

# **BOOKSTORE: E-COMMERCE PLATFORM WITH MERN STACK**

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of  
Technology

in

**Computer Science and Engineering/Information Technology**

By

(SHAAN SRIVASTAVA (191289))

Under the supervision of

(Mr. Nishant Sharma)

**Assistant Professor, Grade II**

To



Department of Computer Science & Engineering and Information Technology  
**Jaypee University of Information Technology Wahnaghat, Solan-173234**  
**Himachal Pradesh**

## Candidate's Declaration

I hereby declare that the work presented in this report entitled “ **Bookstore: E-commerce platform with MERN Stack**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**/ submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **Mr. Nishant Sharma** (Assistant Professor Grade-II - CSE Dept.).

I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Data Science**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

SHAAN SRIVASTAVA

191289

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Supervisor Name: Mr. Nishant Sharma

Designation: Assistant Professor (Grade II)

Department name: Computer Science

Dated:

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_

Name of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
Report Generated on	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

## ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing that made it possible for us to complete the project work successfully.

I am really grateful and wish my profound indebtedness to supervisor **Mr. Nishant Sharma, Assistant Professor**, Department of CSE, Jaypee University of Information Technology, Waknaghat. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Mr. Nishant Sharma**, Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forward or in a roundabout way in making this project a win. In this unique situation, I also want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

SHAAN SRIVASTAVA

191289

## Table of Content

<b>Title</b>	<b>Page No.</b>
<b>Certificate</b>	<b>I</b>
<b>Plagiarism Certificate</b>	<b>II</b>
<b>Acknowledgement</b>	<b>III</b>
<b>Table of Content</b>	<b>IV</b>
<b>Abstract</b>	<b>V</b>
<b>Chapter-1 (Introduction)</b>	<b>1-10</b>
<b>Chapter-2 (Literature Survey)</b>	<b>11-15</b>
<b>Chapter-3 (System Design, Analysis/Design/Development/Algorithm)</b>	<b>16-43</b>
<b>Chapter-4 (Performance Analysis)</b>	<b>44-52</b>
<b>Chapter-5 (Conclusion)</b>	<b>53-56</b>
<b>References</b>	<b>57-58</b>

## **ABSTRACT**

The Book based e-commerce website project is a complete platform for readers and book lovers to explore, discover, and connect with their favorite authors and books. It was created using the MERN stack. With a contemporary, responsive design and simple user interface, this online application offers an enjoyable user experience.

The MERN stack, which consists of MongoDB, Express.js, React, and Node.js, is used to build the project. The application's backend is created with Node.js and Express.js, which offer a powerful API for dealing with user requests and managing the database. The database used to hold all the data about users, books, authors, and reviews is called MongoDB.

The application's frontend was developed using React, which offers a fluid and dynamic user experience. The website is responsive and mobile-friendly for use on all platforms, including PCs, tablets, and smartphones.

Users get access to a sizable selection of books and authors to search through and explore, add titles to reading lists, provide reviews and ratings for books, and interact with other users who have similar interests. Authors are able to set up online accounts, advertise their publications, and communicate with readers.

Overall, the Book Website Project, created with the MERN stack, is a great tool for book enthusiasts to find new books, interact with authors, and meet others who share their interests.

# CHAPTER – 1

## INTRODUCTION

### **1.1 Introduction:**

The Book App, which makes use of the MERN stack, is a cutting-edge web application created to meet the demands of book fans. The Node.js, Express.js, MongoDB, and React.js components of the MERN stack are used to build the project. Because of its adaptability, scalability, and simplicity of usage, this stack is frequently used in web development.

The Book App offers a user interface that is easy to use, contemporary, responsive, and intended to deliver an engaging user experience. Users may visit the website at any time, and any devices, including computers, tablets, and smartphones, can use it.

The Book App allows users to browse a vast library of books, experiment with various genres, and discover fresh writers. With the help of the app, users can easily find new books, read reviews and ratings, and interact with other readers who have similar interests.

The Book App also has features for writers, who may use them to build profiles, advertise their work, and engage with readers. Writers may increase their readership, visibility, and access to constructive criticism with the aid of modern technology.

Through the MERN stack-based Book App project, users and writers have access to a wide-ranging platform for communication, book discovery, and book-related trade. It is an excellent resource for readers and a successful strategy for writers to market their writing.

As a result, the MERN stack-based Book App is a unique and practical tool for readers and writers. It offers a comprehensive platform for discovering new publications, communicating with readers, and marketing and spreading content. Due to its extensive features and user-friendly UI, The Book App is a vital tool for both book lovers and authors.

## **1.2 Problem Statement:**

Due to the growing popularity of online book sales, the usual brick-and-mortar retailers have encountered several difficulties. More individuals are now buying their books from online stores since online book sales have increased significantly recently. The bulk of online book vendors, however, don't offer the same level of interaction and customer care as traditional brick-and-mortar bookstores.

The issue is that there isn't an online bookshop that can provide clients the specialized care and in-person interaction of a brick-and-mortar store while still offering the ease of an online purchase. Most online book retailers do not provide the elements that customers need, such as the ability to explore and discover new books, communicate with other readers, and get individualized suggestions.

Making a book-based e-commerce website that makes use of the MERN stack might solve this issue. This internet application may offer readers a platform for finding new books, interacting with one another, and buying online for their preferred books. The MERN stack can provide an e-commerce website the flexibility, scalability, and user-friendliness needed.

The website has a reader community where users may interact and recommend books, as well as a straightforward user interface and tailored



recommendations. The website may employ a range of social media and marketing strategies, including email marketing and social media advertising, to raise its profile and draw in more visitors.

By providing a platform that offers a tailored user experience, book recommendations, and a reader community while also providing the convenience of online shopping, a book-based e-commerce website built on the MERN stack aims to bridge the gap between traditional brick-and-mortar bookstores and online bookstores.

### **1.3 Objectives:**

- To provide book enthusiasts a convenient online book purchasing and browsing platform: With the help of a user-friendly website, customers will be able to explore and discover books, read book reviews, and buy books online.
- To create a community where readers can communicate, recommend books to one another, and discuss books they have read. The effort aims to create a social network where readers can communicate, recommend books to one another, and discuss books they have read.
- To provide a user-friendly, interactive environment: The project aims to provide a user-friendly, interactive setting that motivates people to browse and purchase books online.
- In order to provide writers a platform where they may advertise their material, interact with readers, and get book reviews, the initiative was created.

- The project tries to enhance the website's search engine visibility in order to increase its prominence in search results and attract more visitors.
- The project aims to integrate a wide range of security measures to safeguard the integrity and safety of user data in order to guarantee its security and privacy.

#### 1.4 Methodology:

- **Planning and Analysis:** The first phase of development consists of analyzing and determining the project's requirements, goals, and scope. The functions, personalities, and user flow of the website must all be determined. Defining the aims and purposes of your website is the first stage. Determine the functionality and services you want to offer, the audience you want to connect with, and the website's objective. To understand your target audience's needs, interests, and behaviors, we must first define your target audience and develop user personas. We could use this data to develop a website that appeals to your target audience. [12]

Researching your competition will enable you to recognise them and discover their strengths and weaknesses. This data may be used to set your website apart from those of your rivals and pinpoint market niches. Define the website's components and features you wish to include, such as user profiles, social sharing, book reviews, and recommendation functions. Create a sitemap and user flow to help you envision how people will explore your website and interact with its features. Plan the architecture and user experience of your website using wireframes and mockups. By doing this, you'll be able to see any possible design flaws

and guarantee a consistent user experience throughout the whole website. [8]

Establish a project plan and timeline to list important project milestones and deadlines. The resources your project will need, such as staff, gear, and software, should be determined. You might compute the project budget and identify potential dangers using this information. Building an excellent book-based website often requires planning and study. To ensure the project's success, it is necessary to identify the target audience, specify the features and objectives of the website, and create a thorough project plan.

- **Design:** Throughout the design phase, wireframes, mockups, and prototypes are created for the user interface, navigation, and layout of the website.

For designing and components we have used **React-Bootstrap:**

Responsive web apps are created using the well-known UI framework React-Bootstrap. The Bootstrap CSS framework was used to generate a collection of pre-built, editable components. [8]

There are several UI components offered by React-Bootstrap, including navigation bars, forms, buttons, modals, and more. These React-built parts are incredibly flexible and easy to customize. [8]

Maintaining and upgrading your application is made simpler by React-Bootstrap's uniform appearance and feel. This is among its key advantages. The components' flexible design allows them to adapt to various screen sizes and devices. [8]

For styling we have used **SCSS (Sassy CSS):**

Sassy CSS, a preprocessor for CSS, adds additional functionality to the CSS syntax. It is an improved version of CSS that enables the development of more sophisticated styles with less code. Given that SCSS is a superset of CSS, every valid CSS code also qualifies as a valid SCSS code. However, SCSS also adds new syntax, such as functions, mixins, variables, and nesting.

SCSS has a many features, including:

**Variables:** In SCSS, variables may be declared and given values to be used throughout your stylesheet. As a result, it is easier to maintain and update your code since you can modify a variable only once and have it affect every instance of your stylesheet.

**Nesting:** You may organize and make your code more readable by nesting selections inside of other selectors using SCSS. Because of this, you might have to write less repetitive code. [13]

**Mixins:** By using reusable code blocks in your CSS, mixins make it simpler to apply sophisticated styles to your HTML components. [13]

**Functions:** Mixins and functions are similar, but functions are more flexible and powerful since they accept parameters and return values.

Overall, SCSS may make the process of creating CSS easier and assist you in writing more effective and manageable code. The preprocessor required to transform the SCSS code into standard CSS, however, can make your development process more challenging.

- **Front-end development:** React is used in the front-end development stage of a website to create the user interface and client-side functionality. Making the website's pages, elements, and interactive features falls under this category.

User interfaces (UIs) for web apps are made using the popular JavaScript library React. It was developed by Facebook and made publically available in 2013. React is now supported by Facebook and a substantial development community.

Reusable UI components may be made by developers using React and used in several places of an application. Simply creating complex user interfaces is possible with these JavaScript-built components. A virtual DOM (Document Object Model) is used by React to manage the state of the user interface and render updates quickly.

React's essential characteristics include things like:

- **Component-based architecture:** With React, you can create reusable user interface (UI) components that can be swiftly combined to create intricate UIs.
- **Declarative programming:** You declare your goals and React takes care of the implementation details in this declarative programming style.
- **Virtual DOM:** Performance is improved by React since it controls the user interface's state and renders changes rapidly.
- **JSX:** By allowing you to write HTML-like code in JavaScript files, the JSX syntactic extension for JavaScript makes it easier to write and understand code.

React supports a wide range of libraries and frameworks, including Redux, React Router, and Material UI. Additionally, it is quite expandable.

React is a robust and flexible framework that can be used to build user interfaces for internet applications. Developers like it because of its component-based architecture, declarative programming, and virtual DOM.

- **Back-end development:** For backend development, Node.js is used to write clean JS code for server development and also Express.js is used to build decent routes and easy to use controllers, services, and databases

using Mongoose. In order to do this, new features must be developed, including social sharing, book search, user authentication, and recommendation engines. [9] [10]

Node.js has a number of important features, including:

- **Event Driven I/O:** Node.js uses a non-blocking, event-driven I/O model that enables it to manage several connections simultaneously without slowing down the program's performance.
- **NPM:** Node.js has an integrated package management called NPM (Node Package management) that enables programmers to quickly install and maintain third-party libraries and modules.
- **Scalability:** Because Node.js is built to be extremely scalable, it is a popular option for developing apps that must manage a lot of traffic.
- **Cross-platform:** Node.js can work on any system (Linux, Mac Windows).

APIs (Application Programming Interfaces) are regularly developed with Node.js, as are web apps and real-time applications. Its interoperability with a variety of web frameworks, such as Express.js, etc. allows developers a range of choices.

### **Express JS:**

An open-source web application framework for Node.js called Express.js is popular. It offers a complete set of capabilities that make it easy and quick to create web apps and application programming interfaces (APIs).

Due to the fact that Express.js, which is based on Node.js, allows JavaScript to be used in both the front end and the back end of a web application. It offers a simple framework for creating web applications, allowing programmers to concentrate on the essential components of

their project without being constrained by needlessly complex framework features.

Express.js's important characteristics include the following:

- **Routing:** Express.js provides a simple and flexible method to build routes for handling HTTP requests and responses.
- **Middleware:** Programmers may integrate functions like logging, authentication, and error handling into their applications using the middleware layer provided by Express.js. [9][10]
- **Template Engine:** EJS and Pug are just a few of the many templating engines that Express.js supports, making it simple to create dynamic HTML pages.
- **Modularity:** Express.js's modularity feature makes it simple for developers to divide their applications into smaller, reusable components.

Express.js is frequently used to create APIs and online apps. It frequently pairs with other well-known Node.js frameworks and modules, like MongoDB.

- **Database development:** During this stage, MongoDB is used to design and create the queries, data models, and database structure that are necessary for the website. [7] [11]

MongoDB, a well-known database management system for documents, makes use of NoSQL (non-relational) technology. It has grown in prominence as a platform for developing cutting-edge web apps since its release in 2009. [11]

MongoDB stores data as flexible documents with varying forms that mimic JSON. This makes it possible for developers to store and retrieve

sophisticated data structures without having to conform to a set model, unlike traditional relational databases.

MongoDB has a number of important characteristics, including:

- **Scalability:** MongoDB can manage massive volumes of data and traffic because of its highly scalable architecture.
- **Flexibility:** Developers may store data in a flexible manner using MongoDB's document-oriented data format, which makes it simple to modify and update data structures.
- **Performance:** MongoDB has built-in caching and automated sharding, making it a very performant database.
- **Aggregation:** MongoDB has strong aggregation features, making it simple to carry out complicated data analysis and aggregation.[11]

MongoDB is commonly used to create real-time applications and application programming interfaces (APIs). It typically works in conjunction with cutting-edge web development technologies like Node. All things considered, MongoDB is a robust and flexible database management system with a lot to offer creators of modern web applications. Because of its document-oriented data format, scalability, and performance, developers regularly employ it. [7] [11]

- **Integration and testing:** During the integration and testing phase, the front-end, back-end, and database components are integrated and the functionality, performance, and security of the website are tested.



## **CHAPTER – 2**

### **LITERATURE SURVEY**

#### **1) Comparative analysis of MEAN stack and MERN stack-**

**Authors:** Sanchit Agarwal and Jyoti Verma

**Publisher:** International Journal of Recent Research Aspects

The MEAN (MongoDB, Express.js, Angular, and Node.js) and MERN (MongoDB, Express.js, React, and Node.js) web development stacks are contrasted in the paper. The two stacks are assessed in the article based on a number of factors, including performance, scalability, simplicity of development, and community support. [1]

The paper's succinct and unambiguous comparison of the two stacks is one of its strong points. Each technology in both stacks is thoroughly studied by the writers, who then compare them according to a number of different factors. They also shed light on which stack would be more appropriate for a certain kind of web app. [1]

The paper does, however, have several shortcomings. It doesn't go into great depth about how the two stacks are implemented or how to leverage them to create particular kinds of web apps. Additionally, the report does not critically examine any of the two stacks' potential shortcomings or limits; instead, it just compares the two stacks.

For developers who are debating which stack to employ for their web application, "Comparative analysis of MEAN stack and MERN stack" is a useful reference. The article offers a thorough comparison of the two stacks and can assist developers in making a choice based on their own needs and specifications. [1]

#### **2) A Review on Technologies used in MERN stack:**

**Authors:** Vaishnavi Joshi, Rujuta Nikam, Ishali Gawande and Prof. Sudesh A. Bachwani, Mohanish Bawane

**Publisher:** International Journal for Research in Applied Science & Engineering Technology (IJRASET)

The paper gives a general overview of the MERN (MongoDB, Express.js, React, and Node.js) stack of technologies. The article analyzes each technology's benefits and drawbacks and offers insights into how they interact to produce reliable and scalable web applications. The paper's ability to concisely and clearly describe each technology is one of its strong points. It is simple for readers to comprehend how each technology functions within the context of the MERN stack since the authors include examples and use cases to show the advantages and disadvantages of each technology. [2]

The paper does, however, have several shortcomings. The MERN stack's implementation and how it might be used to create particular kinds of web applications, such e-commerce websites or social media platforms, are not covered in length. A critical evaluation of the MERN stack or any possible drawbacks is also absent from the study.

For those who are unfamiliar with the MERN stack and wish to learn more about its components in general, this paper is a useful reference. It gives a thorough review of each technology and explains how they combine to form a solid web application. [2]

### 3) **Key Factors for Developing a Successful E-commerce Website:**

**Authors:** Osama Mohammed Ahmad Rababah and Fawaz Ahmad Masoud

**Publisher:** IBIMA Publishing Communications of the IBIMA

The paper gives a thorough examination of the important elements that influence the performance of e-commerce websites. The writers have listed a number of elements, such as user experience, security, usability,

design, customer support, and payment choices, that are essential for creating a successful e-commerce website. [3]

The paper's merits are found in its thorough research of each aspect, which offers insights into their significance and practical use. The writers have also included helpful advice and ideas for enhancing many parts of an e-commerce website, making it a useful tool for programmers and companies wishing to launch or enhance their online presence.

The paper's primary focus on general e-commerce websites and lack of specialized information about book-based websites is one of its weaknesses. Overall, the article offers a thorough analysis of the critical elements that influence the performance of e-commerce websites. [3]

#### **4) Protecting Websites from Attack with Secure Delivery Networks:**

**Authors:** Ramesh K. Sitaraman, David Gillman, Bruce Maggs, Yin Lin

**Publisher:** IEEE

The idea of secure delivery networks (SDNs) is discussed in the research paper along with how they can defend websites against different kinds of assaults. The effectiveness of SDNs in thwarting attacks like distributed denial of service (DDoS), cross-site scripting (XSS), and SQL injection is assessed in the article. [4]

The paper's thorough analysis of SDNs and how well they defend websites from assaults is one of its strong points. The authors offer a thorough examination of SDNs' functionality, capabilities, and threat mitigation applications. They also shed light on the drawbacks of SDNs and how to get around them.

The paper's emphasis on application is another strength. It is simpler for users to comprehend how SDNs may be applied in their own projects

because the authors give instances of how they have really been utilized to defend websites against assaults. [4]

But there are certain restrictions on the document. It does not give in-depth information on alternative security measures that may be employed in addition to SDNs because its main focus is on the usage of SDNs to safeguard websites. The report also does not include a critique of the drawbacks of SDNs or suggestions on how to get around them.

#### **5) Real-World Decision Making: Logging Into Secure vs. Insecure Websites**

**Authors:** Timothy Kelley, Bennett I. Bertenthal

**Publisher:** IEEE

The study looks at the variables that affect users' choices when deciding whether to log into safe or insecure websites. The research looks at how users assess a website's security based on several visual signals and how they balance the advantages and disadvantages of utilizing a certain website. [5]

The paper's emphasis on making decisions in the actual world is one of its advantages. To mimic how consumers actually decide what to do when they enter onto websites, the authors conducted a lab experiment. The study sheds light on the thought processes individuals use to decide how secure a website should be.

The paper's utilization of a multidisciplinary approach is another asset. To present a thorough examination of the elements influencing consumers' judgements about website security, the writers rely on research from the fields of psychology, computer science, and information security. [5]

#### **6) Building an online shop application with MERN stack:**

**Authors:** Tien Pham

**Publisher:** Metropolia University of Applied Sciences

The paper begins by outlining the various MERN stack elements and their significance in creating the e-commerce application. The setup of the MongoDB database, construction of the backend using Node.js and Express, and development of the frontend using React are all covered in the next sections of the article. [6]

The article also discusses how JSON Web Tokens (JWTs) and Passport.js are used to create authentication and authorization functionality. The integration of a payment gateway using Stripe is also covered by the author.

The paper serves as a thorough tutorial for creating an e-commerce application utilizing the MERN stack. It covers the key elements of environment setup, backend and frontend development, authentication and authorization implementation, and payment gateway integration. [6]

## CHAPTER 3

### SYSTEM DEVELOPMENT

#### **Why do we need E-Commerce websites?**

E-commerce websites are essential for many reasons:

- **Convenience:** E-commerce websites make it simple for people to buy things without having to go to real stores since they let them shop from the comfort of their homes or offices at any time of day or night.
- **Global audience:** E-commerce websites allow companies to contact a worldwide clientele, erasing geographical boundaries and enabling them to grow their clientele beyond their local region.
- **Cost-effectiveness:** E-commerce websites sometimes have lower overhead costs than conventional businesses do, such as rent, utilities, and personnel pay.
- **Sales growth:** E-commerce websites may assist firms in growing their sales by facilitating consumer product discovery and purchases, delivering personalized suggestions, and providing discounts and promotions.
- **Analytics:** E-commerce websites give businesses access to useful data and analytics that can be utilized to streamline processes, enhance customer satisfaction, and boost sales.

So, we decided to design a website for book sales which is completely authenticated and secure and easy to use for customers. [7]

Our project can be broken down into the following stages:

- **Planning and Analysis:** During this phase, the project's parameters and the needs for the e-commerce website are established. This includes figuring out the target market, the kinds of books to sell, the payment choices, the shipping procedures, and other aspects that must be put into place. The website's blueprint is created once a thorough examination of the requirements is completed. [5]
- **Design:** The layout, color scheme, typography, and visuals for the website are all produced at this stage. The layout should be simple to use, visually appealing, and navigable.

An essential component of the overall design and user experience of a MERN (MongoDB, Express, React, Node.js) programme is the UI (User Interface). Here are some pointers for creating a MERN app's user interface:

- **Ensure simplicity:** Users may find it easier to explore your app with a clear and straightforward design. Make good use of the white space and refrain from overcrowding the interface.
- **Use a responsive design:** As more and more people use the internet from mobile devices, it's critical to make sure your MERN app is adaptable and can adjust to various screen sizes. [6]
- **Pick a color palette that is dependable:** Choose a color scheme for your MERN app that represents your brand and stick with it.
- **Use icons and graphics:** to assist users rapidly distinguish between different parts in your MERN app and to improve the app's aesthetic attractiveness.

- **Easy to navigate:** Make sure the MERN app is simple and easy to navigate. Users should be able to navigate the app without getting lost and locate what they're searching for promptly.
- **Provide users feedback:** When users engage with various MERN app components, provide them feedback. Give a visual confirmation, for instance, when a button is clicked or a form is submitted. [4]
- **Development:** At this point, the MERN stack is actually used to start building the website. React and React-Bootstrap are used to build the front end, and Node.js, Express.js, and MongoDB are used to build the back end. The website should be enhanced for scalability, security, and performance.
- **Testing:** A website undergoes a rigorous testing procedure once it is developed in order to identify and address any flaws or issues. It is important to do functional, security, performance, and usability tests.

Testing is a key phase in the development of software since it ensures that the programme works as planned. The three forms of testing that may be done are unit testing, integration testing, and end-to-end testing.

Unit testing is a method used to test certain elements or functionalities of an application. The MERN stack may be subjected to unit testing using testing frameworks like Mocha or Jest.

Integration testing examines how various application components interact with one another. Trying things out, like how the frontend and backend interact. [4]

Frameworks like Cypress or Selenium may be used for integration testing.

End-to-end testing entails evaluating the entire programme while mimicking user interactions. Frameworks like Puppeteer or TestCafe may be used for end-to-end testing.



In addition to these many types of testing, other aspects of testing, such as load testing and security testing, must be considered.

Demand testing is the process of analyzing an application's performance under a heavy demand. For this, you can use programmes like Apache JMeter or LoadRunner.

Security testing is the process of putting the application's security measures to the test to make sure it is protected from attacks. Two tools that might be used for this are OWASP ZAP and Burp Suite.

- **Deployment:** Following extensive testing, the website is set up on a web server or cloud hosting platform. To guarantee that the website is constantly current and functioning properly, the deployment process should be automated and incorporate continuous integration and deployment (CI/CD). [4]

#### **Technologies/Libraries used:**

- 1) **React-Bootstrap:** In order to employ helpful React.js components, we choose to use the **React-Bootstrap** framework. Speaking specifically about the library:

React-Bootstrap, a well-known UI framework, is used to build responsive web applications. It is a set of ready-to-use, customizable components created with the help of the Bootstrap CSS framework.

React-Bootstrap provides a wide range of UI components, such as navigation bars, forms, buttons, modals, and more. These React-built elements are easily styleable and adaptable.

React-Bootstrap gives a consistent look and feel throughout your application, making maintenance and updating easier. One of its main benefits is this. Because the components are responsive in design, they will adjust to different screen sizes and devices. [8]

**2) React.js:** During the front-end development stage, React is used to create the website's user interface and client-side functionality. Making the pages, elements, and interactive features of the website falls under this category. [8]

React is a popular JavaScript library that is used to build user interfaces (UIs) for internet applications. It was developed by Facebook and released to the public in 2013. Today, React is supported by both Facebook and a substantial developer community.

Reusable UI components that may be used in many different parts of an application can be made by developers using React. Simple user interface creation may be done using these JavaScript-built components. React uses a virtual DOM (Document Object Model) to maintain the state of the user interface and swiftly render changes.

Redux, React Router, and Material UI are just a few of the many libraries and frameworks that React can be used with. It is also quite extensible.

In general, React is a strong and adaptable toolkit for creating user interfaces in online applications. It is a well-liked option among developers due to its component-based architecture, declarative programming, and virtual DOM.

**3) Node.js:** Node.js is a backend based framework which is used to develop API which can be used for various purposes. It is an open-source platform based on Javascript. Previously only feasible with languages like Python, Ruby, and PHP, writing server-side code is now made available with JavaScript. [9]

Developing server-side apps, online APIs, and microservices with Node.js is common practice in web development. The creation of desktop apps and software for the Internet of Things (IoT) are only a couple of the numerous businesses that employ it. [9]

4) **Express.js:** Express is a Node.js web framework with a lot of features for developing online and mobile apps. It is speedy, flexible, and easy to use. It has developed into one of the web frameworks for building web apps that is most widely used in the Node.js community. [10]

Express provides a simple and approachable API for controlling HTTP requests and responses. In order to improve the functionality of the web application, middleware, which handles duties like authentication, logging, and error handling, is also included. EJS and Pug are only two of the many templating engines that Express supports and allows you to employ to create dynamic HTML pages.

Express is open source, and a substantial developer community actively contributes to and supports its development.[10]

5) **MongoDB:** In this project, mongoose was used for mongoDB. Mongoose is an Object Data Modelling (ODM) module for Node.js and MongoDB. The higher level of abstraction it provides over the MongoDB driver makes working with MongoDB databases in Node.js applications easier. Mongoose uses a schema, which is a description of the document's structure, to build your data models. The data may then be subject to limitations like data types, required fields, default values, and validation procedures. Mongoose provides a wide range of options for dealing with data, including querying, updating, deleting, and aggregating data. [11]

6) **Nodemailer:** For sending email to the admin and user after completing the shopping, we have used the Nodemailer module which is built in Node.js. Email messages may be sent via a variety of transport protocols thanks to its simple and flexible interface. Using Nodemailer makes sending emails with HTML content, attachments, and integrated images straightforward. [16]

Nodemailer supports many recipients and may send emails using a variety of SMTP servers or services, such as Gmail. Since it supports a wide range of email authentication techniques, such as OAuth2, it is genuine to communicate with many email providers. [16]

One of the main advantages of Nodemailer is its capacity to handle errors and automatically repeat failed email sending attempts. Additionally, it might be configured to monitor email sending problems using custom error handling algorithms or third-party error monitoring systems.

- 7) **JWT:** JWT, or JSON Web Tokens, is a sort of authentication technique that lets users securely send data between parties in the form of a JSON object. [14]

With a special secret key that only the server is aware of, the server produces a token in a JWT authentication system. After a successful login, this token is provided to the client and saved there. The client provides this token in the request header for each future request to the server, enabling the server to validate the token and validate the user.

Compared to conventional session-based authentication systems, JWT authentication provides a number of benefits. First off, because JWT tokens are complete and capable of holding all relevant user data, they can do away with the necessity for session storage on the server. As a result, JWT authentication can grow considerably more easily since it requires less server storage. [14]

JWT authentication, however, may potentially have significant drawbacks. An attacker who successfully intercepts a token may be able to access all the secured resources linked to it. A security risk arises if a token is stolen or lost since JWT tokens cannot be invalidated until they expire because they are self-contained.

**8) Passport.js:** A popular middleware for authentication in Node.js apps is Passport.js. It is quite adaptable and works with a variety of authentication methods. Passport.js is renowned for its dependability, simplicity, and ease of usage. It offers a selection of authentication methods, including third-party, local, and social authentication providers. [14]

For username- and password-based authentication, which is frequently used for online applications, a local authentication technique is utilized. Users are authenticated through social authentication strategies using OAuth providers like Facebook, Google, Twitter, etc. Users can also be authenticated in Passport.js using third-party authentication services.

By offering a standard interface and combining several authentication techniques, Passport.js streamlines the authentication and authorisation process in Node.js apps. It simplifies the implementation of safe and dependable authentication in apps by abstracting the specifics of authentication from developers. [14]

Passport.js enables developers to design unique authentication methods depending on their particular needs in addition to the pre-built authentication strategies. Because developers may select the authentication strategy that best matches their application, authentication is now more flexible.

Passport.js is a strong authentication mechanism for Node.js apps overall. It is a popular option for developers wishing to establish safe and dependable user authentication and authorization due to its flexibility, usability, and dependability.

**9) Bcrypt:** We have used the Bcrypt library for encryption and decryption of the user passwords. For password hashing and encryption, Node.js users frequently use the Bcrypt package. By encrypting user passwords

into an unbreakable, irreversible string of characters, it offers a method for securely storing user passwords. Bcrypt creates distinct and unexpected encrypted passwords for each user by combining salting and hashing methods. [15]

For saving the password in encrypted format:

```
let saltRounds = 10;
let hashedPass = bcrypt.hashSync(body.password, saltRounds);
let newUser = new UserModel({
  name: body.name,
  username: body.username,
  password: hashedPass,
  email: body.email,
  role: "User",
});

await newUser.save();
return newUser;
```

For verifying the user:

```
verified=bcrypt.compareSync(enteredPassword, actualPassword);
```

“Verified” is a boolean variable which tells whether the user is verified or not i.e. entered password matches the actual password or not.

Before hashing the user's password, salting includes adding a random string of characters to it. Even if two users share the same password, the resultant hash is unique to each of them as a consequence. The technique of hashing creates an unchangeable string of characters from the initial password and salt. Blowfish, a potent hashing algorithm, is used by Bcrypt to generate the hash.

A way to compare a plaintext password with the hashed password is also included in Bcrypt. Bcrypt fetches the previously saved hash from the database and compares it with the hashed version of the user's password when they log in. The user is allowed access if they match.

Passwords may be safely stored in Node.js with the help of Bcrypt, which also shields user accounts from theft and hacking. It is a commonly used library that security professionals advise using to password-protect web applications.

**10) Axios:** A popular JavaScript package for sending HTTP requests to web servers is called Axios. Requests are often sent from the frontend React components to the backend Node.js server in MERN stack apps. [15]

For performing various HTTP requests, including GET, POST, PUT, DELETE, and others, Axios offers a number of alternative ways. Here are a few succinct descriptions of these techniques:

- **GET:** Data is retrieved from the server using the GET technique. To get data from a database or an API, it is frequently used. The `axios.get()` function in Axios may be used to invoke the GET method, passing the URL to be obtained as an argument.

**Example:**

```
response =
axios.get(`${process.env.REACT_APP_API_URL}/${id}`);
return response;
```

- **POST:** Data is sent to the server via the POST method. A database or an API are frequently updated or added to using this method. The `axios.post()` function in Axios may be used to call the POST method, and the data to send is supplied as an argument.

**Example:**

```
axios.post(
  `${process.env.REACT_APP_API_URL}/cart/compareQuantity`,
  {
    userId: userId,
    bookId: bookId
  }
);
```

- **PUT:** The server's existing data can be updated using this technique. The `axios.put()` function in Axios may be used to invoke the PUT method, passing the data to be updated as an argument.

**Example:**

```
axios
.put(`${process.env.REACT_APP_API_URL}/${id}`, {
  title: title,
  author: author,
  price: price,
  description: description,
  status: status,
  quantity: quantity,
  sale_price: sale_price,
});
```

- **DELETE:** The server's data can be deleted using this technique. The `axios.delete()` function in Axios may be used to invoke the DELETE method, passing the data to be erased as an argument.

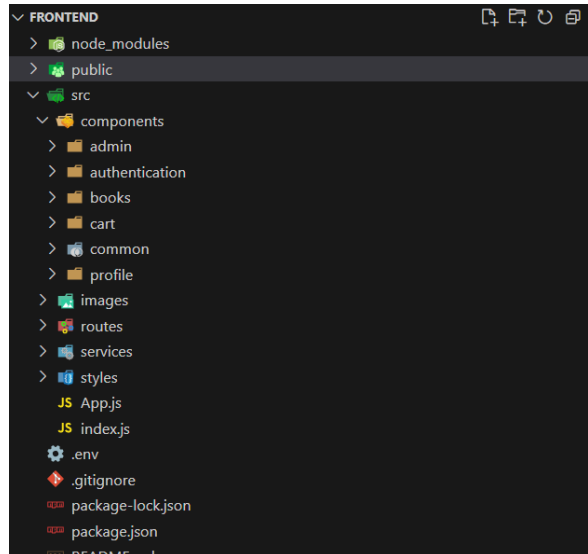


Example:

```
axios.delete(`${process.env.REACT_APP_API_URL}/${id}`);
```

### Folder structure:

- 1) **Frontend:** Main App.js file is present in the parent directory of the project. Execution of the app starts from this file. The Src folder contains the components and each component is segregated into different folders with different purposes. We also have a services folder for the API calls and routes folder for protection of routes from unauthenticated users.

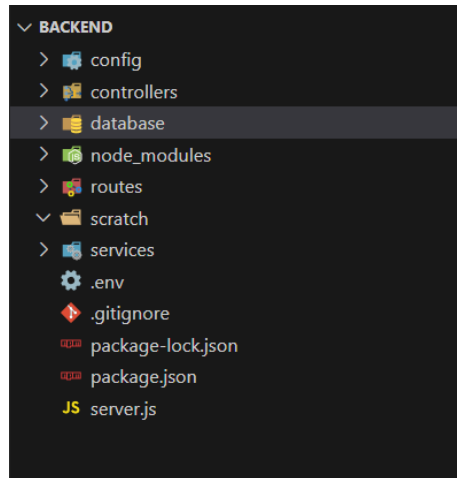


**Fig 3.1: Folder structure for Frontend**

- 2) **Backend:** Execution of the backend folder starts from the server.js file which is present in the parent directory of the folder and different folders are present in the project:

Following line starts the execution of the backend code:

```
app.listen(process.env.PORT, () => {  
  console.log(`Backend server running on port  
  ${process.env.PORT}`);  
});
```



**Fig 3.2: Folder structure for Backend**

**Controller:** Contains files for functions that are to be executed for the routes hit from the frontend. Every controller has a class and they have static methods enclosed inside them.

**Example:**

```
class BookController {
  static async getBooks(req, res) {
    let books;
    try {
      books = await BookService.findBooks(req.body.userId);
    } catch (err) {
      console.log(err);
    }

    if (!books) {
      return res.status(400).json({ message: "No books found"
});
    }

    return res.status(200).json({ books });
  }
}
```

**Routes:** Contains different routes and their description. Every router file contains a “router” instance of the express module.

**Example:**

```
const express = require("express");

const router = express.Router();

router.get("/cart/getQuantities/:userId/:itemId/:bookId",
  CartController.getQuantities)

module.exports = router;
```

**Services:** Contains mongoose calls to the database to fetch and post data for different controllers. Every service file has its own class and it has many static methods inside them.

**Example:**

```
class UsersService {
  static async getUsers() {
    const users = await UserModel.find();
    return users;
  }
}
```

**Database:** Contains schema of different models.

**Schema of the databases:**

- 1) **Books Schema:** Every book is donated by field which is the unique ObjectID of the user that donated the book.

```
const BookSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  author: {
    require: true,
    type: String,
  },
  quantity: Number,
  description: String,
  price: Number,
  sale_price: Number,
  status: String,
  donatedById: ObjectId,
  donatedByEmail: String,
  isDeleted: Boolean,
});
const BooksModel = new mongoose.model("Books", BookSchema);
module.exports = BooksModel;
```

- 2) **Cart Schema:** It contains a collection of the cart where each cart item has a `userId` field which contains the unique `ObjectID` of the user in the `users` collection. Thus, a user is mapped with the cart item.

```

const CartSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  bookId: {
    type: ObjectId,
    required: true,
  },
  author: {
    require: true,
    type: String,
  },
  quantity: Number,
  price: Number,
  sale_price: Number,
  userId: ObjectId,
  userEmail: String,
});
const Books = new mongoose.model("Cart", CartSchema);
module.exports = Books;

```

**3) User Schema:** The user schema is a crucial component of web development because it enables programmers to specify the layout of the user object and guarantee that all necessary data is appropriately gathered and saved. The user's information must be accessed in order to authenticate their identity and establish their degree of access to the features and data of the application. This schema is also essential for authentication and authorization procedures.

```

const mongoose = require("mongoose");
mongoose.set("strictQuery", false);
const dbSchema = new mongoose.Schema({
  name: String,
  username: String,
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: String,
  role: String,
});
const Users = new mongoose.model("Users", dbSchema);
module.exports = Users;

```

Connection file of the database:

```

class DbUtil {
  static connect() {
    mongoose
      .connect(uri, {
        useNewUrlParser: true,
        useUnifiedTopology: true,
      })
      .then(() => {
        console.log("Connected to DB");
      })
      .catch((err) => {
        console.log(err);
      });
  }
}
module.exports = DbUtil;

```

## **Using Passport JS for google login:**

Popular authentication middleware for Node.js called Passport.js offers web applications a straightforward and flexible approach to integrate user authentication. It supports a number of authentication methods, including social network login, local username/password authentication, and third-party authentication providers like Google, Facebook, and Twitter. [14]

A variety of mechanisms are offered by Passport.js for user authentication across various channels. These tactics can be simply changed out or combined to build a unique authentication pathway because they are designed to be modular. For instance, the Google approach may be used to authenticate people using their Google account while the local strategy can be used for username/password authentication. [14]

Additionally, Passport.js offers middleware that may be used to determine if a user has been authenticated and granted access to a certain resource. By limiting access to specific pages or API endpoints, this middleware may make sure that only authorized users have access to sensitive data. [14]

You must first establish a Google API Console project and receive a client ID and secret in order to allow Google login using Passport.js. Installing the passport-google-oauth20 package, which offers a Google OAuth 2.0 authentication method, is the next step. [14]

## Code:

```
passport.use(  
  new GoogleStrategy(  
    {  
      clientID: process.env.CLIENT_ID,  
      clientSecret: process.env.CLIENT_SECRET,  
      callbackURL: "/auth/google/callback",  
      scope: ["profile", "email"],  
    },  
    async function (accessToken, refreshToken, profile, callback) {  
      callback(null, profile);  
      const user = await userServices.findUserByUsername(  
        profile.emails[0].value  
      );  
  
      let body = {  
        name: profile.displayName,  
        username: profile.emails[0].value,  
        password: profile.emails[0].value,  
        email: profile.emails[0].value,  
        role: "User",  
      };  
    }  
  );  
);
```

```
if (!user) {  
  let newUser = userServices.registerUser(body);  
} else {  
  // Set the item in the local-storage folder  
  localStorage.setItem("user", JSON.stringify(body));  
}  
}  
);  
);  
);  
  
passport.serializeUser((user, done) => {  
  done(null, user);  
});  
  
passport.deserializeUser((user, done) => {  
  done(null, user);  
});
```



## Database:

```
_id: ObjectId('643677eb71375bc92154cfb1')
name: "Akshit"
username: "akshit1234"
email: "akshit123@gmail.com"
password: "$2b$10$MHJwzt4vv2r5Qy0T.l/ZiOwZZAo8iFGjQ.HbqaoVEyuno7FQilj9m"
role: "User"
__v: 0
```

```
_id: ObjectId('6438f9a21a9ea9a805f19102')
name: "Kunal Bhandari"
username: "kunalbhandari"
email: "kunalbhandari@gmail.com"
password: "$2b$10$xT8o67WdZCgmr55TkDr3ZecTQbwU46aTnGkSof72X9EtPwV7ty9ey"
role: "User"
__v: 0
```

```
_id: ObjectId('6438fc9822c4eab64bbd3a7c')
name: "Narendra Biceps"
username: "narendrabiceps"
email: "narendrabiceps@gmail.com"
password: "$2b$10$pMPgGQoKTPmNfv4Q30Xua.2T70WqEMGRMtPlVvKtXNN7ue/GqWQLG"
role: "User"
__v: 0
```

**Fig 3.3: Users Collection**

```
_id: ObjectId('645200f70b116eae29c9ec0a')
title: "Abyss Recoiling"
author: "Cornelis Elli"
quantity: 9
description: "The piano sat silently in the corner of the room. Nobody could remembe..."
price: 999
sale_price: 299
status: "available"
donatedById: ObjectId('6433a051c0de20ed41ba2fe8')
donatedByEmail: "pratham0410@gmail.com"
isDeleted: false
__v: 0
```

```
_id: ObjectId('6452013c0b116eae29c9f4d4')
title: "Mists of Artemis"
author: "Jozsi Wulfnoi"
quantity: 10
description: "Hopes and dreams were dashed that day. It should have been expected, b..."
price: 1899
sale_price: 1199
status: "available"
donatedById: ObjectId('6434ea0e51e16203191c66ee')
donatedByEmail: "shaan754@gmail.com"
isDeleted: false
```

**Fig 3.4: Books Collection**

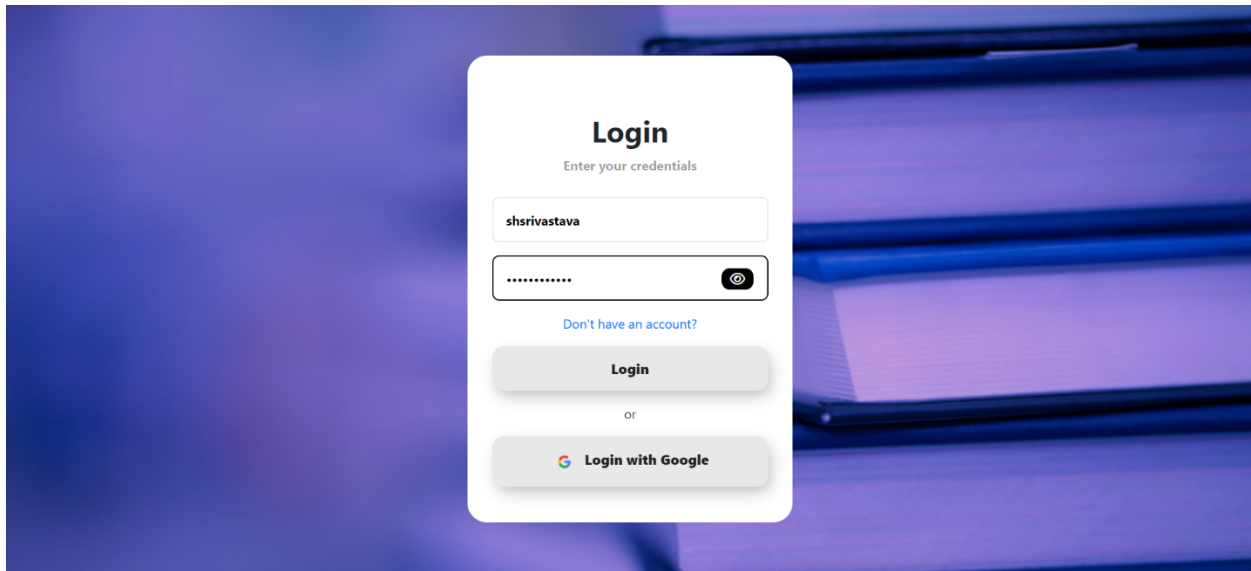
```
  _id: ObjectId('644b750c53a2015df80bbd50')
  title: "Strike the Shadow"
  bookId: ObjectId('6448c5a6eb5438c0e33b7bdf')
  author: "Bevan Dana"
  quantity: 2
  price: 199
  totalPrice: 398
  userId: ObjectId('6433a051c0de20ed41ba2fe8')
  userEmail: "pratham0410@gmail.com"
  __v: 0

  _id: ObjectId('64520aa3332f7f064d519e0e')
  title: "The Deadly Staircase"
  bookId: ObjectId('645201760b116eae29c9f825')
  author: "Penka Iruyel"
  quantity: 2
  price: 2399
  sale_price: 1399
  totalPrice: 4798
  userId: ObjectId('6433dfec2a4905734dd5faa7')
  userEmail: "admin2023@gmail.com"
  __v: 0
```

**Fig 3.5: Cart Collection**

**Flow of the App:**

- 1) Authentication:** First the user will have to login with his credentials. If he does not have the credentials then he can register with a new account. We have used Localstorage to save the data of logged in users and protect the routes. [14]  
Authentication is a crucial security mechanism that aids in guarding against unauthorized access and safeguarding sensitive data. A website and its data might theoretically be accessed by anybody without authentication, which could result in data breaches, identity theft, and other security problems.



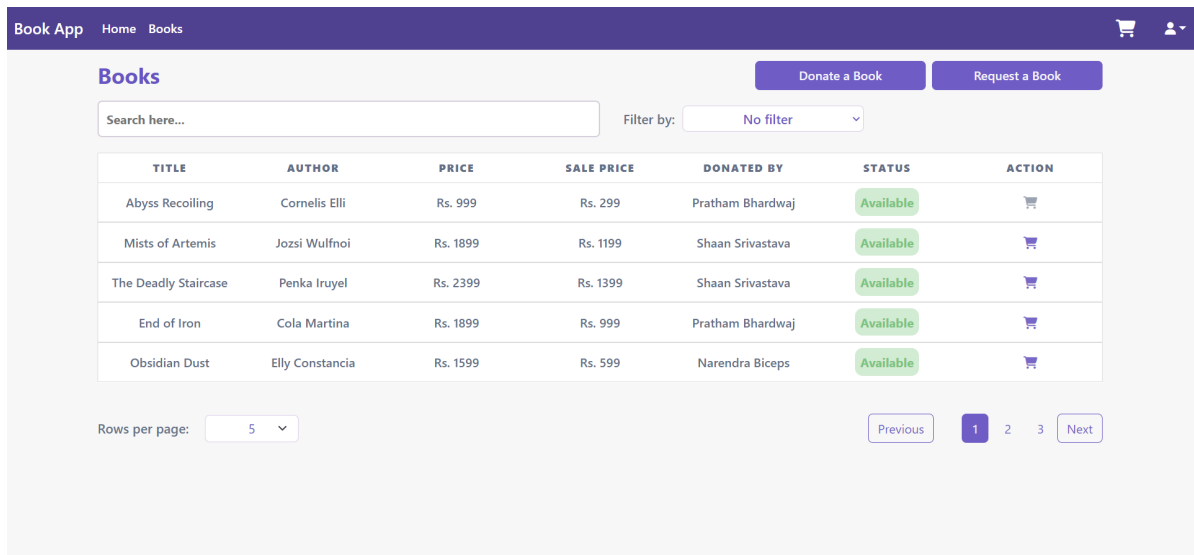
**Fig 3.6: Login Page**

- 2) **Books List:** Books list will be displayed to the user when he is logged in. We have tried to implement the features of a data table into the books table without using any library. [12]

**Datatables:** Intuitive, feature-rich, and configurable HTML tables with sophisticated capabilities like pagination, sorting, filtering, searching, and more may be made with the help of the robust DataTables jQuery plugin. It offers a straightforward and user-friendly interface for visualizing and modifying massive volumes of data from several sources, including JSON, XML, arrays, and server-side processing.

The MERN stack is only one of the many web technologies that DataTables may be linked with. Data from MongoDB collections may be shown using it, and users can interact with the data by using capabilities like sorting, searching, and pagination. Overall, DataTables offers a complete solution for quickly and easily creating sophisticated and interactive HTML tables.

Developers may construct dynamic, responsive tables using DataTables that are optimized for speed and performance. Additionally, it enables modification of the table's behavior and look using a number of choices and APIs. Additionally, it offers server-side processing, allowing the table to effectively manage big datasets.



The screenshot shows a web application interface for a 'Books' section. At the top, there is a navigation bar with 'Book App', 'Home', and 'Books' links, along with a shopping cart icon and a user profile icon. Below the navigation bar, there are two buttons: 'Donate a Book' and 'Request a Book'. A search bar is labeled 'Search here...' and a filter dropdown is set to 'No filter'. The main content is a table with the following data:

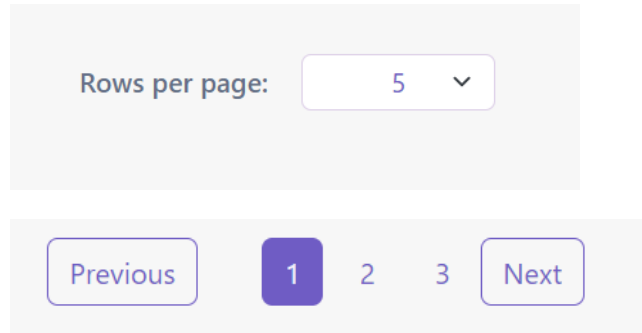
TITLE	AUTHOR	PRICE	SALE PRICE	DONATED BY	STATUS	ACTION
Abyss Recoiling	Cornelis Elli	Rs. 999	Rs. 299	Pratham Bhardwaj	Available	
Mists of Artemis	Jozsi Wulfnoi	Rs. 1899	Rs. 1199	Shaan Srivastava	Available	
The Deadly Staircase	Penka Iruyel	Rs. 2399	Rs. 1399	Shaan Srivastava	Available	
End of Iron	Cola Martina	Rs. 1899	Rs. 999	Pratham Bhardwaj	Available	
Obsidian Dust	Elly Constanca	Rs. 1599	Rs. 599	Narendra Biceps	Available	

Below the table, there is a 'Rows per page:' dropdown set to '5' and a pagination control showing 'Previous', '1', '2', '3', and 'Next'.

**Fig 3.7: Books Table**

**Features of the booklist are:**

- **Pagination:** The entire data is paginated so that when the number of books increases, then data can be displayed without scrolling the page.



Main code for pagination:

```
indexOfLastRowOfCurrentPage = currentPage * rowPerPage;
indexOfFirstRowOfCurrentPage = indexOfLastRowOfCurrentPage -
rowPerPage;

let currentRows = tableBooks?.slice(
  indexOfFirstRowOfCurrentPage,
  indexOfLastRowOfCurrentPage
);

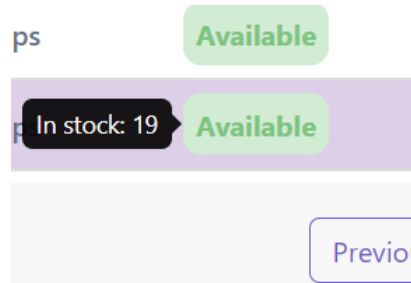
const pageNumbers = [];

let tableBooksLength;
if (tableBooks?.length) {
  tableBooksLength = tableBooks?.length;
}

for (let i = 1; i <= Math.ceil(tableBooksLength / rowPerPage);
i++) {
  pageNumbers.push(i);
}

const paginate = (pageNumber) => {
  setCurrentPage (pageNumber);
};
```

- **Tooltips:** When a user hovers on the status of the book, then he can see the quantity of book in the tooltip. Similarly he can see the email ID of the user on hovering the donated column.



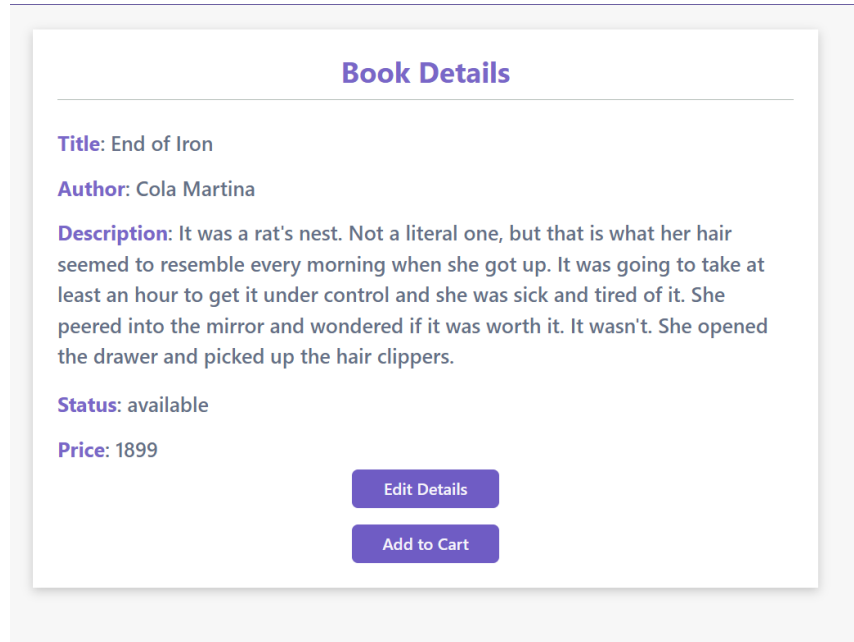
**Fig 3.8: Tooltips**

- **Search Box:** Users can search a particular book with the author name or book name.

TITLE	AUTHOR	PRICE	SALE PRICE	DONATED BY	STATUS	ACTION
End of Iron	Cola Martina	Rs. 1899	Rs. 999	Pratham Bhardwaj	Available	

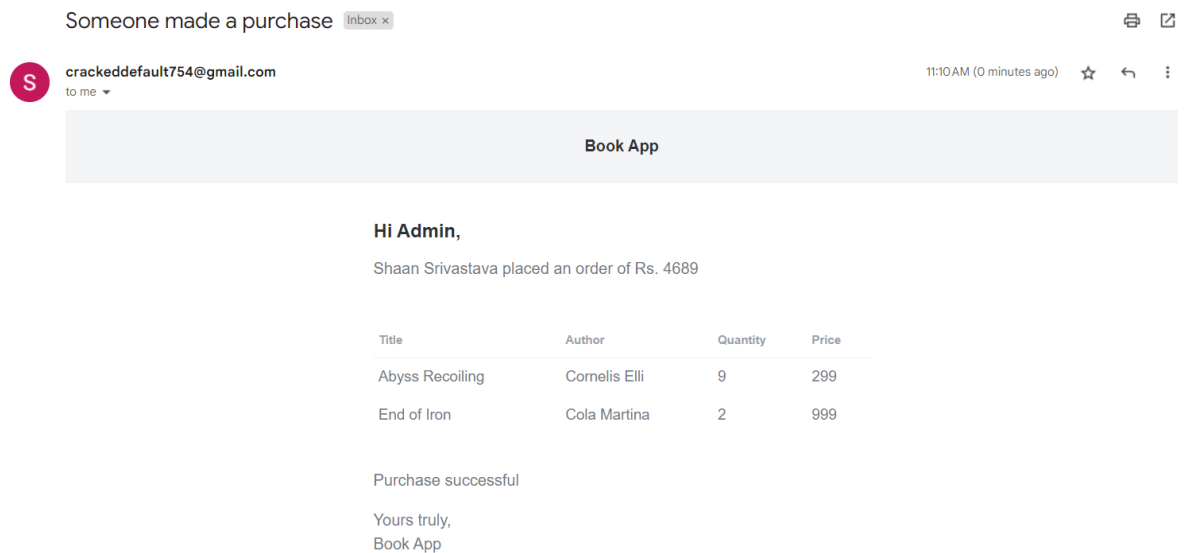
**Fig 3.9: Search Functionality**

**3) Book Details Page:** When a user clicks on the row of a book then he will get the details of the book. After going on the details page, the admin can edit the book details. And also a normal user can add the book to the cart.



**Fig 3.10: Book Details Page**

**4) Cart Page:** After clicking on add to cart the user can manage the quantities of books on the cart page. He can proceed to checkout after deciding which books to purchase. He and the admin both will receive an email with the description of the shopping cart.



**Fig 3.11: Email sent to the admin**

**This Message Is From an Untrusted Sender**  
You have not previously corresponded with this sender.

Book App

Hi Shaan Srivastava,

Your Order Placed

Title	Author	Quantity	Price
Abyss Recoiling	Cornelis Elli	9	299
End of Iron	Cola Martina	2	999

Thank you for the purchase. Total price: Rs. 4689. You will receive your books shortly.

Yours truly,  
Book App

**Fig 3.12: Email sent to the user**

**5) Edit Book Page:** This page can be accessed only by the admin where he can edit the details of a book.

### Edit Details

Title	Author
<input type="text" value="Mists of Artemis"/>	<input type="text" value="Jozsi Wulfnoi"/>
Price	Sale Price
<input type="text" value="1899"/>	<input type="text" value="1199"/>
Quantity	Status
<input type="text" value="10"/>	<input type="text" value="Available"/>
Description	
<input type="text" value="Hopes and dreams were dashed that day. It should have been expected, but it still came as a shock. The warning signs had been ignored in favor of the possibility, however remote, that it"/>	
<input type="button" value="Save Book"/>	

**Fig 3.13: Edit Details of Book**



6) **Users List Page:** This page is visible only to admin.

**Users**

NAME	USERNAME	EMAIL	NUMBER OF DONATIONS
Pratham Bhardwaj	pratham0410	pratham0410@gmail.com	2
Shaan Srivastava	admin2023	admin2023@gmail.com	0
Garv Srivastava	garv1830	garv1830@gmail.com	2
Shaan Srivastava	shaan754	shaan754@gmail.com	2
Akshit	akshit1234	akshit123@gmail.com	0
Kunal Bhandari	kunalbhandari	kunalbhandari@gmail.com	2
Narendra Biceps	narendrabiceps	narendrabiceps@gmail.com	2
Shaan Srivastava	crackeddefault754@gmail.com	crackeddefault754@gmail.com	0
Keshav	keshav123	keshav123@gmail.com	0
Pranav	pranav123	pranav123@gmail.com	0
Temporary	temporary	temporary@gmail.com	0

**Fig 3.14: Users List**

## **CHAPTER – 4**

### **PERFORMANCE ANALYSIS**

Performance study is necessary before developing any website, even one that sells books on the MERN stack. Performance analysis is the measurement and examination of several factors affecting the quickness, responsiveness, and dependability of the website. By doing a performance study, developers may discover and address potential issues that might be harming the website's functionality and user experience.

There are many different methods and approaches that may be used to do performance analysis on an e-commerce website created on the MERN platform. One of the most well-liked methods is the use of performance monitoring tools like Google Analytics, New Relic, or AppDynamics. In order to provide information on how effectively a website is doing, these tools may monitor and evaluate a range of data, such as user behavior, server response times, and page load times.

Another method for doing performance analysis is to use load testing software like JMeter or LoadRunner. The purpose of load testing is to assess how well a website performs when it receives a lot of traffic by simulating a lot of concurrent users. By performing load testing, developers may identify and address any performance bottlenecks and scalability issues that may have an impact on the website's performance and user experience.

Developers can employ a range of performance optimisation techniques in addition to monitoring and load testing technologies to improve the website's speed and responsiveness. One such tactic is to optimize images and other media assets used on the website to reduce their size and quicken load times.

Developers can also improve the website's code and database queries to increase its effectiveness and responsiveness. Reduced HTTP requests, database searches, and website CSS and JavaScript file optimisations can all contribute to this.

Developers may employ a number of performance monitoring and load testing tools, as well as performance optimisation techniques, to provide the best user experience possible. Furthermore, it's essential to do ongoing performance optimisation and analysis in order to address any potential issues that may arise over time and make sure the website stays performing at its peak.

One of the initial tasks in performance analysis is to carry out load testing to evaluate how the website performs under heavy traffic. When a website is being tested under load, several simulated users can visit it at once, and the server's capacity, resource use, and response time are all being monitored. Using load testing software like JMeter and LoadRunner, thorough performance data may be generated by automating this operation.

Database optimization is another important aspect of performance analysis. This comprises optimizing the database structure, queries, and indexing to ensure that operations for data retrieval and manipulation are efficient and effective. Use tools like Robo 3T and MongoDB Compass to analyze and enhance database performance.

In addition to the aforementioned, putting in place a Content Delivery Network (CDN) may help websites function much better by caching commonly used resources and lowering server load times. The website can integrate CDN service providers like Cloudflare and Amazon CloudFront to offer this feature.

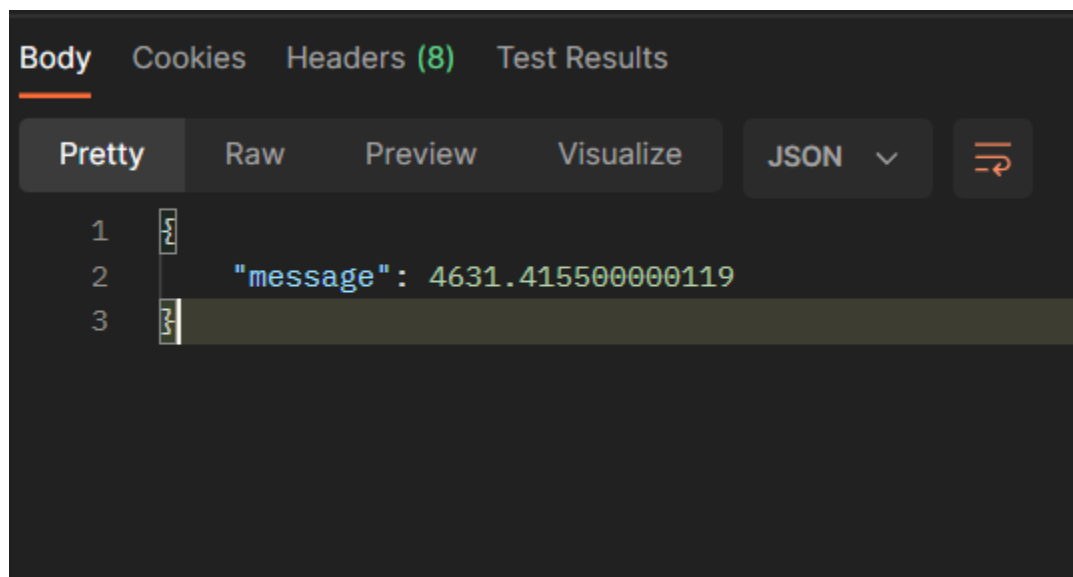
## Response Time of the Website:

A built-in Node.js module called `perf_hooks` offers a means to gauge how well Node.js applications are performing. It is employed to gauge how long it takes for the functions in the code to run.

Developers may detect bottlenecks and improve the speed of their code by measuring the time taken by a particular block of code or function using `perf_hooks`.

## Code:

```
router.get("/performance/getPerformance", (req, res)=>{
  const t1 = performance.now();
  const responseTime = t1 - t0;
  res.status(201).json({message:responseTime});
});|
```



**Fig. 4.1: Output of the response time API**

## React Hooks:

Thanks to hooks, stateful behavior, like processing form input, managing a component's state, and obtaining data from an API, may be reused. Some of the often used Hooks include `useState()`, `useEffect()`, `useContext()`, `useReducer()`, and `useCallback()`.

The `useEffect()` hook allows developers to conduct side effects based on changes to the state or props of the component, such as gathering data from an API or altering the DOM, whereas the `useState()` hook allows developers to add state to functional components. The `useContext()` hook enables developers to consume data from the React context, but the `useReducer()` hook is equivalent to the `useState()` hook and gives a more powerful approach of handling complex state changes.

### Hooks in the project:

- **useState:** Functional components can have state thanks to the `useState` hook that comes with React. Functional components were stateless prior to the addition of hooks in React 16.8 because they were unable to save or modify state. Thanks to `useState`, developers may now use state in functional components instead of a class-based component.

```
// States for form fields
const [title, setTitle] = useState("");
const [author, setAuthor] = useState("");
const [price, setPrice] = useState("");
const [sale_price, setSalePrice] = useState("");
const [description, setDescription] = useState("");
const [quantity, setQuantity] = useState("");
const [disabled, setDisabled] = useState(true);
```

The array's first item is the state value that is in effect right now, and its second value is a function that updates the state value. A beginning state can be provided as an input to the useState hook. The useState hook can be used repeatedly in a component to manage multiple state components.

- **useNavigate:** The useNavigate hook in React Router makes it possible to navigate within React components without generating a Redirect component or manually changing the URL. It provides a way to go between pages in a React application by changing the URL and creating the relevant component.

The useNavigate hook returns a navigate function that may be used to browse to another URL. A string argument is used to specify the URL to go to. The function modifies the URL and renders the corresponding component again. [8]

- **useEffect:** It is a Hook in React that enables the execution of side effects in function components. Examples include getting data from an API, modifying the DOM, setting timers, and more. [8]

Unless it is defined to only run once, the useEffect Hook is called after each render of a component, including the first render. A function that executes the side effect and an optional array of dependents that specify when the effect should be rerun are its two required inputs. [7]

```

const fetchHandler = async () => {
  return await axios
    .post(url, {
      userId: JSON.parse(localStorage.getItem("user"))._id,
    })
    .then((res) => res.data);
};

useEffect(() => {
  fetchHandler().then((data) => {
    setBooks(data.books);
    setTableBooks(data.books);
  });
}, []);

```

The cleaning function can be returned by the effect function, which is executed before the effect is rerun or after the component is unmounted. This function is supplied as the first parameter.

**Security:** Some pages in the project can be accessed by admin only and some of them can be accessed by normal users. Data of the user is being saved in the Local Storage after logging the user into the website.

**Localstorage:** A client-side web storage feature in HTML5 called LocalStorage enables web applications to store key-value pairs in the browser without a time limit. It offers a straightforward method to quickly save data locally on the user's device and access it later. [4]

```

> localStorage
< Storage {user: '{"_id":"6454dab770f84f62e867d9b9","name":"Shaan Sr...hjC06Xats.DKEvBvxS
z0F/EsC","role":"User","_v":0}', length: 1}
> |

```

**Fig 4.2: LocalStorage saves the data**

Similar to cookies, but with larger storage space and no expiration date, is local storage. For web applications that often store and receive tiny quantities of data, LocalStorage is a helpful technology because, unlike cookies, it is not communicated to the server with each HTTP request.

Concept of Private Routes is used in the React for protecting the main pages from unauthenticated users.

Developers may save, retrieve, and remove data using the LocalStorage API. Key-value pairs are used to store data; a string is used as the key and any JavaScript object may be used as the value. The LocalStorage API provides methods for setting, retrieving, and deleting data, such as `setItem()`, `getItem()`, and `removeItem()`.

Modern online applications frequently use LocalStorage to store user preferences, browser settings, and other information that must endure between browser sessions. The efficiency and user experience of online apps can be enhanced by using this straightforward and dependable method of storing tiny quantities of data locally in the browser.

### **React Router:**

A potent library for creating React single-page apps (SPAs), React Router. It gives developers a declarative approach to build and maintain an application's routing by letting them associate particular routes with various components that can be displayed depending on the current URL. [8]

For handling various routing scenarios, React Router provides a number of components, including `BrowserRouter` for client-side routing, `HashRouter` for server-side routing, and `MemoryRouter` for testing. Additionally, it offers



Switch components that only render the first matching route and Route components that specify which component to render for a particular path. [8]

Along with these helpful features, React Router also provides nested routes, route parameters, and route guards. While route parameters enable dynamic routing depending on parameters supplied in the URL, nested routes enable the building of complicated routing structures. Route guards offer a mechanism to extend authentication and permission to certain routes, guaranteeing that only authorized users may access particular pages.

Overall, React Router makes it easier to manage routing in a React application, freeing developers from having to think about how to handle various URLs and routes in order to concentrate on designing the application logic.

**For normal users, pages that can be accessed by a logged in users only:**

```
const PrivateRoutes = () => {  
  let flag;  
  if(localStorage.getItem("user")===null) {  
    flag = false;  
  } else {  
    flag = true;  
  }  
  return (  
    flag ? <Outlet/> : <Navigate to="/login"/>  
  )  
}
```

For unauthenticated users, pages that can be accessed only by users not logged in:

```
const ProtectLogin = () => {
  let flag;
  if(localStorage.getItem("user")===null){
    flag = true;
  } else {
    flag = false;
  }
  return (
    flag ? <Outlet/> : <Navigate to="/books"/>
  )
}
```

Pages that can only be accessed by admin:

```
const ProtectAdmin = () => {
  let flag;

  localStorage.getItem("user")?JSON.parse(localStorage.getItem("user"))[
  "role"]==='Admin'?flag=true:flag=false:flag=false;

  return (
    flag ? <Outlet/> : <Navigate to="/books"/>
  )
}
```

## **CHAPTER 5**

### **CONCLUSION**

Working on our project might be a rewarding and meaningful experience for engineers and users alike. The MERN stack, made up of Node.js, Express.js, React.js and Mongoose is a robust and flexible framework for creating a solid e-commerce website since each technology in the stack adds to its own unique qualities.

MongoDB is an excellent choice for managing enormous volumes of data since it is a powerful NoSQL database with good performance and scalability. Its adaptability allows developers to easily add, amend, and remove data without worrying about the restrictions of a predefined schema. This suggests that an e-commerce website's product data may be swiftly added or changed as necessary, providing users with a seamless experience.

The backend of the e-commerce website is created using Express, a straightforward and adaptable web application framework for Node.js. Express has a number of features that make it simple for developers to manage complicated requests and answers, including middleware support and sophisticated routing. Additionally, it makes managing the website's data simpler by allowing developers to connect to MongoDB quickly.

The front-end of the e-commerce website is created using React, a strong and well-liked JavaScript toolkit for creating user interfaces. React is incredibly quick and effective thanks to its virtual DOM, which gives consumers a seamless experience. Additionally, it provides reusable and modular components, making it simple for developers to design a uniform and user-friendly user experience for the website.

The whole online store is powered using Node.js, a server-side JavaScript engine. Its quick and effective event-driven design enables developers to create scalable and high-performance applications. Additionally, it features a sizable and vibrant community that makes it simple for developers to obtain help and resources.

It is crucial to keep in mind the numerous factors and difficulties involved in creating a safe, effective, and user-friendly platform while constructing an e-commerce website using the MERN stack.

User experience (UX) has a direct influence on customer happiness, retention, and eventually revenue, making it essential for the success of any e-commerce company. Consumers have high expectations for their online buying experiences in today's cutthroat market, and if a website doesn't live up to those standards, customers will probably move on to a rival.

The functioning and design of websites is one of the key elements influencing UX. The chance of a sale will rise if a website is well-designed, simple to use, and offers a smooth user experience. This will attract visitors to remain longer and browse more goods. On the other side, users may leave a website if it is complicated or challenging to use.

Speed and performance are additional key components of UX. A website that loads slowly or performs badly might annoy visitors and give them a bad image of the company. According to study, a one-second delay in the time it takes for a page to load can significantly lower conversion rates. Therefore, boosting website performance and speed is essential for raising user experience.

Personalization may also improve user experience on e-commerce platforms. E-commerce websites may provide users a more interesting and relevant experience by providing personalized suggestions and targeted marketing

messages based on their browsing and buying history. Increased consumer loyalty and repeat business may result from this.

When creating an e-commerce website, security is an important consideration. Since the website will handle sensitive user data, including personal and financial information, it is essential to ensure that it is secure from several types of attacks. One method to ensure website security is JWT authentication and authorization. JWT, or JSON Web Token, is a standard for exchanging data securely between parties. It facilitates user authentication and ensures that only individuals with authorization may access the website's content.

When creating an e-commerce website, performance is an important consideration. Customers expect the website to load quickly and without latency.

One method to speed up a website is to use a caching system, which may save frequently requested information in memory and reduce the time it takes to get information from the database. Using extensive testing techniques and continuous integration and delivery can further enhance the performance of the website.

The MERN stack enables seamless integration of the front-end and back-end components of the website, simplifying the development of a user-friendly, responsive website that provides a good user experience. The project requires in-depth planning and analysis, including user requirements analysis, market research, and competition analysis, to ensure that the website fits the needs and expectations of the target audience.

The website is safe and user-friendly thanks to the integration of several features, including authentication, authorization utilizing JWT, Passport.js

respectively. The website's user interface is both aesthetically beautiful and responsive thanks to the use of SCSS and React-Bootstrap.

In conclusion, throughout the development process, the user experience must be given high importance. The creation of an easy-to-use interface, the provision of succinct and concise product descriptions and images, and the implementation of a powerful search and filtering system may all considerably improve the user experience. Furthermore, providing quick and safe payment options may help you win the trust and respect of your audience.

Additionally, throughout the development process, the user experience must be given high consideration. By following design principles and best practices, developers can create a website that is easy for visitors to use and navigate. Examples of these principles and best practices include developing a straightforward and intuitive interface, providing clear and concise product descriptions and photos, and setting up a powerful search and filtering system.

At all times, it's important to keep in mind the various factors and difficulties that go into developing an e-commerce website, such as security, performance, user experience, and scalability. Developers can make sure their website is safe and operates at its best by putting strict testing methods in place, following best practices, and utilizing technologies like JWT for authentication and authorisation as well as Bcrypt for password hashing.

So, the MERN stack combines flexibility and scalability, giving it the ideal foundation for building a book-based e-commerce website. By putting best practices into practice and giving the user experience first importance, developers may create a stable and user-friendly platform that can satisfy the needs of both sellers and buyers in the book market.

## **REFERENCES**

- [1] Sanchit Agarwal, Jyoti Verma, “Comparative Analysis of MEAN Stack and MERN Stack”, International Journal of Recent Research Aspects, March 2018.
- [2] Mohanish Bawane , Ishali Gawande , Vaishnavi Joshi , Rujuta Nikam , Prof. Sudesh A. Bachwani, “A Review on Technologies used in MERN stack”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Jan 2022.
- [3] Osama Mohammed, Ahmad Rababah and Fawaz Ahmad Masoud, “Key Factors for Developing a Successful E-commerce Website”, IBIMA Publishing, December 2010.
- [4] David Gillman, Yin Lin, Bruce Maggs, Ramesh K. Sitaraman, “Protecting Websites from Attack with Secure Delivery Networks”, IEEE, April 2015.
- [5] Timothy Kelley, Bennett I. Bertenthal, “Real-World Decision Making: Logging Into Secure vs. Insecure Websites”, Springer, June 2016.
- [6] Tien Pham, “Building an online shop application with MERN stack”, Bachelor’s Thesis, November 2020.
- [7] MongoDB official website for understanding what is MERN stack: <https://www.mongodb.com/mern-stack> [Access Date: 10-02-23]
- [8] Official Documentation for ReactJS: <https://react.dev/learn> [Access Date: 13-02-23]
- [9] NodeJS official documentation: <https://nodejs.org/en/docs> [Access Date: 15-02-23]

[10] ExpressJS official documentation: <https://expressjs.com/> [Access Date: 24-02-23]

[11] Getting started with Mongoose: <https://mongoosejs.com/> [Access Date: 06-03-23]

[12] Best Website practices for designing the User Interface of the website: <https://blog.hubspot.com/blog/tabid/6307/bid/30557/6-guidelines-for-exceptional-website-design-and-usability.aspx> [Access Date: 15-03-23]

[13] Understanding the basic difference between CSS and SCSS:<https://sass-lang.com/> [Access Date: 19-03-23]

[14] Passport JS documentation for Google Login: <https://www.passportjs.org/packages/passport-auth0/> [Access Date: 19-03-23]

[15] Using CORS for secure connection between frontend and backend: <https://developer.mozilla.org/en-US/docs/Glossary/CORS> [Access Date: 04-04-23]

[16] Nodemailer official documentation: <https://nodemailer.com/about/> [Access Date: 26-04-23]



## major project report

### ORIGINALITY REPORT

<b>2%</b>	<b>1%</b>	<b>0%</b>	<b>1%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>Submitted to HELP UNIVERSITY</b> Student Paper	<b>&lt;1%</b>
<b>2</b>	<b>Submitted to London School of Science &amp; Technology</b> Student Paper	<b>&lt;1%</b>
<b>3</b>	<b>Submitted to Western International College (WINC London)</b> Student Paper	<b>&lt;1%</b>
<b>4</b>	<b>noexperiencenecessarybook.com</b> Internet Source	<b>&lt;1%</b>
<b>5</b>	<b>Submitted to University of Bedfordshire</b> Student Paper	<b>&lt;1%</b>
<b>6</b>	<b>Submitted to University of Ulster</b> Student Paper	<b>&lt;1%</b>
<b>7</b>	<b>Submitted to CSU, San Jose State University</b> Student Paper	<b>&lt;1%</b>
<b>8</b>	<b>www.researchgate.net</b> Internet Source	<b>&lt;1%</b>
<b>9</b>	<b>Submitted to University of Greenwich</b>	

Student Paper

<1 %

---

10 Osama Rababah, Fawaz Masoud. "Key Factors for Developing a Successful E-commerce Website", Communications of the IBIMA, 2010  
Publication

<1 %

---

11 [gitlab.stud.idi.ntnu.no](http://gitlab.stud.idi.ntnu.no)  
Internet Source

<1 %

---

12 [intellipaat.com](http://intellipaat.com)  
Internet Source

<1 %

---

13 [kupdf.net](http://kupdf.net)  
Internet Source

<1 %

---

14 [www.spec-india.com](http://www.spec-india.com)  
Internet Source

<1 %

---

15 [www.upgrad.com](http://www.upgrad.com)  
Internet Source

<1 %

---

16 Practical Node js, 2014.  
Publication

<1 %

---