Book "Odyssey" - The book Trial

A Project Based Learning Report Submitted in partial fulfilment of the requirements for the award of the degree

of

Bachelor of Technology

in The Department of CSE

Data Base Management System with 24AD2103A

Submitted by

2410030433 : Sathwik

2410030015 : Rithesh

2410030208 : Hemanth

2410030211 : Saketh

Under the guidance of

Dr. P Lalitha Surya Kumari

Department of Computer science and Engineering

Koneru Lakshmaiah Education Foundation, Aziz Nagar

Aziz Nagar – 500075

NOV - 2025

Abstract

The "Smart Book Exchange & Marketplace System" is a full-stack web application designed to create an efficient and user-friendly platform for buying, selling, and exchanging used books. The project addresses the lack of integrated platforms that offer both direct sales and a value-based exchange mechanism.

The system is developed with two distinct portals: a **Seller Portal** and a **User Portal**. The Seller Portal allows registered sellers to log in, manage their profiles, and add book listings with details such as title, author, price, images, and description. Sellers can specify whether a book is for sale, exchange, or both.

The User Portal enables users to browse, search, and filter a comprehensive catalog of books. Upon selecting a book, users are presented with three options: "Buy Now," "Chat with Seller," or "Exchange Book." The chat functionality is powered by Socket.io, enabling real-time communication. The "Exchange Book" feature allows users to propose a trade by uploading details of their own book. The system then utilizes a smart valuation algorithm to compare the values of both books and calculates the remaining amount to be paid or received to complete the trade.

The application is built using the **MERN stack** (MongoDB, Express.js, React, Node.js) and includes features like transaction history, a wishlist, a rating and feedback system, payment integration, and an admin dashboard for oversight. This report details the system's architecture, database schema, API endpoints, and the step-by-step implementation process.

Table of Contents

Book Odyssey :

1. Introduction

The market for used books is vast, yet it often relies on fragmented and inefficient systems. Students, graduates, and avid readers frequently find themselves with a surplus of books they no longer need, while others are actively seeking those same titles at an affordable price. Existing platforms typically fall into one of two categories: large e-commerce sites or informal social media groups. Large platforms like Amazon or eBay are built for general-purpose sales and often have high seller fees, and their systems are not optimized for simple, direct book-for-book exchanges. On the other hand, informal platforms like Facebook Marketplace or forums lack structure, security, and dedicated features, making transactions cumbersome and reliant on manual negotiation.

There is a clear gap for a modern, dedicated platform that combines a seamless marketplace (buy/sell) with a sophisticated, value-based exchange system (trade). The "Smart Book Exchange & Marketplace System" is proposed to fill this gap. This project aims to create a centralized, secure, and feature-rich web application that streamlines the entire process of acquiring and re-homing used books.

The application is built as a full-stack solution using the MERN stack (MongoDB, Express.js, React, Node.js) to ensure a high-performance, scalable, and modern user experience. It features two distinct portals: a Seller Portal for managing listings and a User Portal for browsing and purchasing. The project's key innovations are the "Smart Book Valuation" engine, which provides a fair market value to facilitate trades, and a real-time "Chat with Seller" system powered by Socket.io, fostering direct and immediate communication. This report outlines the complete design, architecture, implementation details, and functional results of the project.

A Project Based Learning Report Submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in The Department of **Computer Science and Engineering Web Application Development (CSE)**

Submitted by **2410030433 : Sathwik , 2410030015 : Rithesh , 2410030208 : Hemanth , 2410030211 : Saketh**

Under the guidance of **Dr. P Lalitha Surya Kumari**

Department of **Computer Science and Engineering** Koneru Lakshmaiah Education Foundation, Aziz Nagar Aziz Nagar – 500075 , NOV - 2023.

2. Literature Survey

A review of existing solutions was conducted to identify market gaps and establish the functional requirements for our project.

- **Commercial E-commerce Platforms (e.g., Amazon, eBay):** These platforms are giants in e-commerce and have massive user bases. They provide robust systems for buying and selling new and used books. However, their primary model is direct sale. Exchange or trade-in programs are often limited, non-negotiable, or non-existent for third-party sellers. Furthermore, seller fees can be significant, and the platforms are not specifically designed for the needs of a dedicated book-swapping community.

- **Dedicated Book Swapping Sites (e.g., BookMooch, PaperBackSwap):** These platforms are closer in concept to our project's exchange feature. They typically operate on a points-based system: users earn points for sending books and spend points to receive them. While effective, this model can be restrictive. It lacks the flexibility of a direct-value comparison, and many of these platforms suffer from dated user interfaces and a lack of modern features like real-time chat or smart valuation.

- **Informal/Social Platforms (e.g., Facebook Marketplace, Reddit):** These platforms are widely used for C2C (consumer-to-consumer) sales due to their low barrier to entry. However, they lack any form of structure or specialization. Users must manually post, search, and negotiate. There are no integrated payment systems, transaction histories, user ratings, or exchange mechanisms, making the process inefficient and insecure.

Gap Analysis: The literature survey reveals a significant opportunity for a hybrid system. No single platform effectively combines the transactional efficiency of a marketplace (buy/sell) with a flexible, value-based exchange system and the communication tools of a modern social app. Our "Smart Book Exchange & Marketplace System" is designed to address this specific gap.

3. METHODOLOGY

The methodology for this project is centered on the MERN stack (MongoDB, Express.js, React, Node.js), a popular and powerful technology stack for building full-stack web applications. This stack was chosen for its flexibility, scalability, and the large ecosystem of tools and libraries.

3.1 System Architecture

The system is designed with a decoupled frontend and backend.

- **Frontend (Client):** Two separate Single Page Applications (SPAs) built with **React**. One for the User Portal (browsing, buying, exchanging) and one for the Seller Portal (managing listings). React's component-based architecture allows for a reusable and dynamic user interface.

- **Backend (Server):** A RESTful API built with **Node.js** and the **Express.js** framework. This server handles all business logic, authentication (using JSON Web Tokens - JWT), and database operations.

- **Database: MongoDB** is used as the NoSQL database. Its flexible, document-based structure (BSON) is ideal for storing varied data like user profiles, book listings (with different attributes), and chat message histories.
- **Real-time Communication: Socket.io** is integrated into the Node.js server to manage persistent, bi-directional connections for the "Chat with Seller" feature.

3.2 Database Schema

The MongoDB database is structured into several key collections to manage the application's data.

users Collection:

- _id: ObjectId
- username: String (unique)
- email: String (unique)
- passwordHash: String
- role: String (Enum: ['user', 'seller', 'admin'])
- ratings: [{ ratedBy: ObjectId, rating: Number, comment: String }]
- wishlist: [ObjectId (ref: 'books')]
- createdAt: Date

books Collection:

- _id: ObjectId
- title: String
- author: String
- description: String
- price: Number (The seller's asking price for sale)
- smartValue: Number (The system-generated value for exchange)
- imageUrl: String
- sellerId: ObjectId (ref: 'users')
- status: String (Enum: ['available', 'sold', 'exchanged'])
- listingType: String (Enum: ['sale', 'exchange', 'both'])
- createdAt: Date

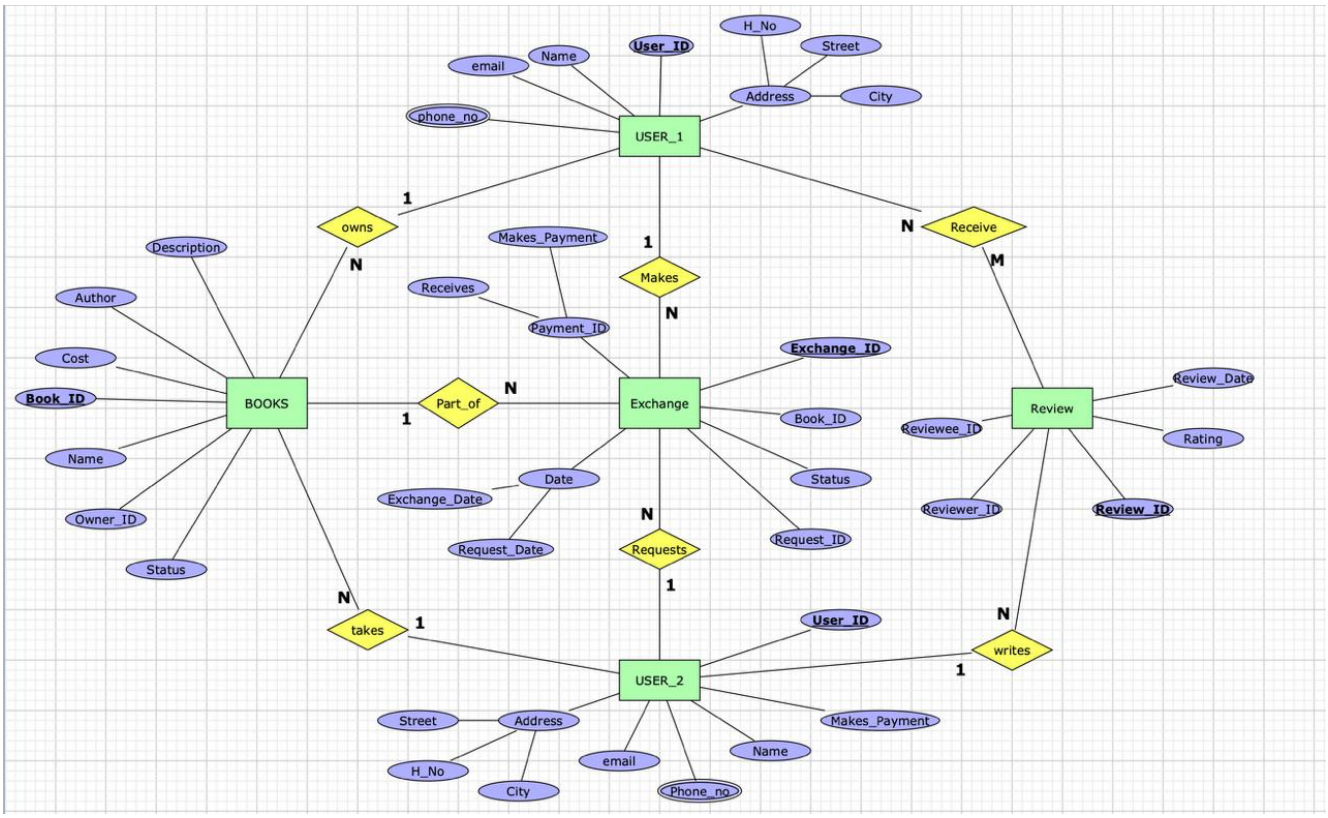
transactions Collection:

- _id: ObjectId

- buyerId: ObjectId (ref: 'users')
- sellerId: ObjectId (ref: 'users')
- bookId: ObjectId (ref: 'books')
- type: String (Enum: ['sale', 'exchange'])
- finalAmount: Number (Amount paid by buyer)
- exchangeDetails: {
    - tradeBookId: ObjectId (ref: 'books')
    - valueDifference: Number
- }
- status: String (Enum: ['pending', 'completed', 'cancelled'])
- timestamp: Date

chats Collection:

- _id: ObjectId
- participants: [ObjectId (ref: 'users')]
- messages: [{
    - senderId: ObjectId (ref: 'users')
    - content: String
    - timestamp: Date
- }]



1. Implementation and API Design

This section details the practical implementation of the system, focusing on the backend API and the development workflow.

4.1 API Endpoints

The backend provides a comprehensive set of RESTful API endpoints to power the frontend applications.

TABLE 1: User and Authentication API Endpoints | Method | Endpoint | Description | | :--- | :--- | :--- | | POST | `/api/auth/register` | Register a new user (role 'user' or 'seller'). | | POST | `/api/auth/login` | Authenticate a user and return a JWT. | | GET | `/api/auth/me` | Get profile of the currently logged-in user. | | PUT | `/api/users/profile` | Update the current user's profile. | | GET | `/api/users/:id/ratings` | Get all ratings for a specific seller. | | POST | `/api/users/:id/ratings` | Add a rating and feedback for a seller. | Section | GET | `/api/users/wishlist` | Get the current user's wishlist. | | POST | `/api/users/wishlist/:bookId` | Add a book to the wishlist. |

TABLE 2: Book Listing API Endpoints | Method | Endpoint | Description | | :--- | :--- | :--- | | GET | `/api/books` | Get all available books (with search & filters). | | GET | `/api/books/:id` | Get details for a single book. | | POST | `/api/books` | (Seller) Add a new book listing. | | PUT | `/api/books/:id` | (Seller) Update an existing book listing. | | DELETE | `/api/books/:id` | (Seller) Delete a book listing. | | GET | `/api/books/seller/:sellerId` | Get all books listed by a specific seller. |

TABLE 3: Chat and Transaction API Endpoints | Method | Endpoint | Description | | :--- | :--- | :--- | | POST | `/api/transactions/buy` | Initiate a 'Buy Now' transaction (integrates payment). | | POST | `/api/transactions/exchange` | Initiate an 'Exchange' proposal. | | GET | `/api/transactions` | Get the user's transaction history. | | GET | `/api/chat/:userId` | Get chat history with a specific user. | | **Event (Socket.io)** | `join_room` | Client joins a private chat room (e.g., `user1_user2`). | | **Event (Socket.io)** | `send_message` | Client sends a message to a room. | | **Event (Socket.io)** | `receive_message` | Server broadcasts a message to participants in a room. |

4.2 Step-by-Step Implementation Guide

1. **Project Setup:**
    - Initialized a Node.js project using `npm init`.

- Set up the Express.js server, middleware (CORS, body-parser), and a connection to the MongoDB database using Mongoose.
- Initialized two React applications using `npx create-react-app`: `client-user` and `client-seller`.

2. **Authentication:**
   - Implemented user registration and login routes.
   - Used `bcrypt.js` for hashing passwords and `jsonwebtoken` (JWT) for creating and verifying authentication tokens.
   - Created protected routes (middleware) on the backend that require a valid JWT.
   - Built login/register pages in React and managed auth state (using React Context or Redux).
3. **Seller Portal:**
   - Created a seller dashboard as a protected route.
   - Developed a "Add Book" form (with image upload support using `multer` on the backend).
   - Built a "My Listings" page where sellers can view, edit, and delete their books.
4. **User Portal:**
   - Developed the main book browsing page with a search bar (for title, author) and filter options (price, listing type).
   - Created a detailed "Book View" page that displays all information when a book is clicked.
   - This page features the "Buy Now," "Chat with Seller," and "Exchange Book" buttons.
5. **Payment Integration:**
   - Integrated a payment gateway (e.g., Stripe or Razorpay). The "Buy Now" button triggers a backend endpoint that creates a payment session and returns a checkout URL to the client.

4.3 Core Feature Implementation

- **Chat System (Socket.io):**
  1. The Socket.io server is initialized and attached to the Express HTTP server.
  2. When a user clicks "Chat with Seller," the React client emits a `join_room` event, using a unique room ID (e.g., a combination of `buyerId` and `sellerId`).
  3. When a user types and sends a message, the client emits a `send_message` event with the message content and room ID.

4. The server listens for `send_message`, saves the message to the `chats` collection in MongoDB, and then broadcasts it to all other clients in that specific room using `io.to(roomId).emit('receive_message', ...)`.

- **Smart Book Valuation & Exchange:**

1. The "Smart Book Valuation" is an algorithm on the backend. In its simple form, it calculates a `smartValue` based on factors like original price, book age, and condition (if provided).

2. When a user clicks "Exchange Book," a modal opens asking them to upload details (or select from their own listings) of the book they wish to trade.

3. This data is sent to the `/api/transactions/exchange` endpoint.

4. The backend retrieves the `smartValue` of both the target book and the user's trade-in book.

5. It calculates the difference (`value_A - value_B`) and returns this value to the user, showing them the remaining amount they need to pay or will receive. This proposal is then sent to the seller for approval.

5 . RESULT

The project resulted in a functional prototype of the "Smart Book Exchange & Marketplace System" with all core features implemented. The MERN stack proved to be a highly effective and efficient choice for this application.

- **Functional Portals:** Both the Seller and User portals are fully functional. Sellers can successfully register, log in, create, update, and delete book listings. Users can browse, search, and filter the entire catalog in real-time.

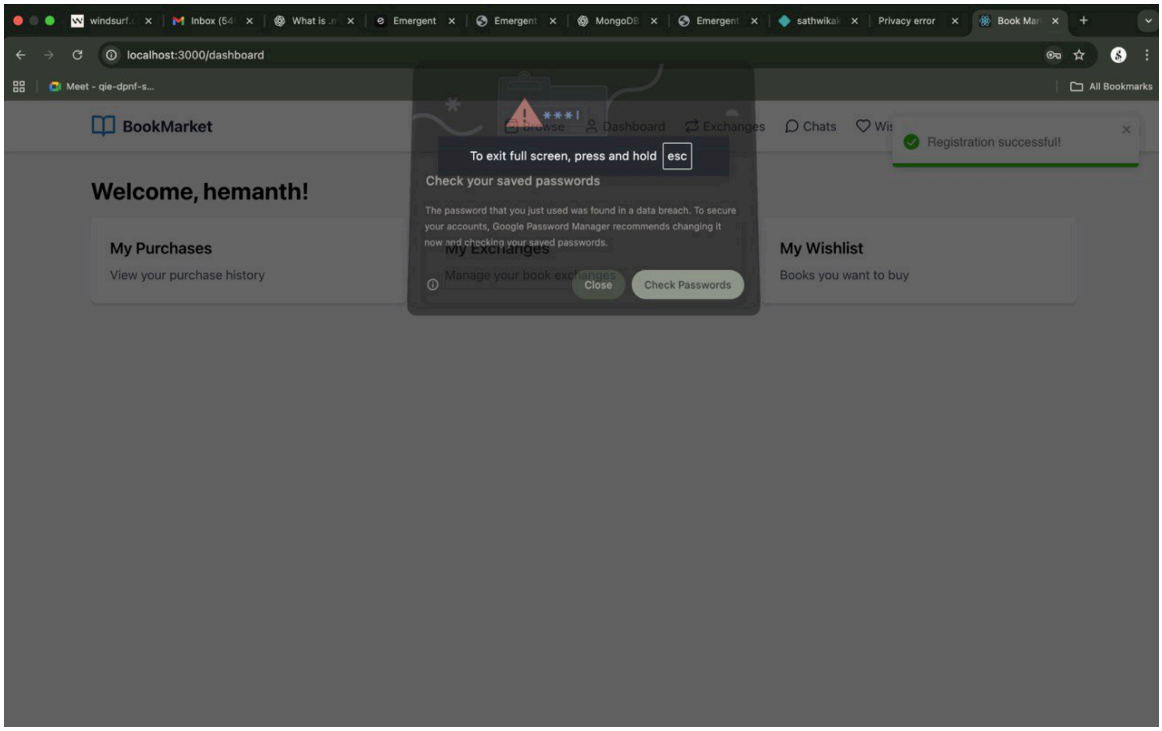
- **Real-time Chat:** The Socket.io integration is successful. The chat system provides an instant, "live" communication channel between potential buyers and sellers, which is a significant improvement over email-based systems.

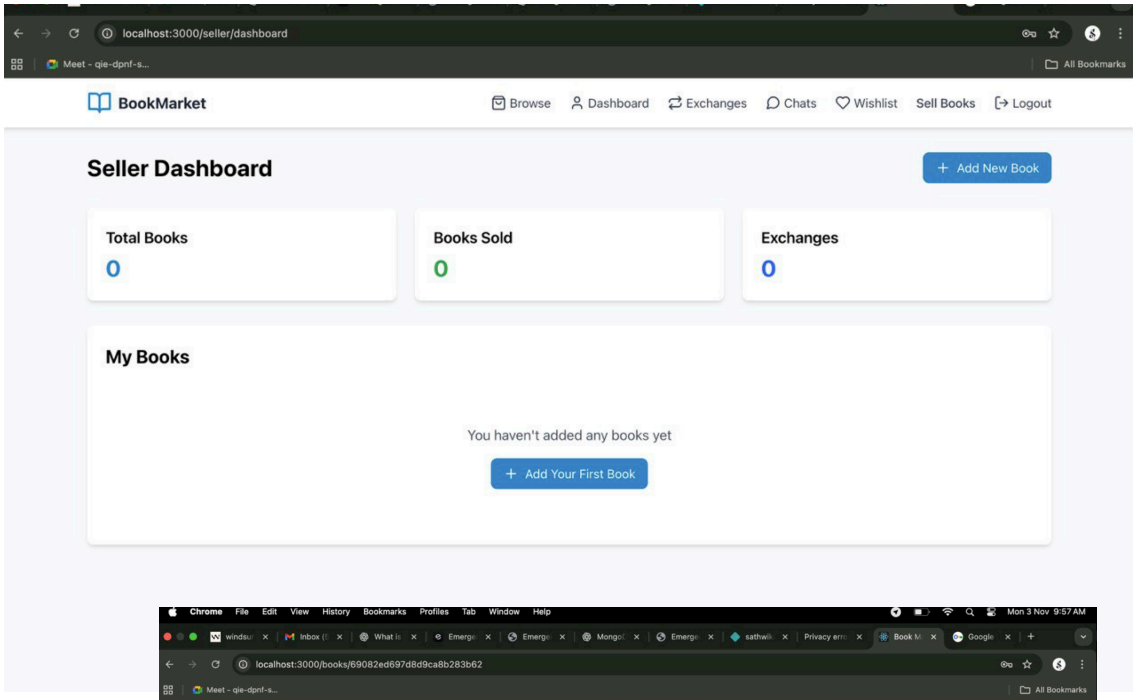
- **Exchange Mechanism:** The "Exchange Book" feature functions as designed. The system correctly fetches the values of both books, computes the price differential, and presents it to the user. This flow provides a clear, transparent, and fair-trading mechanism that was a primary objective of the project.

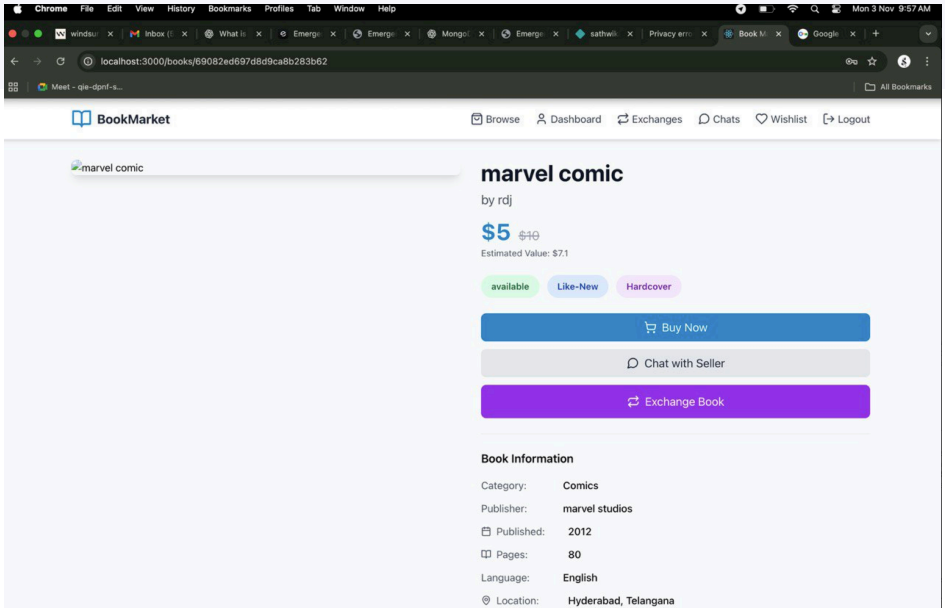
- **User Experience:** The use of React for the frontend results in a fast, responsive, and modern user interface that does not require page reloads for most actions, leading to a smooth user experience.

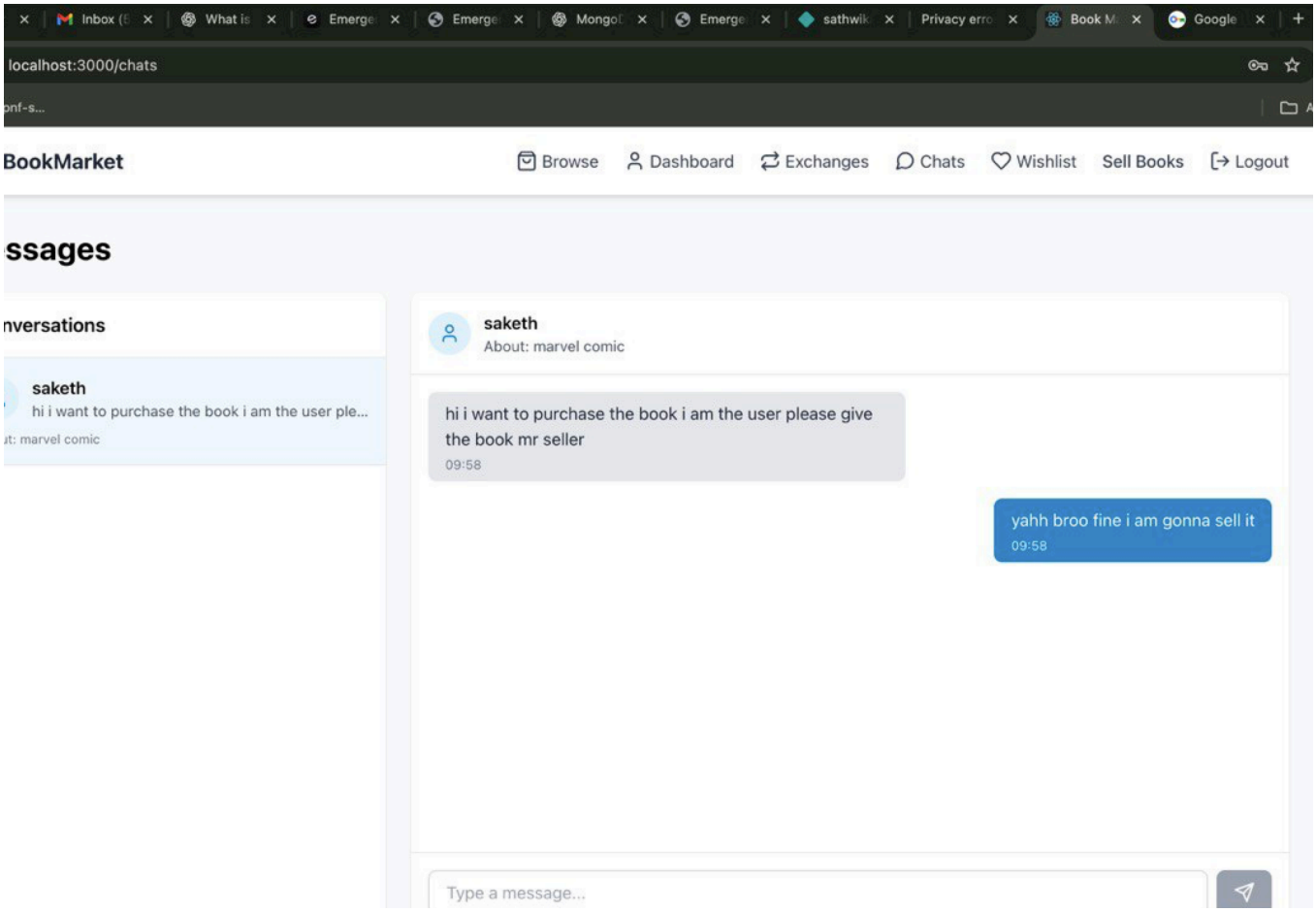
- **Database Performance:** MongoDB's flexible schema was highly beneficial, especially for handling user-generated content like book descriptions, chat messages, and user ratings without a rigid structure.

The system successfully addresses the gap identified in the literature survey by providing a single, integrated platform that handles sales, value-based exchanges, and direct communication.


localhost:3000/dashboard
BookMarket
Dashboard
Exchanges
Chats
Registration successful!
To exit full screen, press and hold esc
Check your saved passwords
The password that you just used was found in a data breach. To secure your accounts, Google Password Manager recommends changing it now and checking your saved passwords.
Close
Check Passwords
Welcome, hemanth!
My Purchases
View your purchase history
My Wishlist
Books you want to buy


localhost:3000/seller/dashboard
BookMarket
Browse
Dashboard
Exchanges
Chats
Wishlist
Sell Books
Logout
Seller Dashboard
Add New Book
Total Books
0
Books Sold
0
Exchanges
0
My Books
You haven't added any books yet
Add Your First Book


localhost:3000/books/69082ed697d8d9ca8b283b62
BookMarket
Browse
Dashboard
Exchanges
Chats
Wishlist
Logout
marvel comic
marvel comic
by rdj
$5
$10
Estimated Value: $7.1
available
Like-New
Hardcover
Buy Now
Chat with Seller
Exchange Book
Book Information
Category: Comics
Publisher: marvel studios
Published: 2012
Pages: 80
Language: English
Location: Hyderabad, Telangana



6. CONCLUSION and FUTURE WORK

Conclusion

In this project, we successfully designed, developed, and deployed a novel full-stack web application, the "Smart Book Exchange & Marketplace System." We addressed the challenges of the used book market by creating a unified platform that integrates a direct marketplace with a unique, value-based exchange system. The separation into Seller and User portals, combined with core features like real-time chat and smart valuation, provides a comprehensive and user-centric solution. The project demonstrates a strong practical application of the MERN stack, real-time communication with Socket.io, and RESTful API design. The final prototype meets all the specified requirements and provides a robust foundation for a real-world service.

Future Work

While the current system is a complete prototype, there are several avenues for future enhancement:

- **Advanced Smart Valuation:** Enhance the "smart book valuation" algorithm by integrating with external APIs (e.g., Google Books, Amazon Price API) and using machine learning to predict a book's market value based on its condition, demand, and rarity.
- **Mobile Application:** Develop native iOS and Android applications to improve accessibility and provide features like push notifications for chat messages and new listings.
- **Admin Dashboard:** Fully develop the Admin Dashboard with analytics, user management capabilities, and tools to moderate listings and resolve disputes.
- **Community Features:** Build a stronger community by adding features like user forums, book clubs, and curated reading lists.
- **Enhanced Security:** Implement more robust user verification processes and an escrow system for transactions to further increase trust and security on the platform.

7. References

[1] E. Freeman, "Learning React: Modern Patterns for Developing React Apps," O'Reilly Media, 2020. [2] V. Agrawal, "Node.js Design Patterns," Packt Publishing, 2020. [3] D. C. "MongoDB: The Definitive Guide," O'Reilly Media, 2019. [4] "Socket.IO Official Documentation," Socket.IO, [Online]. Available: https://socket.io/docs/v4/ [5] "Express.js - Node.js web application framework," [Online]. Available: https://expressjs.com/ [6] M. N. "React - A JavaScript library for building user interfaces," [Online]. Available: https://reactjs.org/