

BOOKLOOP

A Mini Project Report

submitted by

FATHIMATH SANA C V

(MES24MCA-2016)

to the APJ Abdul Kalam Technological University
in partial fulfilment of the requirements for the award of the Degree

of

Master of Computer Applications



Department of Computer Applications

MES College of Engineering
Kuttippuram, Malappuram – 679582

October, 2025

Declaration

I undersigned hereby declare that the project report **BookLoop** submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala, is a Bonafide work done by me under supervision of Dr. Geever C Zacharias, Assistant Professor, Department of Computer Applications. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

FATHIMATH SANA CV

(MES24MCA-2016)

08-10-2025

DEPARTMENT OF COMPUTER APPLICATIONS
MES COLLEGE OF ENGINEERING, KUTTIPPURAM



CERTIFICATE

This is to certify that the report entitled **BookLoop** is a Bonafide record of the Mini Project work during the year 2025-26 carried out by **Fathimath Sana C V (MES24MCA-2016)** submitted to the APJ Abdul Kalam Technological University, in partial fulfilment of the requirements for the award of the Master of Computer Applications, under my guidance and supervision. This report in any form has not been submitted to any other University or Institution for any purpose.

Internal Supervisor

Head of The Department

Acknowledgment

I would like to express my sincere gratitude to Dr. Rahumathunza, Principal of MES College of Engineering, for providing the essential facilities and a supportive academic environment that enabled the successful completion of my postgraduate project titled BookLoop. I am deeply thankful to Prof. Hyderali K, Head of the Department of Computer Application, for his unwavering support, encouragement, and insightful suggestions throughout the course of my studies. My heartfelt appreciation goes to Ms. C M Sulaikha, Project Coordinator, for her continuous guidance, motivation, and constructive feedback, which significantly contributed to the effective progress of this work. I am especially grateful to my Project Guide, Dr. Geever C Zacharias, for his expert advice, thoughtful critiques, and consistent supervision, all of which played a vital role in the realization of this project. I also extend my sincere thanks to all the faculty members of the Department of Computer Application for their academic support and encouragement. Finally, I am profoundly grateful to my colleagues, friends, and family for their constant support, patience, and inspiration throughout this journey. Their presence and belief in me have been invaluable.

FATHIMATH SANA CV

(MES24MCA-2016)

Abstract

The BookLoop project is a web-based application designed to create a convenient and sustainable platform for students to buy, sell, rent, or exchange academic books within their campus community. The main objective of this system is to reduce the cost burden of purchasing new textbooks each semester and to promote the reuse of learning materials among students. In traditional systems, students often struggle to find affordable used books or reliable sources for exchanging them, leading to unnecessary expenses and resource wastage. The proposed system addresses these issues by offering a structured, digital, and user-friendly solution.

The platform enables users to register, upload book details, browse available listings, and communicate securely with other users. It includes separate modules for book management, search and filter operations, rental and sale transactions, payment tracking, complaint handling, reviews, and notifications. Administrators can monitor system activities, verify users, and manage complaints through an integrated admin dashboard. The system thus ensures security, transparency, and reliability in all book-related transactions.

The application is developed using HTML and CSS for the front-end design, ensuring a clean and responsive user interface. The Python programming language is used as the backend, while the Django framework provides a structured environment that supports rapid development and scalability. MySQL is implemented as the database management system to handle user information, book listings, transactions, and reviews efficiently.

By integrating these technologies, BookLoop provides a complete solution for academic book exchange, promoting affordability and environmental sustainability. The system encourages students to recycle educational resources, reduces the need for purchasing new books, and creates a community-driven approach to knowledge sharing. The successful implementation of this project demonstrates the potential of technology in addressing real-world academic challenges through efficient, reliable, and eco-friendly solutions

Table of Contents

List of Figures	viii
List of Tables	ix
Chapter 1. Introduction.....	1
1.1 Motivation	1
1.2 Objectives.....	2
1.3 Contributions.....	3
1.4 Report Organization	3
Chapter 2. System Study	4
2.1 Existing System.....	4
2.2 Proposed System	5
2.3 Functionalities of Proposed System	6
1. User Registration and Authentication	6
2. Book Management	6
3. Search and Filter Functionality	6
4. Rent and Purchase Requests.....	7
5. Payment Management	7
6. Review and Rating	7
7. Notification System.....	7
8. Complaint Management	8
9. Chat System.....	8
10. Admin Management	8
Chapter 3. Methodology	9
3.1 Introduction	9
3.2 Software Tools	10
3.2.1 Python.....	10
3.2.2 Django.....	11
3.2.3 MySQL	11
3.3 Module Description.....	12

3.3.1 User Module	12
3.3.2 Admin Module.....	12
3.4 User Story.....	14
3.5 Product Backlog	16
Table 3.3: Product Backlog	16
3.6 Project Plan	18
Table 3.4: Project plan	18
3.7 Sprint Backlog.....	20
3.8 Database Design.....	23
Chapter 4. Results and Discussions.....	30
4.1 Results	30
Chapter 5. Conclusion	33
References	34
Appendix	35
Appendix A Data Flow Diagram	35
Level 0	35
Level 1.0	36
Level 2.0	37
Appendix B ER Diagram.....	38
Appendix C SourceCode	39
Appendix D Screenshot	52

List of Figures

Figure 4.1: Home page **Error! Bookmark not defined.**0

Figure 4.2: Book Listing Form 31

Figure 4.3: Book search and filter 32

Figure 4.1: Home Page 30

Figure 4.1:The home page displays an overview of the system, providing navigation options for users to browse available books, view categories, and access other sections like login and registration. It serves as the entry point to the system. 30

List of Tables

Table 3.1: Software tools and technologies used for project development	10
Table 3.2: User Story	14
Table 3.3: Product Backlog	16
Table 3.4: Project Plan.....	18
Table 3.5: Sprint Backlog	20
Table 3.6: User_table.....	23
Table 3.7: Category_table.....	24
Table 3.8: Book_table.....	24
Table 3.9: Request_table.....	25
Table 3.10: Rent_table.....	26
Table 3.11: Payment_table	27
Table 3.12: Rent Payment_table	27
Table 3.13: Review_table	28
Table 3.14: Notification_table	28
Table 3.15: Complaint_table.....	29
Table 3.16: Chat_table.....	29

Chapter 1. Introduction

Books are one of the most essential resources for students in any academic environment. However, the high cost of textbooks often places a significant financial burden on students, while many books remain unused after the completion of a semester or course. This leads to both economic inefficiency and material waste. To address these challenges, *BookLoop* introduces a peer-to-peer digital platform designed to facilitate the renting and reselling of academic books within college campuses.

The platform creates a structured and secure ecosystem that allows students to list books for sale or rent, search for required materials, and manage transactions easily. It not only helps students save money but also encourages the reuse of resources, promoting environmental sustainability and a sense of community sharing. Built using Python (Django) for the backend and modern web technologies for the frontend, BookLoop provides a user-friendly interface with features like user authentication, dashboards, book search, in-app messaging, and return reminders.

By integrating technology into the book-sharing process, BookLoop aims to revolutionize the way students access academic materials—making learning more affordable, sustainable, and collaborative.

1.1 Motivation

The high cost of academic books has long been a financial challenge for students, especially in higher education, where each semester often requires multiple textbooks. Many of these books

are used for only a short period and then remain idle, leading to both financial waste and unnecessary consumption of resources. At the same time, other students struggle to access the same materials due to limited availability or high market prices. This imbalance inspired the creation of BookLoop — a digital platform that enables students to rent or resell academic books among peers in a structured and secure manner. By promoting the reuse of textbooks, the platform not only helps reduce students' financial burdens but also supports environmental sustainability by minimizing paper waste. Moreover, it encourages collaboration and community engagement within the campus, fostering a culture of sharing and mutual support. Ultimately, *BookLoop* is motivated by the vision of creating a more affordable, accessible, and sustainable academic ecosystem that benefits both students and the environment.

1.2 Objectives

The main objectives of the BookLoop project are:

- To reduce the financial burden on students by enabling affordable access to academic books through renting and reselling.
- To minimize resource wastage by keeping books in circulation instead of leaving them unused after each semester.
- To create a structured, secure, and user-friendly platform for peer-to-peer book exchange within the campus.
- To promote environmental sustainability by encouraging the reuse and sharing of academic resources.
- To enhance accessibility of study materials for all students, regardless of their financial background.
- To provide efficient features such as book search, rental reminders, user dashboards, and secure in-app communication.
- To build a scalable and reliable digital solution using modern technologies such as Python (Django), MySQL.

1.3 Contributions

- Developed a peer-to-peer web platform for renting and selling academic books.
- Helped students save money by providing affordable access to textbooks.
- Created a secure system using user authentication and admin control.
- Designed an easy-to-use interface for listing, searching, and managing books.
- Added features like rental reminders and in-app messaging for better communication.
- Used modern technologies like **Python (Django)** and **MySQL** for development.

1.4 Report Organization

The project report is divided into five chapters:

- **Chapter 2 – System Study:** Explains the existing system, its limitations, and the proposed system functionalities.
- **Chapter 3 – Methodology:** Describes the project methodology, including module descriptions, user stories, sprint details, and database design.
- **Chapter 4 – Results and Discussions:** Provides outputs of the implemented system and discussion on how objectives were met.
- **Chapter 5 – Conclusion:** Summarizes the work done, contributions, and scope for future enhancements.

Chapter 2. System Study

A system study involves analyzing the existing system, identifying its limitations, and designing a proposed system that overcomes those drawbacks. It helps in understanding the functional requirements, user needs, and overall workflow of the project. For the BookLoop system, the study focuses on improving the process of book exchange, rental, and sales among students in an efficient and organized way.

2.1 Existing System

In the current scenario, students who wish to buy, sell, or rent academic books usually depend on informal and manual methods. These include sharing information through social media groups, word of mouth, or college notice boards. This approach is completely unorganized, time-consuming, and unreliable, as it depends on chance interactions between buyers and sellers.

There is no proper system to maintain records of books available, users involved, or the status of transactions. Students often have to contact multiple people to find the books they need, which can be stressful and inefficient. Similarly, those who want to sell or rent out their books have no guaranteed way to reach interested buyers or renters.

Libraries are another common source for books, but they have a limited number of copies that cannot serve all students, especially during peak academic seasons. Many students are forced to buy new books at high prices, even for short-term use. This not only causes financial strain but also leads to wastage of resources, as most books remain unused after the semester ends.

The lack of a structured system also means there is no verification or security in transactions. Users may face problems such as fake listings, unresponsive buyers, or disputes over book

conditions and payments. Additionally, there is no way to provide feedback, ratings, or complaints, which reduces user trust and transparency.

2.2 Proposed System

The proposed system, BookLoop, is a web-based platform designed to simplify and organize the process of buying, selling, and renting academic books among students. It aims to overcome the drawbacks of the existing manual system by providing an efficient, reliable, and user-friendly online solution. The system connects students who wish to sell or rent out their books with those looking to buy or borrow them, all within a secure digital environment.

The BookLoop application allows users to create an account, upload book details such as title, author, edition, price, and category, and manage their listings easily. Other students can browse or search for the required books using various filters such as subject, author, or semester. Once they find a suitable book, they can send a request to the owner, and the transaction details are securely managed through the platform.

The system also includes a chat feature that enables direct communication between users, a review and rating system to promote trust, and an admin module that monitors all activities, handles complaints, and ensures the safety and reliability of user interactions. Furthermore, **notifications** and **reminders** are integrated to alert users about rental return dates or updates on their requests.

The platform is developed using HTML and CSS for the frontend, providing a clean and responsive interface. The backend is powered by Python with the Django framework, which ensures robust functionality and smooth data handling. The MySQL database is used to store user information, book listings, transactions, and feedback securely.

By automating the process of book exchange, BookLoop saves time, reduces cost, and encourages the reuse of educational materials. It promotes an eco-friendly approach by reducing the demand for new books and minimizing waste. The system enhances transparency, security, and accessibility, offering a practical and sustainable solution for students in academic institutions.

2.3 Functionalities of Proposed System

The proposed system, BookLoop, provides a set of well-defined functionalities that help students and administrators manage book rentals, sales, and exchanges efficiently. The system ensures smooth interaction between users, maintains secure transactions, and promotes reusability of academic materials.

The main functionalities of the proposed system are described below:

1. User Registration and Authentication

- New users can register by providing personal details such as name, email, phone number, and password.
- Existing users can log in securely using their credentials.
- Password protection and authentication mechanisms ensure authorized access to the system.

2. Book Management

- Registered users can add books by entering details such as title, author, category, edition, and price.
- Users can edit or delete their listings at any time.
- Uploaded books are stored in the database with related user information.

3. Search and Filter Functionality

- Users can easily search for books using filters like book name, author, category, or semester.
- Search results are displayed in an organized and user-friendly format.

4. Rent and Purchase Requests

- Users can send a request to the owner to rent or buy a selected book.
- The system records the request details, status, and date.
- Owners can accept or reject requests through their dashboard.

5. Payment Management

- The system manages payment details for both rent and sale transactions.
- Users can view the payment amount, date, and status of each transaction.
- Payment records are securely stored in the database for future reference.

6. Review and Rating

- After a transaction, users can provide reviews and ratings for the book or the transaction.
- This feature helps maintain trust and transparency among users.

7. Notification System

- Automatic notifications are sent to users regarding book requests, payment confirmations, and return reminders.
- This ensures users stay updated on their ongoing transactions.

8. Complaint Management

- Users can submit complaints or issues related to the system or other users.
- The admin can view and reply to these complaints through the admin dashboard.

9. Chat System

- Users can communicate directly with each other through an in-app chat feature.
- This helps clarify book details, prices, and transaction terms easily.

10. Admin Management

- The administrator has full control over the system.
- Admin can monitor users, manage books, review complaints, and maintain system security.
- Ensures smooth functioning and reliability of the platform.

Chapter 3. Methodology

The development of BookLoop followed a structured approach to ensure smooth and efficient implementation. The process began with requirement analysis, where the problems faced by students in renting and selling books were studied and system needs were identified. During the system design phase, the structure of the platform, database, and user interfaces were planned using data flow diagrams (DFD) and ER diagrams. In the development phase, the backend was created using Python (Django), the frontend was designed using HTML and CSS, and MySQL was used as the database. After development, the system underwent testing to check for errors and ensure that all modules worked properly. Once tested, the system was implemented and made available for users. Finally, during the maintenance phase, regular updates, bug fixes, and improvements are made to enhance system performance and user experience.

3.1 Introduction

Books are very important for students, but buying new ones every semester is costly. Many books are used only for a short time and then remain unused, which leads to waste. To solve this problem, BookLoop provides an online platform where students can rent or sell their academic books to others on the same campus.

This system helps students save money, reuse books, and make learning materials easily available to everyone. It is simple to use, safe, and promotes sharing among students.

BookLoop is developed using Python (Django) for the backend, HTML and CSS for the frontend, and MySQL for storing data. It creates an easy and eco-friendly way for students to exchange books.

3.2 Software Tools

The development of **BookLoop** required various tools and technologies that support efficient web application design and implementation. The system was developed using **Python** and the **Django** framework, with **MySQL** as the backend database. The selected tools ensure reliability, security, and scalability of the system.

Table 3.1: List of Software Tools or Languages Used for the Project Development

Operating System	Windows 10/11
Front End	JavaScript HTML, CSS etc..
Back End	Python
Framework	Django
Database	MySQL
IDE	PyCharm
Version Control	Git

3.2.1 Python

Python was chosen as the backend language for BookLoop because it is simple, powerful, and easy to understand. It allows faster development with fewer lines of code, making it suitable for student projects and real-world applications. Using the Django framework, Python provides built-in features for handling databases, authentication, and security, which reduces development time and improves reliability.

Django follows the Model-View-Template (MVT) architecture, which makes the system organized and easier to maintain. It also has strong community support and many ready-to-use libraries that help in building web applications efficiently. Python is compatible with various databases like MySQL, which is used in this project, ensuring smooth data handling.

Overall, Python with Django provides a secure, scalable, and efficient backend solution, making it the best choice for developing the BookLoop platform.

3.2.2 Django

Django was chosen as the framework for BookLoop because it is a powerful, secure, and easy-to-use web development framework based on Python. Django provides built-in features like user authentication, database management, admin panel, and form handling, which help in developing web applications faster and more efficiently.

It follows the Model-View-Template (MVT) architecture, which keeps the project well-structured and easy to maintain. Django also offers strong security features such as protection against SQL injection, cross-site scripting, and cross-site request forgery.

The framework supports multiple databases, including MySQL, which is used in this project, and provides smooth integration with frontend technologies like HTML and CSS. Django's scalability and extensive library support make it suitable for both small projects and large applications.

Overall, Django was chosen because it ensures speed, security, and reliability, making it an ideal framework for developing the BookLoop platform.

3.2.3 MySQL

MySQL was chosen as the database for the BookLoop project because it is reliable, easy to use, and well-suited for web-based applications. It provides excellent performance, scalability, and security, which are essential for handling user and book data efficiently. MySQL supports structured data storage using tables and relationships, making it ideal for managing information such as user accounts, book listings, transactions, and feedback.

It integrates smoothly with Python (Django) and allows fast data retrieval through optimized queries. MySQL also ensures data consistency and supports multiple users accessing the system at the same time without performance issues. Being an open-source and widely supported database, it reduces development costs while providing strong community support. Overall, MySQL was selected for its speed, reliability, compatibility, and ease of integration with the technologies used in *BookLoop*.

3.3 Module Description

The BookLoop system is divided into several interconnected modules that work together to achieve the objectives of the project. Each module performs specific functions and interacts with others to ensure smooth operation of the system. The main modules of the system are described below

3.3.1 User Module

This module manages all the activities related to users. It allows new users to register, existing users to log in, and provides access to their personal dashboard. Users can update profiles, view listed books, and manage their own book uploads.

Functions:

- Register and log in securely.
- Manage user profiles.
- View and manage personal book listings.
- Communicate with other users through chat.

3.3.2 Admin Module

The Admin Module is responsible for overall system supervision. It provides tools for managing users, monitoring activities, handling complaints, and maintaining the system database.

Functions:

- Manage users and book listings.
- Monitor system activity and performance.
- View and respond to complaints.
- Ensure system security and reliability.
- **Book Monitoring:** Admin reviews and manages all uploaded books to ensure authenticity and quality.

- **Complaint Handling:** Admin can view complaints submitted by users and reply with appropriate resolutions.
- **Payment Oversight:** Admin monitors payment transactions to ensure system reliability and prevent errors.
- **Notification Management:** Admin can broadcast important updates or announcements to users.
- **System Maintenance:** Ensures the stability, performance, and security of the application.
- **Report Generation:** Admin can generate summary reports for transactions, users, and activities.

3.4 User Story

The user stories describe the system's functionality from the perspective of different users. They help in understanding the goals and requirements of users interacting with the system. Each story defines what the user wants to achieve and why.

Table 3.2:User Story

User Story ID	As a type of user	I want to	So that I can
1	Student	Register and log in to the system	Access the platform securely
2	Student	Add books for rent or sale	Share my books with other students
3	Student	Search for books by title, subject, or semester	Find the books I need easily
4	Student	Send requests to rent or buy books	Borrow or purchase books conveniently
5	Student	Receive return reminders	Return rented books on time
6	Student	Chat with other users	Discuss book details securely
7	Student	Manage my profile, listings, and requests	Keep track of my activities

8	Student	Send feedback or complaints to admin	Report issues or give suggestions
9	Admin	Log in securely	Manage the system safely
10	Admin	View and manage users and books	Monitor and control all activities
11	Admin	Approve or remove book listings	Maintain the quality of listings
12	Admin	View and reply to user complaints	Resolve issues efficiently
13	Admin	Monitor transactions and activities	Ensure the platform runs smoothly

3.5 Product Backlog

Provide product backlog in tabular form with table number and refer that table in the section. The Product Backlog lists all the features, tasks, and enhancements planned for the BookLoop system. Each item is prioritized by importance and estimated effort, forming the basis for sprint planning and tracking project progress.

Table 3.3: Product Backlog

Product Backlog ID	Name	Priority	Estimate (Hours)	Status
1	User Registration and Login	High	6	Completed
2	Admin Login and Dashboard	High	8	Completed
3	Book Listing (Add, Edit, Delete)	High	10	Completed
4	Book Search and Filter Function	Medium	7	Completed
5	Rent and Sale Management	High	9	Completed
6	Return Reminder Notification	Medium	5	Completed

7	In-App Messaging	Medium	6	Completed
8	Feedback and Complaint Module	Medium Low	4	Completed
9	Admin Book and User Management	High	8	Completed
10	JWT Authentication Setup	High	6	Completed
11	Profile Management (User Side)	Low	4	Completed
12	UI Design using HTML and CSS	High	10	Completed
13	UI Design using HTML and CSS	High	6	Completed

3.6 Project Plan

The Project Plan outlines the timeline, duration, and completion status of each sprint in the development of the BookLoop system. The project was divided into multiple sprints, each focusing on specific functionalities to ensure systematic progress and timely delivery.Sprint Backlog

Table 3.4: Project plan

User Story ID	Task Name	Start Date	End Date	Days	Status
1	Sprint 1 – Database Design & User Authentication	10/07/2025	23/07/2025	14	Completed
2	Sprint 2 – Profile Management & Book Creation	24/07/2025	06/08/2025	14	Completed
3	Sprint 3 – Book Listing, Search & Filter	07/08/2025	20/08/2025	14	Completed
4	Sprint 4 – Rent, PurchasePayment Module	21/08/2025	03/09/2025	14	Completed

5	Sprint 5 – Admin Dashboard & Review Management	04/09/2025	17/09/2025	14	Completed
6	Sprint 6 – Notification, Chat & Complaint Module	18/09/2025	01/10/2025	14	Completed
7	Sprint 7 – UI/UX Enhancement, Deployment & Testing	02/10/2025	18/10/2025	17	Completed

3.7 Sprint Backlog

The **Sprint Backlog** outlines the detailed breakdown of work planned for each sprint during the **BookLoop** system development. It tracks progress, estimated hours, and actual hours spent daily for each backlog item. This approach helped ensure efficient workload distribution and timely completion of all project tasks.

Table 3.5 Sprint Backlog

Backlog Item	Status	Completion Date	Original Estimation (hrs)	Day 1 (hrs)	Day 2 (hrs)	Day 3 (hrs)	Day 4 (hrs)	Day 5 (hrs)	Day 6 (hrs)	Day 7 (hrs)	Day 8 (hrs)	Day 9 (hrs)	Day 10 (hrs)
SPRINT 1													
User Registration & Login	Completed	10/07/2025	6	2	2	2	0	0	0	0	0	0	0
Admin Dashboard	Completed	16/07/2025	8	2	2	2	2	0	0	0	0	0	0
SPRINT 2													
Book Listing Module	Completed	24/07/2025	10	2	2	2	2	2	0	0	0	0	0

Search & Filter Function	Completed	30/07/2025	7	2	2	1	2	0	0	0	0	0	0
SPRINT 3													
Rent & Sale Management	Completed	07/08/2025	9	2	2	2	2	1	0	0	0	0	0
Return Reminder Notification	Completed	13/08/2025	5	1	1	2	1	0	0	0	0	0	0
SPRINT 4													
In-App Messaging	Completed	27/08/2025	6	1	2	2	1	0	0	0	0	0	0
Feedback & Complaint Module	Completed	01/09/2025	4	2	2	0	0	0	0	0	0	0	0

SPRINT 5													
Admin Management	Completed	05/09/2025	8	2	2	2	2	0	0	0	0	0	0
UI Design (HTML, CSS)	Completed	10/09/2025	10	2	2	2	2	2	0	0	0	0	0

3.8 Database Design

The **BookLoop** system uses a **relational database** implemented with **MySQL** as its backend, integrated through Django’s **Object-Relational Mapping (ORM)** framework. The database is designed to efficiently store and manage data related to users, books, categories, requests, payments, reviews, notifications, and chats.

The database schema ensures referential integrity and optimized relationships among entities using **primary keys, foreign keys, and one-to-many associations**. This structure provides a robust and scalable foundation, enabling **secure data management, quick retrieval**, and smooth interaction between application layers.

Table 3.6: User_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each user
Login_id	Integer	Foreign Key (User.id), Not Null	Reference to Django’s built-in User model
Name	Varchar(100)	Not Null	Full name of the user
Email	Varchar(100)	Unique, Not Null	User’s email address
Phone	BigInt	Not Null	Contact number of the user
Place	Varchar(100)	Not Null	User’s location
Pincode	BigInt	Not Null	Postal code of the user
District	Varchar(100)	Not Null	User’s district name

Photo	Varchar(225)	Nullable	Profile image path
latitude	Float	Nullable	Latitude coordinate of user's location
longitude	Float	Nullable	Longitude coordinate of user's location

Table 3.6: User_table – Stores user details including personal and contact information.

Table 3.7: Category_table

Field	Data Type	Constraint	Description
id	Integer	Primary Key, Auto Increment	Unique identifier for each category
category	Varchar(100)	Not Null	Name of the book category

Table 3.7: Category_table – Contains the list of book categories available in the system.

Table 3.8: Book_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each book
User_id	Integer	Foreign Key (User_table.id), Not Null	Owner of the book
Category_id	Integer	Foreign Key (Category_table.id), Not Null	Category of the book

Name	Varchar(100)	Not Null	Name of the book
Author	Varchar(100)	Not Null	Author of the book
Photo	Varchar(100)	Nullable	Path of book cover image
Details	Text	Nullable	Description or additional details of the book
price	Decimal(10,2)	Not Null	Sale price of the book
Rent_price	Decimal(10,2)	Not Null	Rental price of the book

Table 3.8: Book_table – Manages all book-related data such as title, author, price, and owner.

Table 3.9: Request_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each request
User_id	Integer	Foreign Key (User_table.id), Not Null	User who sent the request
Book_id	Integer	Foreign Key (Book_table.id), Not Null	Requested book
Status	Varchar(100)	Default 'Pending'	Current status of the request
date	Date	Auto Now	Date of request submission

Table 3.9: Request_table – Records user requests for buying or renting books.

Table 3.10: Rent_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each rental
User_id	Integer	Foreign Key (User_table.id), Not Null	User renting the book
Book_id	Integer	Foreign Key (Book_table.id), Not Null	Rented book
Status	Varchar(100)	Default 'Active'	Rental status
date	Date	Auto Now	Date of renting transaction

Table 3.10: Rent_table – Tracks rental transactions made by users.

Table 3.11: Payment_table

eld	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each payment
Request_id	Integer	Foreign Key (Request_table.id), Not Null	Associated request
Amount	Decimal(10,2)	Not Null	Payment amount
Status	Varchar(100)	Default 'Success'	Payment status
date	Date	Auto Now	Payment date

Table 3.11: Payment_table – Stores payment details for purchase transactions.

Table 3.12: Rent_payment_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each rent payment
Rent_id	Integer	Foreign Key (Rent_table.id), Not Null	Associated rent record
Amount	Decimal(10,2)	Not Null	Payment amount for rent
Status	Varchar(100)	Default 'Success'	Payment status
date	Date	Auto Now	Payment date

Table 3.12: Rent_payment_table – Manages payment records for rented books.**Table 3.13:** Review_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each review
User_id	Integer	Foreign Key (User_table.id), Not Null	Reviewer user
Book_id	Integer	Foreign Key (Book_table.id), Not Null	Book being reviewed
Review	Text	Nullable	Text of the user's review

Rating	Varchar(10)	Not Null	Rating given by the user
date	Date	Auto Now	Review date

Table 3.13: Review_table – Contains user reviews and ratings for books.

Table 3.14: Notification_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each notification
Rent_id	Integer	Foreign Key (Rent_table.id), Not Null	Related rent record
User_id	Integer	Foreign Key (User_table.id), Not Null	User receiving the notification
Notification	Varchar(255)	Not Null	Notification message
date	Date	Auto Now	Notification date

Table 3.14: Notification_table – Stores notifications for users regarding requests, approvals, or updates.

Table 3.15: Complaint_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each complaint

User_id	Integer	Foreign Key (User_table.id), Not Null	User submitting the complaint
Complaint	Text	Not Null	Complaint content
Reply	Text	Nullable	Admin reply to the complaint
date	Date	Auto Now	Date of complaint submission

Table 3.15: Complaint_table – Records user complaints and admin responses.

Table 3.16: Chat_table

Field	Data Type	Constraint	Description
Id	Integer	Primary Key, Auto Increment	Unique identifier for each message
Form_id	Integer	Foreign Key (User.id), Not Null	Sender of the message
To_id	Integer	Foreign Key (User.id), Not Null	Receiver of the message
Message	Varchar(255)	Not Null	Chat message content
Date	Date	Auto Now	Message sent date

Table 3.16: Chat_table – Maintains chat messages between users for book-related communication.

Chapter 4. Results and Discussions

This chapter presents the outcomes of the BookLoop project and explains how the developed system meets its objectives. It includes the main features and screenshots of important modules such as book listing, search, rental management, and the admin panel. The discussion highlights how the system improves the traditional book exchange process by making it easier, faster, and more secure for students. Overall, the results show that BookLoop is a reliable and user-friendly platform for managing academic book rentals and sales within the campus.

4.1 Results

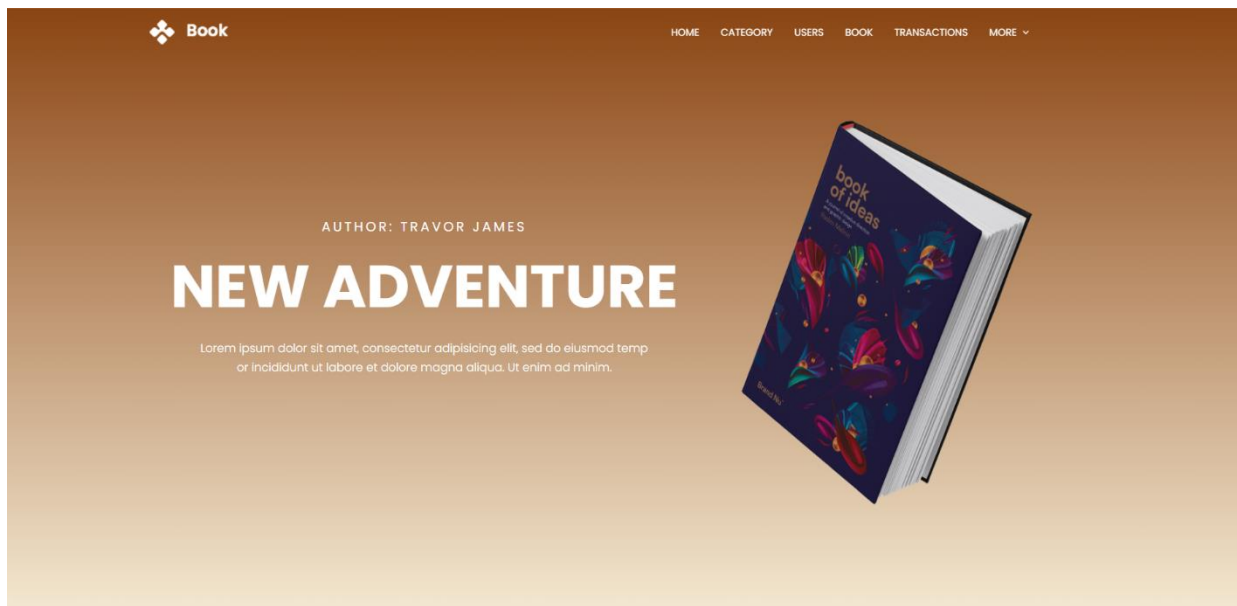


Figure 4.1: Home Page

Figure 4.2: The home page displays an overview of the system, providing navigation options for users to browse available books, view categories, and access other sections like login and registration. It serves as the entry point to the system.

Add New Book

Name
An orphans war

Author
Molly Greens

Photo
Choose File zay.jpg

Details
A historical fiction novel set in England during World War II

Price
500

Rent Price
300

Category
Fiction

Add Book

Figure 4.2: Book Listing Form

Error! Reference source not found. This form allows users to add a book for rent or sale by entering details such as book title, author, edition, price, and description. Once submitted, the book is stored in the database and displayed in the listing section for other users to view.

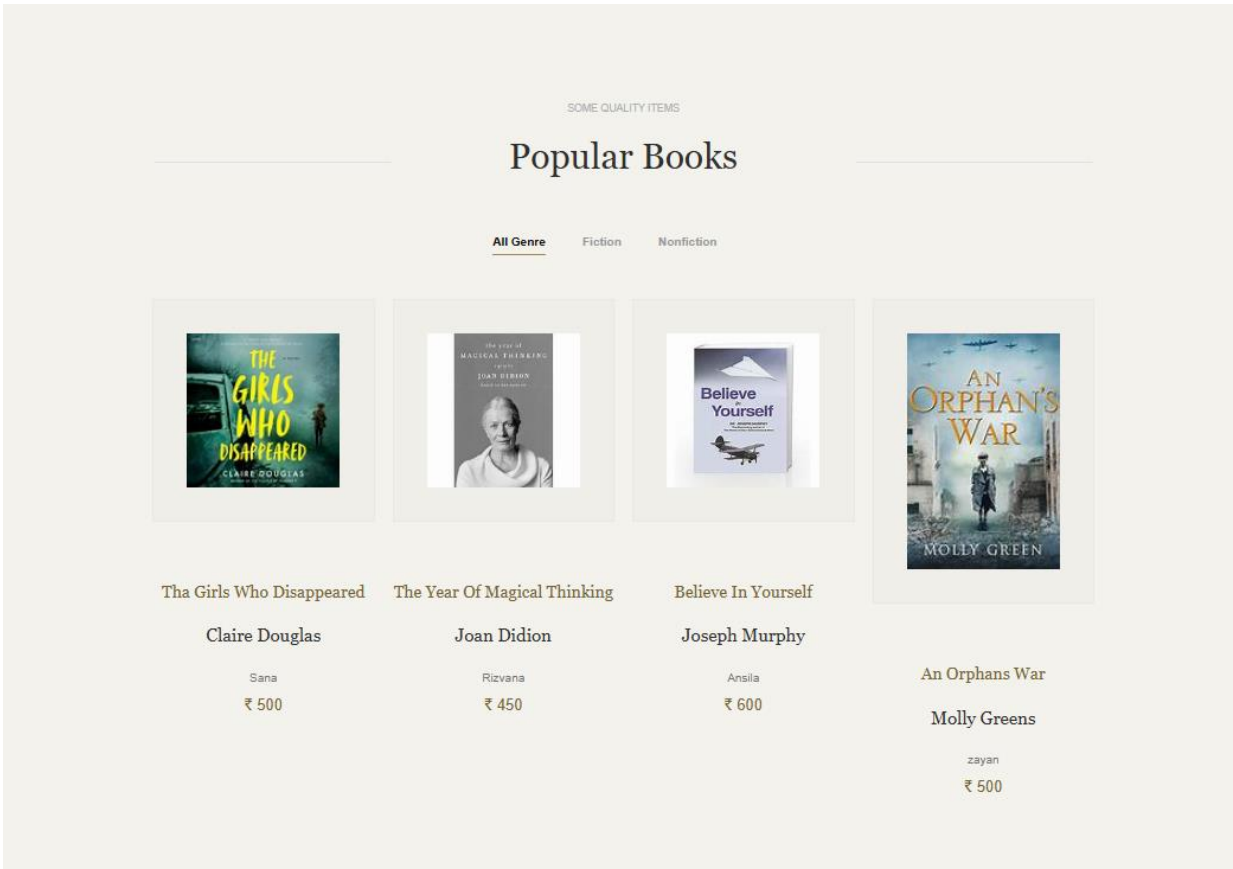


Figure 4.3: Book Search and Filter Page

Figure 4.3:This page helps users search for books based on criteria such as title, subject, or semester. The filter function improves usability by allowing quick access to the required books.

Chapter 5. Conclusion

The **BookLoop** project successfully provides a practical and sustainable solution for managing academic books within a campus environment. It offers an online platform where students can rent and sell books easily, reducing costs and minimizing waste. The system's key features—such as user registration and login, book listing, search and filter options, rental reminders, feedback system, and admin control—work together to create a secure and user-friendly experience. Developed using Python (Django), HTML, CSS, and MySQL, the project achieves its main goal of promoting affordable access to learning resources and encouraging book reuse.

However, certain limitations were encountered during development, such as the lack of advanced payment integration and mobile app compatibility due to time constraints. These can be addressed in future work by adding online payment gateways, mobile responsiveness, and advanced recommendation features. Overall, BookLoop demonstrates the potential of technology in supporting sustainable learning practices and improving accessibility for students.

References

- [1] Django Software Foundation, "*Django Documentation*," [Online]. Available: <https://docs.djangoproject.com/>. [Accessed 20 October 2024].
- [2] Python Software Foundation, "*Python Official Documentation*," [Online]. Available: <https://www.python.org/doc/>. [Accessed 20 October 2024].
- [3] Oracle Corporation, "*MySQL Reference Manual*," [Online]. Available: <https://dev.mysql.com/doc/>. [Accessed 20 October 2024].
- [4] Mozilla Developer Network (MDN), "*HTML and CSS Web Docs*," [Online]. Available: <https://developer.mozilla.org/>. [Accessed 20 October 2024].
- [5] W3Schools, "*Web Development Tutorials – HTML, CSS, Django, and MySQL*," [Online]. Available: <https://www.w3schools.com/>. [Accessed 20 October 2024].
- [6] JWT.io, "*Introduction to JSON Web Tokens (JWT)*," [Online]. Available: <https://jwt.io/introduction/>. [Accessed 20 October 2024].
- [7] Vercel Inc., "*Vercel Deployment Platform Documentation*," [Online]. Available: <https://vercel.com/docs>. [Accessed 20 October 2024].

Appendix

Appendix A Data Flow Diagram

Level 0

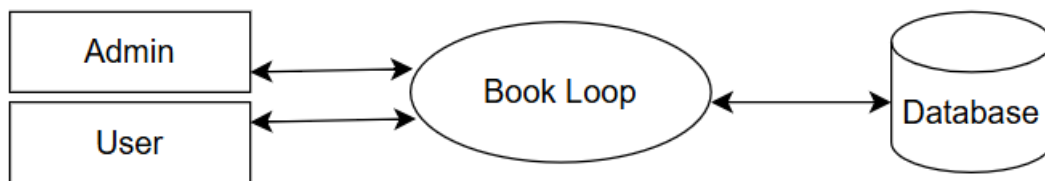


Figure 4.4:level 0

Figure 4.4: Shows how Admin and User interact with the central Book Loop, which connects to the Database

Level 1.0

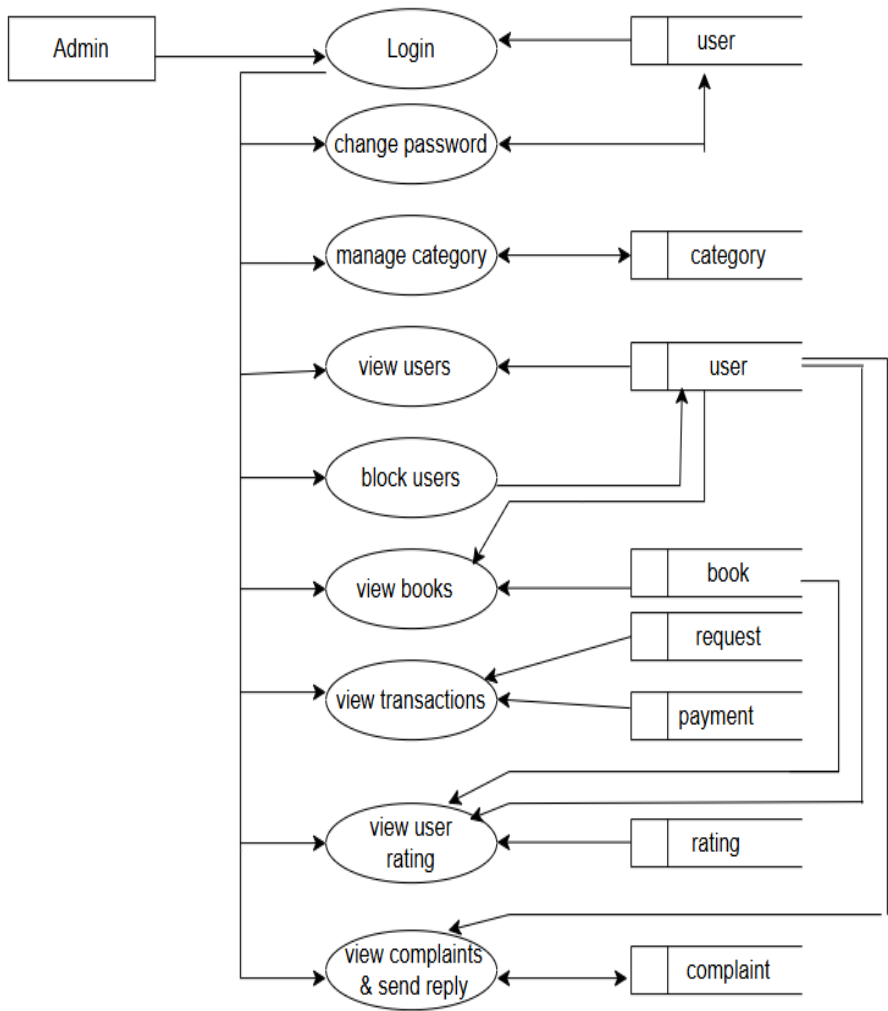


Figure 4.5:level 1.0

Figure 4.5: Outlines Admin panel functions like managing users, books, and complaints after login.

Level 2.0

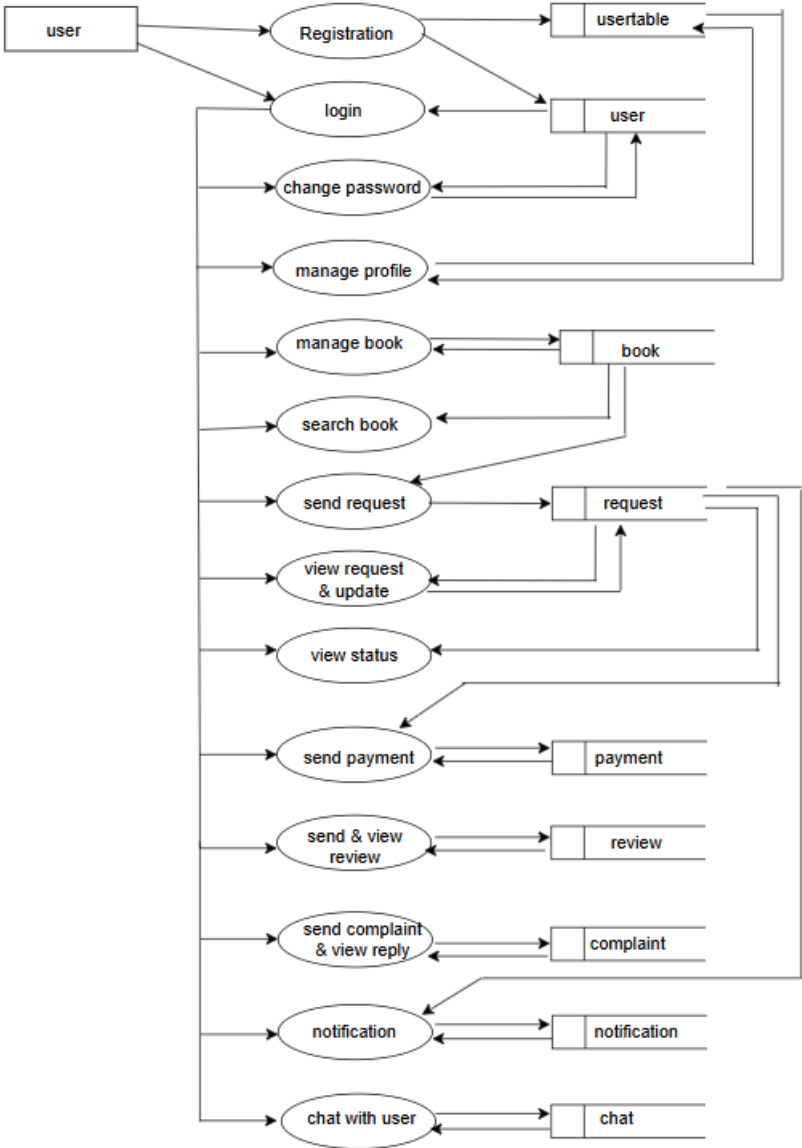


Figure 4.6:level 2.0

Figure 4.6 Maps user interactions with system processes and data stores through a detailed data flow diagram.

Appendix B ER Diagram

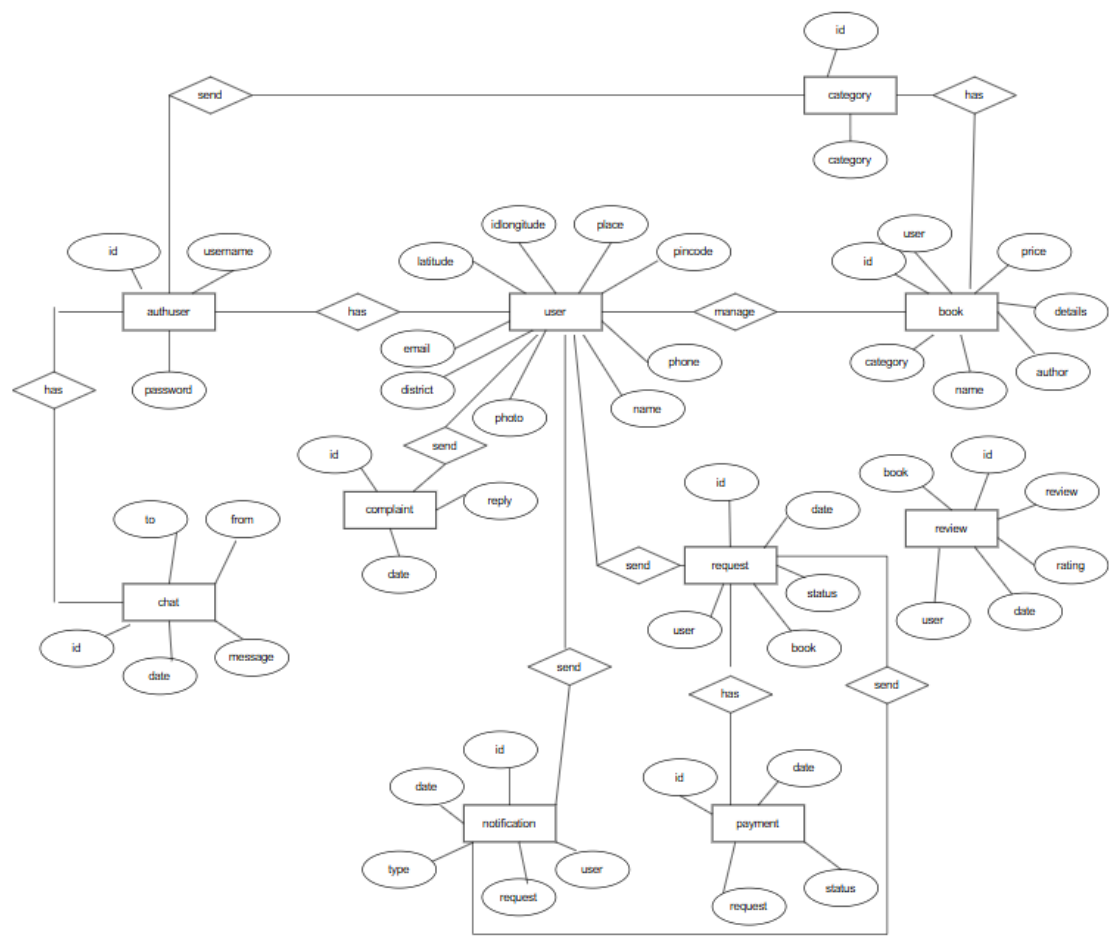


Figure 4.7 : ER Diagram

Figure 4.7 : Shows entity relationships between users, books, authors, and system modules like chat, payment, and review.

Appendix C SourceCode

```
from datetime import datetime

from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.contrib.auth.hashers import check_password, make_password
from django.core.files.storage import FileSystemStorage
from django.http import JsonResponse
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, Group

# Create your views here.

from Myapp.models import category_table, Complaint_table, User_table, Book_table,
review_table, request_table, \
    payment_table, chat_table, rent_table, rent_payment_table, notification_table

def login_get(request):
    return render(request, 'index.html')

def login_post(request):
    username=request.POST['username']
    password=request.POST['password']
    user=authenticate(username=username,password=password)
    if user is not None:
        if user.groups.filter(name='admin').exists():
            login(request,user)
            return redirect('/Myapp/AdminHome/')
        elif user.groups.filter(name='user').exists():
```



```
        login(request, user)
        return redirect('/Myapp/UserHome/')
    else:
        messages.warning(request,'Invalid username or password')
        return redirect('/Myapp/login_get')
    messages.warning(request,'Invalid Username or password exists')
    return redirect('/Myapp/login_get')

@login_required(login_url='/Myapp/login_get')
def editcategory(request,id):
    request.session["did"]=id
    ob=category_table.objects.get(id=id)
    return render(request,'admin/Edit category.html',{'data':ob})

def editcategoryPost(request):
    name = request.POST["textfield"]
    ob = category_table.objects.get(id= request.session["did"])
    ob.categoryname=name
    ob.save()
    return redirect('/myapp/viewcategory')

@login_required(login_url='/Myapp/login_get')
def deletcategory(request,id):
    ob = category_table.objects.get(id=id)
    ob.delete()
    return redirect('/Myapp/viewcategory/#course')

@login_required(login_url='/Myapp/login_get')
def viewusers(request):
    ob=User_table.objects.all()
```

```
    return render(request,'Admin/View users.html',{'data':ob})

@login_required(login_url='/Myapp/login_get')
def viewbooks(request):
    ob=Book_table.objects.all()
    return render(request,'Admin/view books.html',{'data':ob})

@login_required(login_url='/Myapp/login_get')
def viewtransactions(request):
    ob=payment_table.objects.all()
    rent=rent_payment_table.objects.all()
    return render(request,'Admin/View transactions.html',{'data':ob,'rent':rent})

@login_required(login_url='/Myapp/login_get')
def viewuserrating(request):
    ob=review_table.objects.all()
    return render(request,'Admin/viewuserrating.html',{'data':ob})

@login_required(login_url='/Myapp/login_get')

def sendreplypost(request):
    reply=request.POST["reply"]
    ob=Complaint_table.objects.get(id=request.session['rid'])
    ob.reply=reply
    ob.save()
    return redirect('/Myapp/viewcomplaint#course')
```

```
###User###
```

```
@login_required(login_url='/Myapp/login_get')
def UserHome(request):
    user=User_table.objects.get(LOGIN__id=request.user.id)
    ob=Book_table.objects.filter(USER=user.id)
    oba=request_table.objects.filter(status='Accepted')
    oba1=request_table.objects.filter(status='paid')
    l=[]
    for i in oba:
        l.append(i.BOOK.id)
    for i in oba1:
        l.append(i.BOOK.id)
    print(l)
    books=Book_table.objects.exclude(USER=user.id).exclude(id__in=l)
    print(books)
    complaints=Complaint_table.objects.filter(USER=user.id)
    categories = category_table.objects.all()
    return
render(request,'User/user_home.html',{'data':ob,'book':books,'complaints':complaints,'categories':categories})
```

```
def registration_user(request):
    return render(request,'User/Registration.html')
```

```
def RegistrationPost(request):
    name=request.POST['name']
    email=request.POST['email']
    phone=request.POST['phone']
    district=request.POST['district']
    pincode=request.POST['pincode']
    place=request.POST['place']
    longitude=request.POST['longitude']
    latitude=request.POST['latitude']
    photo=request.FILES['photo']
    username=request.POST['username']
    password=request.POST['password']
    fs=FileSystemStorage()
    ps=fs.save(photo.name,photo)
    ob=User_table()
    ob.name=name
    ob.phone=phone
    ob.email=email
```

```
ob.district=district
ob.longitude=longitude
ob.latitude=latitude
ob.photo=ps
ob.pincode=pincode
ob.place=place
user=User.objects.create(username=username,password=make_password(password))
user.save()
user.groups.add(Group.objects.get(name='user'))
ob.LOGIN=user
ob.save()
return redirect('/Myapp/login_get')

@login_required(login_url='/Myapp/login_get')
def View_profile(request):
    ob=User_table.objects.get(LOGIN=request.user.id)
    return render(request,'User/View_profile.html',{'user':ob})

@login_required(login_url='/Myapp/login_get')
def Update_profile(request):
    ob=User_table.objects.get(LOGIN=request.user.id)
    return render(request,'User/Update_profile.html',{'user':ob})

@login_required(login_url='/Myapp/login_get')
def UpdateProfilePost(request):
    name=request.POST['name']
    email=request.POST['email']
    phone=request.POST['phone']
    district=request.POST['district']
    pincode=request.POST['pincode']
    place=request.POST['place']
    longitude=request.POST['longitude']
    latitude=request.POST['latitude']
    ob=User_table.objects.get(LOGIN=request.user.id)
    if 'photo' in request.FILES:
        photo=request.FILES['photo']
        fs=FileSystemStorage()
        ps=fs.save(photo.name,photo)
        ob.photo = ps
        ob.save()
    ob.name=name
    ob.phone=phone
    ob.email=email
    ob.district=district
    ob.longitude=longitude
    ob.latitude=latitude
```

```
ob.pincodes=pincode
ob.place=place
ob.save()
return redirect('/Myapp/View_profile')
```

```
@login_required(login_url='/Myapp/login_get')
def BookPost(request):
    name = request.POST['name']
    author = request.POST['author']
    details = request.POST['details']
    price = request.POST['price']
    category = request.POST['category']
    rent_price = request.POST['rent_price']
    photo = request.FILES['photo']
    fs = FileSystemStorage()
    ps = fs.save(photo.name, photo)
    ob = Book_table()
    ob.USER=User_table.objects.get(LOGIN=request.user.id)
    ob.name = name
    ob.author = author
    ob.details = details
    ob.price = price
    ob.rent_price=rent_price
    ob.photo = ps
    ob.CATEGORY_id = category
    ob.save()
    return redirect('/Myapp/UserHome/#featured-books')
```

```
@login_required(login_url='/Myapp/login_get')
def EditBook(request,id):
    cb=Book_table.objects.get(id=id)
    request.session['eid']=id
    ob=category_table.objects.all()
    return render(request,'User/Edit_book.html',{'data':cb,'category':ob})
```

```
@login_required(login_url='/Myapp/login_get')
def BookEditPost(request):
    name = request.POST['name']
    author = request.POST['author']
    details = request.POST['details']
    price = request.POST['price']
    rent_price = request.POST['rent_price']
```

```

category = request.POST['category']
ob = Book_table.objects.get(id=request.session['eid'])
if 'photo' in request.FILES:
    photo = request.FILES['photo']
    fs = FileSystemStorage()
    ps = fs.save(photo.name, photo)
    ob.photo = ps
    ob.save()
ob.USER=User_table.objects.get(LOGIN=request.user.id)
ob.name = name
ob.author = author
ob.details = details
ob.price = price
ob.rent_price = rent_price
ob.CATEGORY_id = category
ob.save()
return redirect('/Myapp/UserHome/#featured-books')

@login_required(login_url='/Myapp/login_get')
def deleteBook(request,id):
    ob=Book_table.objects.get(id=id)
    ob.delete()
    return redirect('/Myapp/UserHome/#featured-books')

@login_required(login_url='/Myapp/login_get')
def view_request(request):
    ob=request_table.objects.filter(BOOK__USER__LOGIN__id=request.user.id)
    cb=rent_table.objects.filter(BOOK__USER__LOGIN__id=request.user.id)
    print(ob)
    return render(request,'User/view_request.html',{'data':ob,'rent':cb})

@login_required(login_url='/Myapp/login_get')
def view_status(request):
    return render(request,'User/view_request.html')

@login_required(login_url='/Myapp/login_get')
def searchBook(request):
    input=request.POST['search']
    ob=Book_table.objects.filter(Q(name__icontains=input))
    return render(request,'User/user_home.html',{'data':ob})

@login_required(login_url='/Myapp/login_get')
def AddComplaintPost(request):
    complaint=request.POST['complaint']

```

```
ob=Complaint_table()
ob.complaint=complaint
ob.date=datetime.today()
ob.reply='pending'
ob.USER=User_table.objects.get(LOGIN=request.user.id)
ob.save()
return redirect('/Myapp/UserHome/#complaint')

@login_required(login_url='/Myapp/login_get')
def ReviewPost(request,id):
    review=request.POST['review']
    rating=request.POST['rating']
    ob=review_table()
    ob.review=review
    ob.rating=rating
    ob.date=datetime.today()
    ob.USER=User_table.objects.get(LOGIN__id=request.user.id)
    ob.BOOK_id=id
    ob.save()
    return redirect('/Myapp/UserHome/')

@login_required(login_url='/Myapp/login_get')
def RequestBook(request,id):
    ob=request_table()
    ob.USER=User_table.objects.get(LOGIN__id=request.user.id)
    ob.BOOK_id=id
    ob.status='pending'
    ob.date=datetime.today()
    ob.save()
    return redirect('/Myapp/UserHome/')

@login_required(login_url='/Myapp/login_get')
def RequestRentBook(request,id):
    ob=rent_table()
    ob.USER=User_table.objects.get(LOGIN__id=request.user.id)
    ob.BOOK_id=id
    ob.status='pending'
    ob.date=datetime.today()
    ob.save()
    return redirect('/Myapp/UserHome/')

@login_required(login_url='/Myapp/login_get')
def AcceptRequest(request,id):
    ob=request_table.objects.get(id=id)
    ob.status='Accepted'
```

```

    ob.save()
    return redirect('/Myapp/view_request/')

@login_required(login_url='/Myapp/login_get')
def RejectRequest(request,id):
    ob=request_table.objects.get(id=id)
    ob.status='Rejected'
    ob.save()
    ob.delete()
    return redirect('/Myapp/view_request/')

@login_required(login_url='/Myapp/login_get')
def AcceptRentRequest(request,id):
    ob=rent_table.objects.get(id=id)
    ob.status='Accepted'
    ob.save()
    return redirect('/Myapp/view_request/')

@login_required(login_url='/Myapp/login_get')
def RejectRentRequest(request,id):
    ob=rent_table.objects.get(id=id)
    ob.status='Rejected'
    ob.save()
    ob.delete()
    return redirect('/Myapp/view_request/')

@login_required(login_url='/Myapp/login_get')
def raz_pay(request,amount,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 = 200
    amount= float(amount)*100

    # Create a Razorpay order (you need to implement this based on your logic)
    order_data = {
        'amount': amount,
        'currency': 'INR',
        'receipt': 'order_rcptid_11',
        'payment_capture': '1', # Auto-capture payment
    }

```



```

# Create an order
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'],
}

obj = payment_table()
obj.REQUEST_id = id
obj.date = datetime.today()
obj.amount = float(amount/100)
obj.status = 'paid'
obj.save()

request_table.objects.filter(id=id).update(status='paid')

return render(request, 'User/razorpay.html',{ 'razorpay_api_key': razorpay_api_key,
    'amount': order_data['amount'],
    'currency': order_data['currency'],
    'order_id': order['id'], "id":id })

@login_required(login_url='/Myapp/login_get')
def raz_pay_rent(request,amount,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 = 200
    amount= float(amount)*100

    # Create a Razorpay order (you need to implement this based on your logic)
    order_data = {
        'amount': amount,
        'currency': 'INR',
        'receipt': 'order_rcptid_11',
        'payment_capture': '1', # Auto-capture payment
    }

    # Create an order

```

```

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'],
}

obj = rent_payment_table()
obj.RENT_id = id
obj.date = datetime.today()
obj.amount = float(amount/100)
obj.status = 'paid'
obj.save()

rent_table.objects.filter(id=id).update(status='paid')

return render(request, 'User/razorpay.html',{ 'razorpay_api_key': razorpay_api_key,
    'amount': order_data['amount'],
    'currency': order_data['currency'],
    'order_id': order['id'], "id":id})

@login_required(login_url='/Myapp/login_get')
def chat_send(request, msg):
    lid = request.user.id
    toid = request.session["userid"]
    message = msg

    import datetime
    d = datetime.datetime.now().date()
    chatobt = chat_table()
    chatobt.message = message
    chatobt.TOID_id = toid
    chatobt.FROMID_id = lid
    chatobt.date = d
    chatobt.save()

    return JsonResponse({"status": "ok"})

@login_required(login_url='/Myapp/login_get')
def User_sendchat(request):
    FROM_id=request.POST['from_id']

```

```

    TOID_id=request.POST['to_id']
    msg=request.POST['message']

    from datetime import datetime
    c=chat_table()
    c.FROMID_id=FROM_id
    c.TOID_id=TOID_id
    c.message=msg
    c.date=datetime.now()
    c.save()
    return JsonResponse({'status':"ok"})

@login_required(login_url='/Myapp/login_get')
def User_viewchat(request):
    fromid = request.POST["from_id"]
    toid = request.POST["to_id"]
    # lmid = request.POST["lastmsgid"]
    from django.db.models import Q

    res = chat_table.objects.filter(Q(FROMID_id=fromid, TOID_id=toid) |
    Q(FROMID_id=toid, TOID_id=fromid)).order_by('id')
    l = []

    for i in res:
        l.append({"id": i.id, "msg": i.message, "from": i.FROMID_id, "date": i.date, "to":
        i.TOID_id})

    return JsonResponse({"status":"ok",'data':l})

@login_required(login_url='/Myapp/login_get')
def ViewNotification(request):

    ob=notification_table.objects.filter(RENT__USER=User_table.objects.get(LOGIN__id=request.user.id))
    return render(request,'User/view notifications.html',{'data':ob})

@login_required(login_url='/Myapp/login_get')
def SendNotificationPost(request,id):
    notification=request.POST['notification']
    ob=notification_table()
    ob.notification=notification
    ob.date=datetime.today()
    ob.RENT_id=id

```

```
ob.USER=User_table.objects.get(LOGIN__id=request.user.id)
ob.save()
return redirect('/Myapp/view_request/')

def ViewPayment(request):
    current_user = User_table.objects.get(LOGIN__id=request.user.id)

    # Payments made by the user (My Payments)
    sell = payment_table.objects.filter(REQUEST__USER=current_user)
    rent = rent_payment_table.objects.filter(RENT__USER=current_user)

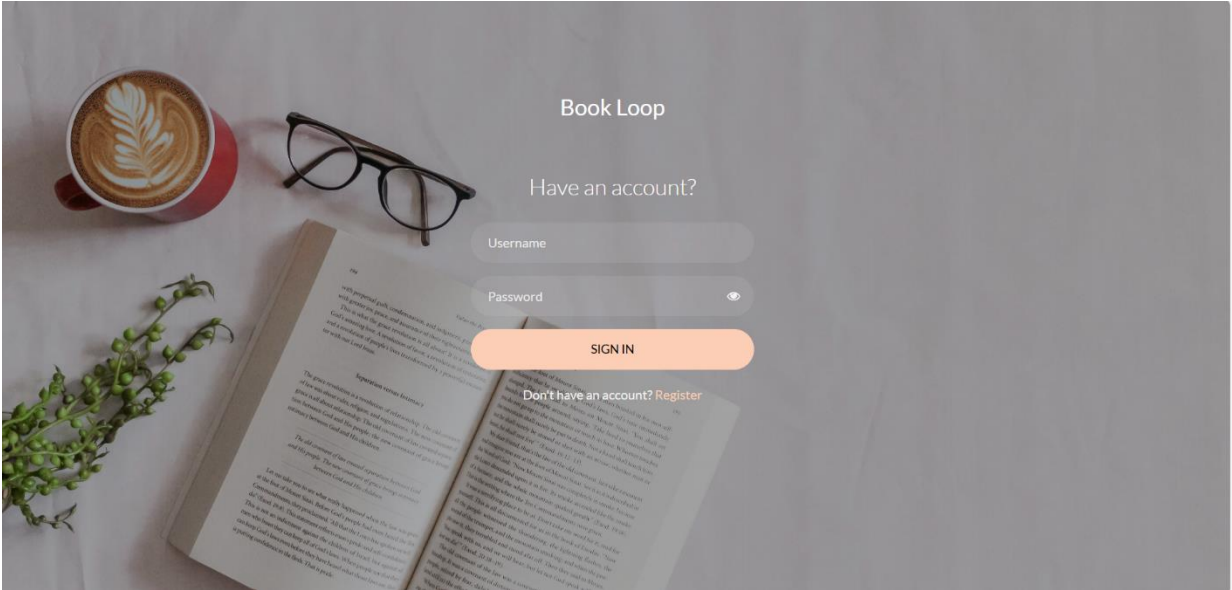
    # Payments received by the user (owner of the book)
    sell_received = payment_table.objects.filter(REQUEST__BOOK__USER=current_user)
    rent_received =
rent_payment_table.objects.filter(RENT__BOOK__USER=current_user)

    return render(request, 'User/View_payment.html', {
        'my_payments': sell,
        'my_rent_payments': rent,
        'received_payments': sell_received,
        'received_rent_payments': rent_received
    })

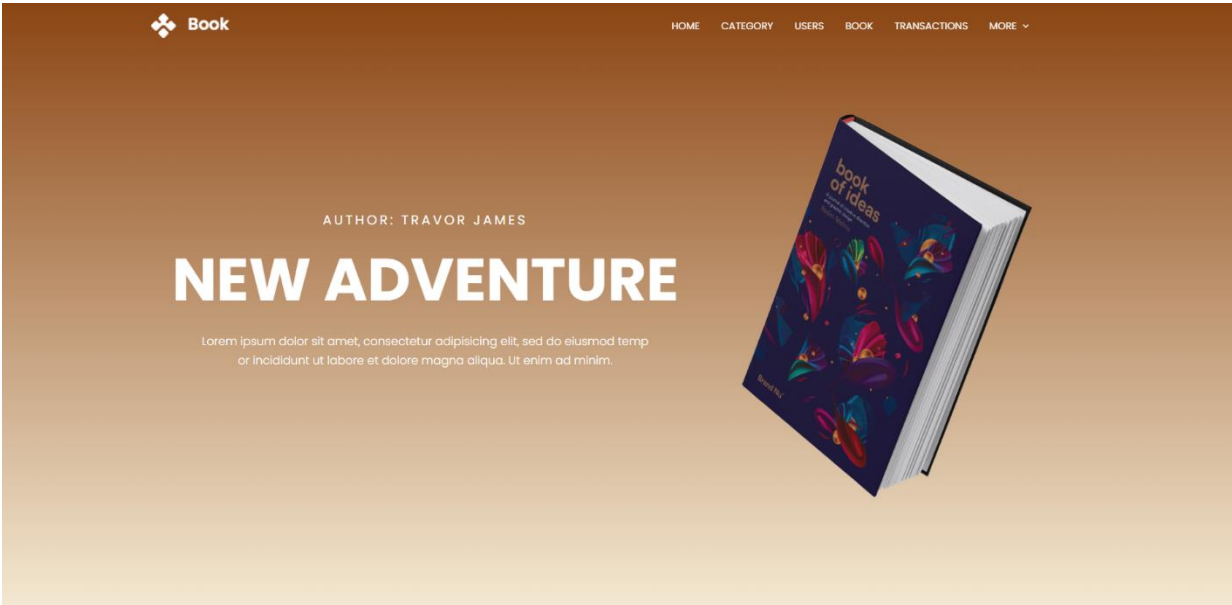
def Logout(request):
    logout(request)
    return redirect('/Myapp/login_get')
```

Appendix D Screenshot

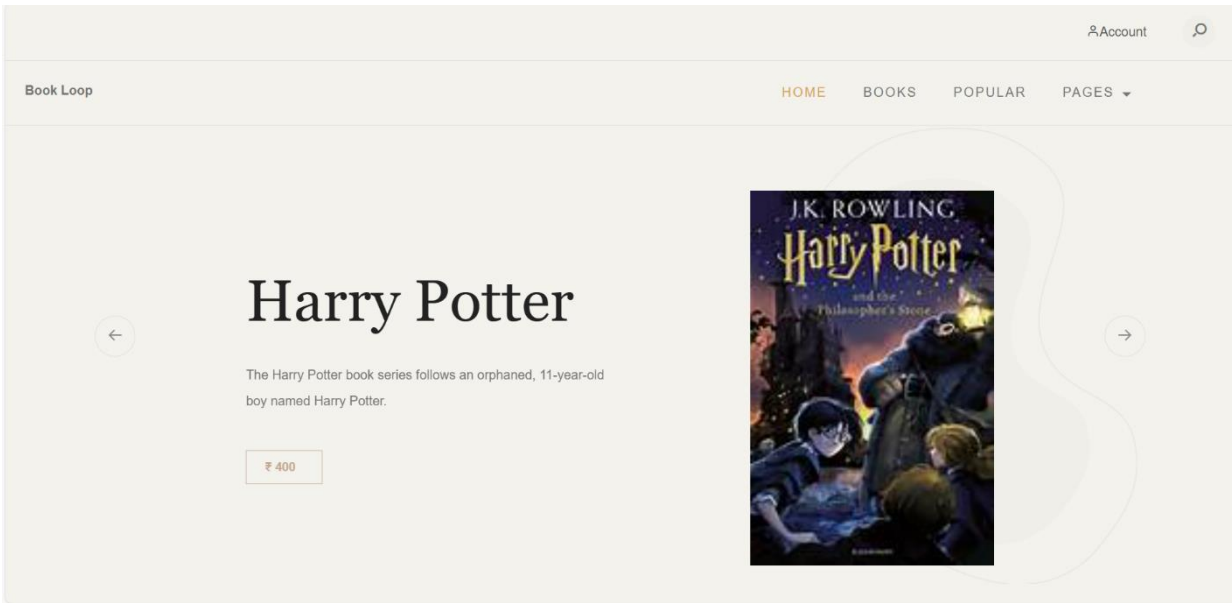
LOGIN PAGE



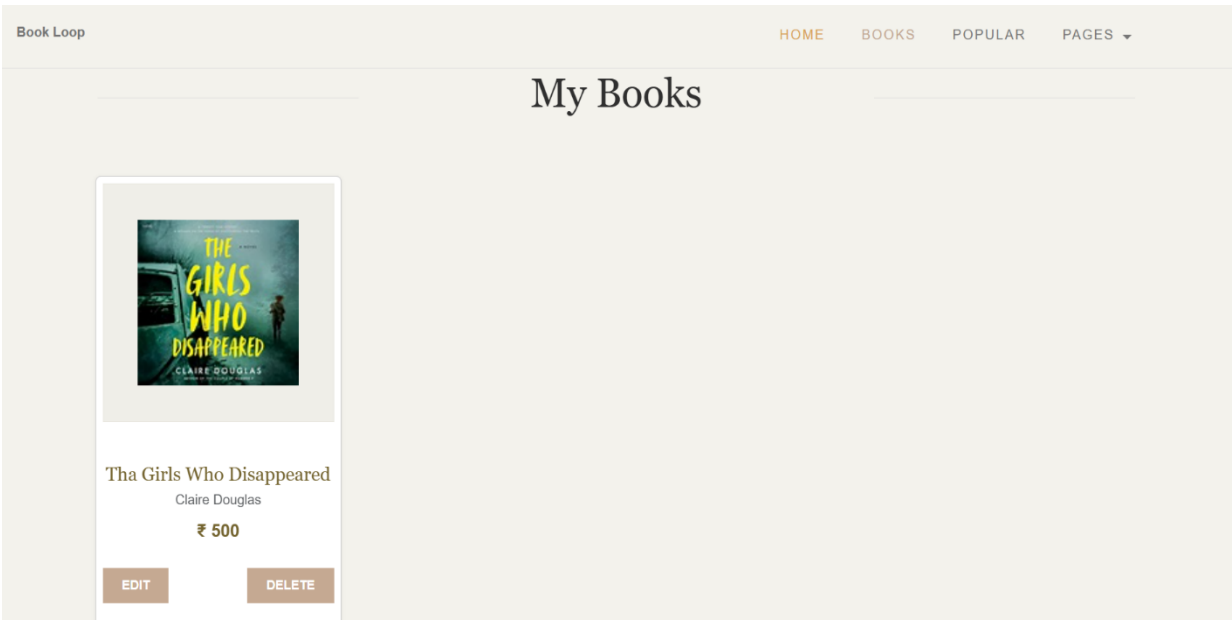
HOME PAGE





HOME PAGE



ADD BOOK



PAYMENT

	<h2>My Payments</h2>
<p>Believe in yourself</p> <p>Type: Purchase</p> <p>Date: Oct. 6, 2025</p> <p>Amount: ₹600</p>	<p>paid</p>
<p>Harry Potter</p> <p>Type: Rent</p> <p>Date: Oct. 7, 2025</p> <p>Amount: ₹100</p>	<p>paid</p>
<p>Harry Potter</p> <p>Type: Rent</p> <p>Date: Oct. 7, 2025</p> <p>Amount: ₹100</p>	<p>paid</p>
<p>Tha Girls Who Disappeared</p> <p>Type: Rent</p> <p>Date: Oct. 7, 2025</p> <p>Amount: ₹150</p>	<p>paid</p>
<p>Tha Girls Who Disappeared</p> <p>Type: Rent</p> <p>Date: Oct. 7, 2025</p> <p>Amount: ₹150</p>	<p>paid</p>
	<h2>Payments Received</h2>
<p>No payments received yet.</p>	

REQUEST

Purchase Requests	
<div><div><div>📖 Harry Potter</div><div><div>Date:</div>Sept. 23, 2025</div><div><div>Status:</div>Accepted</div><div><div>Requested By:</div>Sana</div></div><div>Chat</div></div>	
Rent Requests	
<div><div><div>📖 Harry Potter</div><div><div>Date:</div>Oct. 7, 2025</div><div><div>Status:</div>paid</div><div><div>Requested By:</div>zayan</div></div><div><div>Approved Rent</div><div>Send Notification</div></div></div>	