Bookstore Inventory Using MERN

Project Report for theApproval of Major Project (MCAN-482)

In

**Master of Computer Application**

Ananda Ghosh (14871023031)

Neha Mistry (14871023043)

Rupanjana Koner (14871023014)

Shubhajit Chakraborty(14871023021)

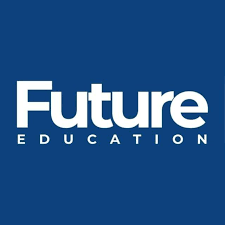
COMPUTER APPLICATION DEPARTMENT

**FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT**

**(Affiliated to Maulana Abul Kalam AzadUniversity of Technology)**

**Kolkata - 700150**

**2024-25**

****



FUTUREINSTITUTEOFENGINEERINGANDMANAGEMENT

SonarpurStationRoad,Kolkata–700150

MCA- DEPARTMENT

**CERTIFICATE FOR PROJECT WORK**

We do here by certify that the work which is being presented in the Major Project Report entitled **“**Bookstore Inventory**”,** inpartial fulfillmentof the requirementsfor theaward of Master of ComputerApplicationssubmitted to the MCA-Dept. of**Future Institute of Engineering and Management, Kolkata,WB**is an authentic record of our own work carried out during the period from **9.12.2024** to **20.5.2025**under the supervision of Sourav Kundu Sir.

The matter presented in this these is has not been submitted by us for the award of any other degree elsewhere.

***Name Signature of the Candidate(s)***

Ananda Ghosh

Neha Mistry

Rupanjana Koner

Shubhajit Chakraborty

Thisistocertifythattheabovestatementmadebythestudents,iscorrecttothebestofmyknowledge.

**Date:**

*Signatureofthe Supervisor*

**Mr. Sourav Kundu**

|  |  |
| --- | --- |
| **Head CA-Dept.**  **FutureInstitute ofEngineeringand**  **ManagementKolkata,WB** | **Signature of the external examiner & Panel Members** |

**ACKNOWLEDGEMENT**

We take this opportunity to express our deep gratitude and sincerest thanks to our Project mentor Sourav Kundu Sir, and HOD sir for giving us most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

We would like to give a special recognition to the colleagues, last but not the least we are grateful to all the faculty members of our department for their support.

**Table of content**

**TITLE PAGE NO.**

**COVERPAGE**

**CERTIFICATE FROM SUPERVISOR**

**& PANEL MEMBERS**

**ACKNOWLEDGEMENT**

**ABSTRACT**

**INTRODUCTION**

**METHODOLOGY**

**CODING**

**FUTURE SCOPE**

**CONCLUSION**

**REFERENCES**

### **ABSTRACT**

ThinkBound is a full-stack web application created to help people easily buy and sell books online. The goal of this project is to provide a simple, user-friendly platform where book lovers can explore a wide range of books and sellers can list their products for sale. The website is built using the MERN stack, which includes MongoDB for the database, Express.js and Node.js for the backend, and React for the frontend. Tailwind CSS is used to style the pages and make the design clean and responsive.

Users can create accounts, browse available books, search for specific titles, add items to a shopping cart, and place orders. The backend services are hosted on Render, while the frontend is deployed using Netlify for smooth and fast performance. By using cloud technologies and modern tools, ThinkBound ensures a reliable and scalable experience for users. This project shows how powerful and flexible full-stack development can be, even for small and medium-sized e-commerce websites. It also provides a strong base for adding more features in the future, such as online payments, book recommendations, and mobile app support.

### **INTRODUCTION**

The shift towards digital platforms has made online shopping more convenient and widely used, even for purchasing books. To meet this growing demand, this project introduces **ThinkBound**, a full-stack web application that allows users to explore and buy books online. The website is designed to be simple, responsive, and easy to use for both customers and administrators.

The application is built using the **MERN stack**, which includes **MongoDB** for storing book and user data, **Express.js** and **Node.js** for backend development, and **React.js** for the frontend interface. For styling, **Tailwind CSS** was used to create a clean, mobile-friendly layout. The frontend of the application is deployed using **Netlify**, while the backend runs on **Render**, ensuring the site is available online and functions smoothly.

Users can browse books, search by title or category, and manage their cart. Admin users can add, update, or delete books from the inventory. This project not only showcases how modern web technologies can solve real-world problems but also demonstrates the practical application of full-stack development in building a live, hosted web solution.

**RELEVANCE OF THE TOPIC**

With the rapid growth of online shopping, even traditional businesses like bookstores are moving towards digital platforms. Customers now expect the ability to browse, search, and purchase books online with ease. However, many small bookstores still lack an efficient system to manage and sell their inventory digitally.

ThinkBound addresses this need by providing a modern, responsive, and easy-to-manage online book selling platform. It helps streamline book management, improve user experience, and make the business accessible to a wider audience. The project is relevant in today’s context as it supports digital transformation in retail and promotes smarter inventory handling through technology.

**PROBLEM DEFINITION**

* Manual inventory management is time-consuming and prone to errors.
* No real-time tracking of available or out-of-stock books.
* Difficult to update book details like price, category, or description.
* Customers cannot browse or buy books online.
* No central system to organise, search, or filter the book collection.
* Generating reports or tracking sales is not possible with basic methods.

ThinkBound addresses these issues by offering a simple, web-based platform for managing and selling books online.

**PROJECT OBJECTIVE**

The main objective of this project is to develop a fully functional online book selling platform that simplifies the process of managing and purchasing books. The system is designed to offer a smooth experience for both users and administrators through a clean interface and efficient backend operations.

Key objectives include:

* To build a responsive and user-friendly web application using the MERN stack.
* To allow users to browse, search, and view books online.
* To implement cart functionality for selecting and reviewing books before purchase.
* To enable admin users to add, update, or delete books from the inventory.
* To store and manage data using a secureMongoDB database.
* To deploy the application using modern hosting platforms (Netlify for frontend and Render for backend).
* To ensure real-time updates and smooth communication between frontend and backend through RESTful APIs.

**METHODOLOGY**

**Requirement Analysis**  
The first step involved identifying the core features needed for an online book selling platform. These included user registration and login, book listing, categorisation, a shopping cart system, order placement, and an admin dashboard for managing books.

**Frontend Development**  
The user interface was created using React.js. Components were developed for the homepage, login/signup forms, book listings, cart, and admin panel. Tailwind CSS was used to design a clean and responsive layout compatible with both desktop and mobile devices.

**Backend Development**  
Node.js and Express.js were used to build the backend. RESTful APIs were developed to handle operations such as user authentication, book management, cart updates, and order processing. The backend ensured smooth communication between the frontend and the database.

**Database Integration**  
MongoDB was used as the database to store user data, book details, cart items, and orders. Mongoose was used to define schemas and interact with the database efficiently and securely.

**Feature Implementation**  
The system included essential features such as user authentication (sign up and login), book browsing and categorisation, cart functionality with the ability to place orders, and an admin panel to manage books and view order details.

**API Integration**  
Axios was used on the frontend to send requests to the backend APIs. This enabled real-time operations such as user login, fetching book data, updating cart items, and submitting orders.

**Testing**  
Manual testing was carried out throughout the development process. Each component and feature was tested in the browser, and API endpoints were tested using Postman to ensure correct data flow and error handling.

**Deployment**  
The frontend was deployed on Netlify, allowing public access and continuous deployment. The backend server was hosted on Render. Environment variables were used to manage configuration and secure API endpoints during production.

The features implemented in the ThinkBound application include:

* **User Authentication**: Users can sign up and log in to access personalised features.
* **Book Listing and Categorisation**: Books are displayed with categories for easier browsing.
* **Cart System and Order Placement**: Users can add books to the cart and place orders.
* **Admin Panel**: Admin users can add, update, and delete books, as well as view order-related information.

**SYSTEM ANALYSIS**

**SOFTWARE REQUIREMENTS**

## Frontend:

* React.js
* Tailwind CSS
* Axios (for API requests)

## Backend:

* Node.js
* Express.js
* MongoDB (via Mongoose)
* JSON Web Token (JWT) for authentication

## Hosting & Deployment:

* Frontend: Netlify
* Backend: Render
* Database: MongoDB Atlas (cloud-hosted MongoDB)

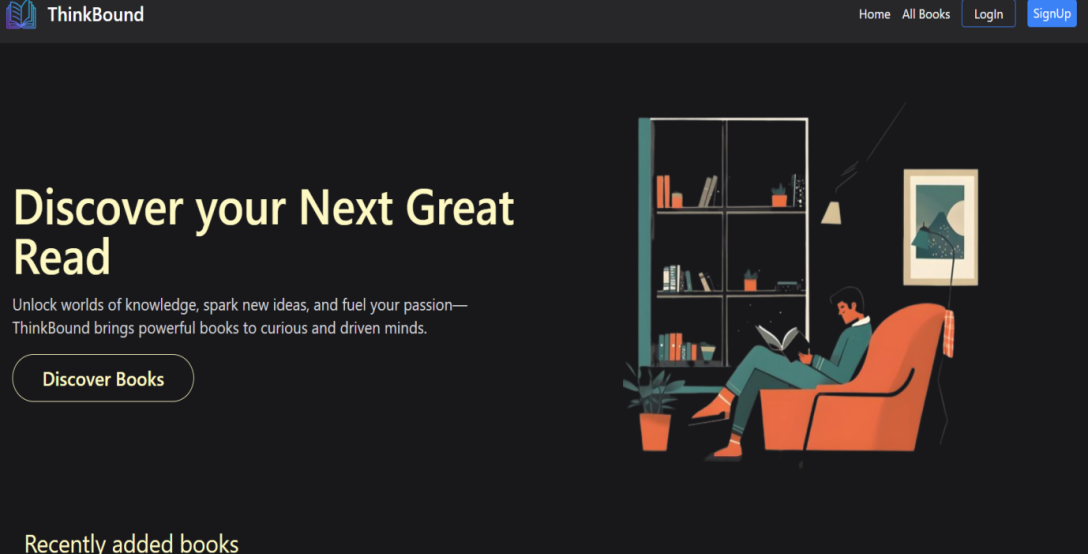
**HARDWARE REQUIREMENTS**

* **Processor**: Intel Core i5 or higher (64-bit, quad-core, 2.0 GHz or above)
* **RAM**: Minimum 8 GB
* **Storage**: At least 256 GB SSD (for faster read/write during development)
* **Display**: HD display (1366×768 or higher resolution)
* **Network**: Stable internet connection for accessing cloud platforms and deployment

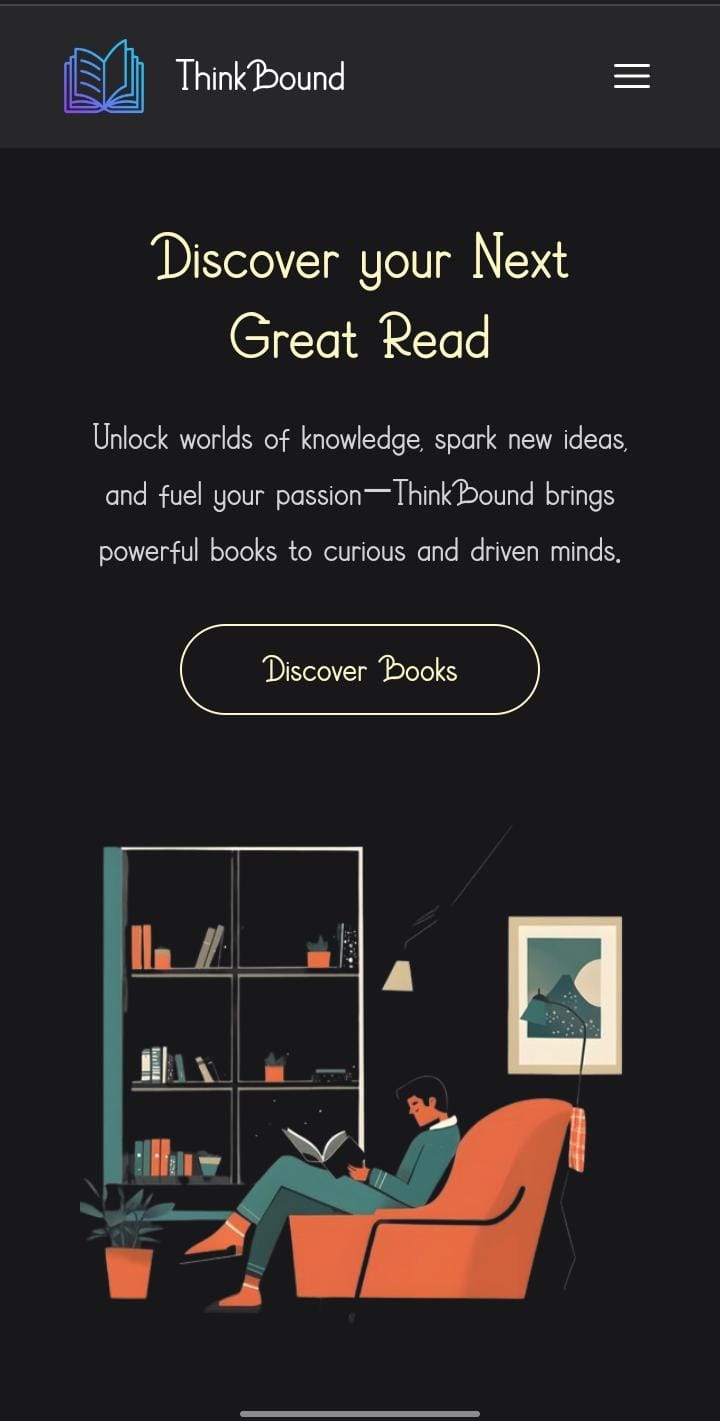
**SCREENSHOT SECTION**

**Home page:**

**i)Desktop view**

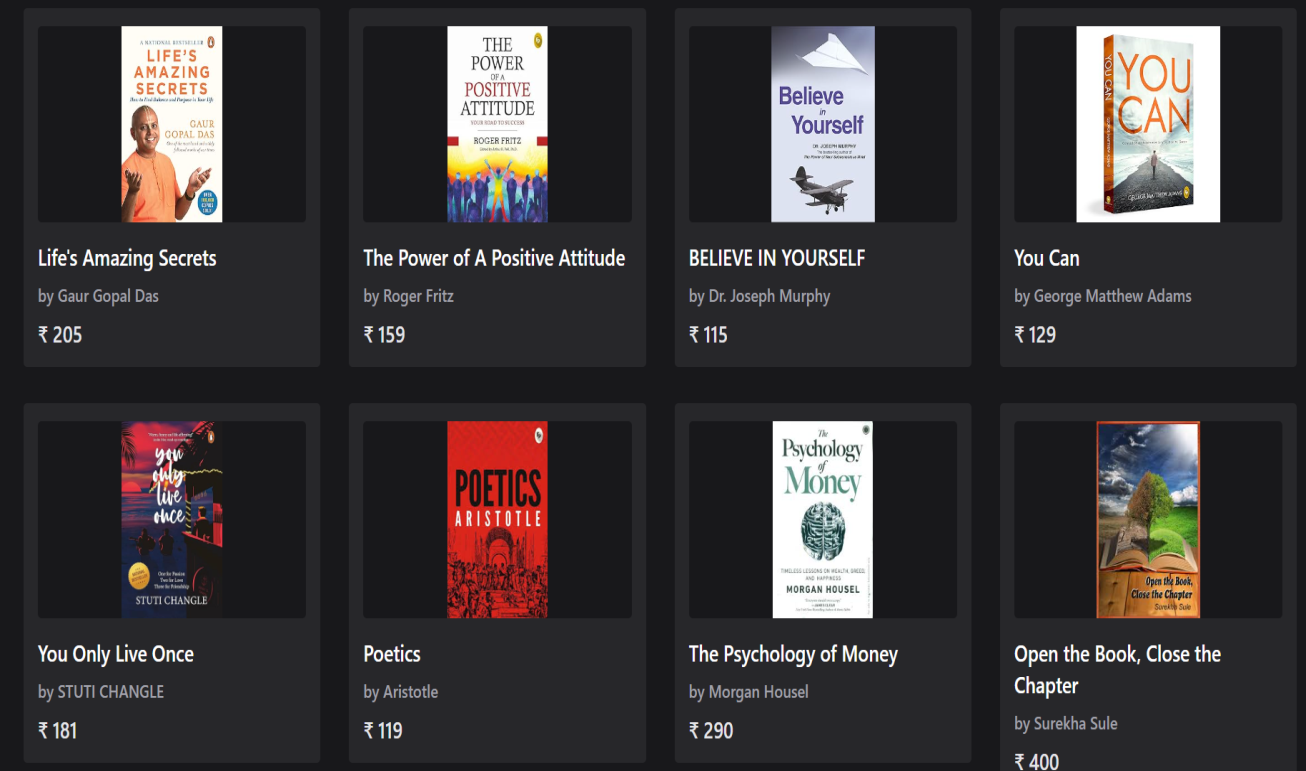
****

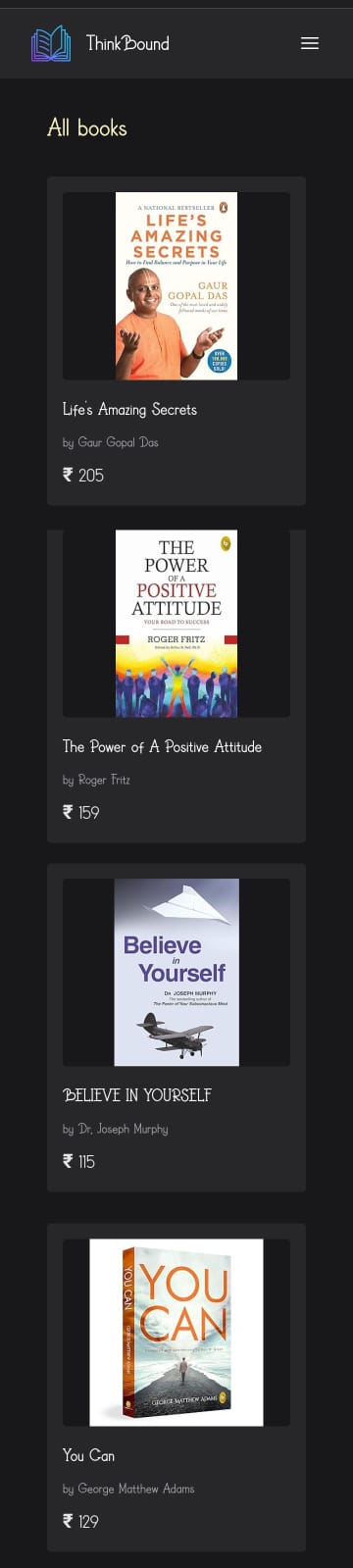
**ii)Mobileview**



**All Books**

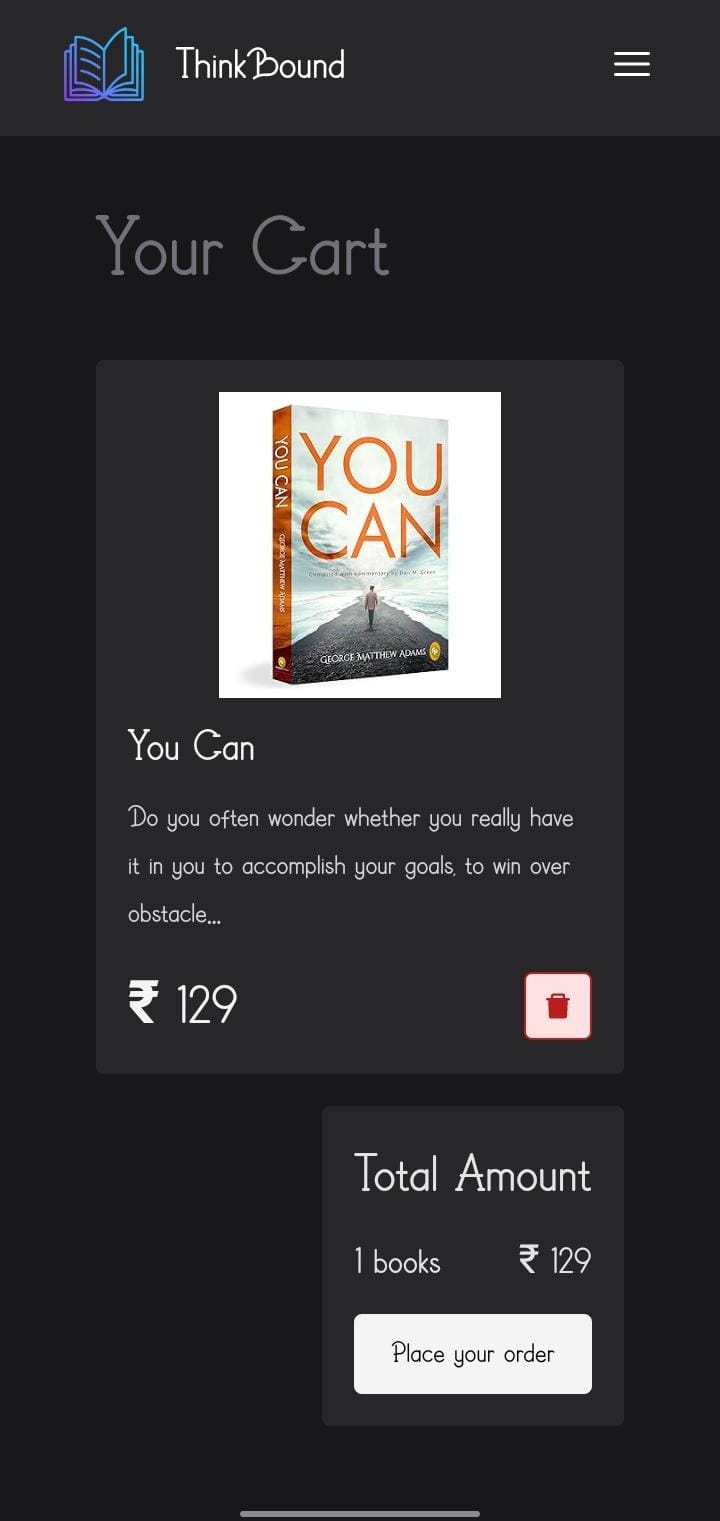
**Desktop view:**

**Mobile View:**

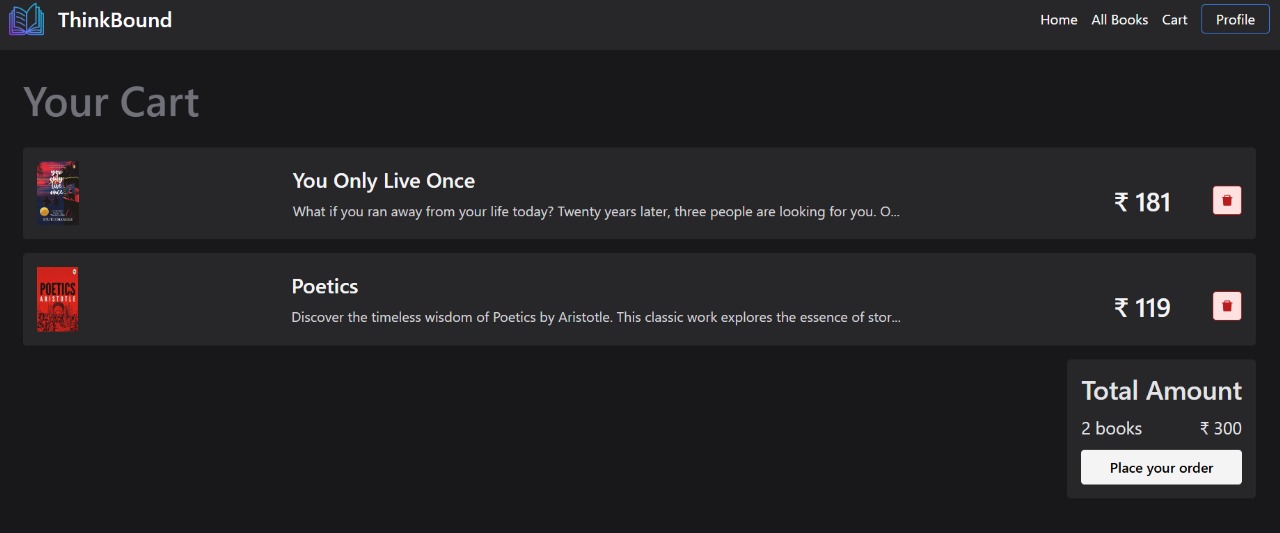


**Cart Value:**

**Mobile view:**

****

**Desktop view:**



**CODING**

**FEATURES OF LANGUAGE**

**JavaScript**: Lightweight, interpreted, and supports both client-side and server-side development. It enables dynamic content, asynchronous operations, and cross-platform compatibility.

**React.js**: A frontend library used for building user interfaces. It offers component-based architecture, fast rendering and a smooth user experience.

**Node.js**: A backend runtime environment that executes JavaScript on the server. It is fast, scalable, and handles multiple requests efficiently using non-blocking I/O.

**Express.js**: A minimal web framework for Node.js used to build RESTful APIs easily and manage routing and middleware.

**MongoDB**: A NoSQL database used to store data in flexible JSON-like documents, making it easy to manage and scale.

**FUTURE SCOPES**

The current version of ThinkBound provides core functionalities such as book listing, cart management, and basic admin control. However, there is significant scope for future enhancements to make the platform more feature-rich and user-friendly. One of the key improvements would be integrating a secure payment gateway to enable users to complete transactions online. Adding user authentication with role-based access control can help protect admin operations and allow personalised user experiences. Features like order history, tracking, book reviews, and ratings can further enhance user engagement. From the admin side, generating sales reports, tracking inventory, and identifying top-selling books can provide useful insights for better management. The platform can also be extended into a mobile application or converted into a Progressive Web App to improve accessibility on smartphones. Overall, these improvements will make ThinkBound more scalable, secure, and aligned with the expectations of modern ecommerce users.

**CONCLUSION**

The development of **ThinkBound** has been a practical application of full-stack web development using the MERN stack. The project successfully delivers an online book selling platform where users can browse and manage books, and admins can perform inventory operations such as adding, editing, and deleting entries. It replaces the limitations of manual systems with a responsive, real-time, and accessible solution.

By using React.js and Tailwind CSS, the frontend provides a clean and user-friendly interface, while the backend, built with Node.js and Express.js, handles data and API communication efficiently. MongoDB serves as a flexible database solution for storing and retrieving book data. The deployment of the frontend on Netlify and backend on Render ensures that the application is fully hosted and accessible from anywhere. Overall, ThinkBound serves as a strong foundation for further development and showcases how technology can be used to solve real-world problems effectively.

**REFERENCES**

1. https://github.com/nishant219/MERN-BookStore
2. https://github.com/mdalmamunit427/build-full-stack-book-store-mern-app
3. https://www.youtube.com/watch?v=eILUmCJhl64&pp=ygUYa25vd2xlZ2UgZ2F0ZSBtZXJuIHN0YWNr
4. https://github.com/robmelfi/online-store
5. https://www.youtube.com/watch?v=eILUmCJhl64&pp=ygUYa25vd2xlZ2UgZ2F0ZSBtZXJuIHN0YWNr
6. https://github.com/rooneyrulz/book\_store
7. https://github.com/Complete-Coding/HTML\_Complete\_Youtube
8. https://www.youtube.com/watch?v=AZzV3wZCvI4&list=PL78RhpUUKSwfeSOOwfE9x6l5jTjn5LbY3
9. https://github.com/Complete-Coding/CSS\_Complete\_YouTube
10. https://github.com/Complete-Coding/JavaScript\_Complete\_YouTube
11. https://ijrpr.com/uploads/V4ISSUE5/IJRPR12659.pdf
12. https://www.theseus.fi/bitstream/10024/873889/3/Lubna\_Rasel.pdf
13. http://www.ir.juit.ac.in:8080/jspui/bitstream/123456789/9831/1/Bookstore %3B%20E-commerce%20Platform%20with%20MERN%20Stack.pdf
14. https://www.geeksforgeeks.org/bookstore-ecommerce-app-using-mern-stack/
15. https://pcs14.medium.com/mern-stack-refresher-crud-operations-for-a-book-store-15ff826636e9
16. https://www.mongodb.com/en-us/resources/languages/mern-stack-tutorial