
**PROJECT REPORT
ON**

Online Book Store Management System

**Submitted to -
Prof. Tanusree Kaibartta**

**Authors -
Pratham Savane (24MT0311)
Rahul Kapardar (24MT0331)
Sandeep Shaw (24MT0388)
Saumya Agrawal (24MT0403)**

M.Tech 1st Year (2nd Semester)

Advanced Database Management System Lab

Table of Contents

1. Introduction	1
1.1 Introduction of our Project.....	1
1.2 Problem Statement.....	1
1.3 Aims and Objectives.....	2
2. Features.....	4
3. AI-Driven Recommendation System.....	4
3.1 Implementation Approach.....	4
3.2 Working of Apriori Algorithm.....	5
3.3 Association Rule Learning.....	5
3.4 Benefits of AI Based Recommendation.....	5
4. Software and Hardware Requirements.....	6
4.1 Software Requirements.....	6
4.2 Hardware Requirements.....	6
5. Data Flow Diagram (DFD).....	7
5.1 DFD Levels.....	7
5.2 DFD Diagram.....	7
6. Entity-Relationship Diagram (ERD).....	8
7. Class Diagram.....	9
8. Code Implementation and Results.....	10
8.1 Code Implementation.....	10
8.1.1 models.py.....	12
8.1.2 urls.py.....	12
8.1.3 requirements.txt.....	12
8.2 Results.....	13
9. Conclusion	20
10.Future Enhancements.....	20

Introduction

1.1 Introduction of our project

Introduction

In the digital era, online bookstores have transformed the traditional book-buying experience. This project aims to design a comprehensive database system for an online book store, inspired by platforms like Amazon. The system supports book browsing, searching, purchasing, and personalized recommendations, enhancing the overall user experience. It ensures efficient stock management, order handling, and automated publisher requisitions. Moreover, it integrates user profiles and e-purses for seamless transactions, backed by role-based authorizations to secure customer data and administrative functionalities.

The proposed database system caters to two primary user groups: customers and bookstore administrators. Customers can browse books by categories, perform keyword searches, create accounts, manage profiles, and purchase books using their e-purses. Administrators can monitor sales, manage stock levels, and place requisitions to publishers. The system automates requisitioning books based on stock levels and order data, ensuring smooth operations and timely stock replenishment.

Additionally, the system incorporates recommendation algorithms — personalized suggestions based on user profiles and collaborative filtering methods like 'customers who bought this also bought.' Automated email notifications are triggered for order arrivals and new arrivals matching customer interests, improving user engagement.

The following sections detail the design of database tables, user views, forms, and authorizations, ensuring a robust and user-friendly online book store experience.

1.2 Problems Statement

Design a Database System for an online book store (e.g., Amazon), based on the following specifications. Books are represented by, ISBN, Title, Author, Publisher, Edition, Year of Publication, Price, Short Reviews if available, Table of Contents if available., an image of the book cover, category e.g., computer science-> operating

systems-> MacOS. Customers will use an web based interface to browse books

based on categories, search books using keywords. Initially only the title and author of the book(s) are displayed, on click other attributes are displayed. Customers can buy books using their e-purse. The store also displays the number of copies of the book left in stock. Out of stock books cannot be purchased immediately, but can be ordered. Customers create accounts in the book store. Each account contains customer profile information: name, age, geographical location, categories of interest, email. Each account has an e-purse. Customers can specify the amount of money to be deposited with the e-purse. Profile and e-purse information can be updated by the customer. Customers will log into the book store using an account name and password. All online sales data are recorded in the database with timestamp. Owner of the bookstore can give requisition for buying books to publishers based on the amount of stock remaining. For each book the owner maintains a stock which is at least the number of copies of the book sold over the last 3 months. Books ordered by some customers are immediately requisitioned. Requisitions are placed in a requisition table. The publishers inspect the table on the 1st of every month and immediately supply the books. Once a book is supplied it is cleared from the requisition table. Design tables for the above system. Create a separate view for customers. Design suitable forms. Implement authorizations. Also Display to the customers a list of books which might be of interest. This may be done based on user profile, or in a collaborative manner- "people who have bought this book have also bought". Send emails to customers when the ordered book has arrived, new books of interest have arrived in the store.

1.3 Aims and Objectives

Aims:

- To design and implement a scalable, user-friendly, and efficient online bookstore database system.
- To streamline book browsing, searching, and purchasing processes for customers.
- To enhance customer engagement through personalized recommendations and email notifications.
- To automate stock management and requisition processes for improved operational efficiency.

Objectives:

1. Develop a relational database system to store and manage book data, user profiles, sales records, and stock levels.
2. Implement a secure user authentication system to manage customer accounts and administrative access.
3. Create an intuitive web-based interface for book browsing, searching, and purchasing.
4. Incorporate an e-purse system for easy and secure online transactions.
5. Design an automated requisition system to manage out-of-stock books and streamline restocking from publishers.
6. Provide personalized recommendations based on user profiles and collaborative filtering methods.
7. Record sales data with timestamps for performance analysis and business insights.
8. Ensure role-based authorization to secure customer data and limit administrative functions.
9. Enable email notifications for order updates and new arrivals tailored to user interests.

2. Features of the Project

1. **User Account Management:** Customers can create accounts, log in securely, and manage their profiles, including name, age, location, and interests.
2. **E-Purse Integration:** Customers can deposit money into their e-purses and use the balance for purchases.
3. **Book Browsing and Searching:** Books can be browsed by categories or searched using keywords, initially displaying only the title and author.
4. **Detailed Book View:** Clicking on a book reveals additional details like price, edition, publication year, reviews, table of contents, and stock availability.
5. **Stock Management:** Tracks the number of copies available for each book. Out-of-stock books can still be ordered.
6. **Order and Requisition System:** Orders for out-of-stock books are added to a requisition table. The bookstore owner places orders to publishers, who fulfill requests monthly.
7. **Sales Recording:** Each purchase is recorded with a timestamp for future analysis.
8. **Personalized Recommendations:** Recommends books based on user profile interests and collaborative filtering ('people who bought this also bought')
9. **Email Notifications:** Sends automated emails when orders arrive or new books matching customer interests are added.
10. **Role-Based Authorization:** Secures data by granting different permissions to customers and administrators.
11. **Admin Dashboard:** Allows bookstore owners to monitor sales, view requisitions, and manage stock levels efficiently.

This feature-rich system aims to provide a seamless, engaging, and efficient online book shopping experience for customers while ensuring smooth backend operations for bookstore management.

3. AI-Driven Recommendation System

3.1 Implementation Approach

The AI-based recommendation system in BookHaven is built using the **Apriori algorithm** from the mlxtend library. This algorithm identifies frequently bought-together books by analyzing user transactions and order patterns.

3.2 Working of Apriori Algorithm

- Identifies frequent itemsets in user purchase history.
- Uses **support, confidence, and lift** metrics to derive association rules.
- Generates book recommendations based on user similarities.

3.3 Association Rule Learning

- Establishes relationships between different book purchases.
- If a user orders a book, the system suggests books frequently purchased by similar users.
- Personalized book recommendations based on order history, browsing behavior, and requisition requests.

3.4 Benefits of AI-Based Recommendations

- **Personalized Experience:** Users receive suggestions tailored to their interests.
- **Increased Sales & Engagement:** Users are more likely to purchase recommended books.
- **Efficient Inventory Management:** Identifies books that require restocking.
- **Enhanced Customer Retention:** Builds loyalty by offering relevant recommendations.

4. Software and Hardware Requirements

4.1 Software Requirements

- **Operating System:** Windows, Linux, macOS
- **Development Tool:** VS Code
- **Programming Languages:** Python, JavaScript, HTML, CSS
- **Framework:** Django (Full Stack API)
- **Database:** SQLite3
- **Libraries Used:** Django ORM, Django REST Framework, mlxtend (Apriori Algorithm)

4.2 Hardware Requirements

- **RAM:** Minimum 4GB (Recommended 8GB)
- **Storage:** Minimum 10GB (SSD recommended)
- **Processor:** Intel i5 or equivalent

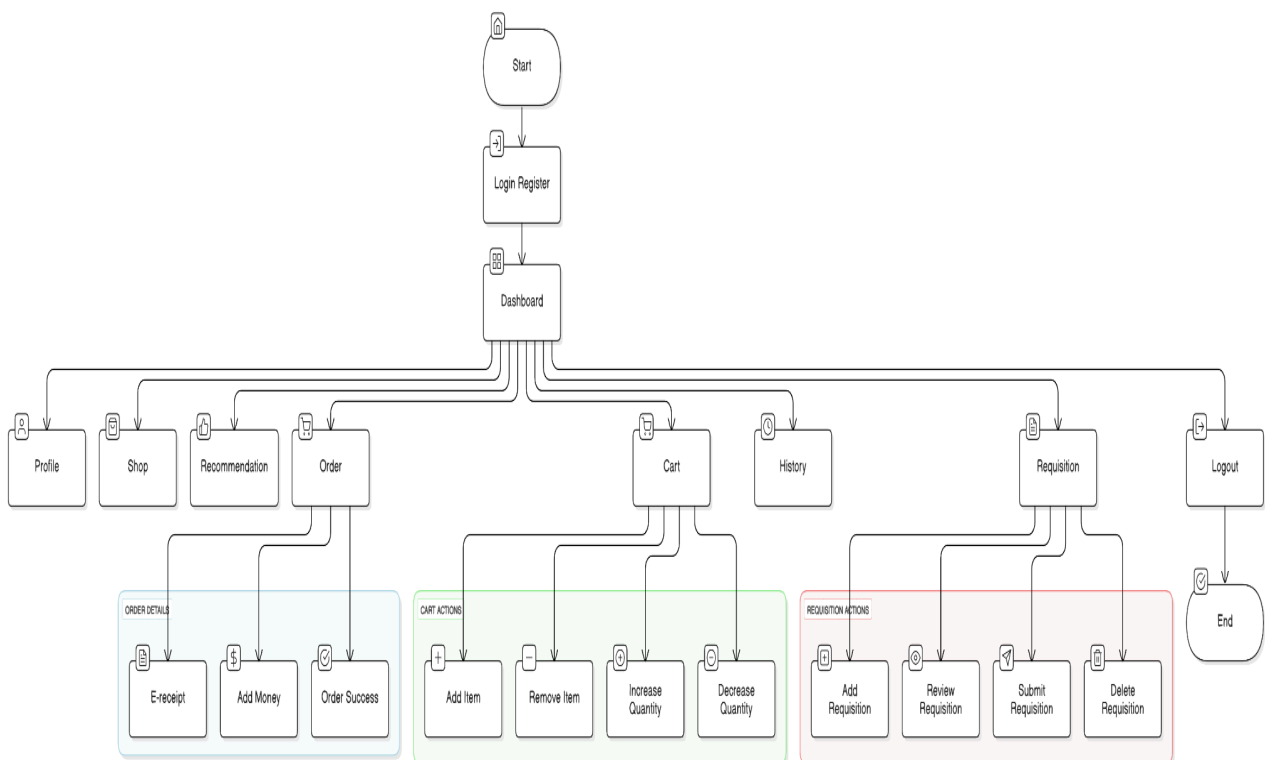
5. Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) provides a visual representation of data movement within the system.

5.1 DFD Levels

- **Level 0:** Represents the system as a single entity interacting with users.
- **Level 1:** Breaks down major processes like User Management, Order Processing, and AI Recommendations.
- **Level 2:** Provides a detailed view of sub-modules such as Authentication, Payment Processing, and Review Management.

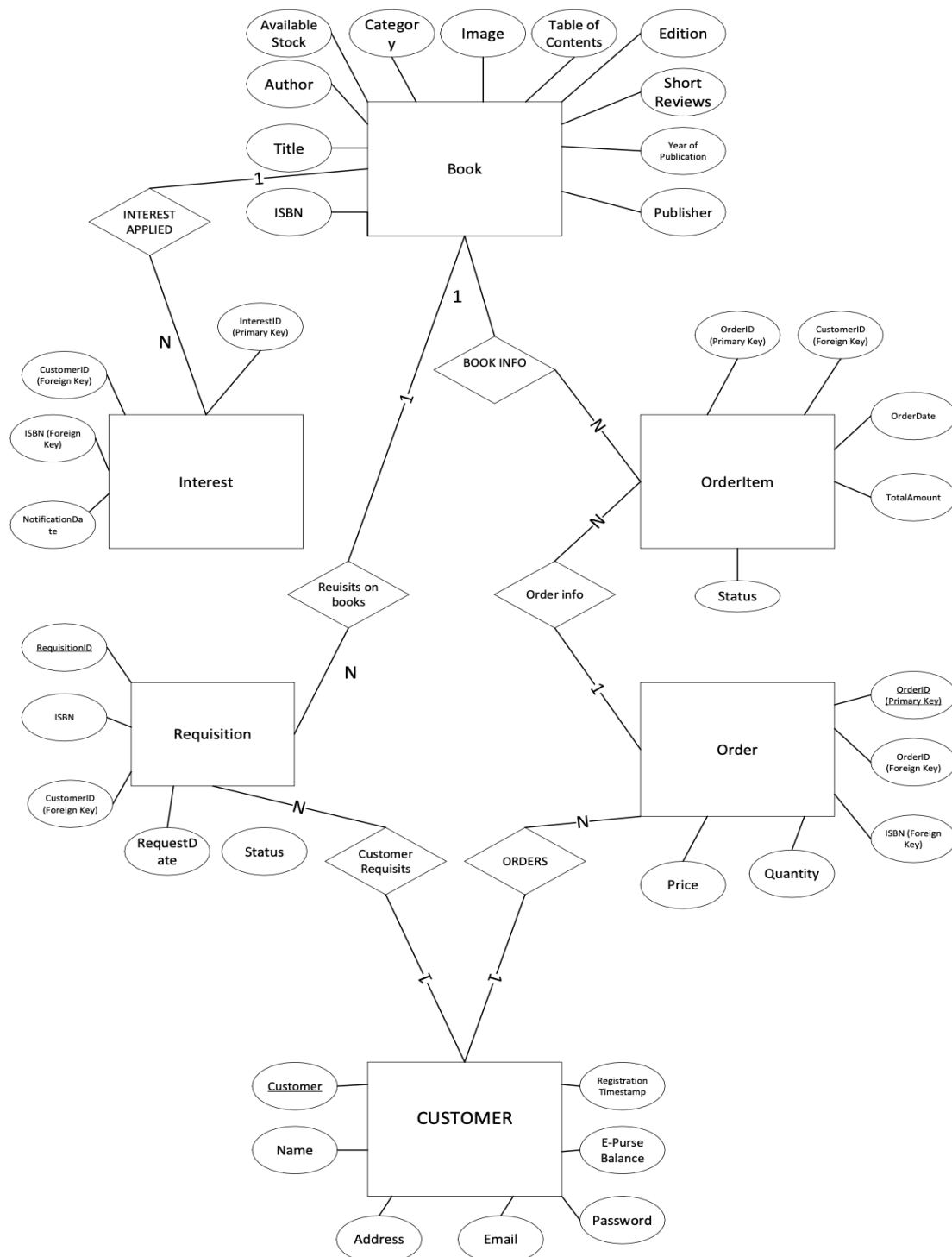
5.2 DFD Diagram



6. Entity-Relationship Diagram (ERD)

The Entity-Relationship Diagram (ERD) illustrates the relationships between key entities:

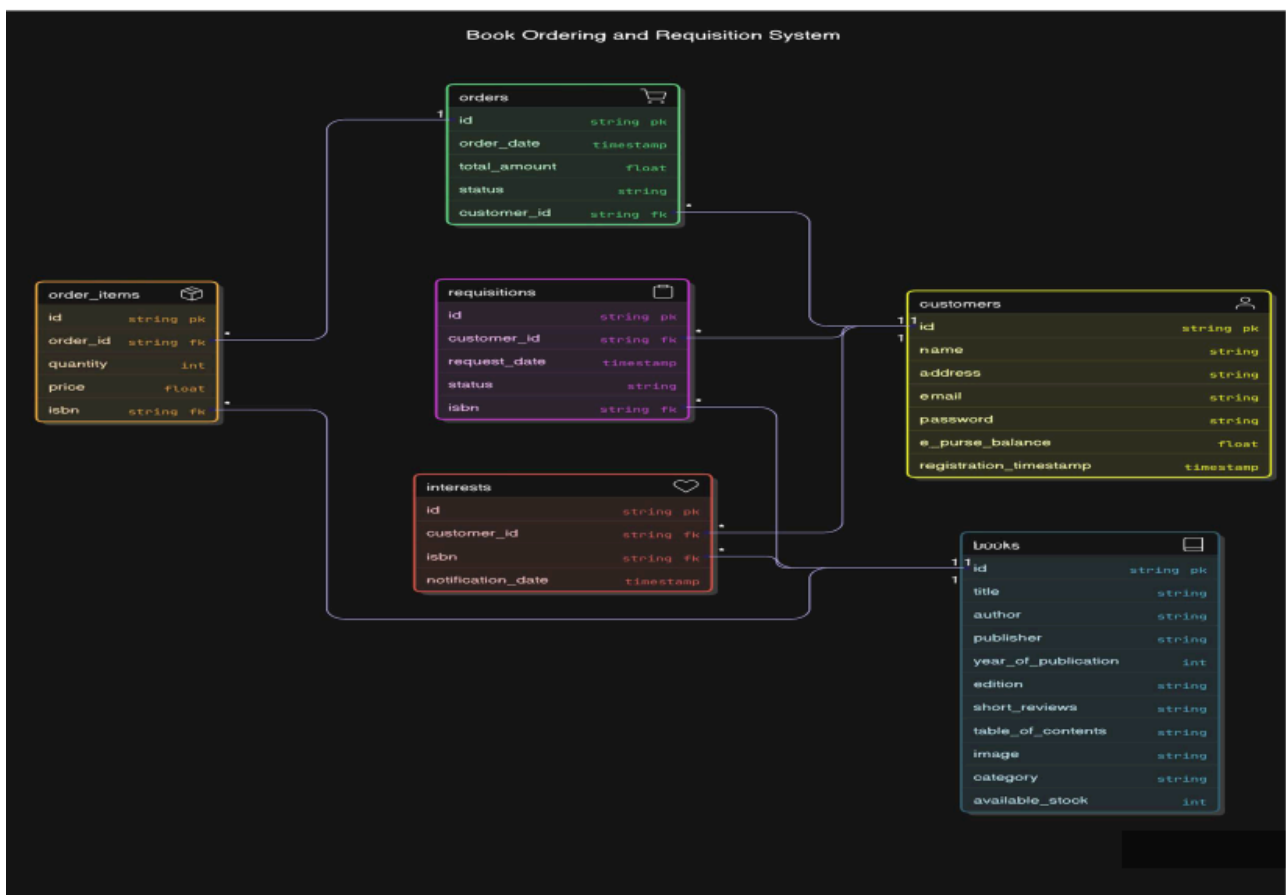
- **Users:** Can browse books, place orders, and manage profiles.
- **Books:** Contains book details such as title, author, category, and price.
- **Orders:** Stores order details linked to users and book purchases.
- **Reviews:** Captures book reviews and ratings from users.
- **Transactions:** Manages payment records and wallet transactions.



7. Class Diagram

The Class Diagram presents the object-oriented structure of the system, including relationships between different classes such as:

- **User Class:** Handles authentication, profile management, and orders.
- **Book Class:** Stores book information, categories, and stock levels.
- **Order Class:** Manages user purchases and transactions.
- **Recommendation Class:** Implements the AI-driven suggestion system.



8. Code Implementation and Results

8.1 Code Implementation

8.1.1 models.py

database models for Users, Books, Orders, and Recommendations.

```
class Category(models.Model):  
    name = models.CharField(max_length=255, unique=True)  
    slug = models.SlugField(max_length=255, unique=True, blank=True)
```

```
✓ class Writer(models.Model):  
    name = models.CharField(max_length=255)  
    image = models.ImageField(upload_to='writers/', blank=True, null=True)  
    about = models.TextField()
```

```
class Book(models.Model):  
    name = models.CharField(max_length=255)  
    authors = models.ManyToManyField(Writer)  
    summary = models.TextField()  
    categories = models.ManyToManyField(Category)  
    cover_image = models.ImageField(upload_to='books/', blank=True, null=True)  
    price = models.DecimalField(max_digits=10, decimal_places=2)  
    stock = models.PositiveIntegerField()  
  
    # New fields  
    isbn = models.CharField(max_length=13, unique=True, blank=True, null=True) # ISBN (optional)  
    edition = models.CharField(max_length=50, blank=True, null=True) # Edition (e.g., "2nd Edition")  
    publisher = models.CharField(max_length=255, blank=True, null=True) # Publisher Name  
    publication_year = models.PositiveIntegerField(blank=True, null=True) # Year of Publication  
  
    # Featured field  
    featured = models.BooleanField(default=False, null=True, blank=True) # Allows null & blank values
```

```
class Customer(models.Model):  
    user = models.OneToOneField(User, on_delete=models.CASCADE) # Link to Django User model  
    e_purse_balance = models.DecimalField(max_digits=10, decimal_places=2, default=0.00) # Default balance 0  
    address = models.TextField(null=True, blank=True) # Optional Address  
    preferred_categories = models.ManyToManyField(Category, blank=True) # Preferred Categories  
    preferred_writers = models.ManyToManyField(Writer, blank=True) # Preferred Writers
```

```
class Review(models.Model):  
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)  
    book = models.ForeignKey(Book, on_delete=models.CASCADE)  
    timestamp = models.DateTimeField(auto_now_add=True)  
    edited = models.BooleanField(default=False) # ✅ New field to track edits  
    rating = models.DecimalField(max_digits=3, decimal_places=1) # Rating out of 5 (1.0 to 5.0)  
    description = models.TextField(null=True, blank=True) # Optional review text
```

```

class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    total_price = models.DecimalField(max_digits=10, decimal_places=2)
    address = models.TextField() # Stores full shipping address
    payment_mode = models.CharField(
        max_length=50,
        choices=[('E-Purse', 'E-Purse'), ('COD', 'Cash on Delivery')]
    )
    status = models.CharField(
        max_length=20,
        choices=[('Pending', 'Pending'), ('Paid', 'Paid'), ('Failed', 'Failed')],
        default='Pending'
    )
    timestamp = models.DateTimeField(auto_now_add=True)

```

```

class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE, related_name='items')
    book = models.ForeignKey(Book, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    price = models.DecimalField(max_digits=10, decimal_places=2) # Price at time of order

```

```

class Requisition(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    book = models.ForeignKey(Book, on_delete=models.CASCADE, null=True, blank=True) # Link to Book
    book_name = models.CharField(max_length=255) # Store book name separately in case the book is deleted
    author_names = models.TextField(null=True, blank=True) # Store multiple author names
    requested_at = models.DateTimeField(auto_now_add=True)

```

```

✓ class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE) # Each cart belongs to a user
    book = models.ForeignKey(Book, on_delete=models.CASCADE) # The book being added
    quantity = models.PositiveIntegerField(default=1) # Quantity of the book

```

8.1.2 urls.py

Contains URL patterns and API endpoints for different functionalities.

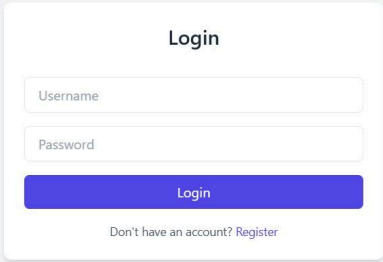
```
urlpatterns = [
    path('', home, name='home'),
    path('admin/', admin.site.urls),
    path('register/', register_view, name='register'),
    path('login/', login_view, name='login'),
    path('logout/', logout_view, name='logout'),
    path('dashboard/', dashboard_view, name='dashboard'),
    path('dashboard/profile', dashboard_profile, name='dashboard_profile'),
    path('dashboard/epurse', dashboard_epurse, name='dashboard_epurse'),
    path('dashboard/shop', dashboard_shop, name='dashboard_shop'),
    path("dashboard/shop/book/<int:book_id>/", book_detail, name="book_detail"),
    path("dashboard/profile/", profile, name="profile"),
    path("add-money/", add_money, name="add_money"),
    path('dashboard/cart/', cart_view, name='cart_view'),
    path('dashboard/cart/add/<int:book_id>/', add_to_cart, name='add_to_cart'),
    path('dashboard/cart/remove/<int:cart_id>/', remove_from_cart, name='remove_from_cart'),
    path('dashboard/cart/increase/<int:cart_id>/', increase_quantity, name='increase_quantity'),
    path('dashboard/cart/decrease/<int:cart_id>/', decrease_quantity, name='decrease_quantity'),
    path('checkout/<int:order_id>/', e_purse_checkout, name='e_purse_checkout'),
    path('order-success/', order_success, name='order_success'),
    path('requisition/add/<int:book_id>/', add_to_requisition, name='add_to_requisition'),
    path('requisition/list/', requisition_list, name='requisition_list'),
    path('review/submit/<int:book_id>/', submit_review, name='submit_review'),
    path('review/delete/<int:review_id>/', delete_review, name='delete_review'),
    path('order/history/', order_history, name='order_history'),
    path('recommendations/', recommended_books, name='recommendations'),
    path('analytics/', analytics, name='analytics'),
]
```

8.1.3 Requirements.txt

Code Blame 25 lines (25 loc) · 433 Bytes

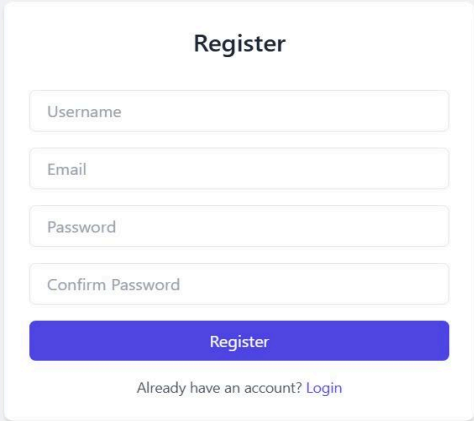
```
1  asgiref==3.8.1
2  contourpy==1.3.1
3  cycler==0.12.1
4  Django==5.1.6
5  djangoRESTframework==3.15.2
6  fonttools==4.56.0
7  joblib==1.4.2
8  kiwisolver==1.4.8
9  matplotlib==3.10.1
10 mlxtend==0.23.4
11 numpy==2.2.4
12 packaging==24.2
13 pandas==2.2.3
14 pillow==11.1.0
15 pyparsing==3.2.2
16 python-dateutil==2.9.0.post0
17 pytz==2025.1
18 scikit-learn==1.6.1
19 scipy==1.15.2
20 six==1.17.0
21 sqlparse==0.5.3
22 threadpoolctl==3.6.0
23 typing_extensions==4.12.2
24 tzdata==2025.1
25 whitenoise==6.9.0
```

8.2 RESULTS:



A login form titled "Login" centered on a light gray background. The form contains two input fields: "Username" and "Password". Below these fields is a blue button labeled "Login". At the bottom of the form, there is a link that says "Don't have an account? Register".

Fig 1:User Login



A register form titled "Register" centered on a light gray background. The form contains four input fields: "Username", "Email", "Password", and "Confirm Password". Below these fields is a blue button labeled "Register". At the bottom of the form, there is a link that says "Already have an account? Login".

Fig 2: Register Page

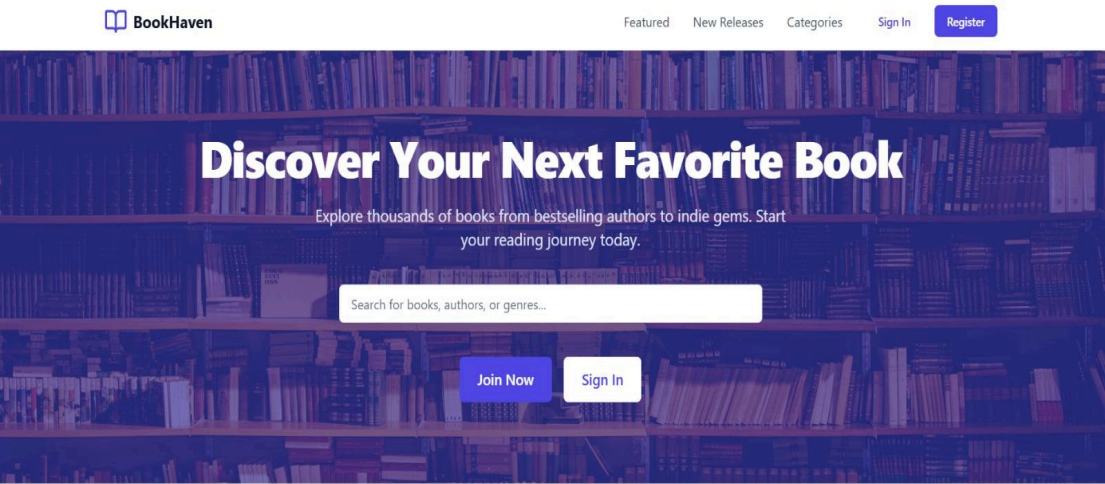


Fig 3: Dashboard

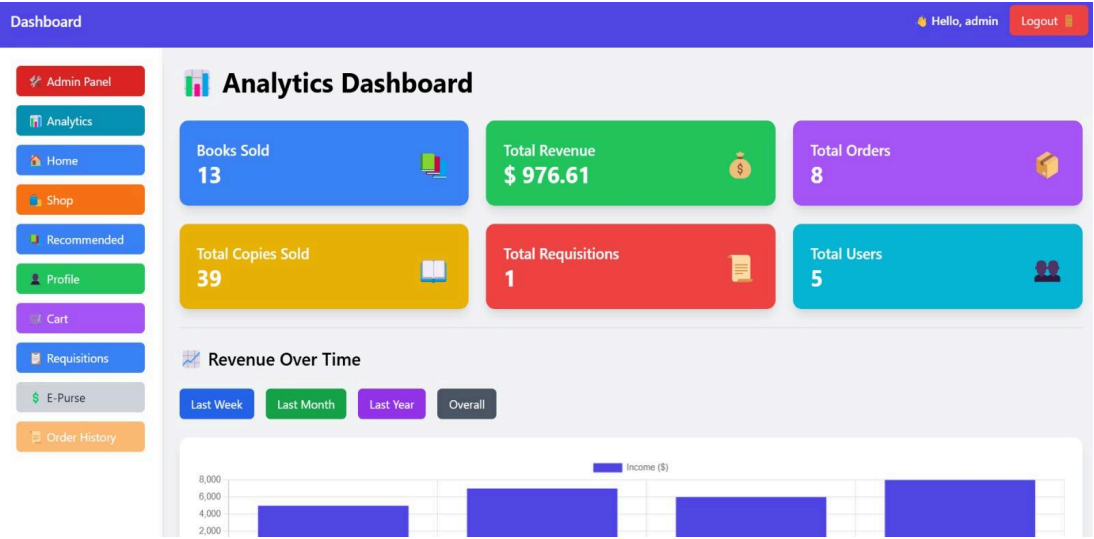


Fig 4: Analytics Dashboard

[Back to Shop](#)

A Journey to the Centre of the Earth

Alice Johnson
 Andrew Scott

Category: Sci-fi, Fiction
Edition: 1st Edition
Publisher: History Chronicles
Year: 2017
\$ 27.99

[Add to Requisition](#)

Book Summary

A historical fiction novel set during the Renaissance period. The protagonist embarks on a journey to uncover lost manuscripts.

Leave a Review

admin - March 24, 2025
 ★ 5.0/5
 m.m.

[Update Review](#)
[Delete](#)

Fig 5:Book Review

Online Book Store Admin

WELCOME: VALAGALA VIEW SITE / CHANGE PASSWORD / LOG OUT

Welcome to Book Store Dashboard

AUTHENTICATION AND AUTHORIZATION

Groups	Add	Change
Users	Add	Change

CUSTOMER

Customers	Add	Change
Order items	Add	Change
Orders	Add	Change
Requisitions	Add	Change
Reviews	Add	Change

STORE

Books	Add	Change
Categories	Add	Change
Writers	Add	Change

Recent actions

My actions

- Order #14 by admin - Pending Order
- Order #13 by user@4 - Pending Order
- Order #1 by user@3 - Pending Order
- Order #12 by user@4 - Pending Order
- Order #11 by user@4 - Paid Order
- Order #3 by user@4 - Paid Order
- Order #2 by user@4 - Paid Order
- admin - A Journey to the Centre of the Earth (5/5) Review
- admin - A Journey to the Centre of the Earth (3/5) Review
- The 5 AM Success Club (1st Edition) Book

Fig 6:Backend Database

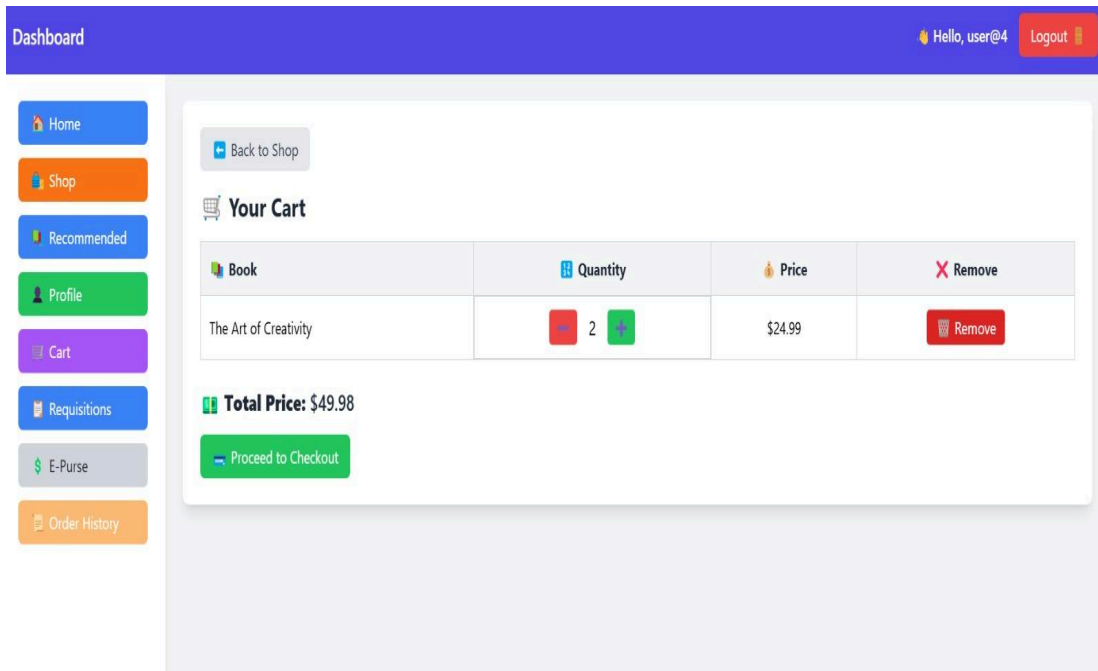


Fig 7:Cart

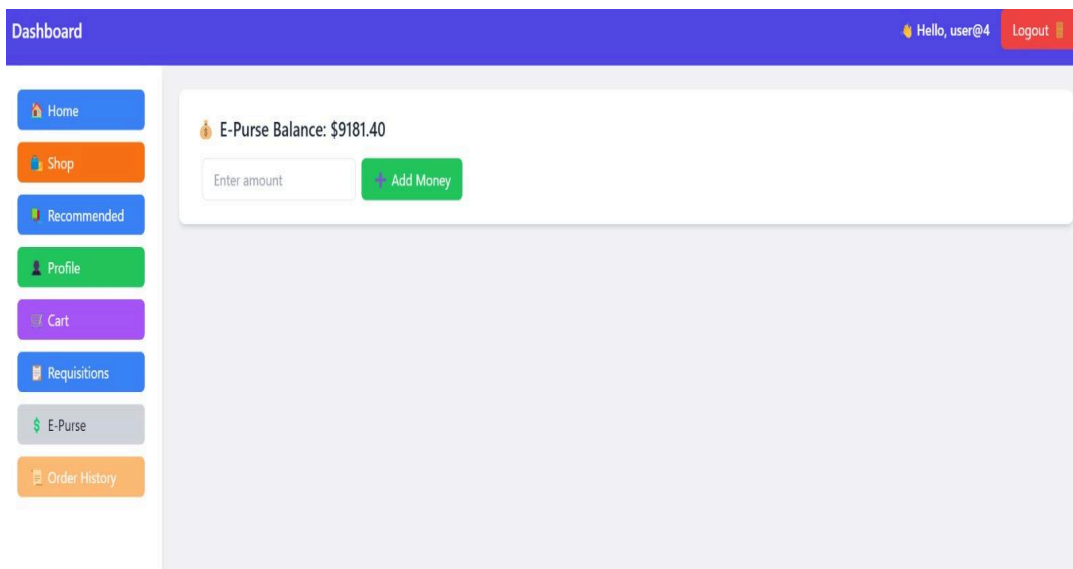


Fig 8:E-Purse

Dashboard
Hello, user@4
Logout

Home
Shop
Recommended
Profile
Cart
Requisitions
E-Purse
Order History

Back to Shop

E-Purse Checkout

Order Total: **\$49.98**
E-Purse Balance: **\$9181.40**

Address:

Enter your full address

State:

Enter your state

Pincode:

Enter your pincode

Country:

Enter your country

Confirm Payment

Back to Cart

Fig 8:E-Purse Checkout

Dashboard
Hello, user@4
Logout

Home
Shop
Recommended
Profile
Cart
Requisitions
E-Purse
Order History

Order History

Ordered on: March 25, 2025 17:22
Total Price: **\$49.98**
Status: **Pending**
Payment Mode: E-Purse

Ordered Books:

Ordered on: March 24, 2025 13:51
Total Price: **\$19.99**
Status: **Paid**
Payment Mode: E-Purse

Ordered Books:

- The Psychology of Money (x1) - \$19.99

Ordered on: March 24, 2025 13:46
Total Price: **\$14.99**
Status: **Paid**
Payment Mode: E-Purse

Ordered Books:

Fig 9:Order History

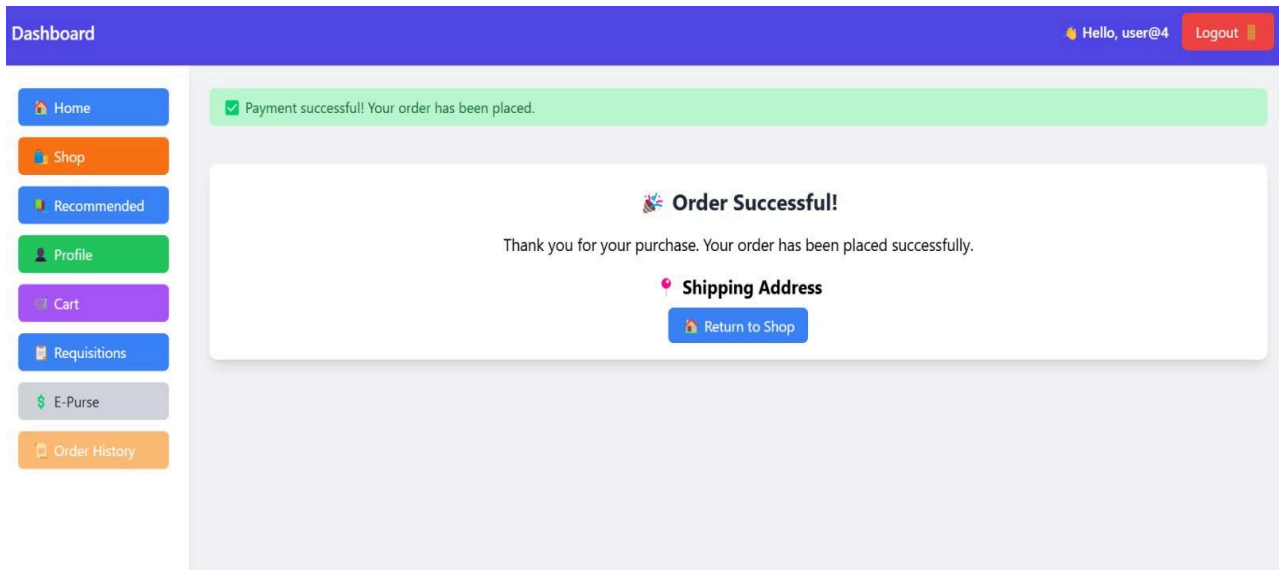


Fig 10:Order Success

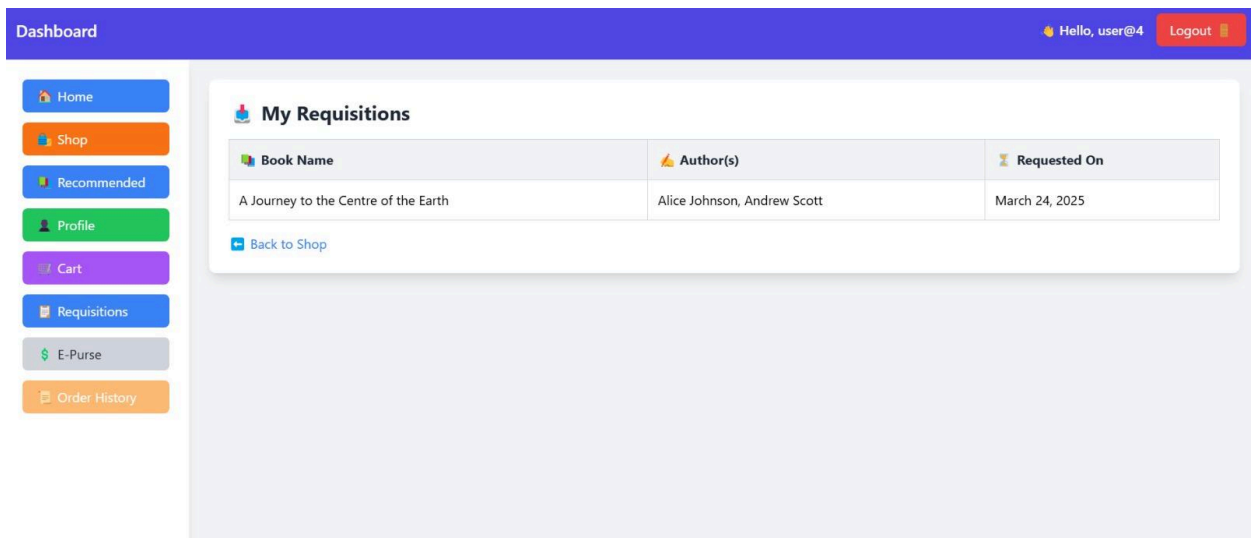


Fig 11:Requisition

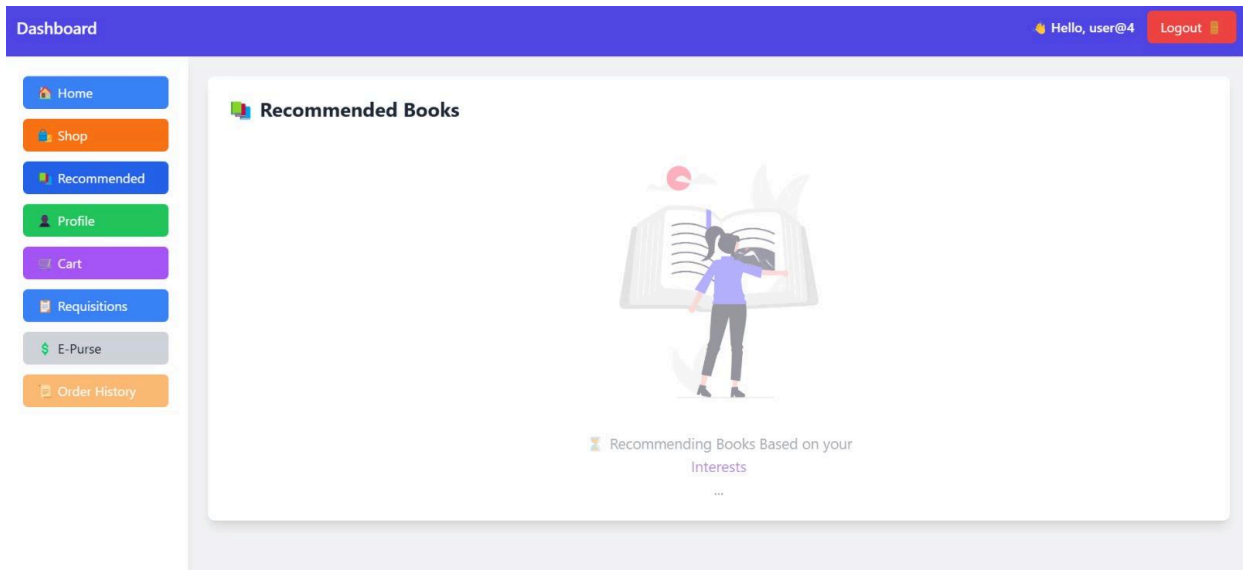


Fig 12: Recommending Books

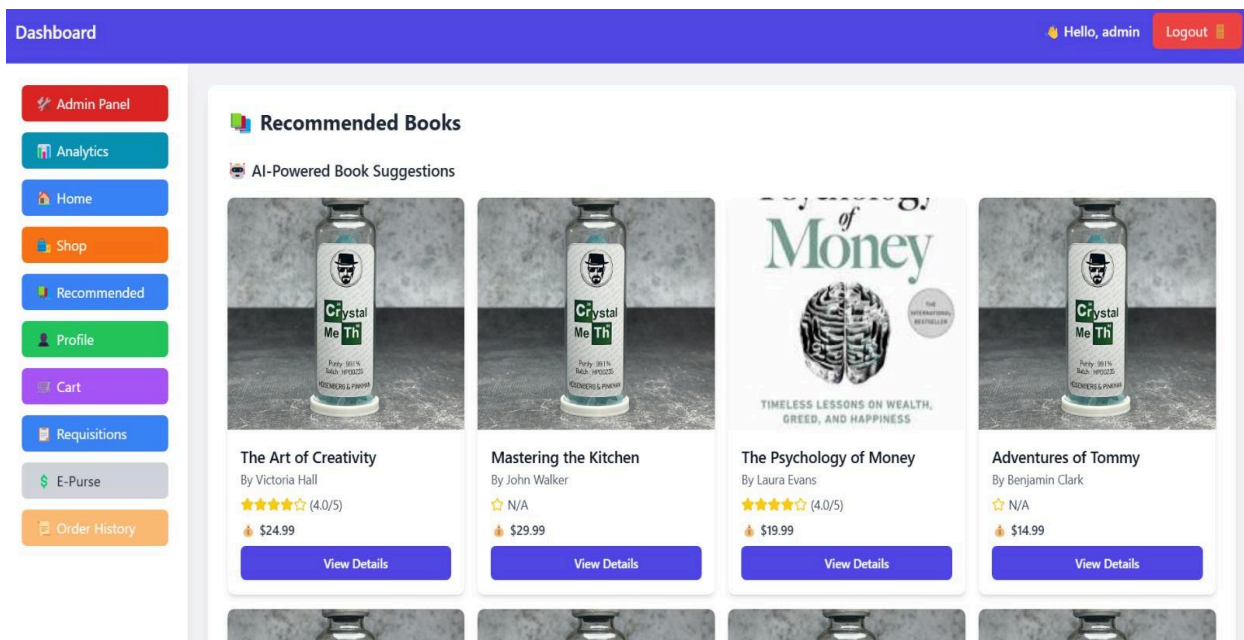


Fig 13: Recommended Books

9. Conclusion

The BookHaven project integrates **Advanced Database Management** with an **AI-driven recommendation system** using the Apriori algorithm from mlxtend. The system ensures secure transactions, efficient order processing, and a personalized user experience.

The implementation of AI-based recommendations significantly enhances customer engagement by analyzing purchase behavior and browsing history. The combination of **Django Full Stack API and SQLite3** provides an efficient and scalable solution for online book management.

10. Future Enhancements

- **Integration with External Payment Gateways** to support multiple transaction methods.
- **Development of a Mobile Application** for better accessibility.
- **Real-Time Stock Updates and Chat-Based Recommendations** for interactive user support.

This **ADBMS Lab Report** comprehensively documents the development, functionality, and technical architecture of **BookHaven**, providing a structured overview of its database management, AI-powered recommendations, and web application framework.

