CHINMAYA ARTS AND SCIENCE COLLEGE FOR WOMEN

GOVINDAGIRI, CHALA, KANNUR

(Affiliated to Kannur University and approved by Govt. of Kerala)

SIXTH SEMESTER BCA

2024-25

PROJECT REPORT

EON

Prepared and Presented by

DEVIKA V

AALIYA HASHIQUE

FATHIMATH SHEBA C

CHINMAYA ARTS AND SCIENCE COLLEGE FOR WOMEN

GOVINDAGIRI, CHALA, KANNUR

CERTIFICATE

This is to certify that the project entitled 'EON' is a bonafide report of the project work done by Devika V (Reg-No CW22BCAR07), Aaliya Hashique Cotticollen(Reg-No CW22BCAR01), Fathimath Sheba C(Reg-No CW22BCAR08) carried out under the supervision of Ms Anitha Haridas, towards partial fulfillment of the award of BCA degree at Kannur University.

Guide: Ms Anitha Haridas **Head of the Department Ms Anitha Haridas**

External Examiners: **Principal**

1.

2.

Submitted for practical examination held on .....……

CHINMAYA ARTS AND SCIENCE COLLEGE FOR WOMEN

GOVINDAGIRI, CHALA, KANNUR

DECLARATION

We, Devika V (Reg-No CW22BCAR07), Aaliya Hashique Cotticollen(Reg-No CW22BCAR01), Fathimath Sheba C(Reg-No CW22BCAR08) VI Semester BCA students of Chinmaya Arts and Science College for Women, do hereby declare that the project report entitled 'EON' is the original work carried out by me under the supervision of Ms Anitha Haridas towards the partial fulfillment for the requirement of BCA Degree at Kannur University.

Signature and Name of the student:

Date of Submission: .....

Kannur:

ABSTRACT

Our project aims to create a user-friendly online platform that brings together book management, user-generated content, and collaborative learning. The platform simplifies the process of registering, managing, and purchasing books while also building a community through "thoughtwaves"—a feature for sharing blogs, stories, and discussions.

A standout feature of the platform is the ability for administrators to provide study materials, which customers can enrich by adding multiple-choice questions (MCQs) related to specific chapters. These questions form quizzes, which are dynamically created by randomly selecting customer-submitted questions, allowing users to track their progress with scores.

The platform also includes a marketplace where customers can buy and sell both new and used books. Extra features, such as an audio player for online books and a chatbot for instant support, make the platform even more accessible and engaging. Sellers can easily manage their inventories, while administrators handle categories, users, orders, and study materials.

By automating processes like user authentication, order tracking, and feedback management, the platform ensures a smooth and secure experience for all users. With its innovative approach and focus on user needs, it provides an efficient solution for book management, learning, and community building.

ACKNOWLEDGMENT

I humbly acknowledge the divine guidance that led me to complete the project. The support and timely direction received greatly contributed to the project's success. Special thanks to Dr. Seema M Thayil, our principal, for granting us permission and facilitating the project. Gratitude to Ms. Anitha Haridas, HOD of the Computer Application Department, for her assistance. We extend our thanks to Ms. Anitha Haridas for her invaluable guidance and continuous cooperation. Our guide's advice played a crucial role in the successful completion of the project. We also thank our guide, Mr. Abhijith of Riss Technology, Kannur for his guidance and valuable suggestions in bringing out this project successfully. We also appreciate the encouragement and assistance from friends, family, and Ms. Shamna, the Lab assistant. Thanks to colleagues for their cooperation and support.

CONTENTS

1. INTRODUCTION

EON

EON is an innovative platform designed to simplify the way people manage books, engage with study materials, and participate in a vibrant community of readers and learners. Our platform integrates book management, user-generated content, and collaborative learning into a seamless experience. EON allows users to purchase, sell, and manage books—whether new, used, or digital—effortlessly. It caters to diverse user needs, from academic resources to leisure reading, and empowers users to contribute by creating and sharing thoughtwaves, including blogs, stories, and discussions. A standout feature of EON is the ability for administrators to upload study materials while enabling customers to enhance these resources by adding chapter-specific multiple-choice questions (MCQs). These MCQs dynamically form quizzes, fostering active engagement and progress tracking through scores. For a more inclusive experience, EON features an audio player for online books, enabling accessibility for diverse users. A chatbot provides instant support, ensuring that users navigate the platform seamlessly. Administrators can oversee categories, users, orders, and study materials, while sellers benefit from tools to manage their inventories efficiently. Whether you're a student, a book enthusiast, or a professional seller, EON transforms the way books are managed and knowledge is shared, creating a one-stop solution for learning, community engagement, and commerce.

OBJECTIVE

The primary objective of EON is to create a unified platform that bridges the gap between book management, collaborative learning, and community engagement. It enables sellers to manage their inventories and customers, and allow administrators to oversee categories, study materials, orders, and user interactions. It also provides administrators with tools to upload study materials and enable users to enrich these resources by adding MCQs for dynamic quizzes that track learning progress. Introduce features like an audio player for online books and a chatbot for immediate support, ensuring a user-friendly and accessible experience. And foster a vibrant community through the thoughtwaves feature, encouraging users to share blogs, stories, and discussions while connecting with like-minded individuals. It simplifies the buying and selling of books, including new and used ones, ensuring reliable and secure transactions for all users.

2. SYSTEM STUDY

2.1 SCOPE

The scope of EON lies in revolutionizing the way books are managed, knowledge is shared, and learning is facilitated. It caters to a diverse audience including students, educators, book enthusiasts, and professional sellers by providing a unified platform that supports book transactions, collaborative learning, and community engagement. The platform includes tools for buying and selling books, managing inventories, accessing study materials, creating and participating in dynamic quizzes, and contributing to a vibrant community through blogs, stories, and discussions. Features like an audio player and chatbot ensure inclusivity and seamless user experience, making EON a comprehensive solution for academic and leisure purposes.

2.2 PRELIMINARY STUDY

The preliminary study involved analyzing existing platforms for book management, study material sharing, and collaborative learning to identify gaps and challenges. Current platforms often lack integration, requiring users to rely on multiple tools for different needs. Surveys and user feedback highlighted the need for a unified system that simplifies book transactions, enhances learning with interactive features like quizzes, and fosters a community for sharing ideas and resources. This study laid the foundation for designing EON as a user-centric, all-in-one platform.

2.3 EXISTING SYSTEM

Existing systems typically focus on isolated functionalities such as online bookstores, learning management systems, or content-sharing platforms. These systems often fail to provide a cohesive experience, leading to inefficiencies in book transactions, limited interactivity in learning resources, and fragmented community engagement. Additionally, accessibility features such as audio support and real-time assistance are often inadequate, leaving a significant gap for diverse user needs.

2.4 PROPOSED SYSTEM

EON addresses the limitations of existing systems by offering an integrated platform that combines book management, collaborative learning, and community interaction. Key features include a marketplace for new and used books, tools for administrators to upload and enhance study materials, dynamic MCQ-based quizzes for progress tracking, an audio player for accessible book reading, and a chatbot for instant support. The platform promotes active engagement through the thoughtwaves feature, enabling users to share blogs and participate in discussions. With robust administrative controls

and efficient inventory management tools, EON provides a seamless and inclusive experience for all users.

2.5 FEASIBILITY STUDY

Feasibility studies are conducted with the objective of gaining a comprehensive understanding of the problem at hand. This phase helps to refine and crystallize the defined problem, identifying the specific aspects that will be incorporated into the system. Moreover, it enables a more accurate estimation of costs and benefits. By conducting these feasibility studies, organizations can make informed decisions about the viability and potential success of a proposed project before committing significant resources.

ECONOMICAL FEASIBILITY

EON is economically viable due to its scalable business model that includes revenue streams from book sales, subscriptions, and premium features. Initial investment costs are offset by the potential for high user adoption and recurring income from sellers and users. The platform's affordability ensures accessibility to a wide audience, further enhancing its economic sustainability.

TECHNICAL FEASIBILITY

The platform leverages modern web technologies and cloud infrastructure to ensure scalability, reliability, and performance. With tools and frameworks for secure transactions, efficient content management, and real-time interactions, EON is designed to handle high traffic and provide a seamless user experience. Integration of APIs for audio playback and chatbot functionality ensures technical robustness.

OPERATIONAL FEASIBILITY

EON is operationally feasible, with a user-friendly interface and intuitive features that require minimal training for users. The platform includes efficient tools for administrators and sellers to manage their tasks effortlessly, reducing operational complexities. Its design ensures smooth onboarding and engagement, making it practical for widespread adoption.

3. SOFTWARE REQUIREMENT SPECIFICATION

HARDWARE SPECIFICATION

Choosing the right hardware is crucial for the proper functioning of any software. The size and capacity of the hardware must align with the specific requirements of the software to ensure optimal performance. Mismatched or inadequate hardware can lead to performance issues and system failures, emphasizing the importance of thoughtful hardware selection for a seamless software experience.

- **Processor**: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00GHz
- **Memory**: 4 GB RAM, 256 GB SSD

SOFTWARE REQUIREMENTS

Choosing software for a system is challenging, involving matching identified requirements with suitable packages. The task is to determine if a specific software aligns with and addresses the system's needs effectively, requiring careful assessment for a seamless integration.

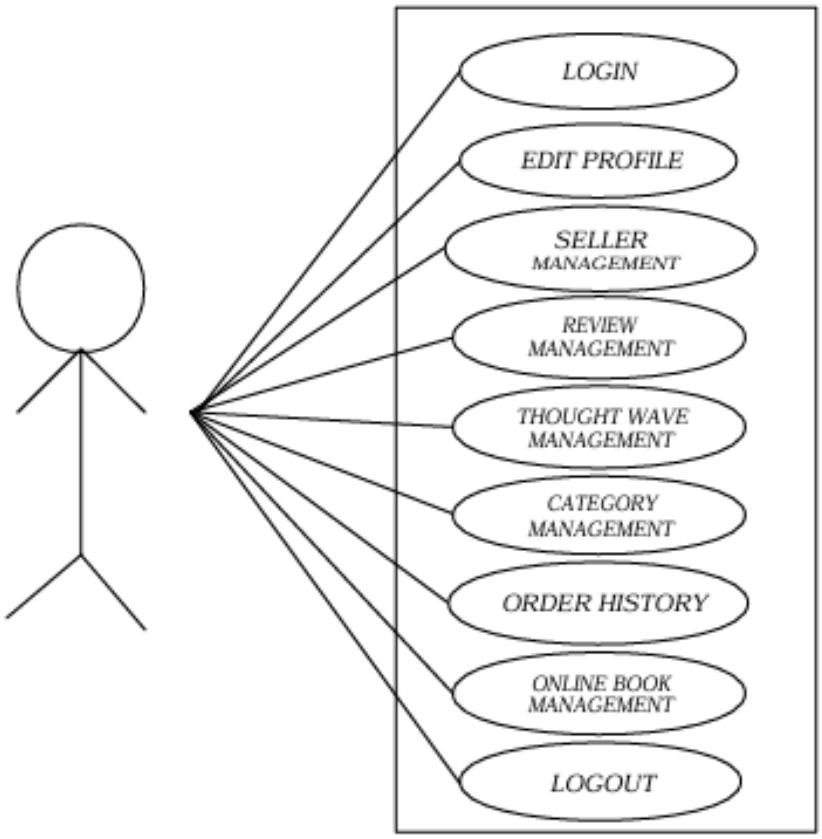
- **Coding Language**: Python
- **Front End**: HTML, CSS, JavaScript
- **Back End**: MySQL, Python
- **Operating System**: Windows 8 or higher
- **Platform**: Android
- **Tools Used**: Android Studio, Pycharm
- **Servers**: XAMPP Control Panel v3.3.0

4. REQUIREMENT SPECIFICATION MODEL

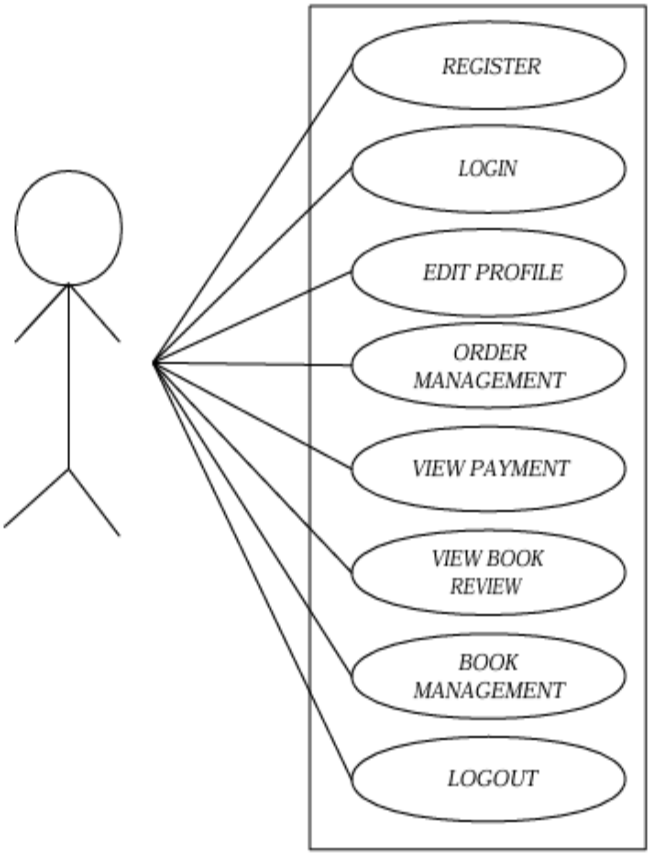
4.1 USECASE DIAGRAM

We have used use case approach in the Unified Modelling Language (UML) to understand the various requirements of each end user. The system is modularized based on users. The main purpose of a use case diagram is to show what system functions are performed by each actor. An actor is a person, organization or an external system that plays a role in one or more interactions of a system.

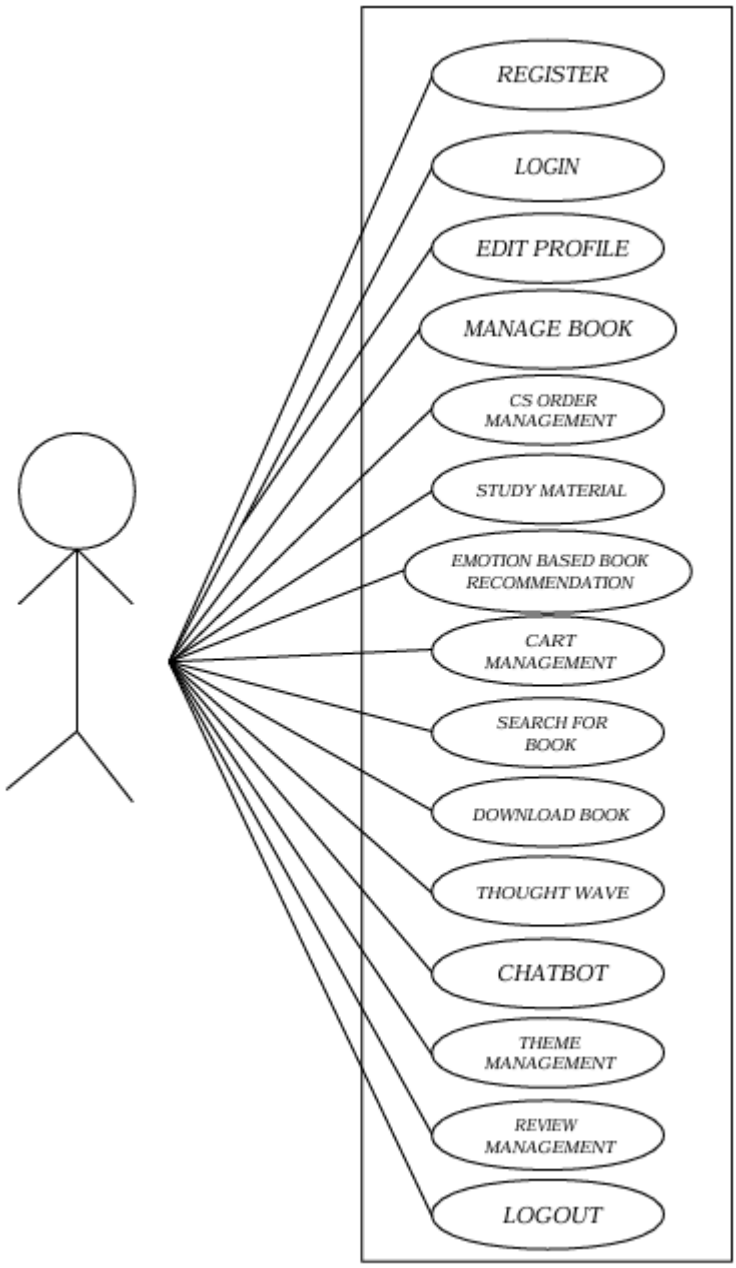
ADMIN



SELLER


REGISTER
LOGIN
EDIT PROFILE
ORDER MANAGEMENT
VIEW PAYMENT
VIEW BOOK REVIEW
BOOK MANAGEMENT
LOGOUT

CUSTOMER

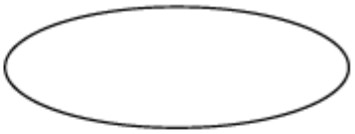

REGISTER
LOGIN
EDIT PROFILE
MANAGE BOOK
CS ORDER MANAGEMENT
STUDY MATERIAL
EMOTION BASED BOOK RECOMMENDATION
CART MANAGEMENT
SEARCH FOR BOOK
DOWNLOAD BOOK
THOUGHT WAVE
CHATBOT
THEME MANAGEMENT
REVIEW MANAGEMENT
LOGOUT

4.2 DATA FLOW DIAGRAM

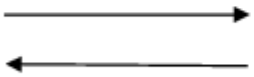
Data flow Diagrams (DFD) is the most commonly used way of documenting the processing of the required system. They are the pictorial way of showing the flow of data into, around and out of the system. They can be understood by the users and are less prone to misinterpretation than textual description. A complete set of DFDs provide a compact top-down representation of the system, which makes it easier for users and analysts to envisage the system as a whole. DFD also known as Bubble Chart has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design phase that functionally decomposes the requirements specifications down to the level of details. It does not show the information about the timing processes or information about whether processes will operate in sequence or in parallel. A DFD shows, what kind of data will be put into and out of the system, where data will come from and go to and where data will be stored. ADFD is often a preliminary way of creating the overview of the system.

DFD mainly uses the following symbols:

Source/Sink



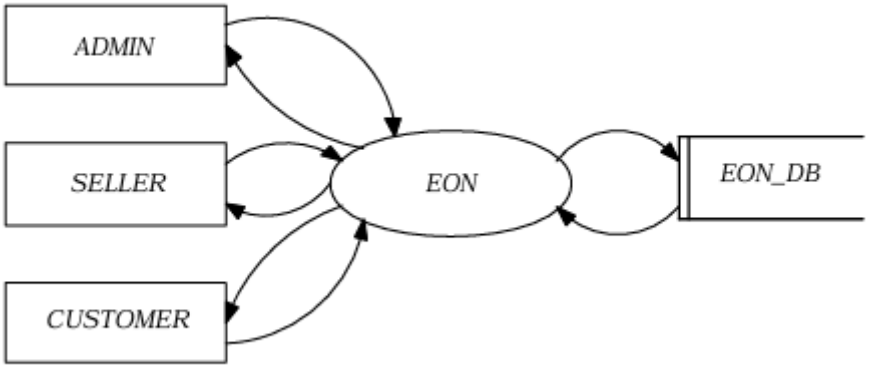
Process



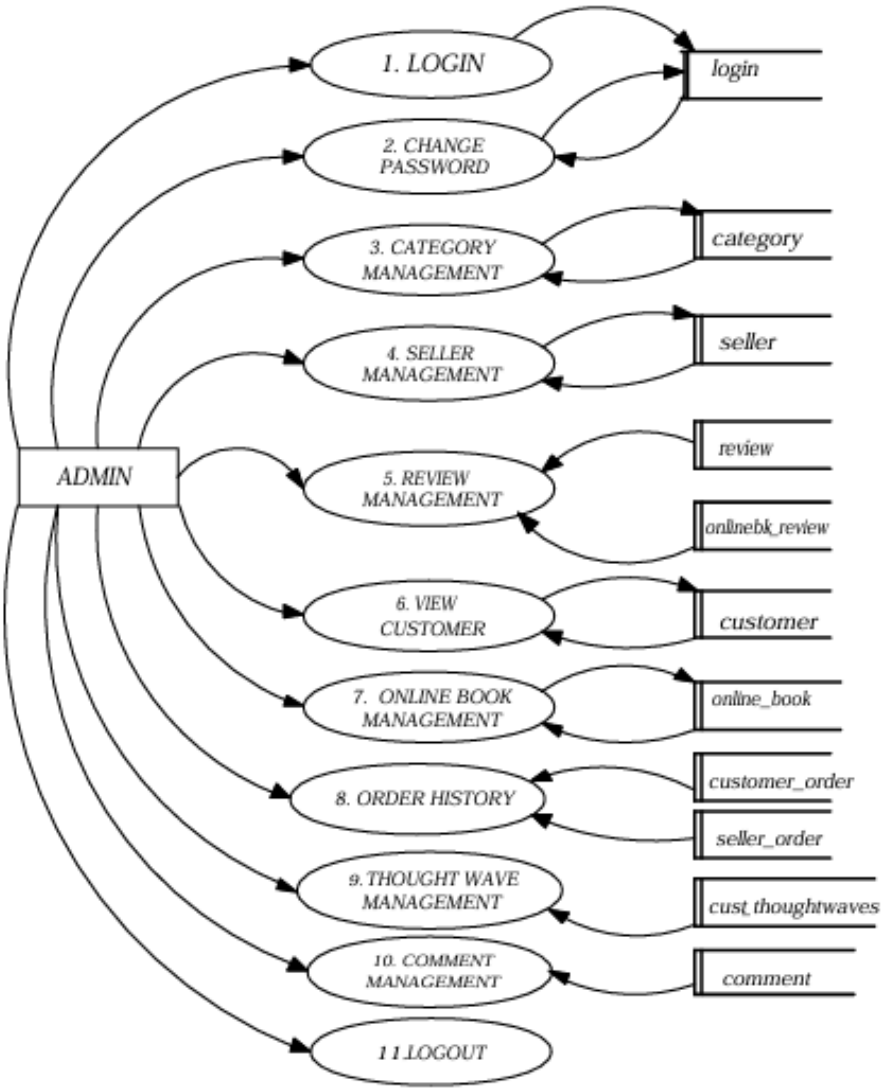
Dataflow

Datastore

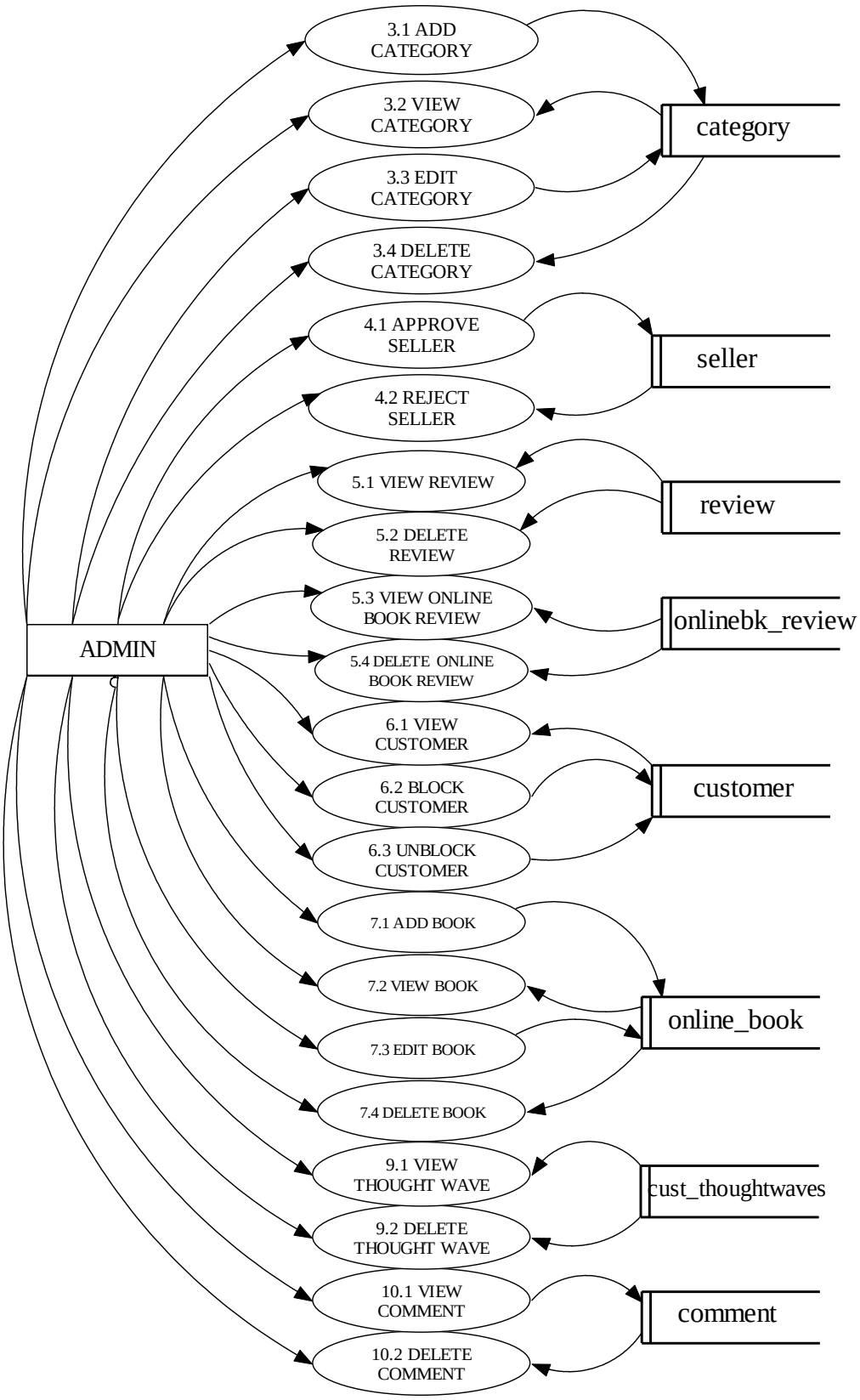
LEVEL0: EON



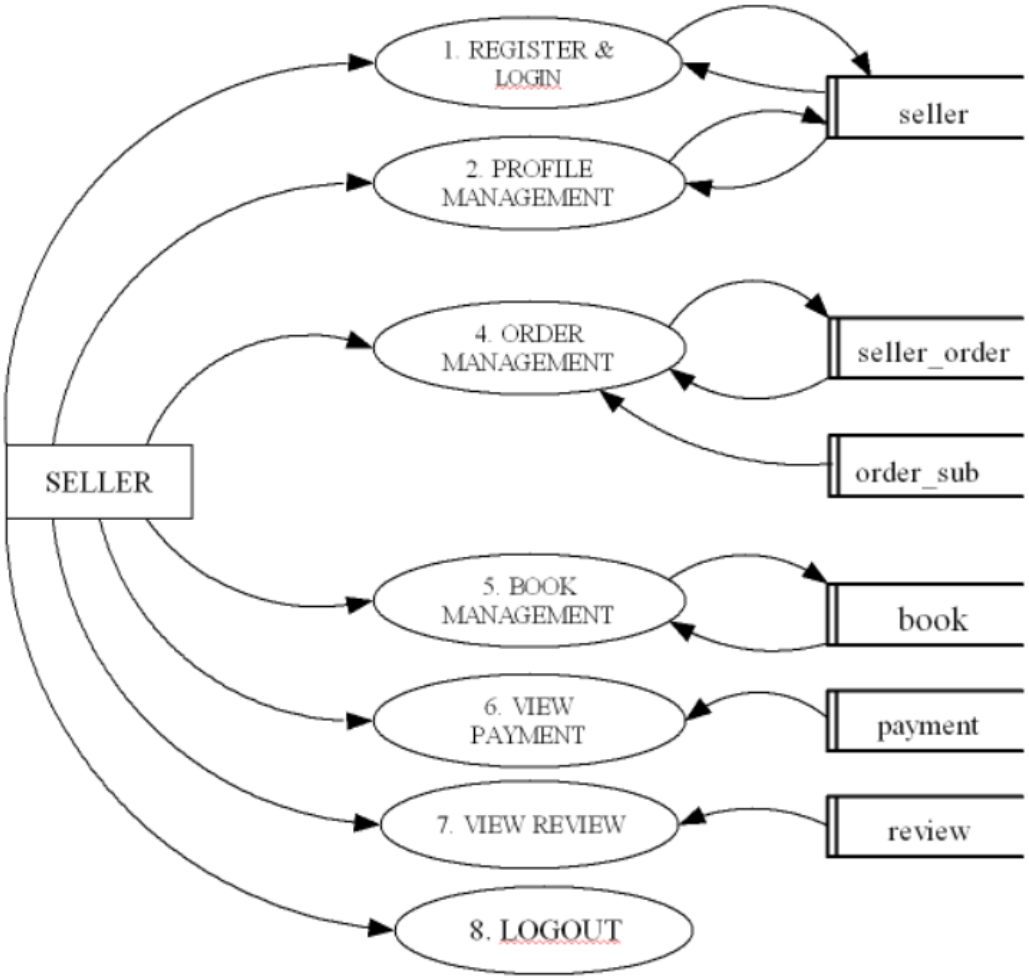
LEVEL 1: ADMIN



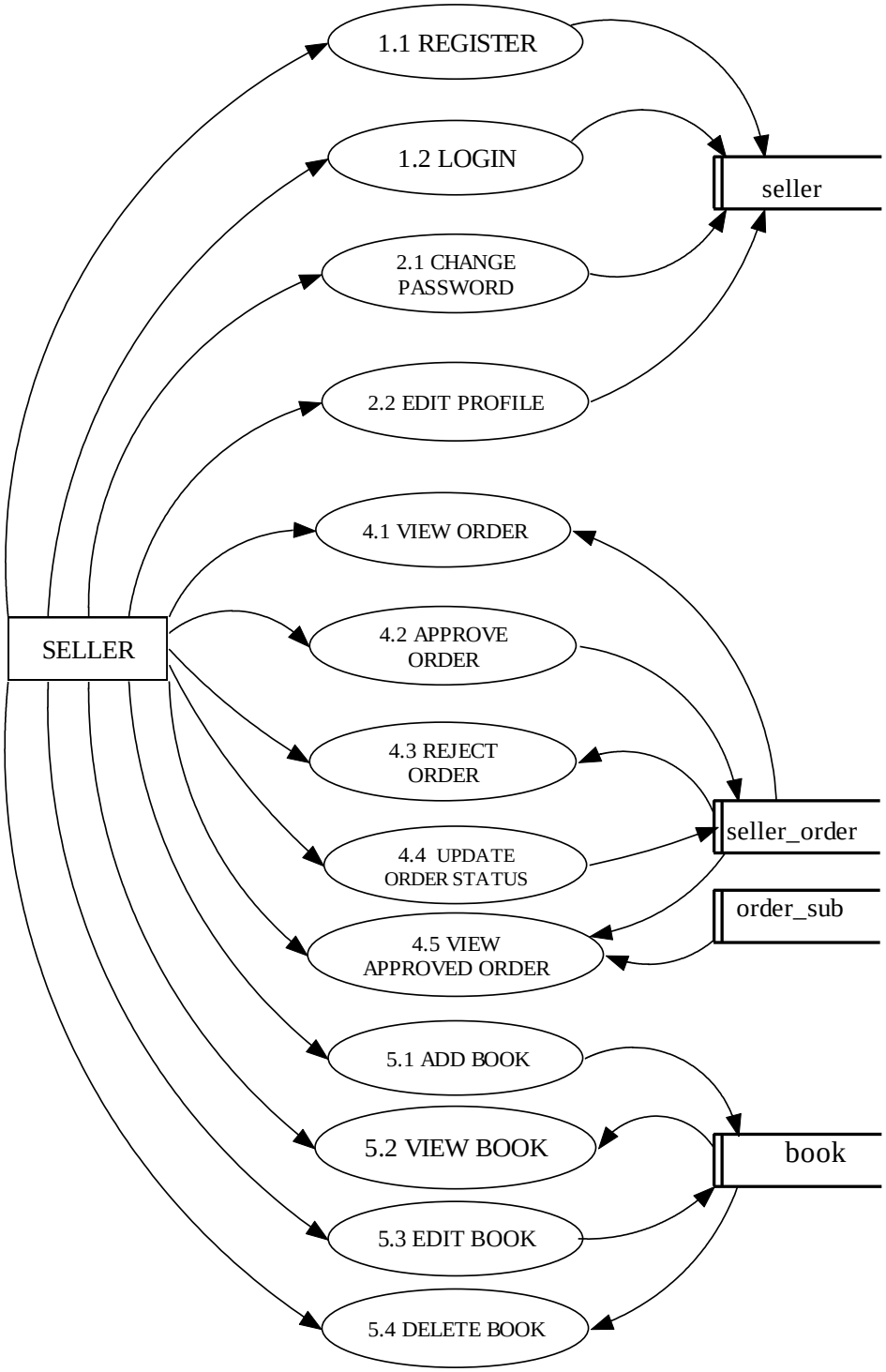
LEVEL 2: ADMIN



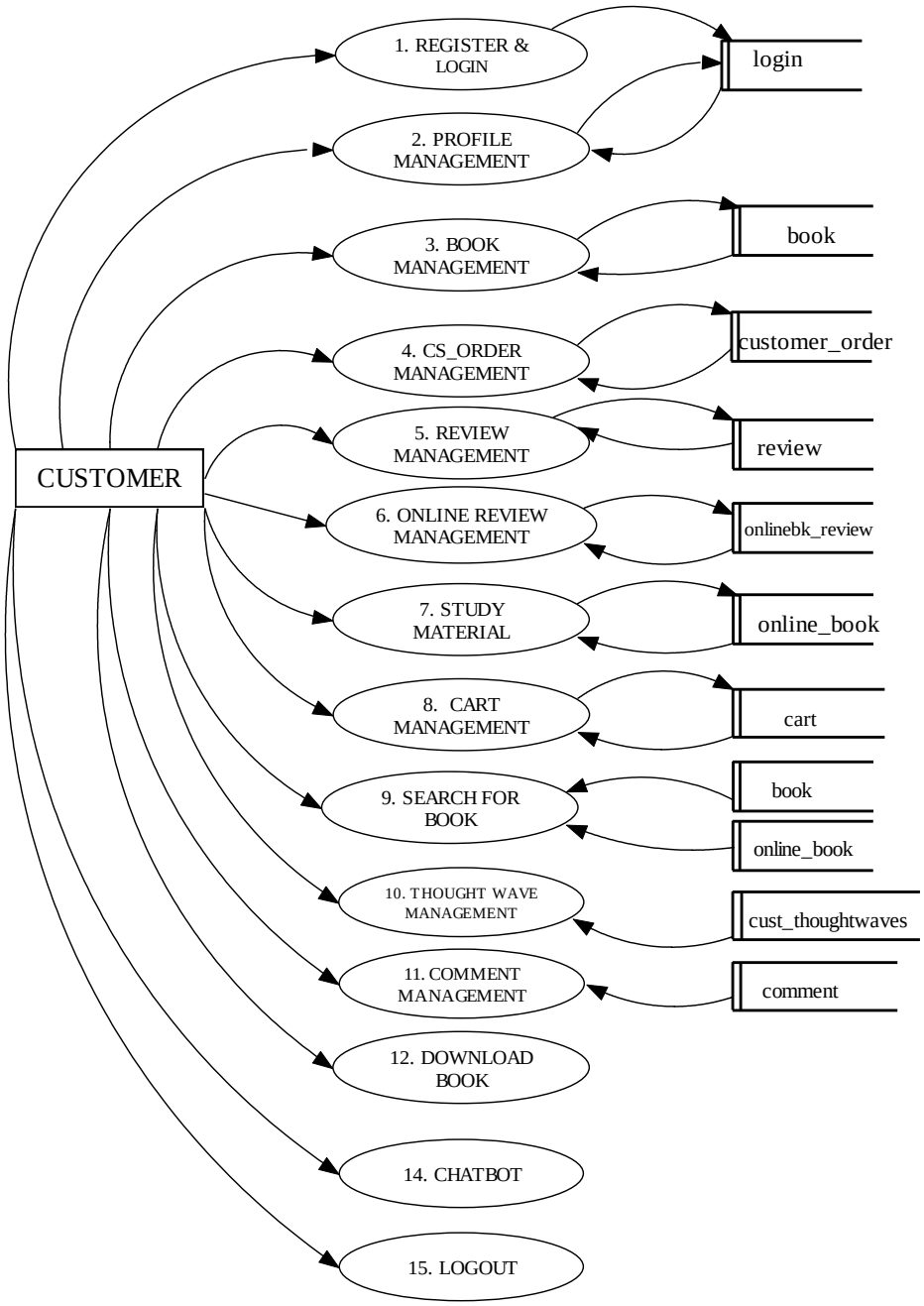
LEVEL 1: SELLER



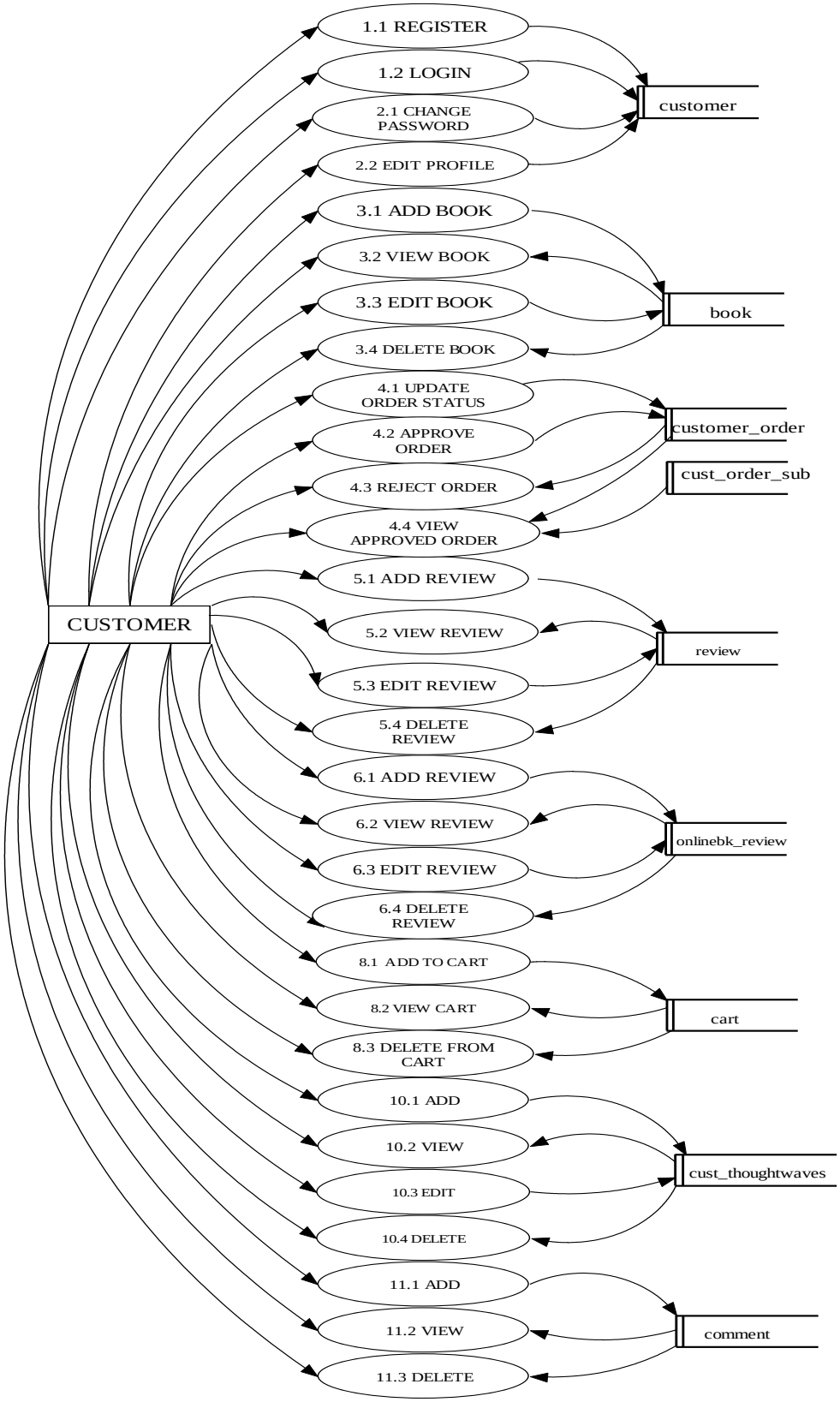
LEVEL 2: SELLER



LEVEL 1: CUSTOMER



LEVEL 2: CUSTOMER



5. SYSTEM DESIGN

System design, the most creative and challenging phase of the development process, serves as the solution to creating the proposed system. It involves crafting technical specifications based on the feasibility study, detailing procedures for implementation. The design specification outlines features, input/output files, and data files, ensuring the system meets requirements for information presentation, accuracy, interaction methods, and overall reliability. Adherence to organizational rules and practices is crucial. Key design objectives encompass practicality, cost efficiency, flexibility, and security, all geared towards fulfilling the specified requirements in the feasibility report.

5.1 MODULES

The proposed system involves 3 modules which are the follows:

1. Admin

2. Seller

3. Customer

5.2 HIGH LEVEL DESIGN

ADMIN

Admin has the right to approve and reject the seller by checking the uploaded credentials and admin can view customer information and can block or unblock customers. They can also view and add different categories. Admin can also add and view online book. They can also view review, view order history, view thoughtwaves, order history and can also view and delete unwanted comment.

SELLER

Seller can accept or update their order and can view approved order. They can add, view and update book. They can view the payment details, view review to their books.

CUSTOMER

Customers can add, view, and edit their books. They can approve, view and update orders. They can add and update reviews for online and seller-provided books. They can add and view comment, cart and thoughtwaves.

5.3 LOW LEVEL DESIGN

ADMIN

1. Login: The admin can login to the website by entering username and password. The username and password are stored in the login table.
2. Category management: Admins can manage categories by viewing, editing, adding, and deleting them.
3. Manage seller: The admin can manage the seller by approving and rejecting them based on their log in credentials which are stored in seller table respectively.
4. Review management: The Admin can view all review placed by the client and delete inappropriate or unwanted reviews.
5. View customer: Admins can view customer information and manage customer accounts, including the ability to block or unblock customers.
6. Order history: The Admin can view order history of both seller and customer.
7. Thoughtwaves management: Admins can view thoughtwaves uploaded by customers and have the authority to delete any inappropriate or unwanted content.
8. Comment management: Admins can also view and delete unwanted comment.

SELLER

1. Login: The seller can login to the website by entering username and password. The username and password are stored in the login table.

2. Order management: The seller can view, approve and update their order.

3. Book management: The seller can manage their books by adding, viewing, editing, and deleting them.

4. View payment: The seller can view the payment done by the customer.

5. View review: The seller can view the reviews done by the customer.

CUSTOMER

1. Login: The customer can login to the website by entering username and password. The username and password are stored in the login table.

2. Manage Book: The customer can manage their books by adding, viewing, editing, and deleting them.

3. Order management: The customer can view, approve and update their order.

4. Online book review: The customer can view, add, edit and delete online book review provided by admin.

5. Customer and seller review: The customer can view, add, edit and delete the reviews to book provided by seller and other customer.

6.Study material: The customer can view study material provided by admin.

7. Emotional based book recommendation: Customers can discover books that match their emotions through emotional-based book recommendation feature.

8. Comment management: The customer can view, add and delete comment

9. Cart management: The Customers can manage their shopping cart by viewing, adding, and deleting books.

10.Search for books: The customer can search for books.

11.Download content: Customers can download book contents for offline reading.

12. Thoughtwaves: Customers can share their thoughts by adding thoughtwaves, and also edit, view, or delete them as needed.

5.4 ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship (ER) model utilizes an Entity Relationship Diagram (ER Diagram) to depict the structure of a database. It serves as a design or blueprint that can be implemented as an actual database. The key components of the ER model are entity sets and relationship sets, with an ER diagram illustrating the relationships among entity sets. Entity sets are groups of similar entities, each with attributes. In the context of a Database Management System (DBMS), an entity corresponds to a table or attribute within a table. By showcasing relationships among tables and their attributes, the ER diagram visually represents the complete logical structure of a database. Following are the main components and its symbols in ER Diagrams:

Rectangles: Represents Entity Set

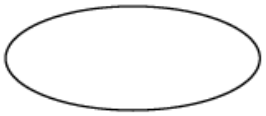
Ellipses: Attributes

Diamonds: Relationship Set

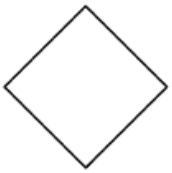
Lines: It links attributes to Entity Set and Entity Set with other Relationship set

Double Ellipses: Multi-valued Attributes

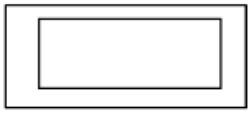
Entity or Strong Entity



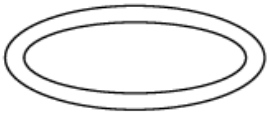
Attributes



Relationship

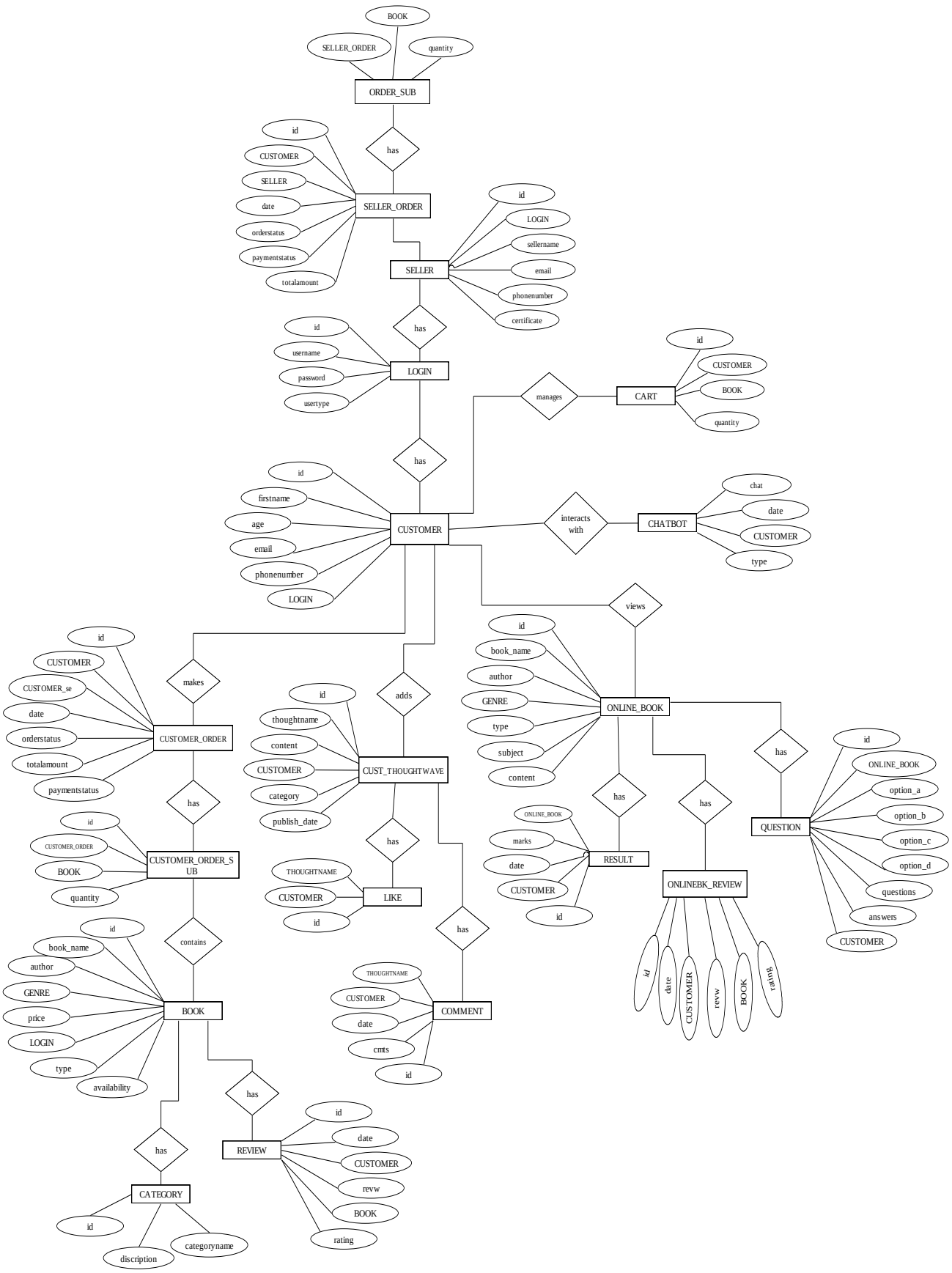


Weak Entity



Multi-valued Attributes

Weak Relationship


BOOK
SELLER_ORDER
quantity
ORDER_SUB
has
id
CUSTOMER
SELLER
date
orderstatus
paymentstatus
totalamount
SELLER_ORDER
id
LOGIN
sellername
email
phonenumber
certificate
SELLER
has
id
username
password
usertype
LOGIN
manages
CART
id
CUSTOMER
BOOK
quantity
has
id
firstname
age
email
phonenumber
LOGIN
CUSTOMER
interacts with
CHATBOT
chat
date
CUSTOMER
type
views
id
CUSTOMER
CUSTOMER_se
date
orderstatus
totalamount
paymentstatus
makes
CUSTOMER_ORDER
id
thoughtname
content
CUSTOMER
category
publish_date
adds
CUST_THOUGHTWAVE
id
book_name
author
GENRE
type
subject
content
ONLINE_BOOK
has
id
CUSTOMER_ORDER
BOOK
quantity
CUSTOMER_ORDER_SUB
has
THOUGHTNAME
CUSTOMER
id
LIKE
has
ONLINE_BOOK
marks
date
CUSTOMER
id
RESULT
has
ONLINEBK_REVIEW
id
date
CUSTOMER
revw
BOOK
rating
has
id
ONLINE_BOOK
option_a
option_b
option_c
option_d
questions
answers
CUSTOMER
QUESTION
contains
id
book_name
author
GENRE
price
LOGIN
type
availability
BOOK
has
THOUGHTNAME
CUSTOMER
date
cmts
id
COMMENT
has
id
date
CUSTOMER
revw
BOOK
rating
REVIEW
has
CATEGORY
id
discription
categoryname

5.5 DATABASE DESIGN

1. LOGIN TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
username	varchar	null
password	varchar	null
usertype	varchar	null

2. CATEGORY TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
categoryname	varchar	null
discription	varchar	null

3. SELLER TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
LOGIN_id	int	Foreign Key
sellername	varchar	null
email	varchar	null
phonenumber	varchar	null
place	varchar	null
post	varchar	null
pincode	varchar	null
certificate	varchar	null

4. BOOK TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
GENRE_id	int	Foreign Key
LOGIN_id	int	Foreign Key
book_name	varchar	null
author	varchar	null
description	longtext	null
price	int	null
discount	varchar	null
image	varchar	null
bookcondition	varchar	null
type	varchar	null
availability	varchar	null
numberofpages	varchar	null

5. ONLINE_BOOK TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
GENRE_id	int	Foreign key
book_name	varchar	null
author	varchar	null
description	varchar	null
book_format	varchar	null
image	varchar	null
filesize	varchar	null

language	varchar	null
no_ofpage	varchar	null
upload_date	varchar	null
type	varchar	null
subject	varchar	null
content	varchar	null
emotion	varchar	null

6. CUSTOMER TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
LOGIN_id	int	Foreign key
image	varchar	null
firstname	varchar	null
lastname	varchar	null
dob	varchar	null
age	varchar	null
gender	varchar	null
email	varchar	null
phonenumber	varchar	null
place	varchar	null
post	varchar	null
pin	varchar	null

7. CUSTOMER_ORDER TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
CUSTOMER_se_id	int	Foreign key
date	varchar	null
orderstatus	varchar	null
paymentstatus	varchar	null
paymentmethod	varchar	null
totalamount	varchar	null
discount	varchar	null
place	varchar	null
city	varchar	null
state	varchar	null
colony	varchar	null
house	varchar	null
pin	varchar	null
deliverydate	varchar	null
ordercancellationreason	varchar	null

8. CUSTOMER_ORDER_SUB TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
BOOK_id	int	Foreign key
CUSTOMER_ORDER_id	int	Foreign key
quantity	varchar	null

9. CUST_THOUGHTWAVES TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
thoughtname	varchar	null
content	varchar	null
category	varchar	null
image	varchar	null
publish_date	varchar	null

10. APPVDSELLER TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
SELLER_id	int	Foreign key
email	varchar	null
phno	varchar	null

11. CART TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
BOOK_id	int	Foreign key
CUSTOMER_id	int	Foreign key
quantity	int	null

13. LIKE TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER _id	int	Foreign key
THOUGHTNAME_id	int	Foreign key

14. COMMENT TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
THOUGHTNAME_id	int	Foreign key
date	varchar	null
cmts	varchar	null

15. SELLER_ORDER TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
SELLER_id	int	Foreign key
date	varchar	null
orderstatus	varchar	null
paymentstatus	varchar	null
paymentmethod	varchar	null
totalamount	varchar	null
discount	varchar	null
city	varchar	null

state	varchar	null
colony	varchar	null
house	varchar	null
pin	varchar	null
deliverydate	varchar	null
ordercnclreason	varchar	null

16. ORDER_SUB TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
BOOK_id	int	Foreign key
SELLER_ORDER_id	int	Foreign key
quantity	varchar	null

18. QUESTION TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
ONLINE_BOOK_id	int	Foreign key
option_a	varchar	null
option_b	varchar	null
option_c	varchar	null
option_d	varchar	null
questions	varchar	null
answers	varchar	null

19. CHATBOT TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
chat	longtext	null
date	varchar	null
type	varchar	null

20. REVIEW TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
BOOK_id	int	Foreign key
CUSTOMER_id	int	Foreign key
date	varchar	null
rating	varchar	null

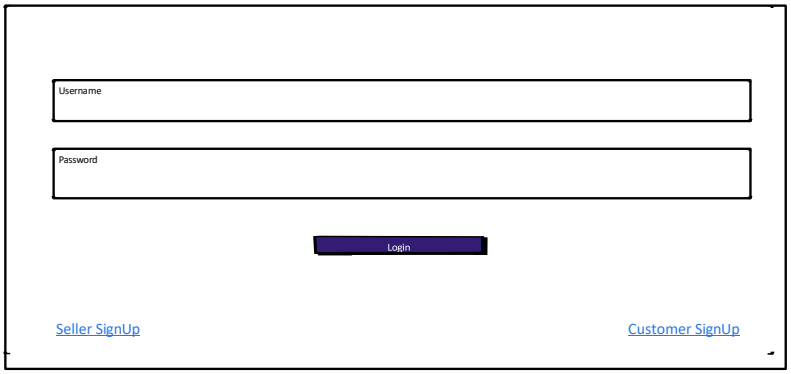
21. ONLINEBK_REVIEW

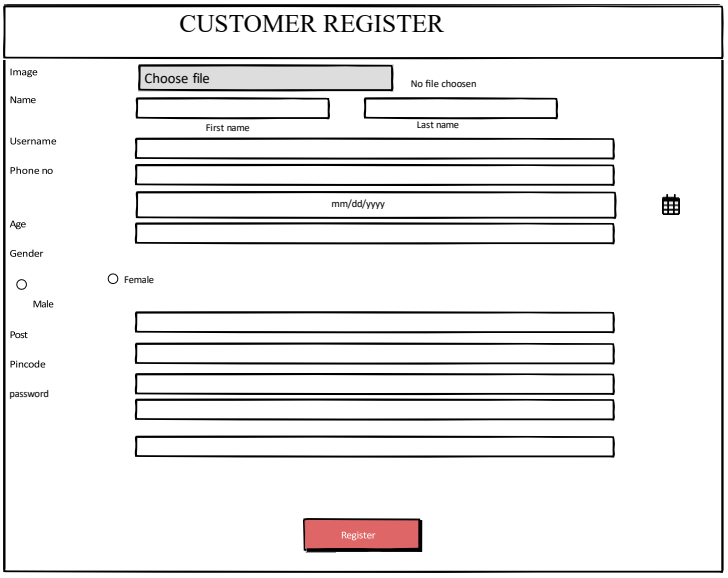
DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
ONLINE_BOOK_id	int	Foreign key
date	varchar	null
revw	varchar	null
rating	varchar	null

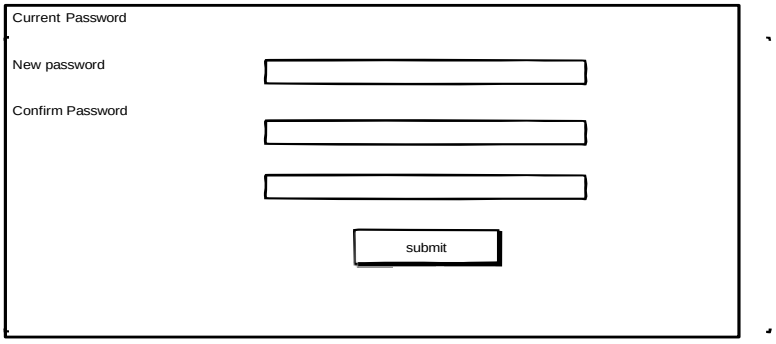
23. RESULT TABLE

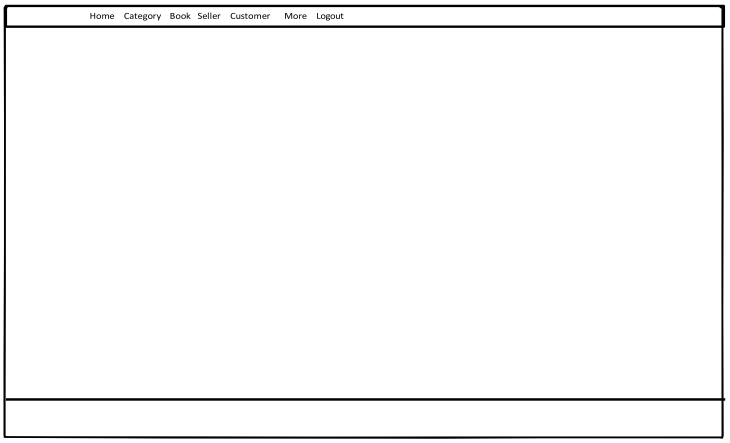
DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary Key
CUSTOMER_id	int	Foreign key
ONLINE_BOOK_id	int	Foreign key
marks	varchar	null
date	varchar	null

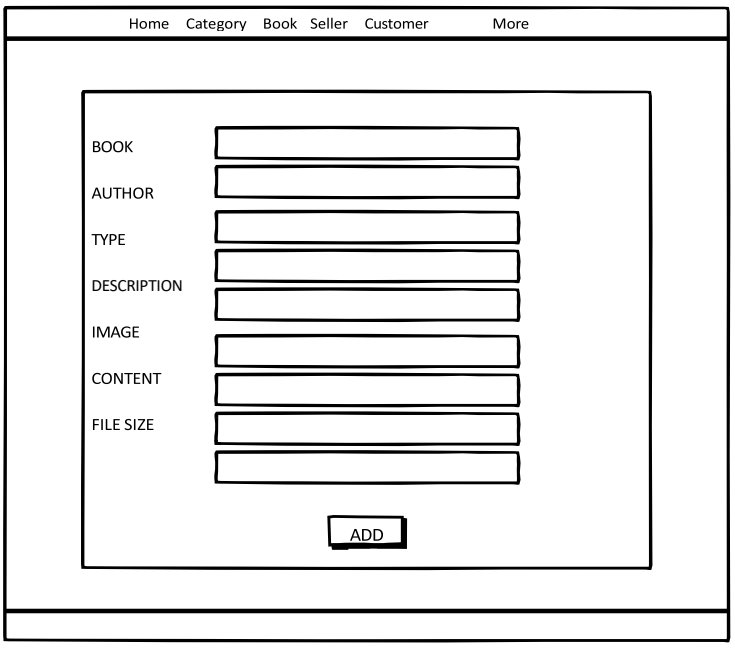
5.6 USER INTERFACE

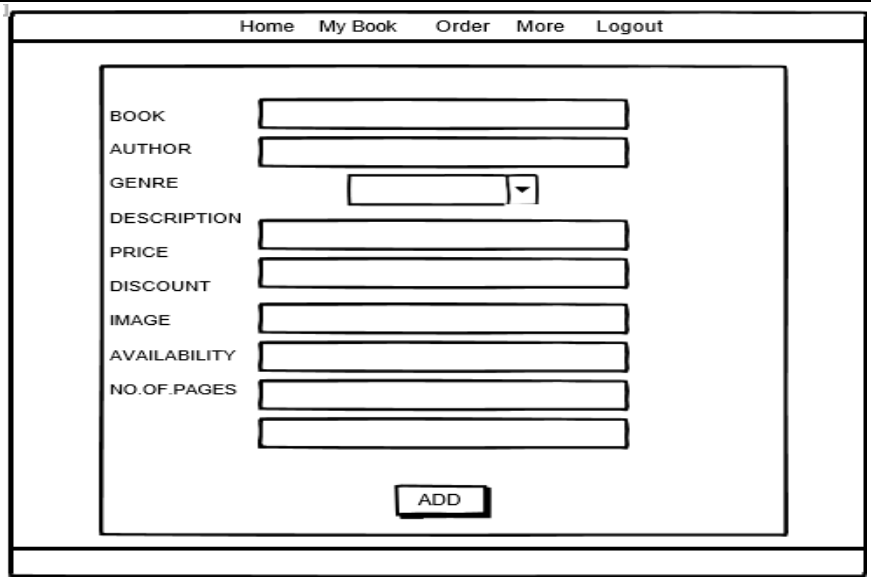


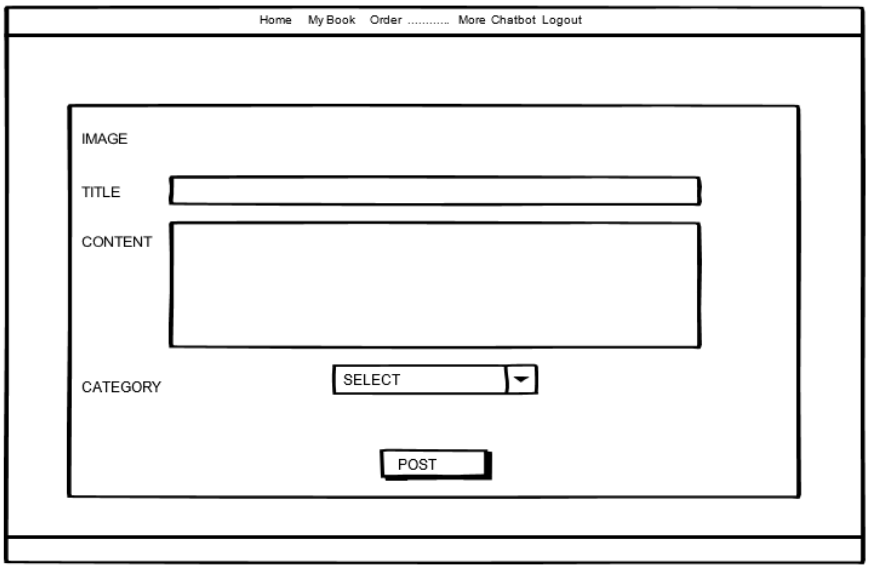



Current Password
New password
Confirm Password
submit

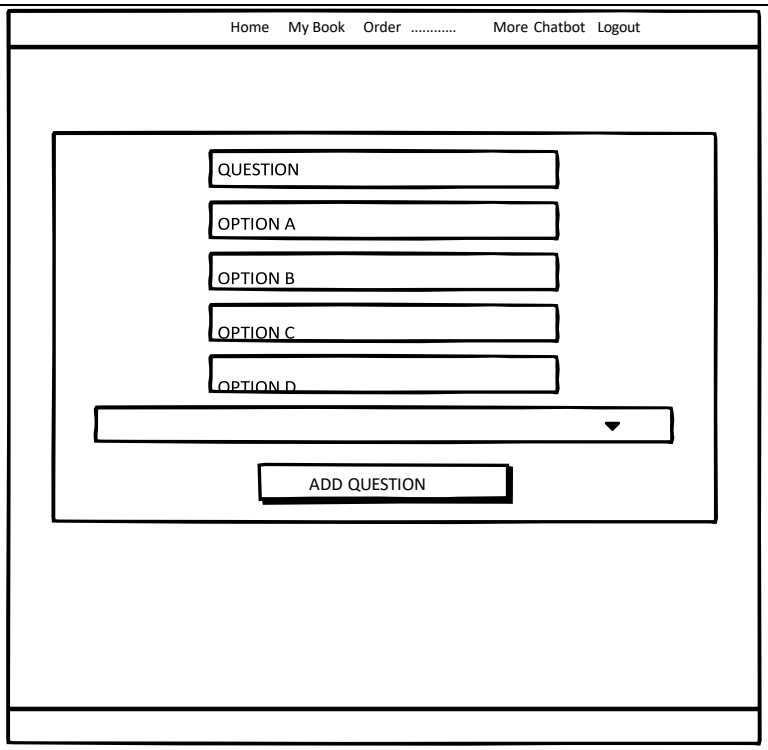

Home Category Book Seller Customer More Logout


Home Category Book Seller Customer More
BOOK
AUTHOR
TYPE
DESCRIPTION
IMAGE
CONTENT
FILE SIZE
ADD


Home
My Book
Order
More
Logout
BOOK
AUTHOR
GENRE
DESCRIPTION
PRICE
DISCOUNT
IMAGE
AVAILABILITY
NO.OF.PAGES
ADD


Home
My Book
Order .....
More
Chatbot
Logout
IMAGE
TITLE
CONTENT
CATEGORY
SELECT
POST


Who are you?
I'm just a virtual assistant .What can i help you with?
Send


Home
My Book
Order .....
More
Chatbot
Logout
QUESTION
OPTION A
OPTION B
OPTION C
OPTION D
ADD QUESTION

6. CODING

LOGIN:

```
def log(request):
    return render(request,"login_new.html")
```

LOGIN ACTION:

```
def login_post(request):
    un=request.POST['u']
    ps=request.POST['p']
    request.session['login']=1
    res=login.objects.filter(username=un,password=ps)
    image_data = request.POST.get('image')
    if image_data:
        image_data = image_data.split(",")[1]  # Remove Base64 header
        image_path = os.path.join(r"C:\Users\hilus\PycharmProjects\eon\\", f'{un}_login_image.png')
        with open(image_path, 'wb') as f:
            f.write(base64.b64decode(image_data))
    if res.exists():
        res = res[0]
        request.session['lid']=res.id
        if res.usertype=='admin':
            return HttpResponse('<script>window.location="/admin_index"</script>')
        if res.usertype=='seller':
            request.session['lid'] = res.id
            request.session['sid'] = seller.objects.get(LOGIN=res.id).id
            return HttpResponse('<script>window.location="/seller_index"</script>')
        if res.usertype == 'customer':
            request.session['lid'] = res.id
            request.session['cid'] =customer.objects.get(LOGIN=res.id).id
            print("cid", request.session['cid'])
```

```
        return HttpResponse('<script>window.location="/cust_index"</script>')
    else:
        return HttpResponse('<script>alert("unauthorised user");window.location="/"</script>')
```

LOGOUT:

```
def logout(request):
    request.session['login']=0
    return HttpResponse('<script>window.location="/"</script>')
```

ADMIN MODULE

ADMIN INDEX:

```
def admin_index(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/index.html")
```

CHANGE PASSWORD:

```
def chpass(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/changepass.html")
```

CHANGE PASSWORD ACTION:

```
def chpass_post(request):
    curps=request.POST['textfield']
    nwps=request.POST['textfield2']
    cnfmps=request.POST['textfield3']
    data = login.objects.filter(id = request.session['lid'],password=curps)
    if data.exists():
        if (nwps == cnfmps):
            login.objects.filter(id=request.session['lid']).update(password=cnfmps)
            return HttpResponse('<script>alert("password changed!");
window.location="/admin_index"</script>')
        else:
```

```
        return HttpResponse('<script>alert("password notsame");
window.location="/chpass"</script>')
    else:
        return HttpResponse('<script>alert("user not found"); window.location="/chpass"</script>')
```

CATEGORY ADD:

```
def add_cat(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/addcat.html")
```

CATEGORY ADD ACTION:

```
def addcat_post(request):
    catname=request.POST['textfield']
    description=request.POST['textarea']
    obj=category()
    obj.categoryname=catname
    obj.discription=description
    obj.save()
    return HttpResponse("<script>alert('Category added'); window.location='/view_cat#aa'</script>")
```

VIEW CATEGORY:

```
def view_cat(request):
    res=category.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/viewcat.html",{'data':res})
```

EDIT CATEGORY:

```
def edit_cat(request,id):
    view=category.objects.get(id=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
```

```
    return render(request,"admin/editcat.html",{"view":view})
```

EDIT CATEGORY ACTION:

```
def editcat_post(request,id):
    catname = request.POST['textfield']
    description = request.POST['textarea']
    category.objects.filter(id=id).update(categoryname=catname,discription=description)
    return HttpResponse('<script>alert(" category edited!");
window.location="/view_cat#aa"</script>')
```

DELETE CATEGORY:

```
def delete_cat(request,id):
    category.objects.get(id=id).delete()
    return HttpResponse('<script>window.location="/view_cat#aa"</script>')
```

ADD ONLINE BOOK:

```
def onlinebookadd(request):
    data = category.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/onlinebook.html",{'data':data})
```

ADD ONLINE BOOK ACTION:

```
def onlinebookadd_post(request):
    bookname=request.POST['textfield']
    author=request.POST['textfield2']
    type=request.POST['select1']
    description=request.POST['textfield5']
    book_content = request.FILES['textfield19']
    image=request.FILES['image']
    pgno=request.POST['textfield14']
    lang=request.POST['textfield15']
    filesize=request.POST['textfield16']
```

```
    fs = FileSystemStorage()
    d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
    path = "/static/image/" + d + ".jpg"
    d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\certificate\\"+d+".pdf", book_content)
    path1 = "/static/certificate/"+d+".pdf"
    obj=online_book()
    obj.book_name=bookname
    obj.author=author
    obj.type=type
    if type == "academic":
        obj.subject = request.POST['select2']
        obj.GENRE_id = category.objects.filter()[0].id
    if type== "non-academic":
        obj.GENRE_id = request.POST['select']
        obj.emotion =1
    obj.description=description
    obj.image=path
    obj.content=path1
    obj.language=lang
    obj.filesize=filesize
    obj.no_ofpage=pgno
    obj.save()
    return HttpResponse('<script>window.location="/onlinebkview_frontview#aa"</script>')
```

VIEW ONLINE BOOK:

```
def onlinebkview_frontview(request):
    res=online_book.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/onlinebkview_interface.html",{'data':res})
```

VIEW ONLINE BOOK ACTION:

```
def onlinebkview_frontview_post(request):
    res=online_book.objects.filter(type = request.POST['select2'])
    return render(request,"admin/onlinebkview_interface.html",{'data':res})
```

VIEW ONLINE BOOK:

```
def onlinebkview(request,id):
    data = online_book.objects.filter(id=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/onlinebookview.html",{'data':data})
```

EDIT ONLINE BOOK:

```
def onlinebookedit(request,id):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    data = online_book.objects.get(id=id)
    data1 = category.objects.all()
    return render(request,"admin/edit_onlinebook.html",{"data":data,"data1":data1})
```

EDIT ONLINE BOOK ACTION:

```
def onlinebookedit_post(request,id):
        description = request.POST['textfield5']
        type = request.POST['select1']
        filesize = request.POST['textfield16']
        language = request.POST['textfield15']
        print("disss",description)
        print("type",type)
        print("filesize",filesize)
        print("lang",language)
        pgno = request.POST['textfield14']
        online_book.objects.filter(id=id).update(description=description, filesize=filesize,
language=language, type=type, no_ofpage=pgno)
```

```
    if 'image' in request.FILES:
        image = request.FILES['image']
        fs = FileSystemStorage()
        d = datetime.datetime.now().strftime('%Y%m%d-%H%M%S')
        fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
        path = "/static/image/" + d + ".jpg"
        online_book.objects.filter(id=id).update(image=path)
        return HttpResponse('<script>alert(" edit successfully");
window.location="/onlinebkview_frontview"</script>')
    if type == "academic":
        subject1 = request.POST['select2']
        online_book.objects.filter(id=id).update(subject=subject1)
        return HttpResponse('<script>alert(" edit successfully");
window.location="/onlinebkview_frontview"</script>')
    elif type == "non-academic":
        genre1 = request.POST['select']
        online_book.objects.filter(id=id).update(GENRE=genre1)
        return HttpResponse('<script>alert(" edit
successfully");window.location="/onlinebkview_frontview"</script>')
    else:
online_book.objects.filter(id=id).update(description=description, filesize=filesize,
language=language, type=type, no_ofpage=pgno)
        return HttpResponse('<script>alert(" edit
successfully");window.location="/onlinebkview_frontview"</script>')
```

DELETE ONLINE BOOK:

```
def onlinebkdelete(request,id):
  online_book.objects.get(id=id).delete()
  return HttpResponse('<script>alert("delete
successfully");window.location="/onlinebkview_frontview#aa"</script>')
```

SELLER MANAGEMENT:

```
def seller_mngt(request):
  res=seller.objects.filter(LOGIN__usertype='pending')
  if request.session['login'] == 0:
```

```
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/seller_mngt.html",{'data':res})
```

APPROVE SELLER:

```
def sellerapprove(request,id):
    login.objects.filter(id = id).update(usertype = 'seller')
    return HttpResponse('<script>alert("Approved successfully");window.location="/seller_approval#aa"</script>')
```

REJECT SELLER:

```
def rejectseller(request,id):
    login.objects.filter(id=id).update(usertype='reject')
    return HttpResponse('<script>alert("Rejected successfully"); window.location="/seller_mngt"</script>')
```

SELLER APPROVAL:

```
def seller_approval(request):
    res=seller.objects.filter(LOGIN__usertype='seller')
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired"); window.location="/"</script>')
    return render(request,"admin/viewapprovedseller.html",{'data':res})
```

ORDER HISTORY:

```
def ordhist(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/type_orderhist.html")
```

ORDER HISTORY POST:

```
def orderhist_post(request):
    type=request.POST['select']
    if type == 'seller':
        res=order_sub.objects.all()
```

```
        print(res,"ressssssssss")
        return render(request,"admin/orderhist.html",{'data':res})
    else:
        res=customer_order_sub.objects.all()
        return render(request,"admin/view_user_orderhist.html",{'data':res})
```

REVIEW:

```
def reviw(request):
    res=review.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/reviewman.html",{'data':res})
```

DELETE REVIEW:

```
def dltreview(request,id):
    review.objects.get(id=id).delete()
    return HttpResponse('<script>alert("delete successfully");window.location="/reviw"</script>')
```

ONLINE BOOK REVIEW:

```
def onlinebk_reviw(request):
    res=onlinebk_review.objects.all()
    res2=review.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/onlinebk_review.html",{'data':res, 'data2':res2})
```

DELETE ONLINE BOOK REVIEW:

```
def dlt_onlinebk_review(request,id):
    onlinebk_review.objects.get(id=id).delete()
    return HttpResponse('<script>alert("delete successfully");
window.location="/onlinebk_reviw"</script>')
```

VIEW USERS:

```
def view_user(request):
    res=customer.objects.filter(LOGIN__usertype = 'customer')
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/viewcust.html",{'data':res})
```

BLOCK USERS:

```
def blockuser(request,id):
    login.objects.filter(id=id).update(usertype='blocked')
    return HttpResponse('<script>alert("Blocked
sccessfully");window.location="/view_user#aa"</script>')
```

VIEW BLOCKED USERS:

```
def blockeduser(request):
    res = customer.objects.filter(LOGIN__usertype='blocked')
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "admin/blockuser.html", {'data': res})
```

UNBLOCK USERS:

```
def unblockuser(request,id):
    login.objects.filter(id=id).update(usertype='customer')
    return HttpResponse('<script>alert("unblocked
sccessfully");window.location="/blockeduser#aa"</script>')
```

VIEW ALL THOUGHTWAVE:

```
def view_allthoughtwave_frontview(request):
    res=cust_thoughtwaves.objects.all()
    return render(request,"admin/all_thoughtwave_frontview.html",{'data':res})
```

VIEW MY THOUGHTWAVE:

```
def view_thoughtwaves(request,id):
```

```
    res=cust_thoughtwaves.objects.filter(id=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/thoughtwaves_view.html",{'data':res})
```

DELETE THOUGHTWAVE:

```
def dlt_thoughtwave(request,id):
    cust_thoughtwaves.objects.get(id=id).delete()
    return HttpResponse('<script>alert("delete successfully");
window.location="/cust_tw#aa"</script>')
```

VIEW COMMENTS:

```
def view_comment(request,id):
    res=comment.objects.all()
    res1 = comment.objects.filter(THOUGHTNAME=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"admin/view_twcomment.html",{'data':res,'data1':res1})
```

SELLER MODULE

SELLER REGISTRATION:

```
def reg_seller(request):
    return render(request,"seller/reg.html")
```

SELLER REGISTER ACTION:

```
def reg_seller_post(request):
    seller_name=request.POST['textfield']
    email=request.POST['textfield2']
    phno=request.POST['textfield3']
    place=request.POST['textfield4']
    post=request.POST['textfield6']
    pin=request.POST['textfield7']
    password=request.POST['textfield8']
```

```
    confirm_password=request.POST['textfield5']
    cert = request.FILES['fileField']
    fs=FileSystemStorage()
    d=datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\certificate\\" +d+ ".pdf",cert)
    path="/static/certificate/" +d+ ".pdf"
    if password == confirm_password:
        obj1=login()
        obj1.username=email
        obj1.usertype="pending"
        obj1.password=password
        obj1.save()
        obj=seller()
        obj.sellername=seller_name
        obj.email=email
        obj.phonenumber=phno
        obj.place=place
        obj.post=post
        obj.pincode=pin
        obj.certificate=path
        obj.LOGIN_id=obj1.id
        obj.save()
    else:
        return HttpResponse('<script>alert("muissmachy");window.location="/reg_seller"</script>')
    return HttpResponse('<script>alert("success");window.location="/"</script>')
```

CHANGE PASSWORD:

```
def seller_chpass(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/seller_chpass.html")
```

CHANGE PASSWORD ACTION:

```
def seller_chpass_post(request):
    curps=request.POST['textfield']
    nwps=request.POST['textfield2']
    cnfmps=request.POST['textfield3']
    data = login.objects.filter(id = request.session['lid'],password=curps)
    if data.exists():
        if (nwps == cnfmps):
            login.objects.filter(id=request.session['lid']).update(password=cnfmps)
            return HttpResponse('<script>alert("password changed!");
window.location="/seller_index"</script>')
        else:
            return HttpResponse('<script>alert("password not same");
window.location="/seller_chpass"</script>')
    else:
        return HttpResponse('<script>alert("user not found");
window.location="/seller_chpass"</script>')
```

SELLER INDEX:

```
def seller_index(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/index.html")
```

VIEW PROFILE:

```
def seller_view(request):
    data = seller.objects.get(LOGIN=request.session['lid'])
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/profile_view.html", {"data":data})
```

EDIT PROFILE:

```
def edit_profile(request,id):
    data = seller.objects.get(id=id)
    if request.session['login'] == 0:
```

```
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/seller_edit.html",{"data":data,"id":id})
```

EDIT PROFILE ACTION:

```
def edit_profile_post(request,id):
    name = request.POST['textfield']
    phno=request.POST['textfield3']
    place=request.POST['textfield4']
    post=request.POST['textfield6']
    pin=request.POST['textfield7']
seller.objects.filter(id=id).update(sellername=name, phonenumber=phno, place=place, post=post, pincode=pin)
    return HttpResponse('<script>alert("edited");window.location="/seller_view#aa"</script>')
```

ADD BOOK:

```
def bookadd(request):
    data = category.objects.all
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/book/add.html",{'data':data})
```

ADD BOOK ACTION:

```
def bookadd_post(request):
    bookname=request.POST['textfield']
    author=request.POST['textfield2']
    genre=request.POST['select']
    description=request.POST['textfield5']
    price=request.POST['textfield6']
    discount=request.POST['textfield7']
    image=request.FILES['fileField']
    fs = FileSystemStorage()
    d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
```

```
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
    path = "/static/image/" + d + ".jpg"
    availability=request.POST['textfield13']
    pgno=request.POST['textfield14']
    obj=book()
    obj.book_name=bookname
    obj.author=author
    obj.GENRE_id=genre
    obj.description=description
    obj.price=price
    obj.discount=discount
    obj.image=path
    obj.LOGIN_id = request.session['lid']
    obj.type='seller'
    obj.availability=availability
    obj.numberofpages=pgno
    obj.SELLER = seller.objects.get(LOGIN=request.session['lid'])
    obj.save()
    return HttpResponse('<script>alert("added");window.location="/bkview_frontview"</script>')
```

EDIT BOOK:

```
def bookedit(request,id):
    data = book.objects.get(id=id)
    print(data.id, " cc")
    data1 = category.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/book/edit.html",{"data2":data,"data1":data1})
```

EDIT BOOK ACTION:

```
def bookedit_post(request,id):
        description = request.POST['textfield5']
        price = request.POST['textfield6']
```

```
    discount = request.POST['textfield7']
    genre = request.POST['cat']
    type = request.POST['textfield12']
    availability = request.POST['textfield13']
    pgno = request.POST['textfield14']
    if 'fileField' in request.FILES:
        image = request.FILES['fileField']
        fs = FileSystemStorage()
        d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
        fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
        path = "/static/image/" + d + ".jpg"
        book.objects.filter(id=id).update(description=description, price=price, discount=discount,
GENRE=genre, image=path, type=type,  availability=availability, numberofpages=pgno)
        return HttpResponse('<script>alert(" edit successfully");
window.location="/bkview_frontview"</script>')
    else:
        book.objects.filter(id=id).update(description=description, price=price, discount=discount,
type=type, availability=availability, numberofpages=pgno, GENRE=genre)
        return HttpResponse('<script>alert(" edit successfully");
window.location="/bkview_frontview"</script>')
```

VIEW BOOK:

```
def bkview(request,id):
  print(request.session['lid'])
  data = book.objects.filter(id = id, type = 'seller')
  if request.session['login'] == 0:
    return HttpResponse('<script>alert("session expired");window.location="/"</script>')
  return render(request,"seller/book/bookview.html",{'data':data})
```

VIEW BOOK INTERFACE:

```
def bkview_frontview(request):
  res=book.objects.filter(LOGIN=request.session['lid'],LOGIN__usertype = 'seller')
  if request.session['login'] == 0:
    return HttpResponse('<script>alert("session expired");window.location="/"</script>')
```

```
    return render(request,"seller/bookview.html",{'data':res})
```

VIEW BOOK INTERFACE ACTION:

```
def bkview_frontview_post(request):
    res=book.objects.filter(LOGIN=request.session['lid'],LOGIN__usertype = 'seller',
book_name__icontains=request.POST['ttt'])
    return render(request,"seller/bookview.html",{'data':res})
```

DELETE BOOK:

```
def bkdelete(request,id):
    book.objects.get(id=id).delete()
    return HttpResponse('<script>alert("delete successfully");
window.location="/bkview_frontview"</script>')
```

VIEW RECEIVED ORDER:

```
def seller_view_order(request):
    print( request.session['lid'])
    res = seller_order.objects.filter(SELLER__LOGIN = request.session['lid'],orderstatus='pending')
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/ordermngt.html",{'data':res})
```

APPROVE ORDER:

```
def orderapprove(request,id):
    seller_order.objects.filter(id = id).update(orderstatus = 'approved')
    return HttpResponse('<script>alert("Approved
successfully");window.location="/seller_view_order"</script>')
```

REJECT ORDER:

```
def rejectorder(request,id):
    seller_order.objects.filter(id=id).update(orderstatus='reject')
    return HttpResponse('<script>alert("Rejected successfully");
window.location="/seller_view_order"</script>')
```

VIEW APPROVED ORDER:

```
def orderapproved(request):
  res=seller_order.objects.filter(orderstatus__in=['approved', 'orderplaced', 'dispatched', 'shipped', 'out for delivery', 'delayed', 'delivered', 'cancelled', 'returned'], SELLER__LOGIN = request.session['lid'])
   if request.session['login'] == 0:
      return HttpResponse('<script>alert("session expired");window.location="/"</script>')
   return render(request,"seller/approvedorder.html",{'data':res})
```

SELLER ORDER SUB:

```
def seller_order_sub(request,id):
   res=order_sub.objects.filter(SELLER_ORDER=id)
   if request.session['login'] == 0:
      return HttpResponse('<script>alert("session expired");window.location="/"</script>')
   return render(request,"seller/suborder.html",{'data':res})
```

UPDATE ORDER STATUS:

```
def update_order_status(request,id):
   return render(request,"seller/orderstatus.html",{'id':id})
```

UPDATE ORDER STATUS ACTION:

```
def update_order_status_post(request,id):
   status=request.POST['ord']
   print("status",status)
   print("iddddddddddd",id)
   if status == 'dispatched' or status == 'shipped' or status == 'delayed':
      return render(request, "seller/delivery.html",{'id':id,'status':status})
   else:
      res = seller_order.objects.filter(id=id).update(orderstatus=status)
      print(res,"ressssssss")
      return HttpResponse('<script>alert("Updated successfully");
window.location="/orderapproved"</script>')
```

UPDATE DELIVERY ACTION:

```
def update_delivery_post(request,id,status):
    delivery=request.POST['textfield']
    seller_order.objects.filter(id = id).update(deliverydate =delivery ,orderstatus = status)
    return HttpResponse('<script>window.location="/orderapproved"</script>')
```

VIEW REVIEWS OF BOOK:

```
def review_book(request,id):
    res=review.objects.filter(BOOK_id=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"seller/review.html",{'data':res})
```

CUSTOMER

CUSTOMER REGISTRATION:

```
def reg_cust(request):
    return render(request,"customer/register_interface.html")
```

CUSTOMER REGISTRATION ACTION:

```
def reg_cust_post(request):
    image = request.FILES['fileField']
    fs = FileSystemStorage()
    d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
    path = "/static/image/" + d + ".jpg"
    first_name=request.POST['textfield']
    last_name = request.POST['textfield9']
    email=request.POST['textfield2']
    phno=request.POST['textfield3']
    dob=request.POST['textfield4']
    age=request.POST['textfield6']
    gender=request.POST['radio']
    place = request.POST['textfield7']
```

```
    post = request.POST['textfield10']
    pin = request.POST['textfield11']
    password=request.POST['textfield8']
    confirm_password=request.POST['textfield5']
    data = login.objects.filter(username=email)
    if data.exists():
        return HttpResponse('<script>alert("Username al exist");
window.location="/reg_cust"</script>')
    else:
        if password == confirm_password:
            obj1=login()
            obj1.username=email
            obj1.usertype="customer"
            obj1.password=password
            obj1.save()
            obj=customer()
            obj.image=path
            obj.firstname=first_name
            obj.lastname = last_name
            obj.email=email
            obj.phonenumber=phno
            obj.dob=dob
            obj.age=age
            obj.gender=gender
            obj.place=place
            obj.post=post
            obj.pin=pin
            obj.LOGIN_id=obj1.id
            obj.save()
        else:
            return HttpResponse('<script>alert("PASSWORD DOES NOT MATCH!!");
window.location="/reg_cust"</script>')
        return HttpResponse('<script>alert("success");window.location="/"</script>')
```

CUSTOMER INDEX:

```
def cust_index(request):
    return render(request,"customer/index.html")
```

CHANGE PASSWORD:

```
def cust_chpass(request):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/cust_chpass.html")
```

CHANGE PASSWORD ACTION:

```
def cust_chpass_post(request):
    curps=request.POST['textfield']
    nwps=request.POST['textfield2']
    cnfmps=request.POST['textfield3']
    data = login.objects.filter(id = request.session['lid'],password=curps)
    if data.exists():
        if (nwps == cnfmps):
            login.objects.filter(id=request.session['lid']).update(password=cnfmps)
            return HttpResponse('<script>alert("password changed!");
window.location="/cust_index"</script>')
        else:
            return HttpResponse('<script>alert("password not same");
window.location="/cust_chpass"</script>')
    else:
        return HttpResponse('<script>alert("user not found");window.location="/cust_chpass"</script>')
```

VIEW PROFILE:

```
def cust_view(request):
    data = customer.objects.get(LOGIN=request.session['lid'])
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
```

```
    return render(request,"customer/viewprofile.html",{"data":data})
```

EDIT PROFILE:

```
def edit_csprofile(request,id):
    data = customer.objects.get(id=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/edit.html",{"data":data,"id":id})
```

EDIT PROFILE ACTION:

```
def edit_csprofile_post(request,id):
    first_name = request.POST['textfield']
    last_name = request.POST['textfield9']
    email = request.POST['textfield2']
    phno = request.POST['textfield3']
    dob = request.POST['textfield4']
    age = request.POST['textfield6']
    place = request.POST['textfield7']
    post = request.POST['textfield10']
    pin = request.POST['textfield11']
    if 'fileField' in request.FILES:
        image = request.FILES['fileField']
        fs = FileSystemStorage()
        d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
        fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
        path = "/static/image/" + d + ".jpg"
customer.objects.filter(id=id).update(image=path, firstname=first_name, lastname=last_name,
email=email, phonenumber=phno, dob=dob, age=age, place=place, post=post, pin=pin)
        return HttpResponse('<script>alert(" edit successfully");
window.location="/cust_view"</script>')
    else:
customer.objects.filter(id=id).update(firstname=first_name,  lastname=last_name, email=email,
phonenumber=phno, dob=dob, age=age, place=place, post=post, pin=pin)
```

```
    return HttpResponse('<script>alert(" edit successfully");
window.location="/cust_view"</script>')
```

ADD BOOK (CUSTOMER):

```
def csbookadd(request):
    data = category.objects.all()
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/book/add.html",{'data':data})
```

ADD BOOK POST:

```
def csbookadd_post(request):
    bookname=request.POST['textfield']
    author=request.POST['textfield2']
    genre=request.POST['select']
    description=request.POST['textfield5']
    price=request.POST['textfield6']
    discount=request.POST['textfield7']
    image=request.FILES['fileField']
    fs = FileSystemStorage()
    d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
    path = "/static/image/" + d + ".jpg"
    condition=request.POST['textfield11']
    availability=request.POST['textfield13']
    pgno=request.POST['textfield14']
    obj=book()
    obj.book_name=bookname
    obj.author=author
    obj.GENRE_id=genre
    obj.description=description
    obj.price=price
    obj.discount=discount
```

```
    obj.image=path
    obj.bookcondition=condition
    obj.type='customer'
    obj.availability=availability
    obj.numberofpages=pgno
    obj.LOGIN_id = request.session['lid']
    obj.save()
    return HttpResponse('<script>window.location="/csbkview_frontview"</script>')
```

VIEW BOOK:

```
def csbkview(request,id):
    res=book.objects.filter(id = id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/book/bookview.html",{'data':res})
```

VIEW BOOK INTERFACE:

```
def csbkview_frontview(request):
    res=book.objects.filter(LOGIN=request.session['lid'],LOGIN__usertype = 'customer')
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/bkview_interface.html",{'data':res})
```

VIEW BOOK ACTION INTERFACE:

```
def csbkview_frontview_post(request):
    res=book.objects.filter(LOGIN=request.session['lid'],LOGIN__usertype = 'customer',
book_name__icontains=request.POST['ttt'])
    return render(request,"customer/bkview_interface.html",{'data':res})
```

EDIT BOOK:

```
def csbookedit(request,id):
    data = book.objects.get(id = id)
    data1 = category.objects.all()
```

```
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/book/edit.html",{"data2":data,"data1":data1})
```

EDIT BOOK ACTION:

```
def csbookedit_post(request,id):
        description = request.POST['textfield5']
        price = request.POST['textfield6']
        discount = request.POST['textfield7']
        genre = request.POST['select']
        condition = request.POST['textfield11']
        availability = request.POST['textfield13']
        pgno = request.POST['textfield14']
        if 'fileField' in request.FILES:
            image = request.FILES['fileField']
            fs = FileSystemStorage()
            d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
            fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image)
            path = "/static/image/" + d + ".jpg"
            book.objects.filter(id=id).update(description=description, price=price, discount=discount,
GENRE=genre, image=path, bookcondition=condition,  availability=availability,
numberofpages=pgno)
            return HttpResponse('<script>alert(" edit successfully");
window.location="/csbkview_frontview"</script>')
        else:
            book.objects.filter(id=id).update(description=description, price=price, discount=discount,
GENRE=genre, availability=availability, bookcondition=condition, numberofpages=pgno)
            return HttpResponse('<script>window.location="/csbkview_frontview"</script>')
```

DELETE BOOK:

```
def csbkdelete(request,id):
    book.objects.get(id=id).delete()
```

```
    return HttpResponse('<script>window.location="/csbkview_frontview"</script>')
```

VIEW REVIEW ABOUT MY BOOK:

```
def view_review_bout_mybk(request):
    res = review.objects.filter(BOOK__LOGIN=request.session['lid'])
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "customer/review_bout_mybk.html", {'data':res})
```

CUSTOMER BOOK INTERFACE:

```
def view_customer_book_frontview(request):
    res=book.objects.filter(~Q(LOGIN = request.session['lid']),type="customer")
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/bookinterface_sell.html",{'data':res})
```

VIEW CUSTOMER BOOK:

```
def view_customer_book(request,id):
    res=book.objects.filter(id=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "customer/otherbook.html", {'data': res})
```

SELLER BOOK INTERFACE:

```
def view_seller_book_frontview(request):
    res=book.objects.filter(~Q(LOGIN = request.session['lid']),type="seller")
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/sellerbook_interface.html",{'data':res})
```

VIEW SELLER UPLOADED BY BOOK:

```
def view_seller_book(request,id):
    res=book.objects.filter(id=id)
    if request.session['login'] == 0:
```

```
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "customer/details_sellerbk.html", {'data': res})
```

ADD TO CART:

```
def add_cart(request,id):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "customer/quantity_cart.html", {"id": id})
```

ADD TO CART ACTION:

```
def add_cart_post(request,id):
    quantity=request.POST['select']
    q1 = request.POST['select']
    availability1 = book.objects.get(id=id).availability
    print("avvaaill", availability1)
    if int(q1) <= int(availability1):
        obj=cart()
        obj.quantity=quantity
        obj.BOOK_id = id
        obj.CUSTOMER_id = request.session['cid']
        obj.save()
        final_quantity = int(availability1) - int(q1)
        book.objects.filter(id = id).update(availability = final_quantity)
        return HttpResponse('<script>window.location="/view_seller_book_frontview"</script>')
    else:
        return HttpResponse('<script>alert("out of stock");
window.location="/view_seller_book_frontview"</script>')
```

VIEW CART:

```
def view_cart(request):
    res = cart.objects.filter(CUSTOMER=request.session['cid'])
    total = 0
    for i in res:
```

```
        i.total = int(i.quantity) * float(i.BOOK.price) * (1 - float(i.BOOK.discount) / 100)
        total += i.total
    type="seller"
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "customer/cartview.html", {'data': res,'type':type,'total':total})
```

REMOVE FROM CART:

```
def rmv_cart(request,id):
    cart.objects.get(id=id).delete()
    return HttpResponse('<script>window.location="/view_cart"</script>')
```

PLACE ORDER FROM CART:

```
def place_orderfromcart(request,id,cslid,type,amt):
    request.session['typee'] = type
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/view_cart"</script>')
    return render(request,"customer/place_quantitycart.html",{"id": id,'cslid':cslid,'amt':amt})
```

PLACE ORDER FROM CART POST:

```
def place_orderfromcart_post(request,id,cslid,amt):
    pin = request.POST['textfield']
    city = request.POST['textfield4']
    state = request.POST['textfield5']
    house = request.POST['textfield2']
    colony = request.POST['textfield3']
    q1 = request.POST['select']
    discount1 = book.objects.get(id=id).discount
    print("discount", discount1)
    total_amount_withoutdis = float(amt) * int(q1)
    discount_amount = float((total_amount_withoutdis) * float(discount1)) / 100
    final_amount = float(total_amount_withoutdis) - float(discount_amount)
    if str(request.session['typee']) == 'customer':
```

```
        obj = customer_order()
        obj.CUSTOMER_id = customer.objects.get(LOGIN=request.session['lid']).id
        obj.CUSTOMER_se_id = cslid
        obj.date = datetime.datetime.now()
        obj.orderstatus = 'pending'
        obj.paymentstatus = 'pending'
        obj.paymentmethod = 'pending'
        obj.totalamount = final_amount
        obj.discount = book.objects.get(id=id).discount
        obj.place="place"
        obj.pin = pin
        obj.city = city
        obj.state = state
        obj.colony = colony
        obj.house = house
        obj.deliverydate = 'pending'
        obj.return_or_refund = 'pending'
        obj.ordernotes = 'pending'
        obj.ordercancellationreason = 'pending'
        obj.save()
        obj1 = customer_order_sub()
        obj1.CUSTOMER_ORDER_id = obj.id
        obj1.BOOK_id = id
        obj1.quantity = q1
        obj1.save()
    else:
        obj2=seller_order()
        obj2.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id
        obj2.SELLER_id = cslid
        obj2.date = datetime.datetime.now()
        obj2.orderstatus = 'pending'
        obj2.paymentstatus = 'pending'
        obj2.paymentmethod = 'pending'
```

```
        obj2.totalamount = 'pending'
        obj2.discount = book.objects.get(id=id).discount
        obj2.pin = pin
        obj2.city = city
        obj2.state = state
        obj2.colony = colony
        obj2.house = house
        obj2.deliverydate = 'pending'
        obj2.return_or_refund = 'pending'
        obj2.ordernotes = 'pending'
        obj2.ordercancellationreason = 'pending'
        obj2.save()
        obj3=order_sub()
        obj3.SELLER_ORDER = obj2
        obj3.BOOK_id = id
        obj3.quantity = q1
        obj3.save()
    cart.objects.get(BOOK=id,CUSTOMER_id = customer.objects.get(LOGIN =
request.session['lid'])).delete()
    return HttpResponse('<script>window.location="/view_cart"</script>')
```

ORDER VIEW:

```
def cs_order_view(request):
    res =  customer_order.objects.filter(CUSTOMER_se=request.session['cid'],orderstatus = 'pending')
    print(res,'aaaaaaaaaaa')
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request, "customer/customer_order_ view.html", {'data': res})
```

VIEW ORDERED BOOKS:

```
def view_ordersed_books(request,id):
    res=customer_order_sub.objects.filter(CUSTOMER_ORDER=id)
    if request.session['login'] == 0:
```

```
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,'customer/bookitems.html',{'data':res})
```

ORDER SUB:

```
def cs_order_sub(request,id):
    res=customer_order_sub.objects.filter(CUSTOMER_ORDER=id)
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/cs_suborder.html",{'data':res})
```

APPROVED ORDER:

```
def csorderapprove(request,id):
    customer_order.objects.filter(id = id).update(orderstatus = 'approved')
    return HttpResponse('<script>window.location="/cs_order_view"</script>')
```

REJECT ORDER:

```
def csrejectorder(request,id):
    customer_order.objects.filter(id=id).update(orderstatus='reject')
    return HttpResponse('<script>window.location="/cs_order_view"</script>')
```

VIEW APPROVED ORDER:

```
def csorderapproved(request):
    res=customer_order.objects.filter(~Q(CUSTOMER  =
request.session['lid']),orderstatus__in=['approved','orderplaced','dispatched','shipped','out for
delivery','delayed','delivered','returned'])
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
    return render(request,"customer/csapprovedorder.html",{'data':res})
```

UPDATE STATUS:

```
def csupdate_status(request,id):
    if request.session['login'] == 0:
        return HttpResponse('<script>alert("session expired");window.location="/"</script>')
```

```
    return render(request,"customer/csorderstatus.html",{'id':id})
```

UPSATE STATUS ACTION:

```
def csupdate_status_post(request,id):
    status=request.POST['ord']
    print("status",status)
    if status == 'dispatched' or status == 'shipped' or status == 'delayed':
        return render(request, "customer/delivery.html",{'id':id,'status':status})
    else:
        customer_order.objects.filter(id=id).update(orderstatus=status)
        return HttpResponse('<script>window.location="/csorderapproved#aa"</script>')
```

UPDATE DELIVERY ACTION:

```
def csupdate_delivery_post(request,id,status):
    delivery=request.POST['textfield']
    customer_order.objects.filter(id = id).update(deliverydate =delivery ,orderstatus = status)
    return HttpResponse('<script>window.location="/csorderapproved#aa"</script>')
```

PLACE ORDER FROM BOOK:

```
def place_orderfrombook(request,id,cslid,amt):
    return render(request,"customer/quantity_place.html",{"id": id,'cslid':cslid,'amt':amt})
```

PLACE ORDER FROM BOOK ACTION:

```
def place_orderfrombook_post(request,id,cslid,amt):
    pin = request.POST['textfield']
    city = request.POST['textfield4']
    state = request.POST['textfield5']
    house = request.POST['textfield2']
    colony = request.POST['textfield3']
    q1 = request.POST['select']
    print("cslid",cslid)
    availability1 = book.objects.get(id=id).availability
    print("avvaaill", availability1)
```

```
if int(q1) <= int(availability1):
    discount1 = book.objects.get(id=id).discount
    print("discount", discount1)
    total_amount_withoutdis = int(amt) * int(q1)
    discount_amount = float((total_amount_withoutdis) * float(discount1)) / 100
    final_amount = float(total_amount_withoutdis) - (float(discount_amount))
    print(final_amount,"Final amountttttttttttttttttttt")
    obj = customer_order()
    obj.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id
    obj.CUSTOMER_se_id = customer.objects.get(LOGIN = cslid).id
    obj.date = datetime.datetime.now()
    obj.orderstatus = 'pending'
    obj.paymentstatus = 'pending'
    obj.paymentmethod = 'pending'
    obj.totalamount = final_amount
    obj.discount = book.objects.get(id=id).discount
    obj.pin = pin
    obj.city = city
    obj.state = state
    obj.colony = colony
    obj.house = house
    obj.deliverydate = 'pending'
    obj.return_or_refund = 'pending'
    obj.ordernotes = 'pending'
    obj.ordercancellationreason = 'pending'
    obj.save()
    obj1 = customer_order_sub()
    obj1.CUSTOMER_ORDER = obj
    obj1.BOOK_id = id
    obj1.quantity = q1
    obj1.save()
    final_quantity = int(availability1)-int(q1)
    book.objects.filter(id = id).update(availability = final_quantity)
```

```
        return HttpResponse('<script>window.location="/view_customer_book_frontview"</script>')
    else:
        return HttpResponse('<script>alert("out of stock");
window.location="/view_customer_book_frontview"</script>')
```

PLACE ORDER FROM BOOK UPLOADED BY SELLER:

```
def place_orderfrombookseller(request,id,slid,amt):
    return render(request,"customer/placeorder_sellerbook.html",{"id": id,'slid':slid,'amt':amt})
```

PLACE ORDER FROM BOOK UPLOADED BY SELLER ACTION:

```
def place_orderfrombookseller_post(request,id,slid,amt):
    pin = request.POST['textfield']
    city = request.POST['textfield4']
    state = request.POST['textfield5']
    house = request.POST['textfield2']
    colony = request.POST['textfield3']
    q1 = request.POST['select']
    availability1 = book.objects.get(id = id).availability
    print("avvaaill",availability1)
    if int(q1) <=  int(availability1) :
        discount1 = book.objects.get(id=id).discount
        print("discount", discount1)
        total_amount_withoutdis = int(amt) * int(q1)
        discount_amount = float((total_amount_withoutdis) * float(discount1)) / 100
        final_amount = float(total_amount_withoutdis) - float(discount_amount)
        obj2 = seller_order()
        obj2.CUSTOMER_id = customer.objects.get(LOGIN=request.session['lid']).id
        obj2.SELLER_id = seller.objects.get(LOGIN=slid).id
        obj2.date = datetime.datetime.now()
        obj2.orderstatus = 'pending'
        obj2.paymentstatus = 'pending'
        obj2.paymentmethod = 'pending'
        obj2.totalamount = final_amount
```

```
        obj2.discount = book.objects.get(id=id).discount
        obj2.pin = pin
        obj2.city = city
        obj2.state = state
        obj2.colony = colony
        obj2.house = house
        obj2.deliverydate = 'pending'
        obj2.return_or_refund = 'pending'
        obj2.ordernotes = 'pending'
        obj2.ordercancellationreason = 'pending'
        obj2.save()
        obj3 = order_sub()
        obj3.SELLER_ORDER = obj2
        obj3.BOOK_id = id
        obj3.quantity = q1
        obj3.save()
        final_quantity = int(availability1) - int(q1)
        book.objects.filter(id=id).update(availability=final_quantity)
        return HttpResponse('<script>alert("Order Placed!!");
window.location="/view_seller_book_frontview"</script>')
    else:
        return HttpResponse('<script>alert("out of stock");
window.location="/view_customer_book_frontview"</script>')
```

VIEW MY ORDER:

```
def view_my_order(request):
    res=customer_order.objects.filter(CUSTOMER = request.session['cid'],orderstatus__in = [
'approved', 'orderplaced', 'dispatched', 'shipped', 'out for delivery', 'delayed','delivered', 'cancelled',
'returned'])
    data=seller_order.objects.filter(CUSTOMER = request.session['cid'],orderstatus__in = [ 'approved',
'orderplaced', 'dispatched', 'shipped', 'out for delivery', 'delayed','delivered', 'cancelled', 'returned'])
    return render(request, "customer/myorder.html", {'res': res,'data1':data})
```

VIEW MY ORDER ACTION:

```
def view_my_order_post(request):
    select1 = request.POST['select']
    res = customer_order.objects.filter(CUSTOMER=request.session['cid'], orderstatus__in=[
'approved', 'orderplaced', 'dispatched', 'shipped', 'out for delivery', 'delayed','delivered', 'cancelled',
'returned'],)
    data = seller_order.objects.filter(CUSTOMER=request.session['cid'],orderstatus__in = [ 'approved',
'orderplaced', 'dispatched', 'shipped', 'out for delivery', 'delayed','delivered', 'cancelled', 'returned'])
    return render(request, "customer/myorder.html", {'res': res, 'data1': data,'select1':select1})
```

VIEW MY ORDER SUB:

```
def my_order_sub(request,id):
    res=customer_order_sub.objects.filter(CUSTOMER_ORDER=id)
    return render(request,"customer/myordersub.html",{'data':res})
```

VIEW SELLER ORDER SUB:

```
def my_seller_order_sub(request,id):
    res = order_sub.objects.filter(SELLER_ORDER=id)
    return render(request,"customer/seller_sub_order.html",{'data':res})
```

ORDER CANCEL: (CUSTOMER)

```
def my_cust_order_cancel(request,id):
    return render(request, "customer/cancellation_reason.html",{'id':id})
```

ORDER CANCEL ACTION:

```
def my_cust_order_cancel_post(request,id):
    reason=request.POST['textarea']
    customer_order.objects.filter(id=id).update(ordercancellationreason=reason,
orderstatus='cancelled')
    return HttpResponse('<script>alert("Order Cancelled!!");
window.location="/view_my_order"</script>')
```

ORDER CANCEL: (SELLER)

```
def my_seller_order_cancel(request,id):
```

```
    return render(request, "customer/seller_order_cancellation.html", {'id':id})
```

ORDER CANCEL ACTION: (SELLER)

```
def my_seller_order_cancel_post(request,id):
    reason=request.POST['textarea']
    seller_order.objects.filter(id=id).update(ordercnclreason=reason, orderstatus='cancelled')
    return HttpResponse('<script>alert("Order Cancelled!!");
window.location="/view_my_order"</script>')
```

VIEW ONLINE BOOK INTERFACE:

```
def cs_onlinebkview_frontview(request):
    res=online_book.objects.filter(type="non-academic")
    return render(request,"customer/onlinebook_interface.html",{'data':res})
```

STUDY MATERIAL INTERFACE:

```
def studymaterial_frontview(request):
    res=online_book.objects.filter(type="academic")
    return render(request,"customer/studymaterial_frontview.html",{'data':res})
```

STUDY MATERIAL ACTION INTERFACE

```
def studymaterial_frontview_post(request):
    res=online_book.objects.filter(type="academic",subject = request.POST['select2'])
    return render(request,"customer/studymaterial_frontview.html",{'data':res})
```

VIEW ACADEMIC BOOK:

```
def cs_academicbk_view(request,id):
    res=online_book.objects.filter(type="academic",id=id )
    return render(request,"customer/cs_academicbk_view.html",{'data':res})
```

VIEW ONLINE BOOK INTERFACE:

```
def cs_onlinebkview_frontview_post(request):
    res=online_book.objects.filter(type="non-academic",subject__contains = request.POST['select2'])
```

```
    return render(request,"customer/onlinebook_interface.html",{'data':res})
```

VIEW ONLINE BOOK:

```
def cs_onlinebkview(request,id):
    res=online_book.objects.filter(type="non-academic" ,id = id)
    return render(request,"customer/nonacademic_bkview.html",{'data':res})
```

VIEW ONLINE BOOK ACTION:

```
def cs_onlinebkview_post(request):
    search1 = request.POST['select2']
    newres = []
    res = online_book.objects.filter(type="non-academic",book_name__contains=search1)
    for i in res:
        if str(search1) in i.book_name:
            newres.append(i)
    return render(request, "customer/nonacademic_bkview.html", {'data': res})
```

ADD REVIEW:

```
def add_review(request,id):
    return render(request,"customer/send_review.html",{'id':id})
```

ADD REVIEW ACTION:

```
def add_review_post(request,id):
    revw=request.POST['textfield3']
    rating=request.POST['star']
    obj=review()
    obj.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id
    obj.revw=revw
    obj.date = datetime.datetime.now()
    obj.BOOK_id= id
    obj.rating=rating
    obj.save()
```

```
    return HttpResponse("<script>alert('review send successfully');
window.location='/cust_index'</script>")
```

VIEW ALL REVIEWS:

```
def view_allreview(request,id):
    res = review.objects.filter(BOOK_id=id)
    cur_cust = customer.objects.get(LOGIN = request.session['lid'])
    return render(request, "customer/view_allreview.html", {'data':res,'id':cur_cust})
```

VIEW MY REVIEWS:

```
def view_own_review(request):
    data = review.objects.filter(CUSTOMER=request.session['cid'])
    return render(request,'customer/csreview.html',{'data':data})
```

EDIT MY REVIEW:

```
def edit_review(request,id):
    view=review.objects.get(id=id)
    return render(request,"customer/edit_review.html",{"view":view})
```

EDIT MY REVIEW ACTION:

```
def edit_review_post(request,id):
    revw1 = request.POST['textfield3']
    rating1 = request.POST['textfield5']
    review.objects.filter(id=id).update(revw=revw1, rating=rating1, date=datetime.datetime.now())
    return HttpResponse('<script>alert(" edited successfully");
window.location="/view_own_review#aa"</script>')
```

DELETE REVIEW:

```
def cs_dlt_review(request,id):
    review.objects.get(id=id).delete()
    return HttpResponse('<script>alert("delete
successfully");window.location="/view_own_review#aa"</script>')
```

ADD ONLINE BOOK REVIEW:

```
def add_onlinebk_review(request,id):
    return render(request,"customer/send_onlinebkreview.html",{'id':id})
```

ADD ONLINE BOOK REVIEW ACTION:

```
def add_onlinebkreview_post(request,id):
    revw=request.POST['textfield3']
    rating=request.POST['textfield5']
    obj=onlinebk_review()
    obj.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id
    obj.revw=revw
    obj.date = datetime.datetime.now()
    obj.ONLINE_BOOK_id= id
    obj.rating=rating
    obj.save()
    return HttpResponse("<script>alert('review send successfully');
window.location='/cs_onlinebkview_frontview'</script>")
```

VIEW ALL REVIEWS OF ONLINE BOOK:

```
def view_all_onlinebk_review(request,id):
    res = onlinebk_review.objects.all()
    cur_cust = customer.objects.get(LOGIN = request.session['lid'])
    return render(request, "customer/view_all_onlinebk_review.html", {'data':res,'id':cur_cust})
```

VIEW MY REVIEW ON ONLINE BOOK:

```
def view_own_onlinebk_review(request):
    data = onlinebk_review.objects.filter(CUSTOMER=request.session['cid'])
    return render(request,'customer/cs_own_onlinebk_review.html',{'data':data})
```

EDIT MY REVIEW ABOUT ONLINE BOOK:

```
def edit_onlinebk_review(request,id):
    view=onlinebk_review.objects.get(id=id)
```

```
    return render(request,"customer/edit_onlinebk_review.html",{"view":view})
```

EDIT MY REVIEW ABOUT ONLINE BOOK ACTION:

```
def edit_onlinebk_review_post(request,id):
    revw1 = request.POST['textfield3']
    rating1 = request.POST['textfield5']
    onlinebk_review.objects.filter(id = id).update(revw = revw1, rating = rating1, date =
datetime.datetime.now())
    return HttpResponse('<script>alert(" edited successfully");
window.location="/view_own_onlinebk_review#aa"</script>')
```

DELETE ONLINE BOOK REVIEW:

```
def cs_dlt_onlinebk_review(request,id):
    onlinebk_review.objects.get(id=id).delete()
    return HttpResponse('<script>alert("delete
successfully");window.location="/view_own_onlinebk_review#aa"</script>')
```

ADD QUESTION:

```
def add_qtn(request,id):
    return render(request, "customer/question_add.html", {'id': id})
```

ADD QUEESTION ACTION:

```
def add_qtn_post(request,id):
    qtn=request.POST['textfield']
    opt_a=request.POST['textfield2']
    opt_b=request.POST['textfield3']
    opt_c=request.POST['textfield4']
    opt_d=request.POST['textfield5']
    ans=request.POST['select']
    if ans == 'OPTION A':
        obj = question()
        obj.ONLINE_BOOK_id = id
        obj.CUSTOMER_id = customer.objects.get(LOGIN=request.session['lid']).id
        obj.questions = qtn
```

```
        obj.option_a = opt_a
        obj.option_b = opt_b
        obj.option_c = opt_c
        obj.option_d = opt_d
        obj.answers = opt_a
        obj.save()
    elif ans == 'OPTION B':
        obj = question()
        obj.ONLINE_BOOK_id = id
        obj.CUSTOMER_id = customer.objects.get(LOGIN=request.session['lid']).id
        obj.questions = qtn
        obj.option_a = opt_a
        obj.option_b = opt_b
        obj.option_c = opt_c
        obj.option_d = opt_d
        obj.answers = opt_b
        obj.save()
    elif ans == 'OPTION C':
        obj = question()
        obj.ONLINE_BOOK_id = id
        obj.CUSTOMER_id = customer.objects.get(LOGIN=request.session['lid']).id
        obj.questions = qtn
        obj.option_a = opt_a
        obj.option_b = opt_b
        obj.option_c = opt_c
        obj.option_d = opt_d
        obj.answers = opt_c
        obj.save()
    else:
        obj = question()
        obj.ONLINE_BOOK_id = id
        obj.CUSTOMER_id = customer.objects.get(LOGIN=request.session['lid']).id
        obj.questions = qtn
```

```
        obj.option_a = opt_a
        obj.option_b = opt_b
        obj.option_c = opt_c
        obj.option_d = opt_d
        obj.answers = opt_d
        obj.save()
    return
HttpResponse('<script>alert("added");window.location="/cs_academicbk_view/'+str(id)+'"</script>')
```

VIEW QUESTION:

```
def view_question(request,id):
    request.session['qid'] = id
    res=question.objects.filter(CUSTOMER=request.session['cid'],ONLINE_BOOK = id)
    return render(request,"customer/question_vw.html",{'data':res})
```

EDIT QUESTION:

```
def edit_question(request,id):
    view=question.objects.get(id=id)
    qid = request.session['qid']
    return render(request,"customer/edit_qtn.html",{"view":view,'qid':qid})
```

EDIT QUESTION ACTION:

```
def edit_question_post(request,id,qid):
    request.session['qid']=qid
    print("qid",qid)
    qtn = request.POST['textfield']
    opt_a = request.POST['textfield2']
    opt_b = request.POST['textfield3']
    opt_c = request.POST['textfield4']
    opt_d = request.POST['textfield5']
    ans = request.POST['select']
    if ans == 'OPTION A':
        question.objects.filter(id =id).update(answers = opt_a)
```

```
    elif ans == 'OPTION B':
        question.objects.filter(id=id).update(answers=opt_b)
    elif ans == 'OPTION C':
        question.objects.filter(id=id).update(answers=opt_c)
    else:
        question.objects.filter(id=id).update(answers=opt_d)
    question.objects.filter(id=id).update(questions=qtn, option_a=opt_a, option_b=opt_b,
option_c=opt_c, option_d=opt_d,)
    return HttpResponse(f'<script>alert("Edited
successfully");window.location="/view_question/'+str(qid)+'#aa"</script>')
```

DELETE QUESTION:

```
def delete_qtn(request,id):
    qid = request.session['qid']
    question.objects.get(id=id).delete()
    return HttpResponse('<script>alert("deleted
successfully");window.location="/view_question/'+str(qid)+'#aa"</script>')
```

ATTEND QUESTION:

```
def attend_qtn(request, id):
    try:
res=question.objects.filter(Q(ONLINE_BOOK_id=id),~Q(CUSTOMER=customer.objects.get(LOGI
N_id=request.session['lid'])))[0]
        request.session['cnt'] = 0
        request.session['score'] = 0
        return render(request, "customer/attend_qtn.html", {'data': res, 'score': request.session['score'],
'id': id})
    except:
        return HttpResponse('<script>alert("No data");
window.location="/cs_academicbk_view/'+str(id)+'"</script>')
```

ATTEND QUESTION POST:

```
def attend_qtn_post(request, id):
    res=question.objects.filter(ONLINE_BOOK_id=id)
    corr_ans = request.POST['ans']
```

```
    ans = request.POST['radio']
    if corr_ans == ans:
        scr=request.session['score']
        request.session['score']=int(scr)+1
    request.session['cnt']=int(request.session['cnt'])+1
    if int(request.session['cnt'])<len(res):
        return render(request, "customer/attend_qtn.html",
                    {'data': res[int(request.session['cnt'])], 'score': request.session['score'], 'id': id})
    else:
        res2=result.objects.filter(ONLINE_BOOK_id=id,
CUSTOMER__LOGIN_id=request.session['lid'])
        if res2.exists():
            res2=res2[0]
        else:
            res2=result()
        res2.date=datetime.datetime.now().date()
        res2.marks=request.session['score']
        res2.ONLINE_BOOK_id=id
        res2.CUSTOMER=customer.objects.get(LOGIN_id=request.session['lid'])
        res2.save()
        return HttpResponse('<script>alert("Score added");
window.location="/cs_academicbk_view"</script>')
```

VIEW MY THOUGHTWAVES:

```
def view_mythoughtwaves(request,id):
    res=cust_thoughtwaves.objects.filter(CUSTOMER=request.session['cid'],id=id)
    for i in res:
        i.like = like.objects.filter(THOUGHTNAME=i.id).count()
    return render(request,"customer/mythoughtwaves_view.html",{'data':res})
```

ADD MY THOUGHTWAVES:

```
def add_mythoughtwaves(request):
    return render(request,"customer/mythoughtwaves_add.html")
```

ADD MY THOUGHTWAVES ACTION:

```
def add_mythoughtwaves_post(request):
    title=request.POST['textfield']
    content=request.POST['textarea']
    category=request.POST['select']
    image1 = request.FILES['image']
    fs = FileSystemStorage()
    d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image1)
    path = "/static/image/" + d + ".jpg"
    obj=cust_thoughtwaves()
    obj.thoughtname=title
    obj.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id
    obj.content=content
    obj.image = path
    obj.publish_date = datetime.datetime.now()
    obj.category=category
    obj.save()
    return
HttpResponse("<script>alert('uploaded');window.location='/mythoughtwave_frontview'</script>")
```

EDIT MY THOUGHTWAVES:

```
def edit_mythoughtwaves(request,id):
    view=cust_thoughtwaves.objects.get(id=id)
    return render(request,"customer/mythoughtwave_edit.html",{"view":view})
```

EDIT MY THOUGHTWAVES ACTION:

```
def edit_mythoughtwaves_post(request,id):
    try:
        title = request.POST['textfield']
        content = request.POST['textarea']
        category = request.POST['select']
```

```
        image1 = request.FILES['image']
        fs = FileSystemStorage()
        d = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
        fs.save(r"C:\Users\hilus\PycharmProjects\eon\eonapp\static\image\\" + d + ".jpg", image1)
        path = "/static/image/" + d + ".jpg"
        cust_thoughtwaves.objects.filter(id=id).update(thoughtname = title,content =
content,category=category,publish_date = datetime.datetime.now(),image=path)
        return HttpResponse('<script>alert(" edit
successfully");window.location="/mythoughtwave_frontview"</script>')
    except Exception as e:
        title = request.POST['textfield']
        content = request.POST['textarea']
        category = request.POST['select']
        cust_thoughtwaves.objects.filter(id=id).update(thoughtname = title,content =
content,category=category,publish_date = datetime.datetime.now())
        return HttpResponse('<script>alert(" edit
successfully");window.location="/mythoughtwave_frontview"</script>')
```

DELETE THOUGHTWAVES:

```
def delete_mythoughtwaves(request,id):
    cust_thoughtwaves.objects.get(id=id).delete()
    return HttpResponse('<script>alert("deleted
successfully");window.location="/mythoughtwave_frontview"</script>')
```

MY THOUGHTWAVES INTERFACE:

```
def mythoughtwave_frontview(request):
    res=cust_thoughtwaves.objects.filter(CUSTOMER__LOGIN_id=request.session['lid'])
    return render(request,"customer/mythoughtwave_frontview.html",{'data':res})
```

ALL THOUGHTWAVES:

```
def all_thoughtwaves(request,id):
    res = cust_thoughtwaves.objects.filter(id=id)
    cur_cust = customer.objects.get(LOGIN=request.session['lid'])
    for i in res:
```

```
        i.like = like.objects.filter(THOUGHTNAME=i.id).count()
        like_new = like.objects.filter(THOUGHTNAME=i.id,CUSTOMER = request.session['cid'])
        if like_new.exists():
            i.flag = 1
        else:
            i.flag = 0

    return render(request, "customer/view_allthoughtwave.html", {'data': res, 'id': cur_cust})
```

ALL THOUGHTWAVES INTERFACE:

```
def allthoughtwave_frontview(request):
    res=cust_thoughtwaves.objects.all()
    return render(request,"customer/all_thoughtwave_frontview.html",{'data':res})
```

ADD COMMENT :

```
def add_comment(request,id):
    request.session['tid'] = id
    res = comment.objects.filter(THOUGHTNAME_id=id)
    data1 = comment.objects.filter(CUSTOMER = request.session['cid'],THOUGHTNAME_id=id)
    if data1.exists():
        flag = 1
    else:
        flag = 0
    return render(request, "customer/tw_comment_add.html",{'res':res,'id':id,'flag':flag})
```

ADD COMMENT ACTION:

```
def add_comment_post(request,id):
    tid = request.session['tid']
    cmt=request.POST['textarea']
    obj=comment()
    obj.THOUGHTNAME_id = id
    obj.cmts=cmt
    obj.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id
```

```
    obj.date = datetime.datetime.now()

    obj.save()

    return HttpResponse(f'<script>alert("added
successfully");window.location="/add_comment/{tid}#aa";</script>')
```

DELETE COMMENT:

```
def delete_comment(request,id):

    tid = request.session['tid']

    comment.objects.get(id=id).delete()

    return HttpResponse(f'<script>alert("deleted
successfully");window.location="/add_comment/{tid}#aa";</script>')
```

THOUGHTWAVES LIKE:

```
def tw_like(request,id):

    obj=like()

    obj.THOUGHTNAME_id = id

    obj.CUSTOMER_id = customer.objects.get(LOGIN = request.session['lid']).id

    obj.date = datetime.datetime.now()

    obj.save()

    return
HttpResponse(f'<script>alert("liked");window.location="/all_thoughtwaves/{id}#cc";</script>')
```

SCORE:

```
def score(request,id):

    res=result.objects.get(ONLINE_BOOK =id,CUSTOMER = request.session['cid'])

    return render(request,"customer/score.html",{'data':res})
```

CHATBOT:

```
def chatbott(request):

    return render(request, "customer/chatbot.html")
```

SEND CHAT:

```
def chatsnd(request):

        m=request.POST['msg']
```

```
    obj=chatbot()
    obj.type="user"
    obj.date=datetime.datetime.now().strftime("%Y-%m-%d")
    obj.chat=m
    obj.CUSTOMER_id=request.session['cid']
    obj.save()
    resp=get_Response(m)
    print("Chatbot : ", resp)
    obj = chatbot()
    obj.type = "chatbot"
    obj.date = datetime.datetime.now().strftime("%Y-%m-%d")
    obj.chat = resp
    obj.CUSTOMER_id = request.session['cid']
    obj.save()
    print("DDDDDD")
    return JsonResponse({'status':'ok'})
```

CHAT REPLY:

```
def chatrply(request):
    res = chatbot.objects.filter(CUSTOMER = request.session['cid'])
    data=[]
    for i in res:
        data.append({'id':i.id, 'message':i.chat, 'date':i.date, 'type':i.type})
    print(data)
    return JsonResponse({'status':"ok", "data":data})
```

TURN PAGES:

```
def turn_pages(request,c):
  c= online_book.objects.get(id=c).content
  print(c)
```

PAYMENT:

```
def payment(request,id,amt):
```

```
    request.session['oid']=id
    import razorpay
    razorpay_api_key = "rzp_test_MJOAVy77oMVaYv"
    razorpay_secret_key = "MvUZ03MPzLq3lkvMneYECQsk"
    razorpay_client = razorpay.Client(auth=(razorpay_api_key, razorpay_secret_key))
    amount = float(amt)*100
    order_data = {
        'amount': amount,
        'currency': 'INR',
        'receipt': 'order_rcptid_11',
        'payment_capture': '1',  # Auto-capture payment
    }
    order = razorpay_client.order.create(data=order_data)
    context = {
        'razorpay_api_key': razorpay_api_key,
        'amount': order_data['amount'],
        'currency': order_data['currency'],
        'order_id': order['id'],
    }
    if customer_order.objects.filter(id = id).exists():
        customer_order.objects.filter(id=id).update(paymentmethod="online", paymentstatus="paid")
    else:
        seller_order.objects.filter(id=id).update(paymentmethod="online", paymentstatus="paid")
        return render(request, 'customer/payment.html',{'razorpay_api_key': razorpay_api_key, 'amount':
order_data['amount'], 'currency': order_data['currency'], 'order_id': order['id']})
```

7. TESTING

7.1 TEST CASES

The application underwent manual testing, where each user interface was subjected to tests with both valid and invalid inputs to verify the accuracy of the application's output and error messages. The application demonstrated effective responsiveness to various input types, producing correct outputs and appropriate error messages for different scenarios.

LOGIN

In the login scenario, we use the password and username as inputs. If both inputs are valid, the login is successful, allowing the user to access the home page. However, if either the username or password is invalid, or if one of them is missing, the login will be unsuccessful.

SELLER REGISTRATION

In seller registration, personal details and identification document, are required fields. Successful registration occurs when all fields are filled and valid. However, if one or more fields are missing, or if the name is not in alphabets and the phone number is below ten digits, an error message will be displayed, and registration will not proceed. Additionally, the password and confirm password must match; otherwise, an error message will be shown. Incomplete or invalid entries for details such as experience, identification proof, and work category will also result in unsuccessful registration.

CUSTOMER REGISTRATION

In customer registration, personal details like name, email, address are required fields. Successful registration occurs when all fields are filled and valid. However, if one or more fields are missing, or if the name is not in alphabets and the phone number is below ten digits, an error message will be displayed, and registration will not proceed. Additionally, the password and confirm password must match; otherwise, an error message will be shown. It will result in unsuccessful registration.

ADDING BOOK

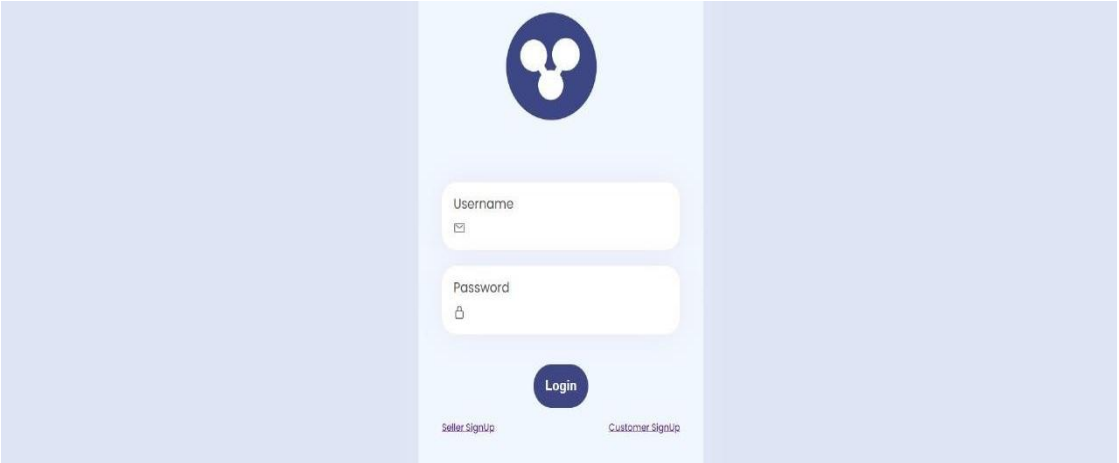
In the book adding process, it includes book name, author, type, description, image, content, language, number of pages are required fields. Added book occurs when all fields are filled and valid. If one or more fields are missing, or if the name is not in alphabets and the content is not uploaded, an error message will be displayed, and cannot add book.

7.2 TESTRESULTS

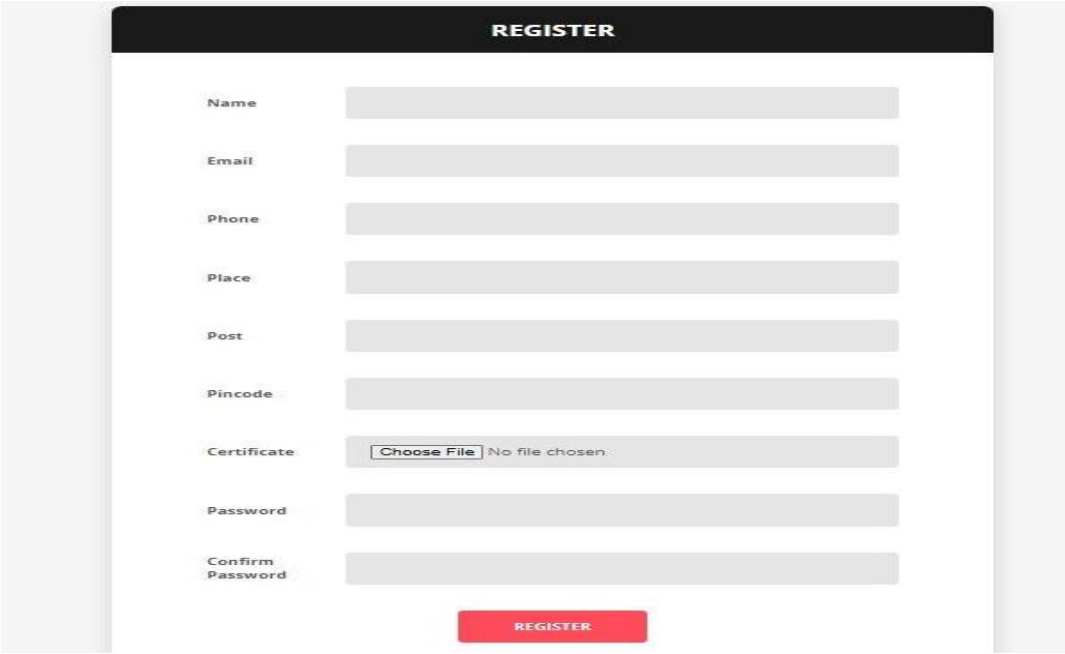
SL.NO	FIELDS	VALID/INVALID	EXPECTED RESULT	REMARKS
1.Login	Username & Password	Valid Invalid	Login successfully Invalid username & password	Success Error
2.Seller Registration	Personal Details	Valid Invalid	Registration successfully Invalid entry	Success Error
3.Customer Registration	Personal Details	Valid Invalid	Registration successfully Invalid entry	Success Error
4.Book Adding	Book details	Valid Invalid	Book Added Please fill out the fields	Success Error

8.SCREENSHOTS

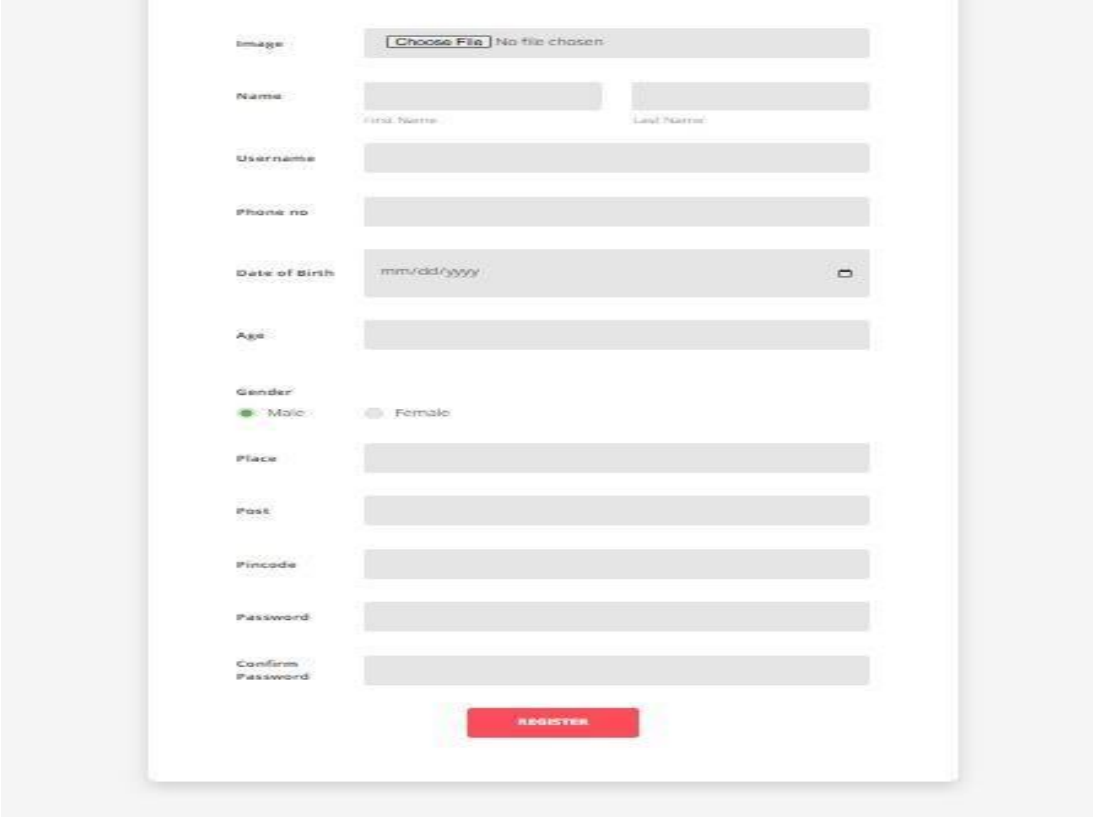
LOGIN PAGE:



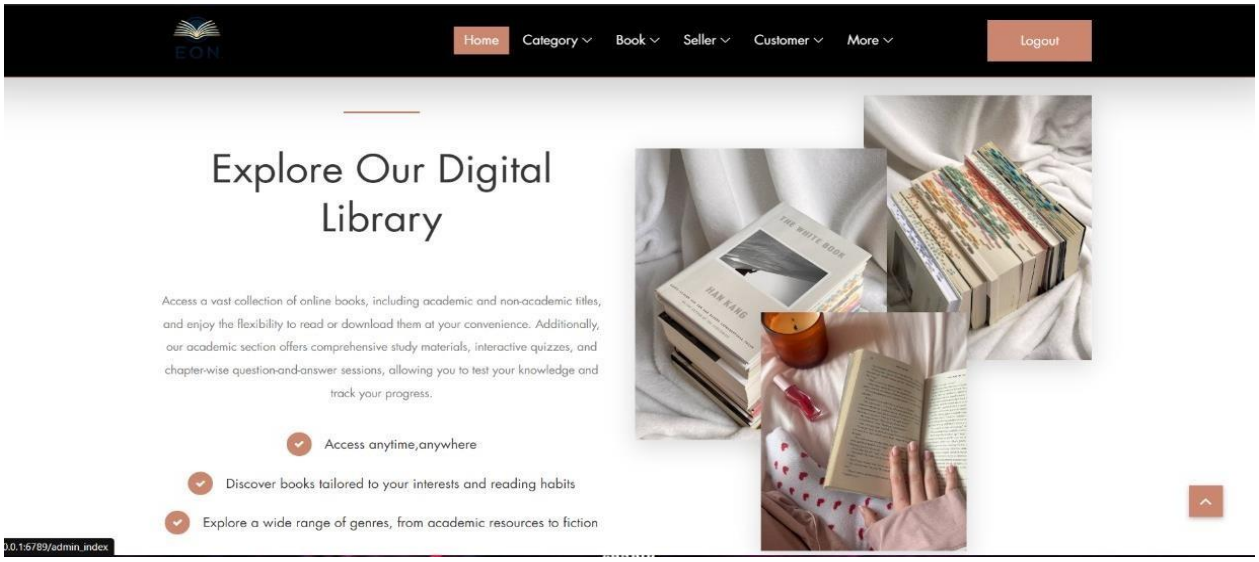
SELLER REGISTER

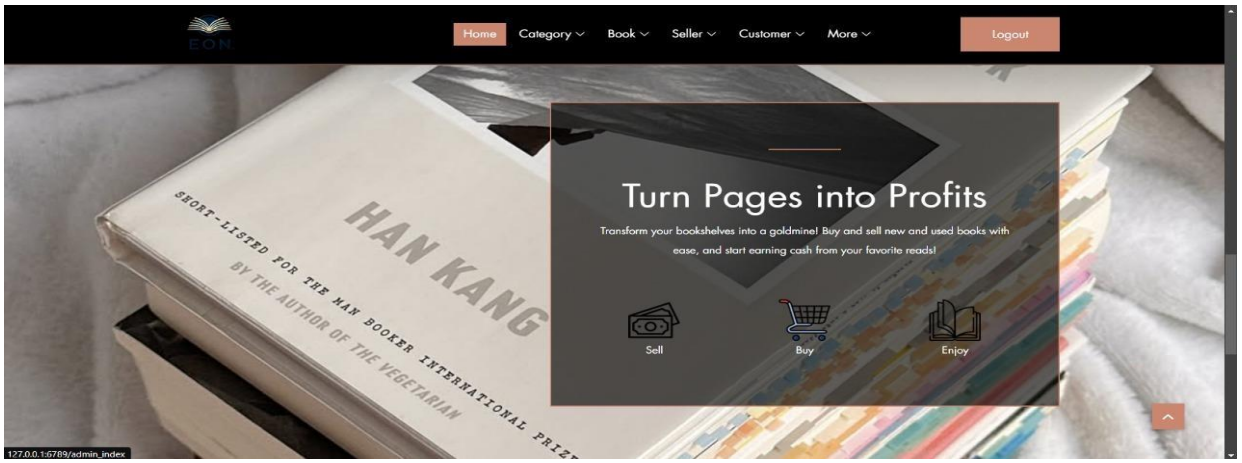


CUSTOMER REGISTER:

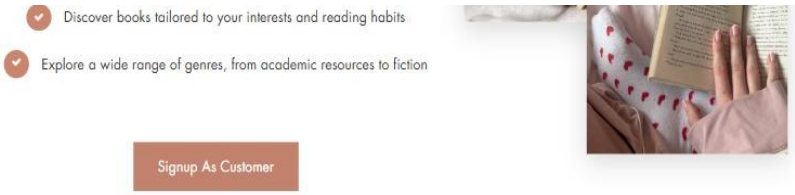


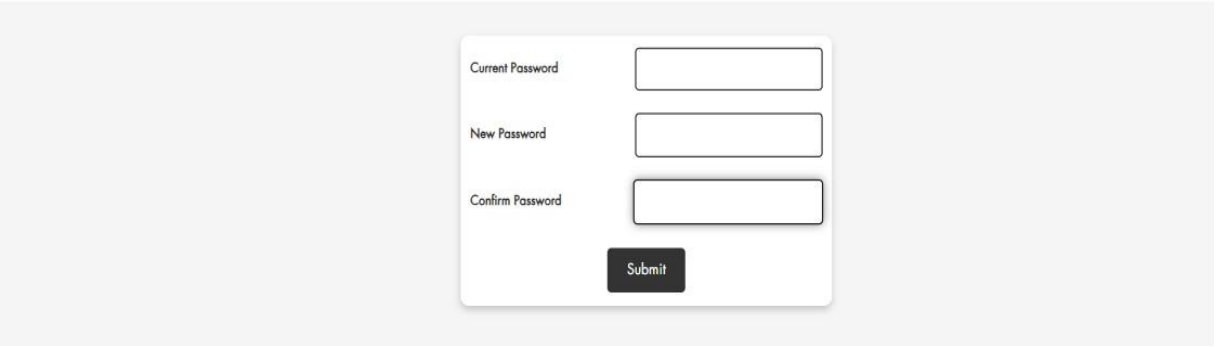
HOME:



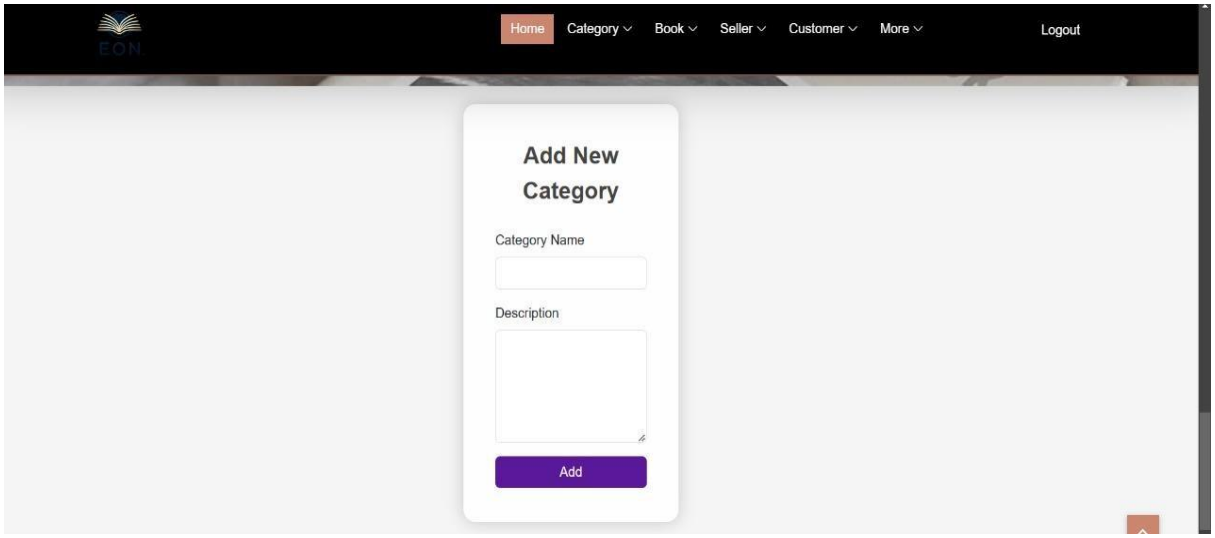


CHANGE PASSWORD

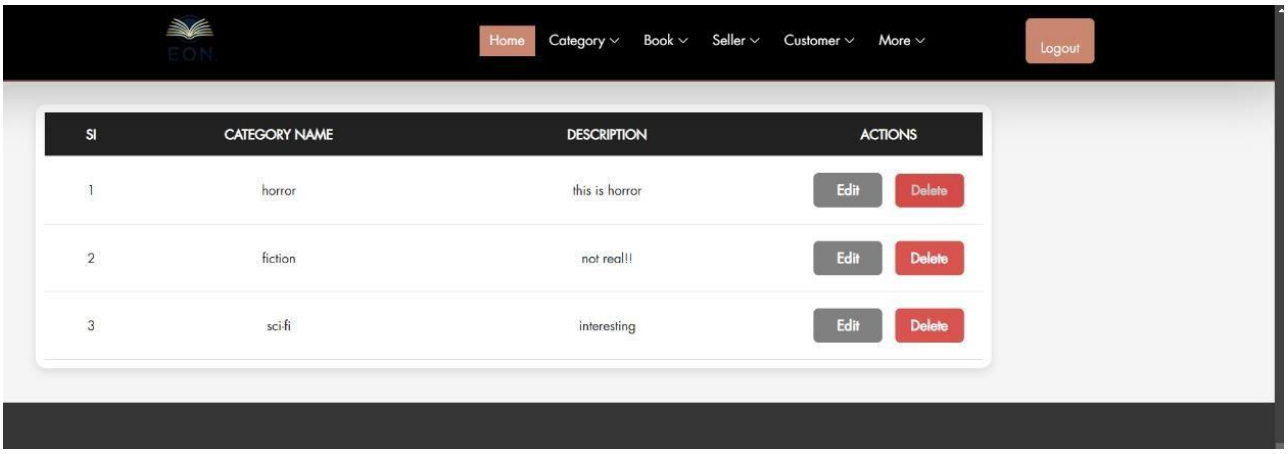




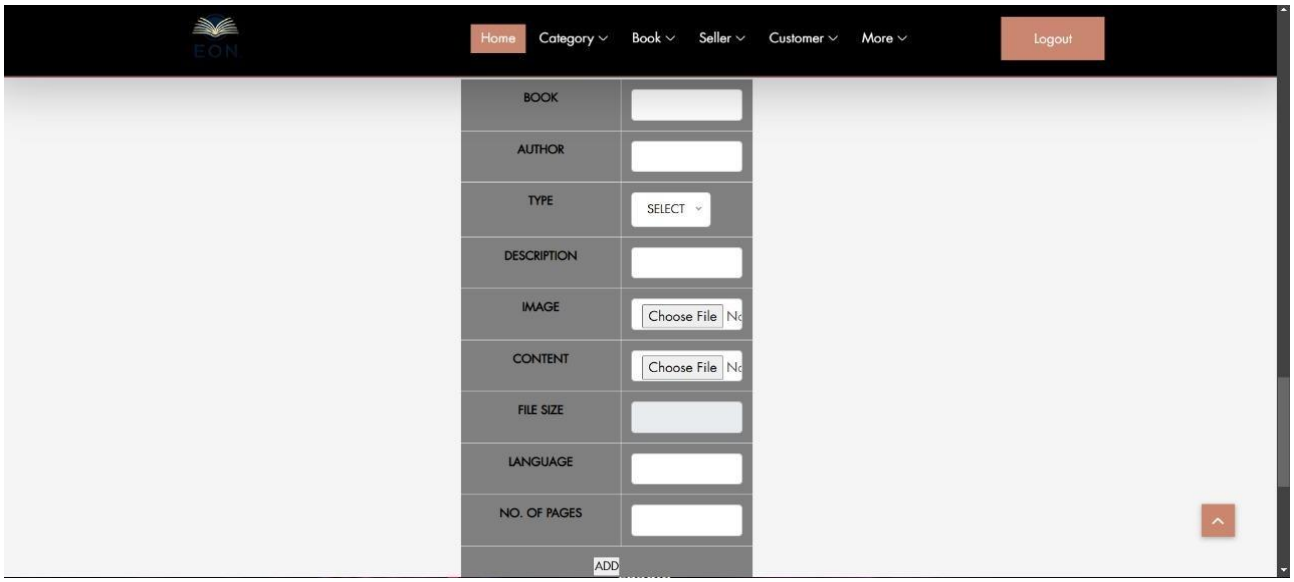
ADD CATEGORY



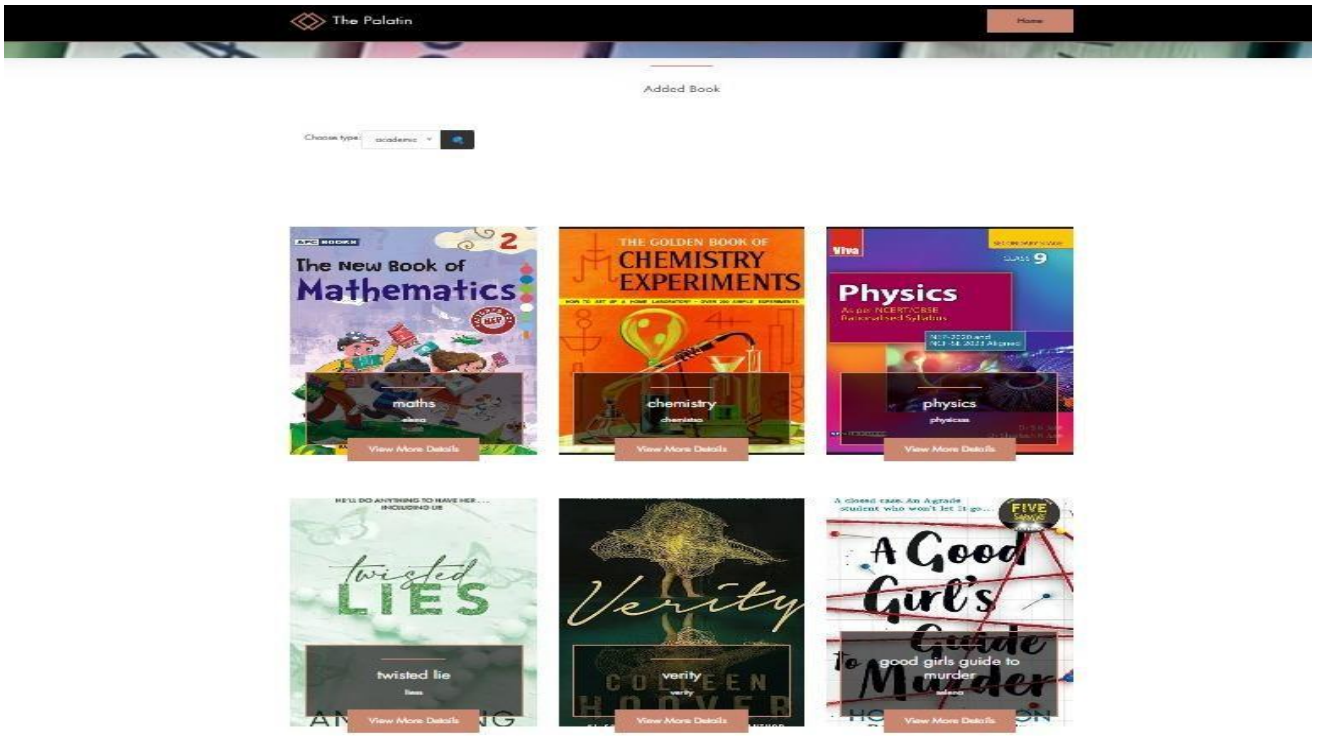
VIEW CATEGORY



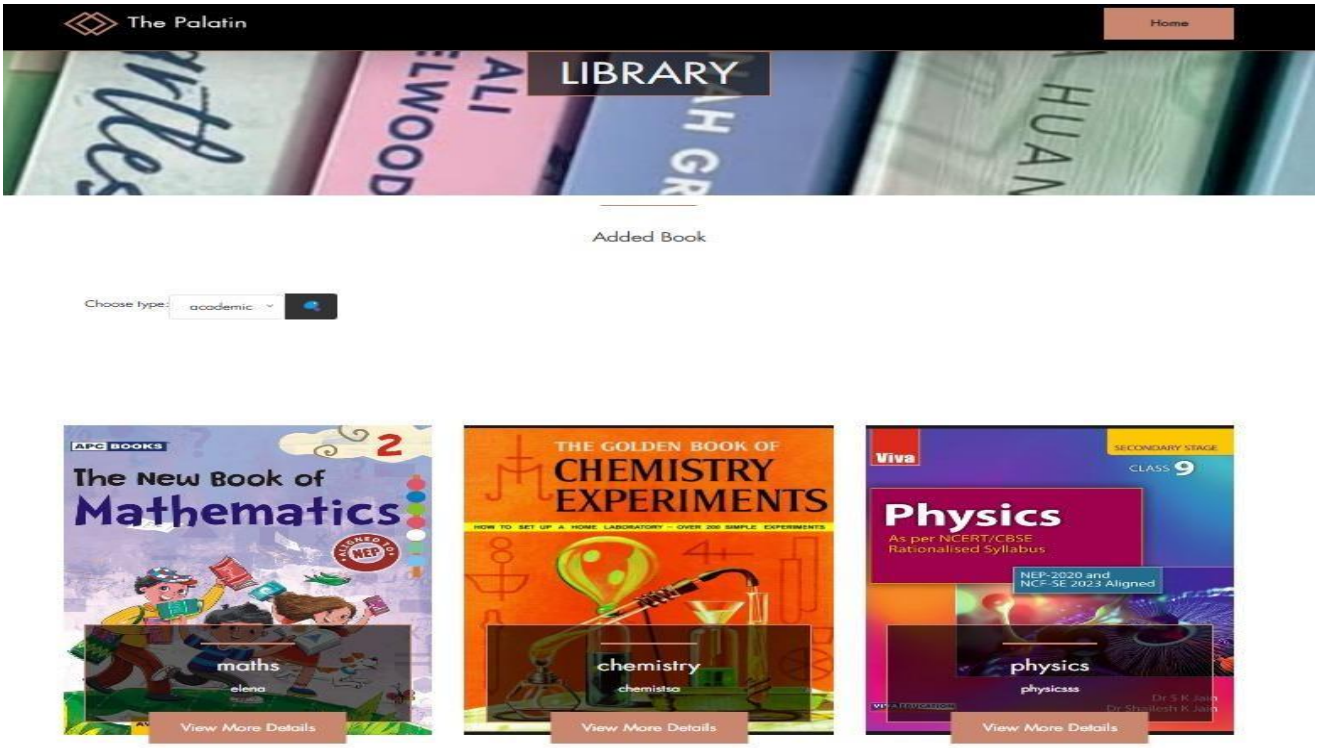
ADD BOOK



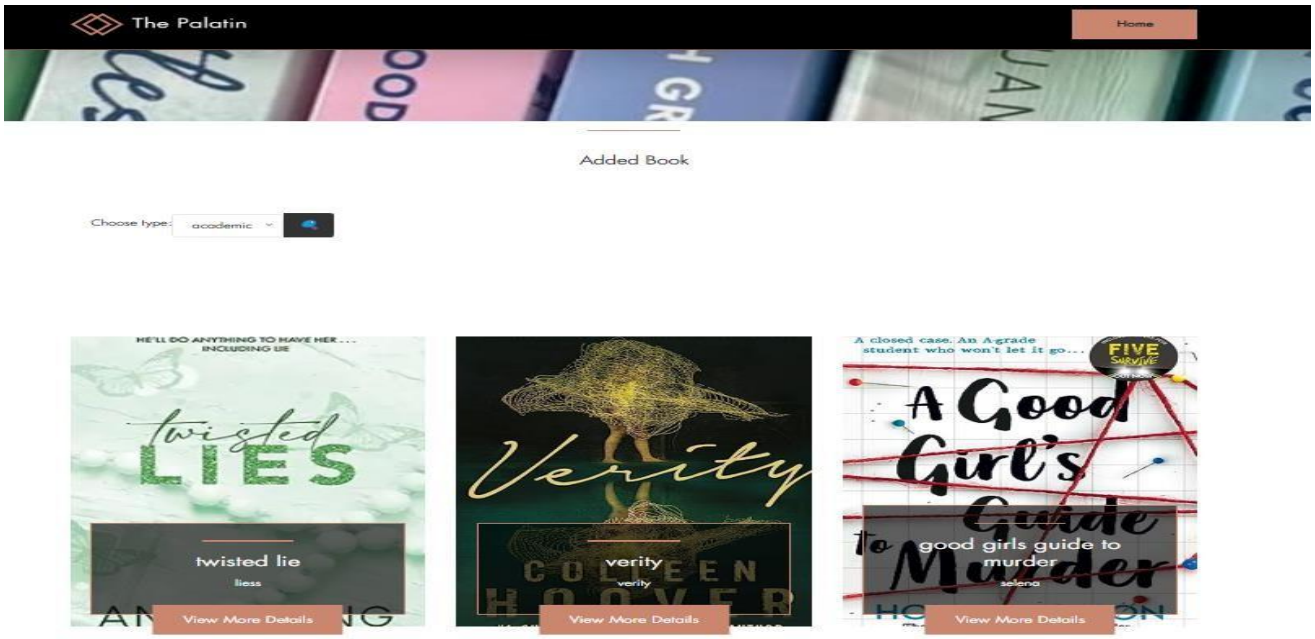
VIEW BOOK



ACADEMIC BOOK VIEW



NON ACADEMIC BOOK VIEW



VIEW SELLER

SI	SELLER NAME	EMAIL	CERTIFICATE	ACTIONS
1	snishitha sunilkumar	snizz	View Certificate	Approve Reject
2	sasuke uchiha	sasuke	View Certificate	Approve Reject
3	aalimol	oola	View Certificate	Approve Reject

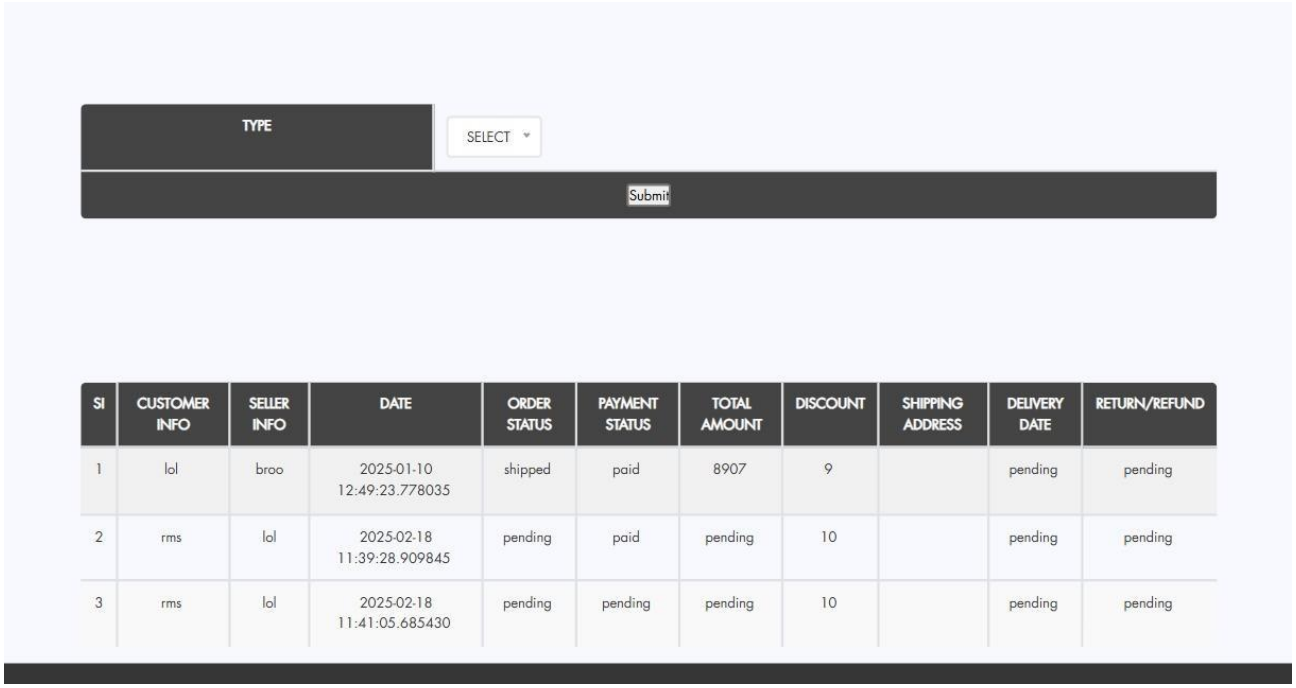
VIEW CUSTOMER

Sl	Name	Username	Phone Number	DOB	Gender	Address	Action
1	broo h	naruto	234567888	2006-03-10	Female	mumbai, kannur, 1235676	Block
2	lol o	lol	12345	2024-12-31	Female	ghjkl, admin, 929292	Block
3	shebu ceee	shebaaaa	12345	2004-04-01	Female	zdgxfhgvjbk, xrcgvjh, jb	Block
4	niranjana sindhu	niranjana	8714404396	2004-01-01	Female	Muscat, jiral, 728839	Block

BLOCKED CUSTOMER

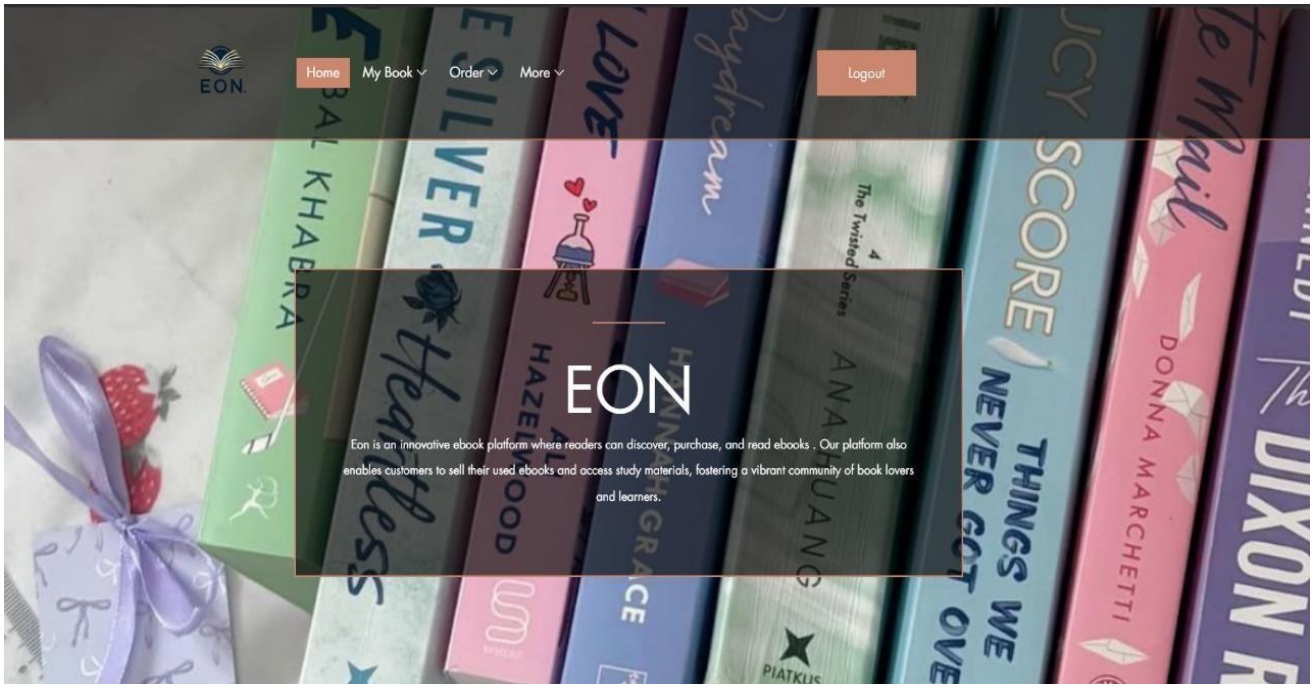
SI	FIRST NAME	LAST NAME	EMAIL	PHONE NUMBER	DOB	ADDRESS	ACTION
1	aaliya	hashique	aaliya	123455	2004-04-22	kakkad, post, 670643	Unblock
2	rms	dubai	rms	67266637892	2000-02-02	mattannur, mattannur, 670005	Unblock

VIEW ORDER HISTORY

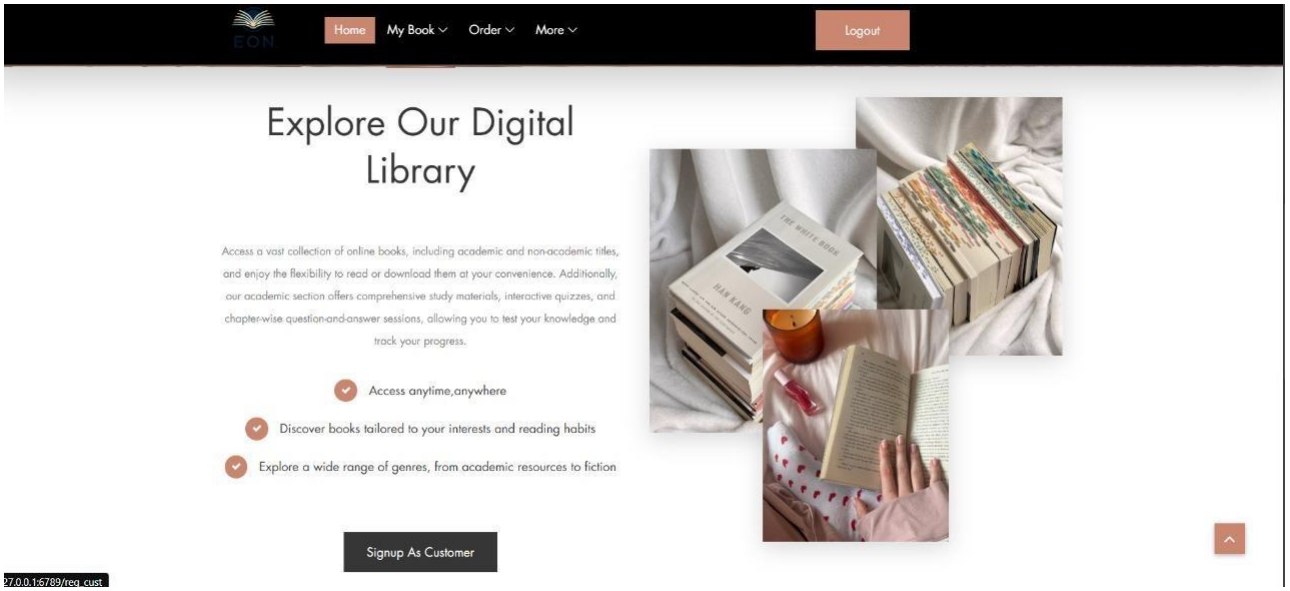


SI	CUSTOMER INFO	SELLER INFO	DATE	ORDER STATUS	PAYMENT STATUS	TOTAL AMOUNT	DISCOUNT	SHIPPING ADDRESS	DELIVERY DATE	RETURN/REFUND
1	lol	broo	2025-01-10 12:49:23.778035	shipped	paid	8907	9		pending	pending
2	rms	lol	2025-02-18 11:39:28.909845	pending	paid	pending	10		pending	pending
3	rms	lol	2025-02-18 11:41:05.685430	pending	pending	pending	10		pending	pending

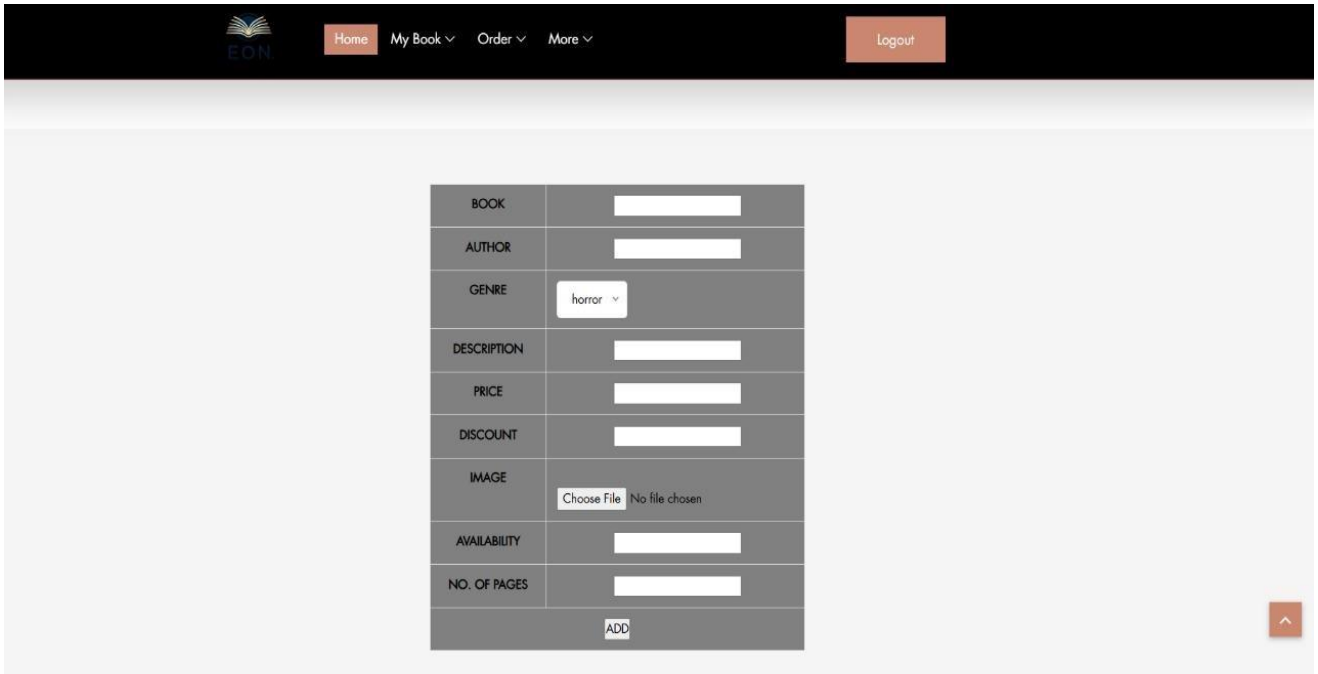
SELLER HOME PAGE



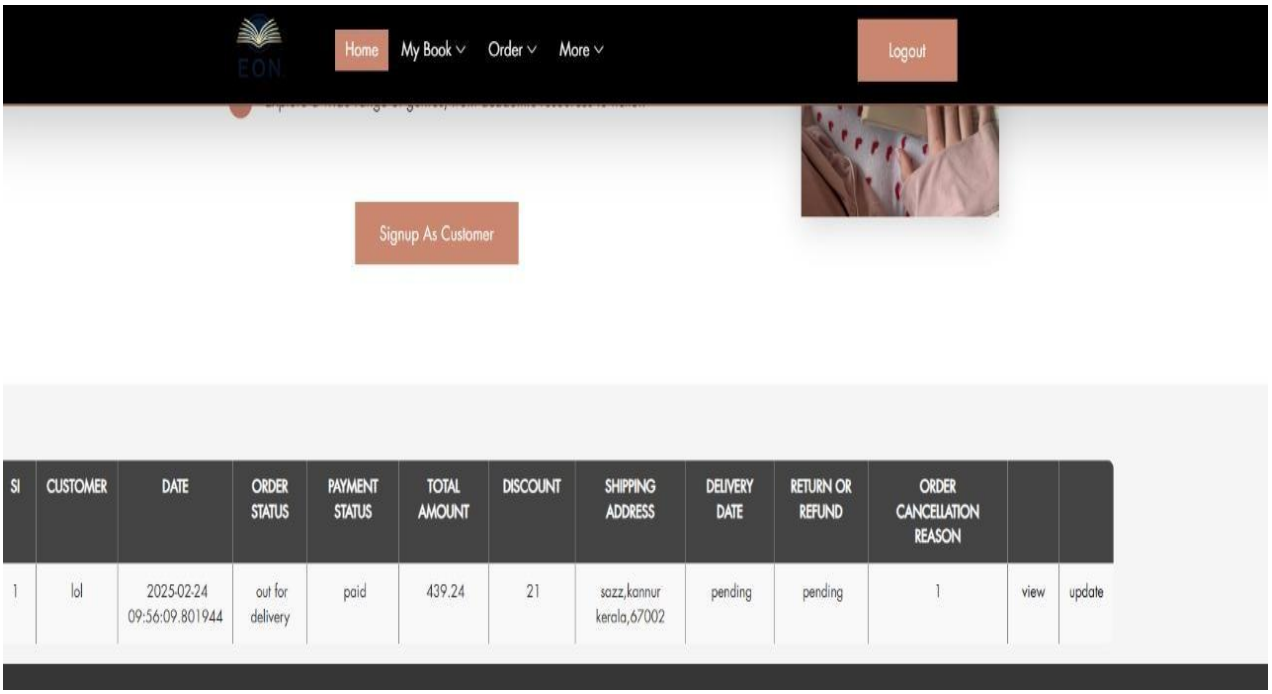
SIGNUP AS CUSTOMER



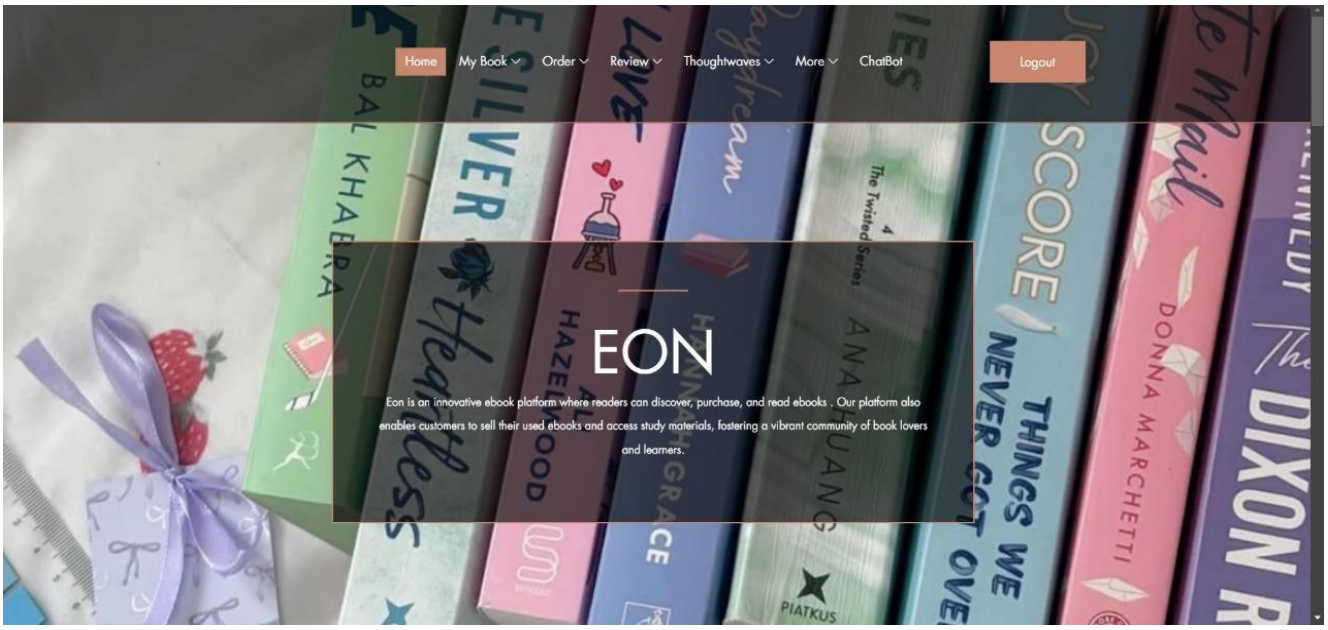
SELLER BOOK ADD

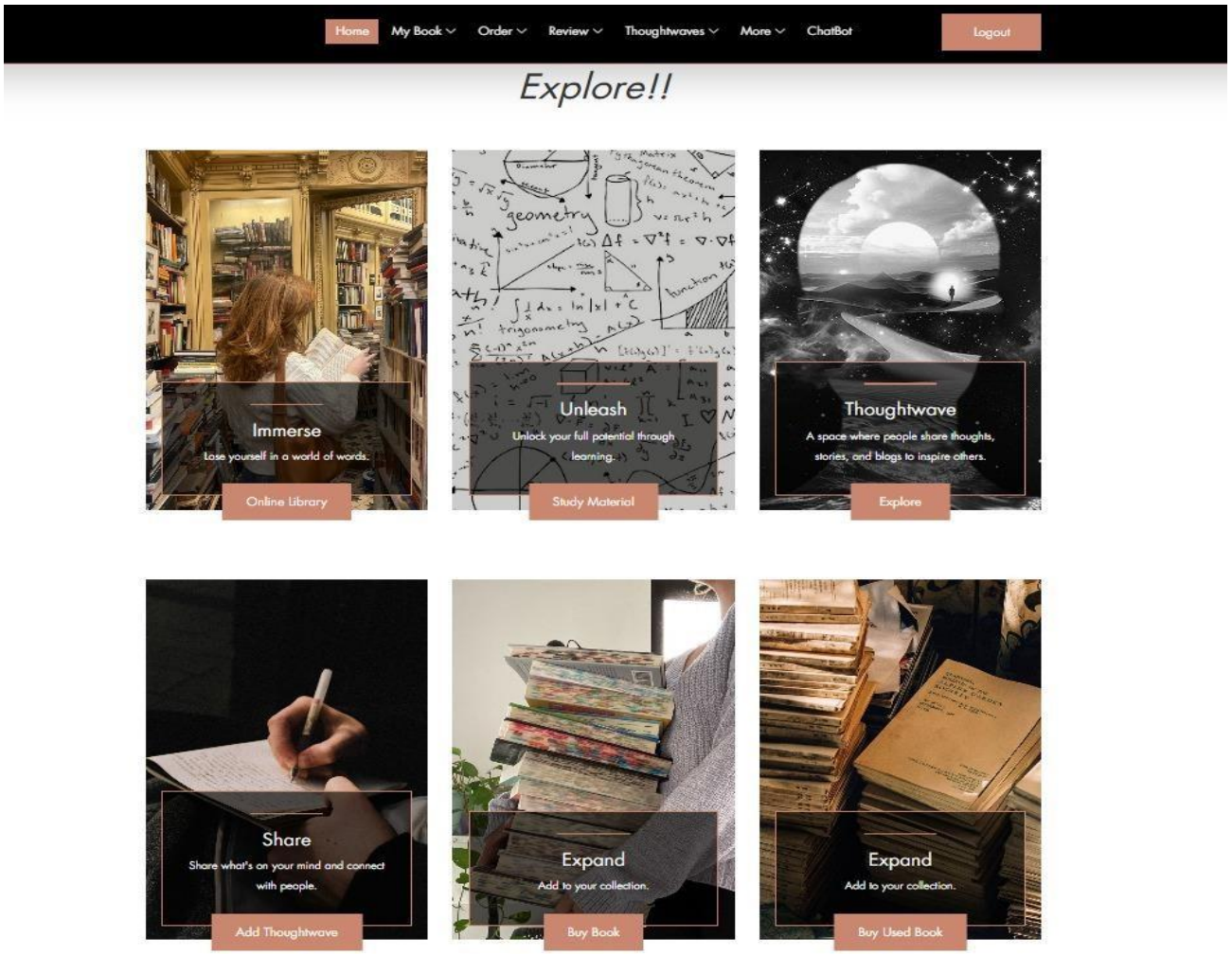


SELLER BOOK ORDER

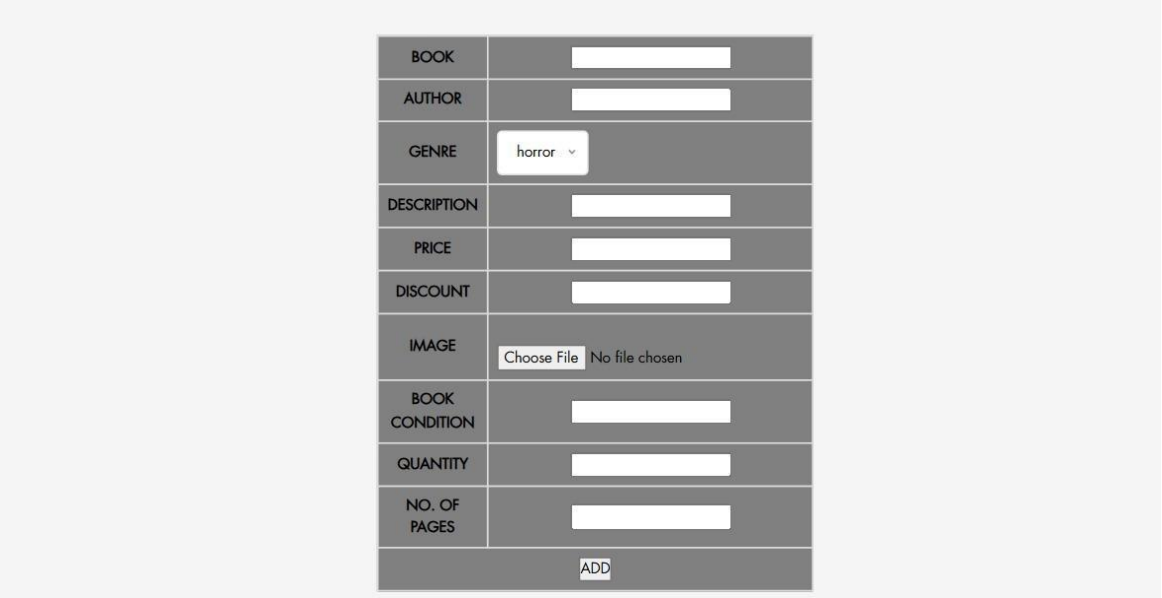


CUSTOMER HOME PAGE

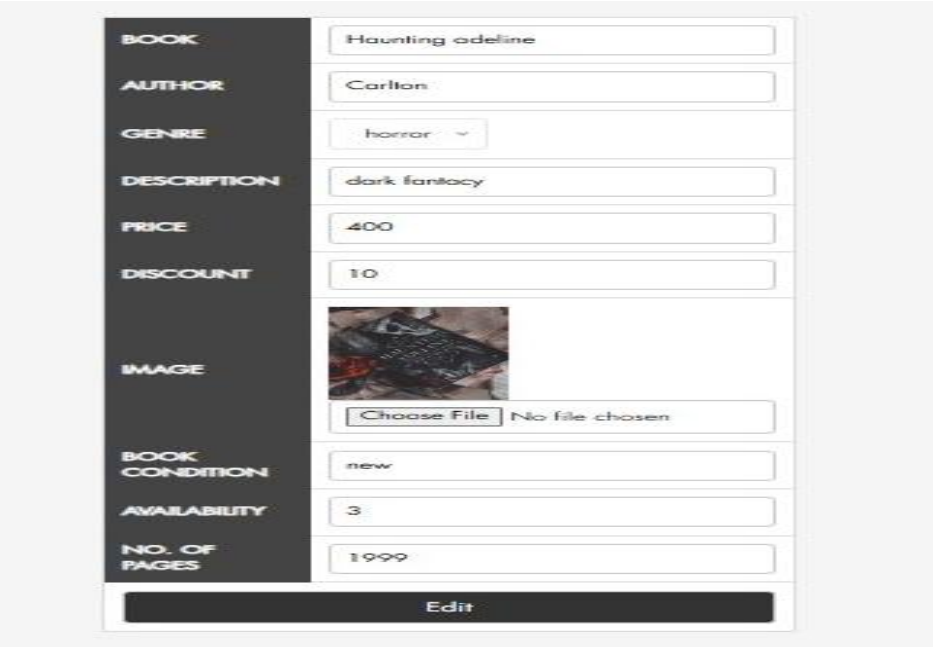



Home
My Book
Order
Review
Thoughtwaves
More
ChatBot
Logout
Explore!!
Immerse
Lose yourself in a world of words.
Online Library
Unleash
Unlock your full potential through learning.
Study Material
Thoughtwave
A space where people share thoughts, stories, and blogs to inspire others.
Explore
Share
Share what's on your mind and connect with people.
Add Thoughtwave
Expand
Add to your collection.
Buy Book
Expand
Add to your collection.
Buy Used Book

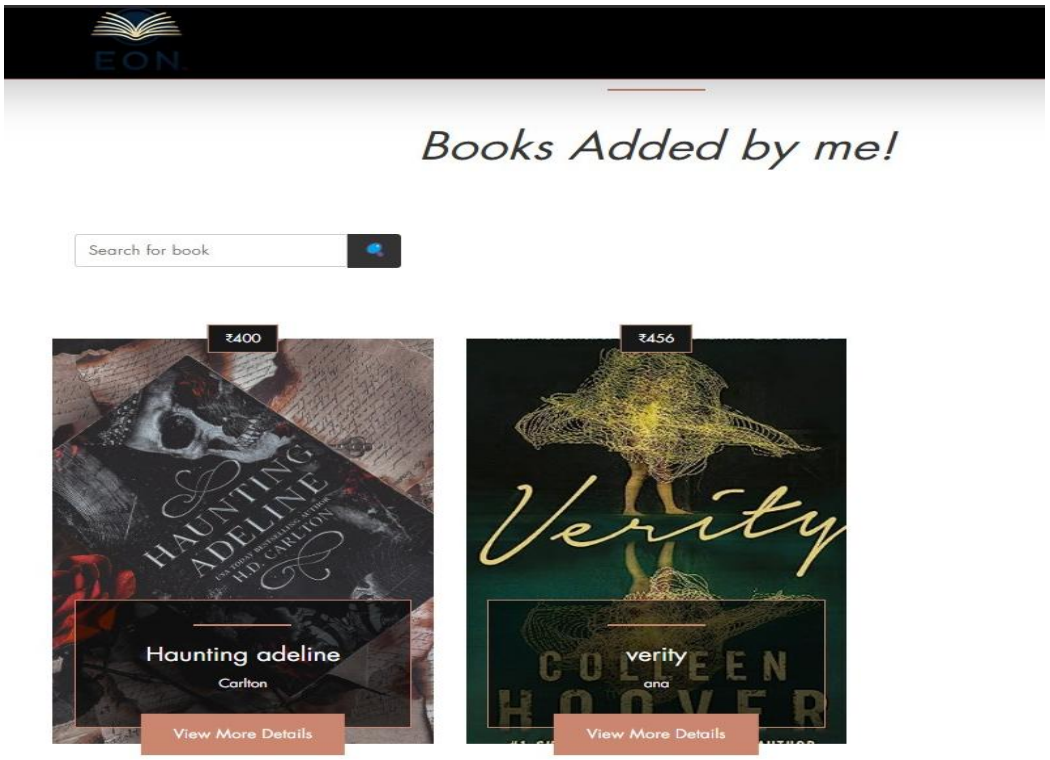
CUSTOMER ADD BOOK



EDIT BOOK



CUSTOMER VIEW BOOK



CUSTOMER ORDER MANAGEMENT

SI	DATE	ORDER STATUS	ADDRESS	ORDER NOTES	MANAGE ORDER	VIEW BOOK
1	2025-02-25 14:03:10.833353	pending	cfy,tuvg, xfc,vguvh, rycgv	pending	Approve Reject	view

CUSTOMER APPROVED ORDER

SI	CUSTOMER	DATE	ORDER STATUS	PAYMENT STATUS	PAYMENT METHOD	TOTAL AMOUNT	DISCOUNT	SHIPPING ADDRESS	DELIVERY DATE		
1	lol	2025-02-24 11:18:45.022623	dispatched	pending	pending	820.4099999999999	41	aseeg,et, aseg,aer, 34	2025-03-07	view	update
2	lol	2025-02-24 11:55:48.092889	delayed	pending	pending	3441.7200000000003	14	sss,ddd, sg,ggg, 35	2025-02-19	view	update
3	rms	2025-02-24 16:14:44.808597	dispatched	paid	online	405.84000000000003	11	xdg,hv, fyh,fyi, 345	2025-03-05	view	update

MY ORDER

SI	CUSTOMER	DATE	ORDER STATUS	PAYMENT STATUS	PAYMENT METHOD	TOTAL AMOUNT	DISCOUNT	SHIPPING ADDRESS	DELIVERY DATE		
1	lol	2025-02-24 11:18:45.022623	dispatched	pending	pending	820.4099999999999	41	aseeg,et, aseg,aer, 34	2025-03-07	view	update
2	lol	2025-02-24 11:55:48.092889	delayed	pending	pending	3441.7200000000003	14	sss,ddd, sg,ggg, 35	2025-02-19	view	update
3	rms	2025-02-24 16:14:44.808597	dispatched	paid	online	405.84000000000003	11	xdg,hv, fyh,fyi, 345	2025-03-05	view	update

VIEW CART

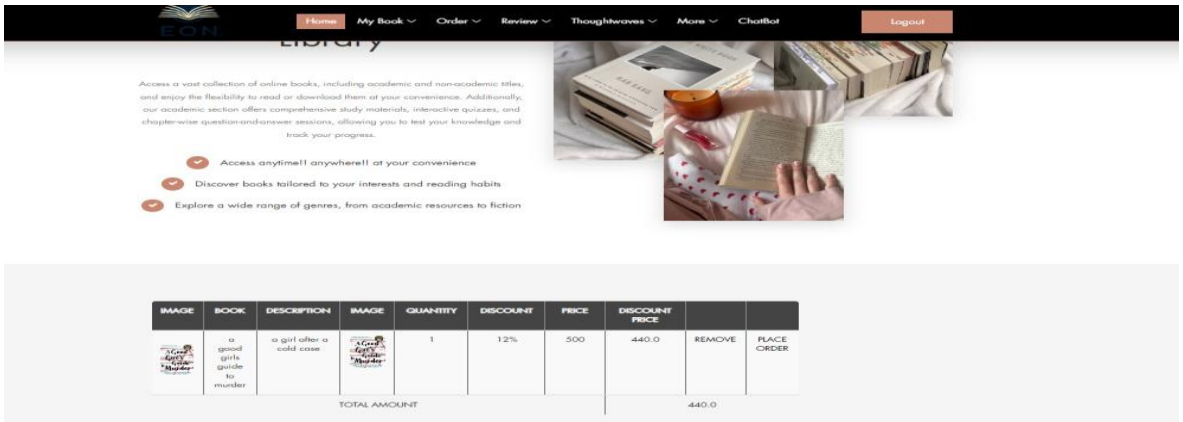
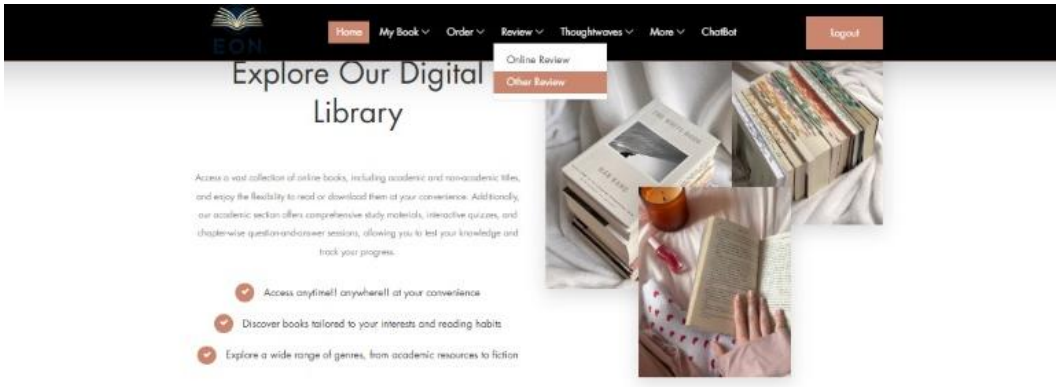


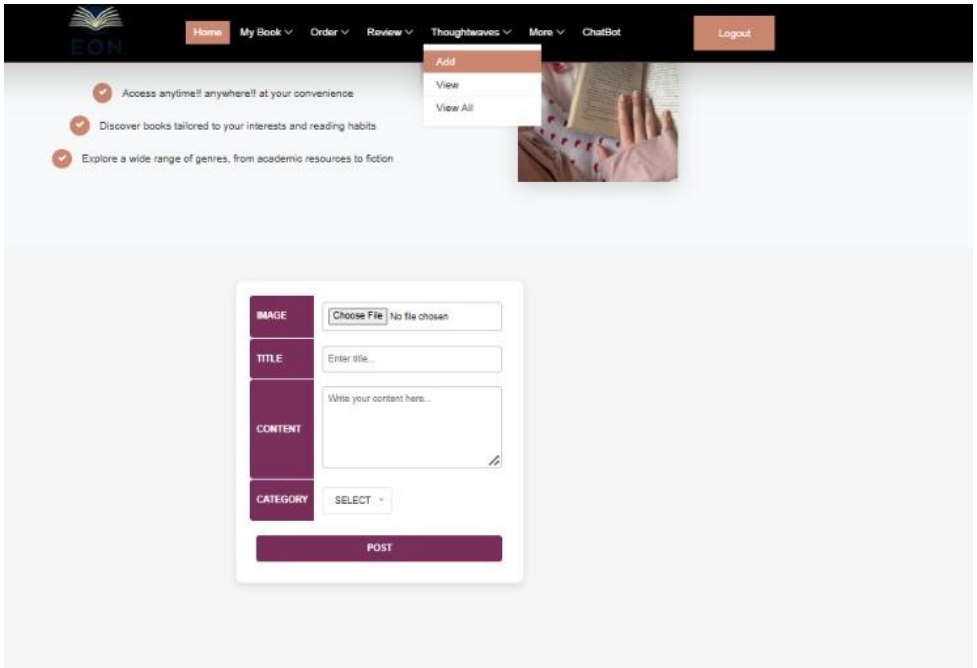
IMAGE	BOOK	DESCRIPTION	IMAGE	QUANTITY	DISCOUNT	PRICE	DISCOUNT PRICE		
	a good girls guide to murder	a girl after a cold case		1	12%	500	440.0	REMOVE	PLACE ORDER
			TOTAL AMOUNT				440.0		

ONLINE REVIEW

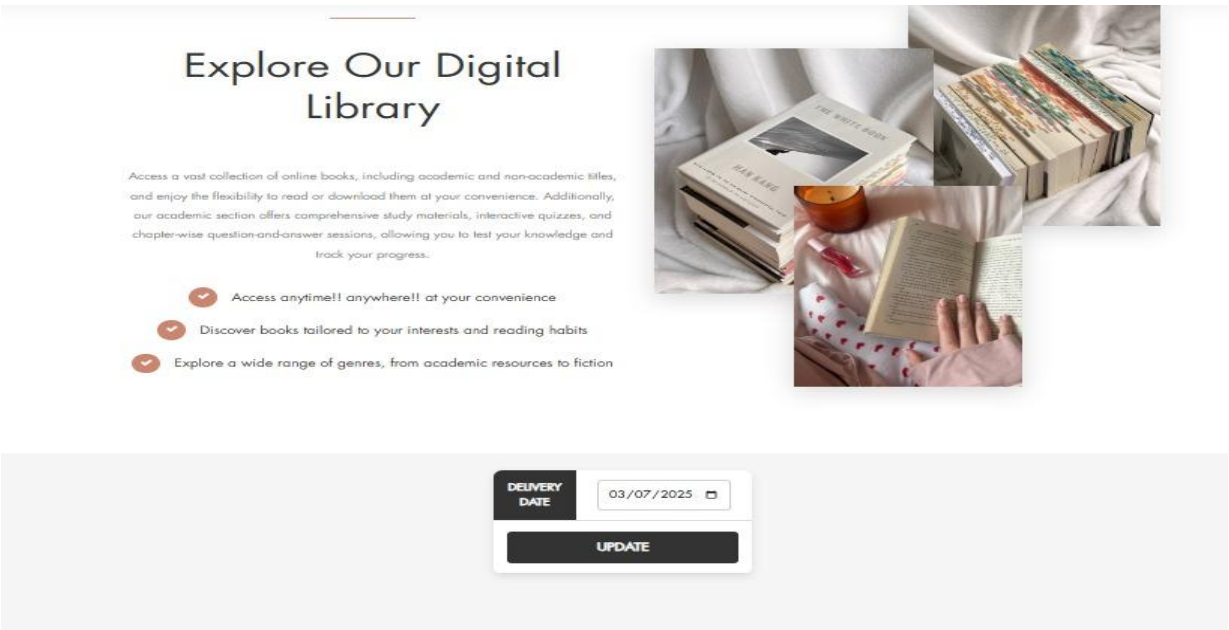


SI NO	BOOK_NAME	REVIEW_DATE	REVIEW	RATING		
1	That night	2025-02-25 14:45:02.977263	bad	★★	Edit	Delete
2	a good girls guide to murder	2025-02-25 14:45:13.651606	avg	★★	Edit	Delete
3	That night	2025-02-24 15:22:14.040648	not bad	★★★	Edit	Delete

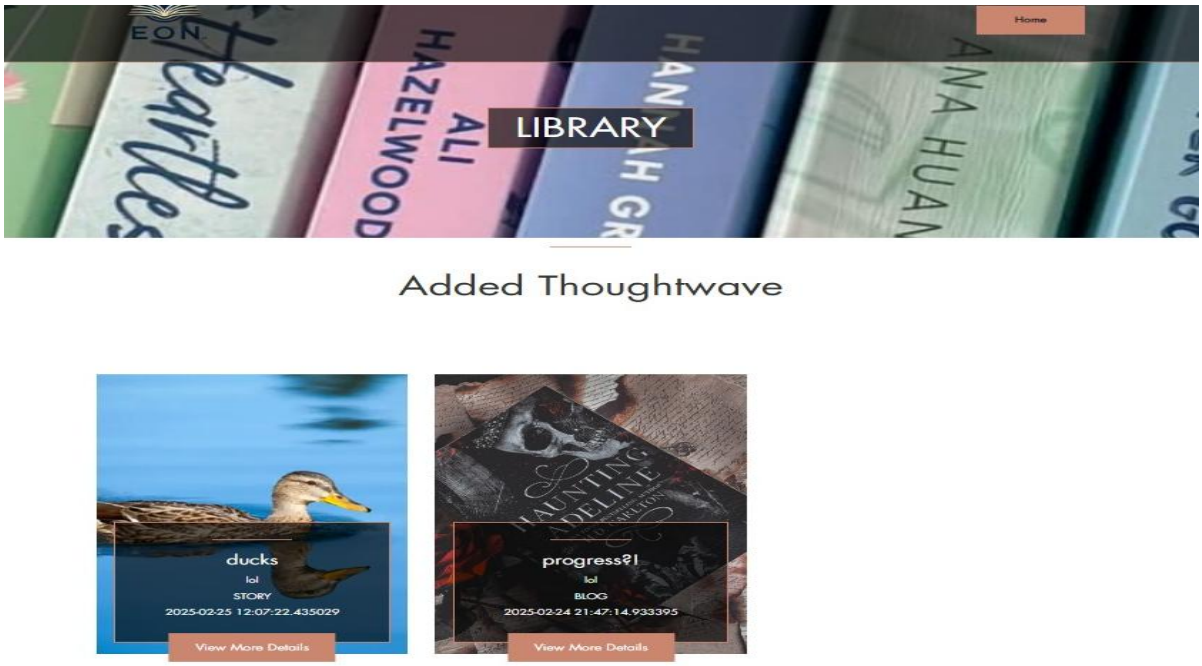
ADD THOUGHTWAVES



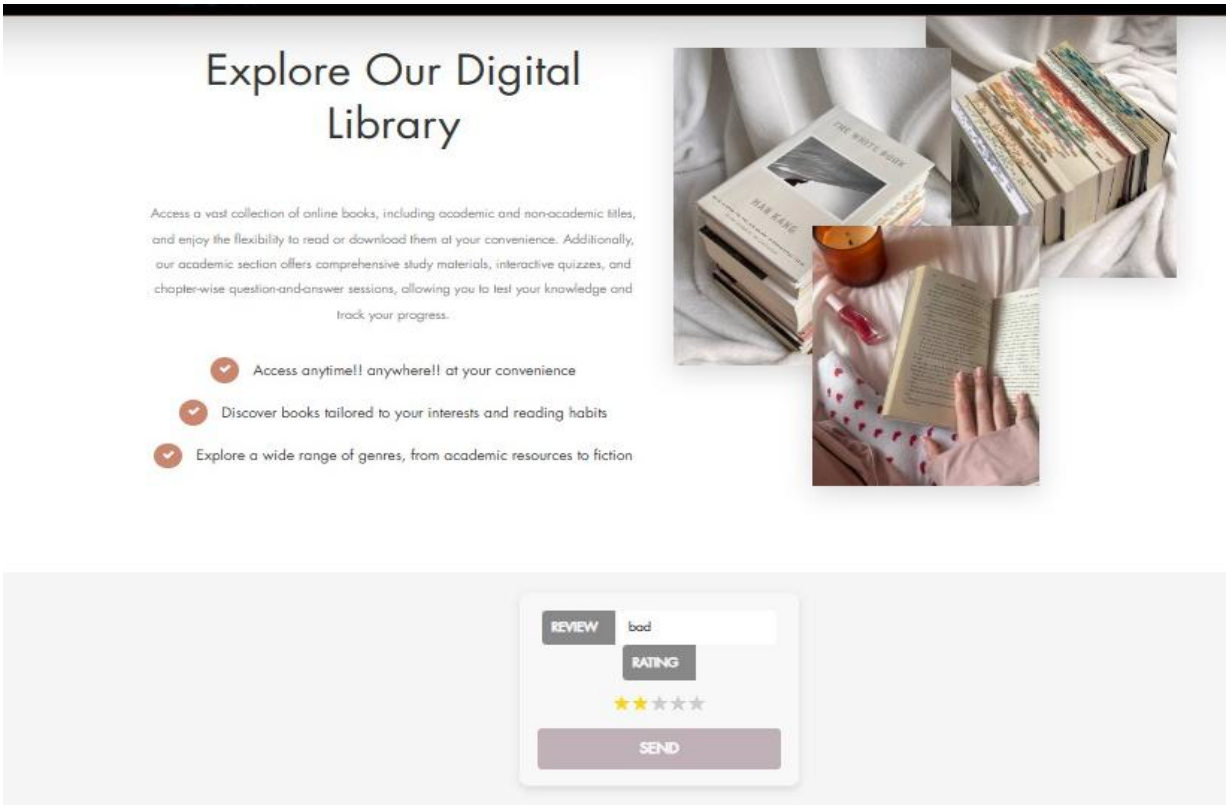
UPDATE DELIVERY



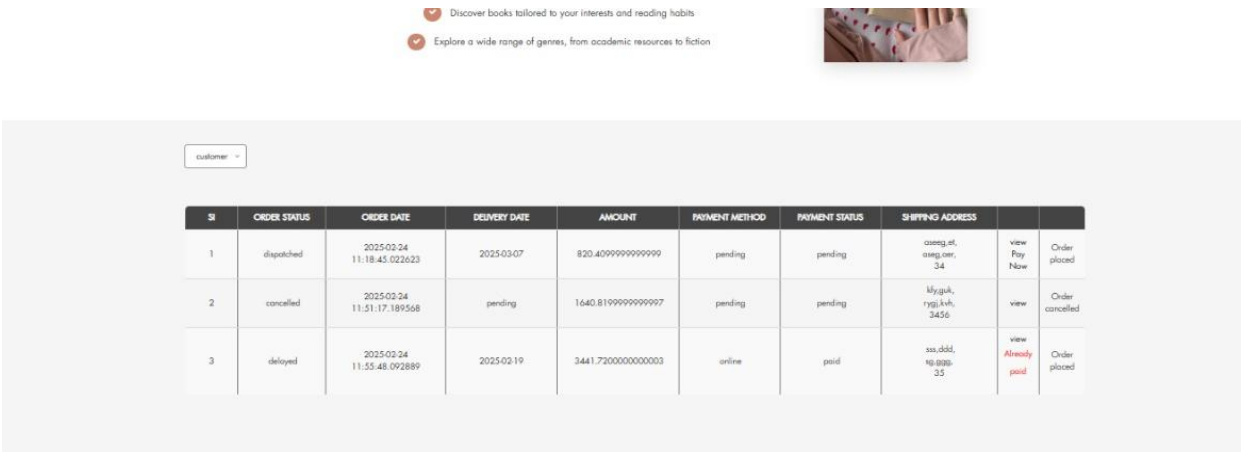
VIEW ADDED THOUGHTWAVES



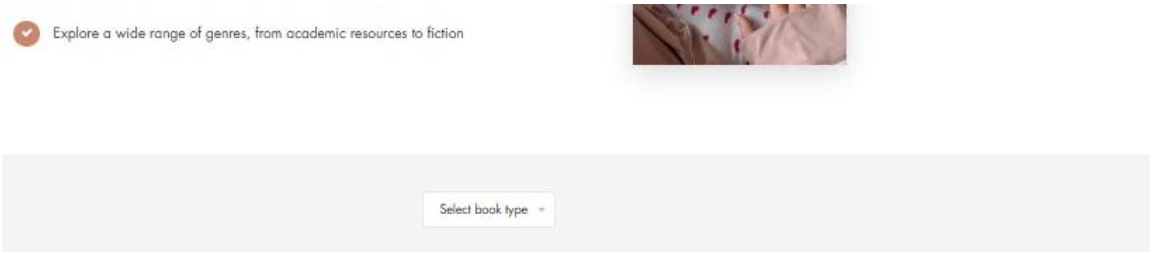
ADD REVIEW AND RATING



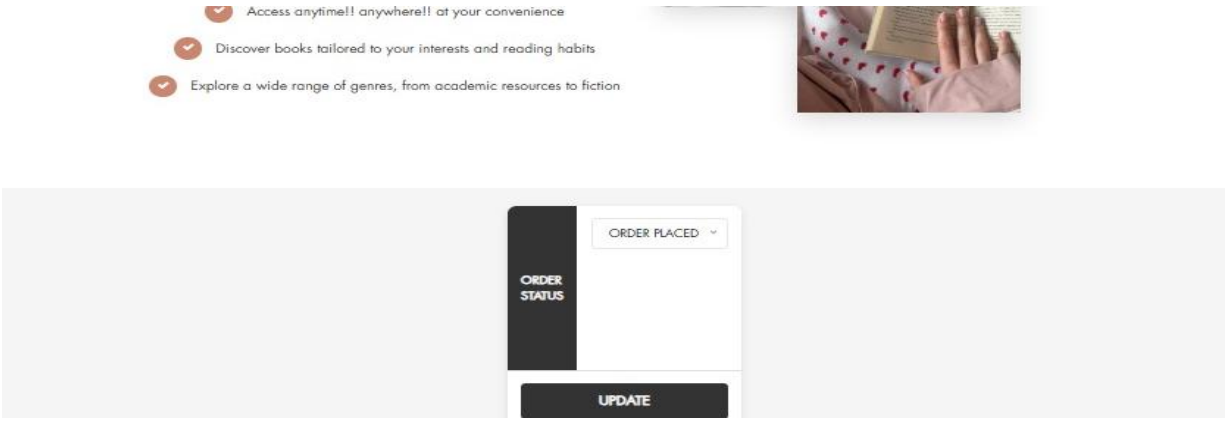
VIEW MY ORDER



SEARCH BOOK



ORDER STATUS



9. FUTURE SCOPE

In the future, we will focus on expanding eon's reach and enhancing its features. We will develop a mobile application, providing users with a more convenient and accessible way to purchase books. Additionally, we will introduce audiobooks, offering users an alternative format for consuming books. We will also enhance our study materials by incorporating interactive flashcards, enabling users to easily note and review important points. Furthermore, we will implement an emotion-based book recommendation system, providing users with personalized suggestions tailored to their reading preferences and emotions.

10. CONCLUSION

In conclusion, we have successfully completed this project, Eon. Eon now serves as a platform where customers can buy and sell books, and also share their thoughts through blogs. The platform enables sellers to sell their books, fostering a transparent and reliable environment for both customers and sellers. Looking ahead, our vision is for Eon to expand its reach and become a leading online book platform, offering services to a wider audience and empowering users to buy, sell, and share knowledge.

11. BIBILIOGRAPHY

- Software Engineering-KKAgarwal
- Python Tutorial | Learn Python Programming Language
- Django Tutorial | Learn Django Framework - GeeksforGeeks

12. GLOSSARY

UML : Unified Modeling

ERD : Entity Relationship Diagram

Language DFD: Data Flow Diagram