

**M.C.A  
Thesis**

**SMART TUTION WEBSITE**

**SMART TUTION WEBSITE**

**Anuj Tripathi, Aayush  
Awasthi, Ramji Gupta**

**JUNE  
2022**

**SMART TUTION WEBSITE**

***A Thesis Submitted***  
**In Partial Fulfillment of the Requirements for**  
**the Degree of**

**MASTER OF COMPUTER APPLICATIONS**  
**in**  
**FIELD OF SPECIALIZATION**

**by**

**Anuj Tripathi**  
**(2000290140024)**  
**Aayush Awasthi**  
**(2000290140002)**  
**Ramji Gupta**  
**(2000290140098)**

**Under the Supervision of**  
**Dr. Amit Kumar**  
**KIET Group of Institutions**  
**Ghaziabad**



**to the**

**FACULTY OF COMPUTER APPLICATION**  
**DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY**  
**LUCKNOW**  
**JUNE 2022**

# **OLD BOOK STORE**

## **A PROJECT REPORT**

**Submitted By**

**ANUJ TRIPATHI**

**(2000290140024)**

**AAYUSH AWASTHI**

**(2000290140002)**

**RAMJI GUPTA**

**(2000290140098)**

**Submitted in partial fulfillment of the  
Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATIONS**

**Under the Supervision of  
Dr. Amit Kumar**



**Submitted to  
DEPARTMENT OF COMPUTER APPLICATIONS  
KIET Group of Institutions, Ghaziabad  
Uttar Pradesh-201206**

**(JUNE 2022)**

## **CERTIFICATE**

Certified that Anuj Tripathi(2000290140024), Aayush Awasthi(2000290140002), Ramji Gupta(2000290140098) have carried out the project work having “ OLD BOOK STORE ”for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Anuj Tripathi (University Roll No.2000290140024)  
Aayush Awasthi (University Roll No.2000290140002)  
Ramji Gupta (University Roll No.2000290140098)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Amit Kumar  
Associate Professor  
Department of Computer Applications  
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner

Signature of External Examiner

**Dr. Ajay Srivastava**  
**Head, Department of Computer**  
**Applications, KIET Group of Institutions**  
**Ghaziabad**

## **ABSTRACT**

The main objective of the project is to create an online book store that allows users to search and purchase a book online based on title, author and subject. The selected books are displayed in a tabular format and the user can order their books online through credit card payment. Using this Website the user can purchase a book online instead of going out to a book store and wasting time.

There are many online book stores like Powell's, Amazon which were designed using Html.

Online Book store is an online web application where the customer can purchase books online. Through a web browser the customers can search for a book by its title or author, later can add to the shopping cart and finally purchase using any type of transaction. The user can login using his account details or new customers can set up an account very quickly. They should give the details of their name, contact number and shipping address. The user can also give feedback to a book by giving ratings on a score of five. The books are divided into many categories based on subject like Software, Database, English, and Architecture etc.

The Online Book Store Website provides customers with online shopping through a web browser. A customer can, create, sign in to his account, place items into a shopping cart and purchase using any mode.

The Administrator will have additional functionalities when compared to the common user. He can add, delete and update the book details, book categories, member information and also confirm a placed order.

This application is developed using Django, HTML, CSS, Python, JavaScript programming language. The Master page sets, data grids, user controls are used to develop the Online Book store.

## **ACKNOWLEDGEMENTS**

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, Dr. Ajay Kumar for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Anuj Tripathi (University Roll No.2000290140024)

Aayush Awasthi (University Roll No.2000290140002)

Ramji Gupta (University Roll No.2000290140098)

# **Table of Contents**

<b>Certificate</b>	<b>2</b>
<b>Abstract</b>	<b>3</b>
<b>Acknowledgements</b>	<b>4</b>
<b>Table of Contents</b>	<b>5</b>
<b>List of Abbreviations</b>	<b>6</b>
<b>List of Tables</b>	<b>7</b>
<b>List of Figures</b>	<b>8</b>

## **Section - 1 Vision Document**

- 1. Introduction
  - 1.1 Purpose and Motivation
- 2. Project Overview
  - 2.1 Background
- 3. Requirement Specifications
  - 3.1 Main Requirements
  - 3.2 Critical Use Case Requirements
  - 3.3 DFD
  - 3.4 Environment1
  - 3.5 Screenshots

## **Section – 2 Architecture Design**

- 1. Introduction
- 2. Architecture
- 3. Presentation Tier
- 4. Middle Tier
- 5. Data Tier

**Coding:-**

Module Wise Code

## **Section – 3 Test Plan**

- 1. Test plan identifier
- 2. Introduction

3. Features to  
be tested

## Section – 4 Project Evaluation

Introduction  
Problems faced  
Metrics  
Lessons  
Learned

## Section – 5

## References

## LIST OF FIGURES

Figure No.	Page No.
2	12
3	13
4	14
5	15
6	21
7	22
8	23
9	24
10	25
11	27
12	28



# **Introduction**

The purpose of this document is to provide an architectural design for the Online Book Store. The design will show the presentation tier, the middle tier consisting of classes, sequence diagrams, and the data tier consisting of the database design diagram.

Customers can get their book delivered instead of actually going and buying the book. They can make payment online itself.

Managing of inventory in the shop for shopkeeper becomes easier as customers are not visiting and ordering online.

This system saves both time and travelling cost of customers.

User can get to know different kinds of books that they were unaware of by just searching in the system using keywords.

The only disadvantage is if the customer receives a book that is not in proper condition or has some kind of defect then there incurs an additional charge of posting it back.

# **Section.1 Vision Document**

## **1. Introduction**

### **1.1 Purpose and Motivation**

The main objective of the project is to create an online book store that allows users to search and purchase a book based on title, author and subject. The selected books are displayed in a tabular format and the user can order their books online through credit card payment. The Administrator will have additional functionalities when compared to the common user.

The motivation to create this project has many sources

- Interest to develop a good user friendly website with manyonline transactions using a database.
- To increase my knowledge horizon in technologies like Django, SQL, CSS, HTML.
- To gain good experience in Django before joining in a full time job.
- To gain expertise using Data Grid, Data Set, Data Table, Data Adapter and Data Readers.

## **2. Project Overview**

### **2.1 Background**

There are many online book stores like Powell's, Amazon which were designed using Html.

Online Book store is an online web application where the customer can purchase books online. Through a web browser the customers can search for a book by its title or author, later can add to the shopping cart and finally purchase using credit card transaction. The user can login using his account details or new customers can set up an account very quickly. They should give the details of their name, contact number and shipping address. The user can also give feedback to a book by giving ratings on a score of five. The books are divided into many categories based on subject Like Software, Database, English, Architecture etc.

This project has the following functionalities:

**1) A Home page with product catalog**

This is the page where the user will be navigated after a successful login. It will display all the book categories and will have a search keyword option to search for the required book. It also includes some special sections like recommended titles, weekly special books.

**2) Search**

A search by keyword option is provided to the user using a textbox .The keyword to be entered should be the book title.

**3) Advanced Search**

Advanced search helps the user to search for a book based on Title, Author, Category and price range. All the books which match the particular search criteria and their total count will be displayed .From here the user can select a book and add to the shopping cart.

**4) Book Description**

If the user would like to know details about a book he can click on the title from where he will be directed to a Book description page. It includes the notes on the book content and also a link to Amazon.com to get the book review.

**5) User Voting**

The user can give rating to a book based on his interest. He can rate it by giving a score of five as Excellent, four for very good, three for good, two for regular and one for deficient. The final rating of a book will depend on all the individual userrating.

**6) Shopping Cart**

The user can manage a shopping cart which will include all the books he selected. The user can edit, delete and update his shopping cart. A final shopping cart summary is displayed which includes all the items the user selected and the final total cost.

**7) Managing user accounts**

Each user should have an account to access all the functionalities of website. User can login using login page and logout using the logout page. All the user sessions will be saved in the database.

**8) Administration**

The Administrator will be provided with special functionalities like

- Add or delete a book category
- Add or delete a member
- Manage member orders.
- Add or delete a Credit Card type.

### 3. Requirement Specifications

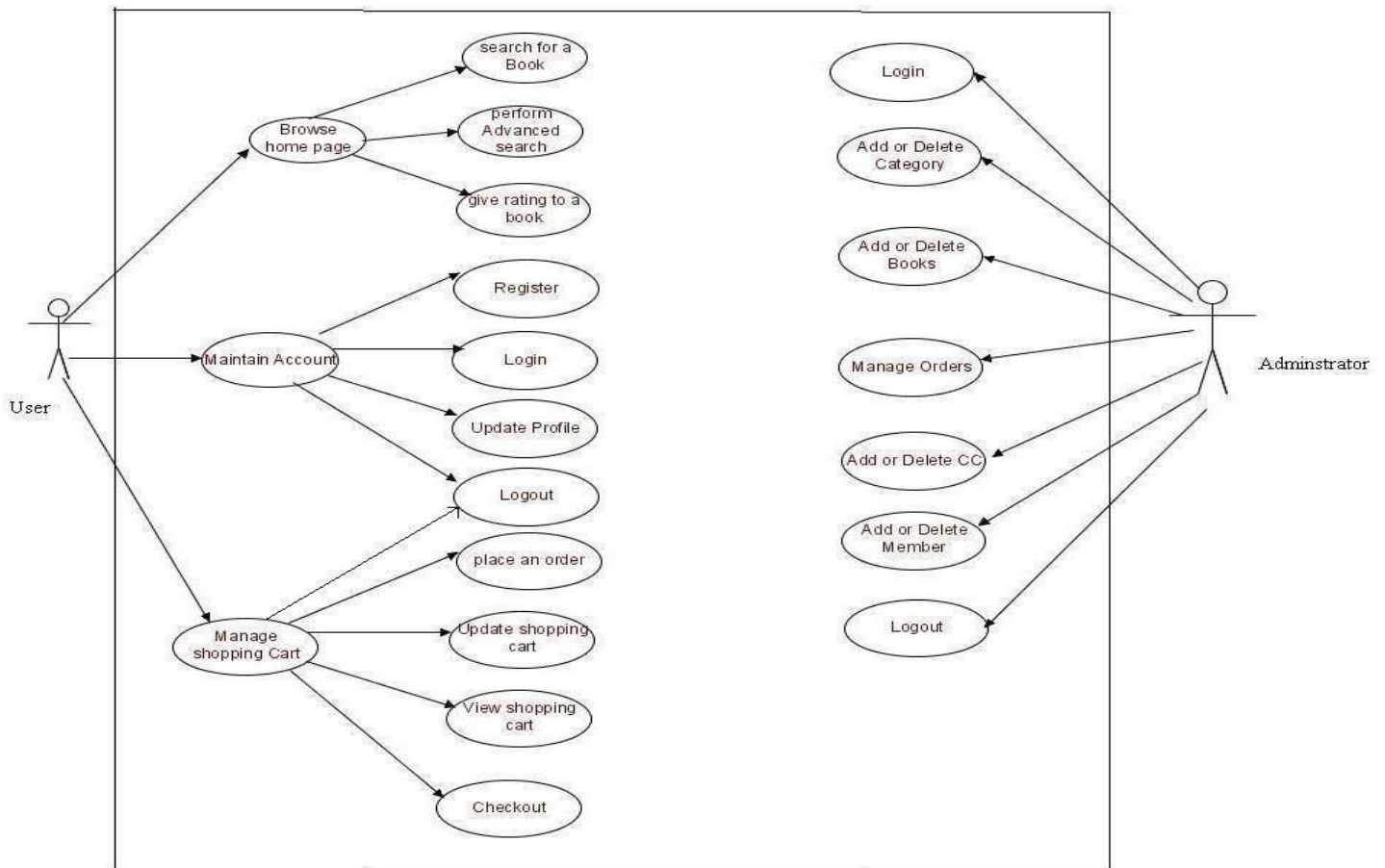
All the requirements are specified using OCL a software specification language in the second phase of my presentation.

#### 3.1 Main Requirements

The Main Requirements include Microsoft Visual Studio 2005 and ASP.NET to develop the web application, SQL Server 2005 to design the database and Mozilla as a main web browser to run the website.

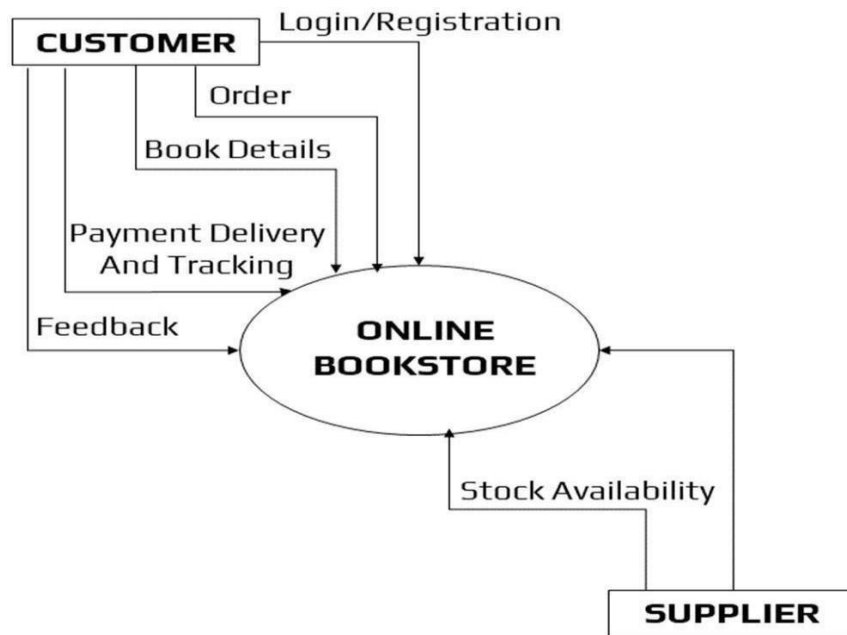
#### 3.2 Critical Use Case Requirements

User and Administrator are the two actors included in the Online Book Store. Fig.2 shows the use case diagram for this website.

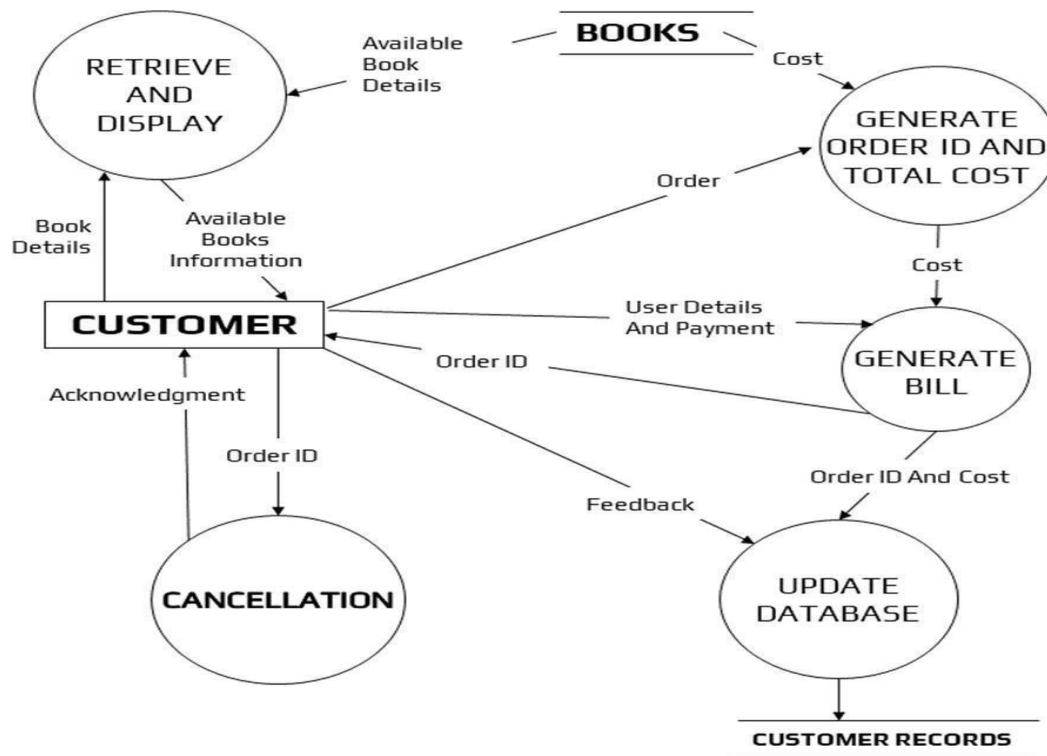


**Fig.2 Use Case Diagram**

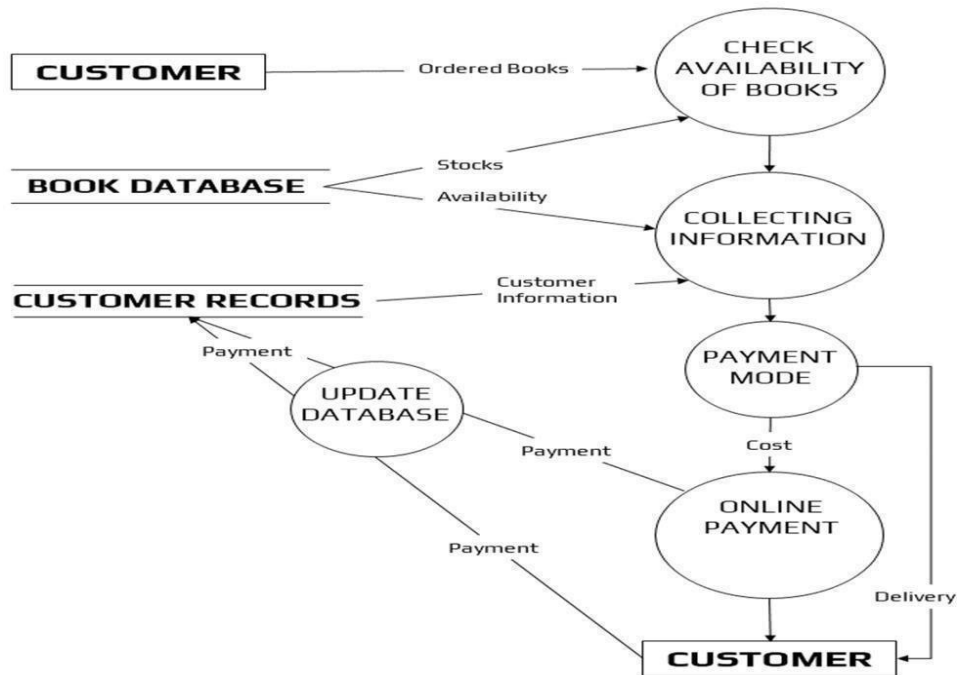
## Zero Level Data flow Diagram (0 Level DFD)



## First Level Data flow Diagram (1 Level DFD)



## Second Level Data flow Diagram (2 Level DFD)



## Use Cases:

### Browse Catalog

#### 1) Search for a Book

- **Purpose:** A user can search for a book of his choice by selecting category and title. Then a select query is used to retrieve data from the database and display the selected information.
- **Actor:** User
- **Input:** The user will select a category and enter title in a text box provided.
- **Output:** The system will display the books which matches the selected search criteria. A dataset is created as a result of select query. Later the dataset is binded to the data repeater to display the selected data.

#### 2) Perform Advanced Search

- **Purpose:** If the user wants to perform an advanced search he can search for a book of his choice by selecting category, title, author and price range. Then a select query is used to retrieve data from the database and display the selected information.
- **Actor:** User
- **Input:** The user will select a category and enter title, author, and price range in a text box provided.
- **Output:** The system will display the books which matches the selected search criteria. A dataset is created as a result of select query. Later the dataset is binded to the data repeater to display the selected data.

#### 3) Give rating to a book

- **Purpose:** If the user wants to give rating according to his opinion for a book he can select either Excellent, Very good, good, regular or deficient. The final rating of a book will depend on all the individual user rating.
- **Actor:** User
- **Input:** The user will select a rating based on his opinion.
- **Output:** The system will display the rating of a book and the total number of votes received. Below is the display for various rating.
  - \*\*\*\*\* Excellent
  - \*\*\*\* Very Good
  - \*\*\* Good
  - \*\* Regular
  - \* Deficient



## Maintain Account

### 1) Register

- **Purpose:** If the user doesn't have an account then he will be asked to register.
- **Actor:** User
- **Input:** The user will enter details in the registration form according to the required fields. The fields include
  1. Username
  2. Password
  3. confirm password
  4. first name
  5. last name
  6. email
  7. Address
  8. Phone
  9. CC details
- **Output:** After registration the user will be directed to the main home page.

### 2) Login

- **Purpose:** If the user wants to get access to all the functionalities of Online Book Store he should login using his username and password.
- **Actor:** User
- **Input:** The user will enter his username and password.
- **Output:** If it is a successful login the user will be directed to the main home page. Else if the user enters invalid information he will be asked to check the entered information.

### 3) Update Profile

- **Purpose:** If the user wants to change his personal account information then he can update his selected fields and the entire data will be updated in the data base through an update query.
- **Actor:** User
- **Input:** The user will update his account information.
- **Output:** The system will update the entered information in the database using an update query.

### 4) Logout

- **Purpose:** If the user wants to end his session and sign out of the website then he can use the logout option.
- **Actor:** User

- **Input:** The user will click the logout button.
- **Output:** The user's account session comes to an end and he should login again if he wants to enter into the website.

## **Manage Shopping Cart**

### **1) Place an order**

- **Purpose:** If the user wants to purchase a book then he can place an order by selecting the add to shopping cart button and entering the quantity required under the book description.
- **Actor:** User
- **Input:** The user will enter the quantity required and click the add to shopping cart button.
- **Output:** The order will be added to the user's shopping cart.

### **2) Update Shopping Cart**

- **Purpose:** If the user wants to change the quantity of a book or change a book then he can update his shopping cart.
- **Actor:** User
- **Input:** The user will click the details button in the shopping cart summary to edit and update his order details..
- **Output:** The updated order details are reflected in the shopping cart summary.

### **3) View Shopping Cart**

- **Purpose:** If the user wants to view the items he added to the shopping cart then he can click the shopping cart link at the top of the page.
- **Actor:** User
- **Input:** The user will click the shopping cart link at the top of every page.
- **Output:** The user's shopping cart summary will be displayed in the form of a tabular format with all the books and their quantity. A total cost of all the items is also displayed at the bottom.

## **Administrator**

### **1) Login**

- **Purpose:** If the Administrator wants to get access to all the functionalities of Online Book Store he should login using his username and password.
- **Actor:** Administrator
- **Input:** The Administrator will enter his username and password.

- **Output:** If it is a successful login the Administrator will be directed to his menu page. Else if the Administrator enters invalid information he will be asked to check the entered information.

## 2) Add or Delete Category

- **Purpose:** If the Administrator wants to add or delete a book category then he can insert or delete a book category using his administration rights and the category table will be updated in the database.
- **Actor:** Administrator
- **Input:** If the Administrator wants to add a book category the he should click the insert link button in the category page else he can delete a particular selected book category.
- **Output:** The updated categories list will be displayed in the main home page.

## 3) Add or Delete Book

- **Purpose:** If the Administrator wants to add or delete a book then he can insert or delete a book using his administration rights and the book table will be updated in the database.
- **Actor:** Administrator
- **Input:** If the Administrator wants to add a book the he should click the insert link button in the book page and fill the following fields related to the book.
  1. Title
  2. Author
  3. Price
  4. Category
  5. Notes
  6. Product url

If he wants to delete a book he can click the delete button to remove it from the database.

- **Output:** The updated books list will be displayed in the main home page under their particular category.

## 4) Manage Orders

- **Purpose:** If the Administrator wants to add or delete an order then he can insert or delete an order using his administration rights.
- **Actor:** Administrator
- **Input:** If the Administrator wants to add an order the he should click the insert link button in the orders page else he can delete a particular selected order
- **Output:** The updated orders list will be processed to the users.

### 5) Add or Delete CC (Credit Card)

- **Purpose:** If the Administrator wants to add or delete a CC type then he can insert or delete a CC type using his administration rights and the CC table will be updated in the database.
- **Actor:** Administrator
- **Input:** If the Administrator wants to add a CC type the he should click the insert link button in the CC page else he can delete a particular selected CC type..
- **Output:** The updated CC list will be displayed in registration page where the user will select it for his future transactions.

### 6) Add or Delete Member

- **Purpose:** If the Administrator wants to add or delete a book category then he can insert or delete a book category using his administration rights and the category table will be updated in the database.
- **Actor:** Administrator
- **Input:** If the Administrator wants to add a book category the he should click the insert link button in the category page else he can delete a particular selected book category.
- **Output:** The updated categories list will be displayed in the main home page.

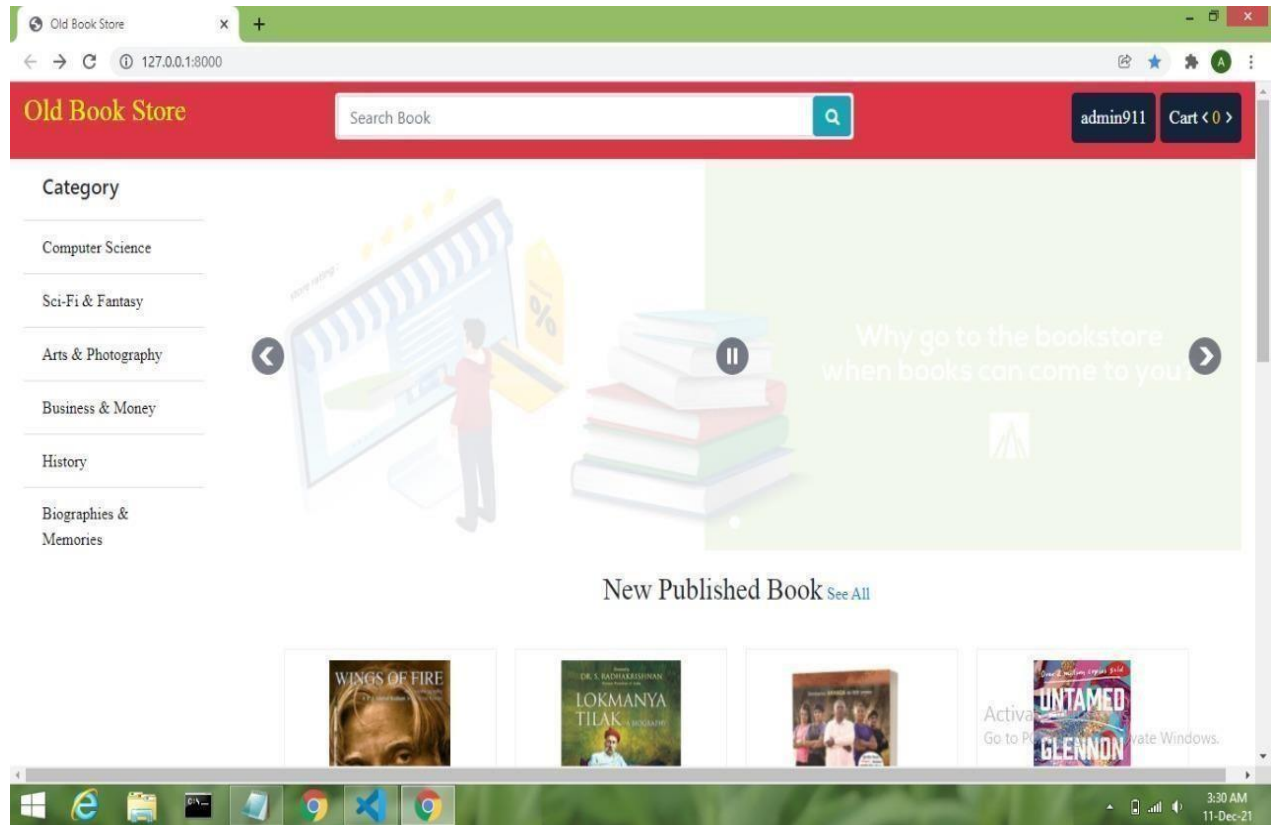
### 7) Logout

- **Purpose:** If the Administrator wants to end his session and sign out of the website then he can use the logout option.
- **Actor:** Administrator
- **Input:** The Administrator will click the logout button.
- **Output:** The Administrator's account session comes to an end and he should login again if he wants to enter into the website.

## 3.3 Environment

- The Online Book store will be developed in Visual Studio 2005 environment.
- C# will be used as the programming language.


# ScreenShots



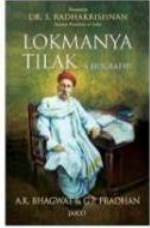
Old Book Store

127.0.0.1:8000


### Top Selling Book



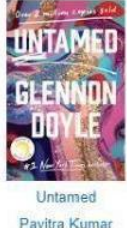
Wings of Fire  
Rajendra Singh Chaudhary  
★★★★★ 0  
₹3000  
[Add to cart](#)



Lokmanya Tilak  
Pavitra Kumar  
★★★★★ 1  
₹6000  
[Add to cart](#)



Akhada  
Saurabh duggal  
★★★★★ 0  
₹3000  
[Add to cart](#)



Untamed  
Pavitra Kumar  
★★★★★ 0  
₹3000  
[Add to cart](#)

**Contact**  
 098977546  
 admin9@gmail.com  
 Ghaziabad

**Menu**  
 > Books

**Old Book Store 2021. All Rights Reserved**

Activate Windows  
Go to PC settings to activate Windows.


Old Book Store

Search Book

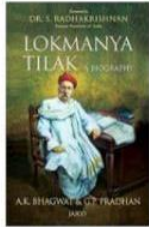
admin911 Cart < 0 >

**Category**


- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- History
- Biographies & Memories



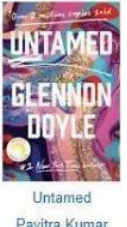
Wings of Fire  
Rajendra Singh Chaudhary  
★★★★★ 0  
₹3000  
[Add to cart](#)



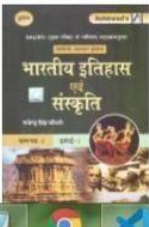
Lokmanya Tilak  
Pavitra Kumar  
★★★★★ 1  
₹6000  
[Add to cart](#)



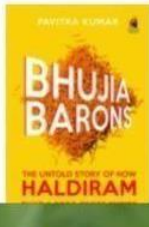
Akhada  
Saurabh duggal  
★★★★★ 0  
₹3000  
[Add to cart](#)



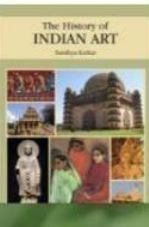
Untamed  
Pavitra Kumar  
★★★★★ 0  
₹3000  
[Add to cart](#)



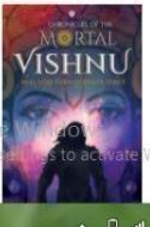
भारतीय इतिहास  
एवं  
संस्कृति  
Rajendra Singh Chaudhary  
₹3000  
[Add to cart](#)



BHUIJA BARONS  
THE UNFOLD STORY OF HOW  
HALDIRAM  
Pavitra Kumar  
₹3000  
[Add to cart](#)



The History of  
INDIAN ART  
₹3000  
[Add to cart](#)



VISHNU  
₹3000  
[Add to cart](#)

Activate Windows  
Go to PC settings to activate Windows.

Old Book Store

Search Book

admin911 Cart < 1 >

**Category**

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- History
- Biographies & Memories

**Bhujia**  
by Pavitra Kumar  
★★★★★ 0 Reviews  
the best book for business

₹Price: 2500  
[Add to cart](#)

★★★★★

Write Your Review

[Submit](#)

Activate Windows  
Go to PC settings to activate Windows.

Old Book Store

Search Book

admin911 Cart < 1 >

**Category**

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- History
- Biographies & Memories

#	Name	Qty	Price	Action
	Bhujia	- 1 +	2500	<a href="#">Remove</a>

[Continue Shopping](#) [Proceed to Checkout](#)

**Checkout Summary**

Books	1
Subtotal	2500
Shipping	100
Payable Total	2600

**Contact**

☎ 098977546  
✉ admin9@gmail.com  
📍 Ghaziabad

**Menu**

> Books

Old Book Store 2021. All Rights Reserved

Activate Windows  
Go to PC settings to activate Windows.

Old Book Store

Search Book

admin911 Cart < 1 >

Category

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- History
- Biographies & Memories

Shipping Address

Name\* Email\*

Phone\*

Address\*

Division\* District\* Zip code\*

Payment method\*

☐ PayTm

☐ Credit Card

Checkout Summary

Books	1
Subtotal	2500
Shipping	100
Payable Total	2600

Activate Windows  
Go to PC settings to activate Windows.

Old Book Store

Search Book

admin911 Cart < 0 >

Category

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- History
- Biographies & Memories

Thank you for you purchase!

OrderID: #201812

Book: 1

Name: Anuj Tripathi

Phone: 09935536710

Email: anujtripathi360@gmail.com

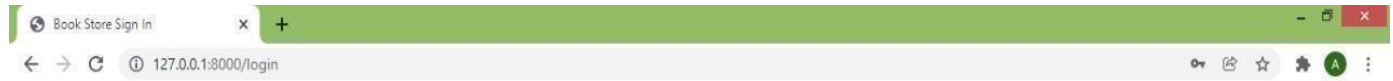
Account No: 97779898776878987876

Payment Method: PayTm

Shipping address: Ghaziabad City:231306, Ghaziabad.1, Goraiya , Sikhar

Activate Windows  
Go to PC settings to activate Windows.





### Sign In

**Username**

**Password**

[Forgot your password?](#)

☐ New to BookStore?

[Create you BookStore account](#)

Activate Windows  
Go to PC settings to activate Windows.



### Create account

**Name:**

**Email:**

**Username:**

**Password:**

**Password confirmation:**

[Create you BookStore account](#)

☐ Already have an account? [Sign in](#)

Activate Windows  
Go to PC settings to activate Windows.



## Section- 2 Architecture Design

### 1. Introduction

The purpose of this document is to provide an architectural design for the Online Book Store. The design will show the presentation tier, the middle tier consisting of classes, sequence diagrams, and the data tier consisting of the database design diagram.

### 2. Architecture

Three-tier (layer) is a client-server architecture in which the user interface, business process (business rules) and data storage and data access are developed and maintained as independent modules or most often on separate platforms.

The Architecture of Online Book Store is based on three-tier architecture. The three logical tiers are

- Presentation tier - ASP.NET Web forms, Master Pages, Images.
- Middle tier – C# classes.
- Data tier- Database

Fig.1 below shows the model of 3-tier architecture.

The main reason for considering three-tier architecture for the Online Book store is as follows:

#### ***Flexibility:***

- Management of data is independent from the physical storage support,
- Maintenance of the business logic is easier,
- Migration to new graphical environments is faster.
- If there is a minor change in the business logic, we don't have to install the entire system in individual user's PCs.

#### ***Reusability:***

- Reusability of business logic is greater for the presentation layer. As this component is developed and tested, we can use it in any other project and would be helpful for future use.

## Security:

- More secured architecture since the client cannot access the database directly.

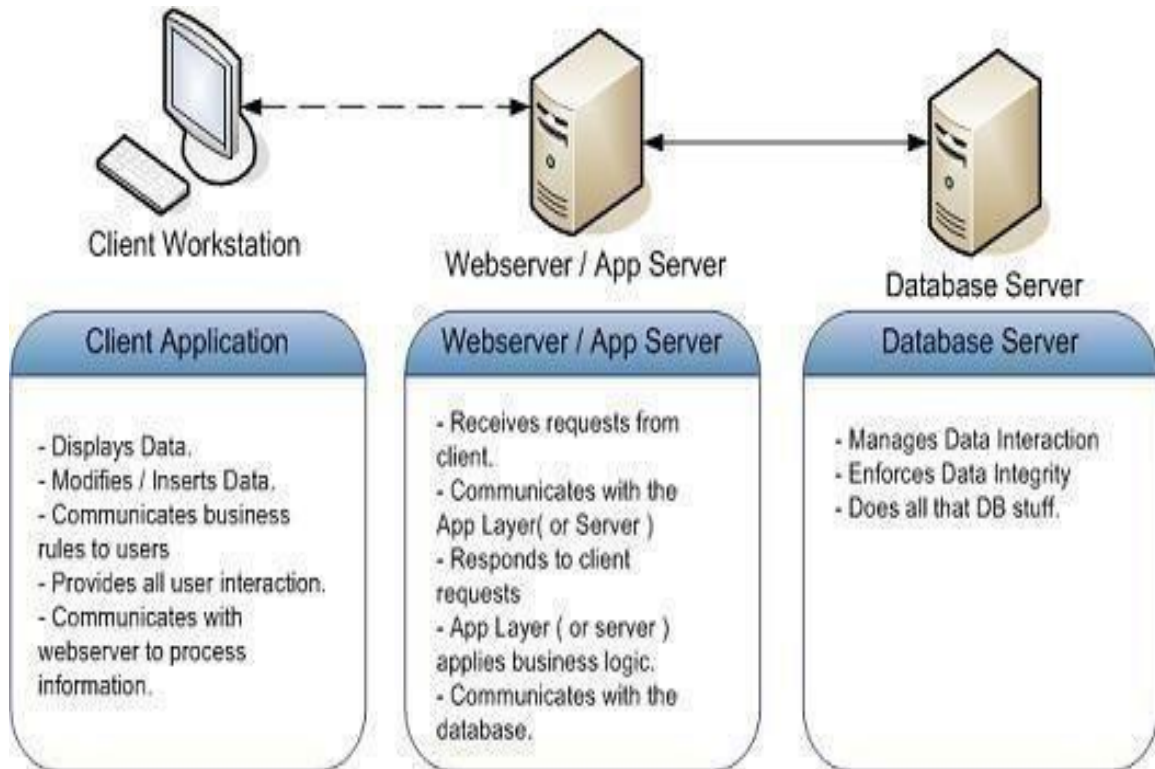
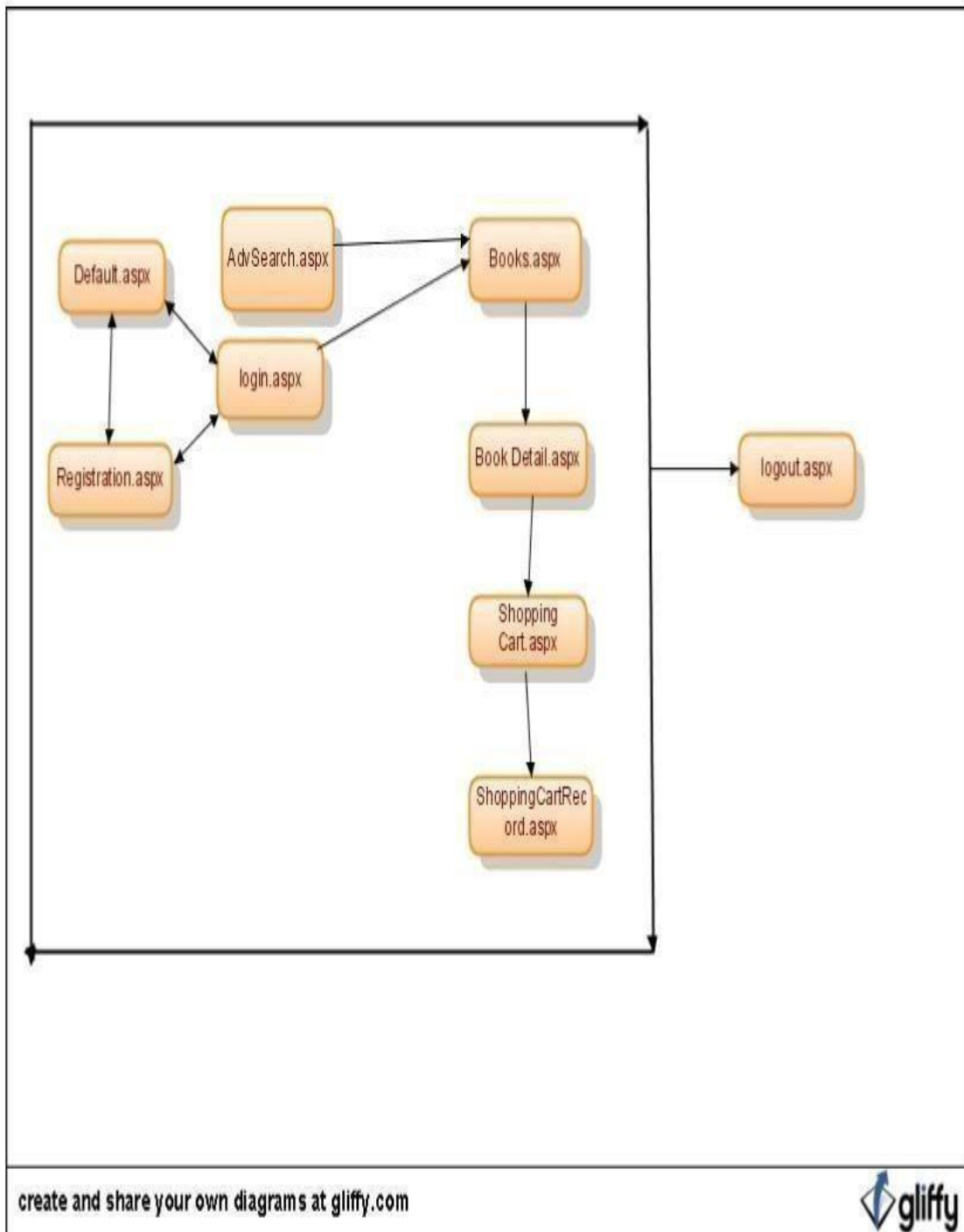


Fig.1 3-tier Architecture

The diagram below captures the page flow for user in the Online Book Store System.



**Fig.2 User-Page Flow**

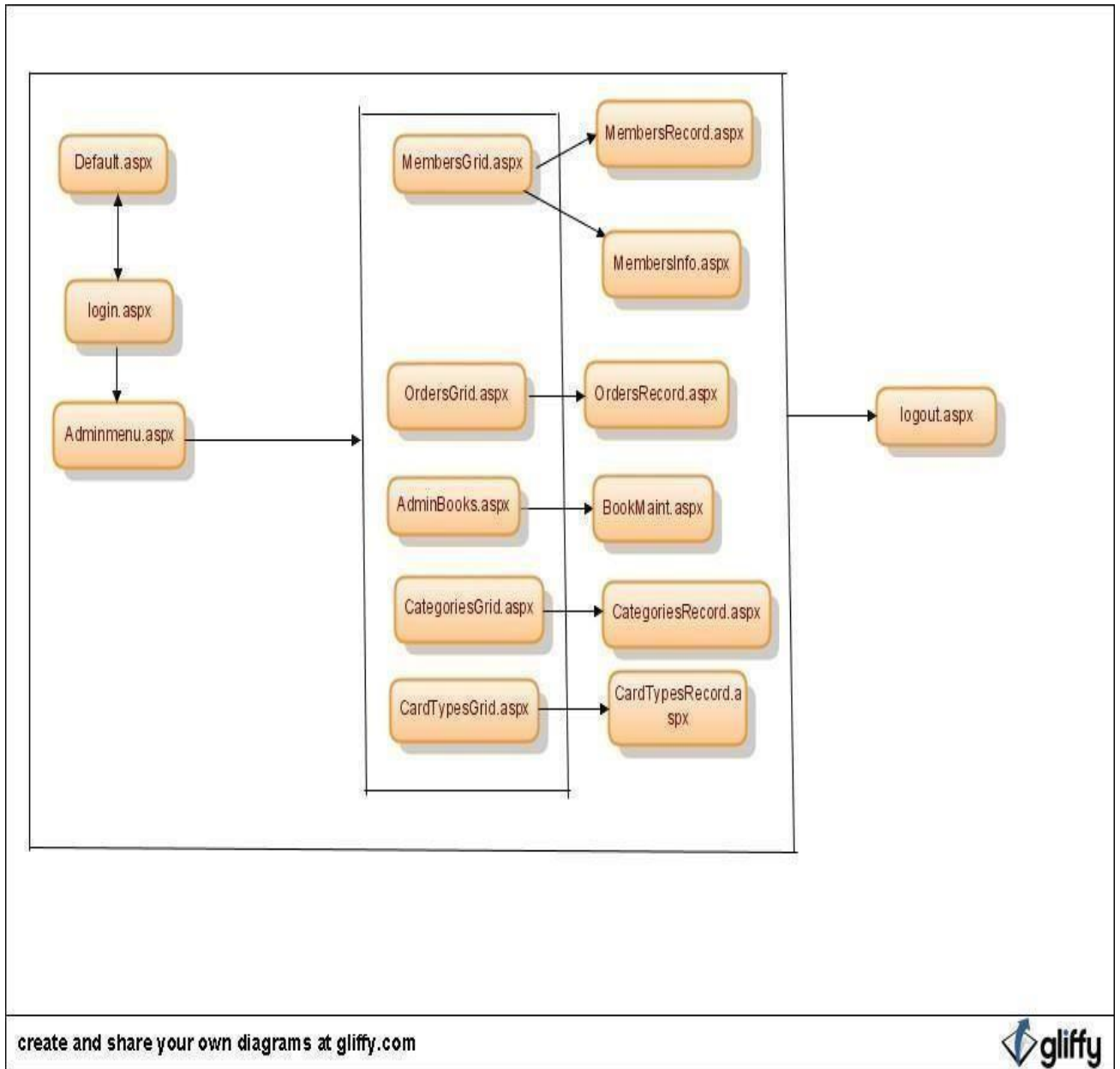


Fig.3 Administrator-Page Flow

### 3. Middle Tier

The Middle Tier or Business Logic layer consists of eleven classes User, Customer, Administrator, SessionManager, Category, BookSet, Book, BooksOrder, Search, Advanced Search and Shopping Cart.

#### Class Diagram

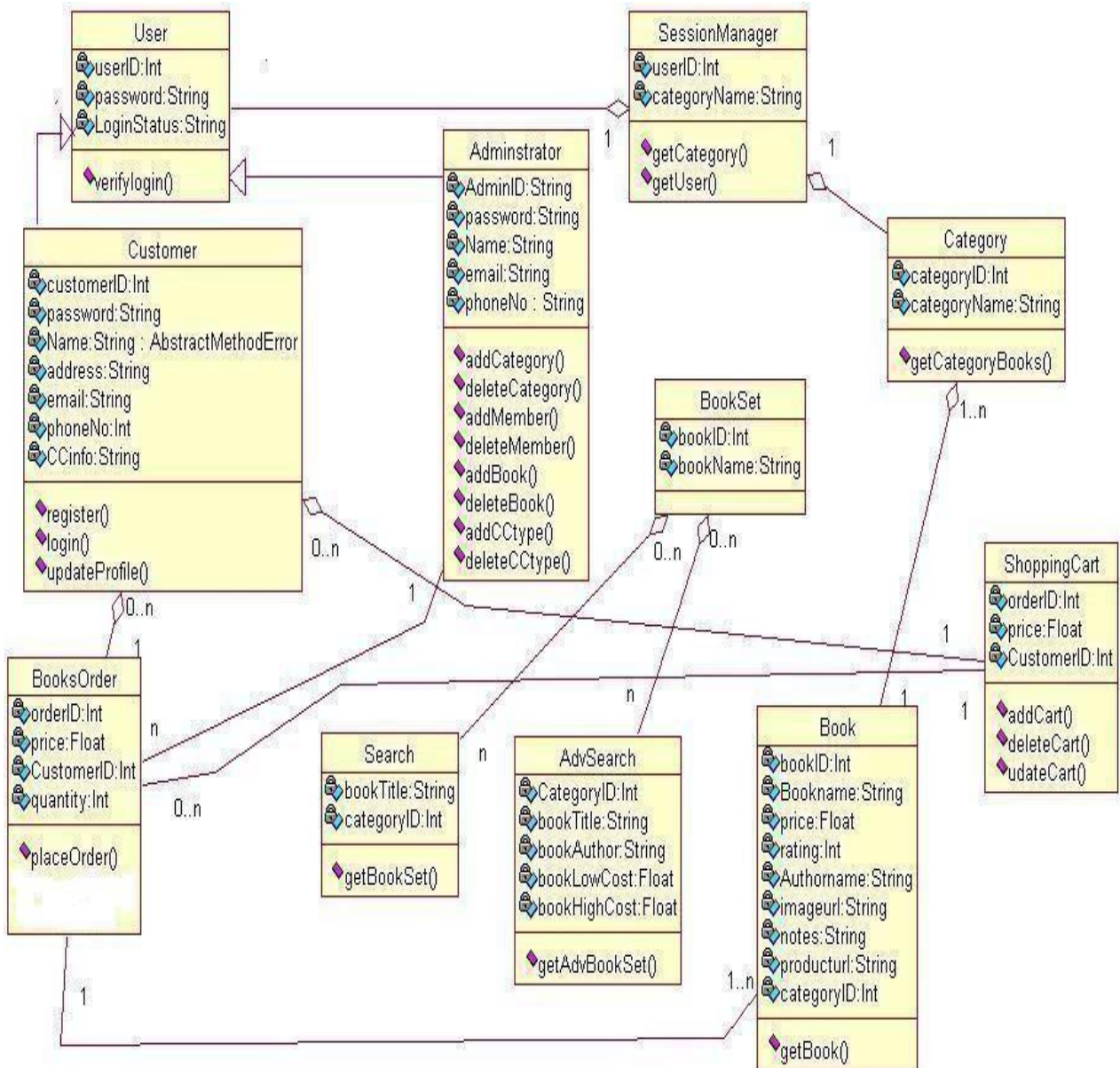


Fig.4 Class Diagram

## Sequence Diagrams

### 1) User Login

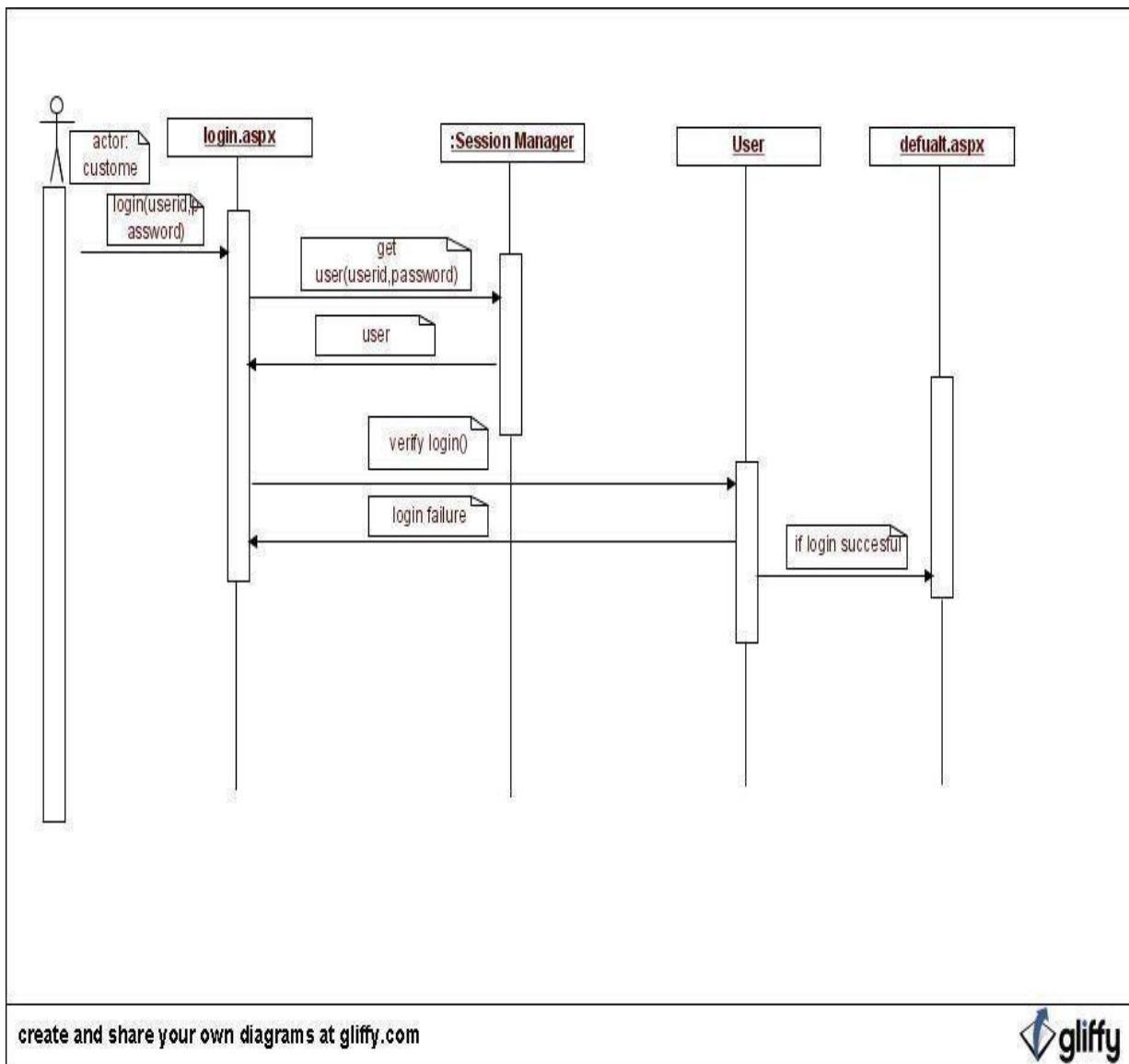


Fig.5 User-Login Sequence Diagram

## 2) Book Search

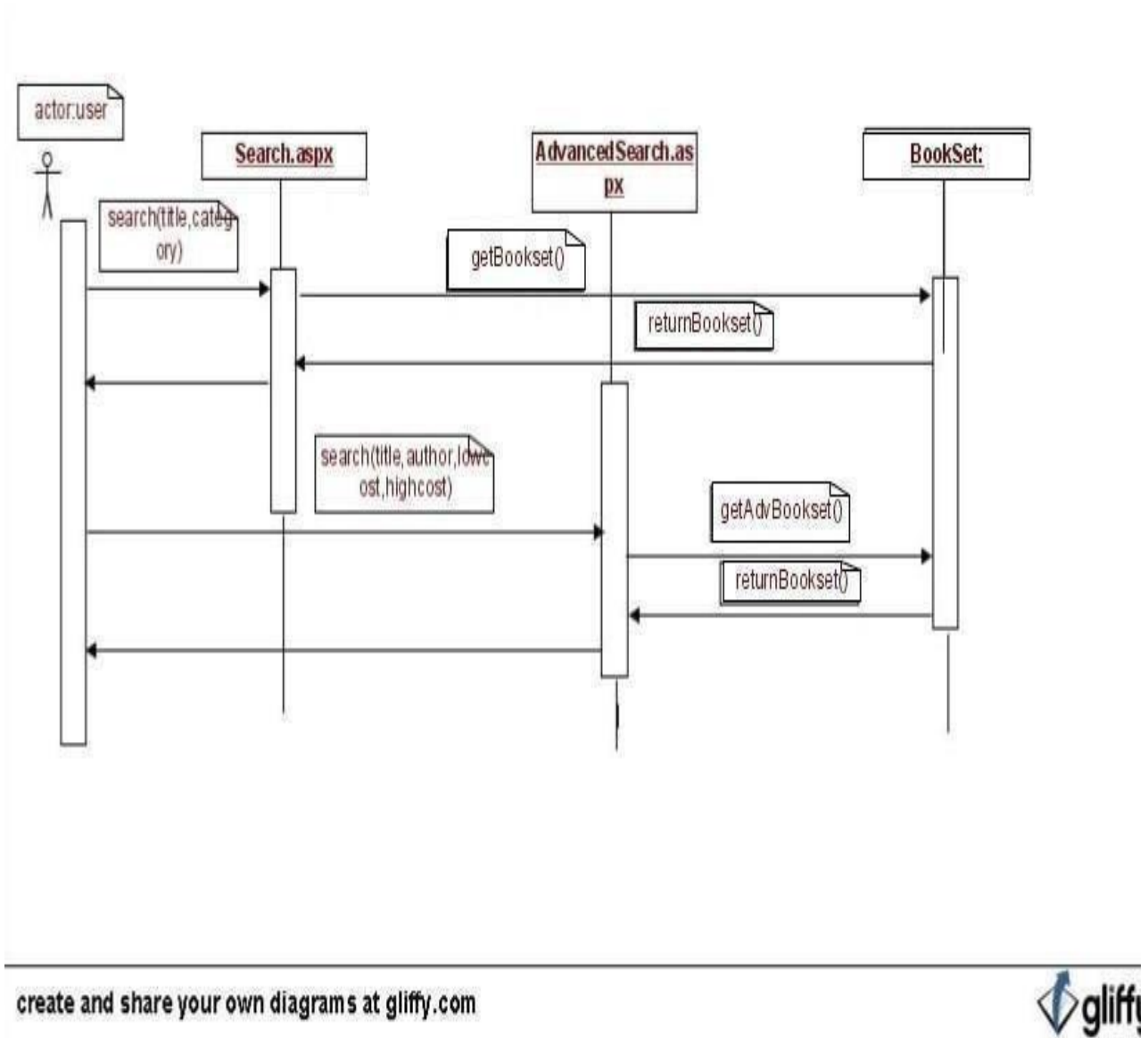


Fig.6 User-Book Search Sequence Diagram



### 3) Add to shopping cart

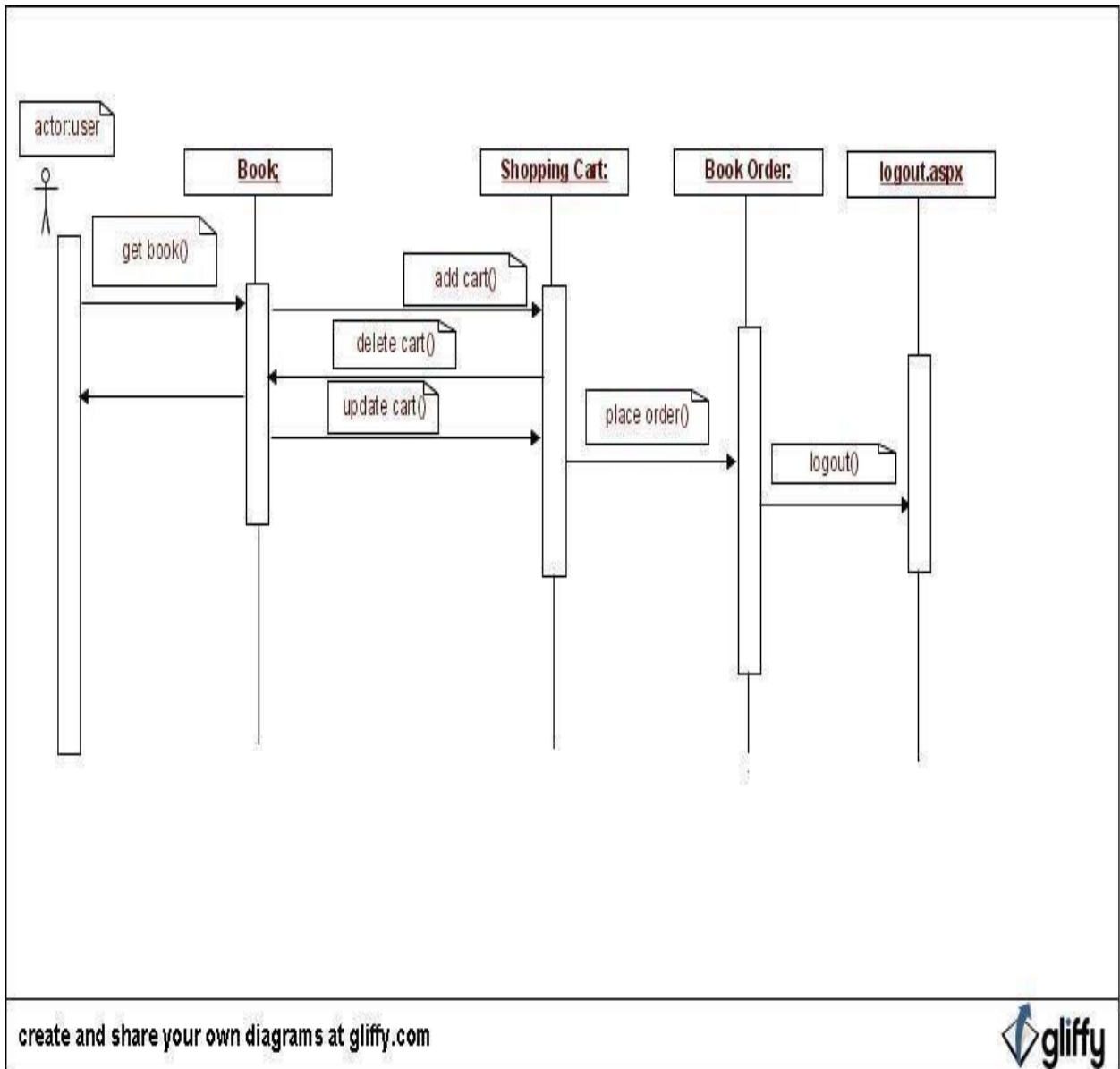


Fig.8 User-Add to Cart Sequence Diagram

#### 4) Administrator

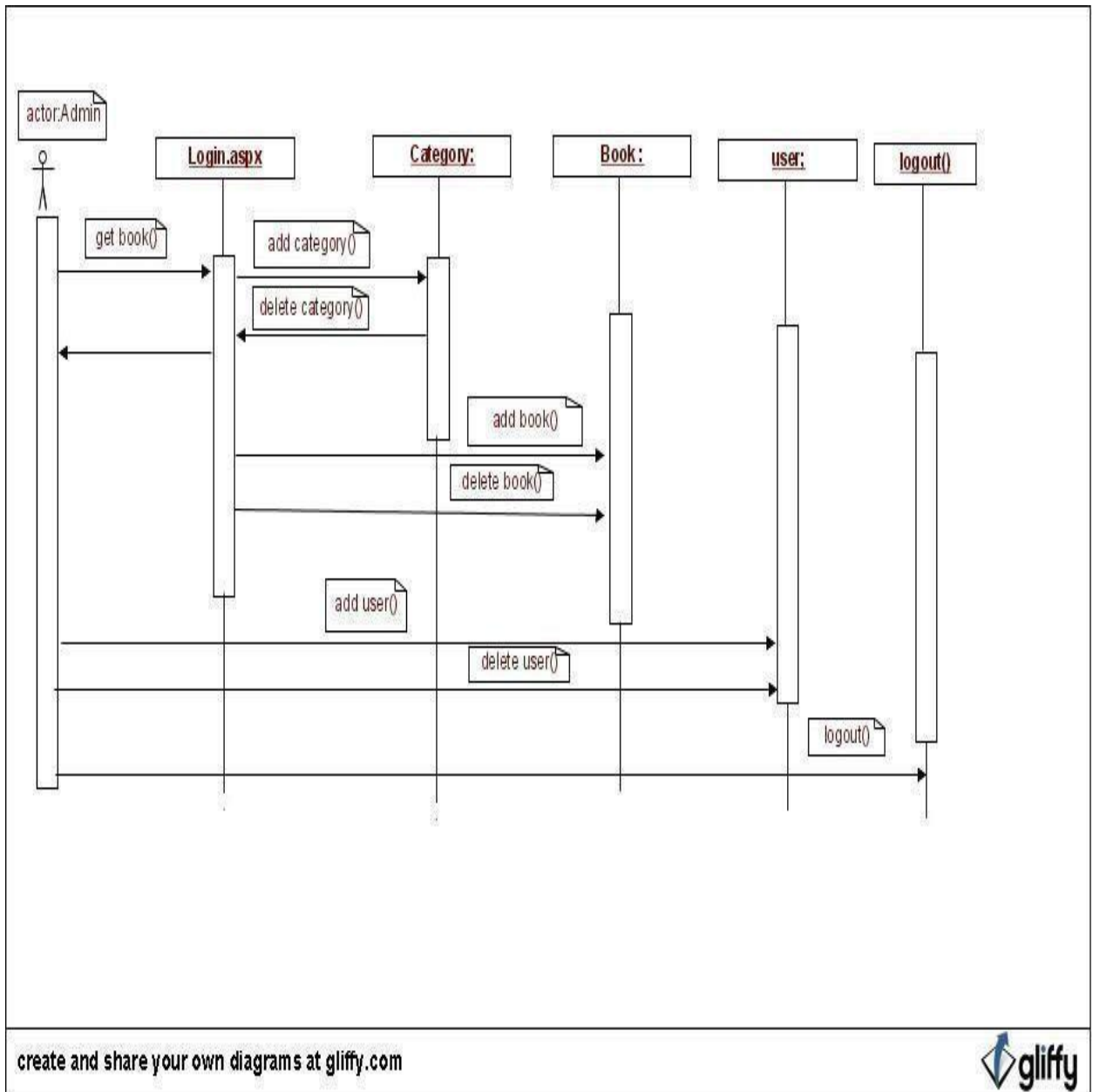


Fig.9 Administrator- Sequence Diagram

#### 4. Data Tier

The system database has five tables Categories, Items, Orders, Members and Card types. The system database design is shown below.

Table Name	Definition
Categories	Contains the Book Categories Information
Items	Contains the Book Information.
Members	Contains the Members Information
Orders	Contains the Book Orders Information
Card Types	Contains the Credit Card Information

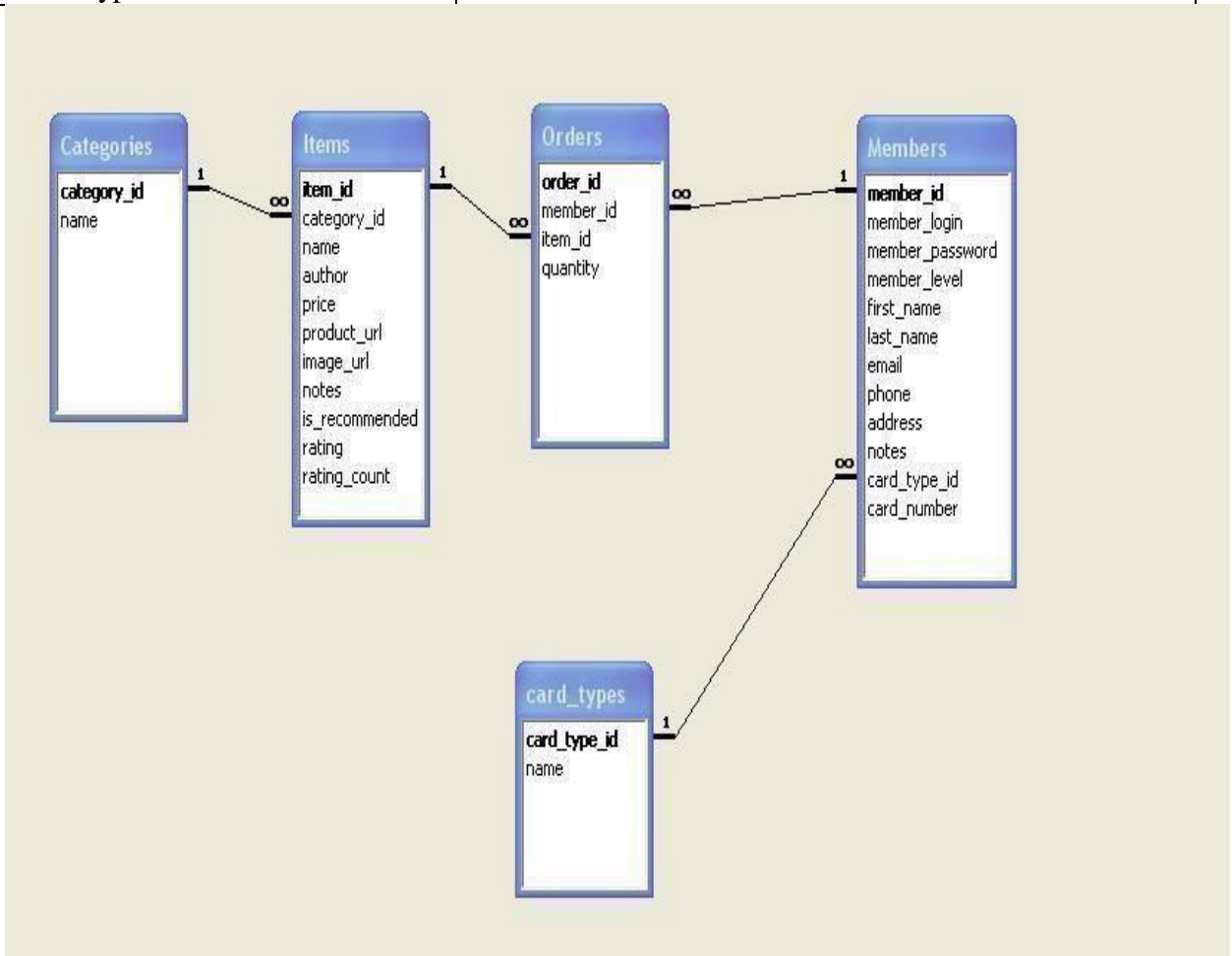


Fig.10 System Database Design

## Coding Part

### Module wise

#### LOGIN PAGE-

```
<!DOCTYPE html>
<html>
<head>
<title>Book Store Sign In</title>
{ % load staticfiles % }
<link rel="stylesheet" type="text/css" href="{ % static 'css/bootstrap.min
.css' % }">
<link rel="stylesheet" type="text/css" href="{ % static 'css/style.css' % }"
>
</head>
<body>

div class="container">
<div class="row justify-content-md-center">
<div class="col-md-4 col-sm-3">
<div class="signin-form">
{ % if messages % }
<div class="messages">
{ % for message in messages % }
<p{ % if message.tags % } class="{{ message.tags }}" { % endif %
}>{{ message }}</p>
{ % endfor % }
</div>
{ % endif % }
```

```

<h3> Sign In</h3>
<form action="" method="POST" class="auth-validate-form">
  {% csrf_token %}
  <div class="form-group">
    <label>Username</label>
    <input type="text" name="user" class="form-control">
  </div>
  <div class="form-group">
    <label>Password</label>
    <input type="password" name="pass" class="form-control">
  </div>
  <button type="submit" class="btn btn-
danger">Login</button>
</form>
<a href="#">Forgot your password?</a>
<div class="a-divider-
break"><h5>New to BookStore?</h5></div>
  <a class="btn btn-
default" href="{% url 'store:registration' %}">Create you BookStore acco
unt</a>
</div>
</div>
</div>
</div>

</body>
</html>

```

## SIGNUP PAGE

```
<!DOCTYPE html>
<html>
<head>
<title>Book Store Registration</title>
{% load staticfiles %}
<link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
</head>
<body>
<div class="container">
<div class="row justify-content-md-center">
<div class="col-md-4 col-sm-3">
<div class="signup-form">
<div class="messages"></div>
<h3> Create account</h3>
<form action="" method="POST" class="auth-validate-form">
    {% csrf_token %}
    {% for field in form %}
        <div class="form-group registration-form">
            {{ field.label_tag }}<br>
            {{ field }}

            {% if field.help_text %}{% endif %}

            {% for error in field.errors %}
                <p style="color: red;font-size: 13px">{{ error }}</p>
            {% endfor %}
        </div>
    </div>
```



```

        <ul id="demo3">
            { % for sld in slide % }
            <li></li>
            { % endfor % }
        </ul>
    </div>
</div>
<div class="col-sm-12">
    <div class="titleheader">
        <h3 class="h2header text-center">New Published Book</h3>
        <a href="#">See All</a>
    </div>
    <div class="regulara sliderzx">
        { % for item in newbooks % }
        <div class="book-wrapper text-center">
            <div class="coverpage">
                
            </div>
            <a href="{ % url 'store:book' id=item.id % }">{ { item.name|text_short
}}</a>
            <a href="{ % url 'store:writer' id=item.writer.id % }">{ { item.writer } }
</a>
            <div class="rating">
                { { item.totalrating|averagerating:item.totalreview } }
                <span class="totalrating">{ { item.totalreview|add:-1 } }</span>
            </div>
            <p>₹{ { item.price } }</p>
            <button class="btn btn-danger" id="addTocart" data-book-
id="{ { item.id } }">
                <i class="fa fa-shopping-cart"></i>Add to cart
            </button>
        </div>
        { % endfor % }
    </div>

```



```

    </div>
</div>

<div class="col-sm-12">
    <div class="titleheader">
        <h3 class="h2header text-center">Top Selling Book</h3>
    </div>
    <div class="regulara sliderzx">
        {% for p in newbooks %}
        <div class="book-wrapper text-center">
            <div class="coverpage">
                
            </div>
            <a href="{% url 'store:book' id=p.id %}">{{ p.name|text_short }}</a>
            <a href="{% url 'store:writer' id=p.writer.id %}">{{ p.writer }}</a>
            <div class="rating">
                {{ p.totalrating|averagerating:p.totalreview }}
                <span class="totalrating">{{ p.totalreview|add:-1 }}</span>
            </div>
            <p>₹{{ p.price }}</p>
            <button class="btn btn-danger" id="addTocart" data-book-
id="{{ p.id }}">
                <i class="fa fa-shopping-cart"></i>Add to cart
            </button>
        </div>
        {% endfor %}
    </div>
</div>
{% endblock %}

```

## **CART**

```
from decimal import Decimal
from django.conf import settings
from store.models import Book

class Cart(object):
    def __init__(self, request):
        self.session = request.session
        cart = self.session.get(settings.CART_SESSION_ID)
        if not cart:
            cart = self.session[settings.CART_SESSION_ID] = {}
        self.cart = cart

    def add(self, book):
        book_id = str(book.id)
        if book_id not in self.cart:
            self.cart[book_id] = {'quantity': 0, 'price': str(book.price)}
            self.cart[book_id]['quantity'] = 1
        else:
            if self.cart[book_id]['quantity'] < 10:
                self.cart[book_id]['quantity'] += 1

        self.save()

    def update(self, book, quantity):
        book_id = str(book.id)
        self.cart[book_id]['quantity'] = quantity

        self.save()
```

```

def save(self): self.session[settings.CART_SESSION_ID]
    = self.cartself.session.modified = True

def remove(self, book):
    book_id = str(book.id)
    if book_id in self.cart:
        del self.cart[book_id]
        self.save()

def __iter__(self):
    book_ids = self.cart.keys()
    books = Book.objects.filter(id__in=book_ids)
    for book in books:
        self.cart[str(book.id)]['book'] = book

    for item in self.cart.values(): item['price'] =
        Decimal(item['price'])
        item['total_price'] = item['price'] * item['quantity']yield
        item

def __len__(self):
    return sum(item['quantity'] for item in self.cart.values())

def get_total_price(self):
    return sum(Decimal(item['price']) * item['quantity'] for item in self.cart.values())

def clear(self):
    del self.session[settings.CART_SESSION_ID]
    self.session.modified = True

```

## ORDER

```
{% extends 'store/base.html' %}

{% load static %}
{% load customfunction %}
{% load crispy_forms_tags %}

{% block container %}
<div class="row">
  <div class="col-md-8 order-md-1 cart_info">
    <h4 class="mb-3 mt-3">Shipping Address</h4>
    {% if messages %}
    <div class="messages">
      {% for message in messages %}
        <p{% if message.tags %} class="{{ message.tags }}" {% endif %}>{{ mes
sage }}</p>
      {% endfor %}
    </div>
    {% endif %}
    <form action="" method="POST" class="needs-
validation" novalidate="">
      {% csrf_token %}
      <div class="row">
        <div class="col-md-6 mb-3">
          {{ form.name|as_crispy_field }}
        </div>
        <div class="col-md-6 mb-3">

          {{ form.email|as_crispy_field }}
        </div>
      </div>
    </div>
```

```

<div class="mb-3">
    {{ form.phone|as_crispy_field }}
</div>

<div class="mb-3">
    {{ form.address|as_crispy_field }}
</div>
<div class="row">
    <div class="col-md-5 mb-3">
        {{ form.division|as_crispy_field }}
    </div>
    <div class="col-md-4 mb-3">
        {{ form.district|as_crispy_field }}
    </div>
    <div class="col-md-3 mb-3">
        {{ form.zip_code|as_crispy_field }}
    </div>
</div>
<hr class="mb-4">
<div class="d-block my-3">
    {{ form.payment_method|as_crispy_field }}
</div>
<div class="row">
    <div class="col-md-6 mb-3">
        {{ form.account_no|as_crispy_field }}
    </div>
    <div class="col-md-6 mb-3">
        {{ form.transaction_id|as_crispy_field }}
    </div>
</div>
<hr class="mb-4">
<button class="btn btn-success btn-

```

```

block" type="submit" style="margin-
bottom: 20px">Continue to checkout</button>
    </form>
</div>
<div class="col-md-4 order-md-2 mb-4">
    <ul class="list-group">
        <li class="list-group-item d-flex justify-content-between align-items-
center"><h4>Checkout Summary</h4></li>
        <li class="list-group-item d-flex justify-content-between align-items-
center">Books<span>{ { cart|length } }</span></li>
        <li class="list-group-item d-flex justify-content-between align-items-
center">Subtotal<span>{ { cart.get_total_price } }</span></li>
        <li class="list-group-item d-flex justify-content-between align-items-
center">Shipping<span>{ { cost|shipping } }</span></li>
        <li class="list-group-item d-flex justify-content-between align-items-
center">Payable Total<span>{ { cart.get_total_price|payabletotal } }</span></li>
    </ul>
</div>
</div>
{% endblock %}

```

```

from django.db import models
from store.models import Book
from django.contrib.auth.models import User

```

```

class Order(models.Model):
    customer = models.ForeignKey(User, on_delete = models.CASCADE)
    name = models.CharField(max_length=30)
    email = models.EmailField()
    phone = models.CharField(max_length=16)

```

```

        address = models.CharField(max_length=150)
        division = models.CharField(max_length=20)
        district = models.CharField(max_length=30)
        zip_code = models.CharField(max_length=30)
        payment_method = models.CharField(max_length = 20)
        account_no = models.CharField(max_length = 20)
        transaction_id = models.IntegerField()
        payable = models.IntegerField()
        totalbook = models.IntegerField()
        created = models.DateTimeField(auto_now_add=True)
        updated = models.DateTimeField(auto_now=True)
        paid = models.BooleanField(default=False)

```

```

class Meta:

```

```

    ordering = ('-created', )

```

```

def __str__(self):

```

```

    return 'Order {}'.format(self.id)

```

```

def get_total_cost(self):

```

```

    return sum(item.get_cost() for item in self.items.all())

```

```

class OrderItem(models.Model):

```

```

    order = models.ForeignKey(Order, on_delete=models.CASCADE)

```

```

    book = models.ForeignKey(Book, on_delete=models.CASCADE)

```

```

    price = models.DecimalField(max_digits=10, decimal_places=2)

```

```

    quantity = models.PositiveIntegerField(default=1)

```

```

def __str__(self):

```

```

    return '{}'.format(self.id)

```

```

def get_cost(self):

```

```

    return self.price * self.quantity

```

## PDF

```
!DOCTYPE html>

<html>
<head>
  <title>Book Store Sign In</title>
  {% load staticfiles %}
  <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
  <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
  <link rel="stylesheet" type="text/css" href="{% static 'css/font-awesome.min.css' %}">
</head>
<body>
  <div class="container">
    <div class="row justify-content-md-center">
      <div class="col-md-12 cart_info">
        <div class="ordered-info">
          <h1>OrderID: #2018{{ order.id }}</h1>
          <h1>Book: {{ order.totalbook }}</h1>
          <h1>Name: {{ order.name }}</h1>
          <h1>Phone: {{ order.phone }}</h1>
          <h1>Email: {{ order.email }}</h1>
          <h1>Account No: {{ order.account_no }}</h1>
          <h1>Payment Method: {{ order.payment_method }}</h1>
          <h1>Shipping address: {{ order.division }}:{{ order.zip_code }}, {{ order.district }}, {{ order.address }}</h1>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```



```
</div>
</body>
</html>
```

## PDF CRREATOR

```
from io import BytesIO
from django.http import HttpResponse
from django.template.loader import get_templatefrom
xhtml2pdf import pisa

def renderPdf(template, content={ }):
    t=get_template(template)
    send_data=t.render(content)
    result=BytesIO()
    pdf=pisa.pisaDocument(BytesIO(send_data.encode("ISO-8859-
1")),result)

    if not pdf.err:
    return HttpResponse(result.getvalue(),content_type='application/pdf') else:
    return None
```

## SUCCESSFULL

```
{ % extends 'store/base.html' % }
```

```

{% load static %}

{% block container %}
    <div class="row">
        <div class="col-md-12 cart_info">
            <div class="successful-order text-center">
                <i class="fa fa-check"></i>
                <h3 class="page-header">Thank you for you purchase!</h3>
            </div>
            <div class="ordered-info">
                <h5>OrderID: <span>#2018{{ order.id }}</span></h5>
                <h5>Book: <span>{{ order.totalbook }}</span></h5>
                <h5>Name: <span>{{ order.name }}</span></h5>
                <h5>Phone: <span>{{ order.phone }}</span></h5>
                <h5>Email: <span>{{ order.email }}</span></h5>
                <h5>Account No: <span>{{ order.account_no }}</span></h5>
                <h5>Payment Method: <span>{{ order.payment_method }}</span></h5>
            >
                <h5>Shipping address: <span>{{ order.division }}:{{ order.zip_code }}, {
{ order.district }}, {{ order.address }}</span></h5>
            </div>
            <div class="download-info">
                <a href="{% url 'order:pdf' id=order.id %}"><i class="fa fa-
download" aria-      hidden="true"></i></a>
            </div>
        </div>
    </div>
{% endblock %}

```

## ADRESS FORM

```
from django import forms from
.models import Order
```

```
class OrderCreateForm(forms.ModelForm):DIVISION_CHOICES = (
    ('Ghaziabad City', 'Ghaziabad City'),
    ('Varanasi City', 'Varanasi City'),
    ('Mirzapur City', 'Mirzapur City'),
    ('Noida', 'Noida'),
    ('Mumbai', 'Mumbai'),
)
```

```
DISCRICT_CHOICES = (
    ('Ghaziabad.1', 'Ghaziabad.1'),
    ('Varanasi.2', 'Varanasi.2'),
    ('Mirzapur.3', 'Mirzapur.3'),
    ('Noida.4', 'Noida.4'),
    ('Mumbai.5', 'Mumbai.5'),
)
```

```
PAYMENT_METHOD_CHOICES = (
    ('PayTm', 'PayTm'),
    ('Credit Card', 'Credit Card')
)
```

```
division = forms.ChoiceField(choices=DIVISION_CHOICES)
district = forms.ChoiceField(choices=DISCRICT_CHOICES)
```

```
payment_method = forms.ChoiceField(choices=PAYMENT_METHOD_CHOICES, w
idget=forms.Ra    dioSelect())
```

```
class Meta:
    model = Order
```

```
fields = ['name', 'email', 'phone', 'address', 'division', 'district', 'zip_code', 'payment_method', 'account_number', 'transaction_id']
```

## CUSTOM FUNCTION

```
#from django.shortcuts import render, redirect, get_object_or_404#from
..models import Category, Writer, Book
from django.http import HttpResponse
from django.db.models import Avg, Max, Min, Sum
from django.utils.safestring import mark_safe
from django import template
register = template.Library()
```

```
shippingConst = 100
```

```
@register.filter(name='text_short')def
text_short(value):
temp = value[0:50]
return temp
```

```
@register.filter(name='shipping')
def shipping(value):
return shippingConst
```

```
@register.filter(name='payabletotal')def
payabletotal(value):
return value+shippingConst
```

```
@register.filter(name='averagerating')
```

```

    def averagerating(value, args):
        temp = value / args
        if int(temp + 0.5) > int(temp):
            temp = int(temp + 0.5)
        else:
            temp = int(temp)

        if temp > 5:
            temp = 5

        if temp == 1:
            temp1 = "<span class='fa fa-star checked'></span> <span class='fa fa-
star'></span> <span class='fa fa-star'></span> <span class='fa fa-
star'></span> <span class='fa fa-star'></span>"
        elif temp == 2:
            temp1 = "<span class='fa fa-star checked'></span> <span class='fa fa-star
checked'></span> <span class='fa fa-star'></span> <span class='fa fa-
star'></span> <span class='fa fa-star'></span>"
        elif temp == 3:
            temp1 = "<span class='fa fa-star checked'></span> <span class='fa fa-
star checked'></span> <span class='fa fa-star checked'></span> <span class='fa fa-
star'></span> <span class='fa fa-star'></span>"
        elif temp == 4:
            temp1 = "<span class='fa fa-star checked'></span> <span class='fa fa-
star checked'></span> <span class='fa fa-star checked'></span> <span class='fa fa-
star checked'></span> <span class='fa fa-star'></span>"
        else:
            temp1 = "<span class='fa fa-star checked'></span> <span class='fa fa-
star checked'></span> <span class='fa fa-star checked'></span> <span class='fa fa-star
checked'></span> <span class='fa fa-star checked'></span>"

        return mark_safe(temp1)

@register.filter(name='subtotal')def
subtotal(value, args):

```

```
return value*args
```

## VIEWS

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.models import User
from .models import Category, Writer, Book, Review, Slider
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator
from .forms import RegistrationForm, ReviewForm
```

```
def index(request):
    newpublished = Book.objects.order_by('-created')[:15]
    slide = Slider.objects.order_by('-created')[:3]
    context = {
        "newbooks": newpublished,
        "slide": slide
    }
    return render(request, 'store/index.html', context)
```

```
def signin(request):
    if request.user.is_authenticated:
        return redirect('store:index')
    else:
        if request.method == "POST":
            user = request.POST.get('user')
            password = request.POST.get('pass')
```

```

    auth = authenticate(request, username=user, password=password)if
    auth is not None:
        login(request, auth)
        return redirect('store:index')
    else:
        messages.error(request, 'username and password doesn\'t match')return

render(request, "store/login.html")

def signout(request):
    logout(request)
    return redirect('store:index')

def registration(request):
    form = RegistrationForm(request.POST or None)if
    form.is_valid():
        form.save()
        return redirect('store:signin')

    return render(request, 'store/signup.html', {"form": form})def

payment(request):
    return render(request, 'store/payment.html')

def get_book(request, id):
    form = ReviewForm(request.POST or None)
    book = get_object_or_404(Book, id=id)
    rbooks = Book.objects.filter(category_id=book.category.id) r_review
    = Review.objects.filter(book_id=id).order_by('-created')

    paginator = Paginator(r_review, 4)
    page = request.GET.get('page')
    rreview = paginator.get_page(page)

```

```

if request.method == 'POST':
    if request.user.is_authenticated:
        if form.is_valid():
            temp = form.save(commit=False)
            temp.customer = User.objects.get(id=request.user.id)
            temp.book = book
            temp = Book.objects.get(id=id)
            temp.totalreview += 1
            temp.totalrating += int(request.POST.get('review_star'))
            form.save()
            temp.save()

            messages.success(request, "Review Added Successfully")
            form = ReviewForm()
        else:
            messages.error(request, "You need login first.")
            context = {
                "book": book,
                "rbooks": rbooks,
                "form": form,
                "review": review
            }
            return render(request, "store/book.html", context)

def get_books(request):
    books_ = Book.objects.all().order_by('-created')
    paginator = Paginator(books_, 10)
    page = request.GET.get('page')
    books = paginator.get_page(page)
    return render(request, "store/category.html", {"book": books})

def get_book_category(request, id):

```



```

book_ = Book.objects.filter(category_id=id)
paginator = Paginator(book_, 10)
page = request.GET.get('page')
book = paginator.get_page(page)
return render(request, "store/category.html", {"book":book})

def get_writer(request, id):
    wrt = get_object_or_404(Writer, id=id) book =
    Book.objects.filter(writer_id=wrt.id)context =
    {
        "wrt": wrt,
        "book": book
    }
    return render(request, "store/writer.html", context)

```

## SETTINGS

Django settings for bookstore project.

Generated by 'django-admin startproject' using Django 2.1.2.

For more information on this file, see <https://docs.djangoproject.com/en/2.1/topics/settings/>

For the full list of settings and their values, see  
<https://docs.djangoproject.com/en/2.1/ref/settings/>"""

```

import os
from django.contrib.messages import constants as messages

```

```

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(____file____)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret! SECRET_KEY
=  '#*k)9t8vzu)4j2hr=l@)emxs^ev%6z5)ri$q4g4%nb^n+pr37y'

# SECURITY WARNING: don't run with debug turned on in production!DEBUG
= True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'store.apps.StoreConfig',
    'cart.apps.CartConfig',
    'search.apps.SearchConfig',
    'order.apps.OrderConfig',
    'crispy_forms',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',

```

```

'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'bookstore.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [ 'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'store.context_processors.sidebar', 'store.context_processors.cart'
            ],
        },
    ],
]

```

```
WSGI_APPLICATION = 'bookstore.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases """
```

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'bookstore',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        }
    }
}

""" DATABASES
= {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3', 'NAME':
        os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Password validation
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },

```

```

    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC_URL = '/static/'
#STATIC_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static")
MEDIA_URL = '/media/'

#MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "files/")
MEDIA_ROOT = os.path.join(BASE_DIR, 'files/')

CART_SESSION_ID = 'cart'

```

```
MESSAGE_TAGS = {  
    messages.INFO: 'alert alert-info',  
    messages.SUCCESS: 'alert alert-success',  
    messages.WARNING: 'alert alert-warning',  
    messages.ERROR: 'alert alert-danger',  
    messages.DEBUG: 'alert alert-info',  
}
```

```
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

## **Section-3 Test Plan**

### **1. Testplan identifier**

CIS 895-MSE Project Test plan Online Book Store V1.0

#### **1. Introduction**

The goal of this document is to develop a test plan for the Online Book Store design system. This document defines all the procedures and activities required to prepare for testing of the functionalities of the system which are specified in Vision document. The objectives of the test plan are to define the activities to perform testing, define the test deliverables documents and to identify the various risks and contingencies involved in testing.

#### **2. Features to be tested**

The following list describes the features to be tested:

##### **USER:**

- Registration
- Login
- Add To Cart
- Edit Cart

##### **ADMIN:**

- Create and Delete book from Category
- Create and Delete a Category
- Manage Orders
- Manage Member

## **Section-4 Project Evaluation**

### **1. Introduction**

This document evaluates the experience of the development of the Online Book Store project. A brief description of the tools, process, techniques employed as well as the mistakes made is presented so that lessons are documented and learned.

### **2. Problems faced**

The following are the problems faced during the design of Online Boos Store Website.

- **Security**

I also had some security issues to be resolved.

- Only the user who will login can order for a book and checkout.
- And also the user should not be able to access the Administrative options.

It was easy for me to create the user security options for checkout but I faced problems with protecting Administrative options from the user. But later I was able to figure it out using Google search engine and some videos about C# security.

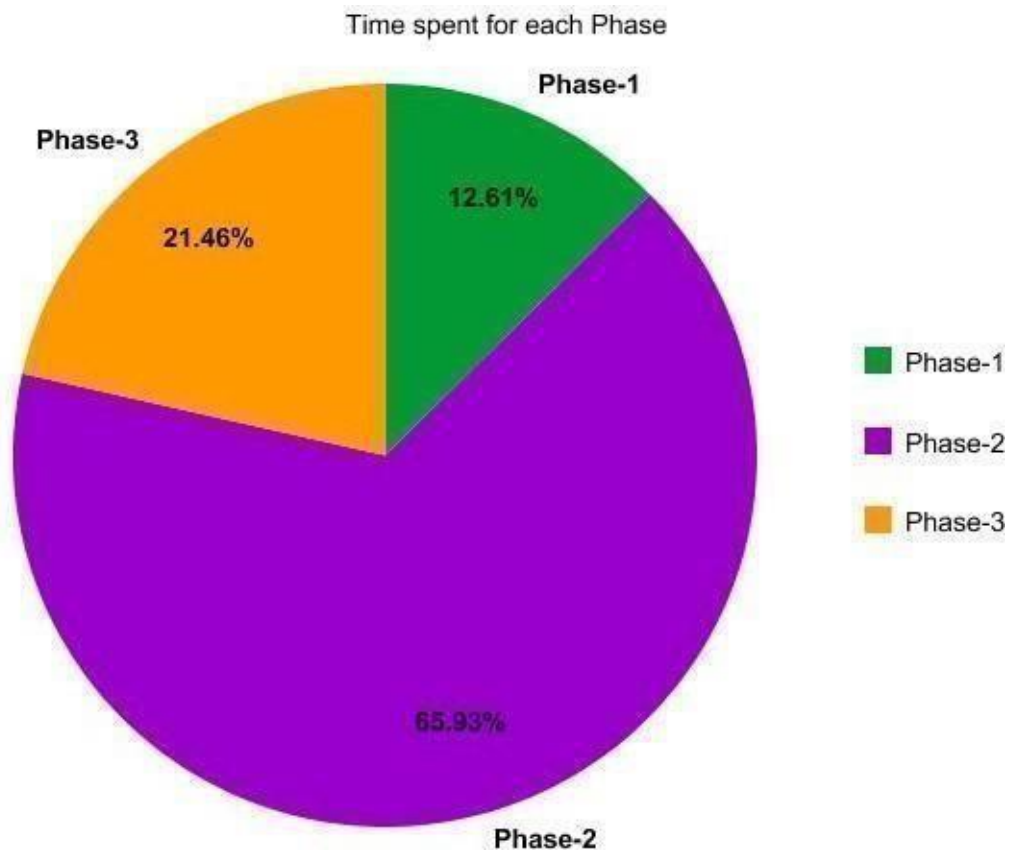
### **3. Metrics**

- \* **Lines of Code**

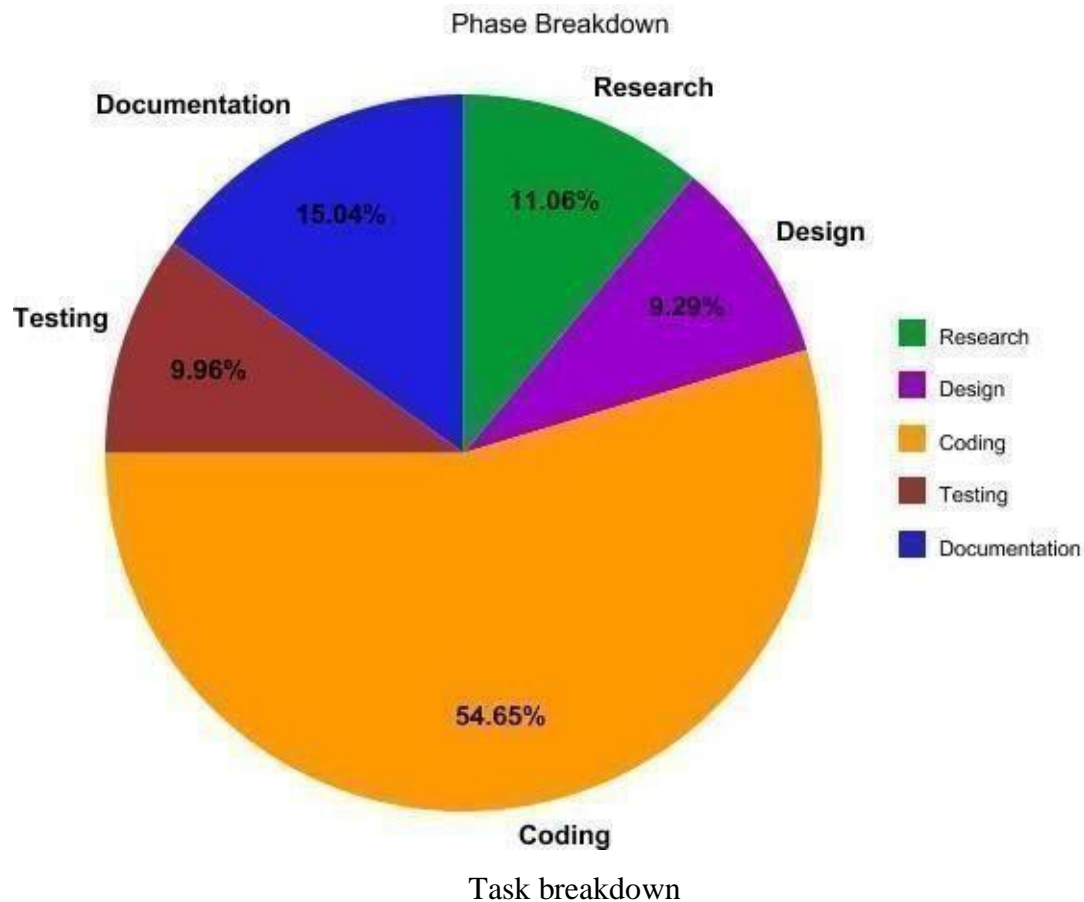
My initial estimate was 3000 LOC during Phase-1.

But now I found the total to be 3203 LOC after the coding part is completed. I used the LOC metrics tool to count the total number of Lines of Code.





Project Phase Schedule



#### 4. Lessons Learned

##### Programming

The Online Book Store Project helped me to improve my confidence level in C# Programming. Though I have made many mistakes during the initial phase I have learnt how to use user controls, master pages, data grid, data set and other data base functionalities.

##### Time Management

Since MSE Project is done as an individual I have learnt how to manage time during the Software Life Cycle Process. I have also learned how to face tense situations and meet the deadlines. This would add as a good experience for me for my future job prospective.

## **UML and Software Lifecycle**

As software student though I have good knowledge in UML and Software LIFE cycle I never had any good practical experience regarding them. Through this project I have learnt how to develop a project following the various stages in Software Life Cycle.

## **Documentation**

I always had a feeling that I am not good at documentation .But through this project and suggestions from my committee members I believe that I have improved my Documentation skills.

## **References**

- *IEEE Recommended Practice for Software Design Descriptions IEEE Std 1016-1998*
- *IEEE Recommended Practice for Software Requirements Specifications - IEEE Std 830-1998*
- *IEEE Standard for Software Test Documentation IEEE Std 829-1998*
- *IEEE Guide for Software Quality Assurance Planning - IEEE Std 730.1-1995*
- <http://www.asp.net/learn/data-access/tutorial-16-vb.aspx>
- [www.glify.com](http://www.glify.com)
- [http://msdn.microsoft.com/en-us/library/system.security.permissions.securitypermissionattribute\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/system.security.permissions.securitypermissionattribute(VS.71).aspx)
- <http://www.locmetrics.com/>
- <http://nces.ed.gov/nceskids/createAGraph/>
- [http://en.wikipedia.org/wiki/Load\\_testing](http://en.wikipedia.org/wiki/Load_testing)
- [http://en.wikipedia.org/wiki/Unit\\_test](http://en.wikipedia.org/wiki/Unit_test)
- <http://www.viveo-oolobject.com/savoirfaire/ecmfinance/concepts/usine/3tiers.php>
- <file:///C:/Documents%20and%20Settings/Owner/Desktop/jakarta-jmeter-2.3RC3/docs/usermanual/build-adv-web-test-plan.html>
- [http://en.wikipedia.org/wiki/Manual\\_testing](http://en.wikipedia.org/wiki/Manual_testing)