

# **BookShelf**

## **eBook Reading Platform**

*Mini Project Report*

*Submitted by*

**Midhun Krishnan M**

**Reg. No.: AJC19MCA-I040**

*In Partial fulfillment for the Award of the Degree of*

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

**(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2023-2024**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**AMAL JYOTHI COLLEGE OF ENGINEERING**  
**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**BookShelf**” is the bona fide work of **Midhun Krishnan M (Regno: AJC19MCA-I040)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

**Ms. Ankitha Philip**

**Internal Guide**

**Ms. Meera Rose Mathew**

**Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

## **DECLARATION**

I hereby declare that the project report “**BookShelf**” is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

**Date: 30/10/2023**

**MIDHUN KRISHNAN M**

**KANJIRAPPALLY**

**Reg: AJC19MCA-I040**

## ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Ankitha Philip** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

MIDHUN KRISHNAN M

## **ABSTRACT**

This project is a dedicated endeavor to create a multifaceted online platform that serves both readers and aspiring authors as an eBook store. In the eBook store section, readers can explore a curated collection of books with dedicated pages providing comprehensive details. Users can personalize their experience, and a convenient cart feature is available for easy book purchases.

Author profiles and browsing history enhance the reading experience, fostering a sense of connection between readers and writers. The project's primary goal is to create a comprehensive and user-friendly platform that caters to the diverse needs of readers, supports aspiring authors, and cultivates a vibrant literary community. By seamlessly integrating a library and an ebook store while offering personalization options and valuable author insights, this platform aims to provide a rich and immersive reading environment that caters to the literary passions and preferences of its users. Thus, fostering a community of literary enthusiasts.

# CONTENT

SL. NO	TOPIC	PAGE NO
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>1.1</b>	<b>PROJECT OVERVIEW</b>	<b>2</b>
<b>1.2</b>	<b>PROJECT SPECIFICATION</b>	<b>2</b>
<b>2</b>	<b>SYSTEM STUDY</b>	<b>4</b>
<b>2.1</b>	<b>INTRODUCTION</b>	<b>5</b>
<b>2.2</b>	<b>EXISTING SYSTEM</b>	<b>5</b>
<b>2.3</b>	<b>DRAWBACKS OF EXISTING SYSTEM</b>	<b>6</b>
<b>2.4</b>	<b>PROPOSED SYSTEM</b>	<b>7</b>
<b>2.5</b>	<b>ADVANTAGES OF PROPOSED SYSTEM</b>	<b>7</b>
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>9</b>
<b>3.1</b>	<b>FEASIBILITY STUDY</b>	<b>10</b>
<b>3.1.1</b>	<b>ECONOMICAL FEASIBILITY</b>	<b>10</b>
<b>3.1.2</b>	<b>TECHNICAL FEASIBILITY</b>	<b>11</b>
<b>3.1.3</b>	<b>BEHAVIORAL FEASIBILITY</b>	<b>11</b>
<b>3.1.4</b>	<b>FEASIBILITY STUDY QUESTIONNAIRE</b>	<b>11</b>
<b>3.2</b>	<b>SYSTEM SPECIFICATION</b>	<b>15</b>
<b>3.2.1</b>	<b>HARDWARE SPECIFICATION</b>	<b>15</b>
<b>3.2.2</b>	<b>SOFTWARE SPECIFICATION</b>	<b>15</b>
<b>3.3</b>	<b>SOFTWARE DESCRIPTION</b>	<b>15</b>
<b>3.3.1</b>	<b>DJANGO</b>	<b>15</b>
<b>3.3.2</b>	<b>SQLite</b>	<b>16</b>
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>17</b>
<b>4.1</b>	<b>INTRODUCTION</b>	<b>18</b>
<b>4.2</b>	<b>UML DIAGRAM</b>	<b>18</b>
<b>4.2.1</b>	<b>USE CASE DIAGRAM</b>	<b>19</b>
<b>4.2.2</b>	<b>SEQUENCE DIAGRAM</b>	<b>21</b>
<b>4.2.3</b>	<b>STATE CHART DIAGRAM</b>	<b>23</b>
<b>4.2.4</b>	<b>ACTIVITY DIAGRAM</b>	<b>24</b>
<b>4.2.5</b>	<b>CLASS DIAGRAM</b>	<b>26</b>
<b>4.2.6</b>	<b>OBJECT DIAGRAM</b>	<b>28</b>
<b>4.2.7</b>	<b>COMPONENT DIAGRAM</b>	<b>29</b>

<b>4.2.8</b>	<b>DEPLOYMENT DIAGRAM</b>	<b>30</b>
<b>4.3</b>	<b>USER INTERFACE DESIGN USING FIGMA</b>	<b>31</b>
<b>4.4</b>	<b>DATABASE DESIGN</b>	<b>34</b>
<b>5</b>	<b>SYSTEM TESTING</b>	<b>42</b>
<b>5.1</b>	<b>INTRODUCTION</b>	<b>43</b>
<b>5.2</b>	<b>TEST PLAN</b>	<b>43</b>
<b>5.2.1</b>	<b>UNIT TESTING</b>	<b>44</b>
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	<b>44</b>
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	<b>45</b>
<b>5.2.4</b>	<b>USER ACCEPTANCE TESTING</b>	<b>45</b>
<b>5.2.5</b>	<b>AUTOMATION TESTING</b>	<b>45</b>
<b>5.2.6</b>	<b>SELENIUM TESTING</b>	<b>46</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>59</b>
<b>6.1</b>	<b>INTRODUCTION</b>	<b>60</b>
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	<b>60</b>
<b>6.2.1</b>	<b>USER TRAINING</b>	<b>60</b>
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	<b>61</b>
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	<b>61</b>
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>62</b>
<b>7.1</b>	<b>CONCLUSION</b>	<b>63</b>
<b>7.2</b>	<b>FUTURE SCOPE</b>	<b>63</b>
<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>64</b>
<b>9</b>	<b>APPENDIX</b>	<b>66</b>
<b>9.1</b>	<b>SAMPLE CODE</b>	<b>67</b>
<b>9.2</b>	<b>SCREEN SHOTS</b>	<b>77</b>

## **List of Abbreviation**

- UML - Unified Modelling Language
- ORM - Object-Relational Mapping
- MVT - Model-View-Template
- MVC - Model-View-Controller
- RDBMS - Relational Database Management System
- 1NF - First Normal Form
- 2NF - Second Normal Form
- 3NF - Third Normal Form
- IDE - Integrated Development Environment
- HTML - HyperText Markup Language
- JS - JavaScript
- CSS - Cascading Style Sheets
- AJAX - Asynchronous JavaScript and XML
- JSON - JavaScript Object Notation
- API - Application Programming Interface
- UI - User Interface
- HTTP - Hypertext Transfer Protocol
- URL - Uniform Resource Locator
- PK - Primary Key
- FK - Foreign Key
- SQL - Structured Query Language
- CRUD - Create, Read, Update, Delete



# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 PROJECT OVERVIEW

This mini-project entails two primary user roles: Readers and Administrators. Readers are provided with functionalities such as registration, login, and personal information management. They can explore a diverse catalog of books, including a popular books carousel and categorized browsing. Detailed book pages offer comprehensive information, and users can access book previews at no cost. A cart feature facilitates book selection, and author profiles enhance the reading experience. Rating and review options enable user feedback.

Administrators, on the other hand, possess the authority to add, delete, and edit books within the website. User management and author profile management are also under their purview. This mini-project combines user-centric features for readers and administrative control for efficient content management, promoting an enriched reading experience.

## 1.2 PROJECT SPECIFICATION

### User-View

#### 1. Homepage:

- Upon accessing the platform, users are greeted with a visually appealing and user-friendly homepage.
- The homepage features a carousel of book covers highlighting featured books, enticing users to explore further.
- Below the carousel, there are personalized book recommendations based on the user's reading history and preferences.
- A "Popular Now" section showcases trending books, enticing users to discover popular titles.

#### 2. Browsing and Searching:

- Users can easily navigate the eBook store by browsing books based on categories, genres, and new releases.
- The search bar at the top allows users to find specific books, authors, or genres with ease.
- As users start typing their search query, the platform provides real-time suggestions to assist them in finding relevant content.

#### 3. Book Details Page:

- Clicking on a book cover or title opens the book's dedicated page, providing in-depth details about the book.

- The book details page displays the book's cover, a comprehensive description, the author's bio, and reader reviews (if available).
- For books available for purchase, users can view the price and add the book to their shopping cart directly from this page.

#### **4. User Account:**

- Users can create an account easily by providing necessary details or log in with existing credentials.
- Upon login, users are directed to their personalized account page, where they can manage their preferences, themes, and profile information.
- They can also view their activity history, such as recently viewed books and purchased items.

#### **5. Shopping Cart:**

- Users can view the contents of their shopping cart, which includes books they have added for purchase.
- The cart shows the book details, including title, author, and price, for easy review before proceeding to checkout.

#### **6. Author Profiles:**

- The platform showcases author profiles, offering insights into their works, background, and achievements.

### **Admin-View**

#### **1. Admin Dashboard:**

- The admin dashboard serves as the command center for platform management and content curation.
- It provides an overview of essential statistics, such as the number of registered users, book uploads, and book sales.

#### **2. Book Management:**

- Admins can easily add new books to the eBook store, ensuring a diverse and up-to-date collection.
- They have the authority to edit book details, including titles, authors, descriptions, and prices.
- If necessary, admins can remove books that violate copyright or platform guidelines.

#### **3. User Management:**

- The admin view allows admins to manage user accounts, review user-generated content, and handle support requests.

## **CHAPTER 2**

### **SYSTEM STUDY**

## **2.1 INTRODUCTION**

A critical stage in the creation of any system is system analysis. Its main objective is to collect and examine data in order to identify issues and provide solutions. The key to this phase is effective communication between system users and developers. In fact, a system analysis should always be the first step in every system development project.

Here, the system analyst assumes the role of an investigator, carefully evaluating the effectiveness of the current system. This requires determining the system's inputs and outputs as well as the relationship between its activities and the outcomes of the organization.

Information is gathered through a variety of methods, including surveys and interviews. The broad objectives include learning how the system works, identifying problem areas, and suggesting solutions to deal with the problems facing the company.

## **2.2 EXISTING SYSTEM**

### **2.2.1 NATURAL SYSTEM STUDIED**

The existing natural system is an offline brick-and-mortar bookstore characterized by physical book inventories organized by genres and themes, manual point-of-sale transactions, personalized customer interactions, community engagement through literary events, a loyalty program for customer retention, and a focus on data security and inclusivity. As technology trends evolve, the system is exploring potential digital expansion, such as an online platform for e-books and e-commerce, while maintaining the cherished traditional bookstore experience and its commitment to diverse literature offerings and customer satisfaction.

### **2.2.2 DESIGNED SYSTEM STUDIED**

The designed system under study is an online platform for a common library, aiming for simplicity and accessibility. It features a digital catalog of books and resources with basic categorization. Transactions are conducted through an automated digital interface for borrowing and returning items. While personalized customer interactions are available through user accounts, the emphasis is on user-friendly navigation.

The online library promotes inclusivity by offering a wide range of digital literature, accessible to anyone with an internet connection. It doesn't have extensive loyalty programs or community events but focuses on providing easy access to reading materials. Data security measures are in

place to protect user information and ensure a safe online experience.

This designed system acknowledges the importance of digital expansion in a tech-savvy world but maintains a straightforward approach to make literature readily available to a broader audience, aligning with the convenience of digital platforms.

## **2.3 DRAWBACKS OF EXISTING SYSTEM**

1. **Limited Accessibility:** Physical bookstores have a restricted reach, limiting access to books for customers who may not live near a bookstore. This can lead to reduced foot traffic and sales.
2. **Limited Inventory:** Offline bookstores can only stock a finite number of books due to space constraints, resulting in a limited selection for customers. It can be challenging to find niche or less-popular titles.
3. **Inconvenience:** Customers must physically visit the store to purchase books, which can be inconvenient, especially for those with busy schedules or mobility issues.
4. **Inefficient Search:** Locating specific books or genres in a physical store can be time-consuming, and customers may need assistance from store staff.
5. **Lack of Customer Data:** Offline bookstores have limited means to collect and analyze customer data, missing out on valuable insights that could enhance the shopping experience and tailor recommendations.
6. **Limited Interaction:** Offline bookstores may lack community features, like book reviews, ratings, or author engagement, which are common on online platforms.
7. **High Operational Costs:** Operating a physical store entails substantial costs for rent, utilities, and staffing. These expenses can impact the store's profitability and sustainability.
8. **Seasonal Trends:** Physical bookstores may struggle to adapt to changing reading habits or seasonal demand fluctuations.
9. **Competition from Online Retailers:** The rise of online book retailers offers customers the convenience of shopping from home, often with a more extensive inventory and competitive prices, posing a significant challenge to offline bookstores.
10. **Environmental Concerns:** Offline bookstores involve the use of paper, contributing to environmental concerns related to deforestation and waste.

## 2.4 PROPOSED SYSTEM

The proposed system is an online book marketplace and reader platform with a range of features. For readers, it offers registration and login, a user-friendly home page, session management, profile editing, a carousel of popular books, browsing and searching capabilities, book categorization, detailed book pages with author information, free book previews, a shopping cart, author profiles, user-driven book ratings and reviews, and the exciting addition of a writer's corner.

In this writer's corner, readers can explore their creative side by trying out writing and related functionalities. They can write and share their stories, connect with fellow writers and readers, and receive feedback. This feature aims to foster a vibrant and interactive writing community within the platform.

For the admin, it provides book management features, including the ability to add, edit, and delete books, user management for account control, oversight of author profiles, and monitoring of the writer's corner activities. This web-based system will enhance the interaction between readers and books while providing the admin with tools for content and user management. Future enhancements may include e-book reading capabilities and integrated payment options for book purchases, further enriching the overall reading and writing experience.

This expanded system encourages creativity and engagement, bridging the gap between readers and aspiring writers within the online book marketplace.

## 2.5 ADVANTAGES OF PROPOSED SYSTEM

1. Convenience for Readers: Users can easily browse, search, and purchase books online, providing them with a convenient way to access a wide variety of reading materials.
2. Enhanced User Experience: The user-friendly interface, book categorization, and book previews improve the overall user experience, making it more enjoyable for readers to discover and engage with books.
3. Author Interaction: Readers can explore author profiles, creating a sense of connection and engagement with the creators of the books they love.
4. User Empowerment: Features like book ratings and reviews allow users to contribute their opinions, helping others make informed decisions about book choices.
5. Admin Control: Admin features enable efficient book and user management, ensuring the platform's integrity and security.

6. Scalability: The system can be expanded to include additional features such as e-book reading and integrated payment options, making it adaptable to evolving user needs and industry trends.
7. Accessibility: Being web-based, the system can be accessed from various devices, making it accessible to a broad audience.
8. Marketing Opportunities: Popular book carousels and categorization offer marketing potential for promoting specific books and authors.
9. Data-Driven Decision Making: The system can provide valuable insights into user preferences and behavior through user reviews, ratings, and browsing history, facilitating data-driven decision-making for future improvements.
10. Revenue Generation: The potential to integrate payment gateways can lead to revenue generation through book sales.



## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

### **3.1 FEASIBILITY STUDY**

The primary purpose of conducting a feasibility study is to comprehensively assess whether the proposed project can successfully meet the organization's objectives in terms of available resources, labor, and time. This crucial study allows the developers and decision-makers to gain valuable insights into the project's potential viability and prospects. By carefully examining various aspects of the proposed system, such as its impact on the organization, its ability to fulfill user requirements, and the optimal utilization of resources, a feasibility study helps in determining the project's feasibility and potential success.

The assessment of the proposed project's feasibility involves multiple dimensions, each playing a critical role in the decision-making process.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

A well-conducted feasibility study provides valuable insights to decision-makers, allowing them to make informed judgments about the project's potential success. It assists in identifying potential risks, challenges, and opportunities associated with the proposed endeavor, enabling stakeholders to devise effective mitigation strategies.

#### **3.1.1 Economical Feasibility**

As a student project, the economic feasibility study for BookShelf takes into account limited resources and a focus on learning outcomes. The market analysis explores the preferences of the target audience while being mindful of the project's scope and objectives. Cost estimation is based on realistic assumptions, considering the available funding for software development, infrastructure, and marketing efforts.

While risks are identified, the emphasis is on learning to manage challenges rather than expecting high financial returns. Funding options might include self-funding or leveraging university resources.

The financial projections reflect the project's limited duration and scope, aligning with the constraints of a student project. In conclusion, the economic feasibility study aims to provide valuable insights for the student team to make informed decisions while prioritizing the educational aspect of the project.

### **3.1.2 Technical Feasibility**

The technical feasibility study for BookShelf, as a student project, aims to assess whether the proposed platform can be practically developed and implemented with the available technical resources and skills. It evaluates the team's technical expertise, the availability of required technologies, and infrastructure needs. The study considers integration possibilities, scalability, security measures, development timeline, technical support availability, and testing processes. By examining these aspects, the study determines the student team's capability to build a stable and functional BookShelf platform within the project's limitations.

### **3.1.3 Behavioral Feasibility**

Assessing the behavioral feasibility of BookShelf is a critical component of our feasibility study, which involves evaluating how well the system aligns with users' needs and behaviors. To undertake this assessment effectively, we establish a robust outline of system requirements, encompassing inputs, outputs, programs, and procedures. This serves as the foundational framework for our evaluation. We also emphasize the importance of identifying the necessary equipment and resources to support the system. Once the system's design is finalized, we ensure various pathways for its seamless operation and a positive user experience.

### **3.1.4 Feasibility Study Questionnaire**

#### **1) Project Overview?**

The project aims to develop an online platform called "BookShelf" that serves as a local library and ebook store. It offers various features for readers and aspiring authors, allowing users to upload and view their EPUB files in the local library and browse books added by administrators in the ebook store. The platform includes personalized user accounts, author profiles, voting for favorite books, and a review moderation feature. Its goal is to create a comprehensive and user-friendly platform for readers to access a wide range of books, support aspiring authors, and foster a community of literary enthusiasts.

#### **2) To what extent the system is proposed for?**

The proposed system, "BookShelf," aims to be a comprehensive online platform serving as a local library and ebook store for readers and aspiring authors. It offers personalized accounts, author profiles, voting, and community engagement, fostering a user-friendly environment. While starting as a student project, it has potential for future scalability.

#### **3) Specify the Viewers/Public which is to be involved in the System?**

Literary Enthusiasts, Amateur Authors, Publishers

#### **4) List the modules in your system**

- **User Management Module:** This module handles user registration, login, and account management functionalities, allowing users to customize their profiles and preferences.
- **eBook Store Module:** This module contains the collection of books available for purchase. It provides browsing, searching, and filtering options for readers to explore and find books of interest.
- **Local Library Module:** This module enables users to upload and manage their own EPUB files in their personal local library for easy access.
- **Author Submissions Module:** Aspiring authors can submit their original books through this module, allowing administrators to review, approve, and manage the addition of new books.
- **Voting and Ranking Module:** Users can vote for their favorite books, and an author ranking system is implemented based on cumulative votes received for all books.
- **Review and Moderation Module:** Users can read and write book reviews, and this module facilitates the moderation and approval of user-submitted reviews.
- **User Activity Module:** This module tracks and displays users' browsing history, wishlist, and purchase history for their reference.
- **Personalization Module:** Book recommendations and theme customization features are included to personalize the user experience.
- **Admin Dashboard Module:** Administrators can manage book inventory, user accounts, author submissions, and review moderation through the admin dashboard.
- **Email Notification Module:** This module sends email notifications to users for order confirmations, receipts, and other important updates.
- **Search and Filter Module:** Users can search for books based on various criteria, such as author, ISBN, publication date, category, and more.
- **Community Engagement Module:** Features like book voting, author profiles, and interaction with other users create a sense of community among readers and aspiring authors.

#### **5) Identify the users in your project?**

The viewers/public involved in the BookShelf system include:

- **Readers:** Individuals who access the platform to browse, search, and purchase books from the ebook store. They can create personalized accounts, manage preferences, and interact with book reviews and voting.
- **Aspiring Authors:** Writers who have the opportunity to submit their original books

to the platform. They can showcase their works, receive votes, and gather feedback from readers.

- **Administrators:** Staff members responsible for managing the platform, adding new books, moderating reviews, and overseeing user accounts and submissions.
- **General Public:** Individuals who visit the platform to explore available books, author profiles, and community engagement features without the need for user accounts or submissions.

**6) Who owns the system?**

Administrator

**7) System is related to which firm/industry/organization?**

The system is related to the book industry and is designed to serve readers, aspiring authors, and book enthusiasts.

**8) Details of person that you have contacted for data collection?**

Sharon, Avid Reader

**9) Questionnaire to collect details about the project? (Min 10 questions, include descriptive answers, attach additional docs (e.g., Bill receipts), if any?)**

- **Are there any significant costs associated with developing BookShelf as part of the project work?**

No, the proposed system is developed as part of project work, and there are no manual costs to spend on the platform.

- **What is the cost of hardware and software required for BookShelf?**

All the necessary resources, including hardware and software, are already available, making it a cost-effective student project.

- **Are there any additional costs for operational expenses, such as maintenance or server hosting?**

No, operational expenses are minimal, as the project is designed to be run with existing resources and within the scope of the student project.

- **Is the project feasible within the limits of current technology?**

Yes, the project is feasible within the limits of current technology, as it involves the development of a relatively straightforward online book platform.

- **Technical issues raised during the investigation are:**

Nothing, the investigation did not uncover any major technical issues that would hinder the development of BookShelf.

- **Can the technology be easily applied to current problems?**

Yes, the technology can be easily applied to current problems, as the platform aims to provide an accessible and user-friendly interface for book enthusiasts.

- **Does the technology have the capacity to handle the solution?**

Yes, the technology has the capacity to handle the solution, considering the small-scale nature of the student project and its focus on limited user interactions.

- **Is the required technology readily available and accessible to the student team for developing BookShelf?**

Yes, the required technology is readily available and accessible, and the student team has access to the necessary development tools and resources.

- **Does the student team possess the necessary technical skills and knowledge to design and develop the platform effectively?**

Yes, the student team possesses the necessary technical skills and knowledge to effectively design and develop BookShelf, keeping in mind the project's scope.

- **Are the infrastructure requirements for BookShelf, such as servers and hosting services, feasible and within the project's scope?**

Yes, the infrastructure requirements for BookShelf are feasible and within the project's scope, considering the small-scale nature of the platform.

- **Will users receive adequate support while using BookShelf?**

Absolutely, BookShelf is committed to providing users with comprehensive support, ensuring a seamless and positive user experience.

- **Will users be exposed to any harmful elements or content while using BookShelf?**

Rest assured, BookShelf has been meticulously designed to maintain a safe and secure environment, free from harmful elements, providing users with a worry-free space to explore and read.

- **Does BookShelf offer user-friendly features and an intuitive interface?**

Yes, BookShelf takes pride in its user-centric approach, offering a user-friendly interface with intuitive features, ensuring that users can effortlessly navigate and access their favorite books.

- **Is there a mechanism in place for users to share their thoughts and suggestions?**

Absolutely, BookShelf actively encourages user feedback and suggestions. We believe in fostering a vibrant community where users can readily share their thoughts, ensuring their voices are heard and valued.

## 3.2 SYSTEM SPECIFICATION

### 3.2.1 Hardware Specification

Processor - i5  
RAM - 16 GB  
Hard Disk - 1 TB

### 3.2.2 Software Specification

Front End - HTML, JS, CSS, jQuery, Ajax, Bootstrap  
Back End - Python-Django  
Database - Sqlite  
Client on PC - Windows 7 and above.  
Technologies used - JS, HTML5, AJAX, jQuery, CSS

## 3.3 SOFTWARE DESCRIPTION

### 3.3.1 Django

Django is a popular and powerful open-source web framework written in Python, designed to facilitate rapid development and maintainable web applications. It follows the Model-View-Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern. Django provides a structured and efficient way to build web applications, offering several key components and features.

At its core, Django includes a robust Object-Relational Mapping (ORM) system that simplifies database interactions, allowing developers to work with Python objects instead of raw SQL queries. It also includes a URL dispatcher for mapping URLs to view functions, an automatic admin interface for managing application data, and a templating engine for creating dynamic and reusable user interfaces.

Django places a strong emphasis on security, with built-in features to protect against common web vulnerabilities. It offers authentication and authorization systems, middleware support for global request and response processing, and compatibility with various databases.

The framework's scalability, extensibility, and a vibrant community of developers make it a popular choice for building web applications, from simple websites to complex, high-traffic platforms. Django's extensive documentation and ecosystem of third-party packages further

enhance its appeal for web developers.

### **3.3.2 Sqlite**

SQLite is a self-contained and serverless relational database management system (RDBMS) known for its simplicity and efficiency. It stores the entire database in a single file, eliminating the need for a separate server process, which simplifies deployment. This makes SQLite an ideal choice for embedded systems, mobile applications, and small to medium-sized projects. Developers interact with the database directly through function calls, making it an embedded database well-suited for various applications, including mobile apps, desktop software, and web applications.

Despite its lightweight nature, SQLite is ACID-compliant, ensuring data integrity with support for transactions. It offers a wide range of data types, and its compatibility with multiple programming languages, including Python, C/C++, and Java, makes it accessible to a diverse developer community. SQLite is open-source, free, and known for its speed and efficiency, particularly for read-heavy workloads. While not intended for extremely high-concurrency or large-scale applications, SQLite excels in scenarios where simplicity, portability, and low resource consumption are key requirements.

It is commonly used as a local data store, cache, or embedded database within larger applications, contributing to its versatility and widespread adoption in software development.



## **CHAPTER 4**

### **SYSTEM DESIGN**

## 4.1 INTRODUCTION

The initial stage of developing any engineered product or system is the design phase, which involves a creative approach. A well-crafted design plays a critical role in ensuring the successful functioning of a system. Design is defined as the process of employing various techniques and principles to define a process or system in enough detail to enable its physical realization. This involves using different methods to describe a machine or system, explaining how it operates, in sufficient detail for its creation. In software development, design is a crucial step that is always present, regardless of the development approach. System design involves creating a blueprint for building a machine or product. Careful software design is essential to ensure optimal performance and accuracy. During the design phase, the focus shifts from the user to the programmers or those working with the database. The process of creating a system typically involves two key steps: Logical Design and Physical Design.

## 4.2 UML DIAGRAM

UML, which stands for Unified Modeling Language, is a standardized language used for specifying, visualizing, constructing, and documenting the elements of software systems. The Object Management Group (OMG) is responsible for the creation of UML, with the initial draft of the UML 1.0 specification presented to OMG in January 1997. Unlike common programming languages such as C++, Java, or COBOL, UML is not a programming language itself. Instead, it is a graphical language that serves as a tool for creating software blueprints.

UML is a versatile and general-purpose visual modeling language that facilitates the visualization, specification, construction, and documentation of software systems. While its primary use is in modeling software systems, UML's applications are not limited to this domain. It can also be employed to represent and understand processes in various contexts, including non-software scenarios like manufacturing unit processes.

It's important to note that UML is not a programming language, but it can be utilized with tools that generate code in different programming languages based on UML diagrams. UML encompasses nine core diagrams that aid in representing various aspects of a system.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram

- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

#### **4.2.1 USE CASE DIAGRAM**

A use case is a tool for understanding a system's requirements and organizing them, especially in the context of creating or using something like a product delivery website. These tools are represented using "use case" diagrams within the Unified Modeling Language, a standardized way of creating models for real-world things and systems.

A use case diagram consists of these main elements:

- The boundary, which delineates the system and distinguishes it from its surroundings.
- Actors, representing individuals or entities playing specific roles within the system.
- The interactions between different people or elements in specific scenarios or problems.
- The primary purpose of use case diagrams is to document a system's functional specifications. To create an effective use case diagram, certain guidelines must be followed:
  - Providing clear and meaningful names for use cases and actors.
  - Ensuring that the relationships and dependencies are well-defined.
  - Including only necessary relationships for the diagram's clarity.
  - Utilizing explanatory notes when needed to clarify essential details.

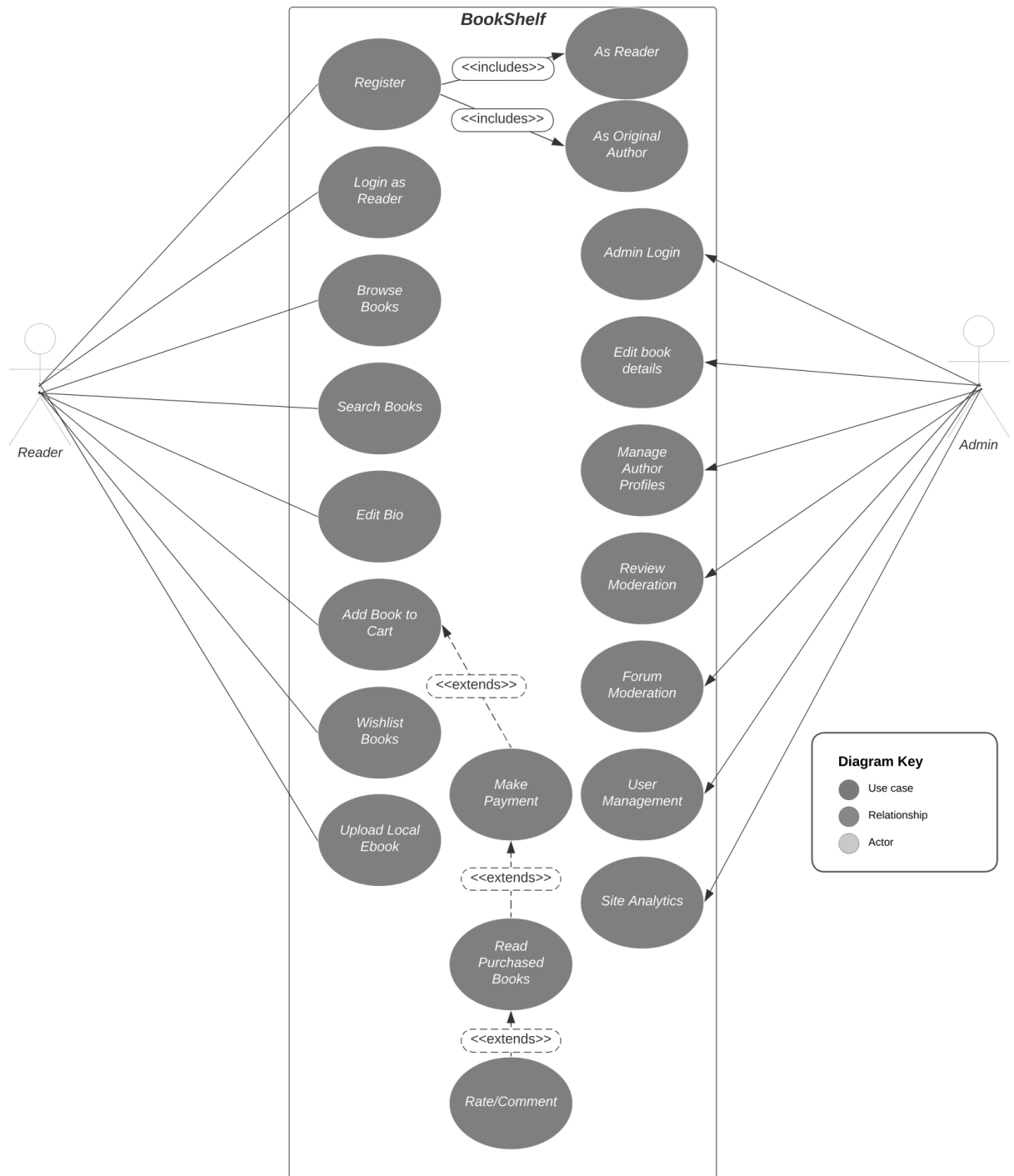


Fig 1: Use Case Diagram

### 4.2.2 SEQUENCE DIAGRAM

A sequence diagram illustrates the specific order in which objects interact with each other, showcasing the sequential flow of events. This type of diagram is also known as event diagrams or event scenarios. Sequence diagrams serve the purpose of elucidating how various components of a system collaborate and the precise sequence in which these actions occur. These diagrams find frequent application among business professionals and software developers, aiding in the understanding and depiction of requirements for both new and existing systems.

Sequence Diagram Notations:

- i. **Actors** - Within a UML diagram, actors represent individuals who utilize the system and its components. Actors are not depicted within the UML diagram as they exist outside the system being modeled. They serve as role-players in a story, encompassing people and external entities. In a UML diagram, actors are represented by simple stick figures. It's possible to depict multiple individuals in a diagram that portrays the sequential progression of events.
- ii. **Lifelines** - In a sequence diagram, each element is presented as a lifeline, with lifeline components positioned at the top of the diagram.
- iii. **Messages** - Communication between objects is achieved through the exchange of messages, with messages being arranged sequentially on the lifeline. Arrows are employed to represent messages, forming the core structure of a sequence diagram.
- iv. **Guards** - Within the UML, guards are employed to denote various conditions. They are used to restrict messages in the event that specific conditions are met, providing software developers with insights into the rules governing a system or process.

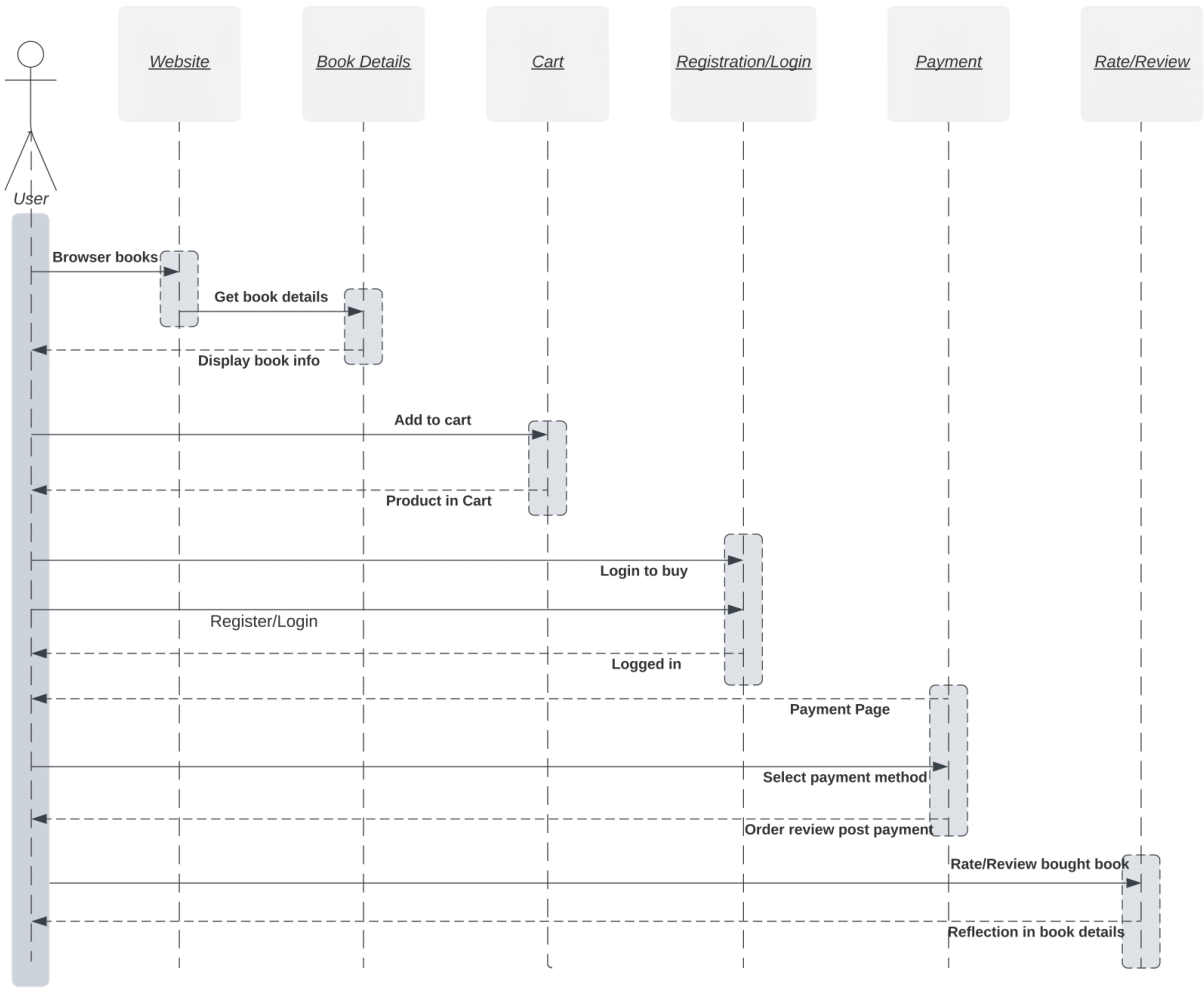


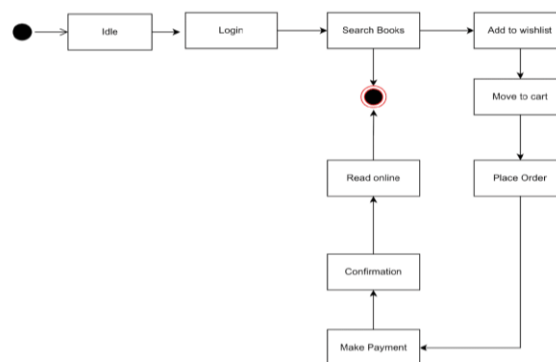
Fig 2: Sequence Diagram

### 4.2.3 State Chart Diagram

A state machine diagram, also known as a state chart, visually represents the various states an object undergoes within a system and the sequence in which these states are traversed. It serves as a record of the system's behavior, illustrating the collaborative functioning of a group of entities, whether it's a team, a collection of students, a large assembly, or an entire organization. State machine diagrams are a valuable method for depicting the interactions of diverse components within a system, outlining how objects evolve in response to events and elucidating the diverse conditions that each entity or component can inhabit.

Notations within a state machine diagram encompass:

- Initial state: Symbolized by a black circle, this signifies the commencement of a process.
- Final state: Representing the conclusion of a process, this is denoted by a filled circle within another circle.
- Decision box: Shaped like a diamond, it aids in decision-making by considering the evaluation of a guard.
- Transition: Whenever a change in authority or state occurs due to an event, it is termed a transition. Transitions are depicted as arrows with labels indicating the triggering event.
- State box: It portrays the condition or state of an element within a group at a specific moment. These are typically represented by rectangles with rounded corners.



*Fig 3: State Chart Diagram*

#### 4.2.4 Activity Diagram

An activity diagram is a visual representation of how events unfold simultaneously or sequentially. It aids in understanding the flow of activities, emphasizing the progression from one task to the next. Activity diagrams focus on the order in which tasks occur and can depict various types of flows, including sequential, parallel, and alternative paths. To facilitate these flows, activity diagrams incorporate elements such as forks and join nodes, aligning with the concept of illustrating the functioning of a system in a specific manner.

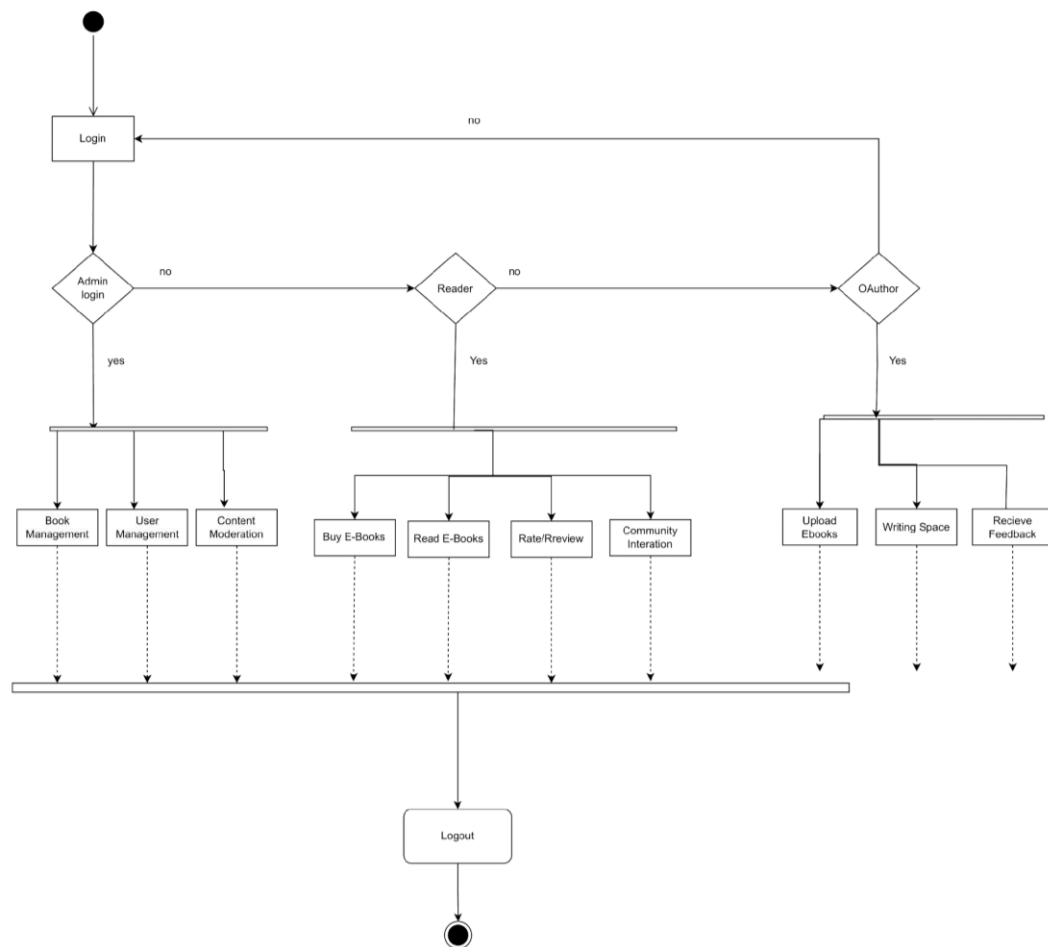
Key Components of an Activity Diagram include:

- a) **Activities:** Activities group behaviors into one or more actions, forming a network of interconnected steps. Lines between these actions outline the step-by-step sequence of events. Actions encompass tasks, controlling elements, and resources utilized in the process.
- b) **Activity Partition/Swim Lane:** Swim lanes are used to categorize similar tasks into rows or columns, enhancing modularity in the activity diagram. They can be arranged vertically or horizontally, though they are not mandatory for every activity diagram.
- c) **Forks:** Fork nodes enable the simultaneous execution of different segments of a task. They represent a point where one input transforms into multiple outputs, resembling the diverse factors influencing a decision.
- d) **Join Nodes:** Join nodes are distinct from fork nodes and employ a Logical AND operation to ensure that all incoming data flows converge to a single point, promoting synchronization.

Notations in an Activity Diagram include:

- **Initial State:** Depicting the beginning or first step in the process.
- **Final State:** Signifying the completion of all actions with no further progress.
- **Decision Box:** Ensuring that activities follow a specific path.
- **Action Box:** Representing the specific tasks or actions to be performed in the process.





*Fig 4: Activity Diagram*

### 4.2.5 Class Diagram

A class diagram serves as a static blueprint for an application, illustrating its components and their relationships when the system is in a dormant state. It provides insights into the system's structure, showcasing the various elements it comprises and how they interact. In essence, a class diagram acts as a visual guide for software development, aiding in the creation of functional applications.

Key Aspects of a Class Diagram:

- **System Overview:** A class diagram offers a high-level representation of a software system, presenting its constituent parts, associations, and collaborative dynamics. It serves as an organizational framework for elements such as names, attributes, and methods, simplifying the software development process.
- **Structural Visualization:** The class diagram is a structural diagram that combines classes, associations, and constraints to define the system's architecture.

Components of a Class Diagram:

The class diagram comprises three primary sections:

- **Upper Section:** This top segment features the class name, representing a group of objects that share common attributes, behaviors, and roles. Guidelines for displaying groups of objects include capitalizing the initial letter of the class name, positioning it in the center, using bold lettering, and employing slanted writing style for abstract class titles.
- **Middle Section:** In this part, the class's attributes are detailed, including their visibility indicators, denoted as public (+), private (-), protected (#), or package (~).
- **Lower Section:** The lower section elaborates on the class's methods or operations, presented in a list format with each method on a separate line. It outlines how the class interacts with data.

In UML, relationships within a class diagram fall into three categories:

- **Dependency:** Signifying the influence of one element's changes on another.
- **Generalization:** Representing a hierarchical relationship where one class acts as a parent, and another serves as its child.
- **Association:** Indicating connections between elements.
- **Multiplicity:** Defining constraints on the number of instances allowed to possess specific characteristics, with one being the default value when not specified.
- **Aggregation:** An aggregation is a group that is a part of a relationship called association.

- Composition: "Composition" is like a smaller part of "aggregation.". This describes how a parent and child need each other, so if one is taken away, the other won't work anymore.

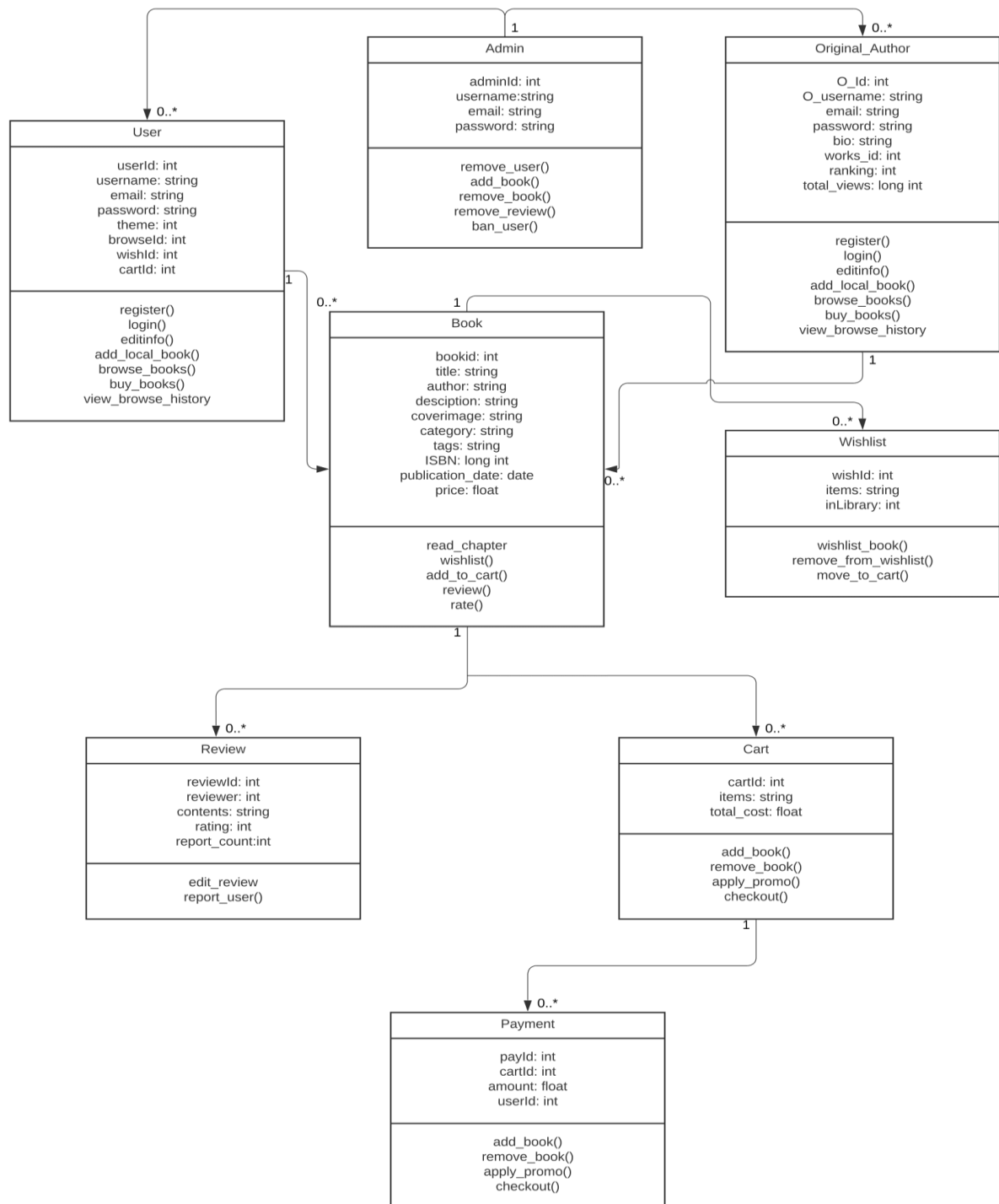


Fig 5: Class Diagram

### 4.2.6 Object Diagram

Object diagrams are derived from class diagrams and rely on them to provide a visual representation. They offer an illustration of a collection of objects related to a particular class. Object diagrams provide a snapshot of objects in an object-oriented system at a specific point in time.

Object diagrams and class diagrams share similarities, but they also have distinctions. Class diagrams are more generalized and do not portray specific objects. This abstraction in class diagrams simplifies the comprehension of a system's functionality and structure.

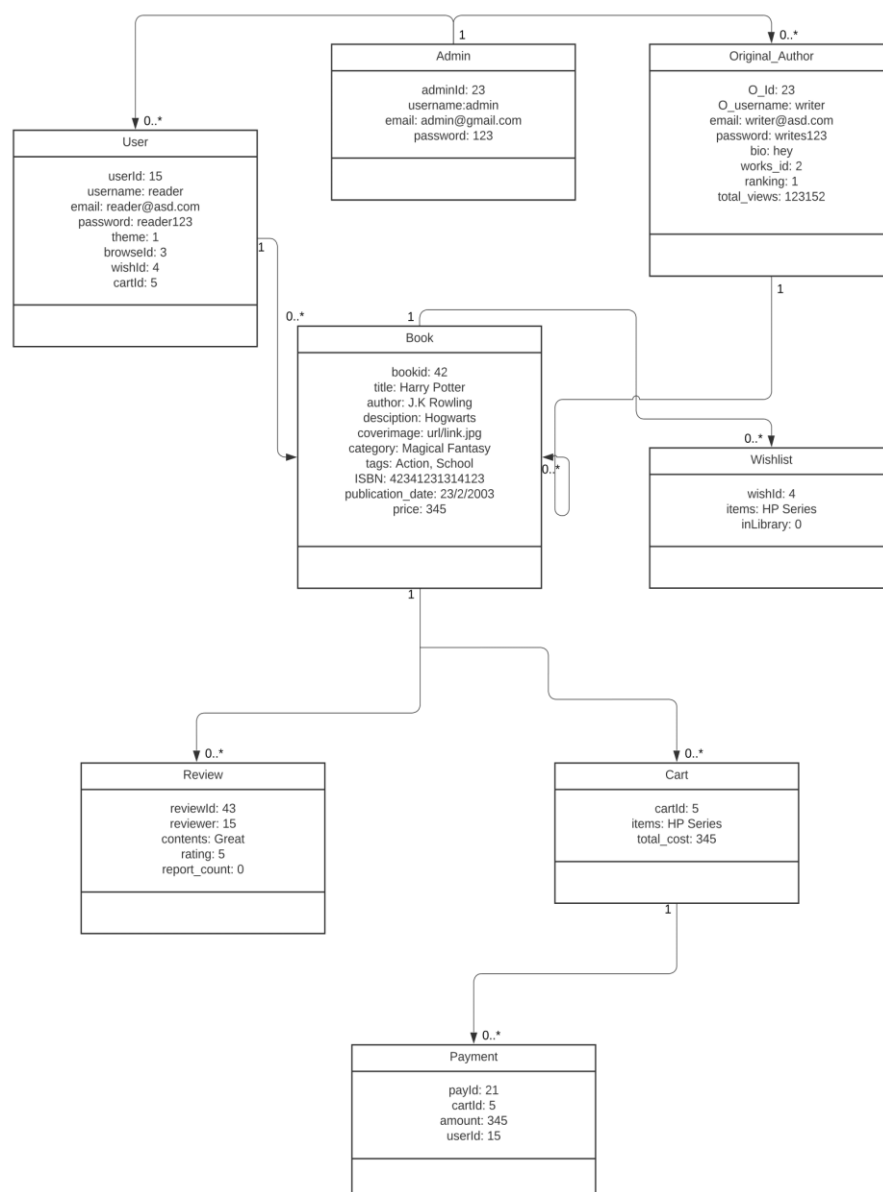


Fig 6: Object Diagram

### 4.2.7 Component Diagram

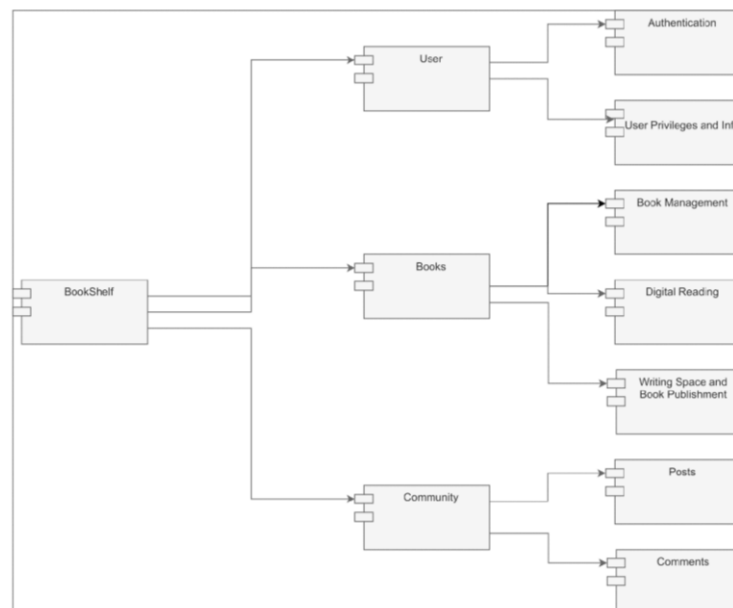
A component diagram serves the purpose of breaking down a complex system that utilizes objects into more manageable segments. It offers a visual representation of the system, showcasing its internal components such as programs, documents, and tools within the nodes.

This diagram elucidates the connections and organization of elements within a system, resulting in the creation of a usable system.

In the context of a component diagram, a component refers to a system part that can be modified and operates independently. It retains the secrecy of its internal operations and requires a specific method to execute a task, resembling a concealed box that functions only when operated correctly.

Notation for a Component Diagram includes:

- A component
- A node



*Fig 7: Component Diagram*

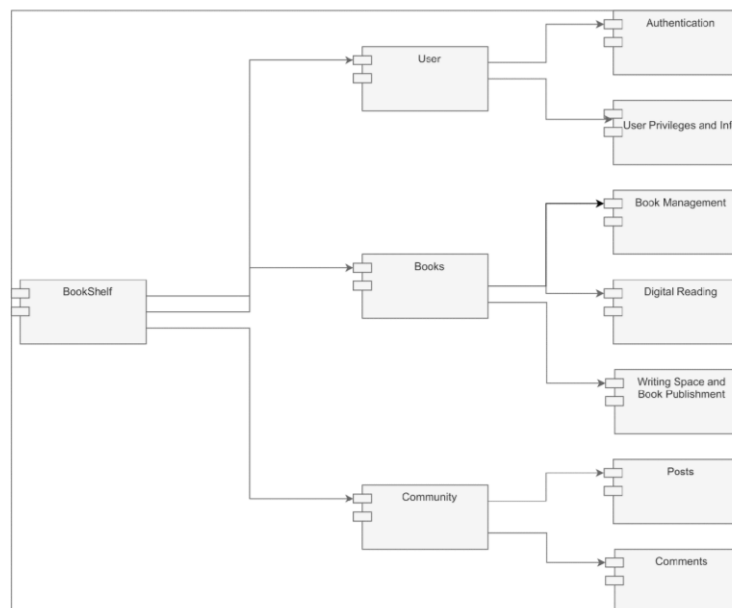
### 4.2.8 Deployment Diagram

A deployment diagram provides a visual representation of how software is positioned on physical computers or servers. It depicts the static view of the system, emphasizing the arrangement of nodes and their connections.

This type of diagram delves into the process of placing programs on computers, elucidating how software is constructed to align with the physical computer system.

Notations in a Deployment Diagram include:

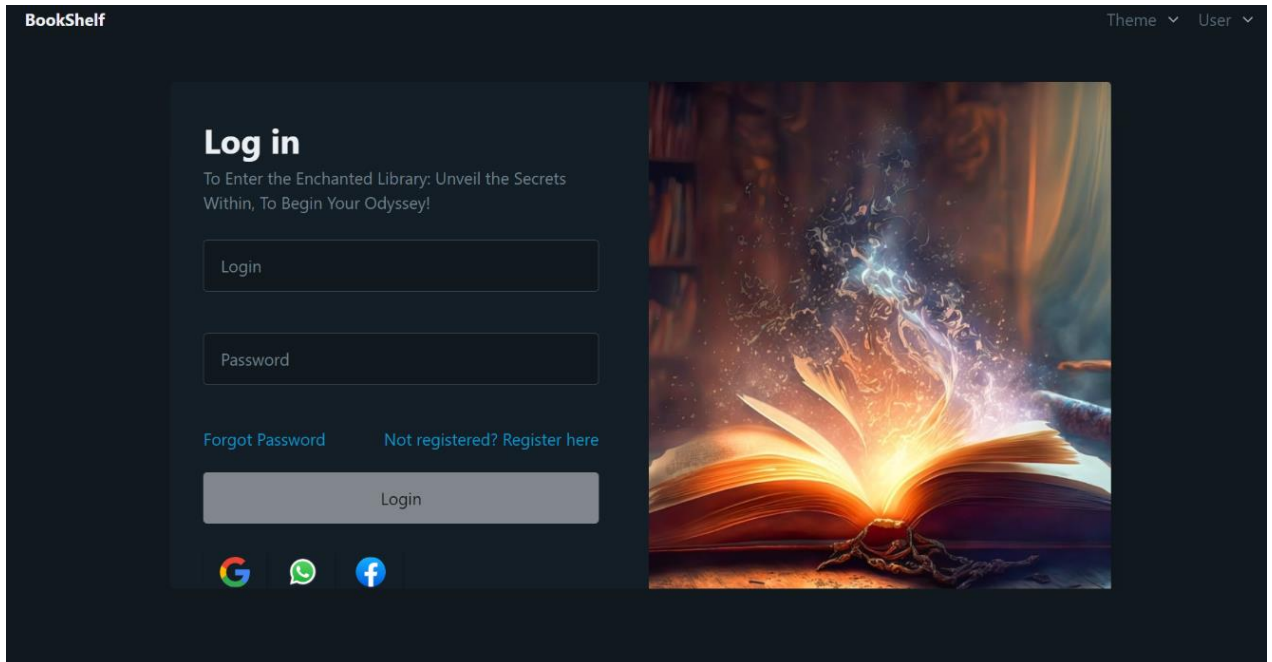
- A component
- An artifact
- An interface
- A node



*Fig 8: Deployment Diagram*

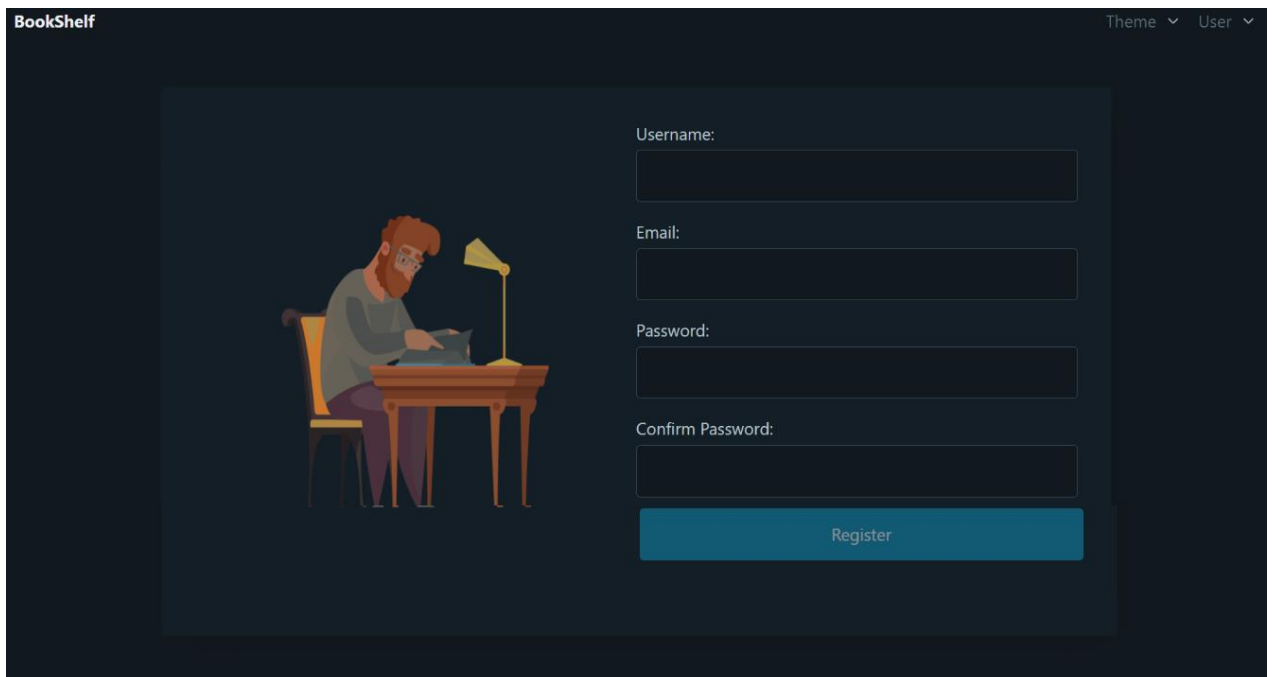
## 4.3 USER INTERFACE DESIGN USING FIGMA

### Form Name: Login Page



The Login Page UI design features a dark blue background. At the top left, the text "BookShelf" is displayed. At the top right, there are dropdown menus for "Theme" and "User". The main content area is divided into two sections. On the left, a white box contains the "Log in" heading, followed by the text "To Enter the Enchanted Library: Unveil the Secrets Within, To Begin Your Odyssey!". Below this are two input fields labeled "Login" and "Password". Under the "Password" field, there are two links: "Forgot Password" and "Not registered? Register here". A large "Login" button is positioned below these links. At the bottom of the white box are three social media icons: Google, WhatsApp, and Facebook. On the right side of the page is a large, vibrant illustration of an open book with a bright, magical light emanating from its pages, surrounded by floating particles and a small, glowing figure.


### Form Name: Registration Page



The Registration Page UI design features a dark blue background. At the top left, the text "BookShelf" is displayed. At the top right, there are dropdown menus for "Theme" and "User". The main content area is divided into two sections. On the left, there is a large illustration of a man with a beard and glasses, wearing a green shirt and purple pants, sitting at a wooden desk and reading a book. A yellow lamp is on the desk. On the right side of the page, a white box contains the registration form. It includes four input fields labeled "Username:", "Email:", "Password:", and "Confirm Password:". Below these fields is a large "Register" button.

**Form Name: Registration OTP verification**

BookShelf Theme ▾ User ▾




Enter OTP:

Verify OTP

**Form Name: Forgot Password**

BookShelf Theme ▾ User ▾



Email:

Reset Password



## Form Name: Home Page


BookShelf

Search...

HomeSeriesLogoutBuy MembershipSupport UsJoin Discord!

Surprise Me!

01




**Lord of the Mysteries**  
Action, Adventure, Drama, Fantasy, Historical, Horror, Mature, Mystery, Psychological, Sci-Fi, Thriller  
With the rising tide of steam power and machinery, who can come close to being a Beyonder? Shrouded in the fog of history and darkness, who or what is the lurking evil that... [more>>](#)

4.7  
692 votes

BOOKMARK

02




**The Legendary Mechanic**  
Action, Adventure, Comedy, Fantasy, Mecha, Romance, Sci-Fi  
What do you do when you wake up and find yourself inside the very game that you love? What do you do when you realize you that you have not only become an NPC — you have even been... [more>>](#)

4.6  
654 votes

BOOKMARK

03




**Omniscient Reader's Viewpoint**  
Action, Adventure, Fantasy, Mature, Mystery, Psychological, Shounen  
Only I know the end of this world. One day our MC finds himself stuck in the world of his favorite webnovel. What does he do to survive? It is a world struck by catastrophe and... [more>>](#)

4.5  
146 votes

BOOKMARK


04



**Soul of Negary**  
Action, Adventure, Fantasy, Horror, Mystery, Supernatural, Xianxuan  
Heed my call O' Dragon of Eternal Sin, the Progressive Disaster, the Forerunner, the Land of Eternal Peace, the Flames of Soul Burning, He Who Owns The Deep Soul, the Sound of... [more>>](#)

4.5  
60 votes

BOOKMARK




**Became an Evolving Space Monster**  
Sci-Fi, Fantasy, Action, Comedy, Adventure  
Lost in the vast expanse of space, trapped in the icy confines of a metal spacecraft, I was forced to face a horrifying truth. That I had transformed into a ruthless alien predator in a space survival game...  
★★★★★ (30) • UP: 2 HOURS AGO

READ

68 26 948

BOOKMARK




**The Primordial Record**  
Action, Adventure, Fantasy, Mystery, Horror, Supernatural  
Rowan awakes awake in a new world inside the body of a dying prince. His new body contain terrifying secrets that he has to protect, for inside him lies the key to Eternity. Rowan must learn to harness his new abilities and fight against his enemies who would stop at nothing to deprive him of his...  
★★★★★ (25) • UP: 3 DAYS AGO

READ

61 18 834

BOOKMARK



**Shocking the Whole Internet! You Are Not a Psychologist at All!**  
Action, Comedy, Adventure, Slice of Life, Thriller, Psychological  
Chen Yu unexpectedly obtained a heavenly book, gaining the ability to peek into the secrets of the universe. As long as he helped others change their fate, he could gain the cultivation to become an immortal. For this reason, Chen Yu started to live stream psychological counseling. In the live...  
★★★★★ (22) • UP: 5 HOURS AGO

READ

42 12 159

BOOKMARK

Filter


Genre All ▼ Type All ▼

Status All ▼ Order by Default ▼

Search


Popular Novels

1




**Novel 1**  
Genres: Academy, Fantasy, Light Novel, Misunderstanding, Regret...  
★★★★★ 9.2

2



**Novel 2**  
Genres: Academy, Action, Adult, Adventure, Fantasy, Transmigration  
★★★★★ 9.00

3



**Novel 3**  
Genres: Academy, Action, Fantasy, obsession, Regret  
★★★★★ 9.00

Privacy PolicyTerms of ServiceContact UsDMCA

A-Z LIST | Searching books order by alphabet name A to Z.

# 0-9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

## 4.4 DATABASE DESIGN

### 4.4.1 Relational Database Management System (RDBMS)

A database is a system designed to store and facilitate the retrieval and utilization of information. The integrity and security of the data within a database are of paramount importance.

Creating a database involves two essential steps. Initially, it is crucial to understand the user's requirements and establish a database structure that aligns with their needs. The first step entails devising an organizational plan for the information, which is not reliant on any particular computer program.

Subsequently, this plan is employed to craft a design tailored to the specific computer program that will be employed to construct the database system. This phase is centered on determining how data will be stored within the chosen computer program.

Key aspects of database development include:

- **Data Integrity:** Ensuring the accuracy and consistency of data stored within the database.
- **Data Independence:** The ability to modify the data storage structure without affecting the application's access to the data.

### 4.4.2 Normalization

A way of showing a database as a group of connections is called a relational model. Every relationship is like a chart of information or a group of data. In the formal way of talking about tables, a row is called a tuple, a column header is called an attribute, and the table is called a relation. A bunch of tables with special names make up a relational database. A row in a story shows a group of things that are connected.

#### **Relations, Domains & Attributes**

Tables serve as containers for organized and related information. Within a table, the rows are referred to as tuples, and a tuple is essentially an ordered list containing  $n$  items. Columns in a table are synonymous with characteristics, each representing a specific aspect of the data.

In a relational database, tables are interconnected, ensuring coherence and meaningful associations between different sets of data. This relational structure forms the foundation of a well-structured database.

A domain can be thought of as a collection of fundamental values with a shared source or data type. Naming domains is a helpful practice as it aids in comprehending the significance of the values they encompass. It's essential to recognize that values within a domain are indivisible.

- Table relationships are established using keys, with the primary key and foreign key being the primary types. Entity Integrity and Referential Integrity are fundamental principles that guide these relationships.
- Entity Integrity mandates that no Primary Key should contain null values, reinforcing the unique and integral nature of primary keys in database relationships.

Normalization is the process of organizing data to ensure its efficient storage and to minimize redundancy while maintaining accuracy and reliability. This approach involves eliminating unnecessary columns and breaking down large tables into smaller, more manageable parts. Normalization helps prevent errors when adding, removing, or modifying data. In data modeling, two key concepts are integral to its normal form: keys and relationships.

Keys serve as unique identifiers to distinguish one row from all others in a table. There are two main types of keys: the primary key, which groups similar records within a table, and the foreign key, which acts as a special code to identify specific records in another table.

1. **First Normal Form (1NF):** This rule dictates that an attribute can only contain a single value that cannot be further divided. Each value in a tuple must match the attribute's type. In 1NF, tables cannot contain other tables or have tables as part of their data. This form ensures that values are indivisible and represent a single entity. Achieving 1NF often involves organizing data into separate tables, with each table having a designated key based on project requirements. New relationships are established for various data categories or related groups, eliminating redundant information. A relationship adhering to primary key constraints alone is termed the first normal form.
2. **Second Normal Form (2NF):** In simpler terms, 2NF stipulates that no additional information should be associated with only part of the primary information used for data organization. This involves breaking down the information and creating separate groupings for each part along with its related details. It's essential to maintain a connection between the original primary key and any data dependent on it. This step helps remove data that relies on only a portion of the key. When a set of information has a primary means of identifying each item, and all other details depend solely on that primary means, it is considered to be in the second normal form.

3. **Third Normal Form (3NF)** is a critical concept in database normalization, aiming to achieve table independence and control over attributes. In 3NF, a table should not contain columns that depend on other non-key columns, ensuring that each column is functionally dependent only on the primary key. It breaks down relationships involving non-key attributes to eliminate dependencies on attributes not part of the primary key. To meet the criteria for 3NF, data must already be in the Second Normal Form (2NF), and non-key attributes should not rely on other non-key attributes within the same table, avoiding transitive dependencies. This process enhances data integrity, reduces redundancy, and optimizes database structure and organization.

#### 4.4.3 Sanitization

Django incorporates various in-built mechanisms for data sanitization. To begin with, it provides a diverse range of field types that come with automatic validation and sanitization capabilities. For instance, the CharField performs automatic validation and cleaning of text input, while the EmailField ensures that email addresses adhere to the correct format. These field types serve as protective measures to prevent the storage of malicious or invalid data in the database.

Moreover, Django strongly advocates for the utilization of form validation to sanitize user input. Django's forms are equipped with predefined validation methods and validators that can be applied to form fields. These validators execute a range of checks and cleansing operations, such as confirming that numerical values fall within specified ranges or verifying that uploaded files adhere to specific formats. These combined features in Django promote data integrity and security, making it a reliable choice for web application development.

#### 4.4.4 Indexing

The index keeps track of a certain piece of information or group of information, arranged in order of its value. Arranging the index entries helps find things quickly and easily that match exactly or are within a certain range. Indexes make it easy to find information in a database without having to search through every record every time the database is used. An index is like a roadmap for finding information in a database. It helps you look up data quickly and also makes it easy to find records that are in a certain order. It can be based on one or more columns in the table.

## 4.5 TABLE DESIGN

### 1. tbl\_Users

Primary key: **UserID**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	UserID	int	Primary Key	Registered user id
2	Username	Varchar(30)	Not Null	User's username
3	Email	Varchar(50)	Not Null	User's email
4	Password hash	Text	Not Null	Hash of user's password
5	RegistrationDate	Datetime	Not Null	Date/Time of the user's registration
6	LastLogin	Datetime	Not Null	Date/Time of user's last login

### 2. tbl\_UserPreferences

Primary key: **PreferenceID**

Foreign Key: **UserID** references **tbl\_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	PreferenceID	int	Primary Key	Key for this table
2	UserID	int	Foreign Key	Foreign Key to table users
3	Theme	Varchar(10)	Not Null	Records user's preferred theme
4	NotificationSettings	Varchar(10)	Not Null	Silent or All notification
5	Language	Varchar(10)	Not Null	Preferred medium of communication

### 3. tbl\_Books

Primary key: **BookID**

Foreign Key: **AuthorID** references **tbl\_Authors**, **LanguageID** references **tbl\_Languages**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	BookID	int	Primary Key	Key for this table
2	Title	Varchar(20)	Not Null	Book's title

3	AuthorID	Int	Foreign Key	Foreign Key to Authors table
4	ISBN	Int	Not Null	Book's ISBN
5	PublicationDate	Datetime	Not Null	Book's publ. date
6	Category	Varchar(20)	Not Null	Category the book belongs to
7	Description	Text	Not Null	A synopsis for the book
8	LanguageID	Int	Foreign Key	Foreign Key to Languages table
9	Price	Int	Not Null	Price of the book

#### 4. tbl\_Authors

Primary key: **AuthorID**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	AuthorID	int	Primary Key	Key for this table
2	AuthorName	Varchar(30)	Not Null	Name of the author
3	Biography	Text	Not Null	A bio for the author
4	SocialMediaLinks	Varchar(40)	Null	Links to the author's social media
5	isAmateur	Boolean	Not Null	To check if the author is publishing via publishers

#### 5. tbl\_Reviews

Primary key: **ReviewID**

Foreign Key: **BookID** references **tbl\_Books**, **UserID** references **tbl\_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	ReviewID	int	Primary Key	Key for this table
2	BookID	int	Foreign Key	Foreign Key to table books
3	UserID	Int	Foreign Key	Foreign Key to the user who wrote the review
4	Rating	Int	Not Null	The rating given
5	ReviewText	Text	Not Null	The review body
6	ReviewDate	Datetime	Not Null	Date of reviewing the book

**6. tbl\_User\_Details**Primary key: **DetailsID**Foreign Key: **UserID** references **tbl\_Users**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	DetailsID	int	Primary Key	Key for this table
2	UserID	int	Foreign Key	Foreign Key to table users
3	Full_Name	Varchar(40)	Not Null	User's full name
4	Bio	Varchar(50)	Not Null	Short bio set by the user
5	Gender	Varchar	Not Null	User's gender
6	DOB	Date	Not Null	Date of user's birth
7	Address	Text	Not Null	Address of the user

**7. tbl\_Languages**Primary key: **LanguageID**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	LanguageID	int	Primary Key	Key for this table
2	LanguageName	Varchar(20)	Not Null	Foreign Key to table users

**8. tbl\_PurchaseHistory**Primary key: **PurchaseID**Foreign Key: **UserID** references **tbl\_Users**, **BookID** references **tbl\_Books**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	PurchaseID	int	Primary Key	Key for this table
2	UserID	int	Foreign Key	Foreign Key to table users
3	BookID	Int	Foreign Key	Foreign Key to books table
4	PurchaseDate	Datetime	Not Null	Date/time of book's purchase
5	PaymentAmount	Decimal	Not Null	Amount paid for the book

**9. tbl\_ReadingProgress**Primary key: **ProgressID**Foreign Key: **UserID** references **tbl\_Users**, **BookID** references **tbl\_Books**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	ProgressID	int	Primary Key	Key for this table
2	UserID	int	Foreign Key	Foreign Key to table users
3	BookID	Int	Foreign Key	Foreign Key to books table
4	PageNumber	Int	Not Null	Page where the user left off
5	TimeSpent	Int	Not Null	Total time spend on this book
6	LastReadDate	Datetime	Not Null	Date/Time when last read

**10. tbl\_Voting**Primary key: **VotingID**Foreign Key: **UserID** references **tbl\_Users**, **BookID** references **tbl\_Books**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	VotingID	int	Primary Key	Key for this table
2	UserID	int	Foreign Key	Foreign Key to table users
3	BookID	Int	Not Null	Foreign Key to table books
4	VoteDate	Datetime	Not Null	Time when vote was given

**11. tbl\_CartItems**Primary key: **CartItemID**Foreign Key: **UserID** references **tbl\_Users**, **BookID** references **tbl\_Books**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	CartItemID	int	Primary Key	Key for this table
2	UserID	int	Foreign Key	Foreign Key to table users
3	BookID	Int	Not Null	Foreign Key to table books



4	AddedDate	Datetime	Not Null	Date/Time when book was added to cart
---	-----------	----------	----------	---------------------------------------

## 12. tbl\_Chapters

Primary key: **ChapterID**

Foreign Key: **BookID** references **tbl\_Books**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	ChapterID	int	Primary Key	Key for this table
2	BookID	int	Foreign Key	Foreign Key to table books
3	ChapterNumber	Int	Not Null	Ordered number of the chapter
4	ChapterTitle	Varchar(30)	Not Null	Title of the chapter
5	ChapterContent	Text	Not Null	Content of the chapter

## **CHAPTER 5**

### **SYSTEM TESTING**

## 5.1 INTRODUCTION

Software testing is an essential procedure employed to verify if a computer program functions as intended. It is conducted to ensure that the software performs its designated tasks accurately and complies with the prescribed requirements and standards. Validation is the process of inspecting and assessing software to confirm its adherence to the specified criteria. Software testing is a method for assessing the performance of a program, often in conjunction with techniques like code inspection and program walkthroughs. Validation ensures that the software aligns with the user's expectations and requirements.

Several principles and objectives guide the process of software testing, including:

1. Testing is the practice of executing a program with the primary aim of identifying errors.
2. An effective test case is one that has a high likelihood of uncovering previously undiscovered errors.
3. A successful test is one that exposes previously undiscovered errors.

When a test case operates effectively and accomplishes its objectives, it can detect flaws within the software. This demonstrates that the computer program is functioning as intended and is performing well. The process of evaluating a computer program encompasses three primary aspects:

1. Correctness assessment
2. Evaluation of implementation efficiency
3. Examination of computational complexity

## 5.2 TEST PLAN

A test plan serves as a comprehensive set of instructions for conducting various types of tests. It can be likened to a map that outlines the steps to follow when evaluating a computer program. Software developers create instructions for both using and organizing the necessary information for the program's proper functionality. They ensure that each component of the program performs its intended functions. To ensure the thorough testing of the software, a group known as the ITG (Information Technology Group) is responsible for verifying its functionality, instead of relying solely on the software's creators for testing.

The objectives of testing should be clearly defined and measurable. A well-structured test plan should encompass details about the frequency of failures, the associated repair costs, the occurrence

rate of issues, and the time required for complete testing. The levels of testing typically include:

- Unit testing
- Integration testing
- Data validation testing
- Output testing

### **5.2.1 Unit Testing**

Unit testing checks the smallest part of a software design - the software component or module. Testing important control paths within a module using the design guide to find errors. This means how difficult the tests are for each small part of a program and what parts of the program haven't been tested yet. Unit testing is a type of testing that looks at how the code works inside and can be done at the same time for different parts of the program.

Before starting any other test, we need to check if the data flows correctly between different parts of the computer program. If the information doesn't move in and out correctly, all other checks are pointless. When designing something, it's important to think about what could go wrong and make a plan for how to deal with those problems. This can mean redirecting the process or stopping it completely.

The Sell-Soft System was tested by looking at each part by itself and trying different tests on it. Some mistakes in the design of the modules were discovered and then fixed. After writing the instructions for different parts, each part is checked and tried out separately. We got rid of extra code and made sure everything works the way it should.

### **5.2.2 Integration Testing**

Integration testing is a critical process in software development that involves constructing a program while simultaneously identifying errors in the interaction between different program components. The primary objective is to utilize tested individual parts and assemble them into a program according to the initial plan. This comprehensive testing approach assesses the entire program to ensure its proper and correct functionality.

As issues are identified and resolved during integration testing, it's not uncommon for new problems to surface, leading to an ongoing cycle of testing and refinement. After each individual component of the system is thoroughly examined, these components are integrated to ensure they function harmoniously. Additionally, efforts are made to standardize all programs to ensure uniformity rather than having disparate versions.

### 5.2.3 Validation Testing or System Testing

The final phase of testing involves a comprehensive examination of the entire system to ensure the correct interaction of various components, including different types of instructions and building blocks. This testing approach is referred to as Black Box testing or System testing.

Black Box testing is a method employed to determine if the software functions as intended. It assists software engineers in identifying all program issues by employing diverse input types. Black Box testing encompasses the assessment of errors in functions, interfaces, data access, performance, as well as initialization and termination processes. It is a vital technique to verify that the software meets its intended requirements and performs its functions correctly.

### 5.2.4 Output Testing or User Acceptance Testing

System testing is conducted to assess user satisfaction and alignment with the company's requirements. During the development or update of a computer program, it's essential to maintain a connection with the end-users. This connection is established through the following elements:

- Input Screen Designs.
- Output Screen Designs.

To perform system testing, various types of data are utilized. The preparation of test data plays a crucial role in this phase. Once the test data is gathered, it is used to evaluate the system under investigation. When issues are identified during this testing, they are addressed by following established procedures. Records of these corrections are maintained for future reference and improvement. This process ensures that the system functions effectively and meets the needs of both users and the organization.

### 5.2.5 Automation Testing

Automated testing is a method employed to verify that software functions correctly and complies with established standards before it is put into official use. This type of testing relies on written instructions that are executed by testing tools. Specifically, UI automation testing involves the use of specialized tools to automate the testing process. Instead of relying on manual interactions where individuals click through the application to ensure its proper functioning, scripts are created to automate these tests for various scenarios. Automating testing is particularly valuable when it is necessary to conduct the same test across multiple computers simultaneously, streamlining the testing process and ensuring consistency.

### **5.2.6 Selenium Testing**

Selenium is a valuable and free tool designed for automating website testing. It plays a crucial role for web developers as it simplifies the testing process. Selenium automation testing refers to the practice of using Selenium for this purpose. Selenium isn't just a single tool; it's a collection of tools, each serving distinct functions in the realm of automation testing. Manual testing is a necessary aspect of application development, but it can be monotonous and repetitive. To alleviate these challenges, Jason Huggins, an employee at ThoughtWorks, devised a method for automating testing procedures, replacing manual tasks. He initially created a tool named the JavaScriptTestRunner to facilitate automated website testing, and in 2004, it was rebranded as Selenium.

## Test Case 1 – Reader Login

### Code

```

from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR,"summary.secondary")
        theme.click()
        time.sleep(1)
        light=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(2) a")
        light.click()
        time.sleep(2)
        theme.click()
        time.sleep(1)
        dark=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(3) a")
        dark.click()
        time.sleep(1)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("dhunan")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("asdASD@1234")
        submit_button = driver.find_element(By.CSS_SELECTOR,
"button[type='submit']")
        submit_button.click()
        time.sleep(2)
        browse=driver.find_element(By.CSS_SELECTOR,"div.navbar-content >
a:first-child")
        if browse:
            print("OK")

```

```

DevTools listening on ws://127.0.0.1:50505/devtools/browser/624889d6-8879-4c0a-8990-96c560b7782f
.
-----
Ran 1 test in 64.262s

OK
Destroying test database for alias 'default'...

```

## Test Report

Test Case 1					
Project Name: BookShelf					
Login Test Case					
Test Case ID: 1			Test Designed By: Midhun Krishnan M		
Test Priority (Low/Medium/High): High			Test Designed Date: 10-10-2022		
Module Name: Login Module			Test Executed By: Ms. Ankitha Philip		
Test Title: Reader Login			Test Execution Date: 10-10-2023		
Description: User has a valid email/password					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to login page		Login form should be displayed	Login form is displayed	Pass
2	Change theme		Changes theme to light and back to dark	Theme changes to light and then back to black	Pass
3	Provide valid username	Username: dhunan	User should be able to login	User logs in	Pass
4	Provide valid password	Password: asdASD@1234			
5	Click on login button				
Post-Condition: User is validated with database and successfully logs into account. The Account session details are logged in database					



## Test Case 2: Reader Interface

### Code

```

from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR,"summary.secondary")
        theme.click()
        time.sleep(1)
        light=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(2) a")
        light.click()
        time.sleep(2)
        theme.click()
        time.sleep(1)
        dark=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(3) a")
        dark.click()
        time.sleep(1)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("dhunan")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("asdASD@1234")
        submit_button = driver.find_element(By.CSS_SELECTOR,
"button[type='submit']")
        submit_button.click()
        time.sleep(2)
        browse=driver.find_element(By.CSS_SELECTOR,"div.navbar-content >
a:first-child")
        if browse:
            print("OK")
            browse.click()
            time.sleep(1)
            driver.execute_script("window.scrollTo(0, 150);")
            time.sleep(1)
            view=driver.find_element(By.CSS_SELECTOR,"a[href*='/books/book/8/']
")
            view.click()
            time.sleep(1)

```

```

driver.execute_script("window.scrollTo(0, 600);")
time.sleep(1)
chaps=driver.find_element(By.CSS_SELECTOR, "button.chap-list")
chaps.click()
time.sleep(1)
driver.execute_script("window.scrollTo(0, 300);")
time.sleep(1)
chap = driver.find_element(By.CSS_SELECTOR,
'a[href*="/books/8/chapter/Text/part0007.xhtml/"]')
chap.click()
time.sleep(1)
driver.execute_script("window.scrollTo(0, 300);")
time.sleep(1)
driver.execute_script("window.scrollTo(0, -300);")
time.sleep(1)
options = driver.find_element(By.CSS_SELECTOR, 'button#options-
button')
options.click()
options.click()
time.sleep(1)
select=driver.find_element(By.CSS_SELECTOR, 'select#font-style')
select.click()
time.sleep(1)
option1=driver.find_element(By.CSS_SELECTOR, 'select#font-style
option[value="calibri"]')
option1.click()
time.sleep(1)
theme2=driver.find_element(By.CSS_SELECTOR, 'select#theme')
theme2.click()
time.sleep(1)
option2=driver.find_element(By.CSS_SELECTOR, 'select#theme
option[value="light"]')
option2.click()
time.sleep(1)
theme2.click()
increaseFont=driver.find_element(By.CSS_SELECTOR, 'span.adjust-
button#font-size-increase')
increaseFont.click()
increaseFont.click()
increaseFont.click()
increaseFont.click()
increaseFont.click()
time.sleep(1)
increasePad=driver.find_element(By.CSS_SELECTOR, 'span.adjust-
button#content-padding-increase')
increasePad.click()
increasePad.click()
increasePad.click()
time.sleep(1)
increaseline=driver.find_element(By.CSS_SELECTOR, 'span.adjust-
button#line-spacing-increase')
increaseline.click()
increaseline.click()
increaseline.click()
time.sleep(1)
options.click()

```

```

DevTools listening on ws://127.0.0.1:50505/devtools/browser/624889d6-8879-4c0a-8990-96c560b7782f
.
-----
Ran 1 test in 64.262s

OK
Destroying test database for alias 'default'...

```

## Test report

Test Case 2					
Project Name: BookShelf					
Reader Test Case					
Test Case ID: 2			Test Designed By: Midhun Krishnan M		
Test Priority (Low/Medium/High): High			Test Designed Date: 13-10-2023		
Module Name: Reader Module			Test Executed By: Ms. Ankitha Philip		
Test Title: Reader Login			Test Execution Date: 13-10-2023		
Description: User accesses the reading module					
Pre-Condition: User need to have the book in library					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to login page		Login form should be displayed	Login form is displayed	Pass
2	Change theme		Changes theme to light and back to dark	Theme changes to light and then back to black	Pass
3	Provide valid username	Username : dhunan	User should be able to login	User logs in	Pass
4	Provide valid password	Password: asdASD@1234			
5	Click on login button				
6	Click on browse navbar item		User should get redirected to catalogue page	User gets redirected to catalogue page	Pass

7	Click on 'View Details' of a boook		The book details page should be visible	Book details page is loaded	Pass
8	Click on 'Chapter List'		Should render the list of chapters for that book	Renders the list of chapters for that book	Pass
9	Click on a chapter		The chapter title and content should be visible to user	User is redirected to the reading page with appropriate chapter title and content	Pass
10	User changes the options of the page		The font-family, theme, font-size, content-padding, line-spacing should change	The font-family, theme, font-size, content-padding, line-spacing changes	Pass
<b>Post-Condition:</b> User is able to read a chapter of the novel he/she has in his library					

### Test Case 3: Book Details

#### Code

```

from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR, "summary.secondary")
        theme.click()

```

```

        time.sleep(1)
        light=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(2) a")
        light.click()
        time.sleep(2)
        theme.click()
        time.sleep(1)
        dark=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(3) a")
        dark.click()
        time.sleep(1)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("dhunan")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("asdASD@1234")
        submit_button = driver.find_element(By.CSS_SELECTOR,
"button[type='submit']")
        submit_button.click()
        time.sleep(2)
        browse=driver.find_element(By.CSS_SELECTOR,"div.navbar-content >
a:first-child")
        if browser:
            print("OK")
        browse.click()
        time.sleep(1)
        driver.execute_script("window.scrollTo(0, 150);")
        time.sleep(1)
        view=driver.find_element(By.CSS_SELECTOR,"a[href*='/books/book/8/']
")
        view.click()
        time.sleep(2)
        driver.execute_script("window.scrollTo(0, 2000);")
        time.sleep(2)
        like = driver.find_element(By.CSS_SELECTOR,"div.like-button
button.btn.btn-primary:first-child")
        like.click()
        time.sleep(2)
        driver.execute_script("window.scrollTo(0, -2000);")
        time.sleep(1)
        toLibrary = driver.find_element(By.CSS_SELECTOR,"div.breadcrumb
a:nth-child(2)")
        toLibrary.click()

```

```

DevTools listening on ws://127.0.0.1:50505/devtools/browser/624889d6-8879-4c0a-8990-96c560b7782f
.
-----
Ran 1 test in 64.262s

OK
Destroying test database for alias 'default'...

```

## Test report

Test Case 3					
Project Name: BookShelf					
Books Test Case					
Test Case ID: 3			Test Designed By: Midhun Krishnan M		
Test Priority (Low/Medium/High): Low			Test Designed Date: 16-10-2023		
Module Name: Books Module			Test Executed By: Ms. Ankitha Philip		
Test Title: Accessing Book page and Review section			Test Execution Date: 16-10-2023		
Description: User goes to book details page and likes/dislikes a review					
Pre-Condition: User need to be logged in					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to login page		Login form should be displayed	Login form is displayed	Pass
2	Change theme		Changes theme to light and back to dark	Theme changes to light and then back to black	Pass
3	Provide valid username	Username : dhunan	User should be able to login	User logs in	Pass
4	Provide valid password	Password: asdASD@1234			
5	Click on login button				
6	Click on browse navbar item		User should get redirected to catalogue page	User gets redirected to catalogue page	Pass
7	Click on ‘View Details’ of a boook		The book details page should be visible	Book details page is loaded	Pass
8	User scrolls to the bottom		The reviews of the book should be	The reviews of the book are actually visible	Pass

			visible		
9	User likes/dislikes a review		User likes if he hasn't liked and dislikes if he already has	The change in like count is reflected	Pass
10	User clicks 'Library' navbar item		The user should get redirected to his library	The user is redirected to his library	Pass
<b>Post-Condition:</b> User can like/dislike a review after accessing the review page and an appropriate notification gets sent to the review author. User then redirects to his library page					

#### Test Case 4: Search and Profile

##### Code

```

from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'
    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR,"summary.secondary")
        theme.click()
        time.sleep(1)
        light=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(2) a")
        light.click()
        time.sleep(2)
        theme.click()
        time.sleep(1)
        dark=driver.find_element(By.CSS_SELECTOR,"ul[role='listbox']
li:nth-child(3) a")
        dark.click()

```

```

        time.sleep(1)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("dhunan")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("asdASD@1234")
        submit_button = driver.find_element(By.CSS_SELECTOR,
"button[type='submit']")
        submit_button.click()
        time.sleep(2)
        browse=driver.find_element(By.CSS_SELECTOR,"div.navbar-content >
a:first-child")
        if browse:
            print("OK")
            browse.click()
            time.sleep(1)
            driver.execute_script("window.scrollTo(0, 150);")
            time.sleep(1)
            view=driver.find_element(By.CSS_SELECTOR,"a[href*='/books/book/8/']
")
            view.click()
            time.sleep(1)
            toLibrary = driver.find_element(By.CSS_SELECTOR,"div.breadcrumb
a:nth-child(2)")
            toLibrary.click()
            time.sleep(1)
            search=driver.find_element(By.CSS_SELECTOR,"input#search-
bar.search-bar")
            search.send_keys('monk')
            time.sleep(2)
            search.send_keys('')
            time.sleep(1)
            navbar_user = driver.find_element(By.CSS_SELECTOR,".navbar-user")
            action = ActionChains(driver)
            action.move_to_element(navbar_user).perform()
            profile = driver.find_element(By.CSS_SELECTOR, "div#dropdown-menu
a[href='/user/profile']")
            profile.click()
            wait = WebDriverWait(driver, 10)
            time.sleep(1)
            bio = wait.until(EC.element_to_be_clickable((By.CSS_SELECTOR,
"div.list-group a:nth-child(2)"))))
            bio.click()
            time.sleep(1)
            contact=driver.find_element(By.CSS_SELECTOR, "div.list-group a:nth-
child(3)")
            contact.click()
            time.sleep(1)
            time.sleep(1)
            wait = WebDriverWait(driver, 10)
            navbar_user =
wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, "div.navbar-
user.dropdown img.profile-image"))))
            action = ActionChains(driver)
            action.move_to_element(navbar_user).perform()
            logout = driver.find_element(By.CSS_SELECTOR, "div#dropdown-menu
a[href='/accounts/logout']")
            logout.click()

```



```
time.sleep(1)
```

```
DevTools listening on ws://127.0.0.1:50505/devtools/browser/624889d6-8879-4c0a-8990-96c560b7782f
.
-----
Ran 1 test in 64.262s

OK
Destroying test database for alias 'default'...
```

## Test report

Test Case 4					
Project Name: BookShelf					
Search and Info Case					
Test Case ID: 4			Test Designed By: Midhun Krishnan M		
Test Priority (Low/Medium/High): Medium			Test Designed Date: 19-10-2023		
Module Name: Library Module			Test Executed By: Ms. Ankitha Philip		
Test Title: Accessing Book page and Review section			Test Execution Date: 19-10-2023		
Description: User goes library, performs AJAX search, navigates to profile and views all info					
Pre-Condition: User need to be logged in					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Navigate to login page		Login form should be displayed	Login form is displayed	Pass
2	Change theme		Changes theme to light and back to dark	Theme changes to light and then back to black	Pass
3	Provide valid username	Username : dhunan	User should be able to login	User logs in	Pass
4	Provide valid password	Password: asdASD@ 1234			

5	Click on login button				
6	User clicks 'Library' navbar item		The user should get redirected to his library	The user is redirected to his library	Pass
7	User types into search bar	Query : monk	The user's library should get filtered based on search query	The user's library gets filtered based on search query	Pass
8	User hovers over the profile icon		The user should get redirected to his profile page	The user gets redirected to his profile page	Pass
9	User clicks on Profile dropdown option				
10	User browses through all info in profile		User should be able to access user info, contact and bio	User can access user info, contact and bio	Pass
<b>Post-Condition:</b> User performs an AJAX based search and navigates to his profile where he can purview all his stored info.					

## **CHAPTER 6**

# **IMPLEMENTATION**

## 6.1 INTRODUCTION

Implementation is the stage where a planned system transforms into a tangible and operational entity. Building trust and instilling confidence in users is of paramount importance for the success of the system. This is a critical phase that places a strong emphasis on user training and the creation of informative materials. The actual transition often occurs during or after user training. Implementation signifies the act of putting a new system into action after its design phase, turning a conceptual design into a functional one.

Implementation involves the process of transitioning from the old method of operation to the new one, whether it replaces the old system entirely or makes incremental changes. Ensuring that the process is executed accurately is vital to establish a system that aligns well with the organization's requirements. System implementation encompasses the following tasks:

- Meticulous planning.
- Assessment of the existing system and its constraints.
- Designing methods to facilitate the transition.

## 6.2 IMPLEMENTATION PROCEDURES

Software implementation involves the installation of software in its intended location and ensuring that it functions as intended. In some organizations, an individual who is not directly involved in using the software may oversee and approve the project's development. Initially, there may be some skepticism regarding the software, and it's essential to address these concerns to prevent strong resistance.

To ensure a successful transition, several key steps are important:

- **Communicating Benefits:** Users need to understand why the new software is an improvement over the old one, instilling trust in its capabilities.
- **User Training:** Providing training to users is crucial to make them feel confident and comfortable using the application.

To evaluate the outcome, it is essential to verify that the server program is running on the server. If the server is not operational, the expected outcomes will not be achieved.

### 6.2.1 User Training

User training is a critical component of ensuring that individuals know how to use and adapt to a new system. It plays a vital role in fostering user comfort and confidence with the system. Even when a system becomes more complex, the need for training becomes even more apparent. User

training covers various aspects, including inputting information, error handling, querying databases, and utilizing tools for generating reports and performing essential tasks. It aims to empower individuals with the knowledge and skills needed to effectively interact with the system.

### **6.2.2 Training on the Application Software**

Once the fundamental computer skills have been covered, the next step is to instruct individuals on using a new software program. This training should provide a comprehensive understanding of the new system, including navigation through screens, accessing help resources, error management, and resolution procedures. The training aims to equip users or groups with the knowledge and skills necessary to effectively utilize the system or its components. It's important to note that training may be tailored differently for various groups of users and individuals in different roles within the organization to cater to their specific needs and requirements.

### **6.2.3 System Maintenance**

Maintaining a system's functionality is a complex challenge in software development. In the maintenance phase of the software life cycle, the software performs critical functions and operates smoothly. Once a system is operational, it requires ongoing care and maintenance to ensure its continued optimal performance. Software maintenance is a crucial aspect of the development process as it ensures that the system can adapt to changes in its environment. Maintenance involves more than just identifying and correcting errors in the code. It encompasses various tasks that contribute to the system's stability and effectiveness.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## 7.1 CONCLUSION

The analysis of the "Bookshelf" system reveals that it is a well-designed and efficient platform for online reading and book management. It addresses the need for a user-friendly online library and ebook store, providing a comprehensive set of features for readers and aspiring authors. The key features of the system, including book browsing, searching, and user account management, contribute to its effectiveness.

The "Bookshelf" project focuses on creating a user-friendly interface with features like a visually appealing homepage, easy book browsing and searching, and detailed book pages. Users can personalize their accounts, manage preferences, and access their shopping carts for book purchases. From the admin perspective, the system offers a comprehensive dashboard for managing content, users, and site settings. Admins have the authority to add, edit, or remove books, manage user accounts, and customize site settings to enhance user experience.

The "Bookshelf" system aims to provide a seamless and efficient experience for users and administrators, leveraging technology to create a comprehensive online reading platform. The successful implementation of this system can greatly benefit its target audience of readers and authors.

## 7.2 FUTURE SCOPE

The future scope for the "Bookshelf" project is filled with exciting possibilities. To promote user engagement and foster a sense of community, the introduction of a new module allowing readers to transform into amateur authors and publish their literary creations is a promising avenue. The platform can further elevate user experiences by implementing advanced features like sophisticated filtering, an expanded multilingual support system, and a comprehensive notification mechanism to keep users updated with the latest content. Enhanced analytics and promotional tools will enable the platform to refine its services and offer greater exposure to authors and their works.

Furthermore, empowering authors with chapter-wise uploads, reader interactions, and collaborative writing opportunities will catalyze a thriving literary ecosystem, creating a vibrant space for authors and readers alike. The "Bookshelf" project is on the cusp of an exciting and dynamic future, catering to the evolving needs and aspirations of its growing user base

## **CHAPTER 8**

### **BIBLIOGRAPHY**



**REFERENCES:**

- "System Analysis and Design" by Gary B. Shelly and Harry J. Rosenblatt, 2009.
- "The Pragmatic Programmer: From Journeyman to Master" by Andrew Hunt and David Thomas, Pearson India, 1st Edition (2008).
- "Agile Software Development with Scrum" by Ken Schwaber and Mike Beedle, Pearson (2008).
- "Agile Testing: A Practical Guide for Testers and Agile Teams" by Lisa Crispin and Janet Gregory, Addison Wesley Professional, 1st Edition (2008).

**WEBSITES:**

- <https://ranobes.top/>
- <https://webnovel.com/>
- <https://www.panda-novel.com/>
- <https://novelupdates.com/>

## **CHAPTER 9**

### **APPENDIX**

## 9.1 Sample Code

### Login Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>{% block title %}Login{% endblock title %}</title>
    <link rel="canonical" href="https://picocss.com/examples/sign-in/" />

    <link rel="stylesheet"
href="https://cdn.jsdelivrivr.net/npm/@picocss/pico@1/css/pico.min.css" />

    <link rel="stylesheet" href={% static 'css/custom.css' %} />
  </head>
  {% block script %}
  <script>
    document.addEventListener('DOMContentLoaded', function() {
      const usernameInput = document.querySelector('#id_username');
      const passwordInput = document.querySelector('#id_password');

      const usernameError = document.querySelector('#username-error');
      const passwordError = document.querySelector('#password-error');
      const submitButton = document.querySelector('button[type="submit"]');
      submitButton.disabled = true;
      usernameInput.addEventListener('input', validateUsername);
      passwordInput.addEventListener('input', validatePassword);
      function validateUsername() {
        const username = usernameInput.value;
        if (username.length < 5) {
          usernameError.textContent = 'Username must be at least 5
characters.';
        } else if (!/^[a-zA-Z0-9]+$/.test(username)) {
          usernameError.textContent = 'Username can only contain letters and
numbers.';
        } else {
          usernameError.textContent = '';
        }
        checkAndEnableSubmitButton();
      }

      function validatePassword() {
        const password = passwordInput.value;
        if (password.length < 8) {
          passwordError.textContent = 'Password must be at least 8 characters.';
        } else {
          passwordError.textContent = '';
        }
        checkAndEnableSubmitButton();
      }

      function checkAndEnableSubmitButton() {
        const errorMessages = [
          usernameError.textContent,
```

```

        passwordError.textContent,
    ];

    if (errorMessages.some(message => message !== '')) {
        submitButton.disabled = true;
    } else {
        submitButton.disabled = false;
    }
}
});
</script>
{% endblock script%}
<style>
.gi{
    width:16%;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
    box-shadow: 1px 2px 4px rgba(0, 0, 0, 0.25);
    transition: background-color 0.4s;
}

.gi:hover {
    background-color: #2659c0;
}
.error-message{
    color:#fc4903;
}
</style>
<body>
    <nav class="container-fluid">
        <ul>
            <li>
                <a href="." class="contrast"
onclick="event.preventDefault()"><strong>BookShelf</strong></a>
            </li>
        </ul>
        <ul>
            <li>
                <details role="list" dir="rtl">
                    <summary aria-haspopup="listbox" role="link"
class="secondary">Theme</summary>
                    <ul role="listbox">
                        <li><a href="#" data-theme-switcher="auto">Auto</a></li>
                        <li><a href="#" data-theme-switcher="light">Light</a></li>
                        <li><a href="#" data-theme-switcher="dark">Dark</a></li>
                    </ul>
                </details>
            </li>
            <li>
                <details role="list" dir="rtl">
                    <summary aria-haspopup="listbox" role="link"
class="secondary">User</summary>
                    <ul role="listbox">

```

```

        <li><a href="/user/login" >Login</a></li>
        <li><a href="/user/register" >Register</a></li>
    </ul>
</details>
</li>
</nav>

<main class="container">
    <article class="grid">
        <div>
            {% block content %}
            <hgroup>
                <h1>Log in</h1>
                <h2>To Enter the Enchanted Library: Unveil the Secrets Within, To Begin
Your Odyssey!</h2>
            </hgroup>
            <form method="POST" action="{% url 'login' %}">
                {% csrf_token %}
                <input
                    id="id_username"
                    type="text"
                    name="username"
                    placeholder="Login"
                    aria-label="Login"
                    autocomplete="nickname"
                    required
                />
                <p class="error-message" id="username-error">
                    {{ error }}
                </p>
                <input
                    id="id_password"
                    type="password"
                    name="password"
                    placeholder="Password"
                    aria-label="Password"
                    autocomplete="current-password"
                    required
                />
                <p class="error-message" id="password-error">
                </p>
                <fieldset>
                    <label for="forgotpass" id="lef">
                        <a href="/user/reset">Forgot Password</a>
                    </label>
                    <a href="/user/register" id="righ">Not registered? Register here</a>
                </fieldset>
                <button type="submit" class="contrast">Login</button>
            </form>
            <a href="/accounts/google/login/?process=login">
                
            </a>
            <a href="#">
                
            </a>
            <a href="#">
                

```

```

        </a>
    </div>
</div></div>
{% endblock %}
</article>
</main>
<script src={% static 'js/theme.js' %}></script>
</body>
</html>

```

### Views function

```

def login(request):
    error_message=""
    if request.user.is_authenticated:
        logout(request)
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            lg(request, user)
            return redirect('/user/home',{'user',user})
        else:
            error_message = "Invalid login credentials"
    return render(request, 'user/login.html',{'error':error_message})

```

### Other view functions from user module

```

from django.contrib.auth.decorators import login_required
from django.contrib.auth import logout
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserChangeForm
from django.contrib.auth import login as lg, authenticate
from django.contrib.auth.models import User
from .forms import
NewUserForm,UserBioChangeForm,CustomUserChangeForm,UserBioChangeForm2,EmailChange,Number
Change,UserProfileChangeForm
from django.contrib import messages
from django.contrib.auth.views import
PasswordResetView,PasswordResetDoneView,PasswordResetCompleteView,PasswordResetConfirmView
from django.urls import reverse_lazy
from django.utils import timezone
from django.contrib.auth.models import User
from .models import OTPModel,ShoppingCart,PurchaseHistory,Notification
from pyotp import TOTP
from django.http import JsonResponse
import secrets
from django.core.mail import send_mail
from django.contrib.auth.views import PasswordChangeView
from books.models import Books
from django.views.decorators.csrf import csrf_exempt
from django.template.loader import render_to_string
from django.http import HttpResponse
from django.template.loader import get_template

```

```

from django.template import Context
import datetime
from xhtml2pdf import pisa
from io import BytesIO
from decimal import Decimal
from django.views.decorators.cache import cache_control

class CustomPasswordResetView(PasswordResetView):
    template_name = 'user/password_reset_form.html' # Your template for the password reset form
    email_template_name = 'user/password_reset_email.html' # Your email template for the password reset
    success_url = reverse_lazy('password_reset_done') # URL to redirect after successful form submission
class CustomPasswordResetDoneView(PasswordResetDoneView):
    template_name = 'user/password_reset_done.html' # Your template for password reset done page

@login_required # Add this decorator to ensure the user is authenticated
def mark_notifications_as_read(request):
    if request.method == 'POST':
        Notification.objects.filter(recipient=request.user, is_read=False).update(is_read=True)
        return JsonResponse({'message': 'Notifications marked as read successfully'})
    else:
        return JsonResponse({'message': 'Invalid request method'}, status=400)

@login_required
def notifications_view(request):
    user = request.user
    unread_notifications = Notification.objects.filter(recipient=user, is_read=False).order_by('-timestamp')
    if unread_notifications:
        notification_messages = [notification.notification_message for notification in unread_notifications]
    else:
        notification_messages = ["No notifications at the moment"]
    # Render the notification messages to HTML
    notification_html = render_to_string('notifications.html', {'notification_messages': notification_messages})
    unread_count = unread_notifications.count()

    # Return the notification messages as a JSON response
    return JsonResponse({'notification_messages': notification_html, 'unread_count': unread_count})

def register(request):
    if request.user.is_authenticated:
        logout(request)
        context=None
    if request.method == "POST":
        form = NewUserForm(request.POST)
        if form.is_valid():
            user = User.objects.create_user(
                username=form.cleaned_data['username'],
                email=form.cleaned_data['email'],
                password=form.cleaned_data['password1'])
            otp = TOTP('JBSWY3DPEHPK3PXP').now() # Replace with your secret key
            user.is_active = False
            user.save()
            otp_record = OTPModel.objects.create(user=user, otp=otp, expires_at=timezone.now() +

```

```

timezone.timedelta(minutes=10))
    # Send OTP to user's email
    send_otp_email(user.email, otp)
    # Redirect user to OTP verification page
    return redirect('verify_otp', user_id=user.id)
    # user = authenticate(username=user.username, password=form.cleaned_data['password1'])
    # lg(request, user)
    # context= {'form': form}
    # messages.success(request, "Registration successful." )
else:
    context={'form':form}
    return render (request,"user/register.html",context)

def verify_otp(request, user_id=None):
    user = User.objects.get(id=user_id)
    otp_record = OTPModel.objects.filter(user=user, expires_at__gt=timezone.now()).first()

    if not otp_record:
        return redirect('registration') # Redirect if OTP is expired or not found

    if request.method == 'POST':
        entered_otp = request.POST.get('otp')

        if entered_otp == otp_record.otp:
            otp_record.delete() # Delete OTP record after successful verification
            user.is_active = True
            user.save()
            return redirect('login') # Redirect to login page

    return render(request, 'user/verify_otp.html')

def send_otp_email(to_email, otp):
    subject = 'Your OTP for Registration'
    message = f'Thank you for registering with our website!\n\n'
    message += f'Your One-Time Password (OTP) for email verification is: {otp}\n\n'
    message += 'Please enter this OTP on our website to complete your registration.\n\n'
    message += 'If you did not sign up for an account, you can ignore this email.\n\n'
    message += 'Thank you for choosing our service!\n\n'
    message += 'Best regards,\nBookShelf'
    from_email = 'midhunkrishnanm2024@mca.ajce.in' # Replace with your email
    send_mail(subject, message, from_email, [to_email])

class ChangePasswordView(PasswordChangeView):
    template_name = 'user/change_password.html' # Create a template for the change password page
    success_url = reverse_lazy('user_profile') # Redirect to the profile page after successfully changing the
password

def validate_username(request):
    username = request.GET.get('username', None)
    data = {
        'is_taken': User.objects.filter(username__iexact=username).exists()
    }
    return JsonResponse(data)

@cache_control(no_cache=True, must_revalidate=True, no_store=True)

```



```
@login_required
def home(request):
    # Fetch book data from the Books model
    books = Books.objects.filter(tags__name='Popular') # You can also filter or order the data as needed
    user_cart = ShoppingCart.objects.filter(user=request.user).first()
    if user_cart:
        cart_item_count = user_cart.items.count()
    else:
        # If the user doesn't have a cart, set item count to 0
        cart_item_count = 0
    context = {
        'books': books,
        'cart_item_count': cart_item_count,
    }

    return render(request, 'user/home.html', context)

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
@login_required
def user_profile(request):
    user_cart = ShoppingCart.objects.filter(user=request.user).first()
    if user_cart:
        cart_item_count = user_cart.items.count()
    else:
        # If the user doesn't have a cart, set item count to 0
        cart_item_count = 0
    return render(request, 'user/profile.html', {'cart_item_count': cart_item_count})

@login_required
def save_profile(request):
    if request.method == 'POST':
        form_name = request.POST.get('form_name')
        if form_name == 'user':
            user_form = CustomUserChangeForm(request.POST, instance=request.user)
            if user_form.is_valid():
                user_form.save()
            form = UserProfileChangeForm(request.POST, request.FILES, instance=request.user.userprofile)
            if form.is_valid():
                form.save()
            return redirect('/user/profile') # Change 'profile' to your actual profile page URL
        elif form_name == 'bio':
            allowed_fields = ('first_name', 'last_name')
            post_data = {key: request.POST[key] for key in allowed_fields}
            form = UserBioChangeForm(post_data, instance=request.user)
            if form.is_valid():
                form.save()
            allowed_fields = ('gender', 'dob')
            post_data = {key: request.POST[key] for key in allowed_fields}
            form = UserBioChangeForm2(post_data, instance=request.user.userprofile)
            if form.is_valid():
                form.save()

        elif form_name == 'contact':
```

```
        post_data = {'email': request.POST['email']}
        form = EmailChange(post_data, instance=request.user)
        if form.is_valid():
            form.save()
        post_data = {'phone_number': request.POST['phone_number']}
        form = NumberChange(post_data, instance=request.user.userprofile)
        if form.is_valid():
            form.save()

    return redirect('/user/profile')

def validate_email(request):
    email = request.GET.get('email')
    is_taken = User.objects.filter(email=email).exists()
    return JsonResponse({'is_taken': is_taken})

@login_required
@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def cart_view(request):
    # Check if the user is authenticated
    if request.user.is_authenticated:
        # Use get_or_create to retrieve the user's shopping cart or create a new one
        shopping_cart, created = ShoppingCart.objects.get_or_create(user=request.user)

        cart_items = shopping_cart.items.all()
        total_price = shopping_cart.total_price
    else:
        cart_items = [] # If the user is not authenticated, display an empty cart
        total_price = 0
    user_cart = ShoppingCart.objects.filter(user=request.user).first()
    if user_cart:
        cart_item_count = user_cart.items.count()
    else:
        # If the user doesn't have a cart, set item count to 0
        cart_item_count = 0
    context = {
        'cart_items': cart_items,
        'total_price': total_price,
        'cart_item_count': cart_item_count,
    }
    return render(request, 'user/cart.html', context)

@login_required
@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def purchase_history_view(request):
    # Get the user's purchase history
    purchase_history = PurchaseHistory.objects.filter(user=request.user).order_by('-purchase_date')
    user_cart = ShoppingCart.objects.filter(user=request.user).first()
    if user_cart:
        cart_item_count = user_cart.items.count()
    else:
        # If the user doesn't have a cart, set item count to 0
        cart_item_count = 0
    context = {
```

```
'purchase_history': purchase_history,
'cart_item_count': cart_item_count,
}

return render(request, 'user/purchase_history.html', context)

@login_required
@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def generate_receipt(request):
    if request.method == "POST":
        start_date_str = request.POST.get("start_date")
        end_date_str = request.POST.get("end_date")

        # Parse the start and end dates from the form
        start_date = datetime.datetime.strptime(start_date_str, "%Y-%m-%d")
        end_date = datetime.datetime.strptime(end_date_str, "%Y-%m-%d")

        # Query the purchase history for records within the specified date range
        purchase_history = PurchaseHistory.objects.filter(
            user=request.user,
            purchase_date__gte=start_date,
            purchase_date__lte=end_date
        ).order_by('-purchase_date')

        # Calculate the total_price of all orders within the date range
        total_price = Decimal(0)
        for history in purchase_history:
            total_price += history.total_price

        # Create a context for the receipt template
        context = {
            'purchase_history': purchase_history,
            'start_date': start_date,
            'end_date': end_date,
            'total_price': total_price, # Pass the total_price to the template
        }

        # Render the receipt template to HTML
        template_name = 'user/receipt.html'
        response = HttpResponse(content_type='application/pdf')
        response['Content-Disposition'] = f'inline;
filename="receipt_for_purchases_from_{start_date}_to_{end_date}.pdf"'

        buffer = BytesIO()
        pdf = pisa.pisaDocument(BytesIO(render_to_string(template_name, context).encode("UTF-8")), buffer)

        if not pdf.err:
            response.write(buffer.getvalue())
            return response
        response = HttpResponse(content_type='application/pdf')
        response['Content-Disposition'] = f'inline;
filename="receipt_for_purchases_from_{start_date}_to_{end_date}.pdf"'

        # Render the HTML template to PDF
```

---

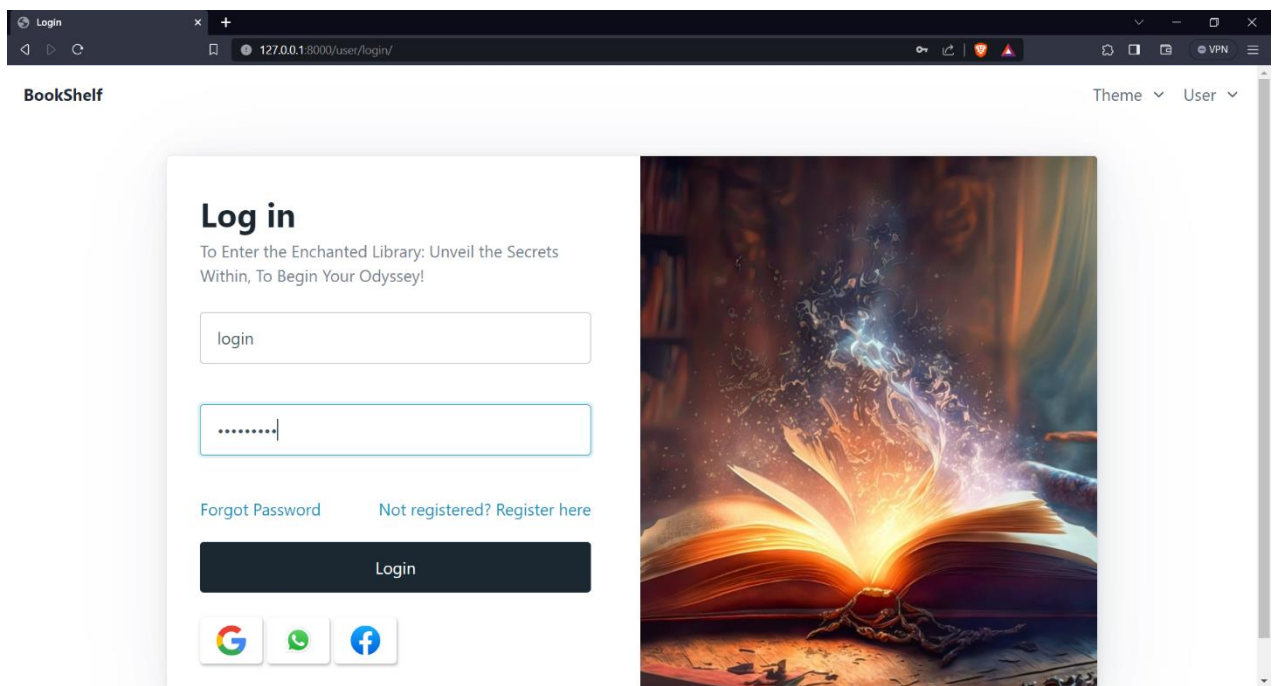
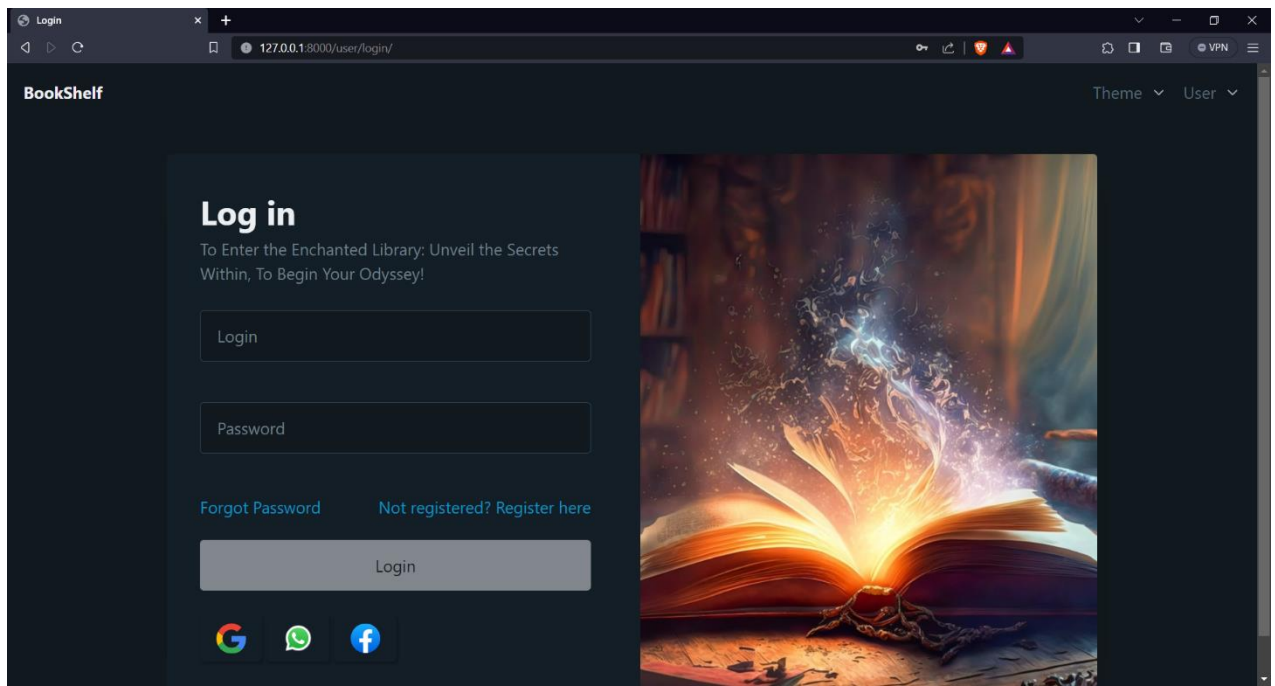
```
with open('user/receipt.html', 'r') as template_file:
    template_content = template_file.read()
    rendered_html = render(request, 'invoice.html', context)

    # Create a PDF using pisa
    pisa_status = pisa.CreatePDF(
        rendered_html.content,
        dest=response,
        link_callback=None # Optional: Handle external links
    )
    return response

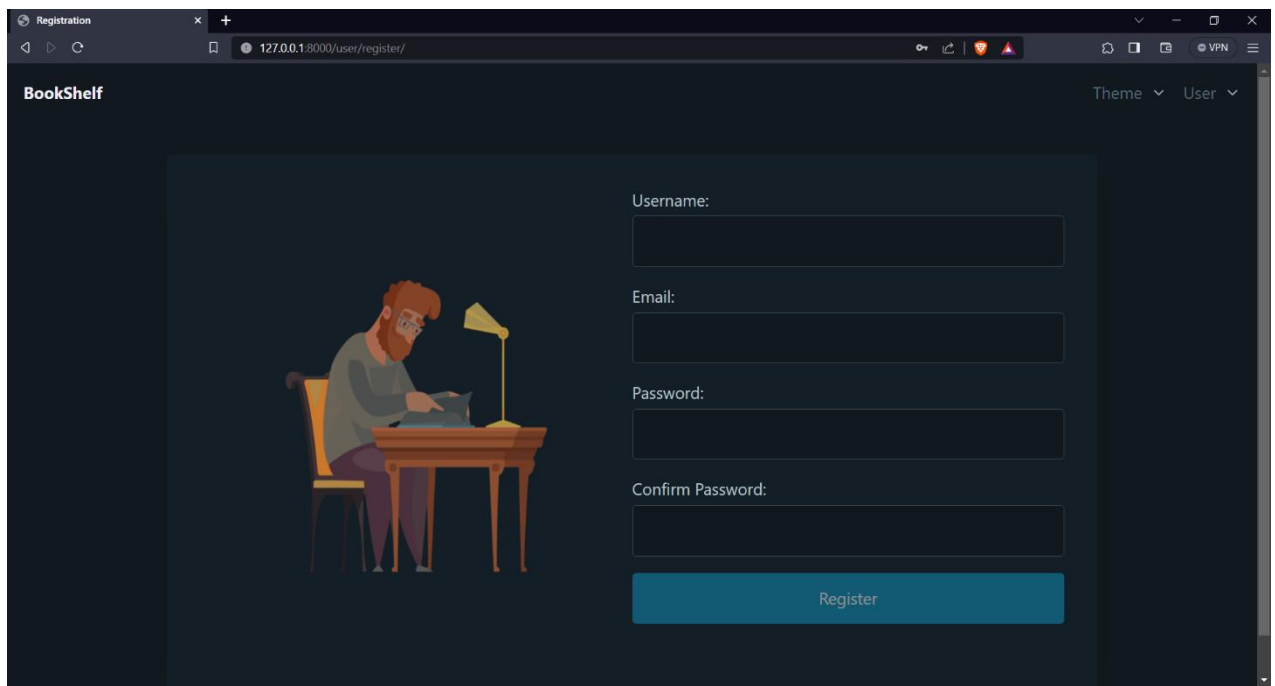
return HttpResponse("Invalid request")
```

## 9.2 Screen Shots

### Login



## Registration



Registration

127.0.0.1:8000/user/register/

BookShelf Theme User

Username:

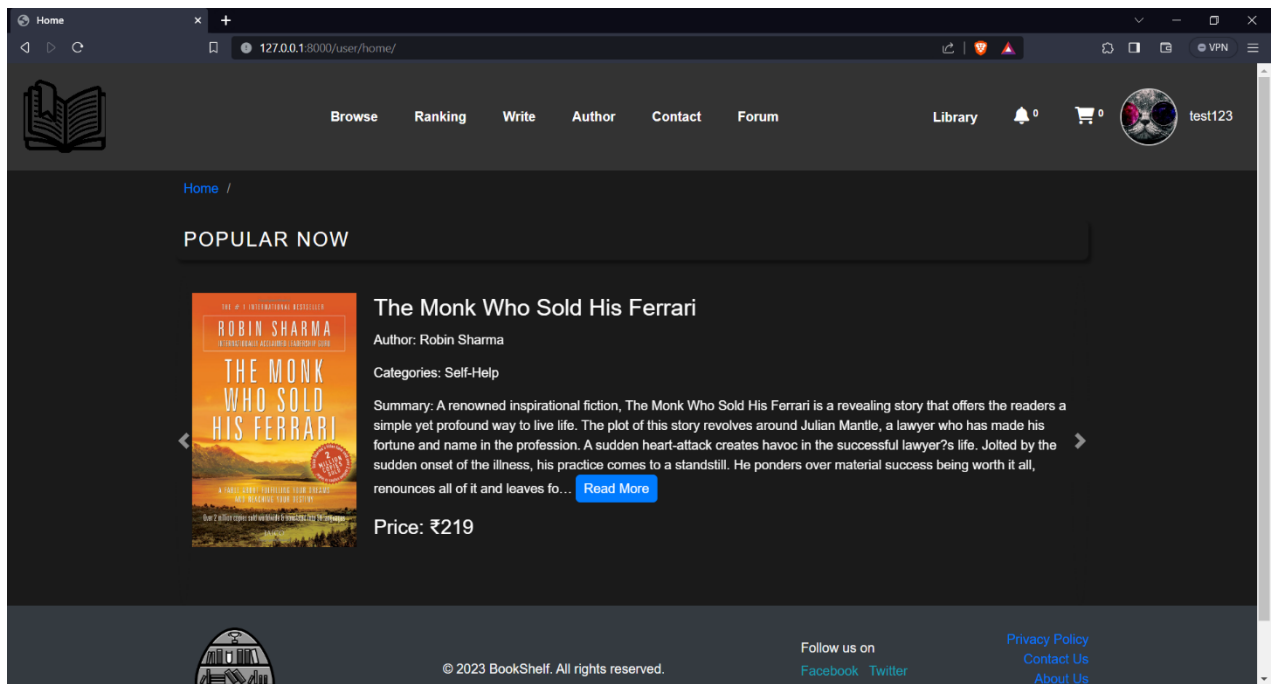
Email:

Password:

Confirm Password:

Register

## Home



Home

127.0.0.1:8000/user/home/

Browse Ranking Write Author Contact Forum

Library 0

test123

Home /

POPULAR NOW

**The Monk Who Sold His Ferrari**

Author: Robin Sharma

Categories: Self-Help

Summary: A renowned inspirational fiction, The Monk Who Sold His Ferrari is a revealing story that offers the readers a simple yet profound way to live life. The plot of this story revolves around Julian Mantle, a lawyer who has made his fortune and name in the profession. A sudden heart-attack creates havoc in the successful lawyer's life. Jolted by the sudden onset of the illness, his practice comes to a standstill. He ponders over material success being worth it all, renounces all of it and leaves fo.... [Read More](#)

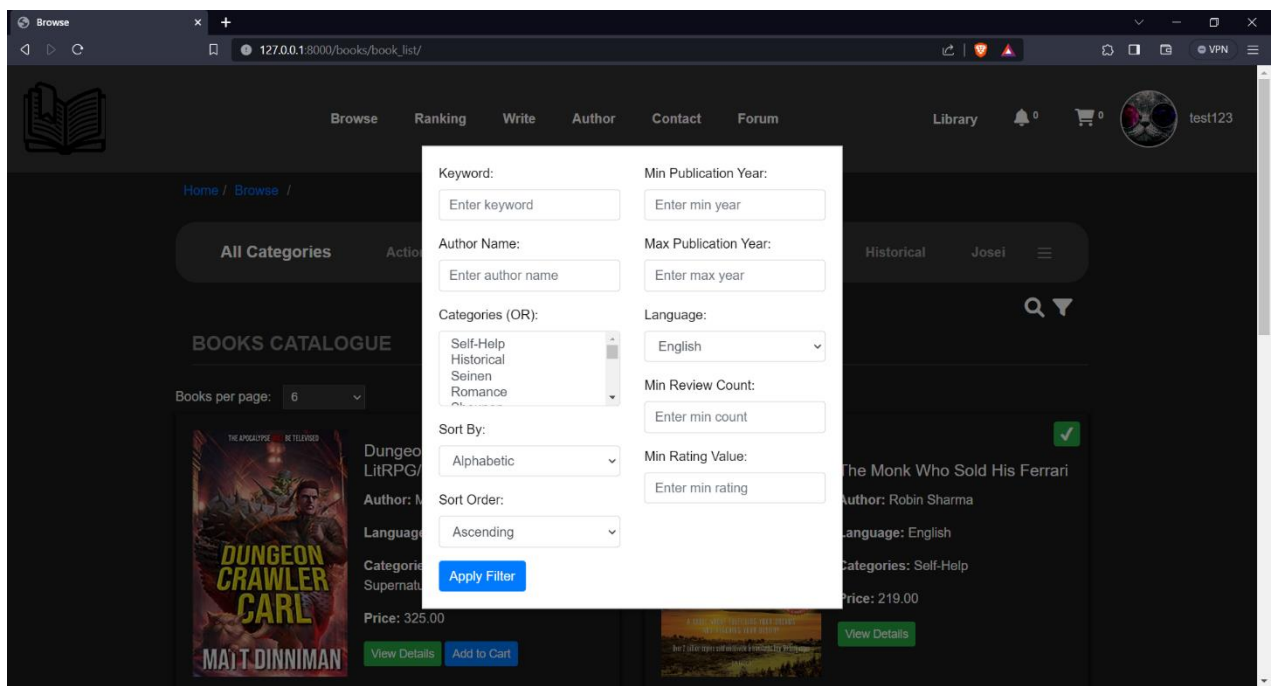
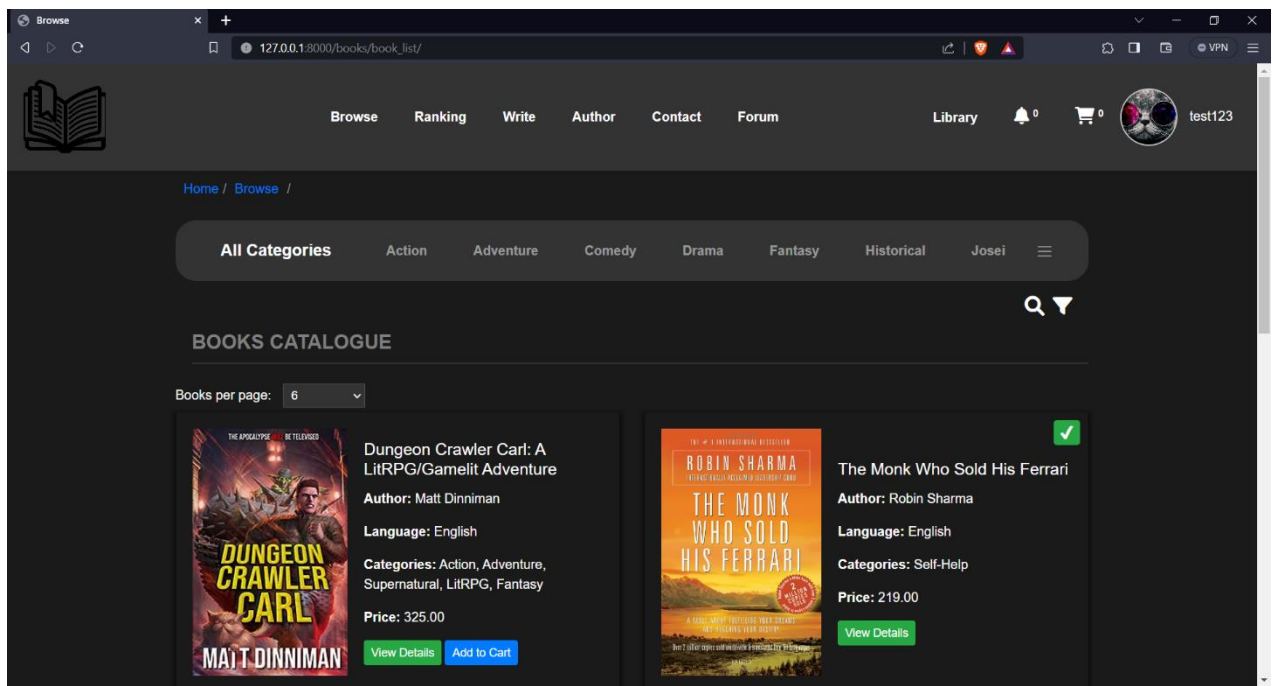
Price: ₹219

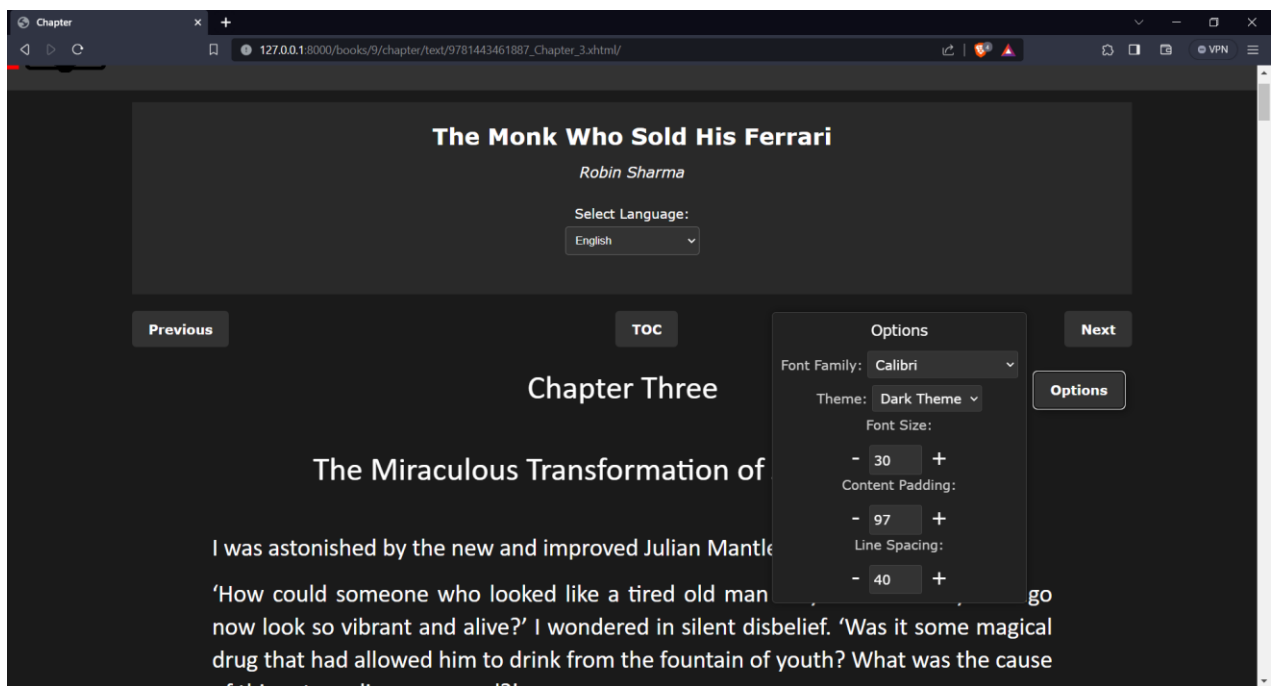
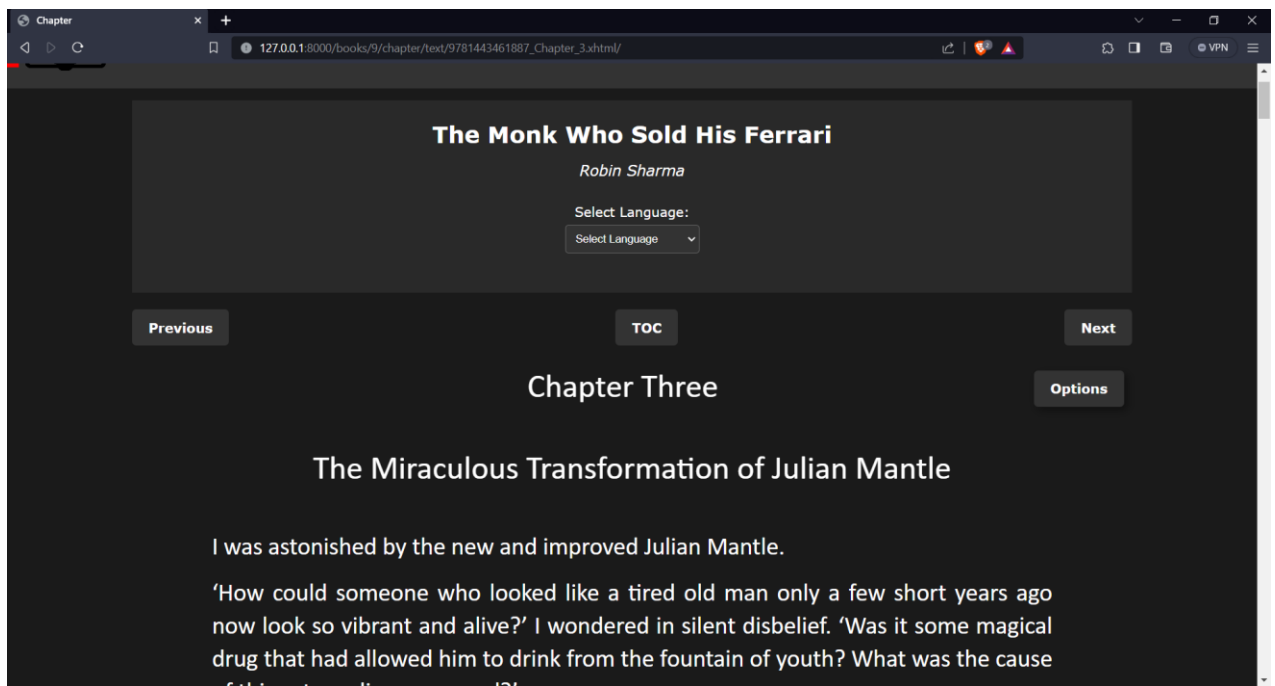
Follow us on Facebook Twitter

Privacy Policy Contact Us About Us

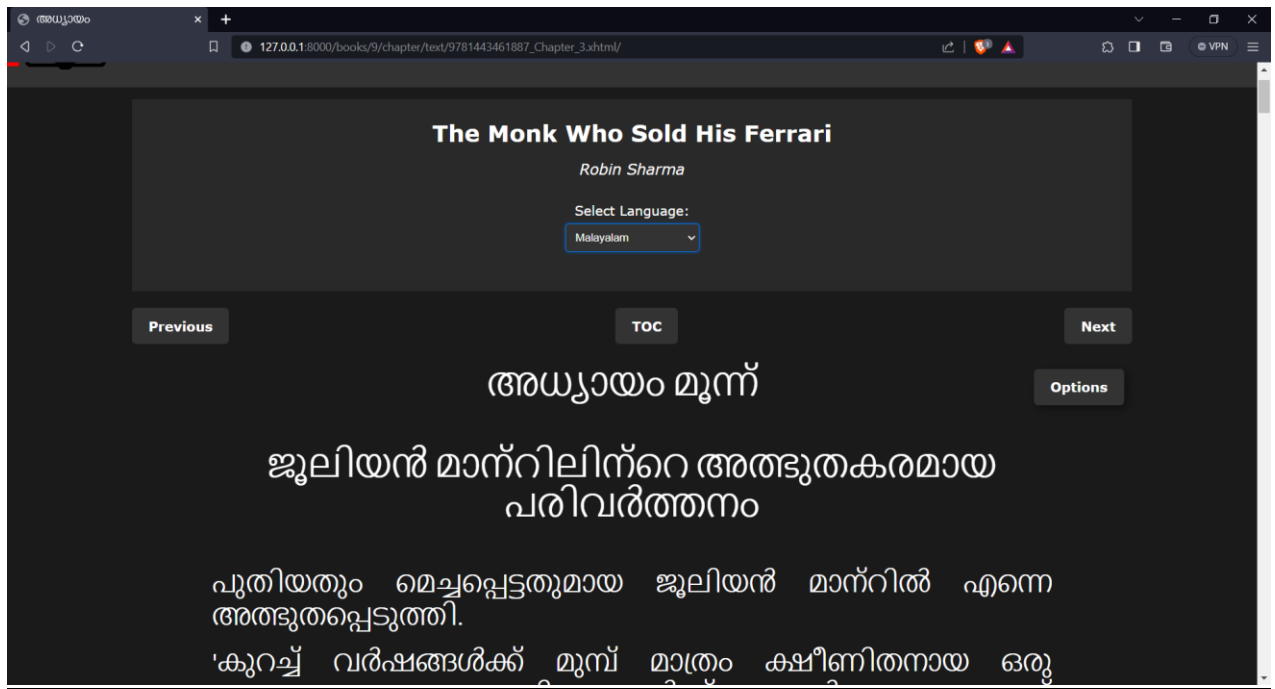
© 2023 BookShelf. All rights reserved.

## Catalogue

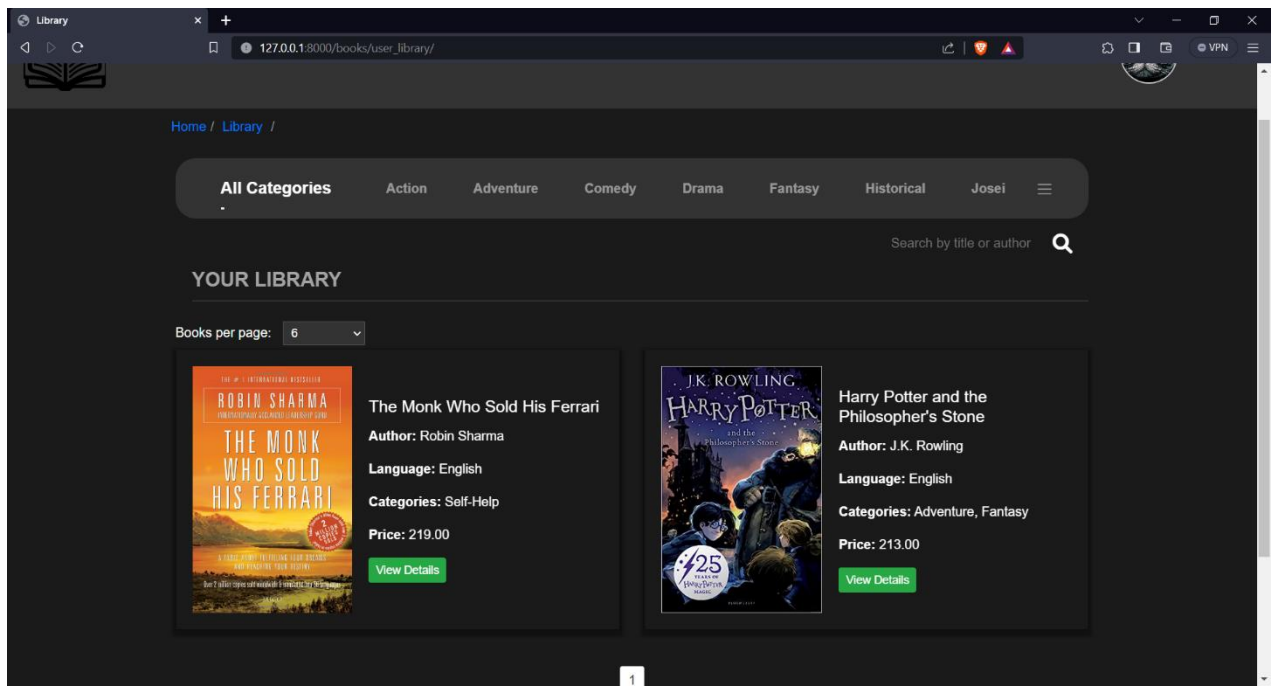


Reading page

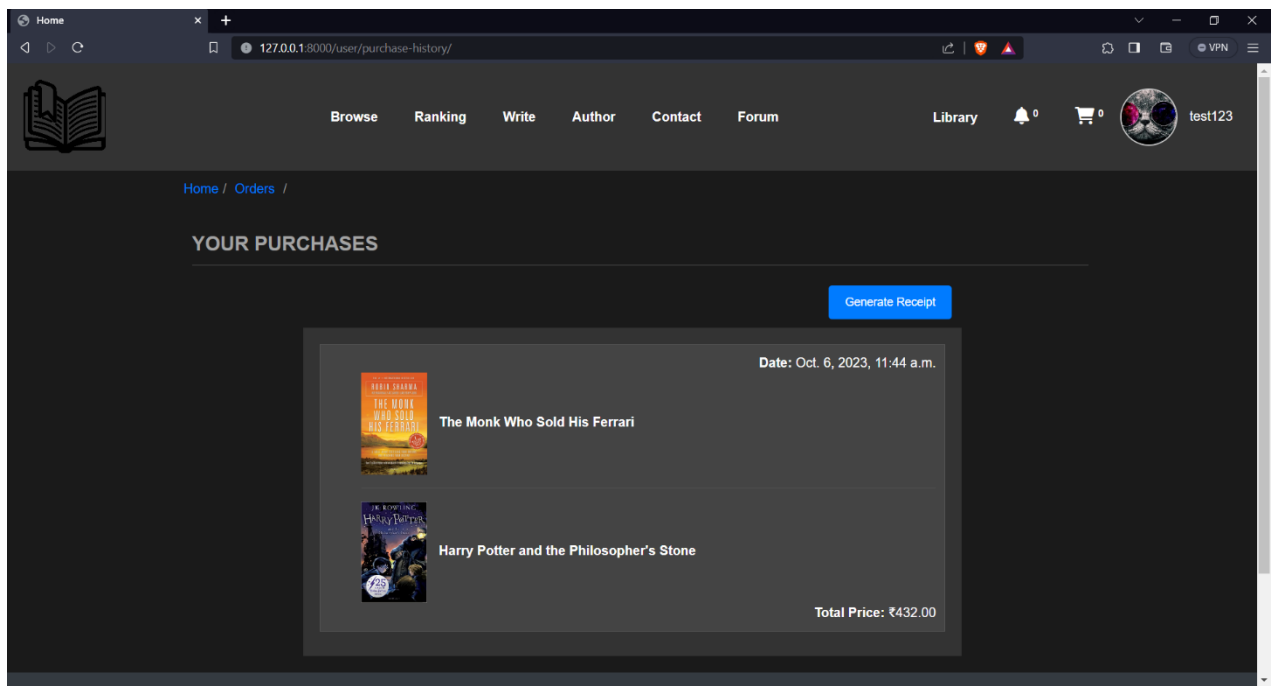




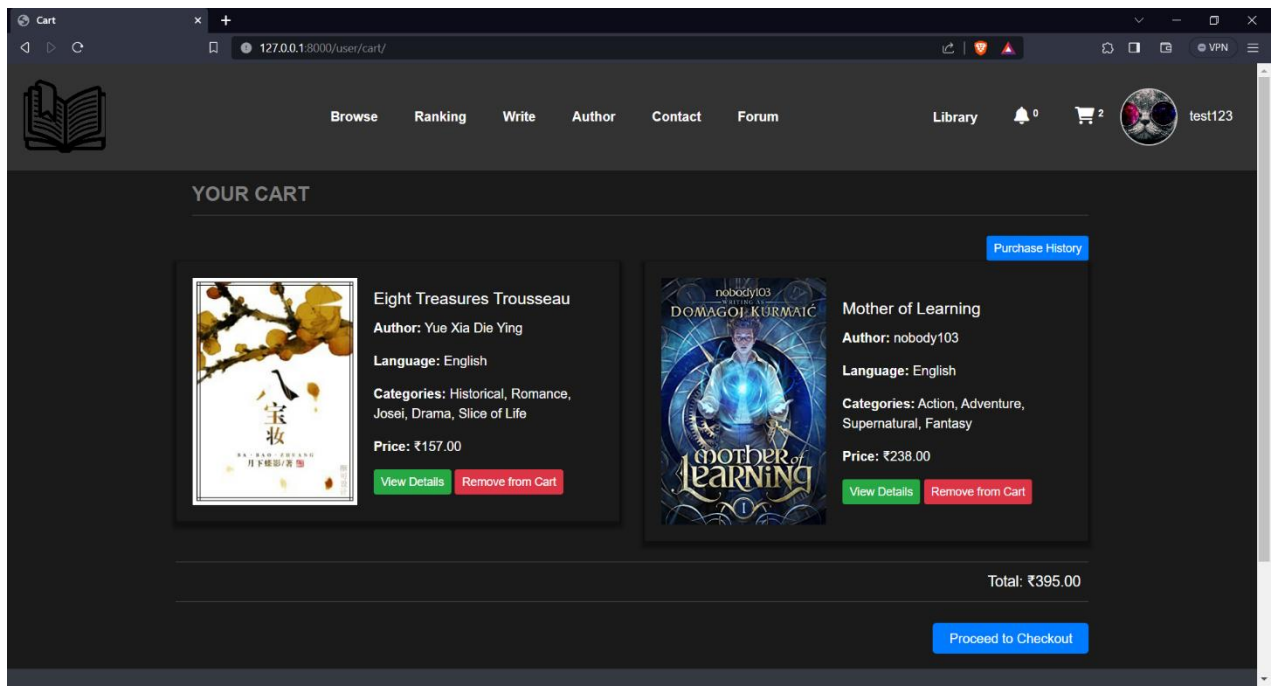
## Library



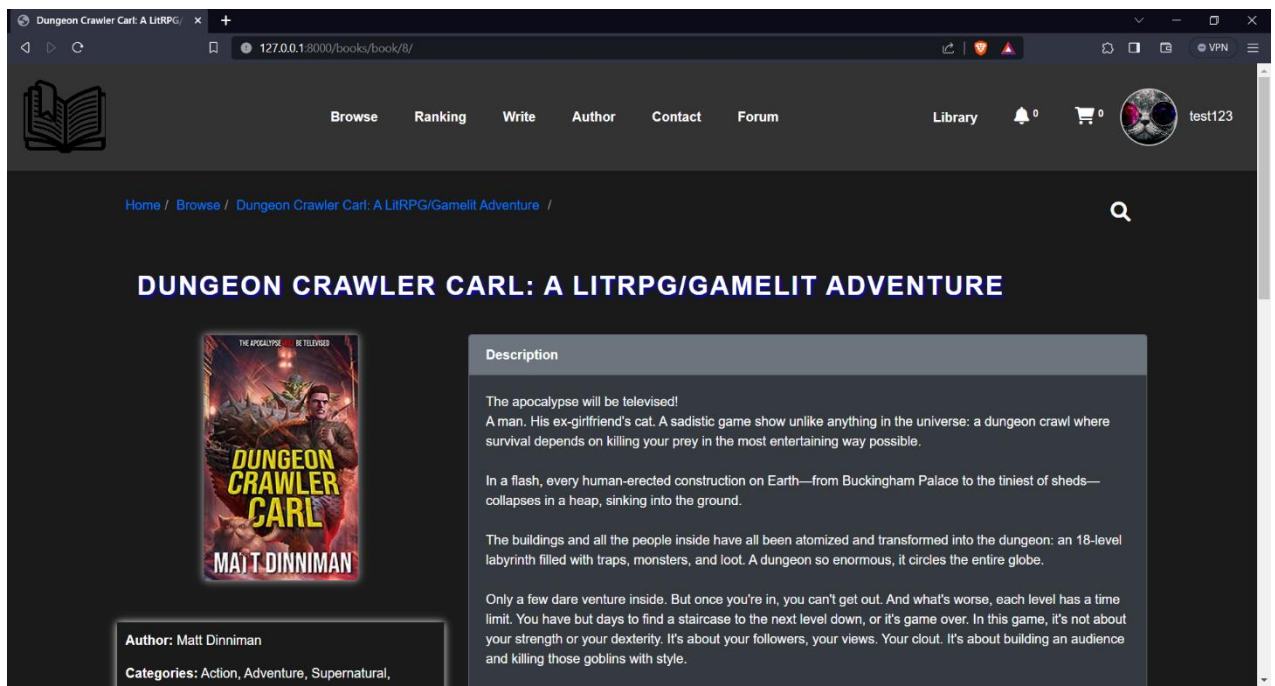
## Orders



## Cart



## Book Details



The screenshot shows the book details page for "Dungeon Crawler Carl: A LitRPG/GameLit Adventure" by Matt Dinniman. The page features a dark theme with a navigation bar at the top containing links for Browse, Ranking, Write, Author, Contact, and Forum. A user profile for "test123" is visible in the top right corner. The book cover is displayed on the left, showing a man in a hat and a cat. The title "DUNGEON CRAWLER CARL: A LITRPG/GAMELIT ADVENTURE" is prominently displayed in the center. Below the title, the author's name "Author: Matt Dinniman" and categories "Categories: Action, Adventure, Supernatural," are listed. The description on the right explains the premise: a man and his cat survive a global apocalypse by navigating a 18-level labyrinth filled with traps, monsters, and loot. The description is divided into three paragraphs, each starting with a bolded sentence.

**DUNGEON CRAWLER CARL: A LITRPG/GAMELIT ADVENTURE**

**Author:** Matt Dinniman

**Categories:** Action, Adventure, Supernatural,

**Description**

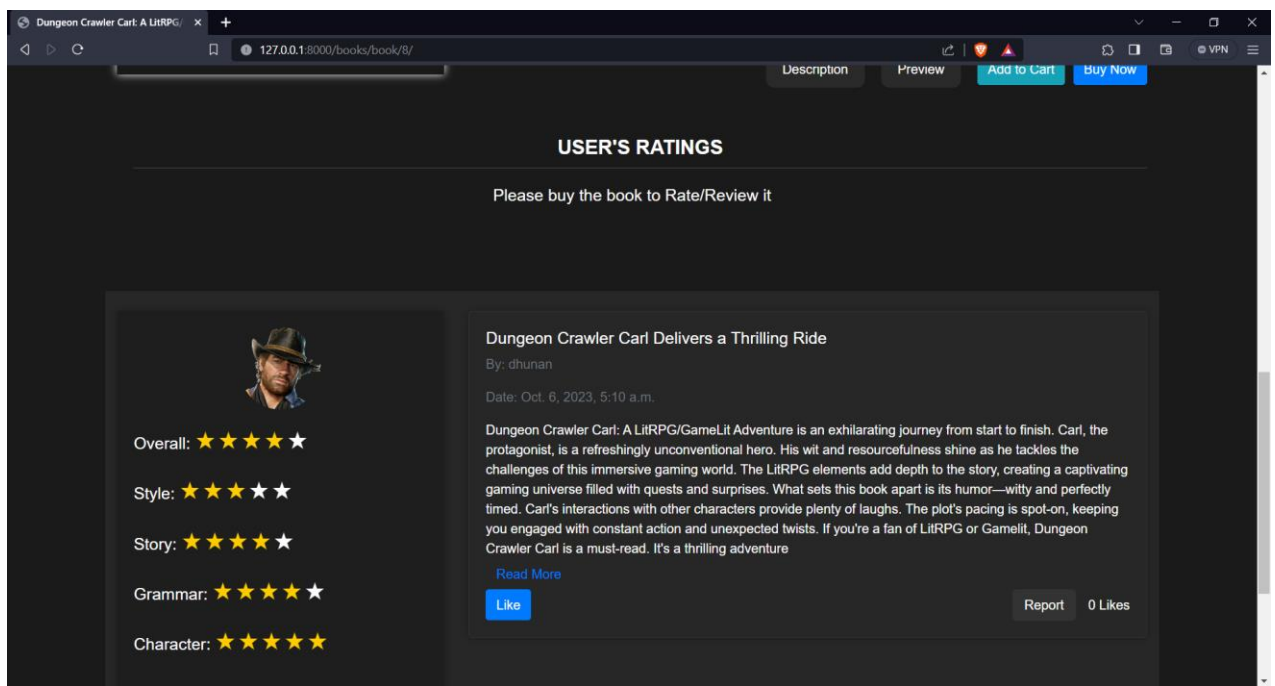
The apocalypse will be televised!

A man. His ex-girlfriend's cat. A sadistic game show unlike anything in the universe: a dungeon crawl where survival depends on killing your prey in the most entertaining way possible.

In a flash, every human-erected construction on Earth—from Buckingham Palace to the tiniest of sheds—collapses in a heap, sinking into the ground.

The buildings and all the people inside have all been atomized and transformed into the dungeon: an 18-level labyrinth filled with traps, monsters, and loot. A dungeon so enormous, it circles the entire globe.

Only a few dare venture inside. But once you're in, you can't get out. And what's worse, each level has a time limit. You have but days to find a staircase to the next level down, or it's game over. In this game, it's not about your strength or your dexterity. It's about your followers, your views. Your clout. It's about building an audience and killing those goblins with style.



The screenshot shows the user ratings section for the same book. The page has a dark theme with a navigation bar at the top containing links for Description, Preview, Add to Cart, and Buy Now. The title "DUNGEON CRAWLER CARL: A LITRPG/GAMELIT ADVENTURE" is prominently displayed in the center. Below the title, the user ratings section is titled "USER'S RATINGS" and includes a prompt "Please buy the book to Rate/Review it". The ratings are displayed as star icons for Overall, Style, Story, Grammar, and Character. A user review by "dhunan" is shown, dated Oct. 6, 2023, 5:10 a.m. The review describes the book as an exhilarating journey and a must-read for fans of LitRPG or GameLit. The review is followed by a "Read More" link, a "Like" button, and a "Report" button with "0 Likes".

**USER'S RATINGS**

Please buy the book to Rate/Review it

**DUNGEON CRAWLER CARL: A LITRPG/GAMELIT ADVENTURE**

**Overall:** ★★★★★

**Style:** ★★★★★

**Story:** ★★★★★

**Grammar:** ★★★★★

**Character:** ★★★★★

**Dungeon Crawler Carl Delivers a Thrilling Ride**

By: dhunan

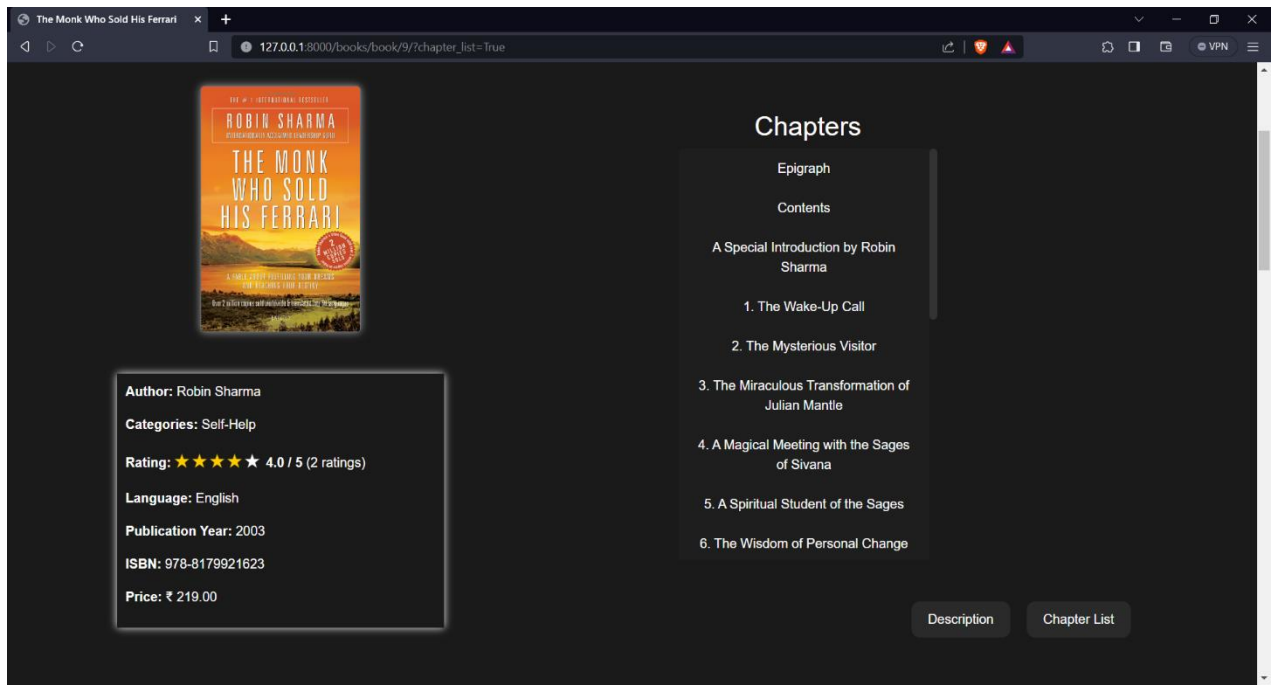
Date: Oct. 6, 2023, 5:10 a.m.

Dungeon Crawler Carl: A LitRPG/GameLit Adventure is an exhilarating journey from start to finish. Carl, the protagonist, is a refreshingly unconventional hero. His wit and resourcefulness shine as he tackles the challenges of this immersive gaming world. The LitRPG elements add depth to the story, creating a captivating gaming universe filled with quests and surprises. What sets this book apart is its humor—witty and perfectly timed. Carl's interactions with other characters provide plenty of laughs. The plot's pacing is spot-on, keeping you engaged with constant action and unexpected twists. If you're a fan of LitRPG or GameLit, Dungeon Crawler Carl is a must-read. It's a thrilling adventure

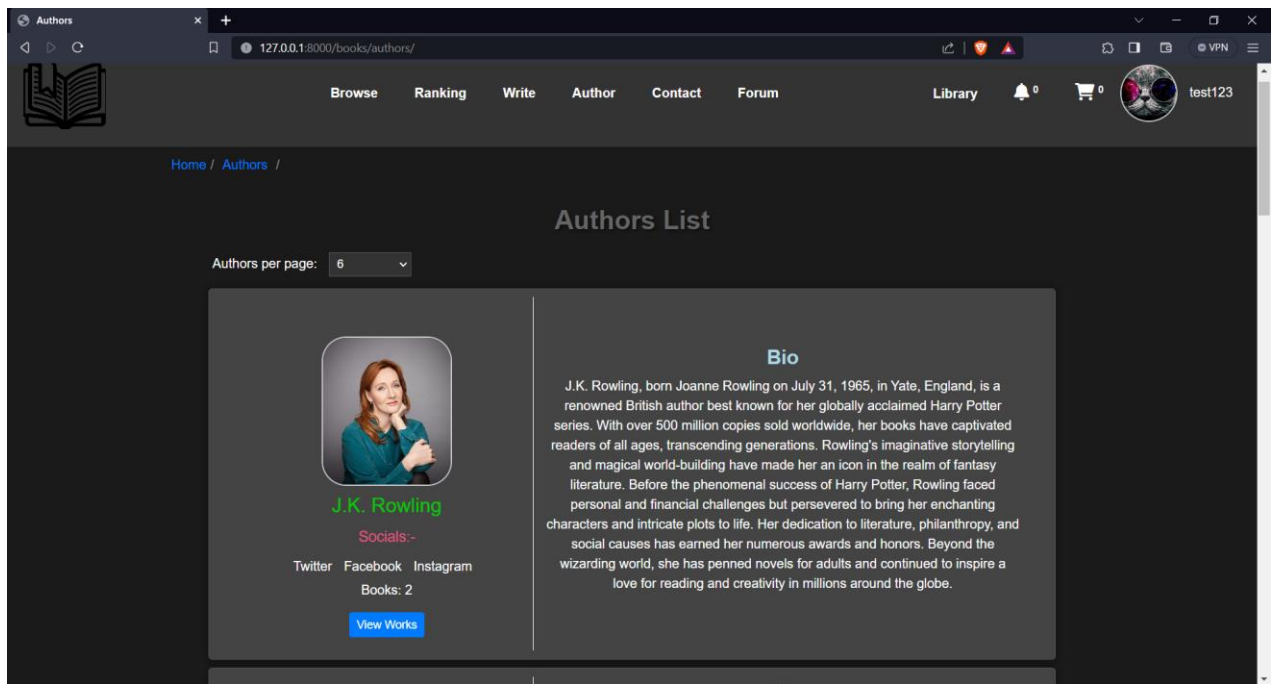
[Read More](#)

[Like](#)

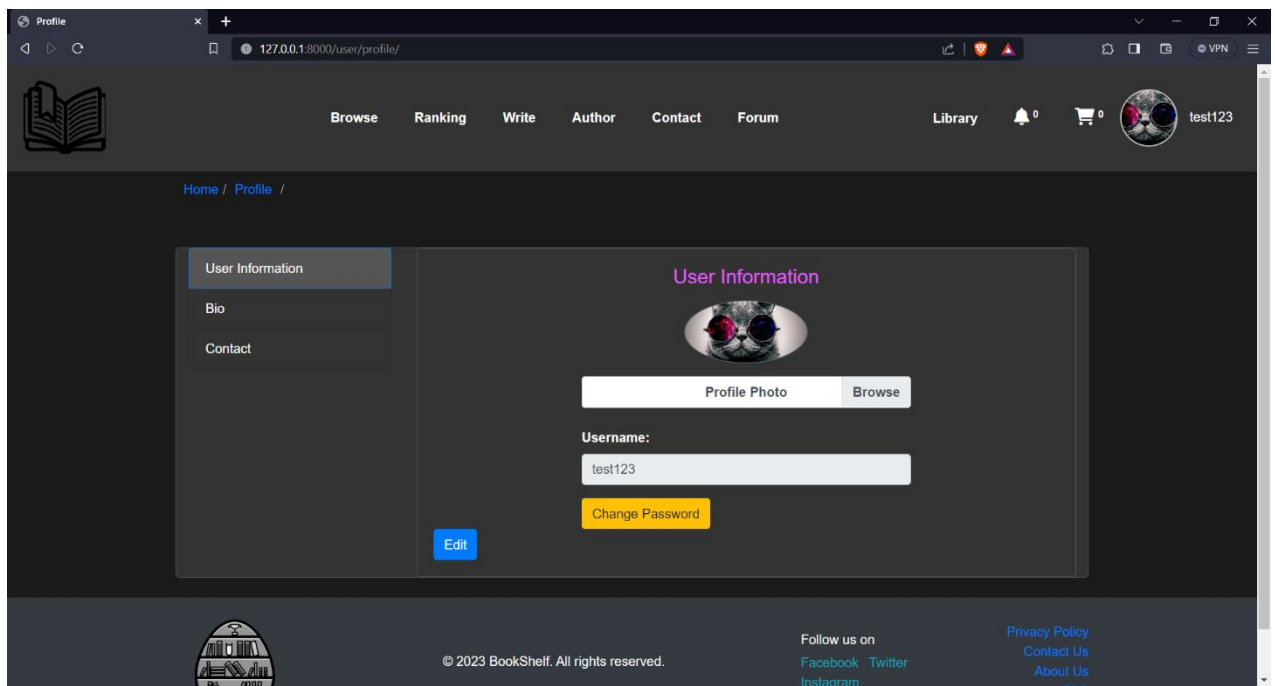
[Report](#) 0 Likes



## Authors



## Profile



## Admin Review Moderation

