TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

LUMBINI CITY COLLEGE

A PROJECT REPORT

ON

PUSTAKHUB - A COMPLETE SYSTEM FOR BOOK LOVERS TO CONNECT AND SELL OLD BOOKS

Submitted By:

Aashish Adhikari [T.U Exam Roll no.: 21946/075]

Poonam Kunwar [T.U Exam Roll no.: 21951/075]

In the partial fulfillment of the requirement for the Bachelor's Degree in Computer Science and Information Technology

April, 2023

LETTER OF APPROVAL

This is to certify that this project prepared by **AASHISH ADHIKARI** and **POONAM KUNWAR** entitled **"PUSTAKHUB - A COMPLETE SYSTEM FOR BOOK LOVERS TO CONNECT AND SELL OLD BOOKS"** in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in scope and quality as a project for the required degree.

……..…… Suraj Khatri Supervisor Lumbini City College	…… Rahul Bhusal Sharma Campus Chief Lumbini City College
……..……..…… Upendra Raj Joshi External Examiner IOST, Tribhuvan University	…….……. Kamal Shrish Internal Examiner Lumbini City College

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision entitled **"PUSTAKHUB - A COMPLETE SYSTEM FOR BOOK LOVERS TO CONNECT AND SELL OLD BOOKS"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

……

Suraj Khatri
Lecturer
Lumbini City College

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from many people, and we feel extremely fortunate to have got all this all along the completion of our final year project. Whatever we have done is only due to such guidance and assistance and we would not forget to thank them. We could not have completed this project without help from our college "**LUMBINI CITY COLLEGE**" which gave the supportive hands before us academically, also regarding other activities related to Information Communication Technology as well as extra curriculum activities that we got to participate in. It gave us family like environment.

We take this opportunity to express our profound gratitude and deep regards to our supervisor **Mr. Suraj Khatri** for his exemplary guidance, monitoring, and constant encouragement throughout the course of this thesis.

We would also like to appreciate the help of various people for providing us essential data required for our project as well as their time in guiding us so as to how our project will be more effective in users prospective. We are also grateful to our teachers for their constant support and guidance.

Finally, we would like to express our sincere thanks to all the friends and others who helped us directly or indirectly during this project work. This project has been a wonderful experience where we have learnt and experienced many beneficial things.

Aashish Adhikari (21946/075)

Poonam Kunwar (21951/075)

ABSTRACT

This project has been submitted in the fulfillment of the requirements for the Bachelor of Science in Computer Science and Information Technology. We, the team members of this project, take pleasure in presenting the detailed project report that will reflect our efforts in this semester.

PUSTAKHUB - A COMPLETE SYSTEM FOR BOOK LOVERS TO CONNECT AND SELL OLD BOOKS aims to provide a platform for users to list their old books to sell and chat with the seller/buyer. In this system, users can find the books as per their reading habits with the help of the String Similarity Algorithm based on Levenshtein Distance.

Keywords: *Old Books, Web Application, Online, Chat, System, Connect.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheet
DB	Database
DFD	Data Flow Diagram
ER	Entity Relationship
HTML	Hypertext Markup Language
IO	Input/Output
JS	Java Script
NoSQL	Not only Structured Query Language
PDF	Portable Document Format
SDLC	System Development Life Cycle
SQL	Structured Query Language
UI	User Interface
USA	United States of America

CHAPTER 1: INTRODUCTION

1.1 Introduction and Background

Since our childhood, we have read so many books. They may be academic or novels or comics. Some books are closest to our heart, which we want to keep forever, but most of the books are treated as trash, and we sell it by weighing in kilos. Still, there are many students who cannot afford the new books, but the same books which are staring at us from the dark corner of our room may have so much impact on someone's life.

Availability of books can be a major problem when shopping at a physical store, especially if the store has a limited selection or if the book or edition that a customer is looking for is out of stock. Physical bookstores often have limited space and need to prioritize the most popular or recent titles, which can result in less demand for older or more obscure books. Additionally, physical stores may have limited buying power compared to online retailers, which can make it more difficult to keep a wide variety of books in stock.

With this there is demand for a system which lets consumers to buy and sell books in an affordable, sustainable, and convenient way. This is due to the rising cost of new books, which can make it difficult for some consumers to afford the books they need or want. Additionally, the growing concern for the environment has led to a trend towards more sustainable and eco-friendly options, which can include buying used books instead of new ones.

PUSTAKHUB - A COMPLETE SYSTEM FOR BOOK LOVERS TO CONNECT AND SELL OLD BOOKS is a web-based application that tends to offer a platform for buying and selling used and rare books, which can be a valuable resource for readers, collectors, and anyone looking for out-of-print or hard-to-find books. This system also offers a platform to connect with other book lovers, readers and discover new books and authors.

1.2 Problem Statement

Buying books from a physical store can present several challenges. These include limited availability of certain titles, higher prices, limited information about the condition and availability of used books and the inconvenience of having to visit multiple stores to find what we are looking for.

Furthermore, many people have old books in their home which may be of no use to them. There is a lack of proper system which lets users sell those books by connecting person to person.

There is a lack of efficient system specifically for book lovers who want to connect. Discover new books, build a community, book swaps and exchanges and meet new people.

1.3 Objectives

PustakHub - a complete system for book lovers to connect and sell old books is created with the aim of providing a platform for individuals to sell their used books to other book lovers in a convenient and cost-effective manner. The main objectives of this project work are:

- To design a system using which one can list their old books to sell and connect directly with the seller to purchase.
- To recommend the best books based on their preferences using the String Similarity Algorithm based on Levenshtein Distance.
- To allow customers to connect with people having similar interests.

1.4 Scope and Limitations

PustakHub is the system developed for providing an efficient and reliable platform to connect bookaholics. Increase in use of internet helps to reach a much larger audience than selling books through traditional brick-and-mortar stores. This project will focus exclusively on finding the required book in an efficient and cost-friendly way. The scope of our system remains within a large set of bookaholics and the goal is to provide users with a huge catalog of books that will satisfy the needs to the maximum. The sole purpose of our system is to recommend the best set of deals and connect seller and buyer directly eliminating the any middleman in between.

Limitations

- The delivery service for the product is not available.
- The payment system is not integrated in the system.

1.5 Development Methodology

For the development of our system, a Prototyping Methodology has been chosen considering all the requirements.

Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved. It also creates a base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail. It is an iterative, trial and error method which takes place between developer and client [1].

The prototyping model is a useful approach for software development, particularly when requirements are not well defined or when there is a need for rapid development.

Prototyping models are commonly used in agile development methodologies, where the focus is on delivering working software quickly and incrementally. By creating a working prototype early in the process, developers can get a better understanding of the client's requirements and adjust the development process accordingly. This can save time and money by identifying and resolving issues early on in the development process [2].

The main advantage of this model is that emphasizes user involvement and feedback throughout the development process. This helps to ensure that the final product meets the needs and expectations of the users. This model can help reduce development time and cost by identifying and addressing design issues early in the development process. This can help prevent costly and time-consuming changes later in the development cycle.

Also, the prototyping model can be used to identify and clarify requirements for the system. By building a working model of the system, stakeholders can better understand the functionality and design of the system and provide feedback on areas for improvement.

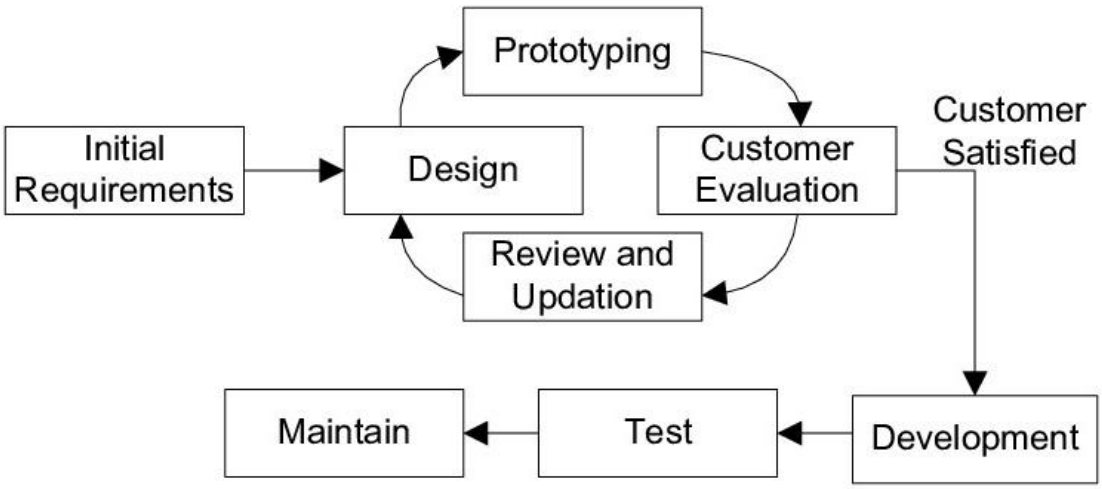

Figure 1. 1: Prototyping Model

This model works best in scenarios where not all of the project requirements are known in detail ahead of time hence this methodology is the best choice for this project.

1.6 Report Organization

Table 1. 1: Report Organization Table

Preliminary Section	• Title Page • Certificate Page • Acknowledgement • Abstract Page
Introduction Section	• Introduction and Background • Problem Statement • Objectives • Scope and Limitation • Development Methodology • Report Organization
Background study and Literature Review Section	• Background Study • Literature Review
System Design Section	• Design • Algorithm Details
Implementation and Testing Section	• Implementation • Testing • Result Analysis
Conclusion and Future Recommendation Section	• Conclusion • Future Recommendation

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Selling second-hand books online has become an increasingly popular way to make money from unwanted books while providing affordable options to readers. The market for used books is driven by factors such as the increasing popularity of e-commerce, a growing interest in sustainable consumption, and the rising cost of new books. Online platforms such as Amazon, eBay, and AbeBooks offer opportunities for individuals and small businesses to sell their used books to a global audience. However, the market for used books is highly competitive, and successful sellers must prioritize factors such as accurate book descriptions, competitive pricing, and fast shipping. They must also be aware of trends such as the rise of e-books and the growing popularity of audiobooks.

Despite the growth and success of online second-hand book selling websites, there are several challenges that these platforms face. One of the major challenges is maintaining the quality of the books being sold. Unlike traditional bookstores, online second-hand book selling websites rely on individual sellers to provide accurate descriptions of the books they are selling. As a result, some consumers may receive books that are in poor condition or not as described.

The main aim of the online book selling companies is to assure the availability of the wide range of the books, allowing the consumers to connect directly to avoid the any middleman in between which provides the better dealing experience.

2.2 Literature Review

Online second-hand book selling systems are useful for consumers who are looking for affordable and sustainable ways to access books. Buying used books is often much cheaper than buying new books, and reusing is an environmentally friendly option. Also, online these systems can be useful for bookstores and libraries who want to dispose of excess inventory or raise funds for their operations.

Collectors and enthusiasts can benefit from second-hand book selling systems because they provide a way to find rare or out-of-print titles. Online sellers often have a wide selection of hard-to-find books, making it easier for collectors to find the titles they are looking for.

Existing System

1. **PuranoBooks -** is a platform where we can sell and buy used books. This only allows the customers to request a quote for the required book. Thus, their current system is not effective enough to meet the proper expectations of customers. The system should not have hidden price tags and there is no means of directly connecting with the seller and buyer.
2. **Alibris -** is an online marketplace for independent sellers of new and used books, music, and movies. Alibris provides a platform for sellers to list and sell their items to a global audience, and for buyers to access a wide selection of books, music, and movies from independent sellers [3].
3. **BookChor -** is an Indian online marketplace for second-hand books. It was founded in 2015 and is based in New Delhi, India. One of the unique features of BookChor is its "Blind Book Date" concept, where buyers can purchase a mystery box of books at a discounted price without knowing what books are included in the box. This feature has proven popular with buyers who are looking for a surprise and are willing to take a chance on books they may not have considered otherwise.

   BookChor also offers a range of other services and features, including a loyalty program called "Chor Club" that rewards frequent buyers, and an option for sellers to donate a portion of their earnings to a charity of their choice [4].

CHAPTER 3: SYSTEM ANALYSIS

3.1 System Analysis

Seeing the possibility of frequent changes that may occur during the development, the Prototyping model was used to develop the application. The detailed methodologies used to develop the application are described in the following subsections.

3.1.1. Requirement Analysis

i. Functional Requirement

The functional requirements of PustakHub System are as follows:

- **User Module** - User can search for the product and view its description.
- **Chat Module** - Seller/Buyer can interact with each other's via chat.
- **Levenshtein Distance Algorithm Module-** This module will look for all the product names available and measures the distance between them to filter out similar products.

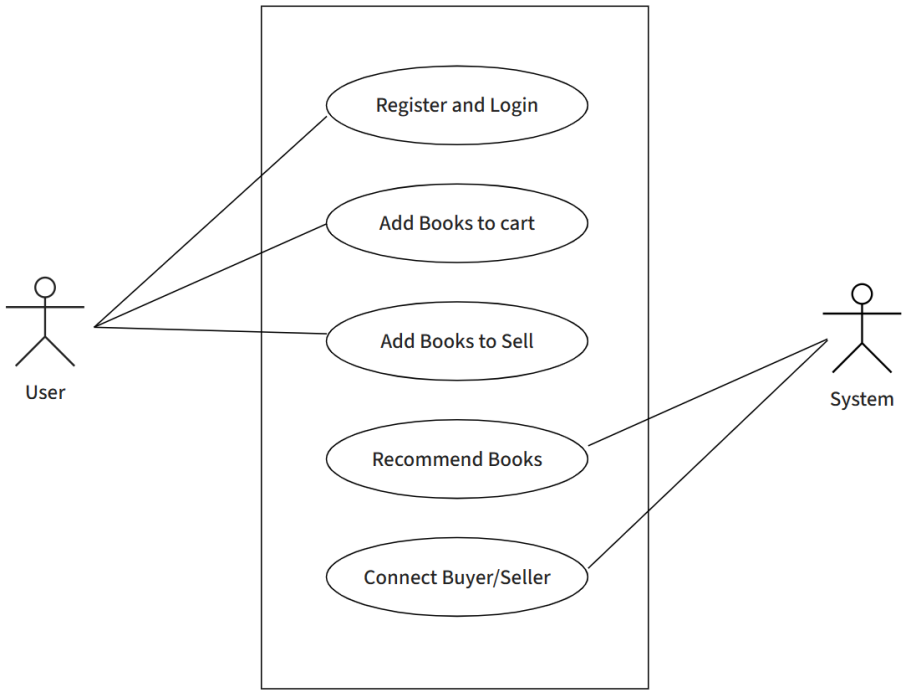

Figure 3. 1: Use Case Diagram for Functional Requirement

ii. Non-functional requirement

. The non-functional requirements of PustakHub System are as follows:

- **User friendly:** The system is user-friendly enough to meet the knowledge and skill of general users. Even the common users who have knowledge of the internet can run this system.
- **Reliability:** The system is effective with a reliable source and tools, so the outcome is reliable. It is highly reliable as it is free of errors.
- **Speed:** The response time of the system is fast. During the time of system progress or runtime while using system and viewing the details system doesn't make user feel as low response time.
- **Availability:** The system is error free, and the rate of system failure is very low. The system is available all the time as required by the user.

3.1.2 Feasibility Analysis

i. Technical Feasibility

The technical portion of the system consist of various tools and algorithms to design and develop the system called "PustakHub – A Complete System for Book Lovers to Connect and Sell Old Books" which are selected carefully and meet high standard for designing the System Architecture and developing it. Various Online tools Like Lucid Chart and offline tools like VS Code with associated packages are used to develop the system which confirms the reliability of the system.

ii. Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems. In this case project, the developed software mostly relies on open-source libraries like React, Material UI, Node JS. So, the project is expected to work without any problem. In the case of background, the algorithm used in our project is Levenshtein Distance Algorithm. This application uses this algorithm to calculate the distance between the products. The system also uses Socket Programming to establish the communication links between the users. Hence, the system is feasible operationally.

iii. Economic Feasibility

The purpose of economic feasibility is to determine the positive economic benefits. In the context of the project work, the system developed is a web application: which requires all the hardware and software support as required by other applications. The cost regarding the development of the application is minimum. All the Software that will be used during the development process is from the open-source

community. Additionally, the system doesn't rely on any paid API's. So, there are no such drawbacks of this application based on costs.

iv. Schedule Feasibility

With the number of resources and platform used, the project seems to be completed within the estimated time period. So, the system is feasible as per the academic schedule.

Activity	1st Week	2nd Week	3rd Week	4th Week	5th Week	6th Week	7th Week	8th Week	9th Week	10th Week	11th Week	12th Week
Research & Analysis												
Design												
Coding												
Testing												
Documentation and Report												

Figure 3. 2: Gantt Chart

3.1.3 Analysis

i) Data modeling using E-R Diagram

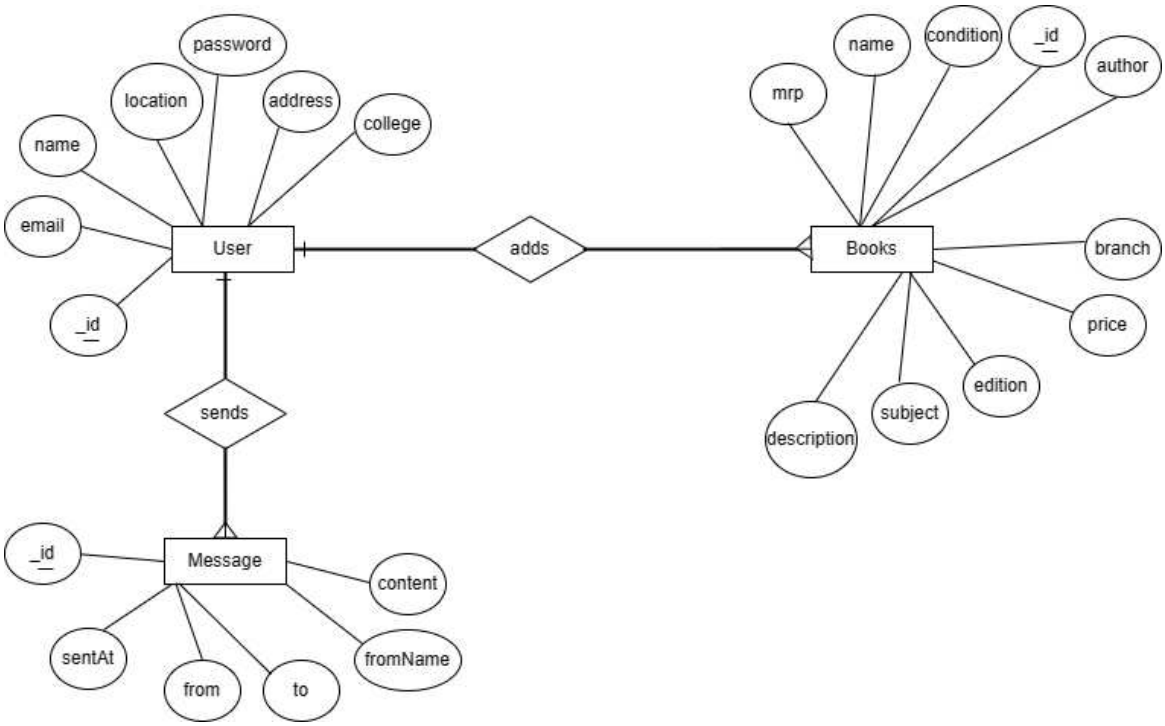

Figure 3. 3: E-R diagram

ii) Level 0 DFD

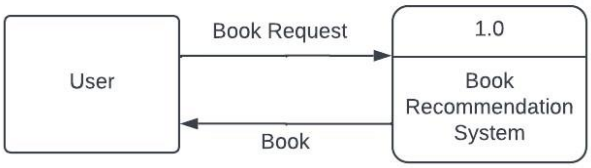

Figure 3. 4: Level 0 Data Flow Diagram

iii) Level 1 DFD

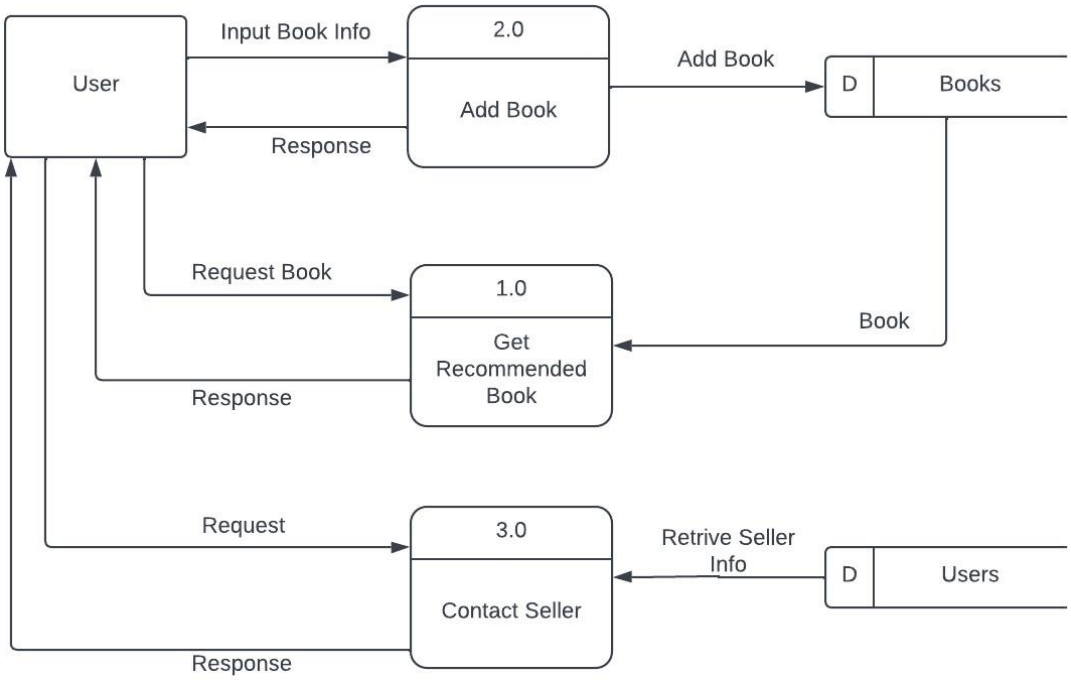

Figure 3. 5: Level 1 Data Flow Diagram of Recommendation System

CHAPTER 4: SYSTEM DESIGN

4.1 Design

System design refers to the process of defining the architecture, components, modules, interfaces, and data for a software system. It involves creating a blueprint or plan for the system that specifies how it will be built, how it will function, and how it will meet the requirements of the clients.

Database Design

Table 4. 1: Users Table

S.No.	Attribute	Datatype	Constraint
1	_id	ObjectId	Primary Key
2	name	String	Not Null
3	email	String	Not Null
4	password	String	Not Null
5	college	String	Not Null
6	location	String	Not Null
7	profilePic	String	Nullable
8	resetPasswordToken	String	default
9	verifyEmailToken	String	default
10	postedBooks	ObjectId	Not Null

11	createdAt	Date	Not Null
12	updatedAt	Date	Not Null

Table 4. 2: Books Table

S.No.	Attribute	Datatype	Constraint
1	_id	ObjectId	Primary Key
2	wishListedBy	Array	Nullable
3	isSold	Boolean	Nullable
4	tags	Array	Nullable
5	createdAt	Date	Not Null
6	updatedAt	Date	Not Null
7	bookName	String	Not Null
8	subject	String	Not Null
9	price	Int32	Not Null
10	condition	String	Not Null
11	author	String	Not Null
12	priceType	String	Not Null
13	mrp	Int32	Not Null
14	branch	String	Not Null
15	noOfPages	Int32	Not Null
16	edition	String	Not Null

17	description	String	Not Null
18	selectedFile	String	Not Null

Table 4. 3: Messages Table

S.No.	Attribute	Datatype	Constraint
1	_id	ObjectId	Primary Key
2	sentAt	Date	Not Null
3	from	ObjectId	Not Null
4	to	ObjectId	Not Null
5	content	String	Not Null
6	fromName	String	Not Null

4.2 Algorithm Details

The Levenshtein Distance algorithm is a String similarity algorithm used to measure the difference between two strings. It calculates the minimum number of single-character edits (insertions, deletions, or substitutions) needed to transform one string into another.

The algorithm works by building a matrix that represents the distance between each pair of characters in the two strings. The matrix is initialized with values that represent the cost of transforming an empty string into each of the two input strings. Then, for each pair of characters in the two strings, the algorithm calculates the cost of transforming one character into the other and updates the matrix accordingly.

Once the matrix is complete, the Levenshtein Distance is the value in the bottom-right corner of the matrix. This value represents the minimum number of edits needed to transform one string into the other.

For example, the Levenshtein distance between “kitten” and “sitting” is 3 since, at a minimum, 3 edits are required to change one into the other.

kitten → sitten (substitution of “s” for “k”)

sitten → sittin (substitution of “i” for “e”)

sittin → sitting (insertion of “g” at the end).

An “edit” is defined by either an insertion of a character, a deletion of a character, or a replacement of a character [5].

In approximate string matching, the objective is to find matches for short strings in many longer texts, in situations where a small number of differences are to be expected. The short strings could come from a dictionary, for instance. Here, one of the strings is typically short, while the other is arbitrarily long. This has a wide range of applications, for instance, spell checkers, correction systems for optical character recognition and software to assist natural language translation based on translation memory.

The Levenshtein Distance between two string a, b (of length |a| and |b| respectively) is given by lev (a, b) where,

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} \operatorname{lev}\big(\operatorname{tail}(a), b\big) \\ \operatorname{lev}\big(a, \operatorname{tail}(b)\big) \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) \end{cases} & \text{otherwise,} \end{cases}$$

Figure 4. 1: Levenshtein Distance Algorithm

Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Tools Used

Front End Tools Used:

- **React:** React is used in this project for creating the front end, that is the user interface. React is a JavaScript library for building user interfaces. It was developed by Facebook and is now widely used by developers for creating dynamic and interactive web applications. React is the backbone for creating the front-end part of this project. Every UI that is created is created with the help of React.
- **Material UI:** Material UI is used in this project for creating the attractive UI for this project. Material UI is a popular open source React component library that provides a set of customizable UI components based on Google's Material Design guidelines.
- **Redux:** Redux is used in this project in order to manage the state of an application. Redux allows predictable and consistent handling of state changes.
- **Styled Components:** Styled components is used in this project for writing the CSS-in-JS code to style the application.

Backend Tools Used:

- **Express.js:** Node.js is used as a backend framework for handling the server-side logic of the project. Express, is a backend web application framework for building RESTful APIs with Node.js. It is designed for building web applications and APIs.
- **Socket.IO:** Socket.IO is used in this project to establish communication between the users using WebSocket. Socket.IO is an event-driven library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers.

Database Technology Used:

- **MongoDB:** MongoDB is used as a database system in this project. MongoDB is a popular open-source, NoSQL database system that uses a document-oriented data model instead of the traditional table-based relational data model. It is designed to

store data as flexible, JSON-like documents that can have different fields and data types, making it ideal for storing unstructured or semi-structured data.

- **MongoDB Compass:** MongoDB Compass provides us with the facility of hosting the project on the laptop. Since the local machines are not built to serve as web servers MongoDB Compass provide us with the feature to host the project locally.

5.1.2 Implementation Details of Modules

Modules are the partitions of any project done to ease the task of development. Different modules are designed so that debugging and other development phases get the easiest implementation.

The different modules of the system are:

i) Home Page

The home page shows the already listed books, signup and sell books options. To upload any book information and list it in a site, user need to register into the site and sign in.

ii) Registration Page

The Registration Page allows users to register or sign in if they have already registered.

iii) Book Info Page

The Book Info Page allows users to view the detailed information of the book along with the option to Contact Seller. This page also shows the recommended books which they might like.

iv) Profile Page

The Profile Page allows users to edit their profile and book details. This page also lets users mark books as sold or delete the book from the system. This is like the control system for the user.

v) Wishlist Page

The Wishlist Page allows users to add their favorites' book into the Wishlist cart. This is like bookmarking the books so that user can easily find out its collection easily.

vi) Chat Page

The Chat Page allows users Communicate with each other so that they can discuss about the books listed by the seller.

5.2 Testing

5.2.1 Test Cases for Unit Testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. The following test scenarios were used to test the system after the build is completed.

Table 5. 1: Test Case for User Log In

TestId	Test Case	Input	Expected Output	Observed Output
1	Successful User Login	Email: test@gmail.com Password: testpassword2	Login successful and redirect to the homepage.	User successfully logged in and redirect to the homepage.
2	User Login Fail	Email: test@gmail.com Password: testpass	Login fail and show the error message to enter right credentials.	User was unable to logged in to the system.

Table 5. 2: Test Case for Adding Books

TestId	Test Case Description	Expected Output	Observed Output	Status
1	Fill all the book's information and click on the submit button.	Book should be added into the database.	Successfully added into the database.	Pass

2	Click on the submit button without filling all the fields.	Book shouldn't be added into the database and should shows error message to fill all the fields.	Unsuccessful to add book into the database.	Pass

Table 5. 3: Test Case for Algorithm Module

TestId	Test Case Description	Expected Output	Observed Output	Status
1	Add new book and check whether it is being recommended in the similar books section or not.	Book should be shown in the similar books section according to its distance order.	Book was shown in the similar books section in an appropriate position.	Pass

Table 5. 4: Test Case for Chat Module

TestId	Test Case Description	Expected Output	Observed Output	Status
1	Enter message into the field and click on the send button.	Receiver should be able to receive the message instantly.	Receiver was able to receive the message.	Pass

5.2.2 Test Cases for System Testing

System testing is defined as testing of a complete and fully integrated software product. System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. system testing falls within the scope of black box testing, and as such should require no knowledge of inner design of the code or logic. One of the types of system testing is the usability testing which is performed in the system.

Table 5. 5: Test Case for System Usability Testing

TestId	Test Case Description	Expected Output	Observed Output	Status
1	Click on various links on the system.	Link should take users to another web page according to the on-page URL.	Users were able to get the expected web pages after clicking the links.	Pass

5.3 Result Analysis

All the modules of the system are working as expected and the output of the system is up to expectations. The developed web application provides us with the feature of adding books into the system, contact seller and view description of the book along with the similar recommended books.

This application uses an efficient algorithm to find the optimal distance to recommend the books and uses socket programming to establish the connection between the buyer and seller.

Main focus is given on creating the user-friendly interface so that user can navigate throughout the system very easily. User can manage their listed items, edit their profile info through the profile section. From their home screen they have the option of viewing the book option, contacting the seller, and adding books into the wish list.

Overall, the web application is working according to expectations and without any problem.

CHAPTER 6: CONLUSION AND FUTURE RECOMMENDATION

6.1 Conclusion

The system is built using React, Express and MongoDB fully meets the objective of the system for which it has been developed. The system is operated at high level of efficiency and all the books' lovers and user associated with the system understands its advantage. Simple and elegant User Interface helps user to navigate easily throughout the application. The system solves the problem of not having the efficient platform for getting the secondhand books by directly connecting with the seller. It was intended to solve the problem as required specification.

6.2 Future Recommendation

PustakHub- a complete system for book lovers to connect and sell old books which has been developed has immense capability in today's digital with the increment of frequent book readers. But its capability can be further extended by integrating the payment system and providing the complete book hub platform where users can communicate each other's in a categorized book lobby.

This System can be further improved by taking into consideration of the following recommendations.

i. Integrating the payment gateway and home delivery system.
ii. Making the Book recommendation system more efficient.
iii. Making Web Application faster and efficient.

REFERENCES

[1] M. Martin, “Prototype model in software engineering,” Guru99, 13-Sep-2022. [Online]. Available: https://www.guru99.com/software-engineering-prototyping-model.html. [Accessed: 15-Dec-2022].

[2] K. Brush and V. Silverthorne, “What is Agile Software Development (agile methodologies)?,” Software Quality, 15-Nov-2022. [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development. [Accessed: 17-Jan-2023].

[3] “The Alibris story.” [Online]. Available: https://www.alibris.com/about/story. [Accessed: 09-Feb-2023].

[4] Bookchor, “Buy secondhand books, sell used books, Cheap Books,” Bookchor. [Online]. Available: https://www.bookchor.com/. [Accessed: 09-Feb-2023].

[5] S. Rani and J. Singh, “Enhancing Levenshtein's edit distance algorithm for evaluating document similarity,” SpringerLink, 01-Jan-1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-0755-3_6. [Accessed: 12-Feb-2023].

APPENDICES

SNAPSHOTS

1. Login Form

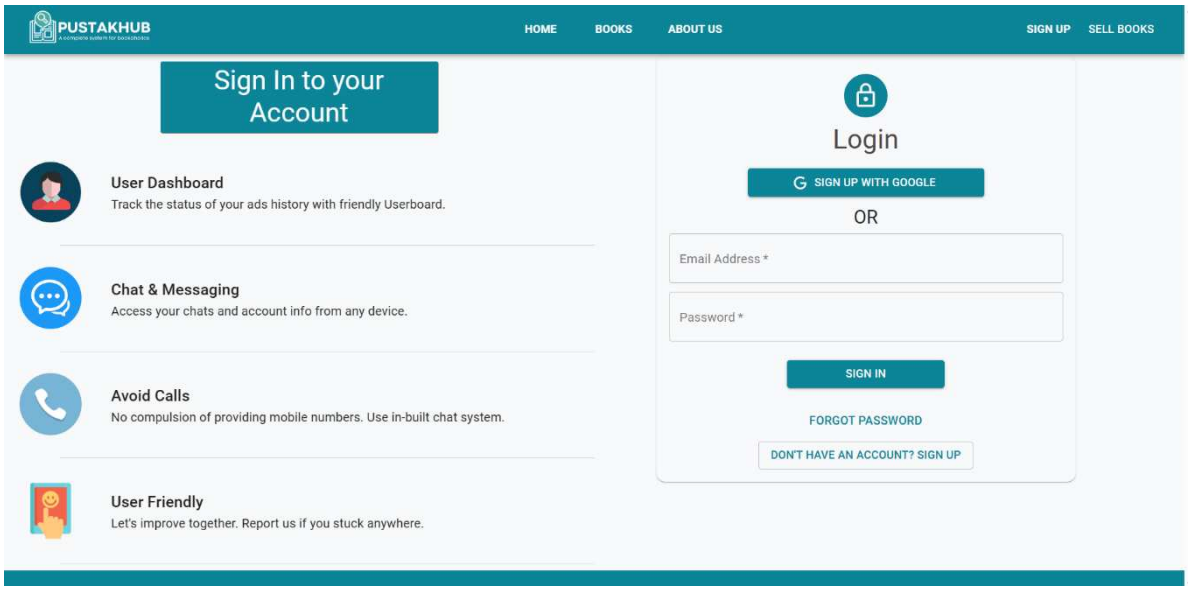

Figure 7: Login Form

2. All Books

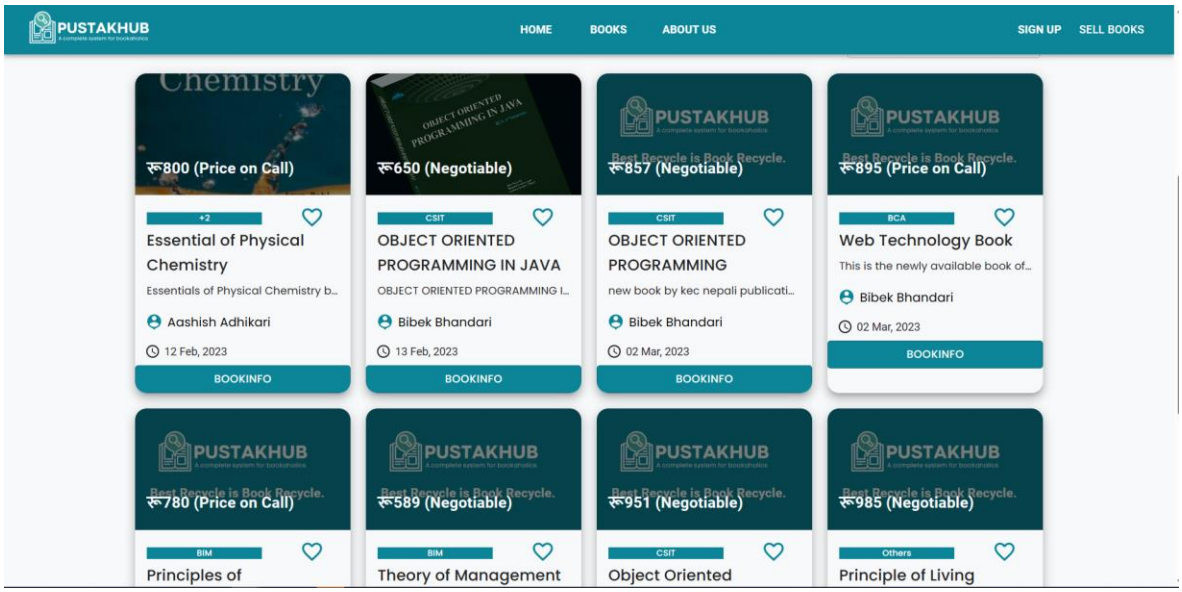

Figure 8: All Books

3. Book Information

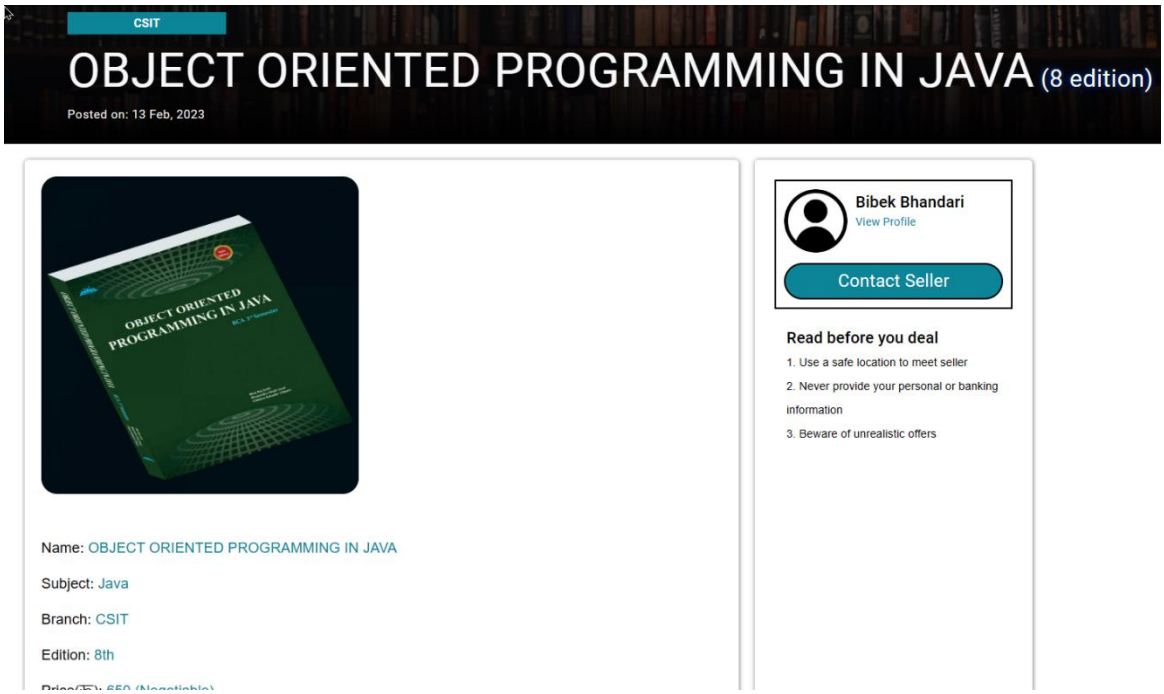

Figure 9: Book Information

4. Similar Books

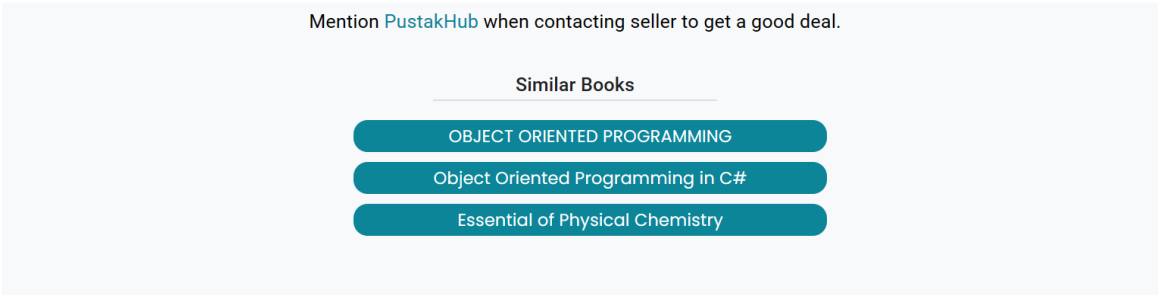

Figure 10: Similar Books

5. Message Section

Figure 11: Message Section

Source Code

1. Recommendation Algorithm

```
export function LevenshteinDistance(obj1, obj2) {
        // Convert objects to strings
        const a = JSON.stringify(obj1);
        const b = JSON.stringify(obj2);
        if (a.length === 0) return b.length;
        if (b.length === 0) return a.length;

        const matrix = [];

        // initialize the matrix
        for (let i = 0; i <= b.length; i++) {
                matrix[i] = [i];
        }
        for (let j = 0; j <= a.length; j++) {
                matrix[0][j] = j;
        }
        // calculate the distance
        for (let i = 1; i <= b.length; i++) {
                for (let j = 1; j <= a.length; j++) {
                        if (b.charAt(i - 1) === a.charAt(j - 1)) {
                                matrix[i][j] = matrix[i - 1][j - 1];
                        } else {
                                matrix[i][j] = Math.min(
                                        matrix[i - 1][j - 1] + 1, // substitution
                                        Math.min(
                                                matrix[i][j - 1] + 1, // insertion
                                                matrix[i - 1][j] + 1 // deletion
                                        )
                                );
                        }
                }
```

```
        }
        return matrix[b.length][a.length];
}
```

2. **Recommendation Client Side**

```
const BookInfo = ({ match }) => {
        const classes = useStyles();
        const history = useHistory();
        const dispatch = useDispatch();
        const books = useSelector((state) => state.books);
        const book = useSelector((state) => state.book);
        const bookId = match.params.bookId;
        const [found] = useState(books.find((bk) => bk._id === bookId) !==
undefined);

        const [contact_URL, setContact_URL] = useState('/auth');
        const localUser = JSON.parse(localStorage.getItem('profile'));
        const [bookNames, setBookNames] = useState([]);
        const [jsonData, setJsonData] = useState('');

        useEffect(() => {
                localUser ? setContact_URL(`/user/${book.owner}`) :
setContact_URL('/auth');
        }, [localUser, book.owner]);

        useEffect(() => {
                if (books.find((bk) => bk._id === bookId) !== undefined) {
                        dispatch({
                                type: GET_BOOK,
                                payload: books.find((bk) => bk._id === bookId),
                        });
                }
```

```
}, [bookId, books, dispatch]);
useEffect(() => {
        axios
                .get(`/books/${book._id}`)
                .then((response) => {
                        const jsonString = response.data;
                        const                stringData                =
JSON.parse(JSON.stringify(jsonString)).bookName;
                        setJsonData(stringData);
                })

                .catch((error) => {
                        console.log(error);
                });
}, [book._id, bookId, books._id]);

function fetchData() {
        axios
                .get('http://localhost:5000/books/getbooksname')

                .then((response) => setBookNames(response.data))

                .catch((error) => console.log(error));
}
useEffect(() => {
        fetchData();
}, []);

console.log(bookNames);

const bookkName = bookNames.map((results) => results.bookName);

const LevenshteinDistances = bookkName.map((book) => {
        const distance = LevenshteinDistance(jsonData, book);
```

```
        return { bookName: book, distance };
    });

    console.log(LevenshteinDistances);

    LevenshteinDistances.sort((a, b) => a.distance - b.distance);

    const similarBooks = LevenshteinDistances.slice(1, 4).map(
        (distance) => distance.bookName
    );
    console.log(typeof similarBooks);

    // console.log(book.bookName);
    //similar books filter
    const allBooks = useSelector((state) => state.books);
    const filterbooks = allBooks.filter(
        (books) =>
            books.isSold === false &&
            books.branch === book.branch &&
            books.owner !== book.owner
    );
```

3. **Socket.IO Implementation**

```
const io = require('socket.io')(server, options);

io.on('connection', async (socket) => {
    socket.on('disconnect', () => {});

    socket.on('landing_page', (data) => {});

    socket.on('login', (data) => {
        if (!socket.rooms.has(data.id)) {
            socket.join(data.id);
        } else {
```

```
        }
});

socket.on('logout', (data) => {
        socket.leave(data.id);
});

socket.on('join', async (data) => {
        var messages = await Message.find({
                $or: [
                        { from: data.id, to: data.receiver },
                        { from: data.receiver, to: data.id },
                ],
        });

        socket.emit('initial_msgs', messages);
});

socket.on('message', async (msg) => {
        try {
                const message = new Message({
                        from: msg.from,
                        to: msg.to,
                        content: msg.content,
                        fromName: msg.fromName,
                        sentAt: Date.now(),
                });
                await message.save();
                if (socket.adapter.rooms.has(msg.to)) {
                        await io.sockets.in(msg.from).emit('send_msg', {
                                content: message.content,
                                from: message.from,
                                to: message.to,
                                fromName: msg.fromName,
```

```
                        sentAt: message.sentAt,
                    });
                    await io.sockets.in(msg.to).emit('send_msg', {
                        content: message.content,
                        from: message.from,
                        to: message.to,
                        fromName: msg.fromName,
                        sentAt: message.sentAt,
                    });
                } else {
                    await io.sockets.in(msg.from).emit('send_msg', {
                        content: message.content,
                        from: message.from,
                        to: message.to,
                        fromName: msg.fromName,
                        sentAt: message.sentAt,
                    });
                    const receiver = await User.findById(message.to);
                    await sendChatMail(
                        receiver.email,
                        receiver.name,
                        message.fromName,

        `http://localhost:3000/user/${message.from}`
                    );
                }
            } catch (err) {}
        });
    });
```