

“SWAP BOOK”

Major Project Report

Submitted in the fulfilment for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

Submitted to:



RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA,

(State Technological University of M.P.)

Submitted by:

Aman Malviya 0133CS201024

Atharve Agrawal 0133CS201047

Abhishek Kumar Jha 0133CS201012

Gautam Singh Chauhan 0133CS201067

Under the Guidance of

Dr. Ritu Shrivastava (HOD CSE Department)



SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY

Department of Computer Science & Engineering

June-2024



DECLARATION

I hereby declare that the work, which is being presented in the major project, entitled " **SWAP BOOK**" Submitted in the **Department of Computer Science & Engineering, Sagar Institute of Research & Technology, Bhopal** is an authentic record of my own work carried out during the period from January 2024 - June 2024, under the guidance of **DR . RITU SHRIVASTAVA, HOD Department of Computer Science & Engineering, Sagar Institute of Research & Technology, Bhopal.**

Date:

Place: Bhopal

Aman Malviya (0133CS201024)

Atharve Agrawal (0133CS201047)

Abhishek Kumar Jha (0133CS201012)

Gautam Singh Chauhan (0133CS201067)



CERTIFICATE

This is to certify that the Major Project **“SWAP BOOK”** being is Submitted by **Aman Malviya(0133CS201024), Atharve Agrawal (0133CS201047), Abhishek Kumar Jha(0133CS201012) , Gautam Singh Chauhan (0133CS201067)**, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Computer Science and Engineering** to **RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)** during the academic year **2023-24** is a record of bona fide piece of work, carried out by her under my supervision and guidance in the **Department of Computer Science and Engineering, Sagar Institute of Research Technology, Bhopal.**

Dr. Ritu Shrivastava,

HOD CSE Department

SIRT, Bhopal

Supervisor



APPROVAL CERTIFICATE

The major project entitled **“SWAP BOOK”** being submitted by **Aman Malviya (0133CS201024), Atharve Agrawal (0133CS201047), Abhishek Kumar Jha (0133CS201012), Gautam Singh Chauhan (0133CS201067),** to **RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)** has been examined by us and is hereby approved for the award of degree **“Bachelor of Technology in Computer Science and Engineering”** in the academic year 2023-24

Prof. Mohit Tomar,

(Internal Examiner)

Date:

(External Examiner)

Date:



ACKNOWLEDGEMENT

This is one of the best moments of my B. Tech program to publicly acknowledgment those who have contributed in many different ways to make my success a part of their own. The completion of the Major Project depends upon the co-operation, coordination and combined effects of several resources of knowledge energy.

We heartily thank to **Dr Ritu Shrivastava**, HOD CSE Department (SIRT, Bhopal) who has supported in Major project, faculties of the Department of Computer Science & Engineering, for accepting me to work under their Valuable Guidance, Closely Supervised this work over the past few months and offering many innovative ideas and helpful suggestions, which led to the successful completion of this dissertation work.

We especially thankful **Dr. Rajiv Srivastava**, Director (SIRT, Bhopal) for his kind co-operation and rendering me all possible the facilities.

Last but certainly not least, I want to express my profound thanks to my family. Their unwavering moral support was the pillar of strength that made it possible for me to persevere and successfully complete this major project. Their constant encouragement and belief in my abilities were the driving force behind my determination.



We're thankful to all staff members of the CSE department and my friends for their timely help co-operation and suggestion during my project work. Lastly but not the least, we must express thanks to my family, without their moral support it was impossible for me to complete this work.

Aman Malviya (0133CS201024),

Atharve Agrawal (0133CS201047),

Abhishek Kumar Jha (0133CS201012)

Gautam Singh Chauhan (0133CS201067).



Table of contents	Page No.
Title Page	1
Declaration Of Student	2
Certificate	3
Approval Certificate	4
Acknowledgement	5-6
List Of Figures	9-10
List Of Tables	11
Abstract	12
Chapter 1 Introduction	13
1.1 Objective	14-15
1.2 Scope	16-18
1.3 Purpose	19-20
1.4 Problem Statement	21-22
1.5 Literature Survey	23-25
Chapter2 Design	26
2.1 E-R Diagram	27-30
2.2 Data Flow Diagram	31-34
2.3 Use Case Diagram	35-75
2.4 Class Diagram	76-80
2.5 Sequence Diagram	81-97
2.6 Collaboration Diagram	98-103
2.7 Activity Diagram	104
2.8 Component Diagram	105
2.9 Deployment Diagram	106
2.10 Flow Chart Diagram	107
2.11 State Chart Diagram	108
2.12 Gantt Chart Diagram	109



Chapter3 Implementation Requirements	110
3.1 Front-End	111-113
3.2 Back-End	114-116
Chapter4 Lay-out	117
4.1 Snapshots	118-124
4.2 Test Cases	125-130
Chapter5 Application	131
5.1 Advantage(s)	132
5.2 Disadvantage(s)	133
5.3 Application(s)	134-135
Chapter 6 Conclusion and Future Work	136
6.1 Conclusion	137-138
6.2 Future Work	139-140
Chapter 7 Implementation	141
7.1 Code	142-164
7.2 Database Snippets	165-166
References	167-168



List of Figures

Fig No	Figure Name	Page No
2.1	ER-Diagram	27
2.2	Data Flow Diagram	31
2.3	Use Case Diagram	35
2.3.1	Register Use Case Diagram	36
2.3.2	Admin User Case Diagram	40
2.3.3	Home Page User Case Diagram	44
2.3.4	Book Page User Case Diagram	48
2.3.5	Writer's Page Use Case Diagram	51
2.3.6	Swap Page Use Case Diagram	55
2.3.7	Extended Book Page Use Case Diagram	59
2.3.8	Writer's Page Use Case Diagram	63
2.3.9	Swap's Page Use Case Diagram	68
2.3.10	Extended Use Case Diagram	71
2.3.11	Crud Operation Use Case Diagram	73
2.4	Class Diagram	76
2.5	Sequence Diagram	81
2.5.1	Create Book Sequence Diagram	82
2.5.2	Update Book Sequence Diagram	83
2.5.3	Delete Book Sequence Diagram	84
2.5.4	Details Book Sequence Diagram	85
2.5.5	List Book Sequence Diagram	86
2.5.6	List Books Owned Sequence	87
2.5.7	Swap Sequence Diagram	88
2.5.8	Update Swap Sequence Diagram	89
2.5.9	Delete Swap Sequence Diagram	90
2.5.9	Details Swap Sequence Diagram	91



2.5.10	List Swap Sequence Diagram	92
2.5.11	List Swap Requested Sequence Diagram	93
2.5.12	List Swap Requested from Me Sequence Diagram	94
2.5.12	List Profile Sequence Diagram	95
2.5.13	Update Profile Sequence Diagram	96
2.514	List Message Sequence Diagram	97
2.6	Collaboration Diagram	98
2.6.1	Registration Collaboration Diagram	99
2.6.2	Add Book Collaboration Diagram	100
2.6.3	Book Browsing Book Collaboration Diagram	101
2.6.4	Swapping Collaboration Diagram	102
2.6.5	Chatting Collaboration Diagram	103
2.7	Activity Diagram	104
2.8	Component Diagram	105
2.9	Deployment Diagram	106
2.10	Flow – Chart Diagram	107
2.11	State – Chart Diagram	108
2.12	Gantt Chart Diagram	109
4.1.1	Home Page Snapshot	118
4.1.2	Home Page Books Segregated on Basis of Genre Snapshot	119
4.1.3	Login Page Snapshot	120
4.1.4	Register Page Snapshot	121
4.1.5	About Page Snapshot	122
4.1.7	Check Out Page Snapshot	123



List of Tables

Table No	Table Name	Page No
4.2.1	Test Case 1	125
4.2.1.1	Test Case 1.1	126
4.2.2	Test Case 2	127
4.2.3	Test Case 3	127
4.2.4	Test Case 4	128
4.2.5	Test Case 5	128
4.2.6	Test Case 6	129
4.2.7	Test Case 7	129
4.2.8	Test Case 8	130
4.2.9	Test Case 9	130



ABSTRACT

Introducing SWAP BOOK, a dynamic platform that unites passionate readers and talented writers. Here, you can effortlessly delve into a rich collection of Mystery, Fiction, Non-Fiction, Fantasy, and more. What sets us apart is our vibrant community, where readers and writers engage in meaningful interactions.

We've revolutionized the way you enjoy books. No more constant purchases – with our innovative swapping feature, you can exchange books with fellow users when your interests align. It's a game-changer for avid readers.

For writers SWAP BOOK offers a unique opportunity. Upload your creations and earn coins as a token of appreciation. It's a platform that values your talent.

Behind the scenes, we've harnessed the power of Node.js and MongoDB for a seamless experience. The interface, designed with HTML and CSS, ensures a user-friendly journey.

Join us in this literary adventure. Explore, connect, and embrace the world of books



SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY Bhopal

Department of Computer Science & Engineering

CHAPTER 1: INTRODUCTION



1.1Objective:

The Primary Objective of "Swap Book":

The primary objective of our project, "Swap Book," is to create an innovative and user-friendly web application that addresses a significant issue in the reading community. This platform is designed to cater to book enthusiasts who often find themselves with a surplus of old books, many of which are rarely, if ever, touched after their initial reading.^[15] It's not uncommon for bookshelves to become repositories for untouched books, resulting in wasted resources and untapped knowledge.

The Problem We Aim to Solve:

Our project recognizes that this is a common dilemma among avid readers. As they accumulate books over the years, they are faced with the dilemma of what to do with books they've already read, books they might not read again, or books that they have outgrown.^[18] This issue not only leads to clutter but also contributes to the environmental impact of book production and disposal. Each book embodies a significant number of resources, from the paper and ink used in printing to the energy expended in distribution. The environmental costs of book production, in terms of tree cutting, water usage, and carbon emissions, are substantial.

The Solution We Offer:

"Swap Book" aims to address this problem by providing a digital platform that offers a convenient and sustainable solution. It's not just about creating a marketplace for used books; it's about transforming the way we think about books and the value they hold. The platform facilitates the exchange and sale of



these books, ultimately extending their utility and promoting a culture of sustainable reading.

Sustainability and Environmental Impact:

In a world where environmental concerns are increasingly relevant, reducing the waste associated with book production and disposal is not just a luxury but a necessity. "Swap Book" contributes to this cause by encouraging users to rehome their old books. By doing so, we not only declutter our physical spaces but also minimize the ecological footprint of the reading community. This shift towards reusing and recycling books aligns with the principles of sustainability, waste reduction, and responsible resource management.

The Value of Shared Knowledge:

Beyond the environmental benefits, "Swap Book" emphasizes the inherent value of shared knowledge and community-building. When users exchange or sell their books, they contribute to the continuous flow of ideas and stories.^[16] A book that sits idly on a shelf is a missed opportunity for someone else to experience its content. "Swap Book" bridges the gap between readers who have books to share and those who seek new titles to explore.^[15] It promotes a culture of shared knowledge, where books find new homes and new readers.

In essence, "Swap Book" is more than just a digital platform; it's a project with a purpose. It addresses a real and common issue in the reading community while aligning with the evolving values of environmental responsibility, community building, and the enduring love of literature.

By using "Swap Book," users not only discover new stories but also become active contributors to a more sustainable and connected reading culture.



1.2 Scope:

The scope of the "Swap Book" project is ambitious, yet well-defined, and it encapsulates the entire book exchange process, catering to the needs of book enthusiasts who seek to declutter their shelves, discover new reading material, and engage in secure transactions. Our goal is to create a comprehensive web application that seamlessly connects users and facilitates the exchange or sale of books.

User Registration and Book Listings:

Users are provided with the functionality to register and create listings for the books they wish to exchange or sell.^[18] This registration process is not just about creating an account but about becoming a part of a larger community of readers, united by their love for books. To add a listing, users can provide book details, including the book's title, author, condition, and a brief description, enabling potential readers to get a sense of the book's quality and relevance.

Efficient Book Discovery:

Book enthusiasts often seek specific titles or genres. With "Swap Book," users can search for books they desire effortlessly. The platform offers a robust search and filtering system, allowing users to discover books based on various criteria, including title, author, genre, or even the location of the book. This streamlined process connects readers with books they're passionate about.

Secure Transactions:

Creating a secure and reliable environment for book exchanges is of paramount importance. The scope includes a secure transaction system that enables users to communicate and negotiate the terms of the exchange. Users can have confidence in their transactions, knowing that they can trust their peers and the quality of the books they're receiving. This not only fosters a sense of trust but also contributes to the growth of a thriving book-sharing community.



Technical Implementation:

To achieve this scope, we've harnessed a multitude of tools and technologies that span the entire project's stack. The user interface design is integral to the project's success. It has been meticulously crafted using industry-standard design tools such as Adobe XD and Figma. This meticulous design ensures that "Swap Book" offers a visually appealing and user-friendly experience, enhancing engagement and usability.

Front-End Technologies:

On the front-end, we've employed a stack of web technologies. HTML is used for structuring the web pages, CSS for styling, and JavaScript for interactivity. This combination breathes life into the user interface, allowing users to seamlessly interact with the platform. The front-end is a crucial aspect of "Swap Book" as it's the user's gateway to the entire exchange process.

Robust Back-End:

The back-end, powered by Node.js, is robust and capable of handling user data, transactions, and communication between users. Node.js's non-blocking, event-driven architecture ensures that "Swap Book" remains responsive even under heavy loads, contributing to the platform's reliability and performance.

Efficient Data Management:

In the realm of data management, MongoDB serves as the backbone of the platform, ensuring efficient and reliable database operations. MongoDB's NoSQL database design is particularly suited for the flexible data structure of book listings and user profiles.



User-Centric Focus:

Our scope extends beyond the technical aspects of the application; we also prioritize user experience and security. It's essential to create an environment where users can confidently exchange their books without concerns about the condition of the books or trustworthiness of their peers. To address this, we've implemented features that allow users to rate and provide feedback on their exchange experiences, creating a self-regulating and trustworthy community.

This comprehensive scope, covering both the technical and user-oriented aspects, sets "Swap Book" on a path to fulfill its objectives effectively. It aims not only to be a platform for exchanging books but also to foster a culture of trust, community, and sustainable reading practices among book enthusiasts.



1.3 Purpose:

"Swap Book" isn't just a web application; it's a purpose-driven project that addresses a pressing concern faced by book readers worldwide. The purpose of our project goes beyond the digital realm; it encompasses broader social and environmental objectives.

Alleviating Book Hoarding:

First and foremost, "Swap Book" aims to alleviate the issue of book hoarding, a common problem encountered by book enthusiasts. Over time, many individuals accumulate a significant number of books, a substantial portion of which end up collecting dust on bookshelves, forgotten and unloved. This situation not only leads to wasted resources but also deprives these books of their true purpose - to be read, cherished, and shared. Every unread book represents an untold story and unexplored knowledge, and "Swap Book" steps in to repurpose these forgotten books.

By offering users the opportunity to list, exchange, or sell books they no longer need, "Swap Book" provides these neglected books with a second chance to be read, appreciated, and valued by new readers. This not only benefits the books themselves but also the individuals who contribute to this culture of sharing. The project aims to bridge the gap between book hoarders and those seeking new titles to explore, fostering a sense of camaraderie among book lovers.

Environmental Consciousness:

The second aspect of our project's purpose is deeply intertwined with the environmental consciousness of our times. "Swap Book" seeks to reduce the ecological impact associated with book production and disposal. Each book that goes to waste represents not only a loss of knowledge and stories but also a burden on our environment. ^[16] The paper and resources used in book



production, along with the energy consumed in transportation and distribution, contribute to environmental degradation.

"Swap Book" promotes the eco-friendly concept of sharing and reusing. By encouraging book exchange and reducing the demand for new books, we indirectly contribute to lessening the environmental burden placed on forests, reducing energy consumption, and lowering carbon emissions. In doing so, we align with global efforts to combat deforestation and mitigate climate change.

In essence, the purpose of our project transcends mere technological innovation; it's about fostering a sense of community among book lovers, promoting sustainable practices, and making books more accessible and cherished. "Swap Book" is not just about bytes and code; it's about rekindling the love for books and stories, connecting readers, and creating a sustainable and environmentally responsible reading culture.

The project's significance goes beyond its technical functionality; it's about making a positive impact on individuals, communities, and the planet we all call home. It's about reminding the world that the magic of books lies not just in their pages but in the shared experiences they offer. "Swap Book" aspires to be a catalyst for change, uniting book enthusiasts, reducing waste, and contributing to a more sustainable and harmonious world.



1.4 Problem Statement:

The problem we are addressing with "Swap Book" is a universal and timeless concern that resonates with virtually every book reader, regardless of their age, location, or reading preferences. It revolves around the accumulation of unused books, a phenomenon experienced by both avid bibliophiles and occasional readers alike. As people delve into the world of literature, their bookshelves inevitably expand, often beyond their control. In the enthusiasm of acquiring new titles, old books are left behind, untouched and unappreciated. This accumulation leads to clutter and, more importantly, an underutilization of valuable literary resources.

Beyond the clutter and personal inconvenience, the core of this issue lies in the fact that many books are languishing on bookshelves, serving no purpose and contributing to environmental waste. Every book represents a wealth of knowledge, stories, and ideas, and when they sit idle, it's a missed opportunity for learning and enjoyment. It's a waste not only of physical space but also of the potential for intellectual and emotional enrichment.

It's worth noting that this issue isn't limited to individual readers; it has broader societal and environmental implications. The production of books, from paper to ink, consumes valuable resources and energy. When books are discarded or left untouched, it adds to the environmental burden, as the materials used in book production go to waste. The disposal of unwanted books can lead to increased landfill waste, contributing to environmental degradation.

"Swap Book" is more than just a solution to personal clutter; it addresses these broader concerns. Our platform acknowledges that a significant portion of these unused books could find new life in the hands of other readers. What if those forgotten books could be made accessible to individuals who wish to read them, creating a win-win situation for all parties involved?



Our platform acts as the bridge that connects these two groups - those who have books they no longer need and those who are eager to read new titles. By facilitating book exchanges, we're turning a common issue into an opportunity for shared reading, resource optimization, and environmental responsibility.

With "Swap Book," we're not just solving a personal problem; we're promoting a cultural shift toward a more sustainable and sharing-oriented approach to reading. We're encouraging readers to connect with one another, to share their passion for literature, and to reduce the environmental footprint associated with book production and disposal. In doing so, we're not only creating a practical solution but also contributing to a larger movement of sustainable.



1.5 Literature Survey:

In Western Culture,

eBooks and digital technology can be very engaging for reluctant readers. The National Literacy Trust's 2015 study of children's access to eBooks found In Western culture, the adoption of eBooks and digital technology has ushered in an era of engagement for readers, especially for those who might have been considered reluctant readers in traditional settings. A notable study by the National Literacy Trust in 2015 shed light on the significant positive impact of digital reading on children. The findings revealed that boys' reading levels increased by an average of 8.4 months, compared to the 7.2 months' progress made by girls. Additionally, the percentage of boys who felt that reading was a difficult task almost halved from 28.0% to 15.9%, indicating that their confidence in their reading abilities significantly improved as a result of the digital reading initiative. Equally remarkable was the surge in the percentage of boys who considered reading as "cool," rising from 34.4% to 66.5%.^[15] This data highlights how embracing digital technology and eBooks can enhance reading experiences and boost confidence among young readers in Western cultures.

However, the scenario in India presents a distinct perspective, as revealed by the Tata Literature Live! Survey in 2014. The survey results showed that a substantial 78% of respondents, spanning various age groups, strongly favoured printed books over electronic reading. This preference for physical books is deeply ingrained in Indian culture, where the tactile experience of flipping through pages and the rich tradition of storytelling have long been cherished. Despite the worldwide digital shift, the affinity for printed books remains deeply rooted in the Indian reading community.

In the midst of this divergence, there arises a unique opportunity to bridge the gap and cater to the preferences of both traditional book lovers and those who



are open to embracing digital reading. "SWAP BOOK" is an endeavour born from the recognition of this divergence. It's a platform designed to bring about a transformation in the mindset of young readers in India through an easily accessible and user-friendly digital interface.

Drawing inspiration from various successful platforms, "SWAP BOOK" has aimed to create a unique blend of engaging digital reading and community building, tailored to the Indian context. Several referential websites have played a pivotal role in shaping the foundation of "BOOK IT":

1. Wattpad: Wattpad stands out as an online literature platform that empowers users to both read and write original stories. Its founders, Allen Lau and Ivan Yuen, had a vision to create social communities around stories and break down the barriers between readers and writers. This community-centric approach serves as a significant source of inspiration for "SWAP BOOK," as it strives to foster a sense of belonging and collaboration within the platform's user base. ^[17]

2. Amazon Kindle: Amazon Kindle, with its series of e-readers, has been a trailblazer in the world of digital reading. Amazon Kindle devices offer users the ability to browse, purchase, download, and read e-books, newspapers, magazines, and other digital media seamlessly. By studying Amazon Kindle, "SWAP BOOK " gains insights into the digital format that Indian readers might find most comfortable, thus aiding in the development of an engaging user interface that resonates with the unique preferences of the local audience. ^[18]

3. OLX Group: OLX is a Dutch-domiciled online marketplace known for its user-to-user trading platform. With a focus on buying and selling, OLX provides a valuable reference for "SWAP BOOK," particularly in the "SWAP" feature. While OLX has similarities to "SWAP BOOK" in terms of the exchange aspect, "BOOK IT" distinguishes itself by prioritizing enhanced security measures to safeguard the interests of its users, a critical improvement over the OLX platform. ^[19]



In summary, "SWAP BOOK" emerges as a transformative platform, where the cultural nuances of India, a nation with deep-rooted reverence for traditional literature, meet the digital wave sweeping the world. By drawing inspiration from successful platforms like Wattpad, Amazon Kindle, and OLX, "SWAP BOOK" aims to offer a unique and engaging reading experience, accompanied by a robust sense of community and enhanced security measures, bridging the gap between traditional and digital reading preferences in India. It is poised to be the catalyst that ushers in a new era of literary exploration and connectivity while respecting and preserving the rich heritage of printed literature.

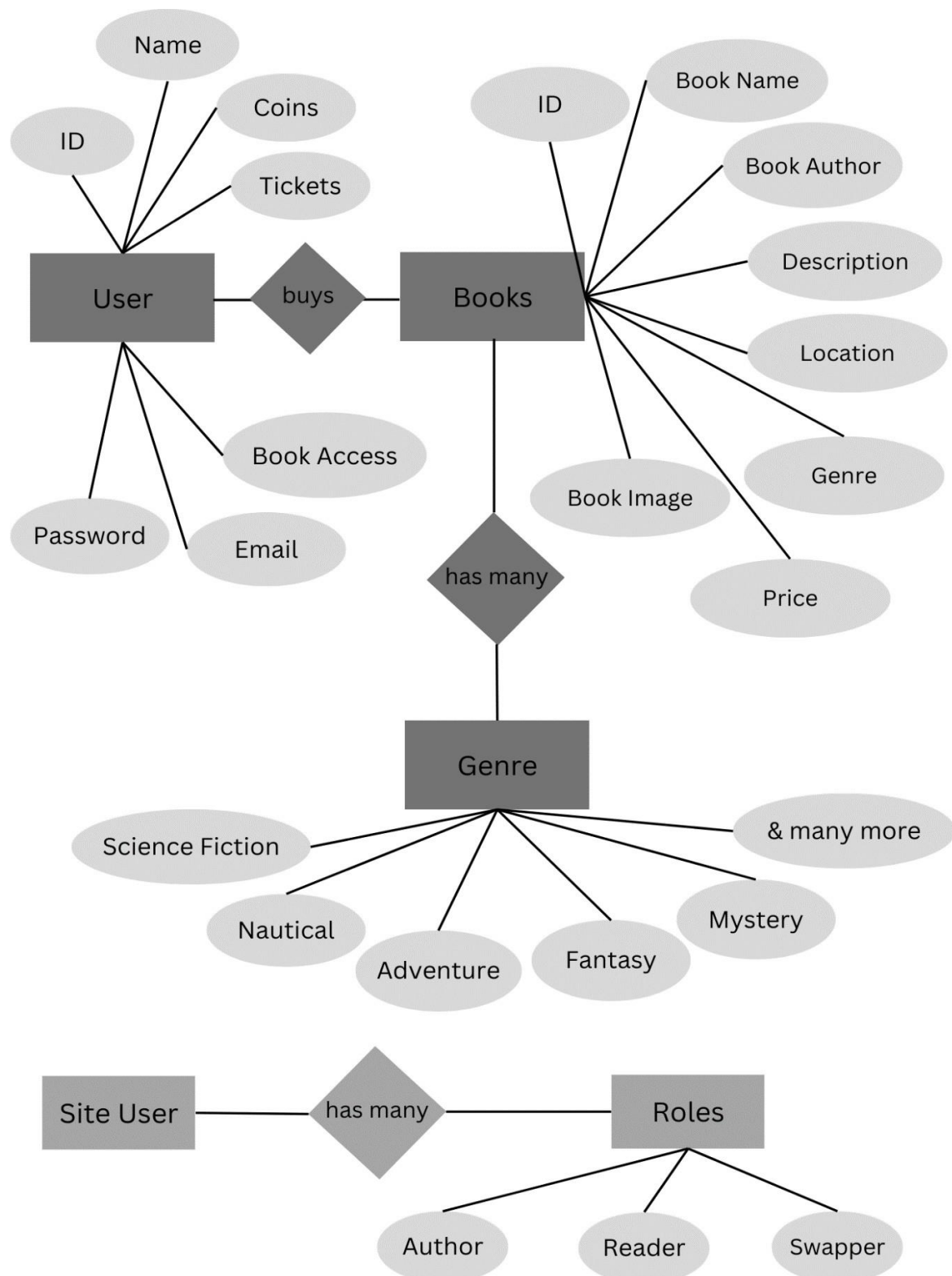


SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY Bhopal

Department of Computer Science & Engineering

CHAPTER 2 :DESIGN

2.1 ER Diagram:





Title: Entity-Relationship Diagram for "Swap Book" Project

Purpose:

The purpose of this Entity-Relationship (ER) diagram is to visually represent the data structure and relationships within the "Swap Book" project. This ER diagram helps stakeholders, developers, and users understand how different entities interact and what information is stored in the database.

Precondition:

- The ER diagram is a representation of the database structure used in the "Swap Book" project.
- Entities, attributes, and relationships have been defined as per the project requirements.

PrimaryFlow:

1. Users Entity:

- Attributes:
 - ID (Primary Key)
 - Name
 - Ticket
- Password
- Email
- Relationships:
 - Users have a "Buys" relationship with the "Books" entity, indicating that users can purchase books.



- Users have a "Roles" relationship, allowing them to have roles such as "Author," "Reader," and "Swapper."

2. Books Entity:

- Attributes:

- ID (Primary Key)
- Book Name
- Author
- Description
- Genre
- Price

- Relationships:

- Books have a "Has Many" relationship with the "Genre" entity, signifying that each book can belong to one or more genres.

3. Genre Entity:

- Attributes:

- Adventure
- Science Fiction
- Nautical

- Description:

- Genre attributes represent the available book genres.

4. Roles Entity:



- Attributes :

- Author
- Reader
- Swaper

- Description:

- Roles attributes define the different roles users can have in the system.

Alternate Flow:

- Users can have additional attributes and relationships based on the specific requirements of the "Swap Book" project. For example, user profiles, reviews, users.

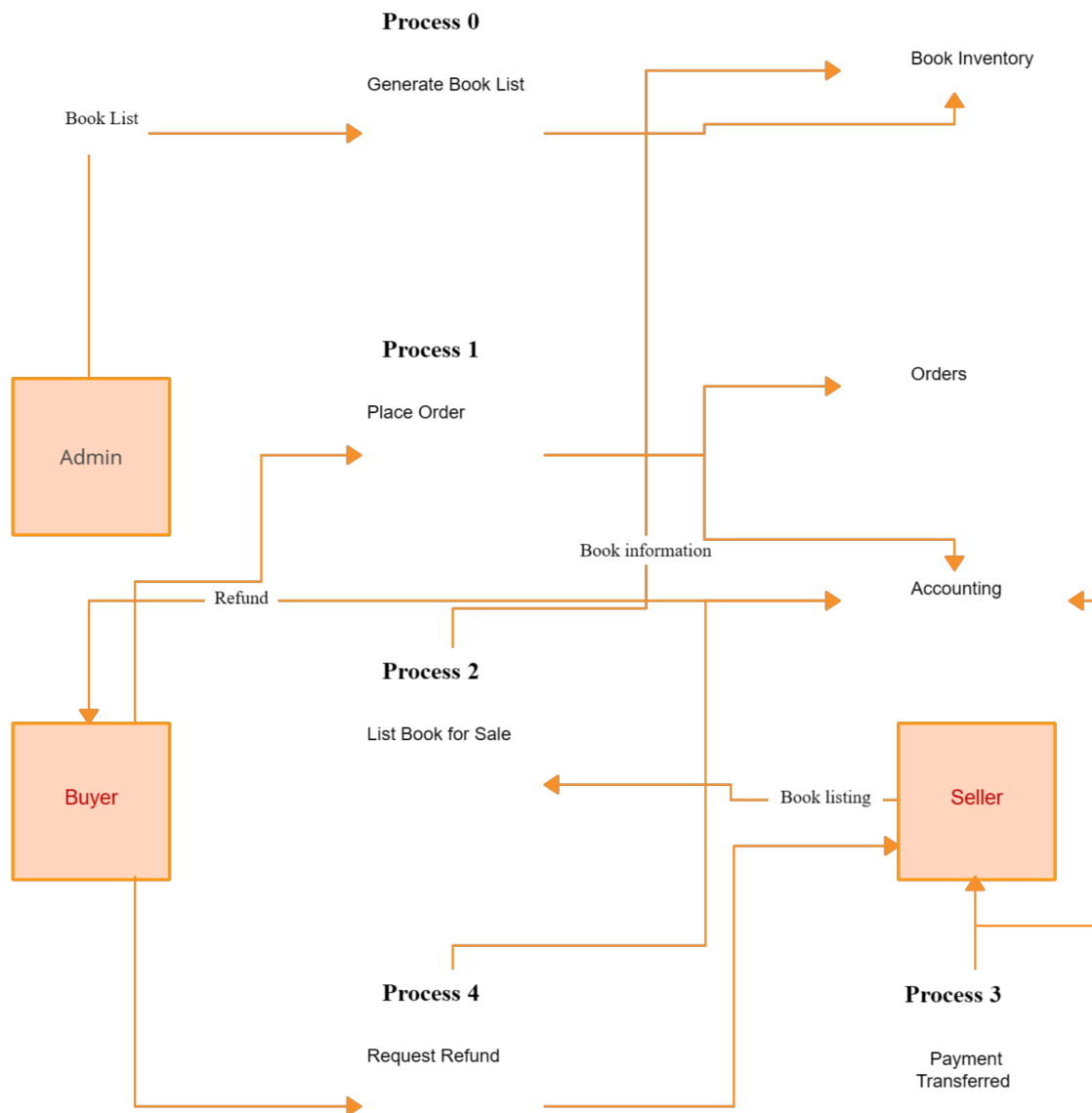
Error Flow:

- Errors can occur when data integrity is compromised, such as duplicate user IDs or inconsistent data in the database. Proper data validation and error handling mechanisms must be in place to address these issues.

Post Condition:

- The ER diagram provides a comprehensive overview of the database structure for the "Swap Book" project, which includes entities, attributes, and relationships.
- It serves as a reference for the development team to create the database schema, also helps to understand stakeholders understand the system.

2.2 Data Flow Diagram:





Title:

Data Flow Diagram for "Swap Book" Project

Purpose:

The purpose of this Data Flow Diagram (DFD) is to illustrate the flow of data and processes within the "Swap Book" project. It provides a visual representation of how different entities interact with the system to perform various functionalities.

Precondition:

- The DFD is a representation of the data and process flow in the "Swap Book" project.
- Entities (Admin, Buyer, Seller) and their respective functionalities have been defined as per the project requirements.

Primary Flow:

1. Admin:

- Functionalities:
 - Book List: The admin can view a list of books available on the platform.
 - Generate Book List: The admin can generate a comprehensive book list.

2. Buyer:

- Functionalities:
 - Refund: Buyers can request refunds for purchases.
 - Place Order: Buyers can place orders for books they wish to purchase.
 - Order: This represents the process of order processing and fulfilment.
 - Accounting: Buyers can view and manage their financial transactions.



3. Seller:

- Functionalities:

- Accounting: Seller can view, access and manage their financial transactions.
- List Book for Sale: Sellers can list books they want to sell on the platform.
- Payment Transfer: The process of transferring payments to sellers after successful book sales.

4. List Book for Sale:

- Functionalities:

- Book Info: Users can view detailed information about the books listed.
- Book Inventory: The process of managing the availability and stock of books listed for sale.



Alternate Flow:

- Within the primary flow, users may interact with additional functionalities based on the specific requirements of the "Swap Book" project. These could include user profiles, reviews, and other book-related interactions.

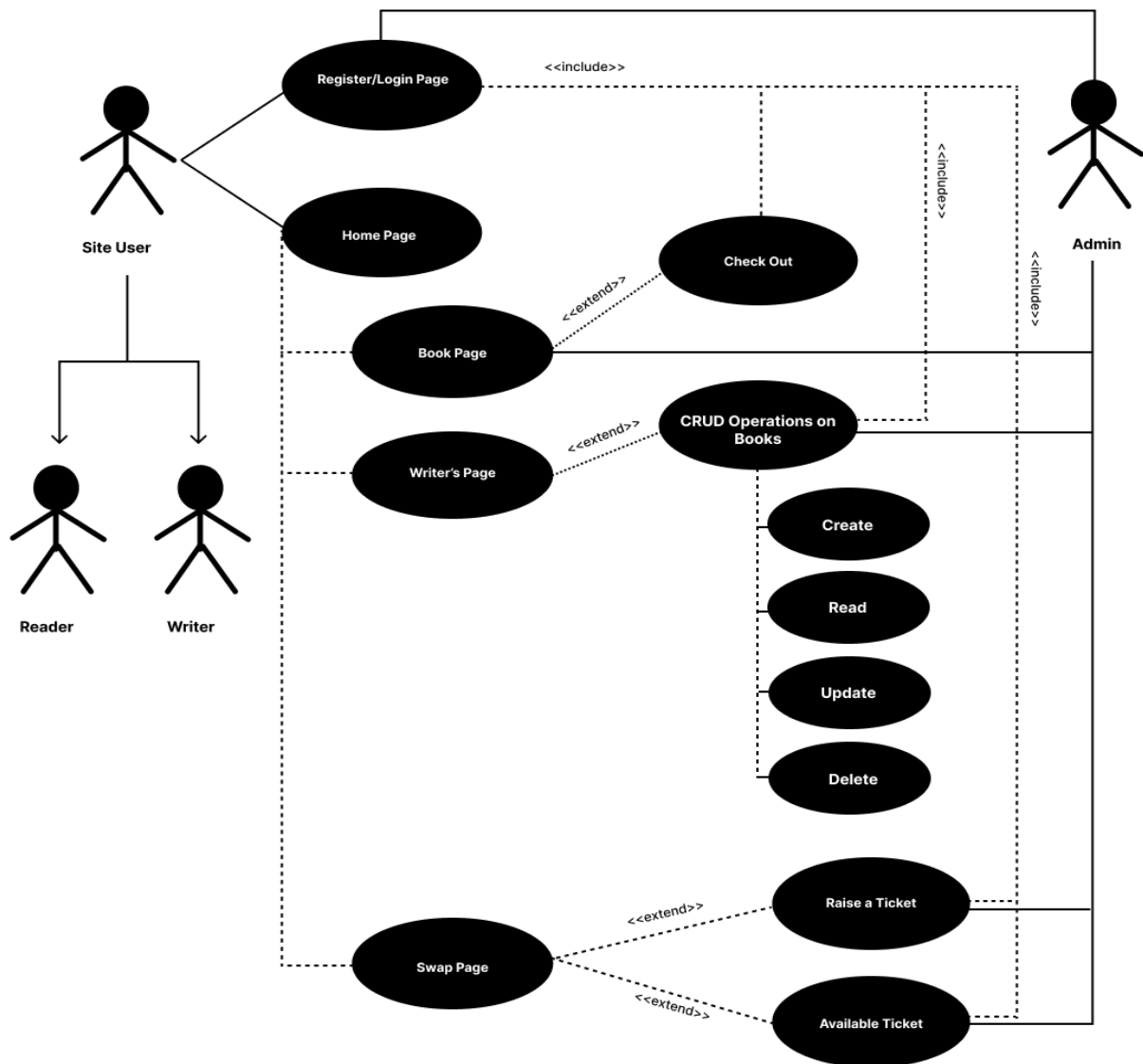
Error Flow:

- Errors can occur in the system, such as data entry errors, payment processing issues, or inventory discrepancies. Proper error handling mechanisms and data validation should be in place to address these issues.

Post Condition:

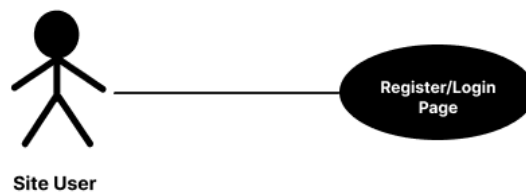
- The DFD provides a comprehensive visual representation of data flow and processes in the "Swap Book" project, including interactions between Admin, Buyers, Sellers, and the "List Book for Sale" functionality.
- It serves as a reference for the development team to design and implement within the system.
- The DFD helps stakeholders understand how data flows between entities and functionalities, ensuring a smooth and efficient operation of the platform.

2.3 Use Case Diagram:





2.3.1 Register/Login Page Use Case Diagram





Title: Register/Login Page Use Case Diagram

Purpose:

The purpose of this use case diagram is to illustrate the interactions and functionalities involving "Site User" and the "Login/Register" processes in the "Swap Book" project. It helps stakeholders and developers understand the core user actions and how they connect with the authentication system.

Precondition:

- The use case diagram is a representation of the user interactions within the "Swap Book" project.
- The "Site User" node represents individuals who interact with the platform.
- The "Login/Register" node symbolizes the key process for user authentication and account creation.

Primary Flow:

1. "Site User" Interactions:

- Login to Existing Account*: A "Site User" can log in to their existing account by providing their username/email and password. After successful authentication, they gain access to their account, including their listed books, saved searches, and settings.
- Register a New Account: A "Site User" can register a new account by providing the necessary details, including username, email, and password. After successful registration, they become a registered user of the platform, allowing them to list books, search for books, and engage with other users.



2. "Login/Register" Process:

- Login: The "Login/Register" process handles the authentication of "Site Users." It verifies the user's credentials against the database. If the login is successful, the user is directed to their account dashboard. If the login fails, an error message is displayed, indicating incorrect credentials.
- Registration: The "Login/Register" process facilitates the registration of new users. It collects user information, checks for data validity, and stores the user's data in the database. After successful registration, the user is redirected to the login page to access their newly created account.

Alternate Flow:

- An alternate flow might occur when user interactions lead to password reset or account recovery processes. These cases are not represented in the core use case diagram but are part of the extended functionalities related to user management.

Error Flow:

- Errors can occur during the login or registration process due to issues like incorrect credentials, validation errors (e.g., invalid email format), or database connectivity problems. Proper error handling and messaging are essential to guide users when such issues arise.

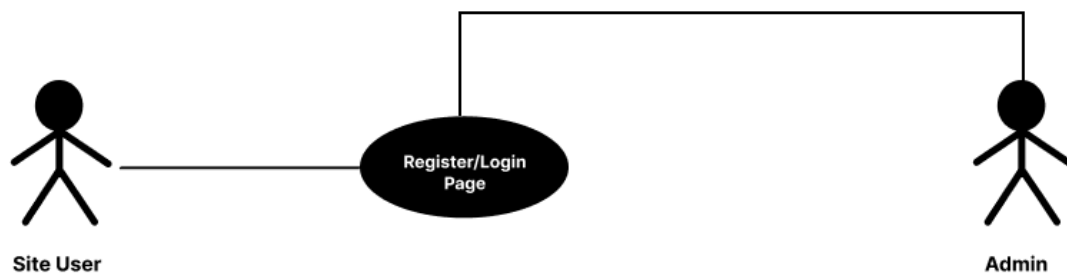


Post Condition:

- The use case diagram offers a clear depiction of the core user interactions with the "Swap Book" platform, particularly focusing on login and registration processes.
- It serves as a reference for developers to implement the authentication and user management components of the system.
- Stakeholders gain an understanding of how "Site Users" engage with the platform and how the "Login/Register" process facilitates user authentication account creation.



2.3.2 Register/Login Page Admin Use Case Diagram





Title: Register/Login Page Admin Use Case Diagram

Purpose:

The purpose of this Use Case Diagram is to illustrate the interactions between different actors and the system in the "Swap Book" project. It visually represents the core functionalities related to user authentication, specifically the "Login/Register" process, and how it connects "Site Users" with "Admin" and "User" roles.

Precondition:

- The Use Case Diagram serves as a representation of the user interactions and system behaviour in the "Swap Book" project.
- Actors and use cases have been defined as per the project requirements.

Primary Flow:

1. Site User:

- Use Cases:
- Login/Register: Site Users can initiate the "Login/Register" use case to create an account or log in to the platform.
- Description: A Site User can either log in with an existing account or user.



- Connection to Actors:

- "Login/Register" is connected to "Admin" and "User" roles, indicating that Site Users can access functionalities available to both Admin and User roles.

2. Admin:

- Actor Description:

- Represents the administrative users responsible for managing and overseeing the "Swap Book" platform.

- Use Cases:

- Admins have access to various functionalities such as user management, content moderation, and system configuration.

3. User:

- Actor Description:

- Represents standard users who use the "Swap Book" platform to exchange and purchase books.

- Use Cases:

- Users can access core functionalities like browsing, searching, and interacting with listings, managing their profiles, and making transactions.

Alternate Flow:

- The Use Case Diagram can be expanded to include additional use cases and actors as needed to cover more functionalities and scenarios within the "Swap Book" platform.



Error Flow:

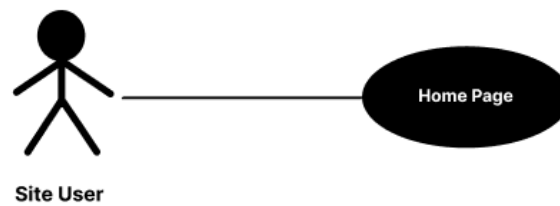
- Errors can occur during the "Login/Register" process, such as invalid credentials, registration failures, or login timeouts. Proper error handling mechanisms must be in place to address these issues.

Post Condition:

- The Use Case Diagram provides a clear overview of the interactions between "Site Users," "Admin," and "User" roles within the "Swap Book" system.
- It serves as a reference for understanding how users can register, log in, and access functionalities available to both Admin and User roles.
- The diagram helps stakeholders and the development team visualize the core user interactions and system behaviour in the context of user authentication.



2.3.3 Home Page Use Case Diagram





Title: Home Page Use Case Diagram

Purpose:

The purpose of this use case diagram is to illustrate the interactions and functionalities within the "Swap Book" project, focusing on the primary use cases associated with "Site User" and the "Home Page."

Precondition:

- The use case diagram represents the high-level functionality of the "Swap Book" web application.
- Use cases have been identified and defined based on the project's requirements.

Primary Flow:

1. Site User (Actor):

- Description:

- The "Site User" represents individuals who interact with the "Swap Book" web application. They can perform various functions within the system.

- Associated Use Cases:

- Register: A user can create an account on the platform.
- Log In: Users can log in to access their accounts.
- Browse Books: Users can search and view available books listed on the platform.
- List Books: Users can create listings to offer books for exchange or sale.
- Purchase Books: Users can buy books listed by others.
- Log Out: Users can log out of their accounts.



- Manage Profile: Users can update their profile information.
- Post Condition:
 - The "Site User" actor interacts with the system by performing these key use cases, enabling them to use the platform for various purposes.

2. Home Page:

- Description:
 - The "Home Page" represents the main landing page of the "Swap Book" web application, where users initiate their interaction with the system.
- Associated Use Cases:
 - View Home Page: Users can access the home page when they visit the platform.
 - Search Books: Users can initiate book searches directly from the home page.

Post Condition:

- Users access the "Home Page" to start their interaction with the "Swap Book" application. From the home page, they can proceed to other use cases.

Alternate Flow:

- The "Home Page" may also include links to other sections or pages of the application, allowing users to explore different features and functionalities.

Error Flow:

- Errors can occur when users encounter issues related to registration, login, or any other use case. Appropriate error messages and handling mechanisms must be in place to guide users and address issues effectively.

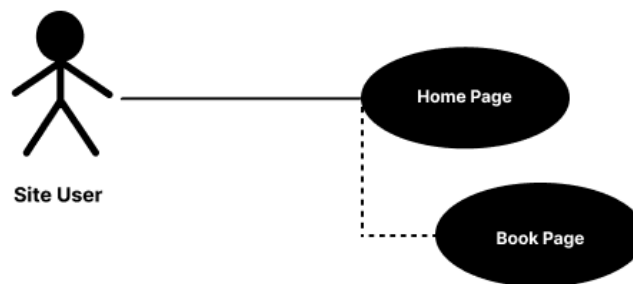


Post Condition:

- The use case diagram provides a visual representation of how "Site Users" interact with the "Swap Book" application, starting from the "Home Page."
- It serves as a reference for both developers and stakeholders to understand the core functionalities of the platform and how users navigate through the system.



2.3.4 Book Page Use Case Diagram





Title: Book Page Use Case Diagram

Purpose:

The purpose of this use case diagram is to illustrate the interactions and functionalities available to "Site Users" within the "Swap Book" project. It provides an overview of how users can navigate from the "Home Page" to the actions.

Precondition:

- "Swap Book" is a live web application with a registered user base.
- "Site Users" have valid login credentials and are logged in.

Primary Flow:

Use Case 1: Browse Books (Home Page)

- Actor: Site User
- Description: The "Site User" accesses the "Home Page" and is presented with a list of available books for browsing. They can view book details, filter books by genre, and initiate actions like buying or swapping books.

Use Case 2: View Book Details (Home Page to Book Page)

- Actor: Site User
- Description: After browsing books on the "Home Page," the "Site User" selects a specific book of interest and clicks on it. This action navigates the user to the "Book Page," where they can view detailed information about the selected book, including its title, author, description, genre, and price.



Alternate Flow:

- The "Site User" can choose to browse books on the "Home Page" without necessarily navigating to the "Book Page" to view book details. This is an alternate flow where users explore the available books without delving into specific book details.

Error Flow:

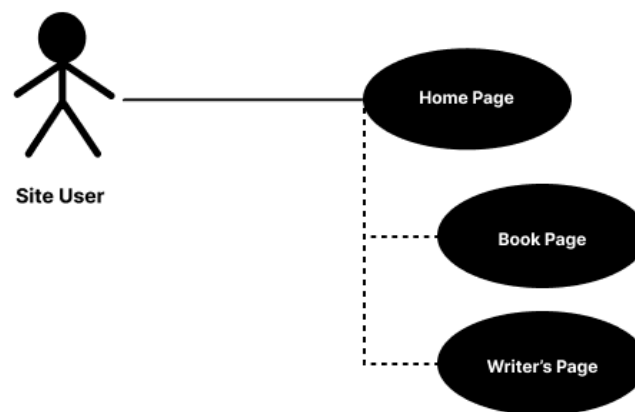
- Errors can occur if a "Site User" encounters issues with the navigation or interaction on the platform, such as:
 - Connectivity issues that prevent access to the "Home Page."
 - Incorrect login credentials, preventing the user from accessing the platform.

Post Condition:

- The use case diagram illustrates the user journey within the "Swap Book" project, including the ability to browse and view book details.
- "Site Users" can engage with the platform, explore books, and make informed decisions regarding buying or swapping books.
- The diagram serves as a reference for developers and stakeholders, ensuring that user interactions are well-defined and align with project requirements.



2.3.5 Writer's Page Use Case Diagram





Title: Writer's Page Use Case Diagram

Purpose:

The purpose of this use case diagram is to illustrate the interactions between different user roles and the key functionalities of the "Swap Book" web application. It helps stakeholders and developers understand how users (Site User) navigate through the system and access specific pages, including the Home Page, Book Page, and Writers Page.

Precondition:

- The use case diagram represents the interaction between the "Site User" and the application's pages.
- The "Swap Book" project is operational and provides the specified user interactions.

Primary Flow:

1. Site User:

- ***Description*:** Site Users are the primary actors who interact with the application. They can perform the following actions:
 - Register: Site Users can create new accounts with a unique username, email, and password.
 - Login: Registered Site Users can log into the application using their credentials.
 - Browse Books: Site Users can view available books on the Home Page.



- View Book Details: Site Users can access detailed information about a specific book on the Book Page.

- Explore Writers: Site Users can explore details about writers and authors on the Writers Page.

2. Home Page:

- Description: The Home Page serves as the entry point for Site Users. It displays a collection of available books that users can browse. Site Users can:

- Browse Books: Users can view a list of books available for exchange or sale.

- Navigate to Book Page: By clicking on a book, users can navigate to the Book Page for more information about that specific book.

3. Book Page:

- Description: The Book Page provides detailed information about a specific book. Site Users can:

- View Book Details: Users can access detailed information about the selected book, including the book's name, author, description, genre, and price.

4. Writers Page:

- Description: The Writers Page allows Site Users to explore information about writers and authors. Users can:

- Explore Writers: Users can view profiles and details about various writers and authors, such as their biographies, works, and contributions.

Alternate Flow:



- While the primary flow covers the key interactions, Site Users may have additional interactions, such as updating their profiles, leaving reviews, or interacting with other users. These interactions can be represented as extensions in the use case diagram.

Error Flow:

- Errors can occur during the interactions, such as incorrect login credentials, unavailable books, or unexpected system issues. Proper error handling mechanisms should be in place to address these issues and ensure a smooth user experience.

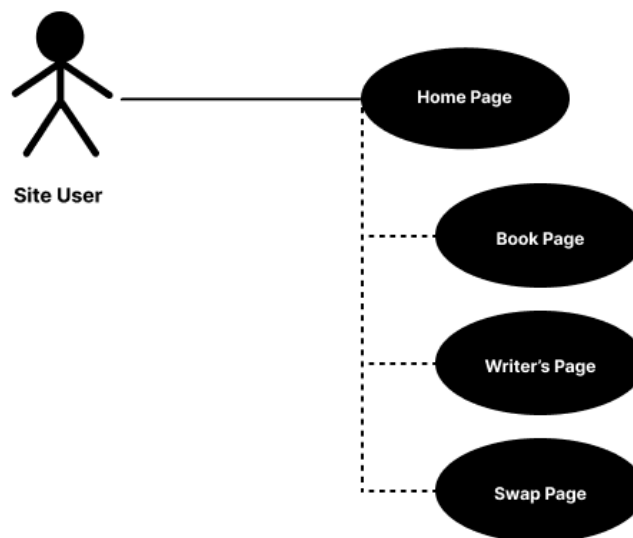
Post Condition:

- The use case diagram effectively represents the interactions between Site Users and the application's pages.
- It provides a clear overview of the primary functionalities of the "Swap Book" project, including user registration, login, book browsing, book details viewing, and exploring writers and authors.

This use case diagram is a valuable tool for understanding and visualizing the user interactions within the "Swap Book" project, helping to ensure a user-friendly and efficient user experience.



2.3.6 Swap Page Use Case Diagram





Title: Swap Page Use Case Diagram

Purpose:

The purpose of this use case diagram is to illustrate the various interactions and functionalities available to "Site Users" in the "Swap Book" project. It provides an overview of the primary use cases and the relationships between users and different pages within the application.

Precondition:

- The use case diagram represents the user interactions in the "Swap Book" web application.
- "Site Users" are registered and logged into the system.

Primary Flow:

1. Site User:

- *Description*: A "Site User" represents an individual registered on the "Swap Book" platform. They have access to various functionalities.

2. Home Page:

- Description: The "Home Page" is the entry point for "Site Users" upon logging in. It provides an overview of the available features and options.



- Use Cases:

- View Book Listings: "Site Users" can view listings of books available for exchange or sale.
- Explore Writers: "Site Users" can explore profiles of writers and authors.
- Access Swap Page: "Site Users" can access the "Swap Page" to list their own books for exchange.

3. Book Page:

- Description: The "Book Page" allows "Site Users" to browse and access detailed information about individual books listed on the platform.
- Use Cases:
- View Book Details*: "Site Users" can view detailed information about a specific book, including its title, author, description, genre, and price (if applicable).
- Add to Favourites: "Site Users" can add books to their favourites for easy access.

4. Writers Page:

- Description: The "Writers Page" enables "Site Users" to explore and learn more about writers and authors.
- Use Cases:
- View Writer Profiles: "Site Users" can access profiles of writers, including their biographies, works, and related information.
- Follow Writers: "Site Users" can choose to follow writers to receive updates on their latest works.



5. Swap Page:

- Description: The "Swap Page" empowers "Site Users" to list their own books for exchange or sale.
- Use Cases:
 - List Books: "Site Users" can create listings for books they wish to exchange or sell, providing details such as book name, author, genre, and price.
 - Manage Listings: "Site Users" can edit, delete, or update existing listings.

Alternate Flow:

- Additional use cases may be added based on the evolving requirements of the "Swap Book" project, such as user interactions, reviews, recommendations, features.

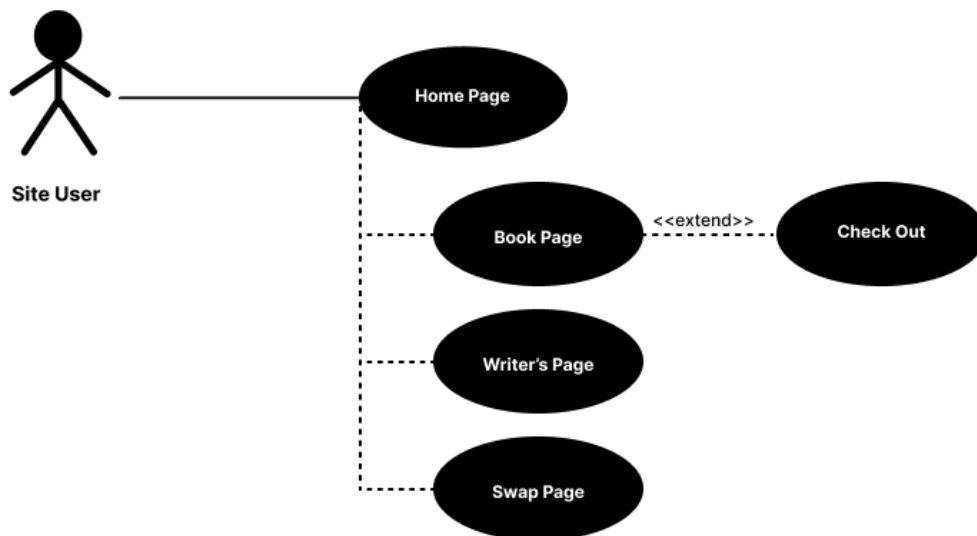
Error Flow:

- Error handling mechanisms should be in place to address issues related to data integrity, user authentication, and system availability.
- Proper error messages and notifications should be provided to guide users.

Post Condition:

- The use case diagram provides an overview of the primary interactions and functionalities available to "Site Users" within the "Swap Book" application.
- It serves as a reference for the development team to design and implement the user interface and functionality.
- The diagram helps stakeholders understand how "Site Users" navigate the platform, interact with books, writers, and listings, and participate in the book exchange process.

2.3.7 Book Page Extended Use Case Diagram





Title: Book Page Extended Use Case Diagram

Purpose:

The use case diagram serves as a visual representation of the primary interactions and functionalities of the "Swap Book" project. It outlines how different user roles, such as "Site User," engage with various features, including the "Home Page," "Book Page," "Writers Page," "Swap Page," and "Checkout."

Precondition:

- The use case diagram reflects the functionalities and interactions defined within the "Swap Book" project.
- User roles and system features have been identified and structured based on project requirements.

Primary Flow:

1. Site User:

- Description: The "Site User" represents individuals who visit the "Swap Book" platform. They can interact with various features.
- Use Cases:
 - View Home Page: Site Users can access the "Home Page" to get an overview of the platform.
 - Browse Books: Users can navigate to the "Book Page" to explore available genres.
 - Interact with Writers: Users can access the "Writers Page" to engage with authors and their works.



- Initiate Swaps: Users can utilize the "Swap Page" to initiate book exchanges with other users.

2. Home Page (Extended):

- Description: The "Home Page" serves as the central hub where Site Users land upon visiting the platform. It is extended with several interactions.

- Use Case:

- View Book Page: Site Users can transition to the "Book Page" to explore book listings.

- Visit Writers Page: Users can navigate to the "Writers Page" to engage with authors.

- Access Swap Page: Users can move to the "Swap Page" to initiate book swaps.

3. Book Page (Extended):

- Description: The "Book Page" allows Site Users to browse and explore books available for exchange or sale. It is further extended with the "Checkout" feature.

- Use Cases:

- Explore Books: Site Users can view book listings, including details about titles, authors, genres, and prices.

- Initiate Checkout: Users can proceed to the "Checkout" process to acquire books of interest.

4. Checkout:



- Description: The "Checkout" feature within the "Book Page" enables users to complete transactions for books they wish to purchase.
- Use Cases:
 - Select Books for Purchase: Users can choose books for purchase.
 - Provide Payment Information: Users can enter payment details for the selected books.

Alternate Flow:

- Users may have additional interactions and use cases based on the specific requirements and features of the "Swap Book" platform. These additional use cases can include user registration, login, listing books, managing user profiles, and leaving reviews.

Error Flow:

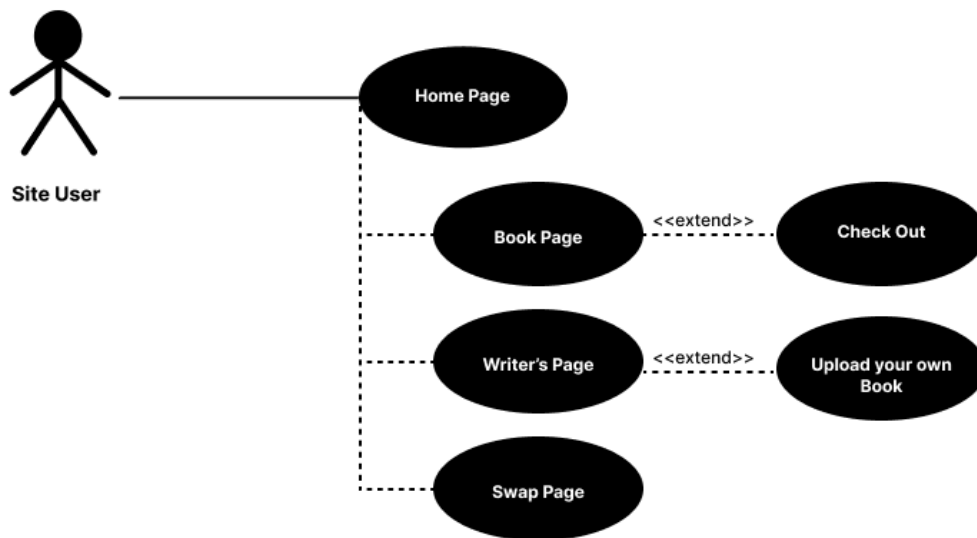
- Error handling and validation mechanisms should be in place to address issues such as incorrect payment information, unavailable books, or unsuccessful transactions.

Post Condition:

- The use case diagram provides a structured representation of how different user roles interact with the key features of the "Swap Book" platform.
- It serves as a reference for developers, stakeholders, investors maintainers ensuring a clear understanding of users interactions, and system functionality



2.3.8 Writer's Page Use Case Diagram





Title: Writer's Page Use Case Diagram

Purpose:

The use case diagram provides an overview of the interactions between different actors and the core functionalities of the "Swap Book" project. It helps in understanding how site users can navigate through the system and perform various tasks, including browsing books, interacting with writers, and initiating book swaps.

Precondition:

- The use case diagram is a visual representation of the user interactions in the "Swap Book" project.
- Actors and use cases have been identified based on project requirements.

Primary Flow:

1. Site User:

- Description:

- The "Site User" represents individuals who visit the "Swap Book" platform. They interact with various features and functionalities.

2. Home Page:

- Description:

- The "Home Page" serves as the entry point for site users. It provides access to key sections, including "Book Page," "Writers Page," and "Swap Page."



Home Page Extended with "Book Page":

- Description:

- The "Book Page" allows users to browse and view details of available books. Users can search for books based on criteria like genre, author, and title.

Home Page Extended with "Writers Page":

- Description:

- The "Writers Page" is a section where users can explore information about writers, their works, and connect with them.

Home Page Extended with "Swap Page":

- Description:

- The "Swap Page" is a feature that enables users to initiate book swap requests. Users can propose book exchanges with other site users.

3. Book Page:

- Description:

- The "Book Page" allows users to access detailed information about specific books. It includes attributes such as book title, author, genre, description, and price (if applicable).

Book Page Extended with "Checkout":



- Description:

- The "Checkout" use case within the "Book Page" allows users to proceed with the purchase of a book. They can add the selected book to their cart and complete the checkout process.

4. Writers Page:

- Description:

- The "Writers Page" provides information about authors, their biographies, and their works.

Writers Page Extended with "Upload Your Own Book":

- Description:

- "Upload Your Own Book" is a feature within the "Writers Page" that enables authors and writers to share their own books with the "Swap Book" community. They can upload book details and make them available for sale.

Alternate Flow:

- Users may have additional use cases based on specific requirements, such as user registration, user reviews, and user messaging.

Error Flow:

- Errors can occur during user interactions, such as failed checkout attempts, incomplete book listings, and incorrect user inputs. Error handling mechanisms and feedback to users must be in place.

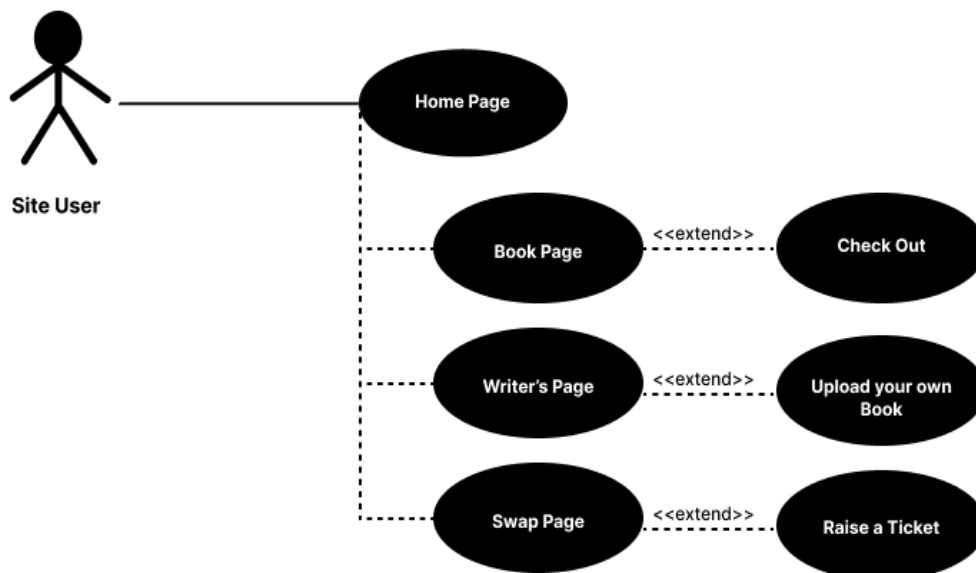


Post Condition:

- The use case diagram provides a clear overview of how site users can interact with the "Swap Book" platform.
- It guides the development team in implementing the necessary functionalities and user interfaces.
- The use case diagram helps stakeholders and users understand the navigation and capabilities of the system.



2.3.9 Swap's Page Use Case Diagram





Title: Swap's Page Use Case Diagram

Purpose:

This diagram outlines interactions between "site users" and the platform's key pages: "home," "book," "writers," and "swap." It illustrates user actions and extensions for book purchasing, writer contributions, and support.

Precondition:

- "Site users" access the "home page" upon platform entry.

Primary Flow:

- "Site users" can:
 1. View Home Page
 2. View Book Page (Extension)
 3. View Writers Page (Extension)
 4. View Swap Page (Extension)
- Book Page (Extension)" includes:
 5. View Book Details
 6. Checkout (Extension)
- "Writers Page (Extension)" offers:



7. Upload Your Own Book (Extension)

- "Swap Page (Extension)" includes:

8. Raise Ticket (Extension)

Alternate Flow:

- Additional use cases may be added based on project requirements.

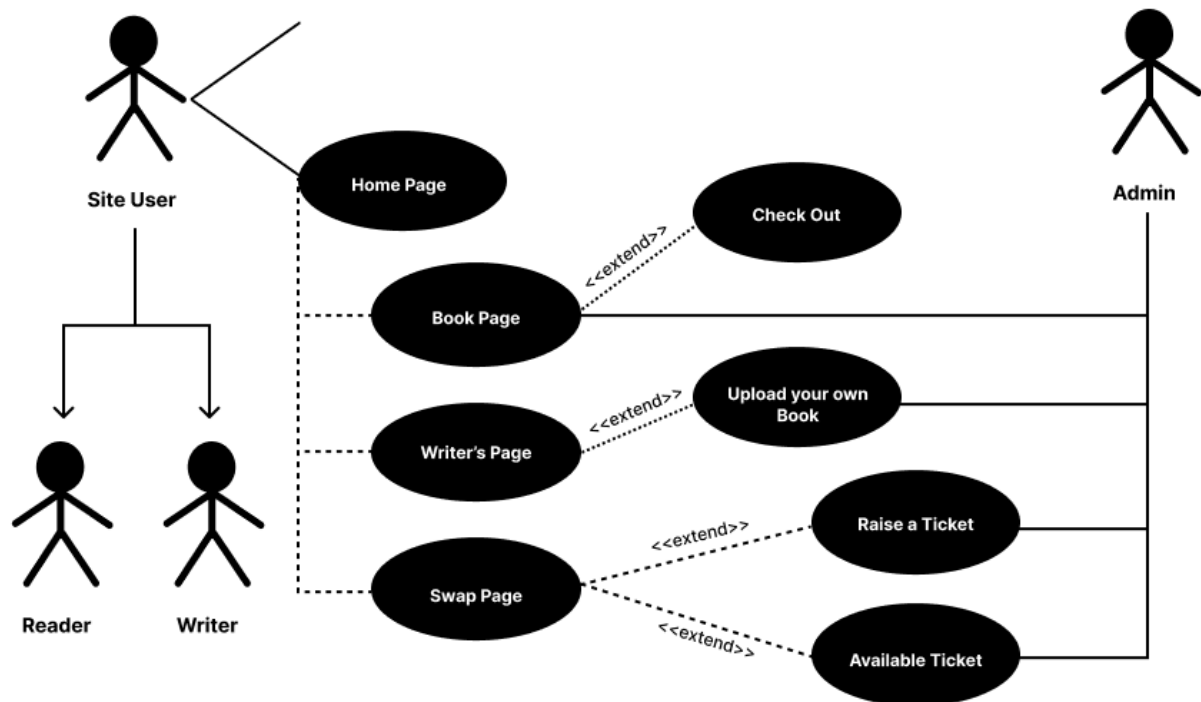
Error Flow:

- Error handling and validation mechanisms should address issues during interactions.

Post Condition:

- The diagram provides an overview of "site users" navigating through the platform, helping developers and stakeholders understand user interactions and system functionalities.

2.3.10 Book Page/Writer's Page / Swap Page Extended Use Case Diagram





Title: Book Page/Writer's Page / Swap Page Extended Use Case Diagram

Purpose:

This diagram illustrates the core interactions in the "Swap Book" platform, highlighting user and admin actions.

Primary Flow:

- Site User: Represents platform users.
- Home Page: Starting point, leading to "Book Page," "Writer Page," and "Swap Page."
- Checkout: Completing book exchanges.
- Upload Your Book: Contributing books.
- Raise Ticket: Seeking support.
- Available Ticket: Managing user inquiries.
- Admin: Admin role for platform management.

Alternate Flow:

- Custom functionalities can be added as needed.

Error Flow:

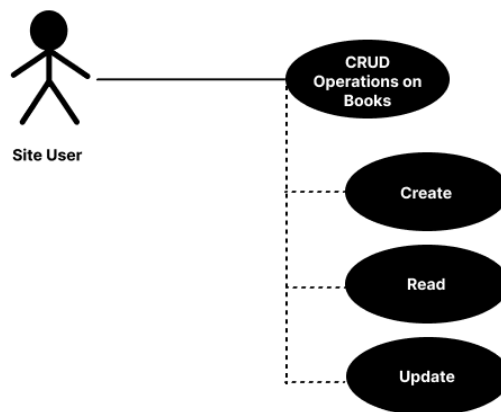
- Implementing error handling within each use case for data validation and user support.

Post Condition:

- Helps stakeholders and developers understand system use cases, design and user interactions.



2.3.11 User's CRUD Operation's Use Case diagram





Title:

User's CRUD Operation's Use Case diagram

Purpose:

This use case diagram illustrates the key interactions and operations of "Site Users" in the "Swap Book" project. It focuses on the core Create, Read, Update, and Delete (CRUD) operations related to book listings.

Precondition:

- The diagram represents the system's functionality.
- "Site Users" are registered and logged in.

Primary Flow:

- "Site Users" can log in, register, view listings, and perform CRUD operations on book listings.
- CRUD operations encompass creating, reading, updating, and deleting book listings.

Alternate Flow:

- Additional user interactions may be included based on project requirements.



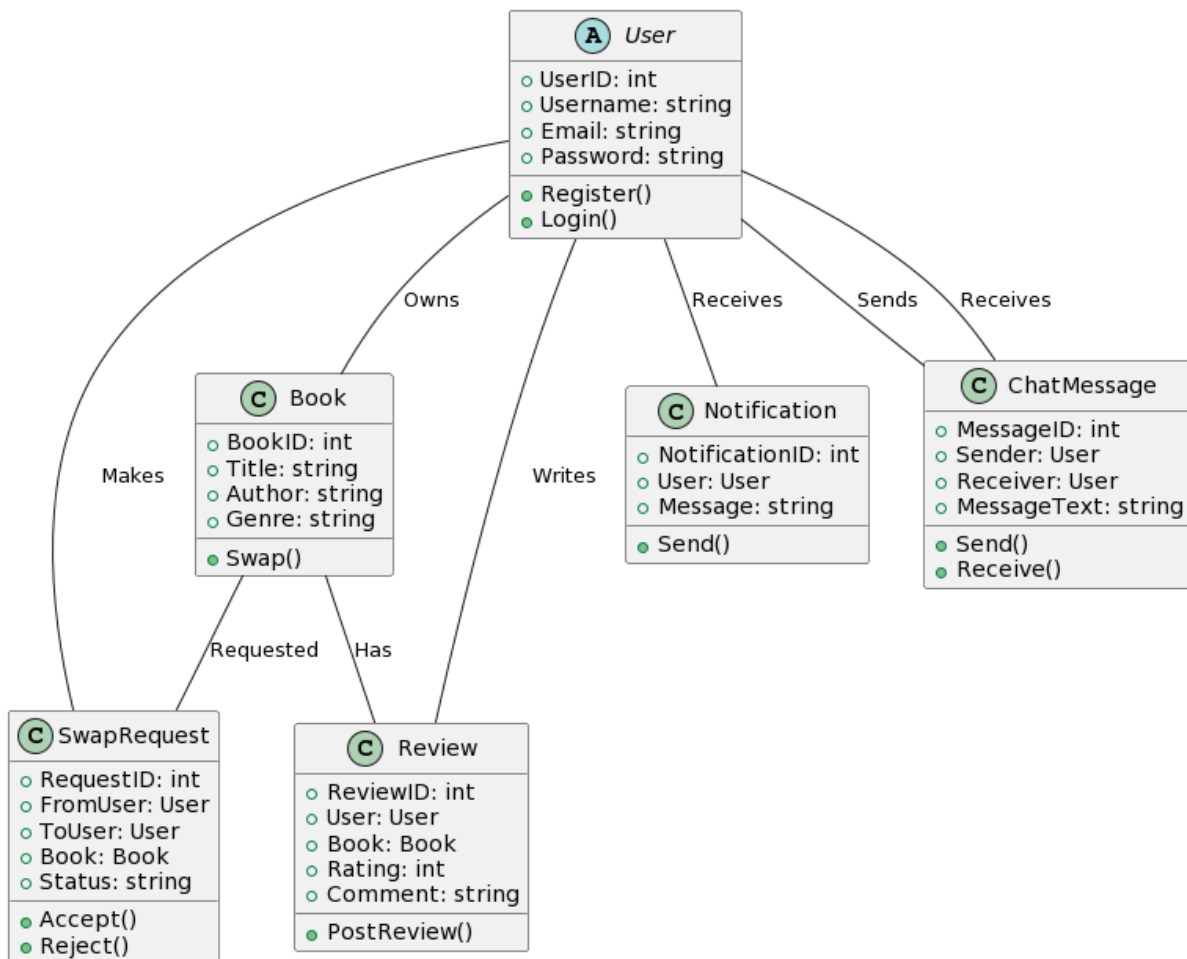
Error Flow:

- The system should handle errors gracefully, providing user-friendly error messages.

Post Condition:

- The diagram serves as a reference for developers and stakeholders, showcasing user interactions and CRUD operations in the system.

2.4 Class Diagram:





Title: Class Diagram for "Swap Book" Project

Purpose:

The purpose of this class diagram is to illustrate the structure and relationships of the classes within the "Swap Book" project. It provides a high-level view of the object-oriented design, helping stakeholders and developers understand the key components, their attributes, and interactions.

Precondition:

- The class diagram is a visual representation of the object-oriented design for the "Swap Book" project.
- Classes, attributes, methods, and relationships have been identified and defined as per project requirements.

Primary Flow:

1. User Class:

- Attributes:

- ID: Identifies a user (Primary Key).
- Name: Stores the user's name.
- Ticket: Represents a unique user identifier.
- Password: Stores the user's password (encrypted).
- Email: Holds the user's email address.



- Methods:

- Register: Allows a user to create an account.
- Login: Authenticates the user.

- Relationships:

- User has a "Buys" relationship with the "Book" class, signifying that users can purchase books.
- User has a "Has Roles" relationship, allowing users to have roles such as "Author," "Reader," and "Swaper."

2. Book Class:

- Attributes:

- ID: Identifies a book (Primary Key).
- Book Name: Stores the book's title.
- Author: Represents the book's author.
- Description: Contains a description of the book.
- Genre: Indicates the book's genre.
- Price: Stores the book's price (if applicable).

- Methods:

- List: Enables users to list their books for exchange or sale.

- Relationships:

- Book has a "Belongs To Genre" relationship with the "Genre" class, specifying the book's genre.



3. Genre Class:

- Attributes:

- Adventure
- Science Fiction
- Nautical

- Description:

- Genre attributes represent the available book genres.

4. Roles Class:

- Attributes:

- Author
- Reader
- Swaper

- Description:

- Roles attributes define the different roles that users can assume in the system.

Alternate Flow:

- Additional classes and relationships can be added to the class diagram based on specific project requirements, such as user profiles, reviews, and interactions between users.



Error Flow:

- Errors may occur when data integrity is compromised, such as duplicate user IDs, missing book information, or inconsistent data. Proper data validation and error handling mechanisms should be implemented in the system to address these issues.

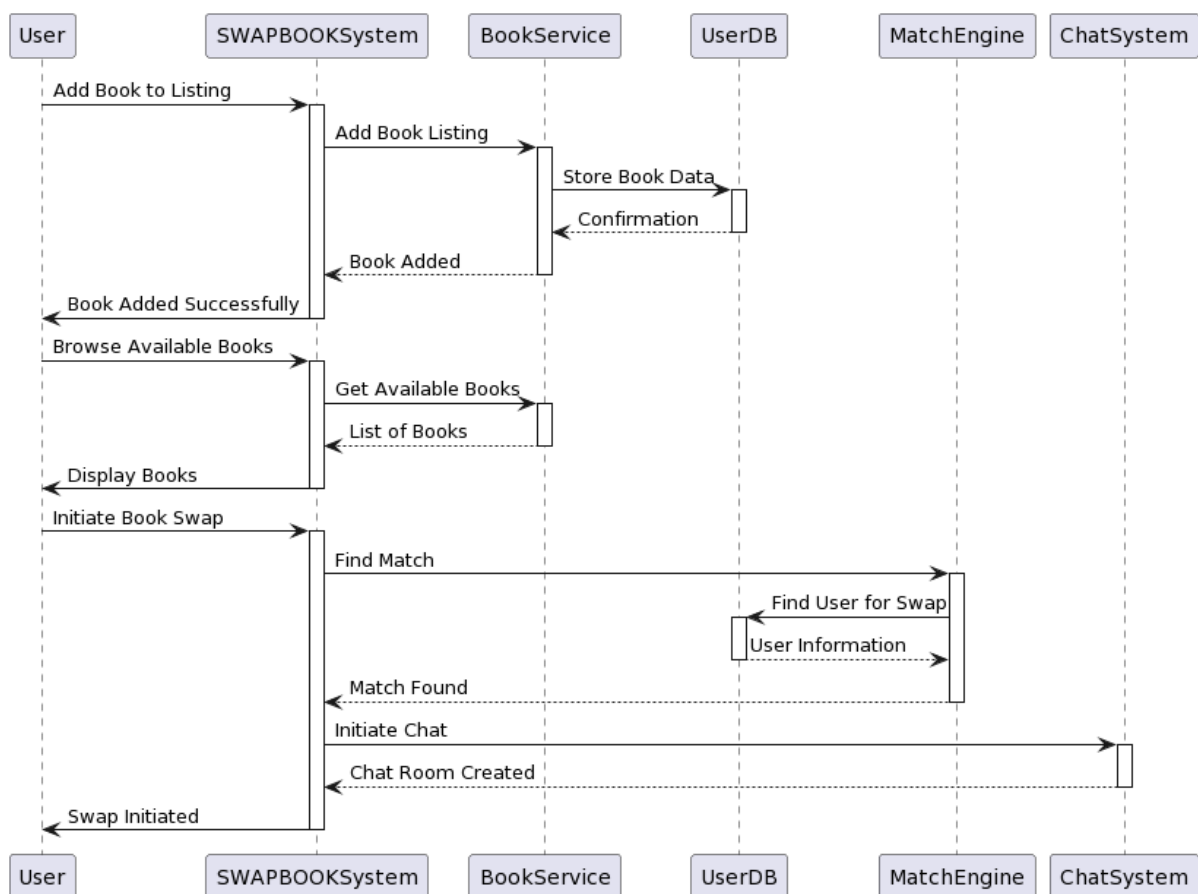
Post Condition:

- The class diagram provides a clear and organized representation of the object-oriented design for the "Swap Book" project.
- It serves as a reference for developers to implement the system's classes, attributes, and relationships.
- The class diagram aids stakeholders in comprehending how user data, book information, genres, and roles are structured and interconnected within the system, facilitating efficient development and maintenance.



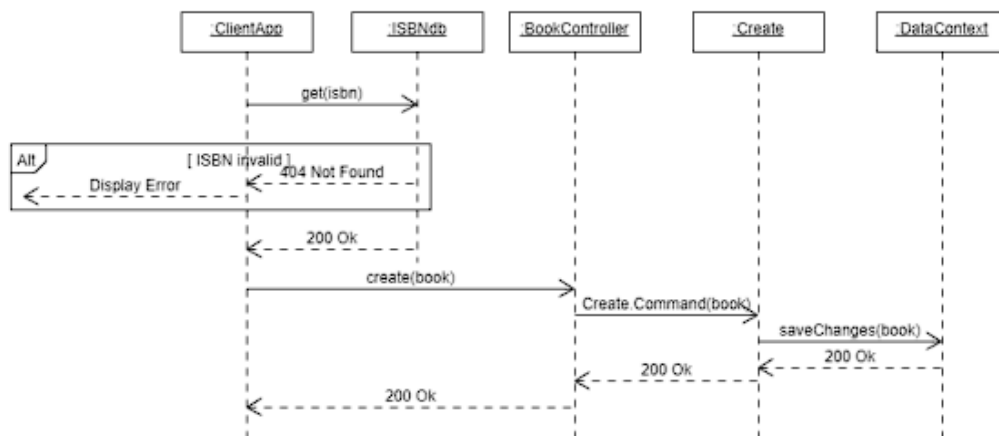
2.5. Sequence Diagram:

2.5.0 Complete Sequence Diagram:

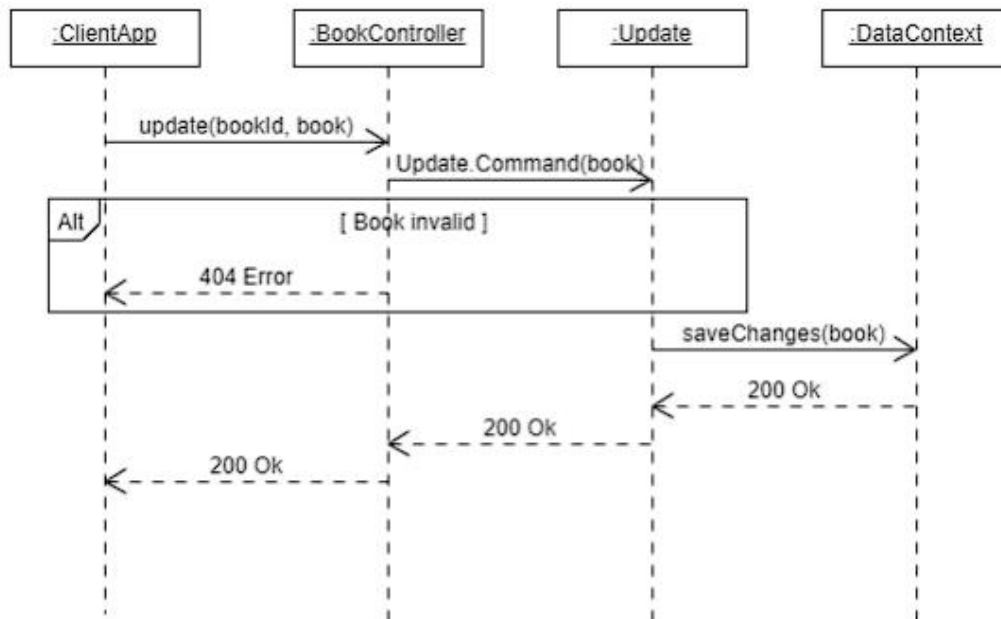




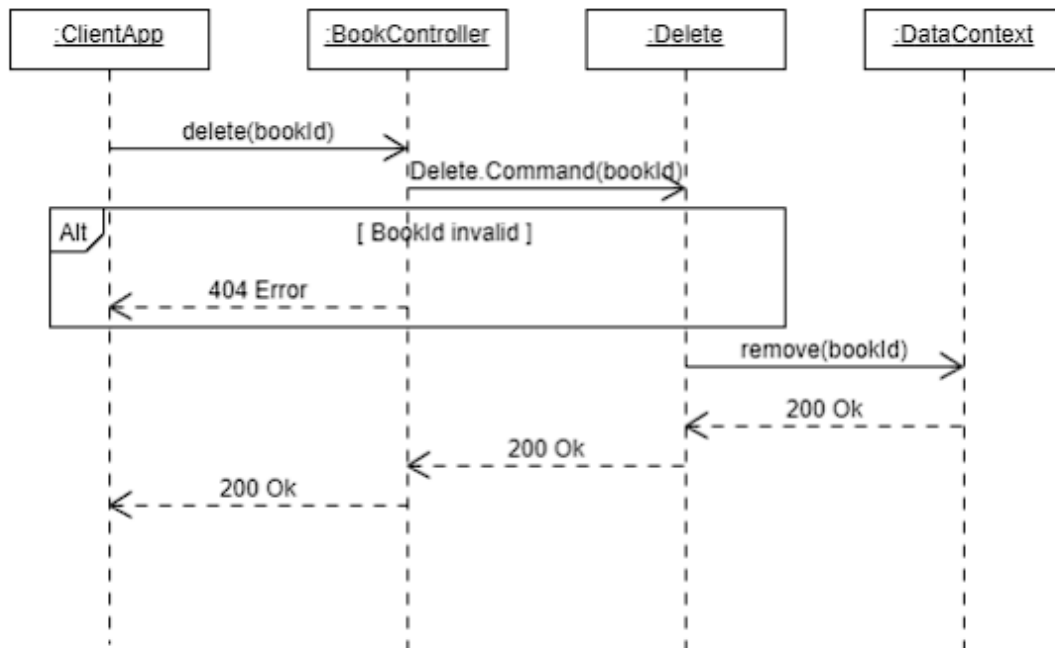
2.5.1 Create Book Sequence Diagram:



2.5.2 Update Book Sequence Diagram:

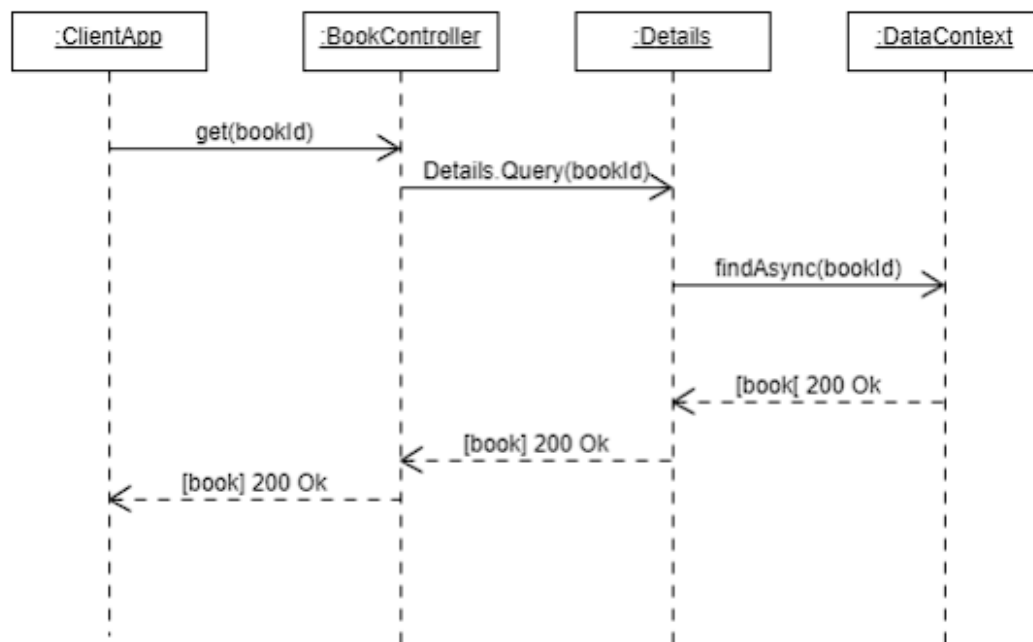


2.5.3 Delete Book Sequence Diagram:



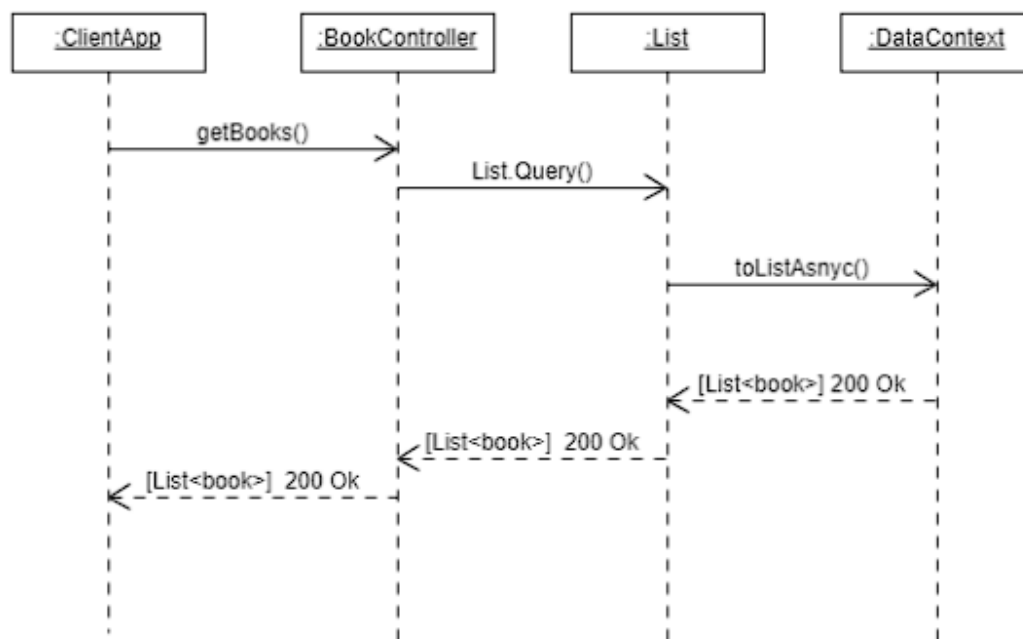


2.5.4 Details Book Sequence Diagram:



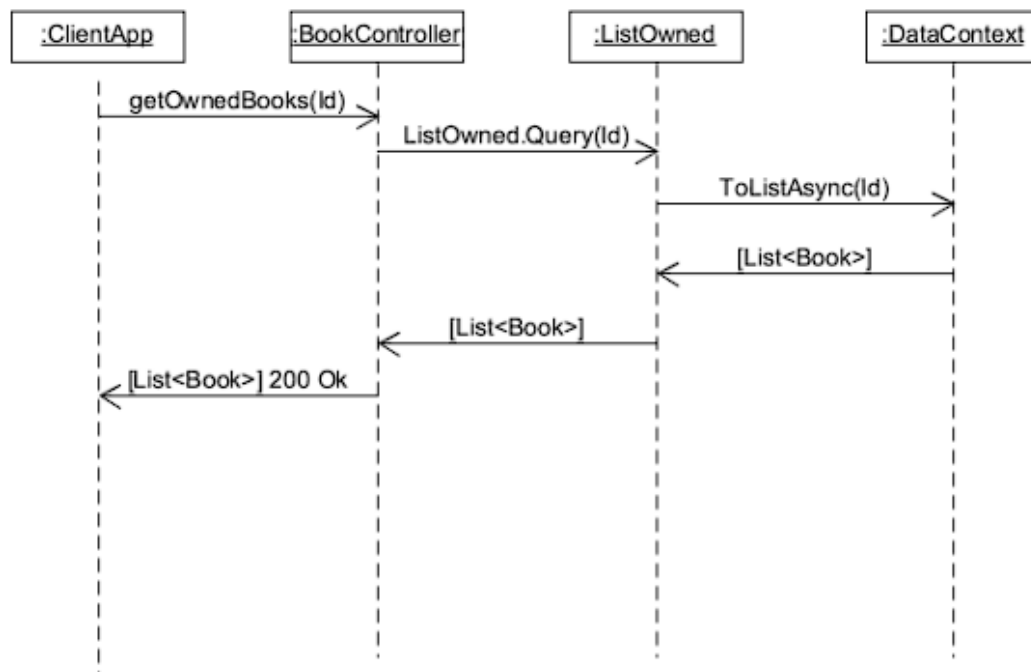


2.5.5 List Book Sequence Diagram:





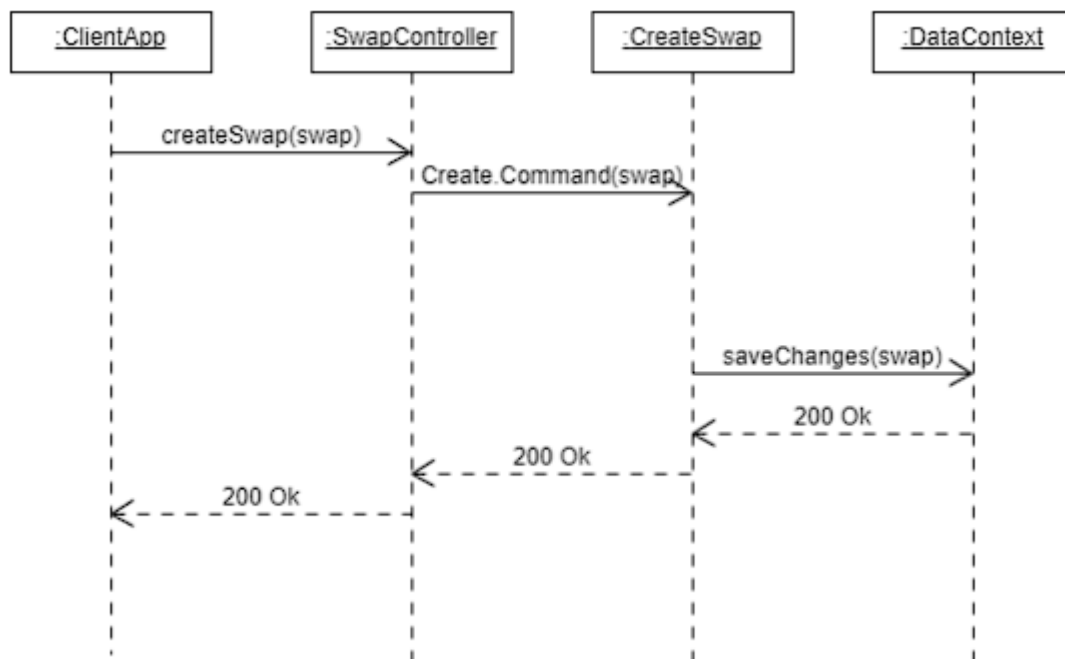
2.5.6 List Books Owned Sequence Diagram:





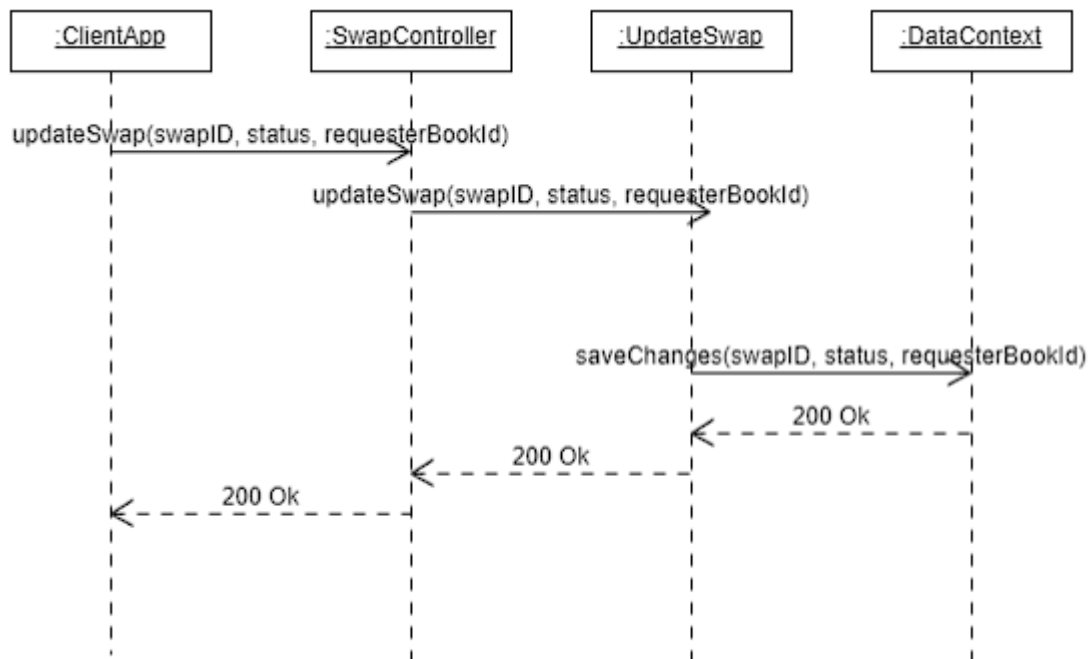
2.5.7 Swap Sequence Diagram:

2.5.7.1 Create Swap Sequence Diagram:

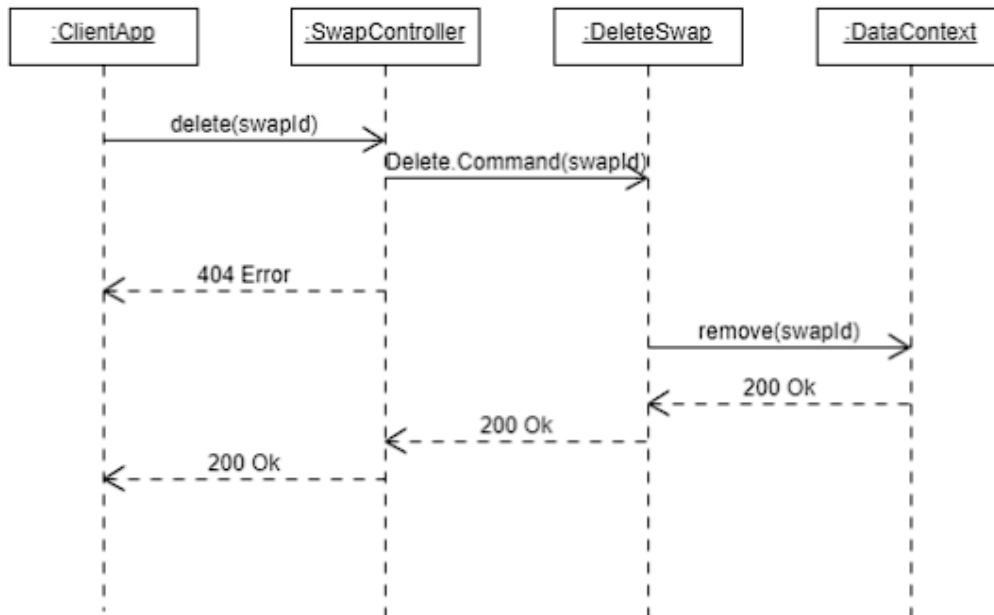




2.5.7.2 Update Swap Sequence Diagram:

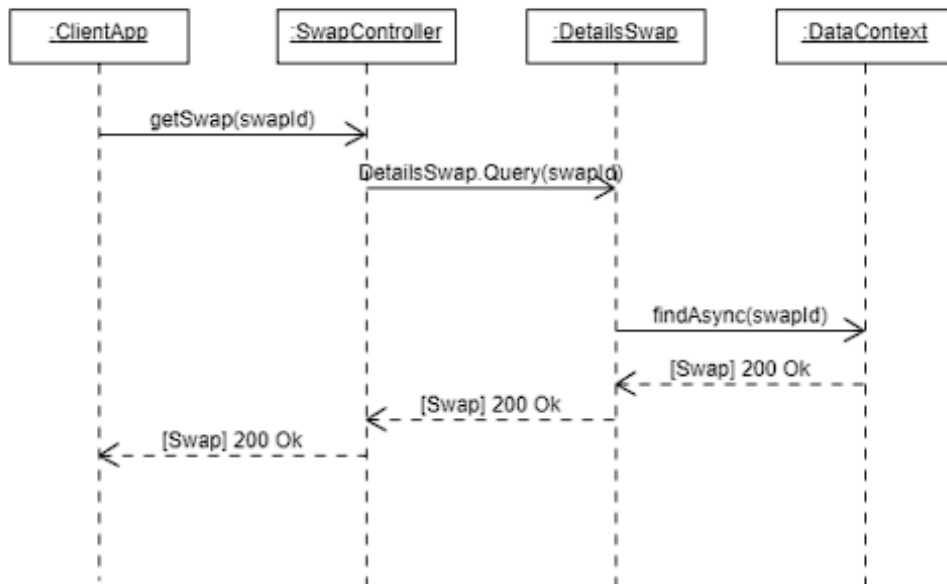


2.5.7.3 Delete Swap Sequence Diagram:



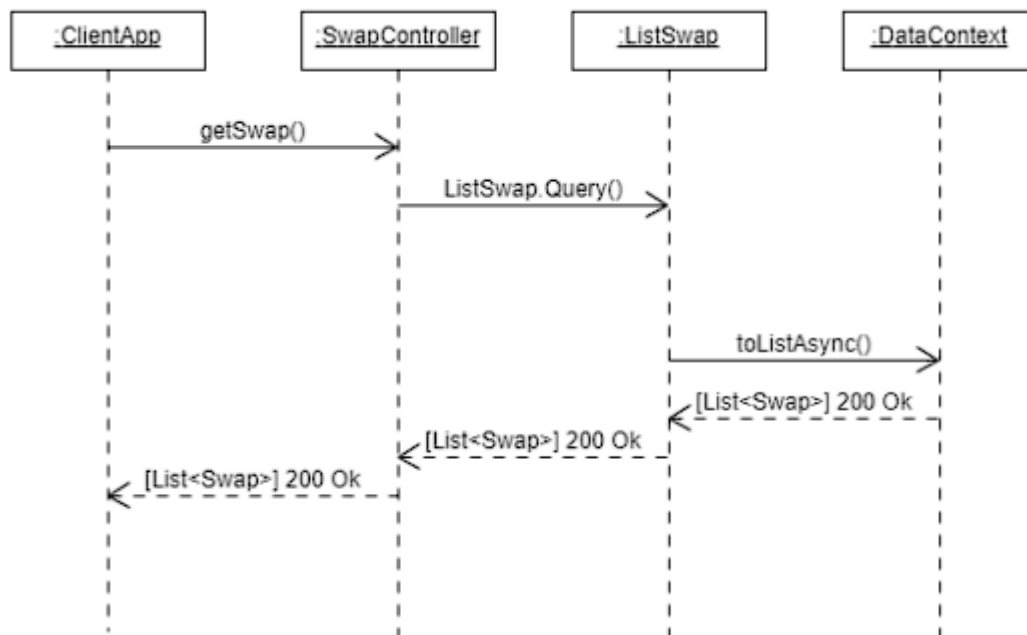


2.5.7.4 Details Swap Sequence Diagram:



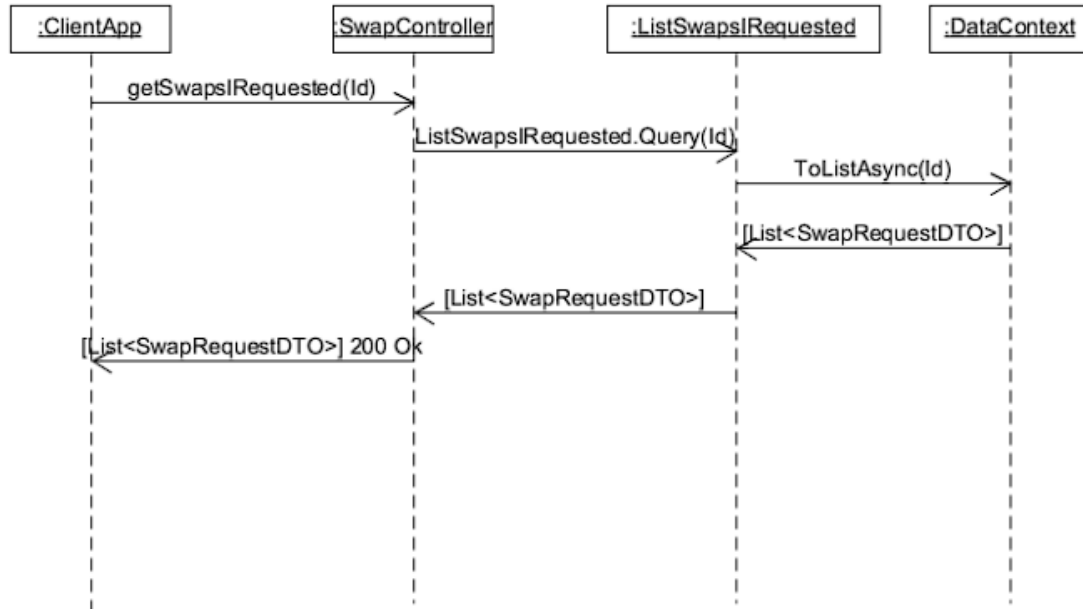


2.5.7.5 List SwapSequence Diagram:



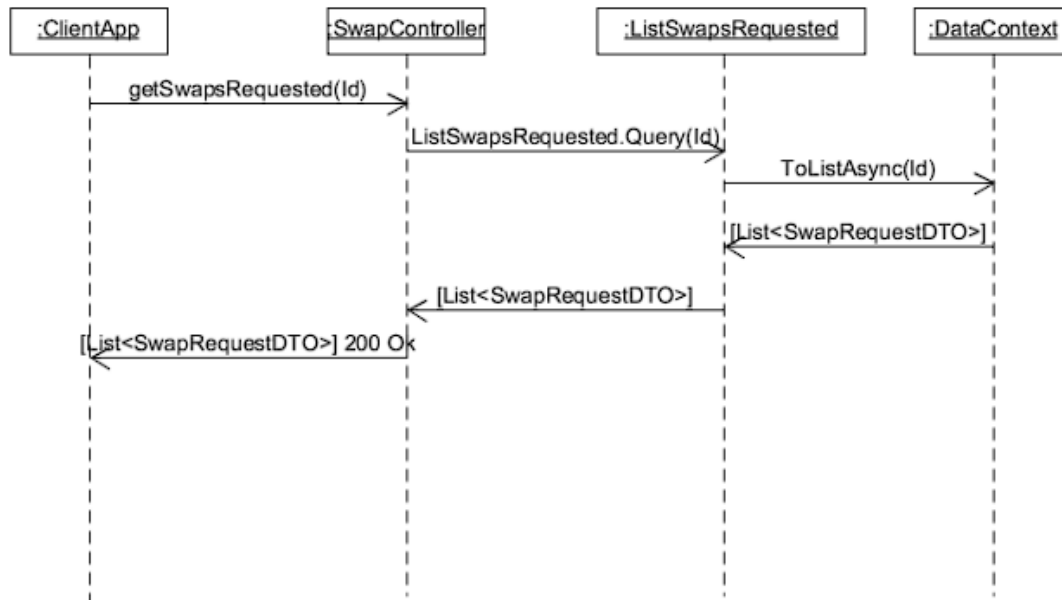


2.5.7.6 List Swap Requested Sequence Diagram:





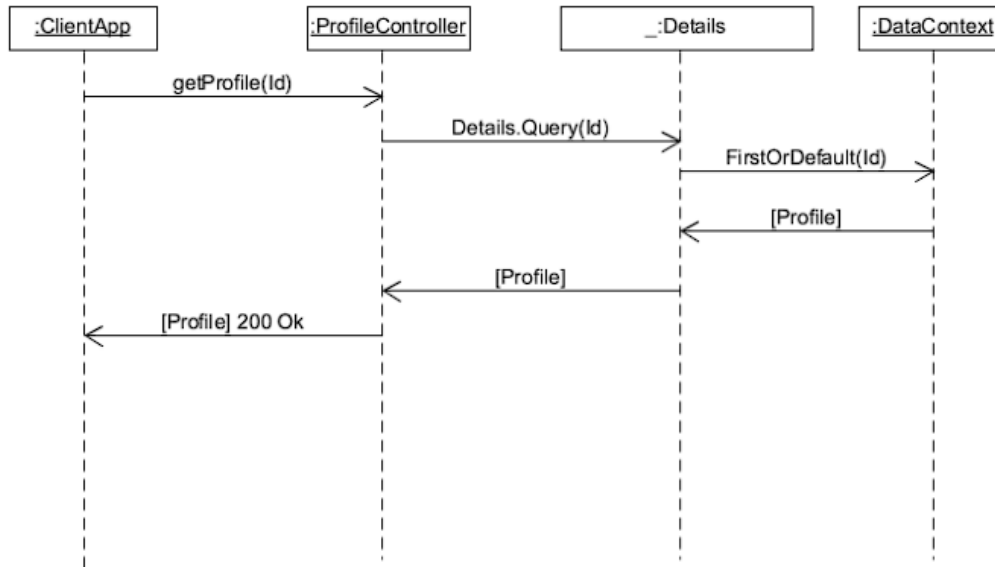
2.5.7.7 List SwapsRequestedFromMe Sequence Diagram:





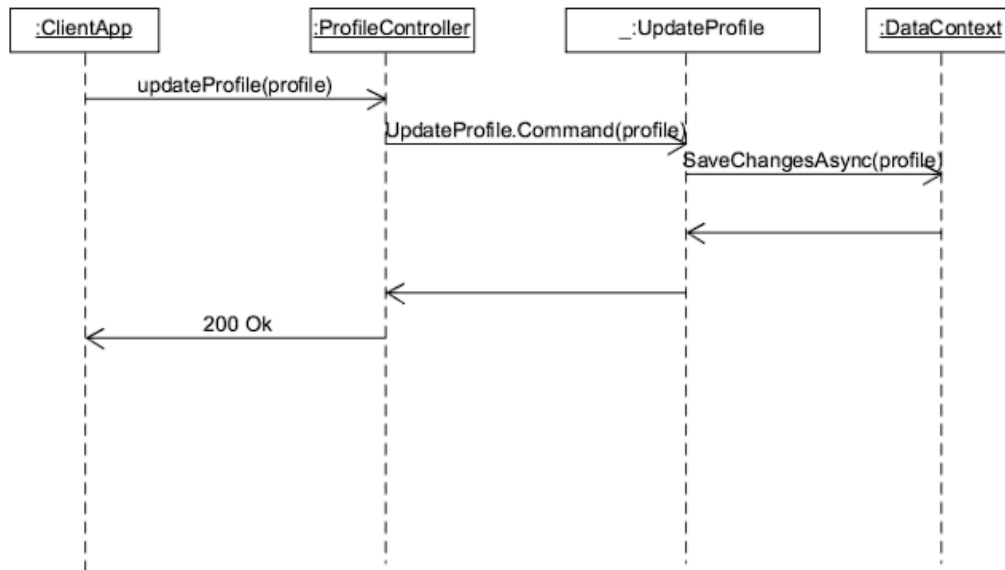
2.5.8 List Profile Sequence Diagram:

2.5.8.0 Get Profile Sequence Diagram:



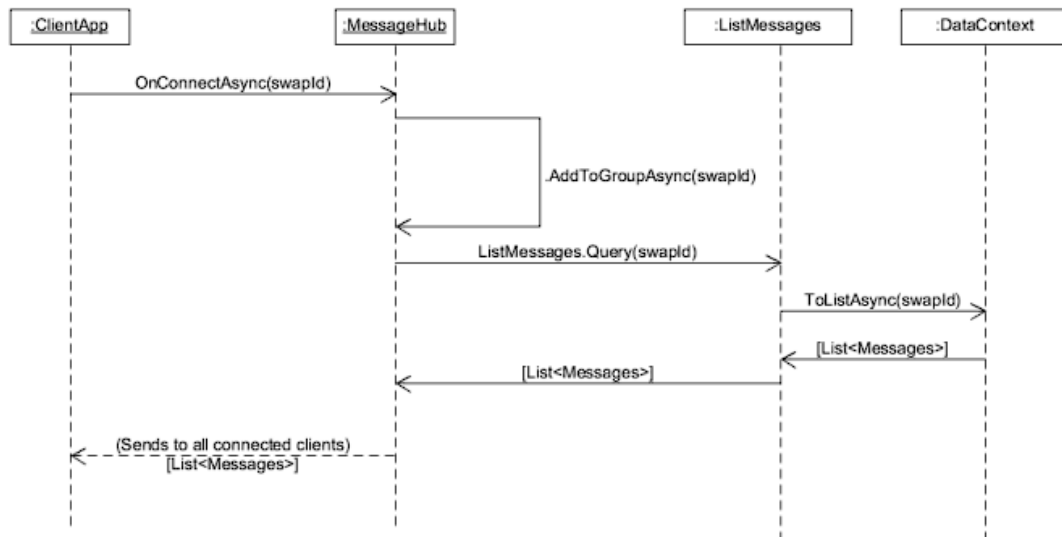


2.5.8.1 Update Profile Sequence Diagram:



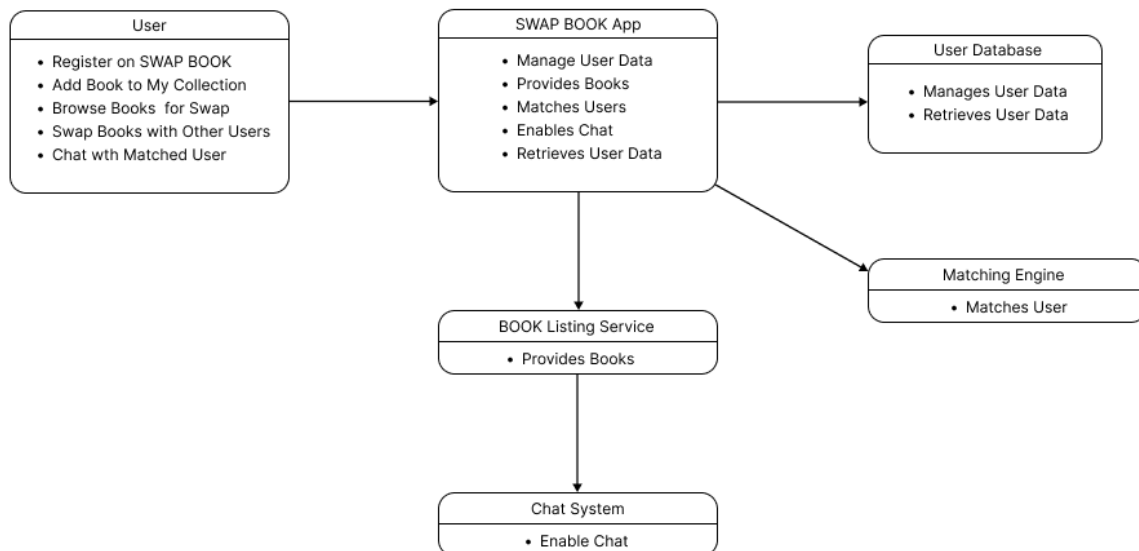


2.5.9 List Message Sequence Diagram:





2.6 Collaboration Diagram:





2.6..0 Registration Collaboration Diagram:



Figure 1: Collaboration Diagram for User Registration



2.6..1 Add Book Collaboration Diagram:



Figure 1: Collaboration Diagram for Adding a Book to My Collection



2.6.2 Book Browsing Book Collaboration Diagram:



Figure 1: Collaboration Diagram for Browsing Books for Swap



2.6.3 Swapping Collaboration Diagram:



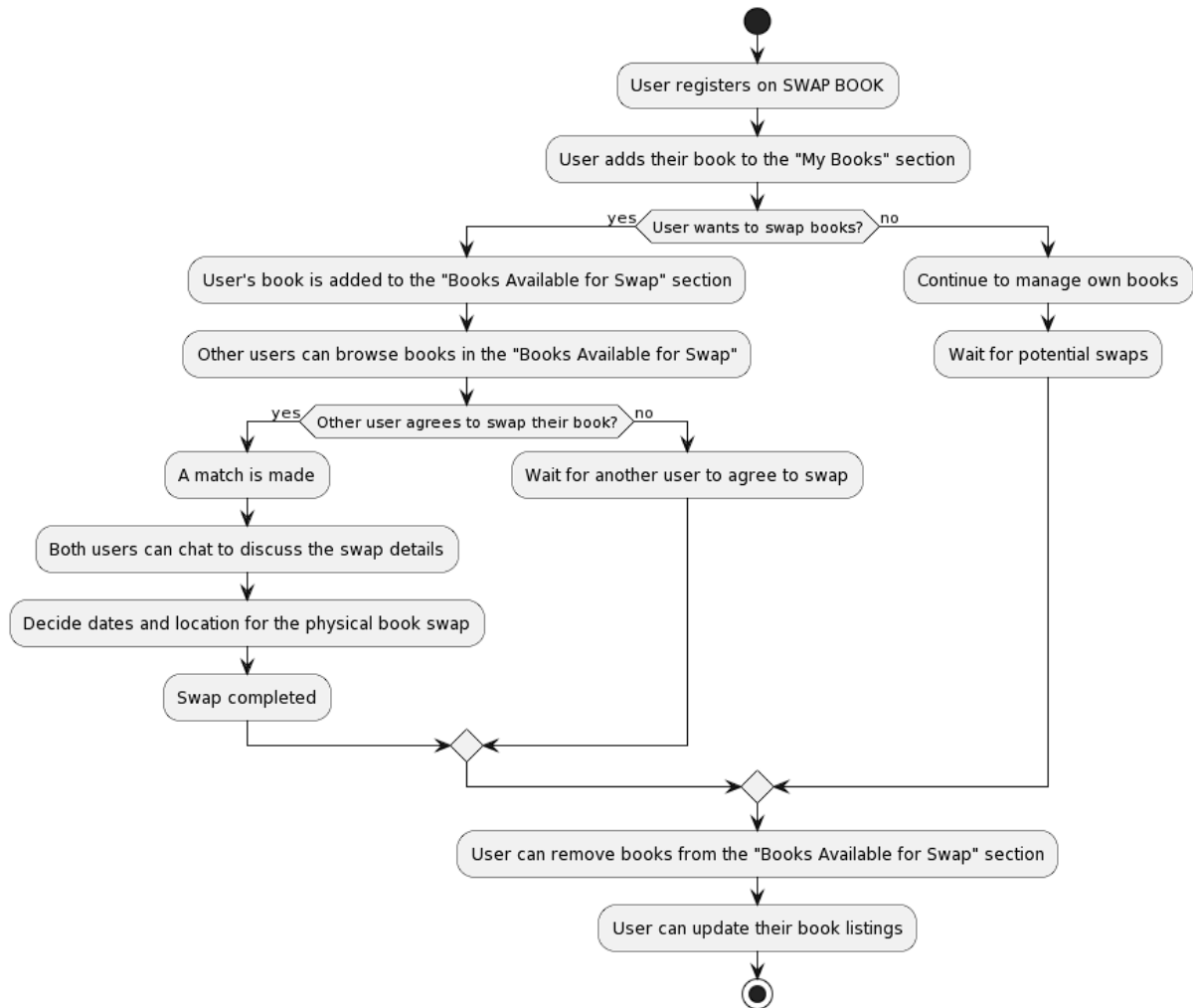
Figure 1: Collaboration Diagram for Swapping Books with Other Users

2.6.4 Chatting Collaboration Diagram:



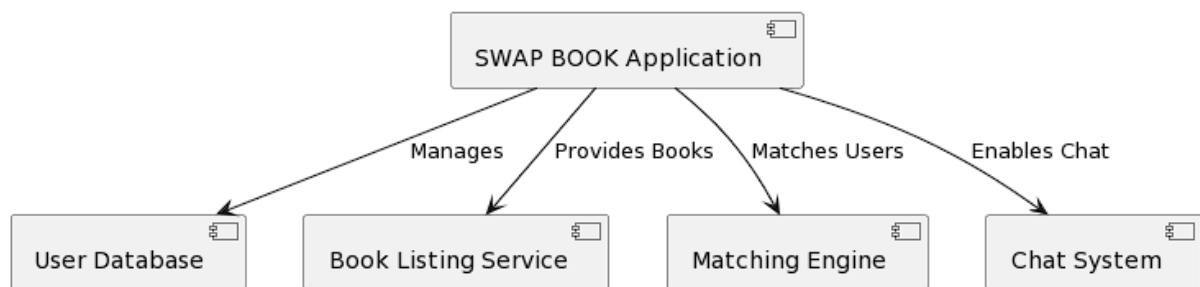
Figure 1: Collaboration Diagram for Chatting with Matched User

2.7 Activity Diagram:

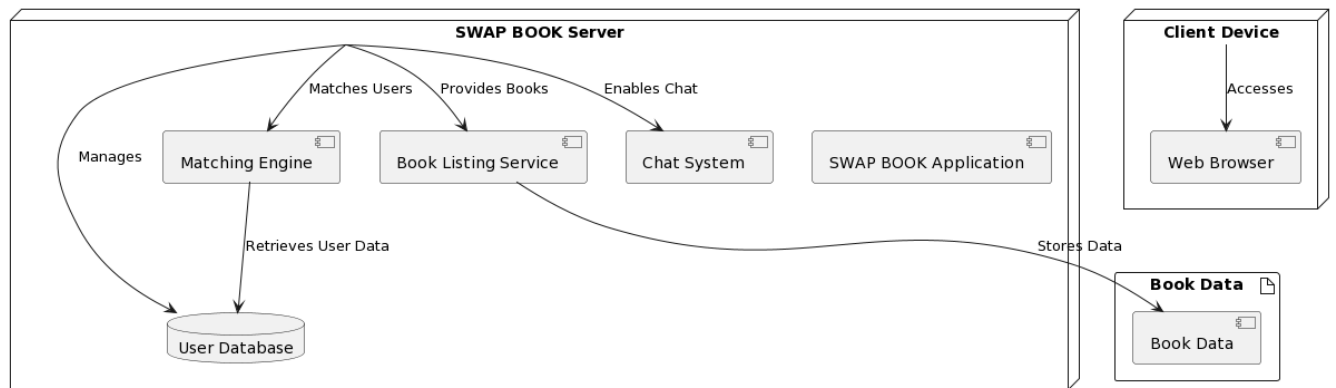




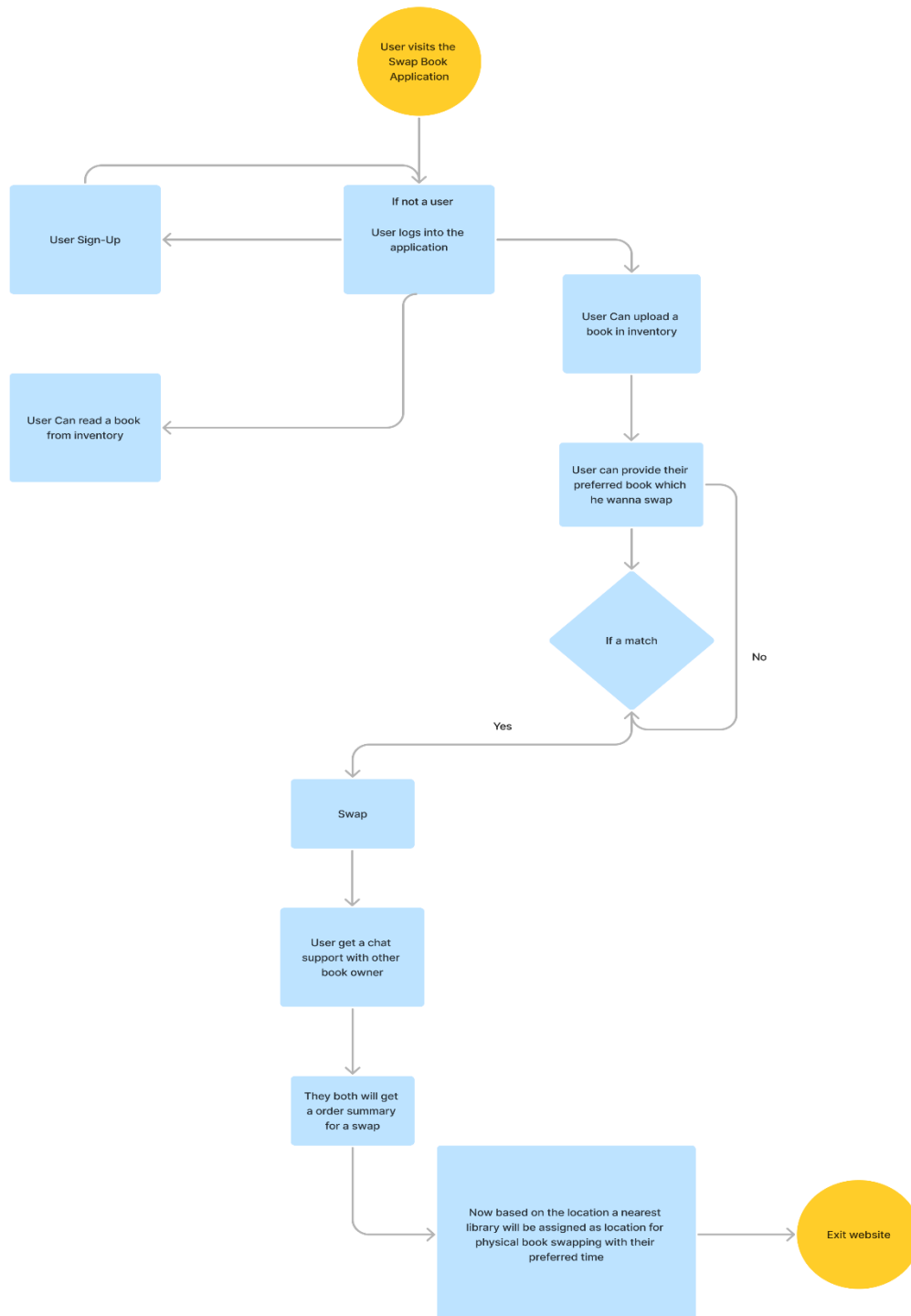
2.8 Component Diagram:



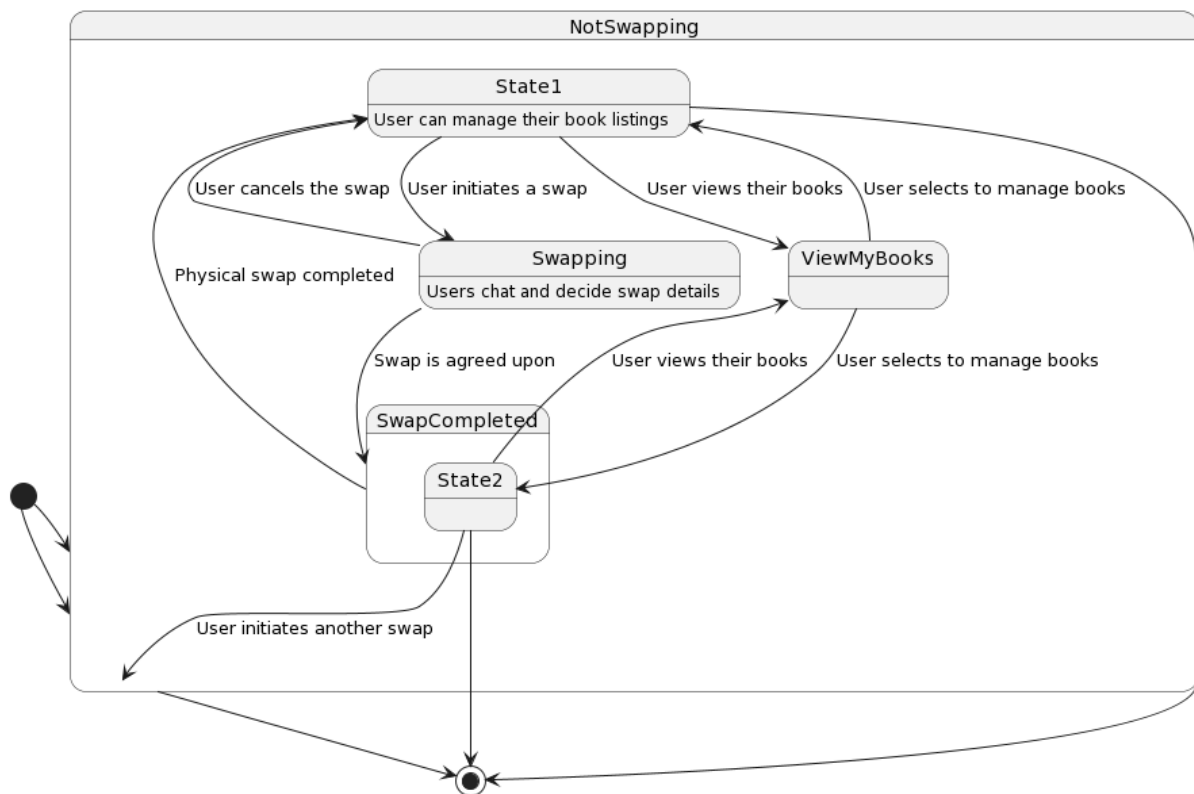
2.9 Deployment Diagram:



2.10 Flow Chart Diagram:



2.11 State Chart Diagram:





2.12 Gantt Chart Diagram:





SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY Bhopal

Department of Computer Science & Engineering

CHAPTER 3: IMPLEMENTATION

3.1 Front-End

3.1.1 User Interface Design

The front-end of "Swap Book" relies on a user interface design that prioritizes user-friendliness and responsiveness.

- **Responsive Design:** Responsive web design ensures that the "Swap Book" platform adapts seamlessly to various devices, including different screen sizes and orientations ^[1]. This approach guarantees a consistent user experience, regardless of the device used.
- **User-Friendly Interface:** A user-friendly interface is achieved through intuitive menus, clear labelling, and a straightforward information architecture ^[1]. This design approach simplifies user navigation and enhances user satisfaction.
- **Consistent Styling:** Consistency in design elements, including colour schemes and typography, maintains a professional and cohesive appearance ^[1]. It contributes to a visually appealing and understandable platform.
- **Accessibility:** Ensuring web accessibility, following standards like Web Content Accessibility Guidelines (WCAG), ensures inclusivity and legal compliance ^[1]. "Swap Book" is designed to be accessible to users with disabilities.

3.1.2 Technologies

- **HTML:** HTML (Hypertext Markup Language) is the foundation of web content. It's used for structuring web pages and organizing content.^[2] Proper HTML markup ensures that the content is well-structured and semantically meaningful, which is important for search engine optimization and accessibility.



- CSS: CSS (Cascading Style Sheets) is used for styling and layout. It allows developers to control the presentation of HTML elements, including aspects like colours, fonts, spacing, and layout. CSS plays a significant role in ensuring that the user interface is visually appealing and user-friendly.

- JavaScript: JavaScript is a versatile programming language used to implement interactive features and enhance the user experience. It allows for dynamic content updates, form validation, and the creation of interactive components like dropdown menus and sliders.

- Responsive Framework: Responsive frameworks like Bootstrap provide a set of pre-designed components and a responsive grid system. They can significantly speed up the development process, making it easier to create a consistent and responsive design. These frameworks also often include built-in accessibility features and can be customized to fit the project's unique requirements.

- Design Tools: Design tools like Adobe XD and Figma are essential for creating and refining the user interface. These tools enable designers to create wireframes, mock-ups, and interactive prototypes. They facilitate collaboration between designers and developers and help ensure that the final design aligns .

The technologies employed for front-end development include HTML, CSS, JavaScript, a responsive framework, and design tools.

- HTML: HTML forms the foundation of web content, ensuring structured and semantically meaningful content ^[2].

- CSS: CSS is used for styling and layout, enhancing the visual appeal and user-friendliness of the interface ^[3]



- JavaScript: JavaScript, as a versatile programming language, enables interactivity and dynamic content updates ^[4].
- Responsive Framework: The use of responsive frameworks like Bootstrap expedites development while providing a consistent and responsive design ^[3].
- Design Tools: Tools like Adobe XD and Figma facilitate the creation of wireframes and interactive prototypes, supporting collaboration between designers and developers ^[5].

3.1.3 User Experience (UX)

- User Flows: User flows are diagrams that outline the paths a user might take while interacting with the platform. They map out the journey from the initial landing page to specific actions, such as creating a listing, searching for books, or managing their profile. Defining user flows helps identify potential pain points and areas for improvement.
- Prototyping: Prototyping tools allow designers to create interactive mockups of the user interface.^[8] These prototypes can be used for user testing and validation, allowing real users to interact with the platform before it's fully developed. Gathering user feedback at this stage is invaluable for making adjustments and ensuring a positive user experience.^[5]
- Feedback Integration: Continuous user feedback is a critical part of the design and development process.^[6] 7]

3.2 Back-End



3.2.1 Server and Hosting

- Node.js: Node.js is a server-side runtime environment that allows for the execution of JavaScript on the server. It's known for its non-blocking, event-driven architecture, which makes it well-suited for building fast and scalable applications. Using Node.js for the back-end provides a consistent language and toolset across the full stack.

- Web Hosting: Choosing a reliable web hosting service is crucial for ensuring high availability and performance. Factors to consider when selecting a hosting provider include server uptime, scalability options, security features, and support for the technology stack being used.^[9]

3.2.2 Database Management

- MongoDB: MongoDB is a NoSQL database management system that's well-suited for storing unstructured or semi-structured data. It's an excellent choice for managing user data, book listings, and related information in a dynamic platform like "Swap Book."^[10] Proper data modelling is essential to define the structure of the database, including the relationships between entities.

3.2.3 Technologies

- Express.js: Express.js is a popular back-end framework for Node.js that simplifies the creation of web applications and APIs. It offers routing, middleware, and a range of features that streamline the development process.



- Authentication: Implementing user authentication and authorization is crucial for securing

user data and ensuring that users can access only their own data. Technologies like JSON Web Tokens (JWT) are commonly used for this purpose.^{[11][12]}

- API Design: Designing a RESTful API allows for effective communication between the front-end and back-end components of the application. A well-designed API defines the endpoints, methods, and data formats used for interactions.^[11]

- Third-Party Integrations: Depending on the project's requirements, third-party services or APIs may need to be integrated. For instance, payment gateways or location-based services might enhance the platform's functionality and user experience.

3.2.4 Security

- Data Encryption: Data encryption is a fundamental security measure for protecting sensitive user data, both in transit and at rest. Utilizing encryption protocols such as HTTPS for data in transit and encryption algorithms for data at rest is essential.

- Authentication & Authorization: Implementing robust authentication and authorization mechanisms ensures that user data is secure and that users can access only the data and features they are authorized to use.

3.2.5 Testing and Debugging



- Unit Testing: Unit testing involves testing individual components or functions in isolation to ensure they work correctly. It's a vital part of maintaining code quality and identifying issues early in the development process.^[13]
- Integration Testing: Integration testing assesses how different components of the front-end and back-end interact with each other. It ensures that the system works as a whole and that various parts communicate effectively.^[14]
- Error Handling: Robust error handling and logging mechanisms help identify and address issues promptly. When errors occur, they should be logged for diagnosis and debugging. Implementing structured error handling can also improve the user experience by providing meaningful error messages.

3.2.6 Performance Optimization

- Caching: Implementing caching mechanisms helps improve response times and reduce server load. Caching can store frequently used data or responses, reducing the need to repeatedly fetch or generate the same content.
- Load Balancing: Load balancing techniques distribute incoming traffic effectively across multiple servers or resources.



SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY Bhopal

Department of Computer Science & Engineering

CHAPTER 4 : LAYOUT



4.1 Snapshots

Fig 4.1.1 Home Page

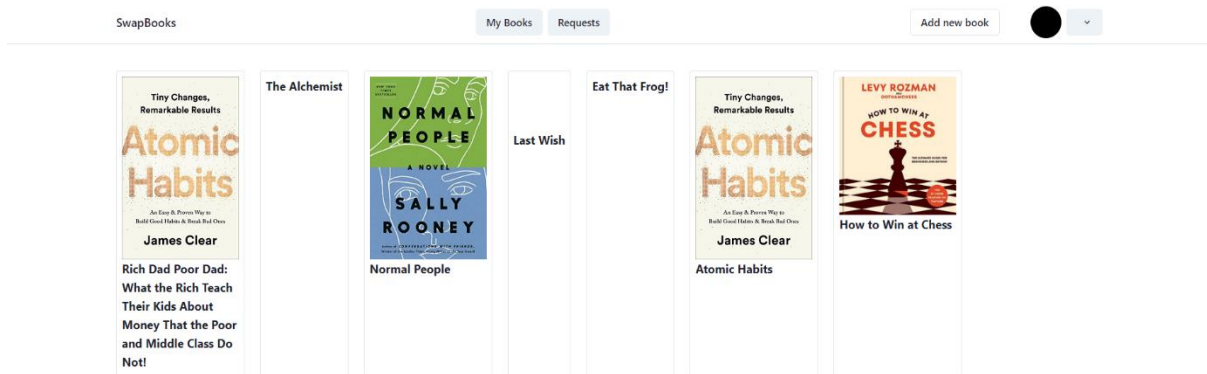


Fig.4.1.2 Home Page : Books segregated on the basis of their genre.

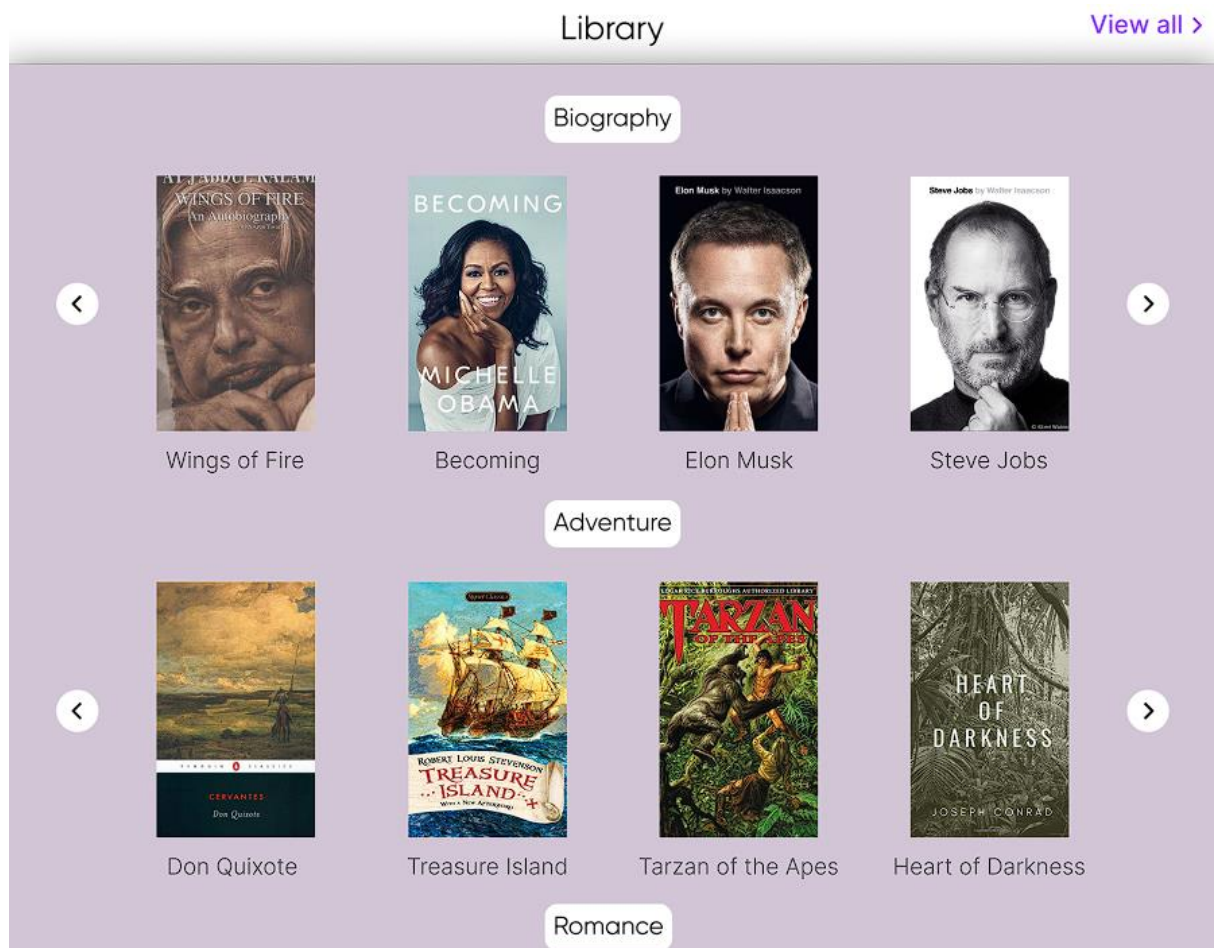




Fig 4.1.3 Login Page



You Can Login Here

or create a [new account](#) for free

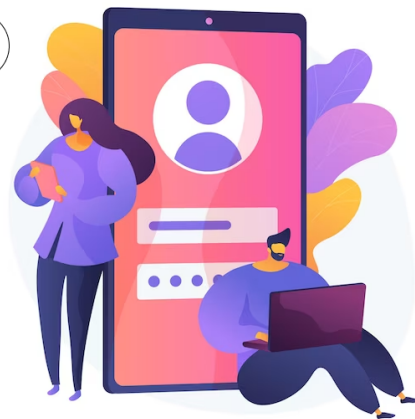


or

☐ Remember me

Log in

[Forgot your password?](#)



Library

Genres

Languages

Authors

Community

Articles

Author Interviews

Newsletter

Company

Author Services

About / Contact

Accessibility Statement

Follow

Facebook

Twitter

Instagram



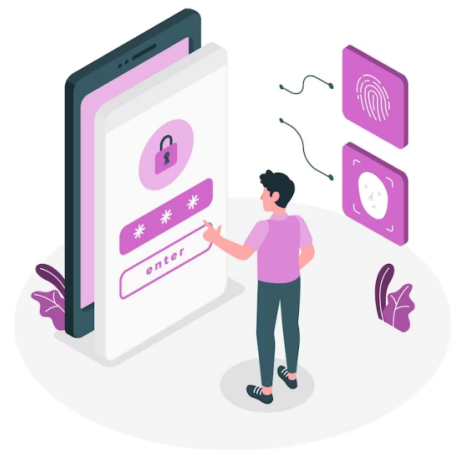
Fig 4.1.4 Register Page



Sign up to start reading

(Use phone number instead.)

[Sign up](#)



[Library](#)

[Genres](#)

[Languages](#)

[Authors](#)

[Community](#)

[Articles](#)

[Author Interviews](#)

[Newsletter](#)

[Company](#)

[Author Services](#)

[About / Contact](#)

[Accessibility Statement](#)

[Follow](#)

[Facebook](#)

[Twitter](#)

[Instagram](#)

Fig 4.1.5 About Page

Buy & Sell Second Hand Books Online In India

Welcome to Swap Books India's #1 online marketplace for books, your go-to online second hand bookstore catering to the diverse reading interests of over 2 lakh readers globally. Our robust platform bridges the gap between 1300+ sellers, 300+ bookstores, and book enthusiasts, making the quest to buy and sell books online a delightful experience. We also suggest best book to read



What Books To Read?

Our expansive library caters to every reader's taste, from thrilling mysteries and heartwarming romances to insightful biographies and enlightening non-fiction. We have more 400+ book genres separating in categories fiction and non fiction books. Books to read on Harry Potter , best books to read on self help, books to read on true crimes, books to read on psychology, best books to be read on classic literature, books to read for teenagers, books to read on military fiction, books to read on Psychological Thriller, best books to be read on travel fiction, best book to be read on history, best books to read on spirituality, top autobiography books to be read.

Best Genres Books To Read: Unveiling A World Of Reading Delights

The universe of literature offers a bounty of genres, each providing a unique lens through which we can explore the human experience, distant galaxies, historical events, and even alternate realities.



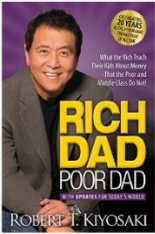
Fig 4.1.6 Inventory Page

SwapBooks

My Books

Requests

Add new book



Rich Dad Poor Dad: What the Rich Teach Their Kids About Money That the Poor and Middle Class Do Not!

Book Images

Narmadapuram, Madhya Pradesh

Delete



Fig 4.1.7 Check Out Page

SwapBooks					
		My Books		Requests	
		Add new book			
TITLE	FROM	EMAIL	LOCATION	OFFERED BOOK	ACTION
The Alchemist	Ayush Soni		Narmadapuram, Madhya Pradesh	Last Wish	Accept
The Alchemist	Ayush Soni		Narmadapuram, Madhya Pradesh	How to Win at Chess	Accept
The Alchemist	Ayush Soni		Narmadapuram, Madhya Pradesh	Last Wish	Accept
The Alchemist	Ayush Soni		Narmadapuram, Madhya Pradesh	How to Win at Chess	Accept
Eat That Frog!	rajesh malviya	rajeshmalviya003@gmail.com	Narmadapuram, Madhya Pradesh		Accept
Eat That Frog!	rajesh malviya	rajeshmalviya003@gmail.com	Narmadapuram, Madhya Pradesh		Accept



4.2 Test Cases

Test Case-1

User Register

Test Case ID	Description	Precondition	Test Steps	Expected Result	Pass/Fail
TC001	User Registration - Valid Data	None	1. Navigate to the registration page. 2. Fill in valid user details (e.g., username, email, password). 3. Click the "Register" button.	- User is registered successfully. - User is redirected to the login page with a success message.	Pass
TC002	User Registration - Invalid Data	None	1. Navigate to the registration page. 2. Fill in invalid user details (e.g., incomplete email, weak password). 3. Click the "Register" button.	- Registration fails with appropriate error messages displayed. - User remains on the registration page.	Pass
TC003	User Registration - Existing Email	Existing user with the same email exists in the database.	1. Navigate to the registration page. 2. Fill in valid user details (e.g., username, email, password) with an email that already exists. 3. Click the "Register" button.	- Registration fails with an error message indicating that the email is already in use. - User remains on the registration page.	Pass



Test Case- 1.1

User Login

TC004	User Login - Valid Credentials	Registered user with valid login credentials.	1. Navigate to the login page. 2. Enter valid login credentials (email/username and password). 3. Click the "Login" button.	- User is logged in successfully and redirected to the dashboard or main page.	Pass
TC005	User Login - Invalid Credentials	None	1. Navigate to the login page. 2. Enter invalid login credentials (e.g., incorrect email/username or password). 3. Click the "Login" button.	- Login fails with an error message indicating invalid credentials. - User remains on the login page.	Pass
TC006	User Login - Account Lockout	User has exceeded the maximum login attempts.	1. Navigate to the login page. 2. Enter incorrect login credentials multiple times until the account is locked. 3. Attempt to log in again.	- Login fails with an error message indicating that the account is locked. - User is unable to log in.	Pass
TC007	User Logout	User is logged in.	1. Click the "Logout" button or link in the application.	- User is logged out and redirected to the login page.	Pass



Test Case-2

Add a book listing

Test Case ID	Description	Input	Expected Output
TC-007	User adds a new book listing with valid data	Book Title: "The Great Gatsby" Author: "F. Scott Fitzgerald" Condition: "Like New"	Listing added successfully.
TC-008	User adds a new book listing with missing title	Author: "Jane Austen" Condition: "Good"	Listing addition failed, title missing.
TC-009	User adds a new book listing with invalid condition	Book Title: "War and Peace" Author: "Leo Tolstoy" Condition: "Unknown"	Listing addition failed, invalid condition.

Test Case-3

Searching for books

Test Case ID	Description	Input	Expected Output
TC-010	User searches for books by title	Search Query: "Harry Potter"	List of books matching the title displayed.
TC-011	User searches for books by author	Search Query: "Agatha Christie"	List of books by the author displayed.
TC-012	User searches for books by condition	Search Query: "Like New"	List of books in the specified condition displayed.

Test Case-4



initializing a book swap

Test Case ID	Description	Input	Expected Output
TC-013	User initiates a book swap with valid data	User selects a book to swap with another user's book	Swap request sent successfully.
TC-014	User initiates a book swap with non-existent book	User selects a book to swap with a non-existent book	Swap request failed, book not found.

Test Case-5

Swap book functionality

Test Case ID	Description	Preconditions	Steps	Expected Result	Pass/Fail
TC01	Verify User Registration	None	1. Click on the "Sign Up" button. 2. Fill in the registration form with valid details. 3. Click "Register."	User is registered successfully, and the user is logged in.	Pass
TC02	Verify Book Listing	User is registered and logged in.	1. Click on "List a Book." 2. Fill in the book details and upload an image. 3. Click "List."	The book is successfully listed, and it appears in the user's listing.	Pass
TC03	Verify Book Search	User is registered and logged in. A book is listed.	1. Enter book title or author in the search bar. 2. Click "Search."	The listed book is displayed in the search results.	Pass
TC04	Verify Book Exchange Request	Two registered users are logged in. Each user has at least one listed book.	1. User A views User B's book and requests an exchange. 2. User B receives the request and accepts it.	A notification is sent to User B, and the book exchange is confirmed.	Pass

Test Case-6



User Registration Data Verification

Test Case ID	Description	Input and Steps	Expected Output
TC-DB-001	Verify that user data is stored correctly during registration	- Register a new user with valid data (e.g., username, email, and password).	User data is successfully stored in the database.
TC-DB-002	Ensure that duplicate usernames are not allowed	- Attempt to register a new user with an existing username.	Registration is denied, and the database remains unchanged.
TC-DB-003	Check for email uniqueness	- Register a new user with an email that is already in use.	Registration is denied, and the database remains unchanged.

Test Case-7

User Login Data Verification

Test Case ID	Description	Input and Steps	Expected Output
TC-DB-004	Verify that a user can successfully log in	- Log in with valid credentials (e.g., username and password).	User is authenticated, and login is successful.
TC-DB-005	Ensure that invalid login attempts are logged	- Attempt to log in with incorrect credentials multiple times.	Invalid login attempts are logged in the database.
TC-DB-006	Check for tracking of successful logins	- Log in with valid credentials.	Successful login is recorded in the database.



Test Case-8

Book Listing Data Management

Test Case ID	Description	Input and Steps	Expected Output
TC-DB-007	Verify that a user can add a new book listing	- Add a new book listing with valid data (e.g., title, author, and condition).	The book listing is successfully stored in the database.
TC-DB-008	Ensure that missing data is not accepted	- Attempt to add a book listing with missing title.	The listing addition is denied, and the database remains unchanged.
TC-DB-009	Check for invalid condition data	- Add a new book listing with an invalid condition.	The listing addition is denied, and the database remains unchanged.

Test Case-9

Book Swap Data Handling

Test Case ID	Description	Input and Steps	Expected Output
TC-DB-010	Verify that users can initiate a book swap	- Initiate a book swap between two users.	The swap request is successfully recorded in the database.
TC-DB-011	Ensure that swaps with non-existent books are rejected	- Initiate a swap request with a book that does not exist.	The swap request is denied, and the database remains unchanged.
TC-DB-012	Check for successful swap completion	- Complete a book swap.	The database records the successful completion of the swap.



SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY Bhopal

Department of Computer Science & Engineering

CHAPTER 5 : APPLICATION



5.1 Advantage(s):

Sustainable Reading:

"Swap Book" plays a pivotal role in promoting a culture of sustainable reading. In an era where environmental concerns are at the forefront, the platform offers a unique advantage. By extending the lifespan of books through exchanges and sales, it actively reduces the demand for new book production. This sustainable approach contributes significantly to environmental conservation, as it minimizes the resources required for printing, binding, and distribution. Furthermore, it aligns with the principles of reducing waste and reusing valuable resources, ultimately benefiting the planet and future generations.

Cost-Efficiency:

In an age when the cost of living continues to rise, and disposable income is often limited, the financial aspect of book ownership can be a significant hurdle for many readers. Books, especially new releases, can be expensive. "Swap Book" offers an economical alternative by allowing users to access a wide variety of titles without the financial burden of purchasing each book individually. This cost-efficiency is particularly advantageous for students, budget-conscious individuals, and anyone looking to explore a diverse range of literature without breaking the bank. It democratizes access to books and makes reading more inclusive.

Community Building:

Beyond the practical advantages of book exchanges, "Swap Book" contributes to building a vibrant and interconnected community of book enthusiasts. It creates a digital space where users can engage in meaningful discussions, share their thoughts on books, offer recommendations, and connect with fellow readers who share similar interests.



5.2 Disadvantage(s):

Transaction Risks:

As with any online exchange platform, it's important to acknowledge the potential transaction risks associated with "Swap Book." Users may have concerns about the condition of the books they receive, as the quality of used books can vary. Additionally, trust is a fundamental aspect of any exchange system. Users may worry about the trustworthiness of the individuals they interact with, especially in a digital environment. Addressing these concerns and implementing measures to mitigate transaction risks is essential for building and maintaining user trust. This could involve implementing a user rating system, clear book condition descriptions, dispute resolution mechanisms, and secure payment and shipping options to ensure a safe and reliable experience for all users.



5.3 Application(s):

Educational Institutions:

"Swap Book" holds significant potential in educational institutions, particularly schools and colleges. Students often face the challenge of acquiring expensive textbooks, which they may only need for a limited duration. Once the academic year concludes, these textbooks often become obsolete. "Swap Book" can provide a convenient solution for students to exchange or sell their textbooks to fellow students. This not only alleviates the financial burden on students but also promotes the eco-friendly reuse of educational materials. It encourages responsible resource management within educational institutions.

Libraries:

Public and private libraries can leverage "Swap Book" to enhance their library services and engage the community more actively. By establishing book exchange programs within the platform, libraries can encourage patrons to contribute their old books and acquire new ones. This dynamic approach not only enriches the library's catalog but also aligns with the sustainability and community-oriented values that libraries often represent. It's an opportunity for libraries to extend their impact beyond their physical locations and reach a broader audience.

Book Clubs:

Book clubs, where enthusiastic readers gather to discuss and share their thoughts on specific titles, can greatly benefit from "Swap Book." By creating dedicated spaces within the platform, book club members can seamlessly swap books related to their reading themes. This feature fosters camaraderie among club members and enhances the overall experience. It encourages more extensive and diverse reading choices, promoting spirited discussions and literary exploration within the book club community.



Online Reading Communities:

Existing online reading communities, forums, and discussion groups can integrate "Swap Book" to provide their members with a convenient method to share and exchange physical books related to the genres and topics they discuss online. This integration expands the services of online reading communities, transforming them from digital spaces of discussion to active platforms for literary exchange. It aligns with the real-world needs and desires of the community members, strengthening the bonds among readers with similar interests.



SAGAR INSTITUTE OF RESEARCH & TECHNOLOGY Bhopal

Department of Computer Science & Engineering

CHAPTER 6 : CONCLUSION



6.1 Conclusion:

In conclusion, "Swap Book" stands as a testament to the transformative power of technology in redefining the very essence of book ownership and the reading culture. This project transcends being a mere web application; it embodies a fundamental shift in the way society perceives, values, and interacts with literature.

Our core belief is that books, beyond their physical or digital forms, are invaluable treasures meant to be read, cherished, and shared. We have squarely addressed the persisting dilemma of unused books, and in doing so, we've engineered a solution that resonates with the core tenets of sustainability and community building.

At its heart, "Swap Book" champions the principles of sustainable reading. It acts as a counterbalance to the environmental toll of book production and the ever-growing pile of discarded books. By facilitating the exchange and sale of pre-loved books, our platform not only makes reading accessible and cost-efficient but also contributes to a collective effort to reduce the carbon footprint associated with the book industry.

Moreover, "Swap Book" thrives as a testament to the innate human inclination to form communities around shared interests. It fosters a sense of belonging among book enthusiasts, allowing them to interact, connect, and discover literary treasures. In an era marked by the rapid digitization of books, our platform emphasizes the significance of tangible reading materials and the emotions they evoke.

As the world turns increasingly towards environmental consciousness and grapples with economic challenges, "Swap Book" emerges as a conduit that enables readers to engage with literature while also playing the role of



responsible global citizens. It forms a bridge between individuals who wish to share their cherished books and those eager to explore new worlds through the pages of pre-loved volumes. Our platform is not just an abstract idea; it represents a genuine solution to a tangible and shared problem, harmonizing with the evolving ethos of our society.

Looking ahead, we are committed to evolving and improving the "Swap Book" experience continually. User feedback remains at the core of our development strategy. We understand that the pulse of our platform is driven by the needs and expectations of our users. It's a journey that has only just begun, and our commitment to it is unwavering.



6.2 Future Work:

Expanded User Base:

One of our primary goals for the future is to expand our user base. We envision "Swap Book" reaching a broader audience, including educational institutions, libraries, and book clubs. By doing so, we aim to create a more dynamic and diverse community of users. An expanded user base will not only lead to an increased selection of books available for exchange but also foster a more vibrant ecosystem of readers.

Enhanced Security Measures:

Recognizing that user trust is of paramount importance, we are dedicated to continuously enhancing security measures. Our goal is to ensure that every transaction on "Swap Book" is safe, secure, and reliable. We will implement robust user verification processes and introduce features like book condition reporting to provide users with a transparent and trustworthy environment for book exchange.

Diversification of Media:

While our core focus has been on physical books, we see immense potential in diversifying into other forms of media exchange. This evolution might encompass the exchange of e-books, audiobooks, and even reading accessories and merchandise. Our vision is to become a comprehensive hub for all things related to the world of literature and reading.

Mobile Applications:

The digital world is increasingly mobile, and we acknowledge the need for flexibility. To cater to the on-the-go needs of our users, we plan to develop dedicated mobile applications for "Swap Book." These apps will provide a more



streamlined and convenient experience, allowing users to swap books anytime and anywhere.

Collaborations:

In our pursuit of promoting sustainable reading practices, we aspire to collaborate with like-minded sustainability-focused organizations, environmental initiatives, and educational institutions. These partnerships will not only expand the reach of "Swap Book" but also magnify its impact on sustainability and community building.

Continuous Feedback and Iteration:

We recognize the invaluable role that user feedback plays in shaping our platform. We are committed to remaining open to user suggestions and actively iterating on our platform to provide the best possible user experience. Our development journey is a continuous loop of learning and improving.

CHAPTER 7: IMPLEMENTATION



7.1 Code:

Components>AuthModals.tsx

```
import React from "react";

import {
  Modal,
  ModalOverlay,
  ModalContent,
  ModalHeader,
  ModalCloseButton,
  ModalBody,
  Flex,
} from "@chakra-ui/react";
import OAuthButtons from "../OAuthButtons";

import { useModalStore } from "../store/useModalStore";

const AuthModal: React.FC = () => {
  const { setOpen, open, view } = useModalStore();

  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Modal isOpen={open} onClose={handleClose}>
        <ModalOverlay />
        <ModalContent>
          <ModalHeader textAlign="center">
            {view === "login" && "Login"}
            {view === "signup" && "Sign Up"}
            {view === "resetPassword" && "Reset Password"}
          </ModalHeader>
          <ModalCloseButton />
          <ModalBody>
```



```
        display="flex"
        flexDirection="column"
        alignItems="center"
        justifyContent="center"
    >
    <Flex
        direction="column"
        align="center"
        justify="center"
        width="70%"
    >
        {view === "login" || view === "signup" ? (
            <React.Fragment>
                <OAuthButtons />
            </React.Fragment>
        ) : (
            <div></div>
        )}
    </Flex>
</ModalBody>
</ModalContent>
</Modal>
</React.Fragment>
);
};

export default AuthModal;
```



Components>BookCard.tsx

```
import { useNavigate } from "react-router-dom";

type BookCardProps = {
  id: string;
  name: string;
  ISBN: string;
};

const BookCard = ({ name, ISBN, id }: BookCardProps) => {
  const navigate = useNavigate();

  return (
    <div
      className="max-w-[200px] p-2 border rounded cursor-pointer"
      onClick={() => navigate(`/${id}`)}
    >
      <img
        src={`https://covers.openlibrary.org/b/isbn/${ISBN}-L.jpg`}
        className="w-full"
      />
      <div className="text-lg font-bold">{name}</div>
    </div>
  );
};

export default BookCard;
```




Components>NavBar.tsx

```
import React, { useState } from "react";

import AuthModals from "../AuthModals";
import { useModalStore } from "../store/useModalStore";

import { useAuthState, useSignOut } from "react-firebase-
hooks/auth";
import { auth } from "../firebase/firebase";

import {
  Button,
  Avatar,
  Menu,
  MenuButton,
  MenuList,
  MenuItem,
  Modal,
  ModalBody,
  ModalContent,
  ModalCloseButton,
  ModalHeader,
  ModalOverlay,
  useDisclosure,
  Input,
  Stack,
  Flex,
  IconButton,
} from "@chakra-ui/react";
import { ChevronDownIcon } from "@chakra-ui/icons";
import { useUploadFile, useDownloadURL } from "react-firebase-
hooks/storage";
import { ref, getDownloadURL } from "firebase/storage";
import { storage, firestore } from "../firebase/firebase";
import { collection, addDoc } from "firebase/firestore";

import { v4 as uuidv4 } from "uuid";
import { useBooksStore, Book } from "../store/useBooksStore";
```



```
import { Link } from "react-router-dom";
import { FaCrosshairs } from "react-icons/fa";

const Navbar = () => {
  const { setOpen } = useModalStore();
  const { setBook } = useBooksStore();
  const [user, loading, error] = useAuthState(auth);

  const [signOut] = useSignOut(auth);
  const [uploadFile, uploading, snapshot] = useUploadFile();

  const [isbn, setIsbn] = useState("");
  const [images, setImages] = useState<File | null>(null);
  const [location, setLocation] = useState("");

  const { isOpen, onClose, onOpen } = useDisclosure();

  const logout = async () => {
    await signOut();
  };

  const setFiles = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (!e.target.files) return;

    setImages(e.target.files[0]);
  };

  const getCurrentLocation = async () => {
    const success = async (pos: GeolocationPosition) => {
      try {
        const crd = pos.coords;

        const response = await fetch(
          `http://api.openweathermap.org/geo/1.0/reverse?lat=${crd.latitude}&lon=${crd.longitude}&appid=4e7484a149d26d11ab8954743cd85fc5`
        );
        const result = await response.json();

        setLocation(`${result[0].name}, ${result[0].state}`);
      }
    };
  };
}
```



```
    } catch (error) {
      console.log(error);
    }
  };

  const error = (err: GeolocationPositionError) => {
    console.warn(`ERROR(${err.code}): ${err.message}`);
  };

  navigator.geolocation.getCurrentPosition(success, error);
};

const saveBook = async (e: React.FormEvent<HTMLFormElement>) => {
  if (!images || !user) return;

  e.preventDefault();

  try {
    const response = await
fetch(`https://openlibrary.org/isbn/${isbn}.json`);
    const details = await response.json();

    const imageRef = ref(storage, `images/${images.name +
uuidv4()}`);
    await uploadFile(imageRef, images);

    const downloadUrl = await getDownloadURL(imageRef);

    const docRef = await addDoc(collection(firestore, "books"), {
      name: details.title,
      imageUrl: downloadUrl,
      createdBy: user.uid,
      ISBN: isbn,
      location,
    });

    const book: Book = {
      id: docRef.id,
      name: details.title,
      createdBy: user.uid,
```



```
        imageUrl: downloadUrl,
        ISBN: isbn,
        location,
    };

    setBook(book);
} catch (error) {
    console.log(error);
}
};

return (
    <nav>
        <div className="flex justify-between items-center p-2 border-
b">
            <div className="container flex justify-between w-full mx-
auto items-center">
                <Link to="/">
                    <div className="text-lg font-medium">SwapBooks</div>
                </Link>

                {user && (
                    <div className="flex gap-2">
                        <Link to="/my-books">
                            <Button>My Books</Button>
                        </Link>
                        <Link to="/requests">
                            <Button>Requests</Button>
                        </Link>
                    </div>
                )}
                <div className="flex gap-2">
                    {user ? (
                        <div className="flex items-center gap-2">
                            <Button variant="outline" mr={10} onClick={onOpen}>
                                Add new book
                            </Button>
                            <Modal isOpen={isOpen} onClose={onClose}>
                                <ModalOverlay />
                                <ModalContent>
```



```

    <ModalHeader textAlign="center">New
book</ModalHeader>
    <ModalCloseButton />
    <ModalBody
      display="flex"
      flexDirection="column"
      alignItems="center"
      justifyContent="center"
    >
      <form onSubmit={saveBook}>
        <Stack>
          <Input
            placeholder="ISBN"
            value={isbn}
            onChange={(e) =>
setIsbn(e.target.value)}
          />
          <Flex gap={2}>
            <Input
              placeholder="Location"
              value={location}
              onChange={(e) =>
setLocation(e.target.value)}
            />
            <Button onClick={getCurrentLocation}>
              <FaCrosshairs />
            </Button>
          </Flex>
          <input
            onChange={setFiles}
            type="file"
            className="border p-2 rounded"
          />
          <Button type="submit">Submit</Button>
        </Stack>
      </form>
    </ModalBody>
  </ModalContent>
</Modal>
<Avatar src={user.photoURL!} />

```



```
        <Menu>
          <MenuButton
            as={Button}
            rightIcon={<ChevronDownIcon />}
          ></MenuButton>
          <MenuList>
            <MenuItem onClick={logout}>Logout</MenuItem>
          </MenuList>
        </Menu>
      </div>
    ) : (
      <React.Fragment>
        <Button onClick={() => setOpen(true)}>Login</Button>
        <AuthModals />
      </React.Fragment>
    )}
  </div>
</div>
</nav>
);
};

export default Navbar;
```



Components>OAuthButton.tsx

```
import React, { useEffect } from "react";

import { Flex, Button, Image, Text } from "@chakra-ui/react";
import { useSignInWithGoogle } from "react-firebase-hooks/auth";
import { auth, firestore } from "../firebase/firebase";
import { setDoc, doc } from "firebase/firestore";
import { User } from "firebase/auth";
import GoogleLogo from "../assets/googlelogo.png";
import { useModalStore } from "../store/useModalStore";

const OAuthButtons: React.FC = () => {
  const [signInWithGoogle, userCred, loading, error] =
    useSignInWithGoogle(auth);

  const { setOpen } = useModalStore();

  const createUserDocument = async (user: User) => {
    const userDocRef = doc(firestore, "users", user.uid);
    await setDoc(userDocRef, JSON.parse(JSON.stringify(user)));
  };

  useEffect(() => {
    if (userCred) {
      createUserDocument(userCred.user);
    }
  }, [userCred]);

  return (
    <Flex direction="column" width="100%" mb={4}>
      <Button
        variant="oauth"
        mb={2}
        isLoading={loading}
        onClick={() => {
          signInWithGoogle();
          setOpen(false);
        }}
      />
    </Flex>
  );
};
```



```
    <Image src={GoogleLogo} height="20px" mr={4} />
    Continue with Google
  </Button>
  {error && <Text>{error.message}</Text>}
</Flex>
);
};

export default OAuthButtons;
```




Main.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.tsx";
import "./index.css";

import { ChakraProvider } from "@chakra-ui/react";
import { BrowserRouter } from "react-router-dom";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <React.StrictMode>
    <BrowserRouter>
      <ChakraProvider>
        <App />
      </ChakraProvider>
    </BrowserRouter>
  </React.StrictMode>
);
```



Store>UseStoreBooks.tsx

```
import { create } from "zustand";

export type Book = {
  id: string;
  name: string;
  imageUrl: string;
  createdBy: string;
  ISBN: string;
  location: string;
};

type BooksStoreType = {
  books: Book[];
  userBooks: Book[];
  setBook: (book: Book) => void;
  updateBooks: (books: Book[]) => void;
  updateUserBooks: (books: Book[]) => void;
};

const useBooksStore = create<BooksStoreType>((set) => ({
  books: [
    {
      id: "1",
      name: "Normal People",
      ISBN: "978-0735211292",
      imageUrl: "url",
      createdBy: "author",
      location: "Bhopal, Madhya Pradesh",
    },
  ],
  userBooks: [],
  setBook: (book) => set((state) => ({ books: [...state.books, book] })),
  updateBooks: (books) => set(() => ({ books })),
  updateUserBooks: (books) => set(() => ({ userBooks: books })),
})));
```



```
import React, { useState, useEffect } from "react";

import { useParams } from "react-router-dom";
import { firestore, auth } from "../firebase/firebase";
import {
  getDoc,
  doc,
  deleteDoc,
  addDoc,
  collection,
  getDocs,
  query,
  where,
} from "firebase/firestore";
import { Book } from "../store/useBooksStore";
import {
  Button,
  Modal,
  ModalOverlay,
  ModalFooter,
  ModalBody,
  ModalCloseButton,
  ModalHeader,
  ModalContent,
  useDisclosure,
} from "@chakra-ui/react";
import { useAuthState } from "react-firebase-hooks/auth";

const BookDetails = () => {
  const params = useParams();
  const [book, setBook] = useState<Book>();
  const [myBooks, setMyBooks] = useState<Book[]>();
  const { isOpen, onOpen, onClose } = useDisclosure();

  const [user] = useAuthState(auth);

  const deleteBook = async () => {
    try {
```



```
    await deleteDoc(doc(firestore, "books", book?.id!));

    console.log("Book was deleted");
  } catch (error) {
    console.log(error);
  }
};

const sendRequest = async (swappedBook: Book) => {
  try {
    await addDoc(collection(firestore, "requests"), {
      from: user?.displayName,
      email: user?.email,
      bookId: book?.id,
      bookOwner: book?.createdBy,
      status: false,
      location: book?.location,
      swappedBook: swappedBook.name,
    });

    console.log("Sent the request");
  } catch (error) {
    console.log(error);
  }
};

useEffect(() => {
  const getMyBooks = async () => {
    if (!user) return;

    const q = query(
      collection(firestore, "books"),
      where("createdBy", "==", user.uid)
    );

    const querySnapshot = await getDocs(q);

    const books: Book[] = [];

    querySnapshot.forEach((doc) => {
```



```
    books.push(doc.data() as Book);
  });

  setMyBooks(books);
};

const getBookDetails = async () => {
  const docSnapshot = await getDoc(doc(firestore, "books",
params.bookId!));

  const book = { ...docSnapshot.data(), id: docSnapshot.id } as
Book;
  setBook(book);

  const response = await fetch(
    `https://openlibrary.org/isbn/${book.ISBN}.json`
  );
  const result = await response.json();

  console.log(result);
};

getBookDetails();
getMyBooks();
}, [user]);

return (
  <div className="container mx-auto">
    <div className="max-w-[200px] mt-10">
      <img
        src={`https://covers.openlibrary.org/b/isbn/${book?.ISBN}-
L.jpg`}
        className="w-full"
      />
    </div>
    <div className="text-3xl font-medium mt-2">{book?.name}</div>
    <div className="text-lg font-medium border-b mt-2">Book
Images</div>
    <div className="max-w-[200px]">
      <img src={book?.imageUrl} />
    </div>
  </div>
);
```



```
</div>
<div>{book?.location}</div>
{user && user.uid == book?.createdBy && (
  <Button colorScheme="red" onClick={deleteBook}>
    Delete
  </Button>
)}
{user && user.uid !== book?.createdBy && (
  <React.Fragment>
    <Button colorScheme="green" onClick={onOpen}>
      Request
    </Button>
    <Modal isOpen={isOpen} onClose={onClose}>
      <ModalOverlay />
      <ModalContent>
        <ModalHeader>Choose a book to swap</ModalHeader>
        <ModalCloseButton />
        <ModalBody>
          <div className="flex flex-col gap-2">
            {myBooks?.map((book) => (
              <div className="p-2 rounded border flex justify-
between items-center">
                <div className="text-lg font-
medium">{book.name}</div>
                <Button
                  onClick={() => {
                    sendRequest(book);
                  }}
                >
                  Select
                </Button>
              </div>
            ))}
          </div>
        </ModalBody>

        <ModalFooter>
          <Button colorScheme="blue" mr={3} onClick={onClose}>
            Close
          </Button>
        </ModalFooter>
      </ModalContent>
    </Modal>
  </React.Fragment>
)
```



```
        </ModalFooter>
      </ModalContent>
    </Modal>
  </React.Fragment>
)}
</div>
);
};

export default BookDetails;

import React, { useState, useEffect } from "react";

import {
  Table,
  Thead,
  Tbody,
  Tr,
  Th,
  Td,
  TableContainer,
  Button,
} from "@chakra-ui/react";

import { firestore, auth } from "../firebase/firebase";
import {
  getDocs,
  query,
  where,
  collection,
  getDoc,
  doc,
  updateDoc,
} from "firebase/firestore";
import { useAuthState } from "react-firebase-hooks/auth";

type Request = {
  id: string;
  name: string;
  from: string;
```



```
email: string;
status: boolean;
location: string;
swappedBook: string;
};

const Requests = () => {
  const [user] = useAuthState(auth);
  const [request, setRequest] = useState<Request[]>([]);

  useEffect(() => {
    if (!user) return;

    const fetchRequest = async () => {
      const q = query(
        collection(firestore, "requests"),
        where("bookOwner", "==", user.uid)
      );

      const querySnapshot = await getDocs(q);

      querySnapshot.forEach(async (snapshot) => {
        const data = snapshot.data();

        const bookDoc = await getDoc(doc(firestore, "books",
data.bookId));
        const book = bookDoc.data();

        setRequest((prev) => [
          ...prev,
          {
            id: snapshot.id,
            name: book?.name,
            from: data.from,
            email: data.email,
            status: data.status,
            location: book?.location,
            swappedBook: data.swappedBook,
          },
        ]);
      });
    };
  });
}
```




```
});  
};  
  
fetchRequest();  
}, [user]);  
  
return (  
  <div className="container mx-auto mt-10">  
    <TableContainer>  
      <Table variant="simple">  
        <Thead>  
          <Tr>  
            <Th>Title</Th>  
            <Th>From</Th>  
            <Th>Email</Th>  
            <Th>Location</Th>  
            <Th>Offered Book</Th>  
            <Th>Action</Th>  
          </Tr>  
        </Thead>  
        <Tbody>  
          {request.map((r) => (  
            <Tr>  
              <Td>{r.name}</Td>  
              <Td>{r.from}</Td>  
              <Td>{r.status && r.email}</Td>  
              <Td>{r.location}</Td>  
              <Td>{r.swappedBook}</Td>  
              <Td>  
                <Button  
                  onClick={async () => {  
                    try {  
                      await updateDoc(doc(firestore, "requests",  
r.id), {  
                        status: true,  
                      });  
                      setRequest((prev) => {  
                        return prev.map((req) => {  
                          if (req.id === r.id) {  
                            return {
```



```
        ...req,  
        status: true,  
      } as Request;  
    }  
    return req;  
  });  
});  
} catch (error) {  
  console.log(error);  
}  
}}  
>  
  Accept  
</Button>  
</Td>  
</Tr>  
  )})  
</Tbody>  
</Table>  
</TableContainer>  
</div>  
);  
};  
  
export default Requests;  
  
import React, { useEffect } from "react";  
  
import BookCard from "../components/BookCard";  
  
import { useBooksStore, Book } from "../store/useBooksStore";  
import { getDocs, query, collection } from "firebase/firestore";  
import { firestore } from "../firebase/firebase";  
  
const Dashboard = () => {  
  const { books, updateBooks } = useBooksStore();  
  
  useEffect(() => {  
    const getBooks = async () => {  
      const books: Book[] = [];
```



```
const q = query(collection(firestore, "books"));
const querySnapshot = await getDocs(q);

querySnapshot.forEach((doc) => {
  books.push({ ...doc.data(), id: doc.id } as Book);
});

updateBooks(books);
};

getBooks();
}, []);

return (
  <div className="container mx-auto mt-10 flex gap-6">
    {books.map((book) => (
      <BookCard name={book.name} ISBN={book.ISBN} id={book.id} />
    ))}
  </div>
);
};

export default Dashboard;

import React, { useEffect } from "react";

import { firestore, auth } from "../firebase/firebase";
import { collection, getDocs, query, where } from
"firebase/firestore";

import { useBooksStore, Book } from "../store/useBooksStore";
import BookCard from "../components/BookCard";
import { useAuthState } from "react-firebase-hooks/auth";

const MyBooks = () => {
  const { userBooks, updateUserBooks } = useBooksStore();
  const [user] = useAuthState(auth);

  useEffect(() => {
```



```
const getUserBooks = async () => {
  if (!user) return;

  const books: Book[] = [];

  const q = query(
    collection(firestore, "books"),
    where("createdBy", "==", user.uid)
  );
  const querySnapshot = await getDocs(q);

  querySnapshot.forEach((doc) => {
    books.push({ ...doc.data(), id: doc.id } as Book);
  });

  updateUserBooks(books);
};

getUserBooks();
}, [user]);

return (
  <div className="container mx-auto">
    <div className="text-3xl font-medium mt-2">My Books</div>
    <div className="flex gap-6 mt-2">
      {userBooks.map((book) => (
        <BookCard
          ISBN={book.ISBN}
          name={book.name}
          key={book.id}
          id={book.id}
        />
      ))}
    </div>
  </div>
);
};

export default MyBooks;
```



7.1 Database Snippets:

The screenshot shows the Cloud Firestore console for a project named 'swapbook'. The 'Data' tab is selected, and the 'books' collection is chosen. A document with ID '8fVXPfQEPH1ZWzB4Hurl' is displayed. The document contains the following fields:

Field	Value
ISBN	"978-1-61268-019-4"
createdBy	"N1mZqa0khEfVDPXBKjvUjm6ecEC3"
imageUrl	"https://firebasestorage.googleapis.com/v0/b/swapbook-e335b.appspot.com/o/images%2FScreenshot%202024-04-29%20191601.png?ad7e784-ee87-4933-9f9a-abcc5ca48cd?alt=media&token=785445db-48c3-4afe-b917-011a8d6fc19e"
location	"Narmadapuram, Madhya Pradesh"
name	"Rich Dad Poor Dad: What the Rich Teach Their Kids About Money That the Poor and Middle Class Do Not!"

The screenshot shows the Cloud Firestore console for the same project, displaying a document in the 'requests' collection with ID '5e3b9PhCc2B9i6Kmpuk'. The document contains the following fields:

Field	Value
bookId	"CKEAd7N3thMQ7OPDORB"
bookOwner	"N1mZqa0khEfVDPXBKjvUjm6ecEC3"
email	"soni.ayush.2212@gmail.com"
from	"Ayush Soni"
location	"Narmadapuram, Madhya Pradesh"
status	false
swappedBook	"Last Wish"



🏠 > requests > 8cMEXRONCYXfMhVAbChz More in Google Cloud		
(default)	requests	8cMEXRONCYXfMhVAbChz
+ Start collection	+ Add document	+ Start collection
books	5e3b9PhCc2B9ii6Kmpuk	+ Add field
requests >	8cMEXRONCYXfMhVAbChz >	bookId: "RgKISfXRDN1FIHLcxDJ"
	KXS146gfr50NblV4IghI	bookOwner: "N1mZqa0khEfvDPXBJvUjm6ecEC3"
	WLjeZoFbypTahZfi08qC	email: "rajeshmalviya003@gmail.com"
	jYIqq7EwwQ6KPk8w45ZP	from: "rajesh malviya"
		location: "Narmadapuram, Madhya Pradesh"
		status: true

🏠 > requests > KXS146gfr50Nb. More in Google Cloud		
(default)	requests	KXS146gfr50NblV4IghI
+ Start collection	+ Add document	+ Start collection
books	5e3b9PhCc2B9ii6Kmpuk	+ Add field
requests >	8cMEXRONCYXfMhVAbChz	bookId: "RPADTQ55W9CjwXjqoLW0"
	KXS146gfr50NblV4IghI >	bookOwner: "izzf0V60JBnm7rDikSlkvif4QvA2"
	WLjeZoFbypTahZfi08qC	email: "spectre.fury.2212@gmail.com"
	jYIqq7EwwQ6KPk8w45ZP	from: "Ayush Soni"
		location: "Kolar Tahsil, Madhya Pradesh"
		status: false
		swappedBook: "How to Win at Chess"



REFERENCES

1. Krug, Steve. "Don't Make Me Think." New Riders, 2018.
2. Wroblewski, Luke. "Mobile First." A Book Apart, 2019.
3. Bootstrap Documentation. [Online] Available:
<https://getbootstrap.com/docs/5.0/getting-started/introduction/>
4. HTML, CSS, and JavaScript Documentation. [Online] Available: Official documentation and tutorials.
5. Adobe XD and Figma. [Online] Available:
<https://www.adobe.com/products/xd.html>, <https://www.figma.com/>
6. Jesse James Garrett. "The Elements of User Experience." New Riders, 2021.
7. Usability.gov. [Online] Available: <https://www.usability.gov/>
8. User Feedback Integration Resources. [Online] Available: Online resources and articles on user feedback integration.
9. Node.js Documentation. [Online] Available: <https://nodejs.org/en/docs/>
10. Shannon Bradshaw, Eoin Brazil, Kristina Chodorow. "MongoDB: The Definitive Guide." O'Reilly Media, 2020.
11. Web Content Accessibility Guidelines (WCAG). [Online] Available:
<https://www.w3.org/WAI/standards-guidelines/wcag/>
12. JSON Web Tokens (JWT) Documentation. [Online] Available:
<https://jwt.io/introduction/>
13. "Testing JavaScript Applications" by Lucas da Costa.
14. Jest Documentation. [Online] Available: <https://jestjs.io/>



15. National Literacy Trust. (2015). "Children's Reading and Technology: Findings from the 2015 National Literacy Trust Survey." Retrieved from <https://literacytrust.org.uk/research-services/research-reports/children-and-their-reading/children-and-their-reading/research-2015-children-and-their-reading/>
16. Tata Literature Live! (2014). "Tata Literature Live! Survey Reveals Interesting Reading Habits of Mumbaikars." Retrieved from <https://timesofindia.indiatimes.com/life-style/books/Tata-Literature-Live-Survey-Reveals-Interesting-Reading-Habits-of-Mumbaikars/articleshow/45169435.cms>
17. Wattpad. (n.d.). "About Wattpad." Retrieved from <https://www.wattpad.com/about>
18. Amazon Kindle. (n.d.). "About Amazon Kindle." Retrieved from <https://www.amazon.com/Amazon-Kindle-Books/b?node=133141011>
19. OLX Group. (n.d.). "About OLX Group." Retrieved from <https://www.olxgroup.com/>