ABSTRACT

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 Fantasy, Science, English, Architecture etc.

This online book store is a user-friendly and efficient platform for book lovers to browse, purchase, and receive books of various genres. The website provides a seamless shopping experience to customers, allowing them to search for books using various filters and categories. The bookstore offers fast and reliable delivery, and customer support is always available to assist with any queries or concerns. In addition, the bookstore provides personalized recommendations to customers based on their reading preferences and purchase history. The website's inventory is regularly updated to ensure that customers have access to the latest releases and bestsellers. The online bookstore has generated significant revenue and profitability and has established a loyal customer base.

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. 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. I want to develop a similar website.

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

TABLE OF CONTENT

TABLE OF IMAGES

CHAPTER 1

INTRODUCTION

Online Book Store is the process whereby consumers directly buy books from a seller interactively in real-time without an intermediary service over the internet.

Online Book Store is the process of buying books from merchants who sell on the Internet. Since the emergence of the World Wide Web, merchants have sought to sell their books to people who surf the Internet. Shoppers can visit web stores from the comfort of their homes and shop as they sit in front of the computer. Consumers buy a variety of items from online stores. In fact, people can purchase just about anything from companies that provide their books online. Books are just some of the hundreds of books consumers can buy from an online store.

Many people choose to conduct shopping online because of the convenience. For example, when a person shops at a brick-and-mortar store, she has to drive to the store, find a parking place, and walk throughout the store until she locates the books she needs. After finding the items she wants to purchase, she may often need to stand in long lines at the cash register.

My true purpose for creating the online book is to say "good bye" to long queue and to save time and energy. People can search for the books they looking for easily from anywhere.
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 payment. The Administrator will have additional functionalities when compared to the common user. Interest to develop a good user friendly website using a database. And to increase my knowledge horizon in technologies like SQL, CSS, HTML. to gain good experience in PYTHON before joining in a full-time job. to gain expertise using Data Grid, Data Set, Data Table, Data Adapter and Data Readers.

There are many online book stores like Powell's, Amazon which were designed using Html. I want to develop a similar website. 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 Fiction, Computer, Literature, Architecture etc.

First, I want to create a home page with product categories. 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 and then to create Search bar, A search by keyword option is provided to the user using a textbox The keyword to be entered should be the book title. And to create a book description page where the user can know details about a book he can click on the title from where he will be directed to a Book description page. The user can also see the author’s description by clicking on the author’s name below the book’s name.

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 user rating and the user can also give feedback about the book . Next to create Cart page where 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.

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

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 groups etc.

CHAPTER 2

SYSTEM STUDY

2.1 Existing System

In the present scenario, people have to physically visit the bookshops or vendors for purchasing books of their need and have to make payment through cash mode most of the times due to unawareness of advanced technologies at certain places. In this method time as well as physical work is required, among which time is something that no one has an ample amount. The traditional book purchasing procedure is not efficient enough for shopkeepers as well as customers, as they have to deal with the crowd, in their shops. The old methods are classified into two ways where the most popular one was you need to go to the shop and ask the shopkeeper for the book. If he has that book then he will give you and demand money, which could be more than you thought for. The other one is if you know any retailer or shopkeepers

you can directly contact him on phone and ask him to give you that books at your place and take the money and extra convenience charges of transportation. These both the methods are slow and cost you more sometimes if you are a novice in marketing or purchasing any new thing.

2.1.1 Limitations of Existing System

- Limited Access: The traditional book purchasing system requires physical access to the bookshop or vendor. This means that people who live in remote areas or have mobility issues may not have access to the books they need.
- Inconvenient Payment Options: In the traditional system, customers have to make payments through cash mode most of the time, which may not be convenient for everyone. It may also be risky to carry large sums of cash, especially in crowded areas.
- Time-Consuming: The traditional book purchasing process can be time-consuming, as customers have to physically visit the bookshop or vendor. This can be inconvenient for people who have busy schedules and cannot take time off to visit a bookshop.
- Limited Book Selection: The traditional system may have limited book selections, especially in smaller bookshops or vendors. This can be frustrating for customers who are looking for specific books.

- Lack of Transparency: In the traditional system, customers may not be aware of the actual prices of the books they are purchasing, as there is no transparency in the pricing. This may lead to overcharging and exploitation of customers.

2.2 Proposed System

In this project you can view any book very easily online. You can purchase online books by selecting any payment gateway like pay-pall or money wire.   This is controlled by a admin who have full control over all the books. All the books are stored into the Database. So, there is fill security of theft of payment etc. Online bookstore provides a platform where all the books are kept in an arranged way and you can choose your required book by just selecting it from the store .after selecting book and view all the description such as name of book, name of author and price ,user can make payment of it in order to get it.

An online book store is a great way for avid readers to purchase books from the comfort of their own homes. However, in order to create a seamless and user-friendly experience for customers, it's important to have a robust and efficient system in place. Here are some proposed system features and their advantages for an online book store: User-Friendly Website Design: The website design should be clean, easy to navigate, and intuitive. A well-designed website can lead to increased customer satisfaction, more return visitors, and higher sales conversion rates. Comprehensive Search Functionality: The search function should allow customers to search for books based on title, author, genre, and other criteria. This will help customers find the books they want more easily, which can lead to higher sales and improved customer satisfaction. Robust Product Catalogue: The product catalogue should be comprehensive and up-to-date, with accurate information about each book. This can help customers make informed purchasing decisions and reduce the likelihood of returns or complaints. Secure Payment Gateway: The payment gateway should be secure and reliable, with multiple payment options available to customers. This can help build trust with customers and increase sales conversion rates. Customer Account Management: Customers should be able to create accounts, view their order history, and manage their personal information easily. This can lead to increased customer loyalty and repeat business.

CHAPTER 3

SYSTEM REQUIREMENTS

System requirements are expressed in a software requirement document. The Software requirement is the official statement of what is required of the system developers. This requirement document includes the requirements definition and the requirement specification. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent.

3.1 Hardware Requirements

Hardware requirements for running this project are as follows:

● System	: Pentium IV 2.4 GHz.
● Hard Disk	: 40 GB.
● Monitor	: 15 VGA Color.
● Mouse	: Logitech.
● Ram	: 512 Mb.

3.2 Software Requirements

● Operating system	: Windows 10.
● Front end	: HTML, CSS, JavaScript
● Middleware	: Python(Django)
● Back end	: SQLite

HTML

HTML or **Hyper Text Markup Language** is the standard markup language used to create web pages.

HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent *empty elements* and so are unpaired, for example <img>. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

CSS

CSS was first developed in 1997, as a way for Web developers to define the look and feel of their Web pages. It was intended to allow developers to separate content from design so that HTML could perform more of the function that it was originally based on the markup of content, without worry about the design and layout.

CSS didn't gain in popularity until around 2000, when Web browsers began using more than the basic font and color aspects of CSS.

Web Designers that don't use CSS for their design and development of Web sites are rapidly becoming a thing of the past. And it is arguably as important to understand CSS as it is to know HTML - and some would say it was more important to know CSS.
Style sheet refers to the document itself. Style sheets have been used for document design for years. They are the technical specifications for a layout, whether print or online. Print designers use style sheets to insure that their designs are printed exactly to specifications. A style sheet for a Web page serves the same purpose, but with the added functionality of also telling the viewing engine (the Web browser) how to render the document being viewed.

JAVA SCRIPT

JavaScript often abbreviated as **JS**, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating thirdparty libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of their features support functional programming and aspect-oriented programming (including metaprogramming and metaobjects). Many other paradigms are supported via extensions, including design by contract and logic programming Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution. Its design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. Its core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as: Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Readability counts. Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Django

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in July 2005. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

URLs: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.

Sending the request to the right view (urls.py)

A URL mapper is typically stored in a file named urls.py the mapper (urlpatterns) defines a list of mappings between routes (specific URL patterns) and corresponding view functions. If an HTTP Request is received that has a URL matching a specified pattern, then the associated view function will be called and passed the request. The urlpatterns object is a list of path() and/or re_path() functions (Python lists are defined using square brackets, where items are separated by commas and may have an optional trailing comma)

The first argument to both methods is a route (pattern) that will be matched. The path() method uses angle brackets to define parts of a URL that will be captured and passed through to the view function as named arguments. The second argument is another function that will be called when the pattern is matched.

View: A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via models, and delegate the formatting of the response to templates.

Handling the request (views.py)

Views are the heart of the web application, receiving HTTP requests from web clients and returning HTTP responses. In between, they marshal the other resources of the framework to access databases, render templates, etc.

index(), which could have been called by our URL mapper in the previous section Request object as a parameter (request) and returns an object.

Models: Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.

Defining data models (models.py)

Django web applications manage and query data through Python objects referred to as models. Models define the structure of stored data, including the field types and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc. The definition of the model is independent of the underlying database — you can choose one of several as part of your project settings. Once you've chosen what database you want to use, you don't need to talk to it directly at all — you just write your model structure and other code, and Django handles all the "dirty work" of communicating with the database for you.

Templates: A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A *view* can dynamically create an HTML page using an HTML template, populating it with data from a *model*. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

Forms: Forms are basically used for taking input from the user in some manner and using that information for logical operations on databases.

SQLite

SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems. Many programming languages have bindings to the SQLite library. It generally follows PostgreSQL syntax, but does not enforce type checking by default. This means that one can, for example, insert a string into a column defined as an integer.

- **MySQL is a database management system.**

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as SQLite Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

SQLite was designed to allow the program to be operated without installing a database management system or requiring a database administrator. Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, a linker integrates the SQLite library — statically or dynamically — into an application program which uses SQLite's functionality through simple function calls, reducing latency in database operations; for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of inter-process communication.

Advantages of SQLite

SQLite is a lightweight, cross-platform, transactional, easy-to-use, scalable, and open-source relational database management system. These advantages make it an attractive choice for developers who need a database management system that is easy to use, cost-effective, and reliable.

CHAPTER 4

SYSTEM ARCHITECTURE

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.

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 – HTML, CSS, Web forms, Master Pages, Images.
- Middle tier – Python(Django).
- Data tier- Database

Presentation layer: This layer is responsible for presenting the user interface to the customer. It includes the website or mobile app that customers use to search for books, browse categories, add items to their cart, and place orders.

Application layer: The application layer handles the business logic of the online bookstore. It receives requests from the presentation layer and communicates with the data layer to retrieve and update data. It is responsible for handling customer authentication, order processing, payment processing, and inventory management.

Data layer: The data layer consists of the database management system that stores and manages the data for the online bookstore. It includes the customer database, order database, inventory database, and other data required by the application layer.

To ensure scalability, performance, and reliability, the system architecture should be designed to handle a large volume of traffic and transactions. It should also include appropriate security measures, such as encryption, firewalls, and access controls, to protect customer data and prevent unauthorized access. In conclusion, the system architecture for an online bookstore includes multiple layers that work together to provide a seamless and efficient shopping experience for customers.

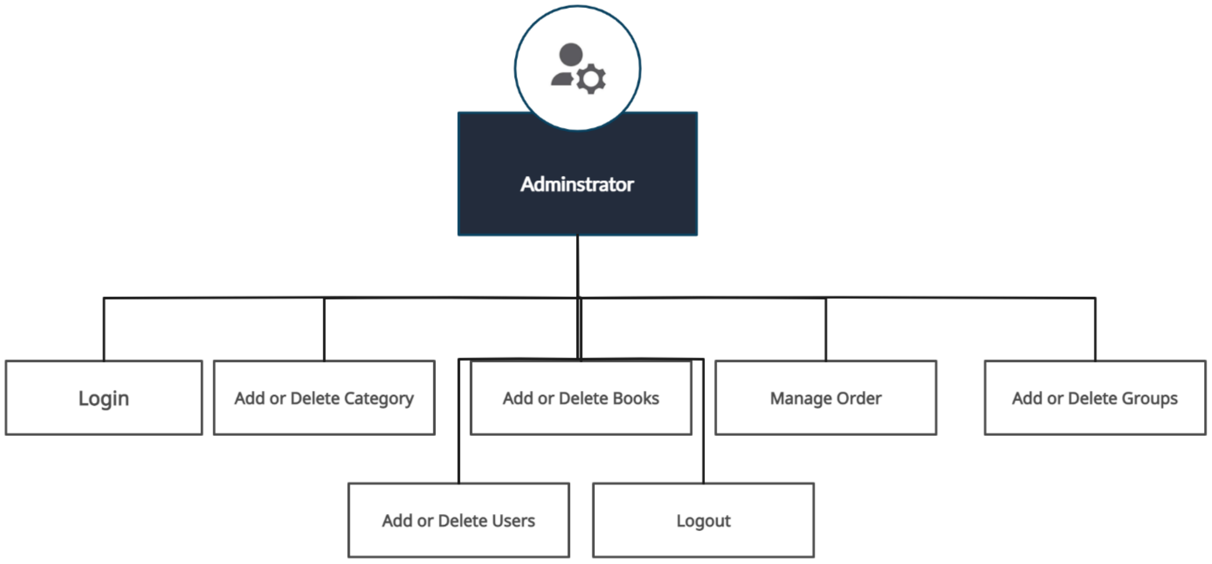


Fig: a. Admin Process Flow Diagram

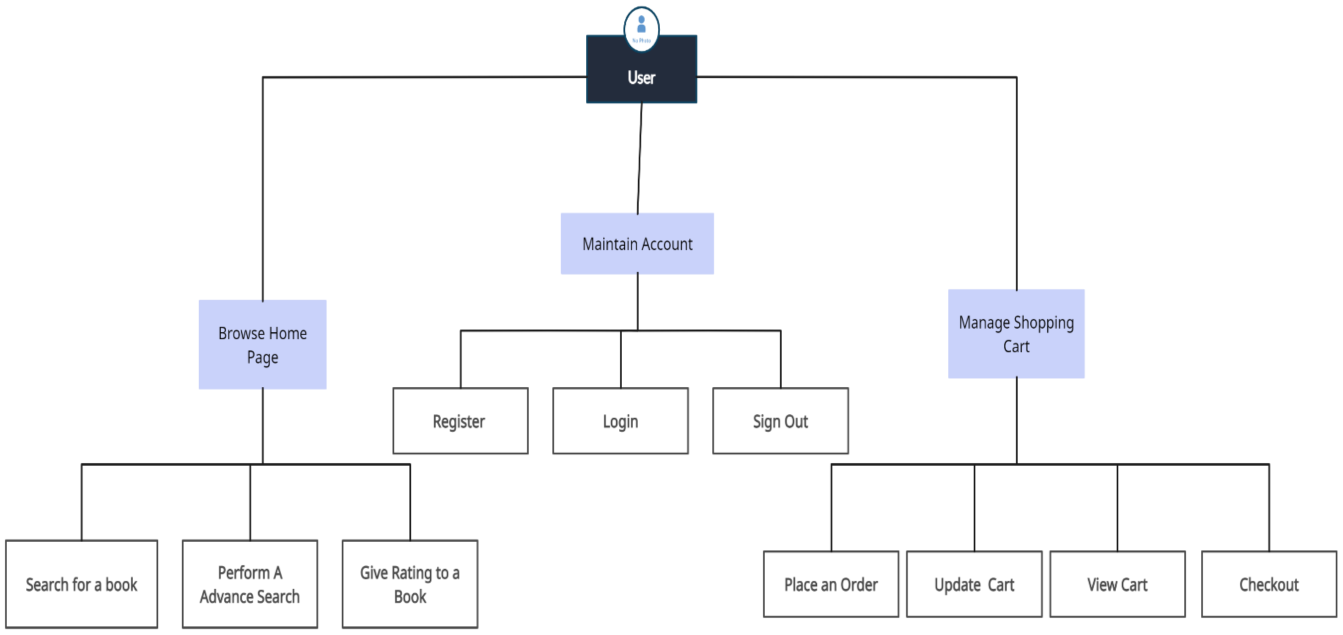


Fig: b. User Process Flow Diagram

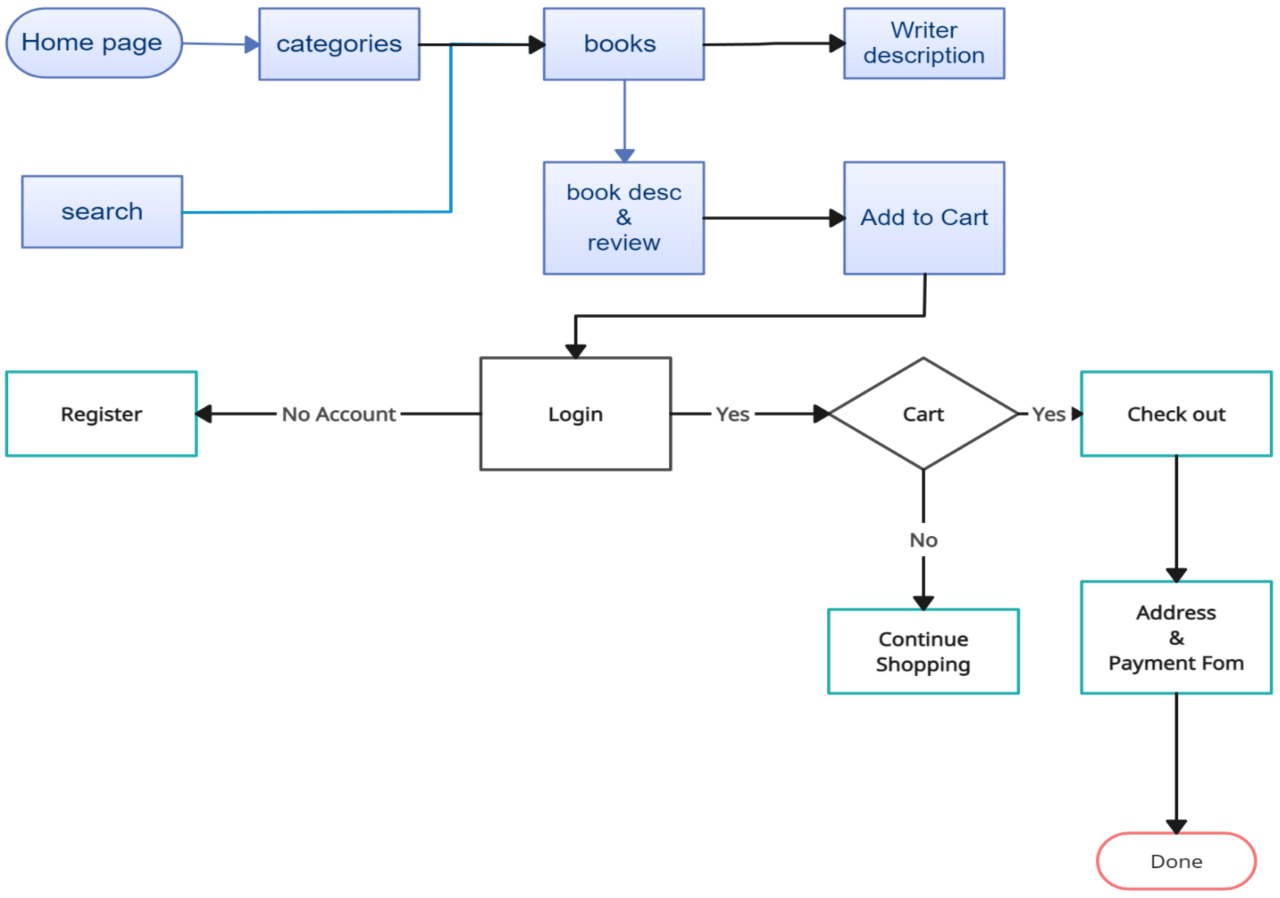
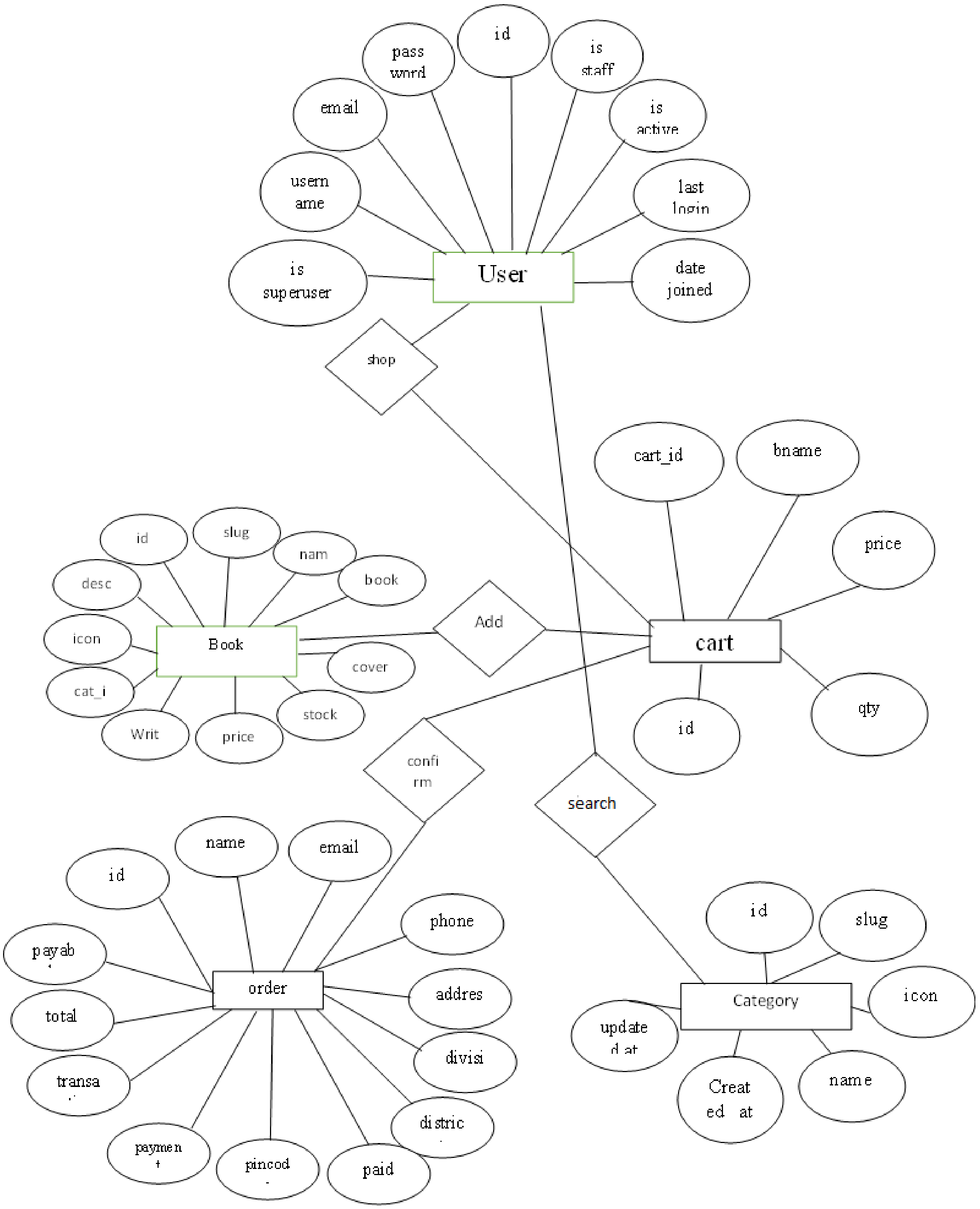


Fig: c. Process Flow Diagram

Fig: d. ER Diagram



CHAPTER 5

MODULES DESCRIPTION

5.1 User

1. **Login Page** – Customer enter their website credentials on this page to gain access in order to log in.
2. **Register Page**– The page where new customer created their login credentials for the website.
3. **Home Page**– When customer visit the website, this is the system's default page. This page shows the books for sale in the store, or by entering a keyword in the search box above the books.
4. **Book View Page** – The page on which the product's specific information is shown, as well as the page on which the customer adds the product to his or her cart.
5. **Cart List Page**– The page that lists the items that customer has chosen. This is the page where the customer can complete the order checkout process.
6. **My Order Page** – The page that lists the customer's orders.

5.2 Admin

1. **Dashboard** – For the admin dashboard, you will be able to all the basic access in the whole system. Such as summary of products, orders, and the categories.
2. **Manage Books**– The admin has access to the books management information system. He can add, update and delete the books.
3. **Manage Categories** – The page where the admin can add, edit and delete categories information.
4. **Manage Orders** – As the main functions of the admin, the admin can accept or reject the order from the customers on a case-to-case basis and the list of customer orders are listed.
5. **Manage User**– The admin can manage the user's account. Admin can add, update and Block user in the system.
6. **Login and Logout** – By default one of the security features of this system is the secure login and logout system.

5.3 Database Tables

User Table

Field Name	**Data Type**	**Size**	**Allow Null**	**Constrain**
Id	Integer	40	No	Primary Key
Username	Varchar	150	No	Unique
Password	Varchar	128	No	
Email	Varchar	254	No	
Is_superuser	Bool		No	
last_login	Datetime			
is_staff	Bool		No	
is_active	Bool		No	
date_joined	Datetime		No	
first_name	Varchar	150	No	

Category Table

Field Name	**Data Type**	**Size**	**Allow Null**	**Constrain**
Id	Integer		No	Primary Key
Name	Varchar	100	No	
Slug	Varchar	150	No	Unique
Icon	Varchar	100	No	
create_at	Datetime		No	
updated_at	Datetime		No	

Book Table

Field Name	Data Type	Size	Allow Null	Constrain
Id	Integer		No	Primary Key
Name	Varchar	100	No	
Slug	Varchar	100	No	
Price	Integer		No	
Stock	Integer		No	
Coverpage	Varchar	100	No	
Bookpage	Varchar	100	No	
Created	Datetime		No	
Updated	Datetime		No	
Totalreview	Integer		No	
Totalrating	Integer		No	
Status	Integer		No	
Description	Text		No	
category_id	Integer		No	Unique
writer_id	Integer		No	Unique

Cart Table

Field Name	Data Type	Size	Allow Null	Constrain
cart_id	Int	4	No	Primary Key
user_name	Varchar	20	No	
Id	Varchar	10	No	Unique
Bname	Varchar	25	No	
Qty	Integer	4	No	
Price	Integer		No	

Review Table

Field Name	Data Type	Size	Allow Null	Constrain
Id	Integer		No	Primary Key
review_star	Integer		No	
review_text	Text		No	
Created	Datetime		No	
customer_id	Integer		No	Unique

Writer Table

Field Name	Data Type	Size	Allow Null	Constrain
Id	Integer		No	Primary Key
Name	Varchar	100	No	
Slug	Varchar	100	No	Unique
Bio	Text		No	
Pic	Varchar	100	No	
create_at	Datetime		No	
updated_at	Datetime		No	

Checkout Table

Field Name	Data Type	Size	Allow Null	Constrain
Id	Integer		No	Primary Key
Name	Varchar	30	No	
Email	Varchar	254	No	
Address	Varchar	150	No	
Division	Varchar	20	No	
District	Varchar	30	No	
Phone	Varchar	16	No	
pin_code	Varchar	30	No	
payment_meth od	Varchar	20	No	
account_no	Varchar	20	No	
transaction_id	Integer		No	Unique
Payeable	Integer		No	
Totalbook	Integer		No	
Created	datetime		No	
Updated	datetime		No	
Paid	Bool		No	
customer_id	Integer		No	Unique

CHAPTER 6

RESULT AND DISCUSSION

Online bookstore has many advantages compared to its counterparts such as physical book store, the online bookstore allows it's user to shop at one place where in physical store the books are scattered at the different places which consumes a lot of time and online bookstore helps in saving that time and it also avoids the problem of unavailability of books at physical store as numerous vendors from different places sell their books at one place. This project is efficient in maintaining users records and can perform operations on it, also reduces the work load on the shop owner of knowing the quantity of books available and which books are available and keeps the records of how many books are purchased and sold.

The main problem encountered during the process was my inexperience in the Django and SQlite and its environment.

It was difficult for me to format the home page of my website using the HTML code. So finally, I choose to use the html div tags which made my work easier. In Django I had no experience so it is so hard to understand how it works. So, I learnt about it using some tutorials which I found using google.

Programming The Online Book Store Project helped me to improve my confidence level in Python and SQL 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. Additionally, my confidence in producing a project has increased.

The online bookstore has many advantages over traditional brick-and-mortar bookstores. It provides customers with the convenience of shopping from anywhere, at any time, and on any device. It eliminates the need for customers to physically visit a store, which can be particularly beneficial for those who live in remote locations or have busy schedules.

CHAPTER 7

CODING

Urls.py Home page And View Books

```
from django.urls import path
from . import views

app_name = 'store'

urlpatterns = [
path('', views.index, name = "index"),
path('login', views.signin, name="signin"),
path('logout', views.signout, name="signout"),
path('registration', views.registration, name="registration"),
path('book/<int:id>', views.get_book, name="book"),
path('books', views.get_books, name="books"),
path('category/<int:id>', views.get_book_category, name="category"),
path('writer/<int:id>', views.get_writer, name = "writer"),
]
```

Urls.py Search

```
from django.urls import path
from . import views

app_name = 'search'

urlpatterns = [
path('', views.search, name = "search"),
]
```

Urls.py Cart

```
from django.urls import path
from . import views

app_name = 'cart'

urlpatterns = [
path('', views.cart_details, name='cart_details'),
path('summary', views.cart_summary, name='cart_summary'),
path('totalcart', views.total_cart, name='totalcart'),
path('add/<int:bookid>', views.cart_add, name='cart_add'),
path('remove/<int:bookid>', views.cart_remove, name='cart_remove'),
path('update/<int:bookid>/<int:quantity>', views.cart_update, name='cart_update'),
]
```

Urls.py Order Page

```
from django.urls import path
from . import views

app_name = 'order'

urlpatterns = [
path('', views.order_list, name="order_list"),
path('<int:id>', views.order_details, name="order_details"),
path('shipping/', views.order_create, name="order_create"),
path('pdf/<int:id>',views.pdf.as_view(), name="pdf"),
]
```

View.py Home page And View Books:

```
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,
        "rreview": rreview
    }
    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)
```

View.py search:

```
from django.shortcuts import render
from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator
from django.db.models import Q
from store.models import Book, Category, Writer

def search(request):
search = request.GET.get('q')
books = Book.objects.all()
if search:
            books = books.filter(
            Q(name__icontains=search)|Q(category__name__icontains=search)|Q(writer__name__ic
ontains=search)

            )

paginator = Paginator(books, 10)
page = request.GET.get('page')
books = paginator.get_page(page)

context = {
            "book": books,
            "search": search,
}
return render(request, 'store/category.html', context)
```

View.py Cart:

```
from django.shortcuts import render, redirect, get_object_or_404
from django.views.decorators.http import require_POST
from store.models import Book, Category
from .cart import Cart

def cart_add(request, bookid):
cart = Cart(request)
book = get_object_or_404(Book, id=bookid)
cart.add(book=book)

return redirect('store:index')

def cart_update(request, bookid, quantity):
```

```
cart = Cart(request)
book = get_object_or_404(Book, id=bookid)
cart.update(book=book, quantity=quantity)
price = (book.price*quantity)

return render(request, 'cart/price.html', {"price":price})

def cart_remove(request, bookid):
    cart = Cart(request)
    book = get_object_or_404(Book, id=bookid)
    cart.remove(book)
    return redirect('cart:cart_details')

def total_cart(request):
return render(request, 'cart/totalcart.html')

def cart_summary(request):

return render(request, 'cart/summary.html')

def cart_details(request):
cart = Cart(request)
context = {
            "cart": cart,
}
return render(request, 'cart/cart.html', context)
```

View.py Order:

```
from django.shortcuts import HttpResponse, render, redirect, get_object_or_404
from django.contrib.auth.models import User
from django.contrib import messages
from django.views import View
from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator
from cart.cart import Cart
from .models import Order, OrderItem
from .forms import OrderCreateForm
from .pdfcreator import renderPdf

def order_create(request):
cart = Cart(request)
if request.user.is_authenticated:
    customer = get_object_or_404(User, id=request.user.id)
    form = OrderCreateForm(request.POST or None, initial={"name": customer.first_name,
"email": customer.email})
    if request.method == 'POST':
        if form.is_valid():
            order = form.save(commit=False)
            order.customer = User.objects.get(id=request.user.id)
            order.payable = cart.get_total_price()
            order.totalbook = len(cart) # len(cart.cart) // number of individual book
            order.save()

            for item in cart:
                OrderItem.objects.create(
                    order=order,
                    book=item['book'],
                    price=item['price'],
                    quantity=item['quantity']
                    )
            cart.clear()
```

```
                return render(request, 'order/successfull.html', {'order': order})

            else:
                messages.error(request, "Fill out your information correctly.")

        if len(cart) > 0:
            return render(request, 'order/order.html', {"form": form})
        else:
            return redirect('store:books')
    else:
        return redirect('store:signin')

def order_list(request):
    my_order = Order.objects.filter(customer_id = request.user.id).order_by('-created')
    paginator = Paginator(my_order, 5)
    page = request.GET.get('page')
    myorder = paginator.get_page(page)

    return render(request, 'order/list.html', {"myorder": myorder})

def order_details(request, id):
    order_summary = get_object_or_404(Order, id=id)

    if order_summary.customer_id != request.user.id:
        return redirect('store:index')

    orderedItem = OrderItem.objects.filter(order_id=id)
    context = {
        "o_summary": order_summary,
        "o_item": orderedItem
}
return render(request, 'order/details.html', context)

class pdf(View):
   def get(self, request, id):
       try:
          query=get_object_or_404(Order,id=id)
       except:
          Http404('Content not found')
       context={
          "order":query
       }
       article_pdf = renderPdf('order/pdf.html',context)
       return HttpResponse(article_pdf,content_type='application/pdf')
```

Model.Py Homepage and View Book

```
from django.db import models
from django.contrib.auth.models import User

class Category(models.Model):
name = models.CharField(max_length = 100)
slug = models.SlugField(max_length = 150, unique=True ,db_index=True)
icon = models.FileField(upload_to = "category/")
create_at = models.DateTimeField(auto_now_add = True)
updated_at = models.DateTimeField(auto_now_add = True)

def __str__(self):
          return self.name

class Writer(models.Model):
```

```
name = models.CharField(max_length = 100)
slug = models.SlugField(max_length=150, unique=True ,db_index=True)
bio = models.TextField()
pic = models.FileField(upload_to = "writer/")
create_at = models.DateTimeField(auto_now_add = True)
updated_at = models.DateTimeField(auto_now_add = True)

def __str__(self):
        return self.name

class Book(models.Model):
writer = models.ForeignKey(Writer, on_delete = models.CASCADE)
category = models.ForeignKey(Category, on_delete = models.CASCADE)
name = models.CharField(max_length = 100)
slug = models.SlugField(max_length=100, db_index=True)
price = models.IntegerField()
stock = models.IntegerField()
coverpage = models.FileField(upload_to = "coverpage/")
bookpage = models.FileField(upload_to = "bookpage/")
created = models.DateTimeField(auto_now_add=True)
updated = models.DateTimeField(auto_now=True)
totalreview = models.IntegerField(default=1)
totalrating = models.IntegerField(default=5)
status = models.IntegerField(default=0)
description = models.TextField()

def __str__(self):
   return self.name

class Review(models.Model):
```

```
customer = models.ForeignKey(User, on_delete = models.CASCADE)
book = models.ForeignKey(Book, on_delete = models.CASCADE)
review_star = models.IntegerField()
review_text = models.TextField()
created = models.DateTimeField(auto_now_add=True)

class Slider(models.Model):
title = models.CharField(max_length=150)
created = models.DateTimeField(auto_now_add=True)
updated = models.DateTimeField(auto_now=True)
slideimg = models.FileField(upload_to = "slide/")

def __str__(self):
        return self.title
```

Model.py Order:

```
from django.db import models
from django.contrib.auth.models import User

class Category(models.Model):
name = models.CharField(max_length = 100)
slug = models.SlugField(max_length = 150, unique=True ,db_index=True)
icon = models.FileField(upload_to = "category/")
create_at = models.DateTimeField(auto_now_add = True)
updated_at = models.DateTimeField(auto_now_add = True)

def __str__(self):
        return self.name

class Writer(models.Model):
```

```
name = models.CharField(max_length = 100)
slug = models.SlugField(max_length=150, unique=True ,db_index=True)
bio = models.TextField()
pic = models.FileField(upload_to = "writer/")
create_at = models.DateTimeField(auto_now_add = True)
updated_at = models.DateTimeField(auto_now_add = True)

def __str__(self):
        return self.name

class Book(models.Model):
writer = models.ForeignKey(Writer, on_delete = models.CASCADE)
category = models.ForeignKey(Category, on_delete = models.CASCADE)
name = models.CharField(max_length = 100)
slug = models.SlugField(max_length=100, db_index=True)
price = models.IntegerField()
stock = models.IntegerField()
coverpage = models.FileField(upload_to = "coverpage/")
bookpage = models.FileField(upload_to = "bookpage/")
created = models.DateTimeField(auto_now_add=True)
updated = models.DateTimeField(auto_now=True)
totalreview = models.IntegerField(default=1)
totalrating = models.IntegerField(default=5)
status = models.IntegerField(default=0)
description = models.TextField()

def __str__(self):
   return self.name

class Review(models.Model):
```

```
customer = models.ForeignKey(User, on_delete = models.CASCADE)
book = models.ForeignKey(Book, on_delete = models.CASCADE)
review_star = models.IntegerField()
review_text = models.TextField()
created = models.DateTimeField(auto_now_add=True)

class Slider(models.Model):
title = models.CharField(max_length=150)
created = models.DateTimeField(auto_now_add=True)
updated = models.DateTimeField(auto_now=True)
slideimg = models.FileField(upload_to = "slide/")

def __str__(self):
        return self.title
```

HTML Template :

Home page & View Books

Header:

```
% load staticfiles %}
<div class="header-top">
        <div class="container-fluid">
                <div class="row">
                        <div class="col-sm-3">
                                <div class="pull-left">
  <a href="{% url 'store:index' %}"><img src="{% static 'images/logo/logo.png' %}" alt=""
                                                  /></a>
</div>
</div>
<div class="col-sm-5">
<form action="{% url 'search:search' %}" method="GET">
<div class="input-group search_box">
```

```
<input type="text" class="form-control" placeholder="Search Book" name="q"
value="{{search}}">
<button class="btn btn-primary" type="submit"><i class="fa fa-search"></i></button>
</div>
</form>
</div>
<div class="col-sm-4">
<div class="pull-right">
<ul class="navbar-nav">
{% if request.user.is_authenticated %}
<li class="dropdown profile"><a href="#">{{ request.user.username }}</a>
<ul role="menu" class="sub-menu">
<li><a href="{% url 'order:order_list' %}"><i class="fa fa-shopping-cart"></i> My
order</a></li>
<li><a href="{% url 'store:signout' %}"><i class="fa fa-sign-out"></i> Sing Out</a></li>
</ul>
</li>
<li class="cart">
<a href="{% url 'cart:cart_details' %}">Cart <i class="fa fa-angle-left"></i>
<span class="totalcart" id="gettotalcart"> {{ cart|length }} </span>
<i class="fa fa-angle-right"></i>
</a>
</li>
{% else %}
<li class="profile"><a href="{% url 'store:signin' %}">Sign In</a></li>
<li class="cart">
<a href="{% url 'cart:cart_details' %}">Cart <i class="fa fa-angle-left"></i>
<span class="totalcart" id="gettotalcart"> {{ cart|length }} </span>
<i class="fa fa-angle-right"></i>
</a>
```

```
</li>
                                                {% endif %}
                                            </ul>
                                        </div>
                                    </div>
                                </div>
                        </div>
</div>
```

Side bar:

```
                        <ul class="list-group list-group-flush">
                                <li class="list-group-item"><h5>Category</h5></li>
                                {% for p in cat %}
                                <li class="list-group-item"><a href="{% url 'store:category' id=p.id
%}">{{p.name}}</a></li>
                                {% endfor %}

                        </ul>
```

Footer

```
{% load static %}
<div class="footer">
        <div class="container">
                <div class="row">
                        <div class="col-sm-6 col-md-4 col-xs-12 footer_menu">
                                <h3>Contact</h3>
                                <ul>
                                        <li class="footer_contact">
                                                <i class="fa fa-phone"></i>
                                                <p>  +91 000000000 </p>
                                        </li>
                                        <li class="footer_contact">
                                <i class="fa fa-envelope"></i>
                                <p> FullMarks@gmail.com </p>
                            </li>
                            <li class="footer_contact">
                                <i class="fa fa-map-marker"></i>
                                <p>தமிழ்நாடு India</p>
                            </li>
                        </ul>
                    </div>

                     <div class="col-sm-6 col-md-4 col-xs-12 footer_menu">
                        <h3>Menu</h3>
                        <ul>
                            <li class="footer_menu">
                                <a href="{% url 'store:books' %}"><i class="fa fa-
angle-right"></i>Books</a>
                            </li>
                        </ul>
                    </div>

                    <div class="col-sm-6 col-md-4 col-xs-12 footer_news">
                        <h3>We accept payment via</h3>
                        <div class="footer_paymentmethod">
                            <img src="{% static 'images/bkash.png' %}" alt="" />
                            <img src="{% static 'images/rocket.png' %}" alt="" />
                        </div>
                    </div>
                </div>
            </div>
        </div>
```

Base

```
<!DOCTYPE html>
<html>
<head>
		<title>Online Book Store</title>
		{% load staticfiles %}
		<script src="https://code.jquery.com/jquery-2.2.0.min.js" type="text/javascript"></script>
		<link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
		<link rel="stylesheet" type="text/css" href="{% static 'css/font-awesome.min.css' %}">
		<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
		<link rel="stylesheet" type="text/css" href="{% static 'css/responsive.css' %}">
		<link rel="stylesheet" type="text/css" href="{% static 'css/slick.css' %}">
		<link rel="stylesheet" type="text/css" href="{% static 'css/slick-theme.css' %}">
		<link rel="stylesheet" type="text/css" href="{% static 'css/skdslider.css' %}">
</head>
<body>
<div class="pre_loader gray_bg text-center">
    <div class="loader">
        <div><div><div><div></div></div></div></div>
    </div>
</div>
{% include 'store/bin/header.html' %}

{% block headermenu %} {% endblock %}

<div class="container-fluid">
		<div class="row">
				<div class="col-sm-2">
						{% include 'store/bin/sidebar.html' %}
				</div>
```

```
                <div class="col-sm-10">
                        {% block container %}{% endblock %}
                </div>

        </div>
</div>
<div id="snackbar"></div>

{% include 'store/bin/footer.html' %}

<script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.3.1/js/bootstrap.min.js"
type="text/javascript"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-touchspin/4.2.5/jquery.bootstrap-
touchspin.js" type="text/javascript"></script>
<script src="{% static 'js/skdslider.min.js' %}"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick.min.js"
type="text/javascript"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Readmore.js/2.2.1/readmore.js"
type="text/javascript"></script>
<script src="https://www.google-analytics.com/analytics.js" type="text/javascript"></script>
<script src="{% static 'js/main.js' %}"></script>
{% block scripts %}{% endblock %}
</body>
</html>
```

Book

```
{% extends 'store/base.html' %}
{% load static %}
{% load crispy_forms_tags %}
{% load customfunction %}
{% block container %}
```

```
                    <div class="row">
                        <div class="col-sm-12">
                            <div class="row">
                                <div class="col-sm-3">
                                    <div class="grid-item">
                                        <div                class="view-details"
style="cursor:pointer">
                                            <img     id="product_image"
class="lookInsideImg" src="{{ book.coverpage.url }}">
                                        </div>
                                    </div>
                                    <div  class="lookInsideDiv"  style="display:
none;">
                                        <div class="exitBtn"></div>
                                        <div class="pagesArea">
                                            <ul          class="list-unstyled
pages">
                                                <li><img       src="{{
book.bookpage.url }}"></li>
                                            </ul>
                                        </div>
                                    </div>
                                </div>

                                <div class="col-sm-6">
                                    <div class="product-information">
                                        <h4>{{ book.name }}</h4>
                                        <p    style="margin:   0px">by    <a
style="font-weight:bold"  href="{%  url  'store:writer'  id=book.writer.id  %}">{{  book.writer
}}</a></p>
```

```
                                        <div class='rating'>
                                            {{
book.totalrating|averagerating:book.totalreview }}
                                            <a    href='#getreview'><span
class='totalrating'>{{ book.totalreview|add:-1 }} Reviews</span></a>
                                        </div>
                                        <article                class="text-justify"
style="margin-top:10px;">
                                            {{ book.description }}
                                        </article>

                                    </div>
                                </div>
                                <div class="col-sm-3">
                                    <div class="pricebox">
                                        <p>Price:  $.  <span>  {{  book.price
}}</span></p>
                                        <button    class="btn    btn-warning"
id="addTocart" data-book-id="{{ book.id }}">
                                            <i    class="fa    fa-shopping-
cart""></i>Add to cart
                                        </button>
                                    </div>
                                </div>
                            </div>
                        </div>

                    <div class="col-sm-9 mt-5" id="getreview">
                            {% if messages %}
                                <div class="messages">
```

```
                                        {% for message in messages %}
                                        <p{% if message.tags %} class="{{ message.tags
}}"{% endif %}>{{ message }}</p>
                                        {% endfor %}
                                    </div>
                               {% endif %}
                               <div class="rating-stars text-left">
                                    <ul id="stars">
                                        <li   id="azaa"   class="star"   data-value="1">   <i
class="fa fa-star fa-fw"></i> </li>
                                        <li   id="azbb"   class="star"   data-value="2">   <i
class="fa fa-star fa-fw"></i> </li>
                                        <li   id="azcc"   class="star"   data-value="3">   <i
class="fa fa-star fa-fw"></i> </li>
                                        <li   id="azdd"   class="star"   data-value="4">   <i
class="fa fa-star fa-fw"></i> </li>
                                        <li   id="azee"   class="star"   data-value="5">   <i
class="fa fa-star fa-fw"></i> </li>
                                    </ul>
                               </div>
                           <form    action=""    method="POST"    class="needs-validation"
novalidate="">
                               {% csrf_token %}

                                    <div class="form-group">
                                         {{ form.review_star|as_crispy_field }}
                                    </div>
                                    <div class="form-group">
                                         {{ form.review_text|as_crispy_field }}
                                    </div>
```

```
                                        <div class="form-group">
                                                <button         class="btn          btn-warning"
type="submit" style="color: #fff">Submit </button>
                                        </div>
                                </form>
                                   {% for lreview in rreview %}
                                   <div class="card">
                                        <div class="card-body">
                                                <img   src="{%   static   'images/profile.png'
%}"/>
                                                <b> {{ lreview.customer }} </b>
                                                <div class='rating'>
                                                        {{
lreview.review_star|averagerating:1 }}
                                                </div>
                                                <div class="given_review_date">
                                                        <span      style="color:#555;">      {{
lreview.created }}</span>
                                                </div>
                                                <p      style="margin-top:      10px">      {{
lreview.review_text }} </p>
                                         </div>
                                   </div>
                                   {% endfor %}

                                   <div class="d-pagination">
                                     <ul class="pagination">
                                        {% if rreview.has_previous %}
                                                <li class="page-item">
                                        <a                      class="page-link"
href="?page=1">First</a>
                                    </li>

                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
rreview.previous_page_number }}">Previous</a>
                                    </li>
                                {% endif %}
                                {% for ord in rreview.paginator.page_range %}
                                    {% if rreview.number == ord %}
                                        <li class="page-item active">
                                            <span    class="page-link">{{
ord }}
                                                <span          class="sr-
only">(current)</span>
                                            </span>
                                        </li>
                                    {% elif ord > rreview.number|add:'-3' and ord
< rreview.number|add:'3' %}
                                        <li class="page-item">
                                            <a              class="page-link"
href="?page={{ ord }}">{{ ord }}</a>
                                        </li>

                                    {% endif %}

                                {% endfor %}
                                 {% if rreview.has_next %}
                                    <li class="page-item">
```

```
                                                                <a class="page-link" href="?page={{
rreview.next_page_number }}">Next</a>
                                                        </li>
                                                        <li class="page-item">
                                                                <a class="page-link" href="?page={{
rreview.paginator.num_pages }}">Last</a>
                                                        </li>
                                                {% endif %}
                                        </ul>
                                </div>
                        </div>
                </div>

{% endblock %}



{% block scripts %}
<script type="text/javascript">
        $(document).on('ready', function() {
                $(document).on('click', '.lookInsideImg,.lookInsideBg', function (e) {
                        $('div.overlay').fadeIn(500);
                        $('div.lookInsideDiv').fadeIn(500);
                });

          $(document).on('click', 'div.lookInsideDiv div.exitBtn', function () {
                        $('div.overlay').fadeOut(500);
                        $('div.lookInsideDiv').fadeOut(500);
                        refCaro($('#bookDR99'));
                });
                $(document).on('click', 'div.overlay', function () {
                        $('div.overlay').fadeOut(500);
                        $('div.lookInsideDiv').fadeOut(500);
                        refCaro($('#bookDR99'));
                        return false
                });
        });
</script>

{% endblock %}
```

Category

```
{% extends 'store/base.html' %}
{% load customfunction %}
{% block container %}
                        <div class="row">
                                {% for item in book %}
                                <div class = 'col-sm-3'>
                                        <div class="book-wrapper text-center">
                                                <div class="coverpage">
                                                        <img src="{{ item.coverpage.url }}"/>
                                                </div>
                                                <a  href="{%  url  'store:book'  id=item.id  %}">{{
item.name }}</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-warning"   id="addTocart"
data-book-id="{{ item.id }}">
                                                        <i          class="fa          fa-shopping-
cart""></i>Add to cart
                                                </button>
                                        </div>
                                </div>
                                {% endfor %}

                        </div>
                        {% if book|length > 0 %}
                        <div class="d-pagination">
                          <ul class="pagination">
                                {% if book.has_previous %}
                                        <li class="page-item">
                                                <a class="page-link" href="?page=1">First</a>
                                        </li>
                                        <li class="page-item">
                                                <a           class="page-link"           href="?page={{
book.previous_page_number }}">Previous</a>
                                        </li>
                                {% endif %}
                                {% for ord in book.paginator.page_range %}
                                        {% if book.number == ord %}
                                                <li class="page-item active">
                                                        <span class="page-link">{{ ord }}
                                                                <span                              class="sr-
only">(current)</span>
                                                        </span>
```

```
                                                    </li>
                                            {% elif ord > book.number|add:'-3' and ord <
book.number|add:'3' %}
                                                    <li class="page-item">
                                                            <a class="page-link" href="?page={{ ord
}}">{{ ord }}</a>
                                                    </li>

                                            {% endif %}

                                    {% endfor %}
                                     {% if book.has_next %}
                                            <li class="page-item">
                                                    <a class="page-link" href="?page={{
book.next_page_number }}">Next</a>
                                            </li>
                                            <li class="page-item">
                                                    <a class="page-link" href="?page={{
book.paginator.num_pages }}">Last</a>
                                            </li>
                                    {% endif %}
                               </ul>
                            </div>
                            {% else %}
                            <h3 class="text-center mt-5">There are no books Found.</h3>
                            {% endif %}

{% endblock %}
```

Index

```
{% extends 'store/base.html' %}
{% load static %}
{% load customfunction %}
{% block headermenu %}
<div class="header-bottom">
        <div class="container-fluid">
                <div class="row">
                        <div class="col-md-12">
                                <div class="responsive categorymenu">
                                        {% for p in cat %}
                                        <div class="cat-wrapper">
                                                <div class="text-center">
                                                        <a href="{% url 'store:category' id=p.id
%}"><img src="{{ p.icon.url }}"/></a>
                                                        <a href="{% url 'store:category' id=p.id
%}">{{ p.name }}</a>
                                                </div>
                                        </div>
                                        {% endfor %}
                                </div>
                        </div>
                </div>
        </div>
</div>
{% endblock %}

                        {% block container %}
                        <div class="row">
                                <div class="col-sm-12">
                                        <div style="margin:0 auto;">
                                                <ul id="demo3">
                                                        {% for sld in slide %}
                                                        <li><img height="100" src="{{
sld.slideimg.url }}"/></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">
                                                                <img src="{{ item.coverpage.url
}}"/>
                                                        </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-warning"
id="addTocart" data-book-id="{{ item.id }}">
                                                                <i    class="fa    fa-shopping-
cart""></i>Add to cart
                                            </button>
                                                </div>
                                                {% endfor %}
                                            </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">
                                                        <img src="{{ p.coverpage.url }}"/>
                                                    </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-warning"
id="addTocart" data-book-id="{{ p.id }}">
                                            <i    class="fa    fa-shopping-
cart""></i>Add to cart
                                        </button>
                                </div>
                                {% endfor %}
                            </div>
                        </div>
                    </div>
                    {% endblock %}
```

Login

```
<!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-
warning">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 account</a>
            </div>
```

```
                </div>
        </div>
</div>

</body>
</html>
```

Sign up

```
<!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>
                                        {% endfor %}
                                        <button class="btn btn-warning">Create you BookStore
account</button>
                                </form>
                                <div class="a-divider-break"></div>
                                <div class="a-divider-inner">
                                        <div class="a-row">
                                          Already have an account?
                                          <a class="a-link-emphasis" href="{% url 'store:signin'
%}"><i class='fa fa-caret-right'></i> Sign in</a>
                                        </div>
                                </div>
                        </div>
                </div>
        </div>
</div>
</body>
</html>
```

Writer

```
{% extends 'store/base.html' %}
{% load static %}
{% load customfunction %}
{% block container %}
            <div class="row">
                <div class="col-sm-12">
```

```
                              <div class="row">
                                   <div class="col-sm-3">
                                        <div class="view-details">
                                             <img src="{{ wrt.pic.url }}">
                                        </div>
                                   </div>

                                   <div class="col-sm-8">
                                        <div class="product-information">
                                             <h3>{{ wrt.name }}</h3>
                                             <article          class="text-justify"
style="margin-top:10px;">{{ wrt.bio }}</article>

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

                         <div class="col-sm-12">
                              <div class="titleheader">
                                   <h3     class="h2header     text-center">Published
Books</h3>
                              </div>
                              <div class="regulara sliderzx">
                                   {% for item in book %}
                                   <div class="book-wrapper text-center">
                                        <div class="coverpage">
                                             <img    src="{{    item.coverpage.url
}}"/>
                                        </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-warning" id="addTocart" data-book-id="{{ item.id }}">
                                        <i class="fa fa-shopping-cart""></i>Add to cart
                                    </button>


                            </div>
                            {% endfor %}
                        </div>
                    </div>
                </div>
{% endblock %}
```

Checkout

```
{% 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 %}>{{ message }}</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 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.pin_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-warning 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 %}
Pdf.html
<!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.pin_code }}, {{
order.district }}, {{ order.address }}</h1>
                     </div>
             </div>
                </div>
        </div>
</body>
</html>
```

CHAPTER 9

SCREENSHOT

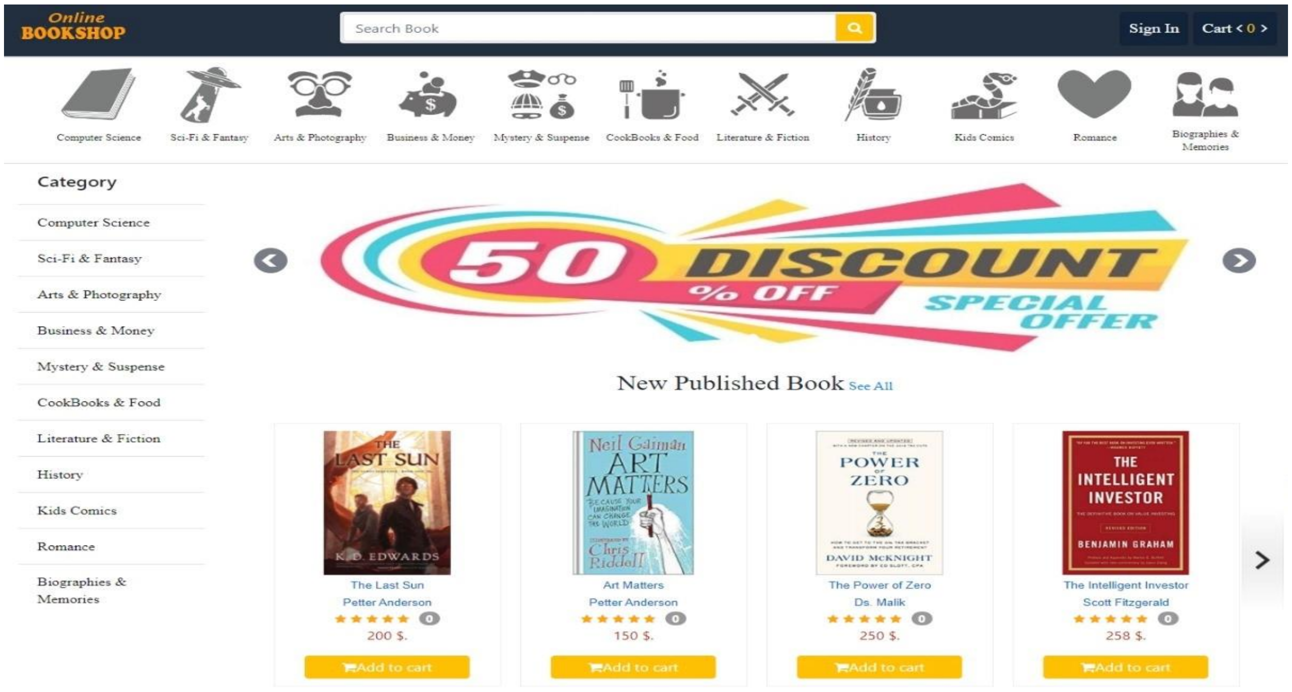


Fig: e. Home Page

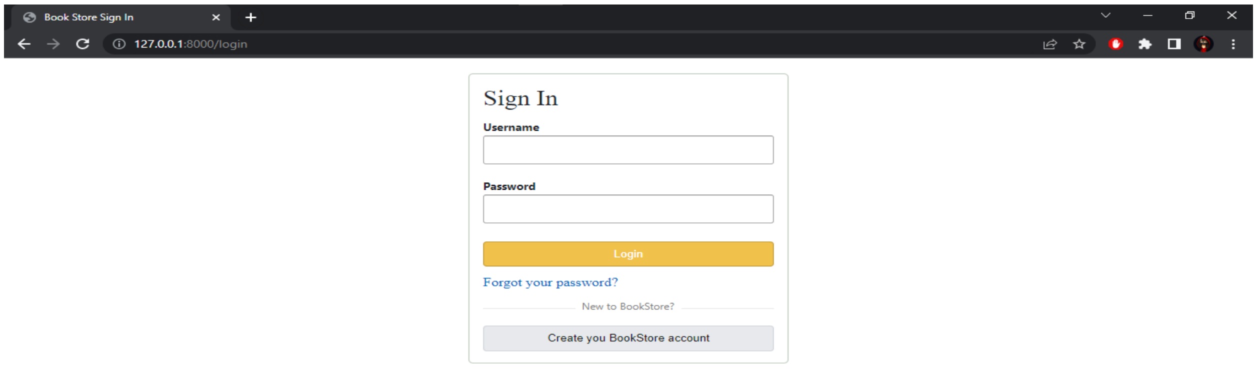
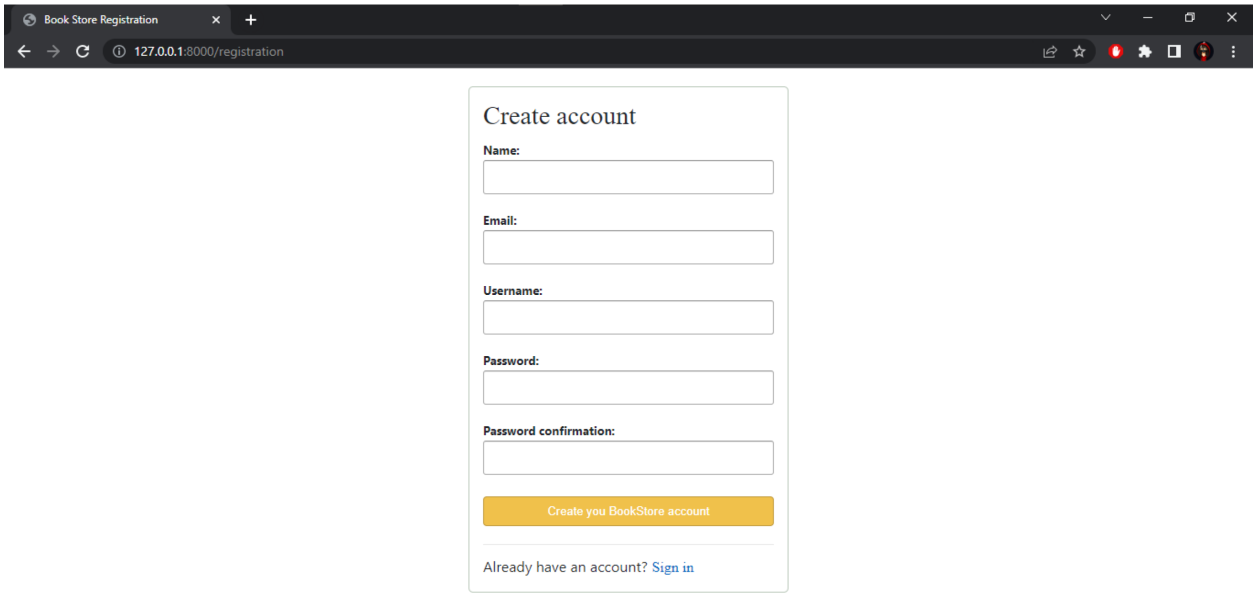


Fig: f. Login Page



Create account

Name:

Email:

Username:

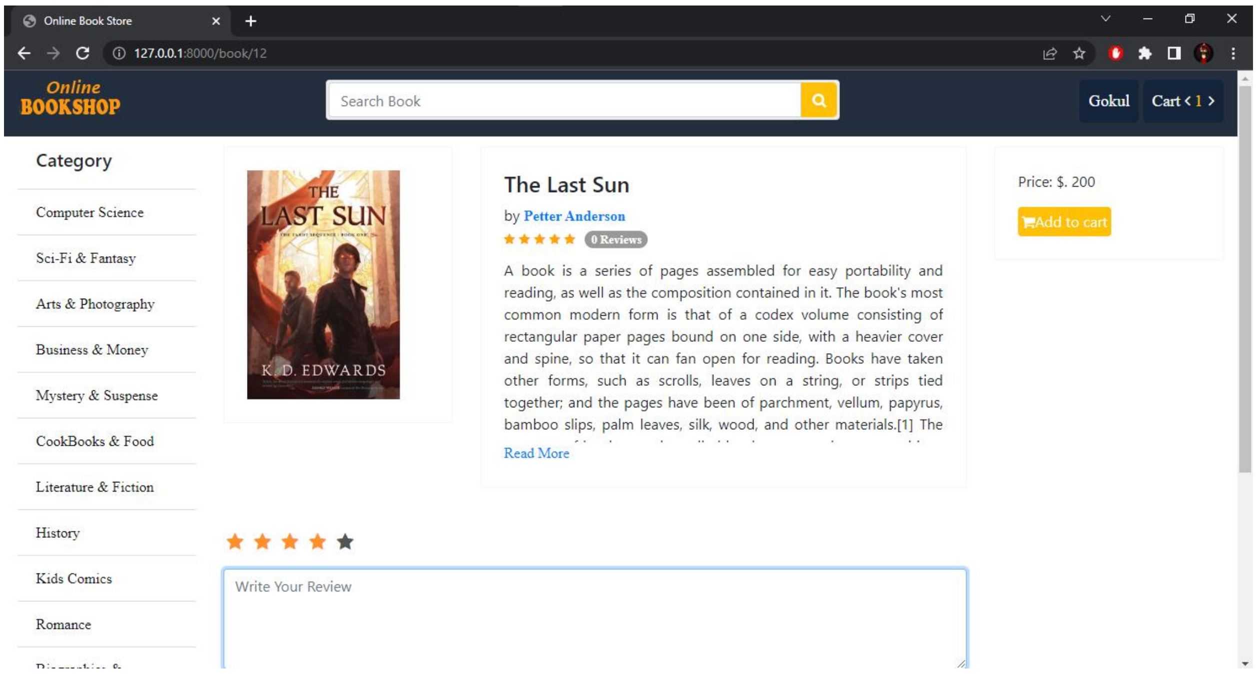
Password:

Password confirmation:

Create you BookStore account

Already have an account? Sign in

Fig: g. Registration Page



Category

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- Mystery & Suspense
- CookBooks & Food
- Literature & Fiction
- History
- Kids Comics
- Romance

The Last Sun

by Petter Anderson

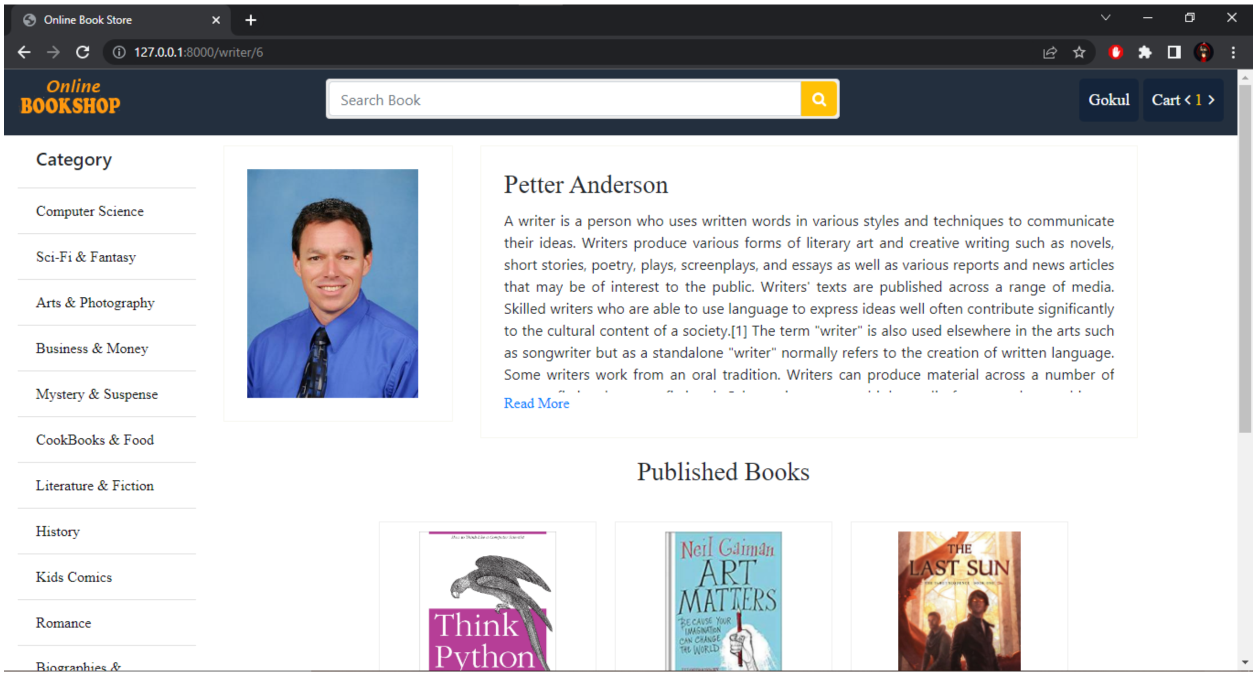
Price: $. 200

A book is a series of pages assembled for easy portability and reading, as well as the composition contained in it. The book's most common modern form is that of a codex volume consisting of rectangular paper pages bound on one side, with a heavier cover and spine, so that it can fan open for reading. Books have taken other forms, such as scrolls, leaves on a string, or strips tied together; and the pages have been of parchment, vellum, papyrus, bamboo slips, palm leaves, silk, wood, and other materials.[1] The

Read More

Fig : h. Book Description

& Review Page

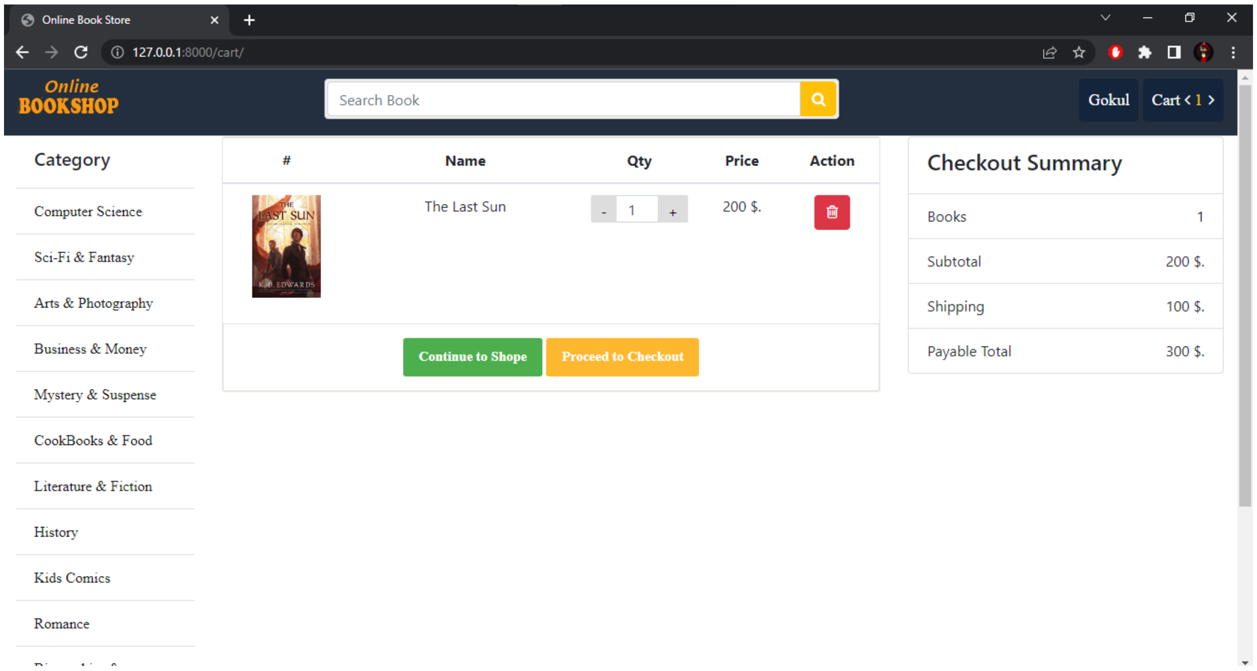


Petter Anderson

A writer is a person who uses written words in various styles and techniques to communicate their ideas. Writers produce various forms of literary art and creative writing such as novels, short stories, poetry, plays, screenplays, and essays as well as various reports and news articles that may be of interest to the public. Writers' texts are published across a range of media. Skilled writers who are able to use language to express ideas well often contribute significantly to the cultural content of a society.[1] The term "writer" is also used elsewhere in the arts such as songwriter but as a standalone "writer" normally refers to the creation of written language. Some writers work from an oral tradition. Writers can produce material across a number of

Published Books

Fig: i. Writer Description Page

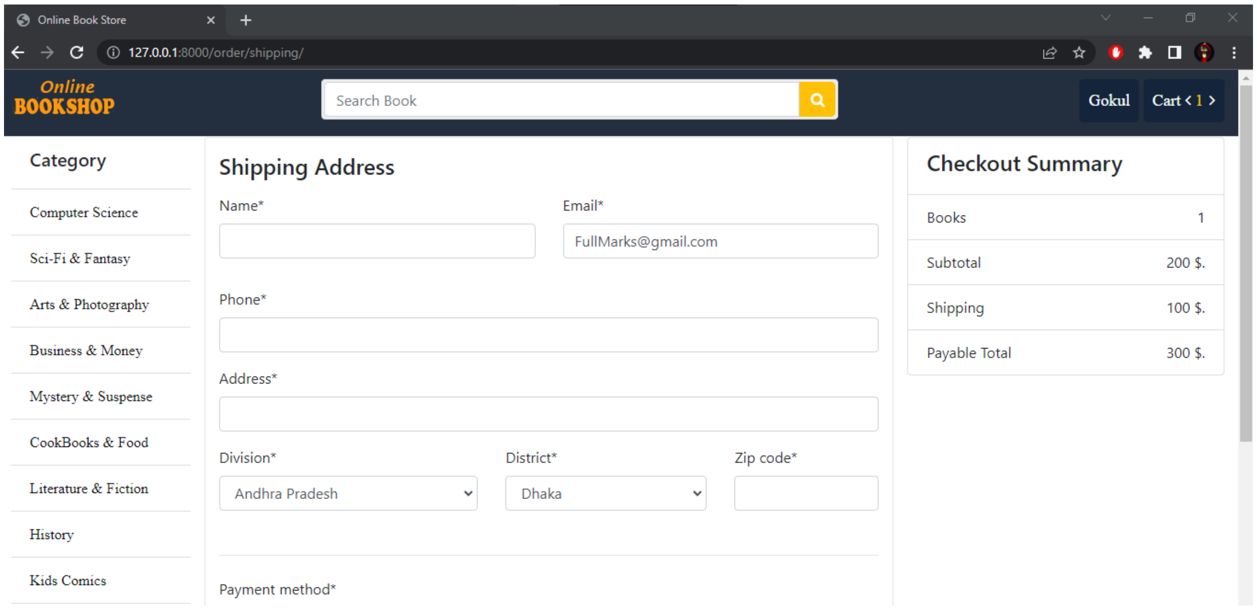


#	Name	Qty	Price	Action
	The Last Sun	1	200 $.	

Checkout Summary

Books	1
Subtotal	200 $.
Shipping	100 $.
Payable Total	300 $.

Fig: j. Cart Page



Category

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- Mystery & Suspense
- CookBooks & Food
- Literature & Fiction
- History
- Kids Comics

Shipping Address

Name*

Email* FullMarks@gmail.com

Phone*

Address*

Division* Andhra Pradesh

District* Dhaka

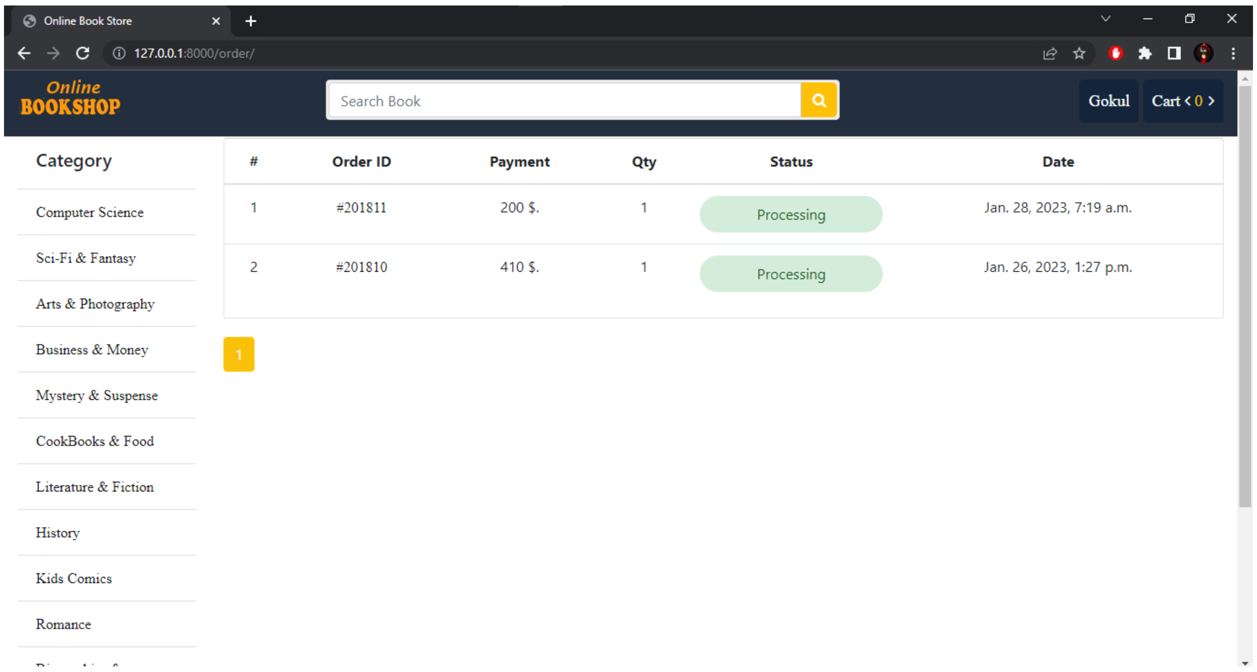
Zip code*

Payment method*

Checkout Summary

Books	1
Subtotal	200 $.
Shipping	100 $.
Payable Total	300 $.

Fig: k. Check Out Page



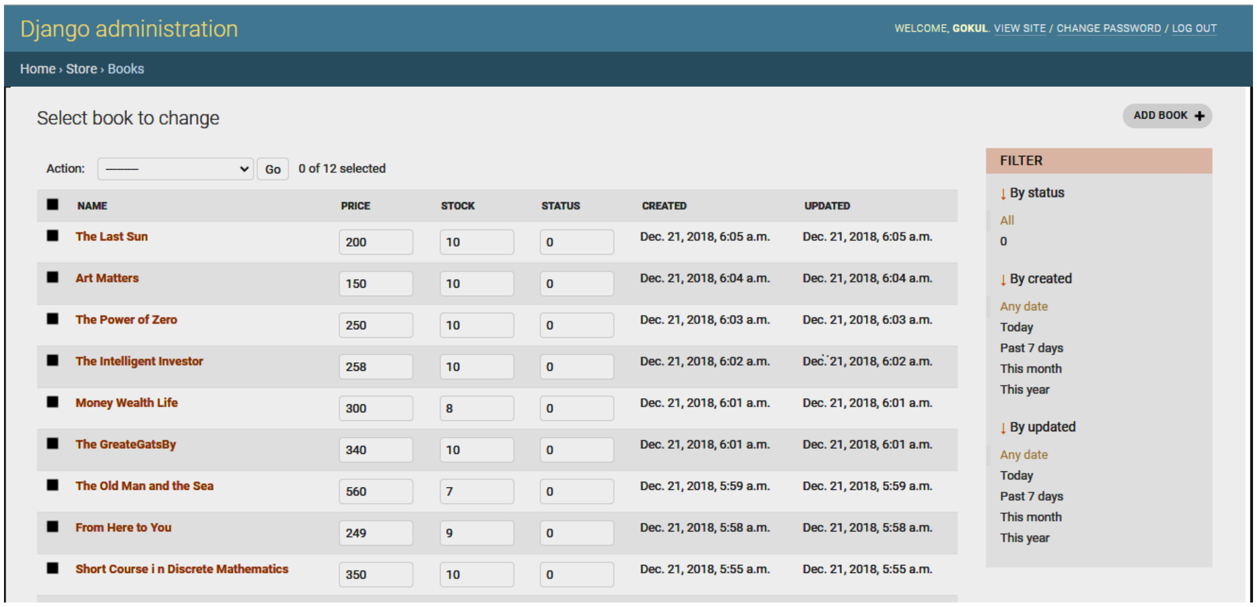
Category

- Computer Science
- Sci-Fi & Fantasy
- Arts & Photography
- Business & Money
- Mystery & Suspense
- CookBooks & Food
- Literature & Fiction
- History
- Kids Comics
- Romance

#	Order ID	Payment	Qty	Status	Date
1	#201811	200 $.	1	Processing	Jan. 28, 2023, 7:19 a.m.
2	#201810	410 $.	1	Processing	Jan. 26, 2023, 1:27 p.m.

1

Fig: l. Order Page


Django administration

WELCOME, **GOKUL**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Store › Books

Select book to change

ADD BOOK +

Action: ——— Go 0 of 12 selected

■	NAME	PRICE	STOCK	STATUS	CREATED	UPDATED
■	The Last Sun	200	10	0	Dec. 21, 2018, 6:05 a.m.	Dec. 21, 2018, 6:05 a.m.
■	Art Matters	150	10	0	Dec. 21, 2018, 6:04 a.m.	Dec. 21, 2018, 6:04 a.m.
■	The Power of Zero	250	10	0	Dec. 21, 2018, 6:03 a.m.	Dec. 21, 2018, 6:03 a.m.
■	The Intelligent Investor	258	10	0	Dec. 21, 2018, 6:02 a.m.	Dec. 21, 2018, 6:02 a.m.
■	Money Wealth Life	300	8	0	Dec. 21, 2018, 6:01 a.m.	Dec. 21, 2018, 6:01 a.m.
■	The GreateGatsBy	340	10	0	Dec. 21, 2018, 6:01 a.m.	Dec. 21, 2018, 6:01 a.m.
■	The Old Man and the Sea	560	7	0	Dec. 21, 2018, 5:59 a.m.	Dec. 21, 2018, 5:59 a.m.
■	From Here to You	249	9	0	Dec. 21, 2018, 5:58 a.m.	Dec. 21, 2018, 5:58 a.m.
■	Short Course i n Discrete Mathematics	350	10	0	Dec. 21, 2018, 5:55 a.m.	Dec. 21, 2018, 5:55 a.m.

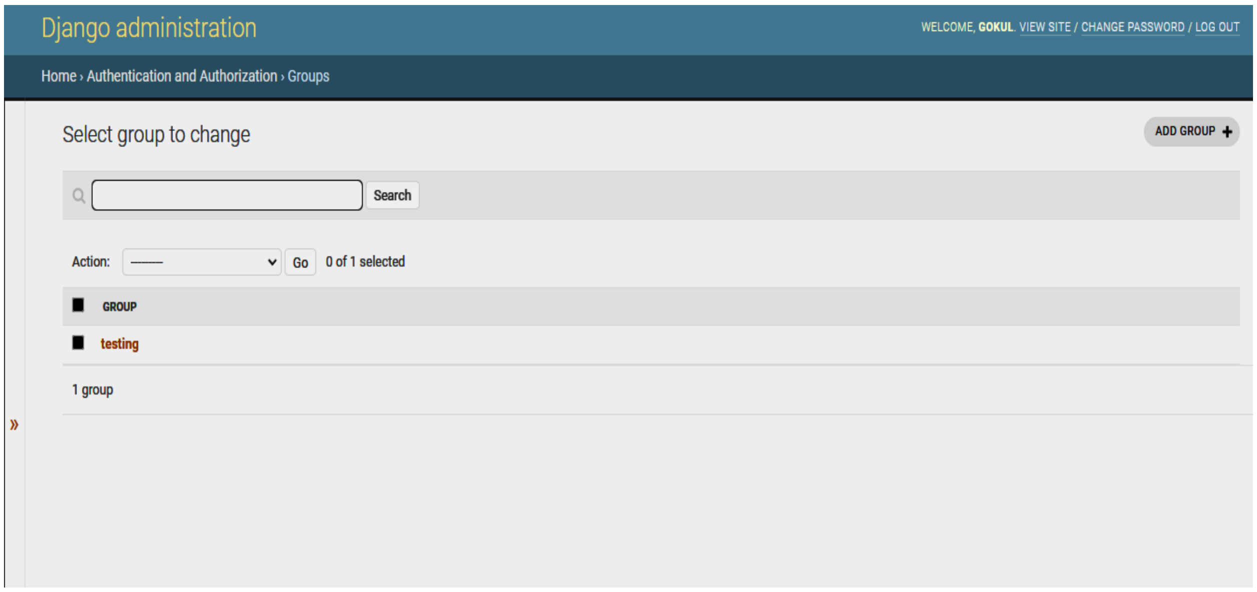
FILTER

↓ By status
All
0

↓ By created
Any date
Today
Past 7 days
This month
This year

↓ By updated
Any date
Today
Past 7 days
This month
This year

Fig: m. Book Edit Page


Django administration

WELCOME, **GOKUL**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Groups

Select group to change

ADD GROUP +

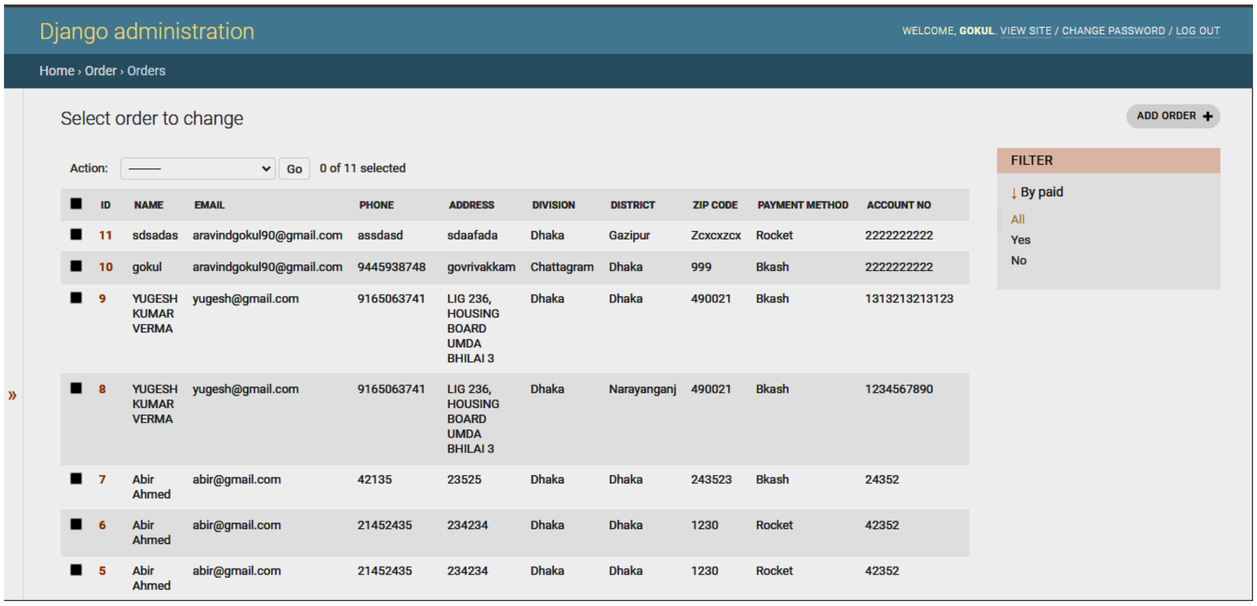
Search

Action: ——— Go 0 of 1 selected

■	GROUP
■	testing

1 group

Fig: n. Group Edit page

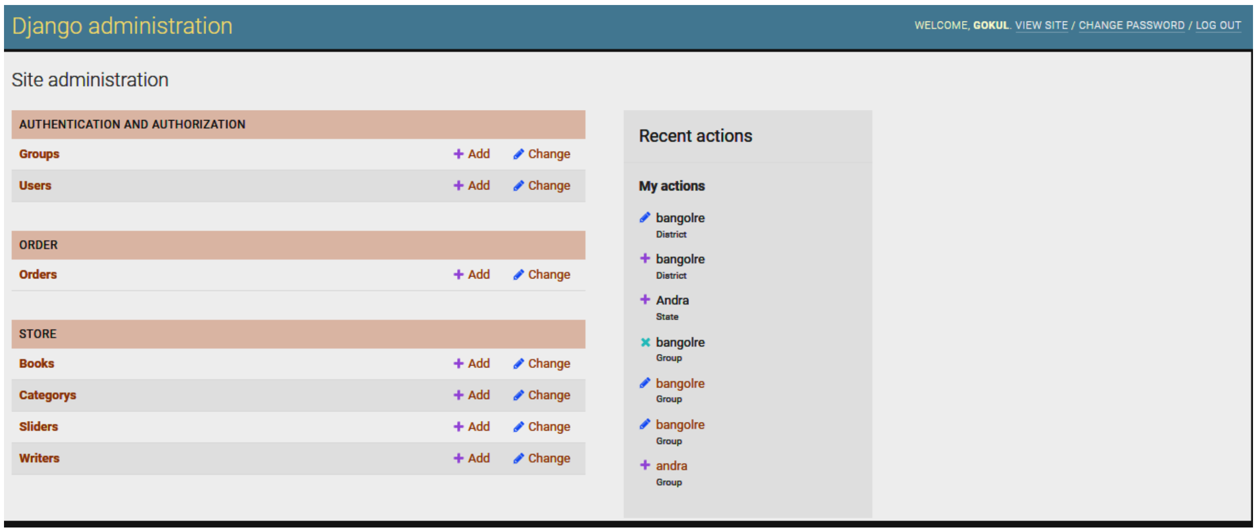


Select order to change

0 of 11 selected

	ID	NAME	EMAIL	PHONE	ADDRESS	DIVISION	DISTRICT	ZIP CODE	PAYMENT METHOD	ACCOUNT NO
■	11	sdsadas	aravindgokul90@gmail.com	assdasd	sdaafada	Dhaka	Gazipur	Zcxcxzcx	Rocket	2222222222
■	10	gokul	aravindgokul90@gmail.com	9445938748	govrivakkam	Chattagram	Dhaka	999	Bkash	2222222222
■	9	YUGESH KUMAR VERMA	yugesh@gmail.com	9165063741	LIG 236, HOUSING BOARD UMDA BHILAI 3	Dhaka	Dhaka	490021	Bkash	1313213213123
■	8	YUGESH KUMAR VERMA	yugesh@gmail.com	9165063741	LIG 236, HOUSING BOARD UMDA BHILAI 3	Dhaka	Narayanganj	490021	Bkash	1234567890
■	7	Abir Ahmed	abir@gmail.com	42135	23525	Dhaka	Dhaka	243523	Bkash	24352
■	6	Abir Ahmed	abir@gmail.com	21452435	234234	Dhaka	Dhaka	1230	Rocket	42352
■	5	Abir Ahmed	abir@gmail.com	21452435	234234	Dhaka	Dhaka	1230	Rocket	42352

Fig: o. Order Edit Page



Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

ORDER		
Orders	+ Add	Change

STORE		
Books	+ Add	Change
Categorys	+ Add	Change
Sliders	+ Add	Change
Writers	+ Add	Change

Recent actions

My actions

- bangolre District
- + bangolre District
- + Andra State
- ✗ bangolre Group
- bangolre Group
- bangolre Group
- + andra Group

Fig: p. Admin Dashboard

CHAPTER 9

CONCLUSION

The switch from written books being from bookstores to being ordered online or even just digital copies has had profound effects on the industry including bookstores and libraries and the general people of the world. The positives include easy access for everyone and cheaper books along with saving natural resources. The negatives however are much greater and cannot be ignored. Some of the negatives include the loss of jobs and businesses through the digitalization of books, the loom threat of Google books, as well as losing a sense of our past. Books have played such a large role in human history that it is bittersweet to be replacing them. To me, it seems that e-books, Google books, and online retailers are the future and that the future is good. But, if you look past all the conveniences and look at the people being affected by this then it becomes clear that this is not a future that I want.

The framework is a general interest for perusing and books on the necessities of the buyer client base, with a specific level of functional online book shop framework. It is essentially not quite the same as the conventional physical book shops; to defeat a progression of physical book shops restricted assortment, fixed area, constrained space and tight deals channels and different issues, for individuals to buy the book has brought accommodation. Online book shop framework not exclusively can without much of a stretch discover the data and buy books, and the working conditions are basic, easy to understand, to a substantial degree to take care of genuine issues in the buy of books, however the plan of the framework as a result of the fruition time Of the limitations, a portion of the elements of the framework isn't impeccable, yet the essential capacities have been accomplished.

Personalized Recommendations: The online book store can incorporate machine learning algorithms to provide personalized recommendations to customers based on their purchase history and reading preferences. Social Integration: Integration with social media platforms can enhance the online book store's marketing strategy. Customers can share their purchase and reading experience with friends and family, which can lead to increased sales and customer engagement. Audiobooks and E-books: The online book store can expand its offerings to include audiobooks and e-books.

CHAPTER 10

REFERENCES

- Django Forms - GeeksforGeeks www.geeksforgeeks.org
- www.djangoproject.com
- Django introduction - Learn web development | MDN developer.mozilla.org
- https://www.youtube.com/watch?v=fuSadH8fr4U&ab_channel=Payilagam
- https://www.w3schools.com/w3css/defaulT.asp
- https://www.geeksforgeeks.org/javascript/
- https://www.youtube.com/watch?v=OXGznpKZ_sA&ab_channel=freeCodeCamp.org