-10-26	<1%
66	Submitted works	
	Bradley University on 2023-11-29	<1%

67 Submitted works	
Colorado Technical University Online on 2010-03-12	<1%
68 Submitted works	
Colorado Technical University Online on 2025-04-19	<1%
69 Submitted works	
ESoft Metro Campus, Sri Lanka on 2025-04-05	<1%
70 Submitted works	
Segi University College on 2014-04-21	<1%
71 Submitted works	
University of Bahrain on 2015-05-20	<1%
72 Internet	
zir.nsk.hr	<1%
73 Submitted works	
Asia Pacific University College of Technology and Innovation (UCTI) on 2017-02-27	<1%
74 Submitted works	
Central Queensland University on 2024-02-12	<1%
75 Submitted works	
City University on 2012-11-26	<1%
76 Submitted works	
City University on 2025-04-28	<1%
77 Submitted works	
De Montfort University on 2011-12-13	<1%
78 Submitted works	
De Montfort University on 2022-05-03	<1%
79 Submitted works	
Higher Education Commission Pakistan on 2023-06-08	<1%
80 Submitted works	
Majan College on 2016-06-04	<1%

81	Submitted works	
	Munster Technological University (MTU) on 2024-09-23	<1%
82	Submitted works	
	National College of Ireland on 2018-12-07	<1%
83	Submitted works	
	Queen Mary and Westfield College on 2025-03-27	<1%
84	Submitted works	
	University of Greenwich on 2011-05-25	<1%
85	Submitted works	
	University of Greenwich on 2024-11-29	<1%
86	Submitted works	
	University of Melbourne on 2017-10-28	<1%
87	Submitted works	
	University of Wales, Lampeter on 2024-07-24	<1%
88	Submitted works	
	University of Wales, Lampeter on 2024-07-25	<1%
89	Submitted works	
	Asia Pacific International College on 2023-08-17	<1%
90	Submitted works	
	Asia Pacific University College of Technology and Innovation (UCTI) on 2021-11-13	<1%
91	Submitted works	
	General Sir John Kotelawala Defence University on 2019-10-29	<1%
92	Submitted works	
	Higher Education Commission Pakistan on 2021-06-17	<1%
93	Submitted works	
	Nanyang Technological University on 2024-11-15	<1%
94	Submitted works	
	South Bank University on 2024-05-02	<1%

95 Submitted works

TAFE Queensland Brisbane on 2024-06-14 <1%

96 Submitted works

University of Greenwich on 2025-04-26 <1%

97 Submitted works

University of Wales, Lampeter on 2025-04-23 <1%

98 Submitted works

HELP UNIVERSITY on 2018-05-21 <1%

99 Submitted works

Letterkenny Institute of Technology on 2024-06-06 <1%

100 Submitted works

Southampton Solent University on 2025-02-07 <1%

101 Submitted works

University of Dayton on 2025-02-10 <1%

102 Submitted works

University of Greenwich on 2025-05-02 <1%

Introduction

1.1 Problem/Purpose

42 In today's digital age, online shopping has become increasingly popular because it's convenient and saves a lot of time. Many people prefer shopping online these days, Ghyal Bhaag Book Store, a private book library, wants to get more customers by creating an online platform where people can browse through books and buy them without having to physically visit the store. Our main goal for this project is to create a user-friendly C#.NET application that lets users explore, search, and order books from the library's collection. We also want to help Ghyal Bhaag Book Store manage their books and customer orders better, so they don't get confused with all the orders coming in. 23

The current system at Ghyal Bhaag Book Store is pretty old fashioned, customers have to physically go there to see what books are available, which is time-consuming and sometimes frustrating if they don't find what they're looking for. With our online system, customers can check availability and place orders from the comfort of their homes, which is much more convenient.

1.2 Solution/Scope

Our solution for Ghyal Bhaag Book Store is a comprehensive C#.NET web application that connects to a database to store all the information. The system will let users search for books in many ways, they can look up for books by title, author, genre, ISBN, publisher, or even keywords in the description. This makes finding exactly what you want much more efficient. Users can also filter books based on whether they're available, how much they cost, what ratings they have, what language they're in, and what format they come in.

The Ghyal Bhaag Book Store website will have different sections that make browsing straightforward and enjoyable. There will be tabs for All Books where you can see everything, Bestsellers for the most popular ones, Award Winners for books that won prizes, New Releases for books published in the last three months, New Arrivals for books that just came to the store in the last month, Coming Soon for books that aren't out yet but will be soon, and Deals where you can find discounted books.

Right now, Ghyal Bhaag Book Store only lets people pick up their books in person. After someone places an order online, their order status is set to "pending" in the system. They'll get an email with a special claim code and their bill. Then they must go to the store with this code to get their books. When they arrive at the store, the staff check the system using their claim code, verify the order, change the status from "pending" to "completed," and hand over the books to the customer. It's not delivered to your doorstep yet, but maybe that could be a future upgrade.

Our system will have different types of users who can do different things:

Guests can just look through the books without signing up

Members/Customers can do useful things like save books they're interested in (like a wishlist), add books to their shopping cart, place orders that they can cancel if they change their mind, and write reviews for books they've bought

Staff at Ghyal Bhaag Book Store can process the claim codes when members come to pick up their books, check the pending orders in the system, and update the order status after handing over the books

67 Admin have full control, they can add new books, update information about existing books, delete books if needed, manage how many books are in stock, set up discounts, and post announcements about sales or new arrivals

84

1.3 Aims and Objectives

Our main aim is to create an effective C#.NET application that helps Ghyal Bhaag

85

Book Store get more customers online and makes it easier for people to find and order books. We want the experience to be smooth and enjoyable so customers will keep coming back.

Objectives:

Create a system where users can sign up and log in securely to Ghyal Bhaag Book Store

Make a paginated book catalog so users can browse through books easily without getting overwhelmed

Add detailed information about each book so customers know exactly what they're getting

Create effective search, sort, and filter options so users can find exactly what they want

Let members bookmark books they're interested in for later (because sometimes you want a book but can't buy it right away)

Create a shopping cart system where members/customers can add books they want to buy

Let members place orders and cancel them if they change their mind

Implement an order status tracking system (pending, completed, cancelled)

Add automatic discounts to make customers happy - like 5% off when they order 5 or more books at once, and a special 10% discount after they've successfully completed 10 orders.

Let members/customers write reviews and rate books they've purchased to help other customers decide

Send confirmation emails with claim codes and bills so members have proof of their order.

Create a system for Ghyal Bhaag Book Store staff to process orders when members come to pick up their books.

Allow staff to check pending orders, verify claim codes, and update order status after handing books to customers.

Give admin tools to manage all the books, update inventory, and set discounts

Let admin create special "On Sale" flags for books on discount

Create a system for timed announcements so admin can post about deals or new books

1.4 Deliverables

By the end of this project, we'll provide Ghyal Bhaag Book Store with:

A complete, working C#.NET book library web application that does everything mentioned above

Documentation explaining what the system needs to run properly

Helpful diagrams showing how the system works (Usecases, erd flowchart etc)

All the source code with helpful comments so others can understand how it works

Screenshots showing what the different pages look like

A report showing what each person in our team did (if it's a group project)

A final project report with our results and ideas for how to make the system even better in the future

We're really excited about this project because it combines our programming skills with a real-world problem that needs solving. Plus, who doesn't love books? This system will make finding and buying books from Ghyal Bhaag Book Store much easier for everyone.

2. Feature and functionality

2.1 Features

Feature 1: User can browse the paginated book catalogue

All users can view the book catalogue. The catalogue is paginated, meaning the books are shown page by page to make browsing smoother and faster. Use the "Next" and "Previous" buttons to navigate.

Page 1 of the book catalogue

86 Figure 1: Book catalogue page 1

Page 2 of the book catalogue

Figure 2: Book catalogue page 2

Feature 2: User can view book details

77 If you click any book from the catalogue, you will be able to view complete details like the book's name, author, price, description, reviews, and availability.

53 Book Catalogue page

Figure 3: Book catalogue page

Book Details page

Figure 4: Book Details page

Feature 3: User can apply search, sort and filters

62 You can search for books by the title or author in the search bar. And also, books can be sorted by price, newest, highest rated and can be filtered by category, availability and on sale.

Searching by title, Description

Figure 5: Searching by title

Figure 6: Searching by Description

Sorting by Price (Min and Max).

Figure 7: Sorting by Price (Min and Max).

Filtering by Genre.

Figure 8: Filtering by Genre.

Feature 4: User can register to become member

Users can create an account using the "Register" option. You will need to enter your email and other required details. After successful registering, you can log in to access member facilities.

Figure 9: Registering user

Figure 10: Email Confirmation

Figure 11: Loggin in

Figure 12: Database

Feature 5: Member can bookmark books (Add to Whitelist)

Members can add favorite books on their Whitelist (a personal bookmark list) to keep a record of books they like or wish to buy in the future.

22 Adding to wish list

Figure 13: Adding to wish list

Figure 14: Added to wish list

Figure 15: Wishlist page

Feature 6: Member can cart books (Add to Cart)

Members can add books to their Cart, and when it suits them, proceed to place an order. The order will be in their order history.

Figure 16: Adding to book to Cart

Figure 17: Added to Book to Cart

Figure 18: Cart Page

Features 7: Member can place cancellable order

Members can cancel their orders any time before the order is processed by staff.

Figure 19:Cart items

Figure 20: Deleting the books

Figure 21: After deleting

Features 8: Member can review only to purchase books

18 Members can only leave reviews for the books they have bought, ensuring genuine feedback.

Figure 22: Adding review

Figure 23: after adding review

Feature 9: Member can get 5% discounts for an order of 5+ books and the 10% stackable discount after 10 successful orders

Members get an automatic 5% discount if they purchase 5 or more books at one time. Members also get an additional 10% discount for the successful completion of 10 orders.

Figure 24: 5% discount on more than 5 items on cart

Feature 10: Member can get claim code with bill issued in registered email inbox

When placing the order, the member will receive an order bill and claim code in his/her registered email address. The claim code is for future use to pick up orders or verify them.

Figure 25: Bill and claim code in email inbox

Feature 11: Staff can process the member claim code to fulfil the order

Staff have access to the order system where they can enter the members' claim code to confirm and process the order, ensuring the right person receives the right order.

Figure 26: Staff Process Order page

Figure 27: after successful order process

Feature 12: Admin can manage book catalogue (CRUD) and Inventory

Admin has full control to add, edit, delete, and manage books, stock, and inventory.

Creating new books

Figure 28: Creating new pages

Figure 29: Database

Figure 30: Book Page

Deleting books

Figure 31: Books details

Figure 32: Deleting Book

Figure 33: After Deleting the book

Editing Details of the book

Figure 34: Book Details Page

Figure 35: Editing the price

Figure 36: After Editing

Feature 13: Admin can set timed discounts, optionally with “On Sale” flag

Admin can give time-based discounts on some books. They can also mark these books as "On Sale" to highlight them.

Feature 14: Admin can write timed banner announcement

Admin can create banner messages to display on top of the site. They can be display for short periods of time to promote books, offers, or vital notices.

Figure 37: Creating Announcement

Figure 38: Announcement in the announcement page

Figure 39: Announcement banner in the home page

Feature 15: Books from a successful order is broadcasted at real-time with a suitable message.

When an order is made by a member, the books purchased are reported live on the site with a message, inviting others to check them out.

Figure 40: Successful Broadcasted of order

2.2 User Manual

The navigation bar on the page provides access to different sections.

Figure 41: Navbar

Browse the Paginated Book Catalogue

Steps:

Go to Home page or choose “Books” from the navbar.

Scroll down to view books listed page by page.

Move from page to page using Next or Previous buttons.

Figure 42: paginated guide

90 Clicking on the Next button navigates to the next book catalogue. And clicking on the Previous Button will navigate to the previous book catalogue.

View Book Details

Steps:

Click on a book title or cover from the Catalogue. This displays the Book Details page, which includes Title, Author, Genre, Description, Price, Stock, status, Rating & Reviews.

Figure 43: Book Page Manul

Clicking on any book cover/title will show their details.

Figure 44: Book details Manual

Apply Search, Sort, and Filters   Steps:

87 Navigate to the Catalogue page. use the Search Bar to find books by title, author.

Use the Filter panel (sidebar or top): By Genre, By Rating, By Price Range, By Availability

Figure 45: Filter logic manual

In here is all the search, sorting and filter logic. You can search by Title, Author, Genre, Description, Price (min to max), Stock.

Register to Join as a Member Steps:

Click Register from the navigation bar.

Enter the registration form:

Name, Email, Password, Address, etc.

Click Submit.

A confirmation email is sent.

Once verified, log in to use member features.

Figure 46: Registering user

Figure 47: Email Confirmation

Figure 48: Loggin in

Bookmark Books (Add to Whitelist) Steps:

Open a Book Details page by click on a book title or cover from the Catalogue.

Click the Bookmark / Add to Whitelist icon.

22 The book is now in your Wishlist (available on Wishlist page).

Figure 49: Adding to wish list

Figure 50: Added to wish list

Figure 51: Wishlist page

Add to Cart Steps:

Open a Book Details page by click on a book title or cover from the Catalogue.

Click Add to Cart icon.

9 View your Cart by clicking the Cart icon in the navbar

Figure 52: Adding book to Cart

Figure 53: Added to Book to Cart

Figure 54: Cart Page

Review of Purchased books

Steps:

Go to Order History on your dashboard.

Choose a complete order.

Write a Review and hit Submit.

Books from only completed orders will allow review.

Figure 55: Adding review

Figure 56: after adding review

Process Member Claim Code Steps:

Staff login.

Go to the Claim Code Processing Panel from the dashboard.

Enter or scan the Claim Code.

View the order details attached to the code.

Click Mark as complete order.

Figure 57: Staff Process Order page

Figure 58: after successful order process

Management of CRUD (Create, Read, Update, Delete) of Book Catalogue and Inventory Steps:

Admin login.

Navigate to Book Management page.

To Create/add new book. Click Add New Book, then insert all details of the book and then click submit.

To Edit: Click on any book, then click Edit button and update the details you want and then click Save button.

25 To Delete: Click on the book you want to delete and click Delete button.

Figure 59: Creating new pages Manual

Figure 60: Book Page Manual

Deleting books

Figure 61: Books details

Figure 62: Deleting Book

Figure 63: After Deleting the book

Editing Details of the book

Figure 64: Book Details Page

Figure 65: Editing the price

Figure 66: After Editing

Create Timed Banner Announcements Steps:

Admin Dashboard, then navigate to Announcement page.

Create New Announcement.

fill all the required details and also Start and End time/date of the announcement. And

Publish the Announcement.

3. Proof of Work (PoW)

3.1 Designs

Low Fidelity

Signup page

11 Figure 67: Singup Wireframe

Home Page

Figure 68: Home Page Wireframe

Login Page

Figure 69: Login Page Wireframe

Manage Books Page

Figure 70: Manage Books Wireframe

High Fidelity

Signup Page

26

Figure 71: Signup page Prototype

Login Page

Figure 72: Login Page Prototype

Home Page:

Figure 73: Home Page Prototype

Books Page:

Figure 74: Books Page Prototype

Admin Manage Books Page:

Figure 75: Admin Manage Books Page

3.2 ER Diagram

18 An Entity Relationship Diagram (ERD) is a visual tool that shows how items in a database are connected using symbols and lines. It helps design databases by mapping relationships between entities and supports deeper analysis by organizing scattered data into clear structures. ERDs also serve as a foundation for creating efficient and organized relational databases. (Ivan Belcic, 2024)

Figure 76: Entity Relationship Diagram (ERD)

Entities and Attributes

User

Represents the customer or user of the system.

UserID (PK)

FirstName

LastName

UserName

Email

PhoneNumber

PasswordHash

Roles

Defines roles that users can have (e.g., Admin, Customer).

RoleID (PK)

Name

UserRoles

Links users with roles (many-to-many relationship).

ID (PK)

UserID (FK)

RoleID (FK)

RoleClaims

Claims associated with a role (for authorization).

ID (PK)

RoleID (FK)

ClaimType

ClaimValue

UserClaims

Claims specific to users.

ID (PK)

UserID (FK)

ClaimType

ClaimValue

Fulfilled

UserToken

Tokens used for authentication sessions or recovery.

UserID (FK)

LoginProvider

Name

Value

Book

Details about books in the bookstore.

BookID (PK)

Title

Author

DatePublished

Description

Genre

PhysicalAccess

Price

Stock_Initial

Stock_Avail

ISBN

Language

Format

Bookmark

Tracks books bookmarked by users.

ID (PK)

UserID (FK)

BookID (FK)

Review

User-submitted reviews for books.

66 ReviewID (PK)

UserID (FK)

BookID (FK)

Rating

Comment

CartItem

Items added to the user's shopping cart.

CartItemID (PK)

UserID (FK)

BookID (FK)

Quantity

Order

Orders placed by users.

OrderID (PK)

UserID (FK)

CreatedDate

TotalAmount

DiscountApplied

OrderItem

Books included in an order.

OrderItemID (PK)

BookID (FK)

OrderID (FK)

Quantity

Discount

Promotional discounts apply to orders.

DiscountID (PK)

Code

Percentage

IsValid

Expiry

Announcement

System-wide announcements (e.g., for promotions).

ID (PK)

Title

Description

StartDate

EndDate

Relationships Between Entities

Entity A

Entity B

Cardinality

Description

User

UserToken

1: N

14 A user can have multiple tokens

User

UserClaims

1: N

A user can have multiple claims

Roles

RoleClaims

1: N

A role can have multiple claims

User

UserRoles

1: N

A user can have multiple roles

User

CartItem

1: N

A user can have many cart items

User

Order

7 1: N

A user can place many orders

User

Review

1: N

A user can make many reviews

User

Bookmark

1: N

A user can bookmark many books

CartItem

Book

N:1

Each cart item is for one book

OrderItem

Book

7 N:1

Each order item contains one book

7 Order

OrderItem

1: N

An order includes multiple items (books)

Order

Discount

N:1

Each order can apply one discount

Review

Book

N:1

A review is linked to a single book

Bookmark

Book

N:1

A bookmark references a single book

User

Announcement

7 1: N

A user can view multiple announcements.

Order

UserClaims

1:1

Each order generates one claim

3.3 Use Case Diagram

47 A use case is a method used to define and organize system requirements. It shows the

34 steps of interaction between users and the system to achieve a specific goal, and it

helps document the user’s actions to complete an activity. (Brush, 2022)

3 Figure 77: Use Case Diagram

3.3.1 High Level use case description

39 Use case: Register Account

Table 1: Use case of Register Account

Use Case:

Register Account

Actors:

Guest

Description:

Guest user creates account to become a registered member of the bookstore. They enter their details and submit the form and the system verifies their details. Once the account is activated they get access to member only 16 features like adding books to their wish list and placing orders, etc.

55 Use case: Login into System

Table 2: : Use case of Login

Use Case:

Login Into System

Actors:

Admin, Customer

Description:

9 Users who have registered can log in to the system and access their accounts. Members can add books to their wish list, add items to their cart, and use other

member only features. Admins have a separate login page to access and manage the system.

2 Use case: View Announcements

Table 3: Use case of View Announcements

Use Case:

View Announcements

Actors:

Customer

Description:

Users can read bookstore important announcements like updates and news uploaded by the admin.

Use case: Browse Catalogue

21 Table 4: Use case of Browse

Use Case:

Browse Catalogue

Actors:

Customer, Guest User

Description:

11 Users can view available books. The system displays book details like title, author, price, and cover image. Users can filter the catalog by criteria such as author, genre,

4 stock, price range, ratings, language, and format and users can also search for books

11 by title, ISBN or description. Sorting options will allow users to organize books by title, publication date, price, or popularity.

Use case: View Availability Status

Table 5: Use case of View Availability Status

3 Use Case:

View Availability Status

Actors:

Customer

Description:

Registered member can only view the availability status of the books, if the book is available in the store or not.

2 Use case: Add To Favorites

Table 6: Use case of Add To Favorites

Use Case:

21 Add To Favorites

Actors:

Customer

Description:

Only member can save books to their favorites list for quick access in the future. If a

40 guest tries to add a book to favorite then they are prompted to log in or register.

2 Use case: Add Book to Cart

4 Table 7: Use case of Add Book to Cart

Use Case:

Add Book To Cart

Actors:

Customer

Description:

Only member can add books to their shopping cart and select quantities. The system updates stock availability and display users if the book is out of stock. They can update quantities, remove books, and view the total cost of the books. After finalizing the

32 books, users move to checkout process. If a guest is trying to add items to the cart they are asked to log in first.

41 Use case: Place Order

Table 8 : Use case of Place Order

Use Case:

Place Order

Actors:

Customer

Description:

Members can place an order, after placing an order users receive a confirmation email with their claim code and receipt. Users can then review, comment or cancel the order if it has not been fulfilled. Discounts can be applied according to specific conditions.

71 Use case: Manage Books

Table 9 : Use case of Manage Books

46 Use Case:

Manage Books

Actors:

Admin

Description:

Admins can add, update, or delete books in the bookstore. They can enter book details like title, price, and stock when adding a new book. They can edit fields like price and description and update it. They can also delete books.

2 Use case: Manage Announcements

Table 10 : Use case of Manage Announcements

Use Case:

Manage Announcements

Actors:

Admin

Description:

Admins can post, edit or remove announcements for users such as promotions and updates about the book store. The system make sure valid dates so that outdated announcements don’t appear. Users see announcements in real time.

2 Use case: Manage Discounts

Table 11 : Use case of Manage Discounts

Use Case:

Manage Discounts

Actors:

Admin

Description:

Admins mark books as "On Sale." On some specific books and the discounts apply automatically at checkout.

49 Use case: Manage Orders

Table 12 : Use case of Manage Orders

Use Case:

Manage Orders

Actors:

Admin

Description:

Admins handle customer orders, including viewing order details and fulfilling pickups.

Admin also track order status and manage records.

94 Use case: Process Orders

Table 13 : Use case of Add To Favorites

Use Case:

Process Orders

Actors:

Staff

Description:

The Staff verify claim codes and mark orders as completed. If the code is invalid or the order is already fulfilled, the system alerts staff.

98 3.3.2 Expanded Use Case

51 Expanded Use Case: Register Account

Use Case:

51 Register Account

Actors:

Guest

Description:

Guest user creates account to become a registered member of the bookstore. They enter their details and submit the form and the system verifies their details. Once the account is activated they get access to member only features like adding books to their wish list and placing orders, etc. 16

23 Typical course of event:

Actor Action

System Response

1. The system displays the registration page to the user.

2. The user navigates to the registration page. 15

3. The system asks the user to enter the required information on the form.

4. The user enters the needed information for registration then submits the registration form. 20

5. The system checks for the completeness of all required fields and validates the format of the entered data for e.g., email format, password strength, unique email.

6. If there are validation errors, the user corrects the information asked by the system then again submits the form.

7. The system stores the user data securely if all validations pass then sends successfully created account message to the user.

10. The user receives message.

Alternative course:

If, in line 2, the user decides not to proceed with registration, the case ends.

If, in line 4, the user decides not to provide their information or the user forgets to fill some fields or username/email already exists, the system displays message and asks the user to correct the information, the case ends if user does not follow requirement.

If, in line 6, the user does not submit the form, the case ends.

4 Expanded Use Case: Add Book To Cart

Use Case:

Add Book To Cart

Actors:

Customer

Description:

Only member can add books to their shopping cart and select quantities. The system updates stock availability and display users if the book is out of stock. They can update quantities, remove books, and view the total cost of the books. After finalizing the books, users move to checkout process. If a guest is trying to add items to the cart they are asked to log in first. 32

50 Typical course of event:

Actor Action

System Response

1. The system displays catalogue of the books.

15 2. Customer selects a book and clicks on Add to Cart button

3. The system adds the book to the cart and shows the updated total of the books.

101 4. Customer views the cart.

5. The system displays all selected books, quantities, and total cost of the books.

6. Customer updates quantities or removes books.

7. The system recalculates the total and updates stock availability.

8. Customer confirms the cart and moves to checkout or order the books.

9. The system prepares the order for finalization.

Alternative course:

If, in line 2, customer selected quantity exceeds stock, the system shows message, the case ends.

If, in line 6, the customer does not update the cart and leaves the page, the case ends without proceeding to checkout.

75

3.4 Class Diagram

A class diagram shows how classes are connected and depend on each other in UML. It defines the methods and variables inside an object, which represents something in a program. Class diagrams are used in object-oriented programming and have improved over time as programming methods changed. (Contributor, 2019)

65

Figure 78: Class Diagram

12

3.5 Data Dictionary

A data dictionary is a centralized collection of information that describes the structure, definitions, and attributes of data elements within a database or information system. It contains metadata-data about data-such as names, data types, sizes, relationships, constraints, and usage details of tables, columns, and other database objects.Invalid source specified.

3.6 System Architecture

This application uses the Monolithic ClientServer Architecture, it is built with the ASP.NET Core MVC framework which means everything (frontend, backend, and database). This follows a 3tier architecture dividing them into three important parts the Presentation Layer, Application Logic Layer, a Data LayerInvalid source specified..

Presentation Layer (Client Interface)

-The clientfacing interface that’s what the user see and interact and it is develop using the Razor views (.cshtml) and is rendered in the in server side with the ASPNET Core Razor engine. The layer is responsible for providing the user interactions such as browsing book, managing their accounts and add to cart, place order, and giving reviews. The interface is developed using HTML, CSS and JavaScript and It is made dynamic by using Razor syntax by pulling data from the server.

Application Logic Layer (ASP.NET Core MVC)

-The backend logic follows the ModelViewController (MVC) pattern:

Controllers handles the request from the server and manage the data flow between the user interface and the underlying data

Models represent core business entities such as Book, Member, Order etc.

Services contains the business logic such as user registration, login authentication, order processing , discount and admin task.

This layer is also responsible for session management, validation and routing. All HTTP request from the client are processed in this layer and the return the proper views or data responses.

Data Layer (MSSQL Database)

-The data layer uses Microsoft SQL server (MSSQL) to store the structured data. It is managed through the Entity framework core which is used make the work easy with the database using LINQ. All the key entities such as users, books, orders, reviews, inventory, and discounts are stored in this layer

Hosting Environment

-The application is currently hosted locally (local host) with the help of kestrel web server which is the part of the .Net Core development environment. This is perfect for development and testing phase, it enables the debugging without the need for external deployment. The application could be also hosted using IIS or a cloud platform.

Architecture Diagram

MVC Architecture

Monolithic ClientServer Architecture

Architectural Characteristics

83 Monolithic Deployment: All part of the app including frontend, backend and database access together into one project and hosted together.

ClientServer Model: The browser sends HTTP requests to the server which processes them and returns rendered HTML.

ServerSide Rendering: It generates the content on the server making it first load faster and improve SEO.

MVC Pattern:The application is divided into 3 parts (Model, View, Controller) which makes the code easier to maintain and test.

Local Development Hosting: It’s hosted locally for easy testing and developmentInvalid source specified..

Technology Stack Overview

Layer

Technology

Frontend (UI)

Razor Views (.cshtml), HTML/ CSS/ JavaScript

Web Framework

ASP.NET Core MVC

Programming Language

C#

ORM/Data Access

Entity Framework Core / ADO.NET

Database

Microsoft SQL Server (MSSQL)

Hosting

Localhost via Kestrel

This architecture is suited for the smaller project like academic or the prototype work, since it keeps everything in one place and makes testing and development straight forward.

3.7 Flowchart

45 A flowchart is a graphical diagram that visually represents the sequence of steps, actions, or decisions involved in a process or workflow. It uses standardized symbols connected by arrows to depict the flow and order of operations from start to finish, making complex processes easier to understand, analyze, and communicate.

Overall Flowchart

Figure 79: Flowchart

Login Flowchart

Figure 80: Login Flowchart

Register Flowchart

Figure 81: Register Flowchart

Browse Catalogue and Book details

Figure 82: Browse Catalogue and Book details

9 Add to cart

Figure 83: Add to cart

Add to Wishlist

Figure 84: Add to Wishlist

Member Places Order

Figure 85: Member Places Order

Staff process order

Figure 86: Staff process order

Write a review

Figure 87: Write a review

Manage Book Catalogue (Admin)

Figure 88: Manage Book Catalogue (Admin)

Announcement (Admin)

Figure 89: Announcement (Admin)

Discount Logic

Figure 90: Discount Logic

3.8 Method description

API Endpoints

Home Controller

Index Method

10 HTTP Method: GET

Endpoint: /Home/Index

Description: Displays the homepage (requires authentication)

Privacy Method

10 HTTP Method: GET

Endpoint: /Home/Privacy

Description: Shows the privacy policy page

Errors Method

HTTP Method: GET

Endpoint: /Home/Error

Description: Displays error details on exceptions

User Controller

Index Method

HTTP Method: GET

Endpoint: /User/Index

Description: Lists all users and their roles

Create Method

HTTP Method: GET

Endpoint: /User/Create

Description: Displays the user creation form

CreateAsync Method

HTTP Method: POST

Endpoint: /User/CreateAsync

Description: Submits new user data

Edit Method

HTTP Method: GET

Endpoint: /User/Edit/{id}

Description: Displays edit form for a user

OnPostAsync Method

HTTP Method: POST

Endpoint: /User/OnPostAsync

Description: Updates existing user information

Delete Method

HTTP Method: GET/POST

Endpoint: /User/Delete/{id}

Description: Deletes a user account

RoleManager Controller

Index Method

HTTP Method: GET

Endpoint: /RoleManager/Index

Description: Lists all defined roles in the system

AddRole Method

1 HTTP Method: POST

Endpoint: /RoleManager/AddRole

Description: Adds a new role

Books Controller

Index Method

1 HTTP Method: GET

Endpoint: /Books/Index

Description: Lists all books with filtering and sorting

Details Method

1 HTTP Method: GET

Endpoint: /Books/Details/{id}

Description: Displays details of a selected book

Create Method

10 HTTP Method: GET/POST

Endpoint: /Books/Create

Description: Displays and submits the new book form

Edit Method

1 HTTP Method: GET/POST

Endpoint: /Books/Edit/{id}

Description: Displays and updates the book edit form

Delete Method

1 HTTP Method: GET/POST

Endpoint: /Books/Delete/{id}

Description: Deletes a selected book

AnnouncementController Methods

Index Methods

1 HTTP Method: GET

Endpoint: /Announcements

Description: Shows details of a specific announcement

Details Methods

1 HTTP Method: GET

Endpoint: /Announcements/Details/{id}

Description: Shows details of a specific announcement

Create Methods (GET)

HTTP Method: GET

Endpoint: /Announcements/Create

Description: Displays the form to create a new announcement

Create Methods (POST)

HTTP Method: POST

Endpoint: /Announcements/Create

Description: Processes the form submission to create a new announcement

Edit Methods (GET)

HTTP Method: GET

Endpoint: /Announcements/Edit{id}

Description: Displays the form to edit an existing announcement

Edit Methods (POST)

HTTP Method: POST

Endpoint: /Announcements/Edit/{id}

Description: Processes the form submission to update an announcement

Delete Methods (GET)

HTTP Method: GET

Endpoint: /Announcements/Delete/{id}

Description: Displays the confirmation page to delete an announcement

Delete Methods (POST)

HTTP Method: POST

Endpoint: /Announcements/Delete /{id}

Description: Processes the deletion of an announcement

BookmarksController Methods

Index Method

1 HTTP Method: GET

Endpoint:/Bookmarks

Description: List all bookmarks for the current user

Details Method

1 HTTP Method: GET

Endpoint:/Bookmarks/Details/{id}

Description: Shows details of a specific bookmark

Create Method (GET)

HTTP Method: GET

Endpoint:/Bookmarks/Create or/Bookmarks/create/{id}

Description: Displays the form to create a new bookmark, optionally with a preselected

book

Create Method (Post)

1 HTTP Method: GET

Endpoint:/Bookmarks/Details/{id}

Description: Shows details of a specific bookmark

Create Method(GET)

HTTP Method: GET

Endpoint:/Bookmarks/Create or /Bookmarks/Create/{id}

Description: Display the form to create a new bookmark, optionally with a preselected

book

Create Method (POST)

HTTP Method: GET

Endpoint:/Bookmarks/Create

Description: Processes the form submission to create a new bookmark

Edit Method (GET)

HTTP Method: GET

Endpoint:/Bookmarks/Edit/{id}

Description: Displays the form to edit an existing bookmark

Edit Method (POST)

HTTP Method: POST

Endpoint:/Bookmarks/Edit{id}

Description: Processes the from submission to update a bookmark

Edit Method (GET)

HTTP Method: GET

Endpoint:/Bookmarks/Delete/{id}

Description: Displays the confirmation page to delete a bookmark

DeleteConfirmed Method (POST)

1 HTTP Method: GET

Endpoint:/Bookmarks/Delete/{id}

Description: Processes the deletion of a bookmark

BookmarkExists Method

Private helper method

No endpoint (internal use only)

Description: Check if a bookmark with the specified ID exists

AddToWishlist Method

HTTP Method: POST

Endpoint:/Bookmarks/AddToWishlist/{id}

Description: AJAX endpoint to add a book to the user’s wishlist

BooksController Methods

Index Method

1 HTTP Method: GET

Endpoint:/Books

Description: List all books with optional filtering, sorting, and pagination

Details Method

1 HTTP Method: GET

Endpoint:/Books/Details/{id}

Description: Shows details of a specific book, including reviews and purchase status

Create Method (GET)

10 HTTP Method: GET

Endpoint:/Books/Create

10 Description: Displays the form to create a new book

Create Method (POST)

HTTP Method: POST

Endpoint:/Books/Create

Description: Processes the form submission to create a new book, including image

upload

Edit Method(GET)

57 HTTP Method: GET

Endpoint:/Books/Edit/{id}

Description: Displays the form to edit an existing book

Edit Method(POST)

1 HTTP Method: POST

Endpoint:/Books/Edit/{id}

Description: Processes the deletion of a book and its related entities(bookmarks, reviews, order items)

BookExists Method

Private helper method

No endpoint (internal use only)

Description: Check if a book with the specified ID exists

CartItemsController Methods

Index Method

1 HTTP Method: GET

Endpoint:/CartItems/Details/{id}

Description: Shows details of a specific cart item

Create Method (GET)

HTTP Method: GET

Endpoint:/CartItems/Create or / CatItems/Create/{id}

Description: Displays the form to create a new cart item, optionally with a preselected book

Create Method(POST)

HTTP Method: POST

Endpoint:/CartItems/Create

Description: Processes the form submission to create a new cart item with automatic discount calculation

Edit Method(GET)a

HTTP Method: GET

Endpoint:/CartItems/Edit/{id}

Description: Displays the form to edit an existing cart item

Edit Method (POST)

HTTP Method: GET

Endpoint:/CartItems/Edit/{id}

Description: Processes the form submission to update a cart item

Edit Method (POST)

HTTP Method: GET

Endpoint:/CartItems/Delete /{id}

Description: Displays the confirmation page to delete a cart item

Delete Method (GET)

HTTP Method: POS

Endpoint:/CartItems/Edit/{id}

Description: Processes the deletion of cart item

CartItemExists Method

Private helper method

No endpoint (internal use only)

Description: Checks of a cart items with a specified ID exists

AddToCart Method

HTTP Method: POST

Endpoint:/CartItems/AddToCart/{id}

82 Description: AJAX endpoint to add a book to the users cart or increment quality if already in cart

HomeController Methods

Index Method

HTTP Method: GET

Endpoint:/Home or/ (default route)

Description: Displays the home page of the application

Privacy Method

HTTP Method: GET

Endpoint:/Home/Privacy

Description: Displays the privacy policy page

Error Method

HTTP Method: GET

Endpoint: /Home/Error

Description: Displays error information when an exception occurs

OrderItemsController Methods

Index Method

HTTP Method: GET

Endpoint:/OrderItems

Description: List all order items with their associated books and orders

Details Method

HTTP Method: GET

Endpoint:/OrderItems /Create

Description: Displays the form to create a new order item

Create Method (POST)

HTTP Method: POST

Endpoint:/OrderItems/Edit/{id}

Description: Processes the form submission to update a order item

Delete Method (GET)

HTTP Method: GET

Endpoint:/OrderItems/Delete /{id}

Description: Displays the confirmation page to delete an order items

DeleteConfirmation Method (POST)

HTTP Method: POST

Endpoint:/OrderItems/Delete /{id}

Description: Processes the deletion of an order item

OrderItemExists Method

Private helper method

No endpoint (internal use only)

Description: Checks if an order item with the specified ID exists

OrdersController Methods

Index Method

HTTP Method: GET

Endpoint: /Orders

Description: Lists all orders for the current user

Details Method

1

HTTP Method: GET

Endpoint: /Orders/Details/{id}

Description: Shows details of a specific order including its order items

Create Method (GET)

HTTP Method: GET

Endpoint: /Orders/Create

Description: Displays the form to create a new order

Create Method (POST)

HTTP Method: POST

Endpoint: /Orders/Create

Description: Processes the form submission to create a new order

CartToOrder Method

HTTP Method: POST

Endpoint: /Orders/CartToOrder

Description: Converts all pending cart items to a new order with discount calculation

Edit Method (GET)

HTTP Method: GET

Endpoint: /Orders/Edit/{id}

Description: Displays the form to edit an existing order

Edit Method (POST)

HTTP Method: POST

Endpoint: /Orders/Edit/{id}

Description: Processes the form submission to update an order

Delete Method (GET)

HTTP Method: GET

Endpoint: /Orders/Delete/{id}

Description: Displays the confirmation page to delete an order

DeleteConfirmed Method (POST)

HTTP Method: POST

Endpoint: /Orders/Delete/{id}

Description: Processes the deletion of an order and its related order items

OrderExists Method

Private helper method

No endpoint (internal use only)

Description: Checks if an order with the specified ID exists

ReviewsController Methods

Index Method

HTTP Method: GET

Endpoint:/Reviews

Description: Lists all reviews created by the current user

Details Method

1 HTTP Method: GET

Endpoint:/Reviews/Details/{id}

Description: Shows details of a specific review

Create Method (GET)

1 HTTP Method: GET

1 Endpoint:/Reviews/Edits/{id}

Description: Displays the form to edit an existing reviews

Edit Method (POST)

HTTP Method: POST

Endpoint:/Reviews/Edit/{id}

Description: Processes the form submission to update a review

Delete Method (GET)

HTTP Method: GET

Endpoint:/Reviews/Delete /{id}

Description: Displays the confirmation page to delete a review

DeleteConfirmation Method (POST)

HTTP Method: POST

Endpoint:/Reviews/Delete/{id}

Description: Processes the deletion of a review

ReviewExists Method

Private helper method

No endpoint (internal use only)

Description: Check if review with the specified ID exists

RoleManagerController Methods

Index Method

1 HTTP Method: GET

Endpoint:/RoleManager

Description: List all roles in the application

AddRole Method

1 HTTP Method: POST

Endpoint:/RoleManager/AddRole

Description: Creates a new role with the specified name

Parameters: roleName (String)

UserController Methods

Index Method

1 HTTP Method: GET

Endpoint:/User

Description: List all users with their assigned roles

Delete Method (GET)

HTTP Method: GET

Endpoint:/User/Create

Description: Displays the form to create a new user

CreateAsync Method (POST)

HTTP Method: POST

Endpoint:/User/Create

Description: Processes the form submission to update a users information

Delete Method (GET)

HTTP Method: GET

Endpoint:/User /Delete /{id}

Description: Delete a user and redirects to the user list

CreateUser Method

Private helper method

No endpoint (internal use only)

Description: Creates a new instance of ApplicationUser

GetEmailStore Method

Private helper method

No endpoint (internal use only)

Description: Returns the email store interface for user management

GetUserRoles Method

Private helper method

No end points (internal use only)

Description: Retrieves the role assigned to a specific user

UserRolesController Methods

Index Method

HTTP Method: GET

Endpoint:/UserRoles

Description: List all users with their assigned roles

Manage Method (GET)

HTTP Method: GET

Endpoint:/UserRoles/Manage/{userId}

Description: Displays the form to manage roles for a specific user

Manage Method (POST)

HTTP Method: POST

Endpoint:/UserRoles/Manage/{userId}

Description: Displays the form to manage roles for a specific user

Manage Method (GET)

HTTP Method: POST

Endpoint:/UserRoles/Manage/{userId}

Description: Processes the form submission to update a user's roles

GetUserRoles Methods

Private helper method

No endpoint (internal use only)

Description: Retrieves the roles assigned to a specific user

3.9 VCS repository

Initialize private Git repository with an empty .NET Web project.

We created a private Git repository on GitHub and cloned it into our local environments. After creating a new ASP.NET Core MVC project via Visual Studio. We structured the application with CSHTML views and tied it to an MSSQL database. Having established the initial setup, we committed the alterations and pushed them into the repository. We also invited all six members of the group as co-operators to enable simple coordination, versioning, and contribution during development.

Figure 91: Private Git Repository

We created a Git repository to store our code for the Ghyal Bhaag Book Store project. Our team of six members (Dhanraj, Simran, Roshan, Achyut, Ranjan, and Sabin) organized our work using three main branches:

master:

This was our main branch where Simran and Dhanraj worked directly. Simran focused on bookmarking, cart functionality, and order management, while Dhanraj handled admin features and staff order processing.

achyut:

This branch was where Achyut and Sabin collaborated. Achyut worked on admin CRUD operations and real-time broadcasting, while Sabin developed the reviews, ratings, and discount systems.

Roshan:

This branch was managed by Roshan, who also integrated code from Ranjan. Roshan implemented user registration, login, and claim code handling, while Ranjan developed the homepage UI, category tabs, filters, and book listing features.

An interesting aspect of our collaboration was that Ranjan didn't make direct pushes to GitHub since he wasn't familiar with it. Instead, he and Roshan established an alternative workflow where Ranjan would develop his features locally and share his code with Roshan through an external drive. Roshan would then review the code, integrate it with his own work, and push the combined changes to their branch.

This arrangement, while unconventional, worked well for their part of the project. Roshan essentially acted as both a developer and an integrator for their branch, giving him a comprehensive understanding of all the code they were contributing.

We faced some challenges with merge conflicts when integrating changes from different branches, but we overcame these by improving our communication and holding more frequent integration meetings. Despite the unusual workflow with Ranjan's contributions, we successfully completed the project with all team members making valuable contributions.

When working on our Ghyal Bhaag Book Store project using Git, we encountered several challenges, some challenges we faced are given below:
Merge Conflicts: This was probably our biggest headache! When two people changed the same part of a file, Git couldn't automatically decide which change to keep. We had to manually resolve these conflicts by looking at both versions and deciding what to keep. This happened a lot when Roshan tried to merge Ranjan's UI changes with his own work.
Forgetting to Pull Before Working: Sometimes team members would start working without pulling the latest changes first. This led to conflicts that could have been avoided. We learned to always start our day with a `git pull` to get everyone's latest changes.

Committing to the Wrong Branch: A few times, someone would accidentally commit their work to the master branch instead of their development branch. We had to learn how to use `git reset` and `git checkout` to move those changes to the right branch.
Branching Strategy Confusion: At the beginning, we weren't clear about when to create new branches and when to merge them back. This led to some very long-lived branches that became difficult to merge later. We eventually established clearer rules about our branching strategy.
These challenges taught us a lot about how to use Git effectively as a team. Despite the issues, Git was essential for our project's success, allowing six people to work on different parts of the code simultaneously.

Figure 92: Resolving merge conflicts 1

Figure 93: : Resolving merge conflicts 2

Figure 94: Commit

3.10 Milestone Chart

Figure 95: Milestone Chart

19 3.11 Gantt Chart

A Gantt chart is a horizontal bar chart used in project management to visually represent a project's schedule over time. It displays a list of tasks or activities vertically on the left side and a timeline horizontally across the top. Each task is represented by a horizontal bar whose length corresponds to the task's duration, showing its start and end dates.Invalid source specified.

Figure 96: Gantt Chart

3.12 Tech Stack

Framework

36 ASP net Core:

ASP.NET Core is a modern, open-source web framework developed by Microsoft that is used to build web applications and APIs. It is designed to be fast, flexible, and cross-platform, meaning it can run on Windows, macOS, and Linux. ASP.NET Core is well suited for both small and large projects, offering high performance and strong security features. It supports modern web development tools and is cloud-ready, making it easy to deploy applications on platforms like Microsoft Azure. Being open source, it is freely available and continuously improved by a large community of developers. Overall, ASP.NET Core provides a reliable and efficient environment for building modern web applications. (Chauhan, 2024)

MVC

Model-View-Controller (MVC) is a widely used architectural design pattern that organizes an application into three interconnected components: Model, View, and Controller. This separation of concerns enhances modularity, maintainability, and scalability in software development, especially for web and GUI applications. (Otieno, 2020)

Table 16: MVC Table

Component

Responsibility

Model

Manages the application's data, logic, and rules. It represents the business layer, and it is responsible for retrieving, storing, and processing data.

View

Handles the presentation layer. It displays data provided by the model in a specific format and manages what the user sees and interacts with.

Controller

Acts as an intermediary between the Model and View. It processes user input, manipulates data using the Model, and updates the View accordingly

(Sheldon, 2023)

Why?

72 We have chosen ASP.NET Core with the MVC design pattern for our web application Ghyal Bhaag Online Book Store because it offered the structure, tools, and flexibility we needed for building a modern and efficient online bookstore. 24 ASP.NET Core provided us with a high-performance, secure, and cross-platform framework, which aligned well with our project goals. It also came with many built-in features that helped speed up our development process, such as routing, authentication, and database integration.

The MVC pattern supported a clean separation of concerns, which made our code easier to manage, test, and scale as the project grew. Since we were dealing with various components like books, customers, and orders, having a structured and organized approach was important for keeping everything clear and maintainable.

External Libraries

In our project, we have used several external libraries to extend the functionality of the default 99 ASP.NET Core MVC framework. The External libraries we have used till now are given below:

Microsoft.AspNetCore.Identity: This is a newer identity management system tailored for ASP.NET Core. It allows us to securely manage user accounts, enable login and registration features, and implement role-based access control.

Swashbuckle.AspNetCore: This Library is used to generate Swagger/OpenAI documentation for our web APIs. This library creates a user-friendly interface where developers or users can test API endpoints, view request and response formats, and understand how the API works from the browser. These libraries greatly enhanced the efficiency, maintainability, and professionalism of our application.

Database

29 Microsoft SQL Server is a relational database management system (RDBMS) developed by Microsoft. It enables applications to store, retrieve, and manage data efficiently using Structured Query Language (SQL), specifically its extended version called Transact-SQL (T-SQL). 69 SQL Server is designed to handle a wide range of data management tasks, including transaction processing, business intelligence, and data analytics. (Rahul Awati, 2024)

Why?

We have also selected MSSQL for the database because it is a reliable, scalable, and secure solution that integrates well with ASP.NET Core. MSSQL provided us with the ability to efficiently store, manage, and retrieve data for the bookstore, such as books, customers, and orders. Its powerful querying capabilities and support for complex transactions made it the ideal choice for ensuring data integrity and performance.

43 Overall, we chose this technology stack because it gave us a strong foundation for building a reliable, user-friendly, and scalable application, while also helping us learn and follow industry-standard development practices.

4. Testing

3 Testing in software development is the process of evaluating a system or its
79 components to verify whether it meets specified requirements and functions correctly.
3 It involves executing the software to identify any gaps, errors, or missing requirements compared to the actual requirements, ensuring the software behaves as expected before it goes live.Invalid source specified.

70 For the Ghyal Bhaag online bookstore, we applied various testing methods such as unit testing, regression testing, white-box testing etc, to ensure reliability and accuracy. The corresponding test cases are outlined below:

View Book Catalog

Table 1: View Book Catalog

Objective

To check whether user can browse the paginated book catalog.

Action

9 Navigate to the book catalog page and scroll through the paginated list.

Expected Output

Paginated list of available books should be displayed with navigation control for next and previous pages.

Actual Output

Paginated list of books is displayed with the functional controls.

Test Result

Successful

Figure 97: Paginated page 1

Figure 98: Paginated page2

View Book details

Table 2: View Book Details

Objective

To check whether user can view detailed information of selected book.

Action

Click book title from the book catalog.

Expected Output

Detailed information of selected books is shown on the book details page.

Actual Output

Should display all book details from selected book.

Test Result

Test was successful

Figure 99: Book details page

97 Search book by title

Table 3: Search book by title

Objective

93 To check whether users can search books by title.

Action

89 Enter the book title in search bar and press search button.

Expected Output

A list of books matching search title should be displayed.

Actual Output

Books matching search title are displayed.

Test Result

Successful

Figure 100: Search book by book title working successfully

Filter books by Genre

Table 4: Filter book by genere

Objective

To check whether user can filter book by genre.

Action

Select a genre from dropdown and apply the filter.

Expected Output

Books from selected genre should be displayed.

Actual Output

Displays only books from the selected genre.

Test Result

Successful

Figure 101: Filter books by genre working successfully

Sort Books by Price.

Table 5: Sort books by price

Objective

To check whether users can sort books by price using minimum and maximum price inputs.

Action

Enter a minimum and maximum price range on the catalog page and apply the filter.

Expected Output

74 Books are displayed within the specified range in ascending order of price.

Actual Output

40 Books were successfully displayed in ascending order of price within the selected range.

Test Result

Test cases was successful

Figure 102: sort books by price working successfully

Register member with valid details

Table 6: Register member with valid details

Objective

To check whether user can register as member with valid details.

Action

Fill the valid information like email, password and enter register button.

Expected Output

6 Members account should be created successfully and confirmation message should be displayed.

Actual Output

Account is created and message is shown.

Test Result

Test was successful.

Figure 103: registering a new user

Figure 104: Confrim mail message

Figure 105: Use details successfully added in database

Register member with existing email

Table 7: Register member with existing email

Objective

To check whether the system prevents registration with an existing email.

Action

38 Fill the details that are already registered and click register button.

Expected Output

38 Error message should show that should say email is already in use.

Actual Output

Error message saying email already registered is displayed.

Test Result

Test Case was Successful

Figure 106: Email already exists error message is shown

Register Password validation:

Table 8: Register password validation

Objective

To check if the system shows errors when the password is too weak during sign-up.

Action

Try signing up with a password that:

Doesn't have a symbol like @ or #

Doesn't have a number

Doesn't have a capital letter (A–Z)

Expected Output

The system should show messages like:

"Password must have at least one special symbol (like @)"

"Password must have at least one number"

"Password must have at least one capital letter"

Actual Output

The system showed all the right error messages when we tried using weak passwords.

Test Result

Test was successful.

Figure 107: Password validation working successfully

Register email validation:

Table 9: Register email validation:

Objective

6 To check if the system shows an error when an invalid email is entered during sign-up.

Action

A guest user try signing up using emails like:

usergmail.com (missing @)

user@.com (missing domain)

@gmail.com (missing name).

Expected Output

80 The system should show a message like: "Please enter a valid email address."

Actual Output

The system showed the correct error message for all invalid emails we tried.

Test Result

Test case was successful

Figure 108: Email validation working

Add book to whitelist

Table 10; Add book to whilst

Objective

To check whether members can bookmark books to their whitelist.

Action

61 Log in as a member navigating to book details page and click add to whitelist button.

Expected Output

Book should be added to members bookmark list and confirmation message should be shown.

Actual Output

Book appears in a member's whitelist with a book added message.

Test Result

Successful

Figure 109: Adding book items to whitelist

Figure 110: Added book shown in whitelist page

Figure 111: Book successfully added in whitelist database

34 Add book to cart

Table 11: add to book to cart

Objective

To check whether members can add books to their shopping cart.

Action

Login a member navigating to books details page and click add to cart button.

Expected Output

Book should be added to the cart and the cart icon should reflect the updated item count.

Actual Output

Book is added to the cart reflecting the updated item count.

Test Result

Successful

Figure 112: Before adding book to add to cart

Figure 113: After adding book to add to cart

Figure 114: Adding others books in add to cart too after removing the previous one

Remove book from Cart

Table 12: Remove book from cart

Objective

5 To click whether members can remove book from their shopping cart.

Action

5 Click remove button next to a book.

Expected Output

Book should successfully be removed from the cart.

Actual Output

Book is removed and cart reflects updated contents.

Test Result

Successful

Figure 115: Removing book from cart

31 Figure 116: Before removing book from add to cart

Figure 117: After removing book from add to cart

Place order with valid cart

Table 13: Place order with valid cart

Objective

To check whether members can place an order with a valid cart.

Action

5 Proceed to checkout by adding books to the cart and click place order button.

Expected Output

48 Order should be created and confirmation email should be sent to the registered

member of email.

48 Actual Output

Order is placed and confirmation email is received.

Test Result

Successful

Figure 118: Add to cart to checkout

Figure 119: Orders Page

Cancel order before Processing

Table 14: Cancel order before processing

Objective

To check whether members can cancel an order before it is processed.

Action

Navigate to order history page and click cancel button for pending order.

Expected Output

Order status should be update to cancelled and confirmation message is shown.

Actual Output

Order is marked as cancelled by displaying a confirmation message.

Test Result

Successful

Figure 120: Cancel order before processing

Figure 121: cancelation successfully

Member review purchased book

Table 15: Member review purchased book

Objective

To check whether members can submit review for purchased books.

Action

Enter review from details page of purchased book and click submit review button.

Expected Output

Review should be submitted and visible under the book’s details page.

Actual Output

Review appears under the book with a review submitted message.

Test Result

Successful

Figure 122: adding review in purchased book

Figure 123: Review was successfully added and displayed under the book ratings & review section

Apply 5% discount on 5+ books

Table 16: Apply 5% discount on 1+books

Objective

To check if a member makes an order of more than 5 books then 5% discount should be applied to their total amount.

Action

Member adds 6 books in the cart and the system displays the sub amount, discount amount and total payable amount.

Expected Output

When a member adds more than 6 books, system displayed the sub amount (the total before any discount). Since the member ordered more than 5 books, a 5% discount was automatically applied. The total payable amount after applying the discount should be updated accordingly.

Actual Output

The system displayed the sub amount (the total before any discount). Since the member ordered more than 5 books, a 5% discount was automatically applied. The total payable amount after applying the discount was updated accordingly.

Test Result

Test was successful

Figure 124: Only one book item added in my cart

Figure 125: More than 5 books were added and the discount amount was also shown successfully

Apply 10% Stackable discount after 10 orders

Table 17: Apply 10% stackable discount after 10 orders

Objective

To check whether a10% stackable discount is applied after 10 successful orders.

Action

Add books to the cart after placing 10 successful orders then proceed to checkout for the 11th order.

Expected Output

A 10% discount should be applied on the total bill for 11th order.

Actual Output

10% discount is applied showing the discounted price for 11th order.

Test Result

Test was sucessfull

Figure 126: 10 Order successfully completed

Figure 127: Stackable discount applied successfully

Send confirmation email with claim code

Table 18: Send confirmation email with claim code

Objective

To check whether a confirmation mail with a claim code is sent after placing an order.

Action

Click place order button after proceeding to checkout after adding books to cart.

Expected Output

An email with a claim code and bill details should be sent to the members registered email.

Actual Output

Email with a claim code and bill details is received.

Test Result

Successful

Figure 128: Claim code received in mail

Validate claim code at store

Table 19: Validate claim code at store

Objective

To check whether staff can validate a members claim to fulfill the order.

Action

Go to order fulfillment page and enter member claim code then click validate button.

Expected Output

6 Claim code should be verified and order status should be updated to completed.

Actual Output

6 Claim code is verified and order status is updated to completed.

Test Result

Successful

Figure 129: Entering claim code by staff

Figure 130: status updated to pending

37 Admin can add new book to a catalog

Table 20: Admin add new book to a catalog

Objective

37 To check whether an admin can add a new book to catalog.

Action

100 Go to book management page, add a new book by filling all the book details.

Expected Output

28 The new book should be added and should be visible in the catalog.

Actual Output

28 The new book was successfully added and displayed in the catalog.

Test Result

Test was successful.

Figure 131: Adding new book named pride & prejudice

Figure 132: Book Catlog page before adding the new book

73

Figure 133: Book catlog page after adding the new book and the book was successfully displayed

56

Figure 134: Book table before adding new book

Figure 135: Book table after adding a new book

Admin update book details

Table 21: Admin update book details

Objective

To check whether and admin can update the details of and existing book in the catalog.

Action

Select a book and edit its details.

Expected Output

The books information should be updated and shown in the catalog.

Actual Output

The new details of book is updated in the catalog.

Test Result

Successful

Figure 136: Before editing price of a book

Figure 137: Changing the price of book

Figure 138: After changing the price of the book

Admin delete book from catalog

Table 22: Admin delete book from catalog

Objective

To check whether an admin can delete a book from catalog.

Action

Navigate to an book page and clicks on to the click delete button.

Expected Output

59 The selected book should be removed from the catalog and from the database as well.

Actual Output

The book is no longer available in the catalog and database.

Test Result

Test was successful.

Figure 139: Selecting the book

Figure 140: Deleting the book

Figure 141: Book CatLog page before deleting book

Figure 142: Book CatLog page after successfully deleted.

Figure 143: Before deleting the book

Figure 144: After deleting the book

Admin Announcement Creation and Display

Table 23:Admin Announcement Creation and Display

Objective

To check if the admin can create a timed announcement and if it shows correctly to users.

Action

Admin creates a new announcement titled and sets the display.

Expected Output

The announcement should be visible to users on the homepage during the selected period.

Actual Output

The announcement appeared correctly on the homepage and disappeared after the end date.

Test Result

Test was successful.

Figure 145: Creating announcemnt

Figure 146: Announcement added in announcement page

Figure 147: Announcement displayed in homepage

5. Individual reflections and reflections

Member(s)

Assigned Features

Main Focus

Documentation Responsibilities

Roshan Gurung

User registration, login, email confirmation, claim code handling, error handling, bug fixing, backend refinement

Backend-heavy+ Minimal frontend+ Logic

Authentication flow, API endpoints, use case diagram, ERD, flowcharts, data validation design, VCS, Database dictionary, error handling, bug fixing, backend refinement

Dhanraj Gurung

Admin CRUD (books, inventory), banner announcements, staff order handling, real-time broadcast, UI refinements, 5% and 10% discount mechanism

Backend + Frontend + Real-time

Admin UI wireframe, class diagram, staff portal design, flowchart, testing, prototype, VCS, Introduction & Conclusion, User manual, UI refinements

Achyut Tamang

Admin CRUD (books, inventory), 5% and 10% discount mechanism, banner announcements, staff order handling, real-time broadcast, UI refinements

Backend + Frontend + Real-time

Admin UI wireframe, class diagram, Method description, staff portal design, flowchart, testing, VCS, Introduction & Conclusion, UI refinements, Discount calculation logic

Simran Shrestha

Bookmarking, adding to cart, placing & cancelling orders, UI refinements

Minimal backend+ Frontend-heavy + Logic

flowcharts, UI guides with labels/arrows, high level and expanded use case, testing for cart/order functionalities, Prototype, flowcharts, system architecture description, User manual, UI refinements

Ranjan Bahadur Thapa Chhetri

Homepage UI, category tabs, filters, sorting, paginated book listing

Backend + Frontend+ DB Rules

wireframe, data dictionary, catalog API descriptions, technology stack description, testing, milestone chart

Sabin Das Pariyar

Reviews, ratings, paginated book listing

Backend + Frontend

UML, validation logic testing, screenshots of review features, wireframes, gantt chart

Individual Reflection

Roshan Gurung

I was mostly responsible for the backend in this project as I overviewed and guided the development. My major contributions were in authentication, role assignment and bug fixing. I had to implement user registration, login, email verification as well as a claim-code generation for secure order processing. This involved setting up password hashing and configuring an SMTP service for sending confirmation emails. Utilizing secure coding methods (hashing of passwords) and extensive testing. I fixed major bugs and helped out my teammates in their development journey. I also worked with team members to connect their frontend components to the backend services, to facilitate data flow through the application.

Further to the authentication level I contributed in adding a bunch of interaction user-friendly pages. I organized the backend code into modules with middleware and purpose-built controller, thereby enhancing its maintainability and understandability. To help the team understand our data and workflows, I created ER diagrams and flowcharts, meticulously documented backend methods. By making regular Git commits I ended up with a very clear reflection of the history of the work I completed which also allowed me to easily collaborate with the other people in my team. In general, I grew a ton from this project in terms of backend development: I was much more efficient at debugging, understood how to best design a secure system, and tackled a complex feature without feeling overwhelmed. These lessons, particularly those involving secure authentication and error recovery will be pivotal in influencing my future experiences in software engineering.

81

I also faced many problems mainly with database migrations and foreign key conflicts as I worked mainly in backend. Also, the fact that one of our team members was always ill made work distribution challenging as all of his part of work fell into mine and my work became complicated as I had to continue someone else’s work. But I take it as a learning opportunity as such cases could arise in professional environments as well and experience with handling problems is key to survive in any industry.

Simran Shrestha

On this project I worked almost exclusively on frontend features like bookmarking, adding to cart, placing and cancelling orders, UI enhancements etc with only small backend impact. I was responsible for the seamless user experience, refining the logic flow for order placement and the ease to use platform.

Documentation: I developed a logic flowchart, UI design guides interspersed with labels/arrows., high levels and expanded use-cases and tested operations for cart and order to verify smooth operations. I have also participated to the prototyping of the system, and to the system architecture descriptions and user manual, which has helped to keep this software clear and organized.

The most difficult part was team working, specially in the Git. Because I was from another section, it was hard to work with my team. It took awhile for me to wrap my head around Git workflows and how everything integrated into a project, so I primarily worked in my group mates laptop in order to stay in sync. The other challenging aspect was time management, which ultimately was made possible by effective communication, Git commits and teamwork to guarantee the project’s timely execution.

Despite these challenges, I learned a lot, including mastering frontend logic designing, improving user experiences and knowledge about system architecture how it interconnects different aspects. Studying and adapting to code collaboration using Git, and working through real-world project struggles improved my coordination skills, flexibility, and problem-solving skills which will be very useful to transfer over to any future work.

Achyut Tamang

In our Ghyal Bhaag Book Store project, I mainly worked on the admin side of things. My job was to create all the CRUD operations for books and inventory, which means I had to make sure admins could add new books, see all the books, update book information, and delete books when needed. I also had to make the banner announcement system where admins can post messages about deals and new arrivals.

58

One of the hardest parts was making the real-time broadcast feature work. Whenever a customer successfully ordered books, I needed to show a notification to all admins right away. I had never done this before, so I had to learn how to make messages appear instantly without refreshing the page. After trying many different approaches and lots of testing, I finally got it working! My teammate Dhanraj helped me a lot with this part since we were both working on the same stuff at the same time. When I got stuck with the frontend part of the notifications, Dhanraj would help me figure it out, and when he had questions about the backend logic, I would explain my approach to him.

I also worked on the discount system, making sure customers got 5% off when they ordered 5 or more books, and loyal customers got an extra 10% discount after 10 successful orders. This was tricky because I had to keep track of how many orders each customer had made and calculate the discounts correctly. Dhanraj and I spent many late nights debugging this feature together, taking turns to test different scenarios to make sure it worked perfectly.

For the documentation part, I created wireframes for the admin pages, made diagrams to show how different parts of our code connected, and drew flowcharts to explain how the staff would process orders. These documents helped our whole team understand how everything fit together.

This project taught me so much about backend development and making websites interactive. I'm now much better at designing systems and writing clean code. I also learned how important it is to work closely with teammates - Dhanraj and I accomplished much more by collaborating than we could have done individually. These skills will definitely help me in my future career as a software developer.

Dhanraj Gurung

For our Ghyal Bhaag Book Store project, I focused on creating the inventory management system and staff portal. I had to build interfaces that let staff keep track of book quantities, update stock levels, and manage the store inventory. This meant I had to design database tables to store all this information and create ways for the system to access and update this data.

The most challenging part for me was building the staff order handling portal. I needed to make a system where staff could easily process customer orders using claim codes, check member IDs, and update order statuses. I wanted to make it secure but also easy to use. I solved this by creating a step-by-step process with clear instructions at each stage. My teammate Achyut was super helpful during this process - whenever I ran into problems with the backend logic, he would help me debug since we were working on similar stuff at the same time. We often stayed after class to solve problems together.

Another challenge was building the timed discount system with "On Sale" flags. I handled the UI while Achyut focused on backend logic, ensuring discounts applied and expired automatically. We also teamed up on the real-time broadcasting feature. Achyut worked on instant server messaging, and I handled the user-side notifications. Our close collaboration made both features smoother and more effective

I also created detailed user manuals with screenshots and step-by-step instructions to help future users understand how to use the admin features. Achyut and I would review each other's documentation to make sure everything was clear and complete. This project really improved my skills in making websites, especially in creating admin interfaces that are powerful but still easy to use. I learned a lot about making websites interactive and handling different types of users. Working closely with Achyut taught me the importance of good communication and teamwork. These lessons will be super valuable as I continue learning about programming, especially when building complex systems with different user types.

Sabin Das Pariyar

I had some important responsibility in this Ghayalbaag project to add the rating and review facilities. I added backend functionality as well as frontend UI to these features. My tasks were coding database functionality for reviews and ratings, and adding the calculation of 5%/10% discount in checkout. I also created UML diagrams to design these features and wrote validation tests to ensure the discount logic was correct.

Speaking briefly, my work consisted of:

Developing review and rating functionality, including UI controls and backend data handling.

Creating and implementing the 5% and 10% discount functionality with adequate validation logic.

Creating UML diagrams and conducting tests to ensure the discount calculations.

Completion of these tasks was problematic for me as C# was my weakest programming language. I spent some time learning C# syntax and.NET framework specifications from the beginning. On top of this, communication with my team members was problematic as we were in different sections. Scheduling meetings was problematic, and sometimes I was out of the loop in what the team had accomplished.

I am also a professional bass guitarist and was touring in music during the project duration. Between live concerts, recording, rehearsals, college classes, and family time, my agenda was very busy. Booking all these activities taxed my time management skills to their limits. Despite these challenges, I was able to keep up. I communicated with the team constantly through messages whenever I could not attend meetings. My team members were very cooperative: they helped me debug and assist me in interpreting the challenging parts of C# when I was confused. I'd often stay late or utilize spare breaks to read through the C# material they pointed me to. Over time, I could feel improvement in my understanding. For example, I discovered how to use Git for version control by committing regularly whenever I was modifying the review feature or discount logic. Experimenting with Git and gathering opinions from my co-workers helped me speed up quickly. Through this effort and the cooperation of the team, I managed to implement the discount mechanism within the system successfully and finish the review/rating module. From this experience, I gained a lot of experience and new knowledge. Now I am more comfortable with C# and.NET programming. I also gained experience in Git workflow like branching and merge strategy. Most importantly, I gained experience in how to combine the backend logic with the frontend interface in a real application. I practiced validating business logic by writing tests for the discount rules (verifying a 5% discount is applied properly under the correct conditions). Most importantly, I proved to myself that I can learn quickly under pressure and make useful contributions even when working with extremely experienced developers. Overall, this project helped me grow in the following ways:

Technical Skills: Boosted confidence in C#/.NET programming, learned how to use Git efficiently, and grasped full-stack integration (connecting UI forms to backend code).

Project Work: Mastered UML design, code validation tests, and documenting workflows.

Personal Growth: Developed better time management and collaboration skills. I learned how to seek help, pick things up quickly, and remain organized.

52 Overall, I am pleased with what I have achieved and how far I got. The experience was frustrating at times, but I ended up with a much stronger set of skills and the feeling that I am able to deliver challenging coding work in future projects.

Ranjan Bahadur Thapa Chhetri

In this project, my main focus was on developing the features that are visible to the users in our book catalog system, specifically the Homepage UI, category tabs, filters, sorting and paginated book listing. My responsibilities in this project went from developing front end using basic HTML, CSS to create responsive and intuitive interface, backend to support dynamic data retrieval and database rules to ensure efficient querying. I also contributed in documentation part by producing wireframes, data dictionary, catalog API description, technology stack overview, test cases and milestone chart to guided our project timeline.

My main contribution in this project was to design a user-friendly homepage with category tabs and filters which enabled users to browse books efficiently. Optimizing backend queries were required while implementing paginated book listings to handle large datasets. I did it using SQL indexing and caching. On the frontend, I used basic HTML, CSS to ensure smooth sorting and filtering interactions. I faced challenges while integrating features with backend APIs mainly when aligning the paginated data with real-time updates. My documentation efforts like wireframes, API description helped to create a clear communication across the team members while the milestone chart helped us to stay on track.

This project helps me understand full-stack development and the importance of user-centric designs. I also faced challenges while optimizing query performance and managing documentation deadlines along with the coding tasks. I overcame these challenges by adopting systematic debugging and prioritizing task effectively. I got a chance to boost up my knowledge and skills in HTML, CSS, SQL and API integration from this project. I also got chance to improve my ability to create clear documentation. These challenges pushed me to grow as a developer by balancing technical and organizational responsibilities. I became more confident in designing efficient systems while collaborating across subsystems. Moving forward, I will prioritize early performance testing and simple documentation to enhance project efficiency. Reflecting on my performance on this project I am proud of delivering a polished UI and good documentation whereas, I need to improve my time allocation for testing. After working in this project, I have gained an experience that has equipped me with versatile skills for future full-stack projects.

Group Reflection

The project of Ghyal Bhaag Books became an amazing learning experience, which finetuned our skills to the best. Every team member has made their own delightful contributions; one behind the backend, other in frontend design, one in inventory, and the last one in the user experience. Roshan led towards the backend side, looking to it that secure authentication and role assignment were implemented with bug fixing, also providing ER diagrams and flowcharts of great value. Secure coding practices were constantly at the front so that an application never turns vulnerable and exposed to all kinds of users' information. Simran's attention was more concerned on frontend features by enhancing the interface and order logic to provide a very seamless and intuitive platform. She did this while surmounting issues with Git workflows and team coordination through persistent communication, underlining the role of supporting other team members in the distributed teamwork environment. Achyut and Dhanraj had gradually worked on the admin and inventory management systems together in a ticked manner. Their collaboration was so strong that they implemented realtime notifications, discount functionality, and even more complex fiddly stuff effectively, crunching the most challenging features together at night, which was used as debugging time.

In a Nutshell, Sabin juggled his music career, which was already well into professionalism, and initial difficulties with C# and .NET programming to provide rating and review functionality. If his drive and adaptability under pressure to deliver the expected results in an unknown territory for him, while actively seeking and applying every assistance available gives credit to the capturing moments of resilience with a growth mindset. Our team had to face one extra challenge due to illness of a teammate, thereby redistributing the responsibilities and making the workload complex for a few individuals; this situation brought flexibility, better communication, and a reason to strengthen support for one another. It's only through this project that relevant tradesecure coding, system architectural designing, and implementation of some more complex features have thrived out. Collectively, all these experiences from this and many more have, and will remain to be, the right dwelling blocks for facing up upcoming challenges in software development, again working within a team and be able to pride in a robust system developed in the past despite challenging constraints. These lessons have a lasting influence far beyond this project, as it plants a continuous growth mindset of skill enhancement and refinements in software development.

6. Conclusion

So, we finally finished making the online book ordering system for Ghyal Bhaag Book Store! It was a really long journey with lots of ups and downs, but our team of six members managed to create a working C#.NET application that helps people buy books online instead of having to go to the store just to see what's available.

Our project was about making an online platform for Ghyal Bhaag Book Store where customers can look through books, search for what they want, and order them. We also wanted to help the store keep track of their books and orders better. Looking back at what we wanted to do, we actually managed to add all the important features like letting people make accounts, browse through books page by page, search and filter books in different ways, save books they like for later, add books to their cart, place orders they can cancel if needed, write reviews, get discounts, and let the store staff and admins manage everything.

One of the hardest parts was making the real-time order broadcasting work. We had to figure out how to show notifications right away when someone successfully orders books without making the website slow. The email system with claim codes was also tricky because we had to make sure the emails actually got sent and didn't end up in spam folders.

Our team faced a bunch of challenges while working on this. We had a lot of GitHub conflicts because we were all working on different parts at the same time, so we had to learn better ways to use branches and talk to each other more. Time management was really hard because we were also working on our final year projects at the same time. After we finished our FYPs, we dedicated all our time to this project, often staying up all night to meet deadlines.

Some of our team members got sick during the project, so others had to take over their work temporarily. Even with these problems, everyone stayed committed to finishing the project, and we changed how we worked to deal with these unexpected issues.

In the absence of some teammates during important meetings, other teammates were always there to clear doubts with our module teacher, Sanam Katula. We would like to thank Sanam Katula for all the help and guidance throughout this project. Whenever we got stuck or had questions about how to implement something, he was always willing to point us in the right direction without just giving us the answers.

We decided to use Microsoft SQL Server for our database even though most of us hadn't used it much before. Setting up the database, connecting it to our application, and writing efficient queries was really challenging. We spent hours watching YouTube tutorials and reading documentation to figure it out, but we eventually got it working properly.

Making the timed discount system with "On Sale" flags was harder than we expected. We had to be careful about how we handled dates to make sure discounts appeared and disappeared at the right times. The loyalty discount system was also complicated because we had to keep track of how many orders each member had made and apply stackable discounts correctly.

If we could keep working on this project, we'd probably add home delivery options, make a mobile app version, add payment methods like credit cards and digital wallets, 76 and create a recommendation system that suggests books based on what users have bought before.

Even though we faced a lot of challenges, our team successfully delivered a complete online book ordering system that meets all the requirements for Ghyal Bhaag Book Store. This project not only accomplished what it was supposed to do but also taught us a lot about working in teams, managing our time, solving problems, and using new technologies. The things we learned from this project will definitely help all of us in our future careers.

References

3 Brush, K. (2022). techtarget. Retrieved May 13, 2025, from https://www.techtarget.com/searchsoftwarequality/definition/use-case

Chauhan, S. (2024). scholarhat. Retrieved April 17, 2025, from

24 https://www.scholarhat.com/tutorial/aspnet/introduction-to-aspnet-core

33 Contributor, T. (2019). techtarget. Retrieved May 13, 2025, from https://www.techtarget.com/searchapparchitecture/definition/class-diagram

64 Ivan Belcic, C. S. (2024). ibm. Retrieved May 13, 2025, from https://www.ibm.com/think/topics/entity-relationship-diagram

68 Otieno, C. (2020). hackernoon. Retrieved April 17, 2025, from https://hackernoon.com/understanding-the-model-view-controller-design-pattern-p3d6258k

102 Rahul Awati, A. H. (2024). techtarget. Retrieved April 17, 2025, from https://www.techtarget.com/searchdatamanagement/definition/SQL-Server?utm_source

Sheldon, R. (2023). techtarget. Retrieved April 17, 2025, from

63 https://www.techtarget.com/whatis/definition/model-view-controller-MVC