“ECOMMERCE WEB APPLICATION”

A Major-Project report submitted for 7th Semester B. Tech Course requirements

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

MAJOR-PROJECT REPORT

Submitted by

Hirak Jyoti Das - 200710007024
Partha Pratim Das-200710007039
Shoaib Alom-200710007052
Hirak Jyoti Kakati-200710007063

Under the guidance Of
Mrs. Monmayuri Baruah
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JORHAT ENGINEERING COLLEGE, JORHAT
ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY, GUWAHATI

ACKNOWLEDGEMENT

We would like to express our sincere gratitude and heartfelt appreciation to Assistant Professor Mrs. Monmayuri Baruah ma'am for her invaluable guidance and support throughout the duration of our major project. Her expertise, enthusiasm, and dedication have been instrumental in shaping our understanding of the subject matter and enhancing our learning experience.

We are extremely fortunate to have had the opportunity to work under her mentorship. Mrs. Monmayuri Baruah ma'am's profound knowledge, keen insights, and unwavering commitment to academic excellence have been a constant source of inspiration for us. She has consistently encouraged us to push our boundaries, think critically, and explore innovative solutions to complex problems.

Her guidance and feedback were invaluable in refining our project objectives, methodology, and implementation. She provided constructive criticism, offered suggestions for improvement, and motivated us to strive for excellence in every aspect of our work. Ma'am's approachable nature, patience, and willingness to address our queries have made our learning experience both enjoyable and enriching.

In conclusion, we extend our heartfelt thanks to Assistant Professor Mrs. Monmayuri Baruah ma'am for her invaluable guidance, mentorship, and support. We are truly grateful for the knowledge and skills we have gained under her tutelage. Her impact on our academic journey will always be cherished and remembered.

Thank you.

PROJECT STUDENTS:

Hirak Jyoti Das (200710007024)

Shoaib Alom (200710007052)

Partha Pratim Das (200710007039)

Hirak Jyoti Kakati (200710007063)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JORHAT ENGINEERING COLLEGE, JORHAT
ASSAM SCIENCE AND TECHNOLOGY UNIVERSITY: GUWAHATI

CERTIFICATE

This is to certify that the project report titled **"Ecommerce Web Application"** has been successfully completed by **Hirak Jyoti Das (200710007024)**, **Shoaib Alom (200710007052), and Partha Pratim Das (200710007039)** of 6th Semester, **Bachelor of Technology in Computer Science and Engineering** at Jorhat Engineering College, Jorhat. This report is a testament to their genuine and diligent efforts, conducted under my guidance and supervision during the session 2022-23.

Project Guide

Mrs. Monmayuri Baruah
ASSISTANT PROFESSOR
Department of CSE
Jorhat Engineering College

Head of the Department

Mr. Diganta Baishya
Department of CSE
Jorhat Engineering College

DECLARATION

We hereby declare that the work is being presented in the report entitled **"Ecommerce Web Application"** is an authentic record of our own work carried out during the August 2023 to November 2023, for the fulfilment of the 7th Semester course requirements, under the guidance of Mrs. Monmayuri Baruah and Prof. Diganta Baishya of Department of Computer Science and Engineering, Jorhat Engineering College, Jorhat – 785007 (Assam) and has not been submitted to any other university for the award of any kind of degree.

Shoaib Alom (200710007052)

Hirak Jyoti Das (200710007024)

Partha Pratim Das (200710007039)

Hirak Jyoti Kakati (200710007063)

ABSTRACT

This project introduces a customer-centric e-commerce website developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack, specifically tailored for the electronic device market. The application prioritizes user engagement, offering a diverse product catalog featuring laptops, mobiles, headphones, and speakers. Key functionalities include user authentication, wishlisting , and cart management to optimize the user experience.

Key Features:

1. **Product Catalog:** The application boasts a user-friendly product catalog, allowing visitors to explore a variety of electronic devices. Each product includes detailed specifications, images, and pricing information.

2.**User Authentication and Authorization:** A secure authentication system allows users to register, log in, and access personalized features. Only registered users can log in, wishlist products, and manage their shopping cart. Authorization mechanisms ensure that users can only view and modify their own data.

3.**Wishlist Feature:** Registered users can create wishlists , curating a personalized collection of desired products. This feature enhances user engagement, allowing them to revisit and manage their wishlist over multiple sessions.

4.**Shopping Cart Functionality:** Users can add products to their shopping cart for future consideration. The cart management system enables users to review their selections and adjust quantities.

5.**Responsive Design:** The website is designed to be responsive across devices, ensuring a seamless experience on desktops, tablets, and mobile phones.

6.**User Profile:** Registered users have access to a profile section where they can view and manage their wishlisted products, enhancing personalization and convenience.

7.**Add to Cart Functionality:** Users can add products to their cart for potential future purchases. However, as of the current implementation, the project does not include a checkout process or payment integration.

The successful implementation of this customer-centric e-commerce website using the MERN stack underscores its ability to create a responsive, secure, and feature-rich platform. The focus on user authentication, wish listing, and cart management reflects a commitment to enhancing the overall user experience in the context of electronic device shopping. Future iterations may include the addition of checkout functionality and payment integration to further elevate the application's capabilities.

Content

List of Abbreviations:

DOM Document Object Model

W3C World Wide Web Consortium

ISO International Organization for Standardization

NPM Node Package Manager

API Application Program Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

URL Uniform Resource Locator

JSON JavaScript Object Notation

JS JavaScript

JSX JavaScript

XML Extensive Markup Language

INTRODUCTION

It is true that technology has become an essential tool for online marketing nowadays. However, there are numerous small shops and grocery stores with mostly offline business model in Vietnam recently. With this commerce model, it will bring a lot of bad experiences for both buyers and sellers. For instance, the seller has the product want to offer but the buyer may not know it, or the buyer may urgently need to purchase something, but the store is out of stock. Moreover, online shopping helps customers to choose a wide range of products, prices and they can compare them to each other easily.

Encountering the inadequacies and the weaknesses of the offline business model, making a website application for searching and buying things for each shop is very necessary right now. Recently, there have been many e-commerce sites exported such as Amazon, e-bay or the stores that can sell products via social media channels like Facebook. However, customers still find it difficult to choose the products they want because of the large variety of products on these sites and not focus on specific things. Moreover, the sellers have to spend a high amount of money on marketing or paying for fees. From there disadvantages, implement an online e-commerce web application for small grocery stores helps retailers can manage products on their own systems and not depend on the 3rd party website. For the customers, they can quickly search the products if it is available and come to store to pick it up and they can contact directly to the shop owner to learn more about the products that they are looking for.

In order to make a website that can acquire the needs of both customers and retailers, MERN (MongoDB, Express.js framework, ReactJS library, NodeJS platform) is one of the powerful stacks that can help us to develop an e-commerce web application.

2 E-commerce

2.1 Definition

E-comm, EC for short (E-commerce) is a concept referring to transactions, purchase and sale of goods and services by the internet. E-commerce was first known in the 1960s. After years of development, as mobile devices became popular, social media increasingly affirmed the power and the boom of the webpage. Launchers promote the rapid development of commerce (E-commerce).[1]

2.2 Types

Currently, there are many forms of e-commerce, including the following basic forms:

B2B (Business to Business): is a trade between companies, businesses and organizations. About 80% of e-commerce today falls into this category.

B2C (Business to Consumer): is an Internet-based business to directly exchange the goods and services it creates or distributes to consumers.

C2B (Consumer to Business): is a consumer who sells their products or services to a business or organization.

C2C (Consumer to Consumer): is when a consumer sells his goods or services to another consumer.
There are also G2C, G2B, etc., but used less often than these four basic forms.

2.3 Advantages

Global market: Clearly, when you open a physical store, you will only be able to deliver your goods and services in a small geographic area. E-commerce will help you solve that problem. E-commerce helps you reach the market quickly, expanding the market to the maximum level compared to direct sales, so that products and services are easily introduced, purchased and sold through retailers. and online market.

Always open: In e-commerce, running an online business is much easier, it's always open 24h / 7/365. For businesses, it's a great opportunity to increase sales opportunities all the time.

Budget savings: Compared with traditional forms of commercial business, all costs when e-commerce business are reduced: the cost of renting booths, salespeople and management is much more economical. Naturally, when sellers save operating costs, they can offer more incentives and better discounts for their customers. At this time, the customer is the next beneficiary. Mutual benefit, isn't it great?

Inventory management: By using electronic tools to speed up the ordering, delivery, and payment processes, e-commerce businesses can save billions of operating costs and reduce amount of inventory.

Most accurate customer marketing: With access to customer data and the opportunity to track customers' buying habits, e-commerce businesses can quickly identify and market products and services. service. Service most suitable for consumers.

Work anywhere, buy anywhere: Running an e-commerce business allows you to not need to sit in the office, and buying does not force you to go to the supermarket. Everything the seller and the buyer needs is an internet-connected device and that's all.[2]

2.4 Challenges

Internet access required: When participating in the EC, to be able to buy and sell, you need a device connected to the internet. Currently, most people have internet access but, in many areas, it is still very limited.

Not enough to trust: Products and services that cannot be seen, touched, held or felt directly, are not allowed to try as a prudent buyer. Doubt in both buyers and sellers leads to many incomplete transactions, especially when they have dealt with untrusted partners before.

Limited payment methods: Currently, the most popular payment method in Vietnam when buying goods online is to receive and pay. Payment gateway in Vietnam is growing quite strong, but not reliable enough for users to use as the main payment method. Therefore, it also contributes to teething.

In addition, e-commerce business also faces many other challenges: technical, competitors, payment, etc.

3) MERN Stack

3.1 JavaScript

JavaScript is a scripting, object-oriented, cross-platform programming language. Objects of host environment can be connected to JavaScript and arrange ways to operate them. Standard libraries for objects are contained by JavaScript, for such as Array, Date, Math, and the essence component of programming languages for instance managers, control framework and statements. By adding objects, JavaScript could be protracted for many principles, such as:

Client-side JavaScript: JavaScript is developed by implementing objects for controlling the browser and DOM. For instance, an application is granted by client-side extensions to influence components on an HTML page and answer to user behavior like mouse hovers, form input and page changeover.

Server-side JavaScript: JavaScript is developed by implementing the supplementary objects required to run JavaScript on the server. For instance, an application is granted by this server-side extension to connect to a database, transfer data frequently from one request to other section of the application or execute application with another function file on the server.

In 1996, JavaScript was officially named ECMAScript. ECMAScript 2 was released in 1998 and ECMAScript 3 was released in 1999. It is continuously evolving into today's JavaScript, now works on all browsers and devices from mobile to desktop. Open standard language can be used by association to establish their own JavaScript applications. The ECMAScript Standard is one of the parts of the ECMA-262 specification. ISO has approved the ECMA-262 standard at ISO-16262. The ECMAScript standard does not include descriptions for the DOM, it is standardized by the W3C. The DOM specifies how your scripts display HTML objects. To get a advance anticipate of the distinctive innovations used when programming with JavaScript, check out the JavaScript technology analysis article. [3]

3.2 NodeJS

Node.js is an open source, a system application and furthermore is an environment for servers. Nodejs is an independent development platform built on Chrome's JavaScript Runtime that we can build network applications quickly and easily. Google V8 JavaScript engine is used by Node.js to execute code. Moreover, a huge proportion of essential modules are written in JavaScriptNode.js accommodate a built-in library which allows applications to serve as a Webserver left out demanding software like Apache HTTP Server, Nginx or IIS. An event-driven, non-blocking I / O mechanisms (Input / Output) are implemented by Node.js. It optimizes application throughout and is extremely high extensible. Node.js use asynchronous in it functions. Therefore, Node.js processes and executes all tasks in the background (background processing). products that have a lot of traffic are applying Node.js. Nonetheless, Node.js handle the application that need to spread expeditiously, develop innovation, or build Startup projects as rapidly as possible.[4]

Applications using NodeJS:

• WebSocket server
• Notification system
• Applications that need to upload files on the client.
• Other real-time data applications.

NodeJS Pros:

• Node.js is the exclusive application that with only a single thread, it can obtain and handle numerous connections. Building new threads for each query is not needed, therefore the structure expends the least amount of RAM and run rapidly. Secondly, Node.js produces the most of server property without generate latency with the JavaScript’s non-blocking I/O.

• **JSON APIs**. JSON Web services can take advantages of that because of the event-driven, non-blocking I/O structures and JavaScript-enabled model.

• **Single page application**. NodeJS is very suitable with an application on a single page. Node.js has the capability to handle different requests concurrent and quick return. Node JS should be used in an application that does not have to reload the page, including users who makes a vast number of requests and need a quick procedure to show professionalism.

• **Shelling tools Unix**. Node.js usually uses Unix to work. They can handle multiple processes and return them for best performance. Programmers often use Node.js to build real Web applications like chat, feeds, etc.

• **Streaming Data**. Typical websites send HTTP requests and also receive responses. Node.js can handle many questions and feedback, so they are suitable if the developer wants to create an application on the page. In addition, Node.js also builds proxies to stream the data, this is to ensure maximum operation for other data streams.

• **Real-time Web Application**. Node.js is sufficient to develop real-time innovations like chat apps, social networking services like Facebook, Twitter because of the opening of mobile application.

NodeJS Cons:

• **Resource-intensive applications**. Node.js is written in C ++ & JavaScript, so when programmers need to handle applications that use a lot of file conversion, video encoding, decoding, etc., they should not be used Node.js. Programmers need to use it more carefully in this case.

• The final purpose of NodeJS is like other programming languages such as Ruby, PHP, .NET, Python, that is developing web application. Therefore, do not expect NodeJS to outperform other language for now. But with NodeJS the application can be developed successfully as expected. [5]

NodeJS should not be used when:

• **Build resource-intensive applications**: Do not use Node.js when creating a video converter application. Node.js often comes down to bottlenecks when working with large files.

• **An all-CRUD-only application**: Node.js is not faster than PHP when doing heavy I/O tasks. In addition, with the long-term stability of other webserver scripts, its CRUD tasks have been optimized. Node.js will come up with odd APIs and

never be used

• **Stability in the application**: Within 11 years of development (2009-2020), the current version of Node.js is already v14.2.0. Every API can be changed – in a way that is not backwards compatible.
• **Lack of knowledge about Node.js**: Node.js is extremely dangerous in this case, you will fall into a world full of difficulties. With most non-blocking/async APIs, not understanding the problem will cause an error that you do not even know where it came from. Moreover, when the Node.js community is not strong enough, and there will be less support from the community.

NodeJS should be used when:

• **Building RESTful API (JSON)**. You can use Node.js in building RESTful API (JSON). They handle JSON very easily, even more than JavaScript. API servers when using Node.js usually do not have to perform heavy processing, but the number of concurrent requests is high.

• **Applications that demand alternative connection protocols**, not just http. With TCP protocol backing, any custom protocol can be built easily.

• **Real-time applications**.

• **Stateful websites**. Every request on the invariable procedure is handled by Node.js, therefore building caching is simpler: store it to a comprehensive variable then all requests can approach the cache. The status of one client can be
saved and shared with other clients and do not have to go through external memory. [6]

3.3 Express.js

Express.js is a framework built on top of Nodejs. It provides powerful features for web or mobile development. Express.js supports HTTP and middleware methods, making the API extremely powerful and easy to use.

Express implements extra features to developer which help them get a better programming environment, not scaling down the speed of NodeJS.
Importantly, the well-known frameworks of NodeJS apply Express.js as a substance
function, for instance: Sails.js, MEAN.[7]

3.4 MongoDB

MongoDB is an open source database; it is also the leading NoSQL (*) database currently used by millions of people. It is written in one of the most popular programming
languages today. In addition, MongoDB is cross-platform data that operates on the concepts of Collections and Documents, providing high performance with high availability
and ease of expansion.[8]
(*) NoSQL is a source database format that does not use Transact-SQL to access information, this database was developed on JavaScript Framework on JSON data type. With
its introduction, it has overcome the disadvantages of RDBMS relational data model to
improve operating speed, functionality, model scalability, cache ...
Furthermore, MongoDB is a cross-platform database, performing on Collection and Document approach, it produces sharp production, huge availability, and effortless scalability.

Commonly used terms in MongoDB:

• **_id:** Almost every document required this field. The _id field illustrates a exceptional value in the MongoDB document. The _id field can also be interpreted as
the primary key in the document. If you add a new document, MongoDB will automatically generate a _id representing that document and be unique in the MongoDB database.
• **Collection:** A group of many documents in MongoDB. Collection can be interpreted as a corresponding table in the RDBMS (Relational Database Management System) database. Collection resides in a single database. Collections do
not have to define columns, rows or data types first.
• **Cursor:** This is a pointer to the outcome set of a query. The client can emphasize
over a cursor to get the result.
• **Database:** The location of the collections, similar to the RDMS database that
contains the tables. Each Database has a separate file stored on physical
memory. Some MongoDB owners may contain various databases.

• **Document:** A transcript belonging to a Collection. Documents, in turn, include name and value fields.

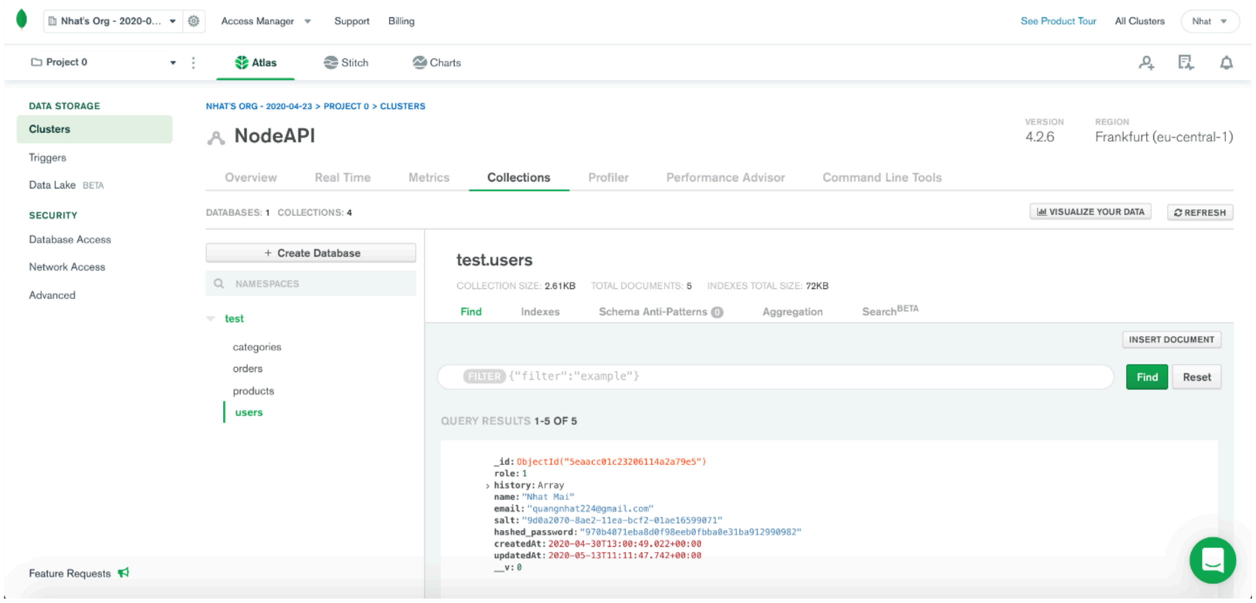
• **Field:** A name-value pair in a document. A document may not need all the fields. The fields are like columns in a relational database.

• **JSON:** Short for JavaScript Object Notation. Human readability is in the plain text format representing structured data. JSON currently supports a lot of programming languages.

• **Index:** Exclusive data structures used to save a small allocation of data sets for simple scanning. The index puts the value of a individual field or sets of fields, sorted by the value of these fields. Index effectively supports the analysis of queries. Without an index, MongoDB will have to scan all the documents of the set to choose the documents that pair the query. This scan is ineffective and requires MongoDB to progress a vast amount of data.

MongoDB Atlas is MongoDB's cloud database launched in 2016 on AWS, Microsoft Azure and Google Cloud Platform.

The data in each Cluster in the Atlas is stored by Replication mechanism, with 3 nodes: 1 master (primary) and 2 slaves (secondary).



```
{"filter":"example"}
```

```
  _id: ObjectId("5eaacc01c23206114a2a79e5")
  role: 1
> history: Array
  name: "Nhat Mai"
  email: "quangnhat224@gmail.com"
  salt: "9d0a2070-8ae2-11ea-bcf2-01ae16599071"
  hashed_password: "970b4071eba8d0f98eeb0fbba0e31ba912990982"
  createdAt: 2020-04-30T13:00:49.022+00:00
  updatedAt: 2020-05-13T11:11:47.742+00:00
  __v: 0
```

Figure 1. MongoDB Atlas screenshot.

3.5 ReactJS

3.5.1 Virtual-DOM

Virtual-DOM is a JavaScript object, each object contains all the information needed to create a DOM, when the data changes it will calculate the change between the object and the real tree, which will help optimize re-render DOM tree. It can be assumed that is a virtual model can handle client data.[9]

3.5.2 Component

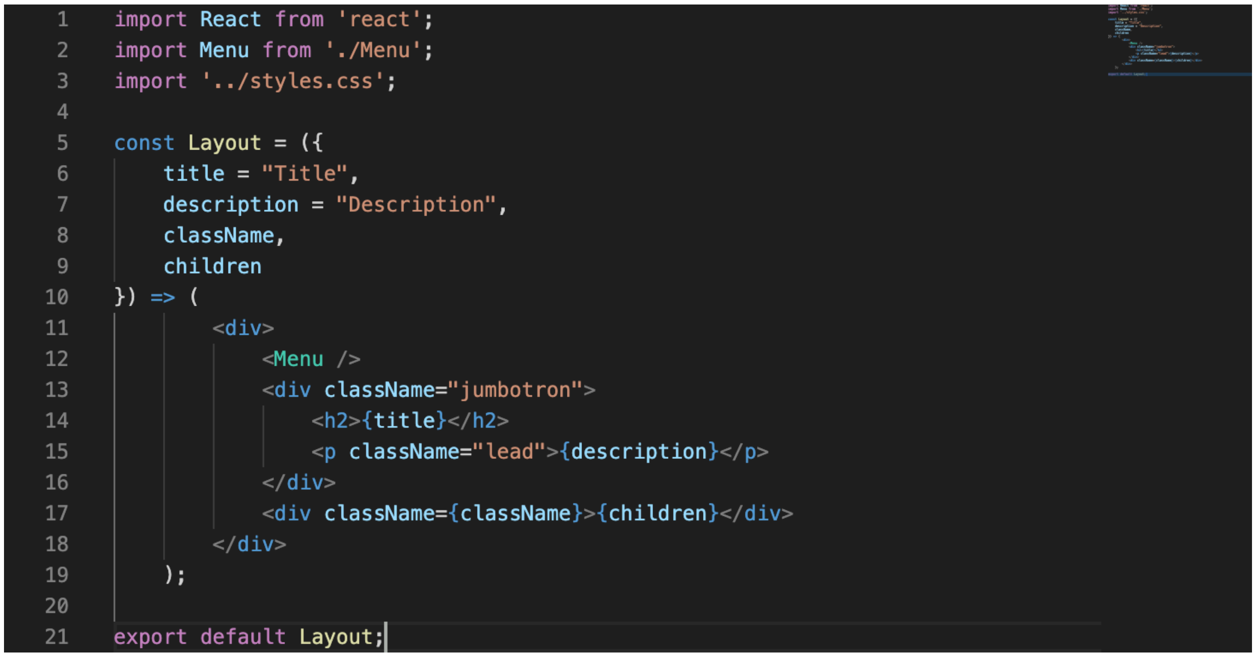
React is built around components, not templates like other frameworks. A component can be created by the create Class function of the React object, the starting point when accessing this library.

ReactJS creates HTML tags unlike we normally write but uses Component to wrap HTML tags into stratified objects to render.
Among React Components, render function is the most important. It is a function that handles the generation of HTML tags as well as a demonstration of the ability to process via Virtual-DOM. Any changes of data at any time will be processed and updated immediately by Virtual-DOM.

[10]

3.5.3 Props and State

Props: are not controlled by Component, actually stands for Properties.



```
import React from 'react';
import Menu from './Menu';
import '../styles.css';

const Layout = ({
    title = "Title",
    description = "Description",
    className,
    children
}) => (
        <div>
            <Menu />
            <div className="jumbotron">
                <h2>{title}</h2>
                <p className="lead">{description}</p>
            </div>
            <div className={className}>{children}</div>
        </div>
    );

export default Layout;
```

Figure 2. Layout component.

The title = “Title” line creates a name attribute with the value "Title". It looks like a function call. It is true that props are passed to the component in the same way that an argument is passed to a function.

A component may also have a default props, so if the component does not pass any props, it will still be set.

State: private data is controlled by Component.

Like props, sate also holds information about the component. However, the type of information and how to handle it varies. State works differently than Props. The state is a component of the component, while props are passed in from the outside into the component. It should be noted that we should not update the state directly using this.state but always use setState to update. Update the state of the objects. Use setState to re renders one component and all its children. This is great, because you don't have to worry about writing event handlers like any other language.[11]

3.5.4 Pros and Cons of ReactJS

Pros of ReactJS:

• Update data changes quickly.

• React is not a framework so it offloads the constraints of libraries together.

• Easy access to who understands JS.

Cons of ReactJS:

• ReactJS only serves the View tier, but the library size is on par with Angular while Angular is a complete framework.

• Incorporating ReactJS within common MVC frameworks demands reconfiguration.

• Hard to reach for beginners on website developing.[12]

3.6 MERN Stack in Website Development

3.6.1 Concept of Stack technology

The technical stack is a combination of technologies / frameworks / programming languages, etc. to create a complete software.

With current software, there are usually two parts: client side and server side, also known as frontend and backend. Therefore, people also split the backend stack, the frontend stack as well. We often use the first letter to name the technical stack: LAMP (Linux, Apache, MySQL, PHP), MEAN (MongoDB, Express, Angular, NodeJS).[13]

3.6.2 Concept of MERN Stack

The MERN stack is a complete open source combination, all JavaScript-related technologies are also the hottest today: MongoDB, Express.js, React / React Native, NodeJS.

People use the MERN stack to build Universal React App.[14]

3.6.3 Highlights in MERN Stack

• Hot Reloading for React Components.

• The code snippets are divided by React Router.

• Multi-language support.

• Generate code support.

• Modular structure.

• Docker support.

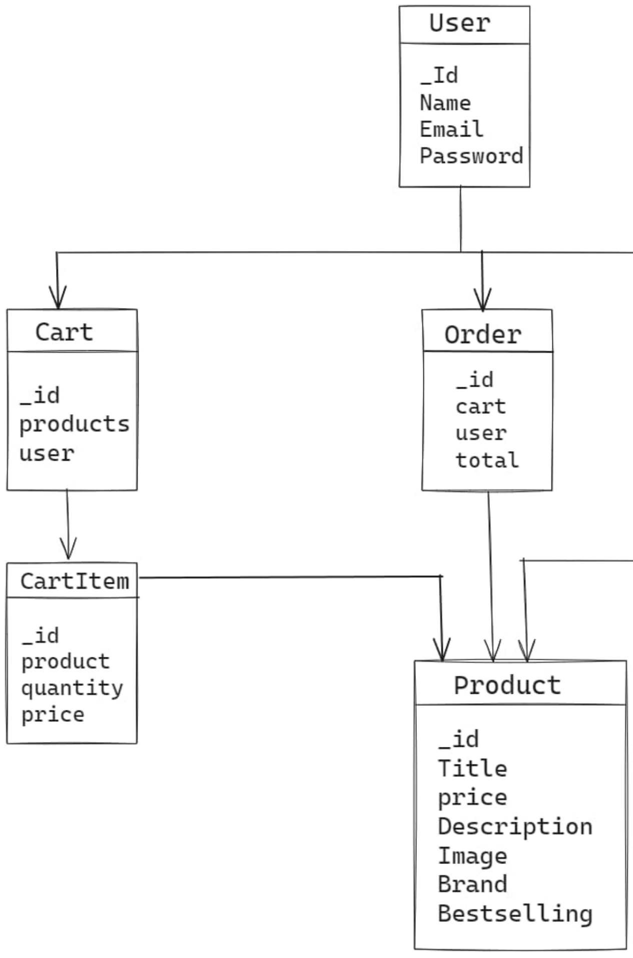
• Can customize the MERN version for individual users. [8]15

4 E-commerce Web Application

It is an E-commerce Web Application using MERN stack that can help customers buy different Products.

Main function:

- Sign up and log in: Requires Users to register using their email
- Shopping cart: this feature helps users buy and check goods directly on the application
- Wish List: This feature allow users to wish list the product they like.
- Search: Users can search directly by typing in the search box for the product they want to see.
- Buy and pay: Customers who buy through the app can pay through many different payment gateways.



4.1 Database Design

User

Field	Type
_id	string
Name	string
Email	string
Password	string

Cart

Field	String
_id	String
Products	Array of CartItem schema
User	Object Id (Ref:`User`)

Order

Field	Type
_id	String
cart	ObjectId(Ref:`Cart`)
user	Object Id(Ref:`User`)
Total	Number

Wishlist

Field	Type
_id	String
Product	Object Id(Ref:`Product`)
User	Object Id(Ref:`User`)
isLiked	Boolean

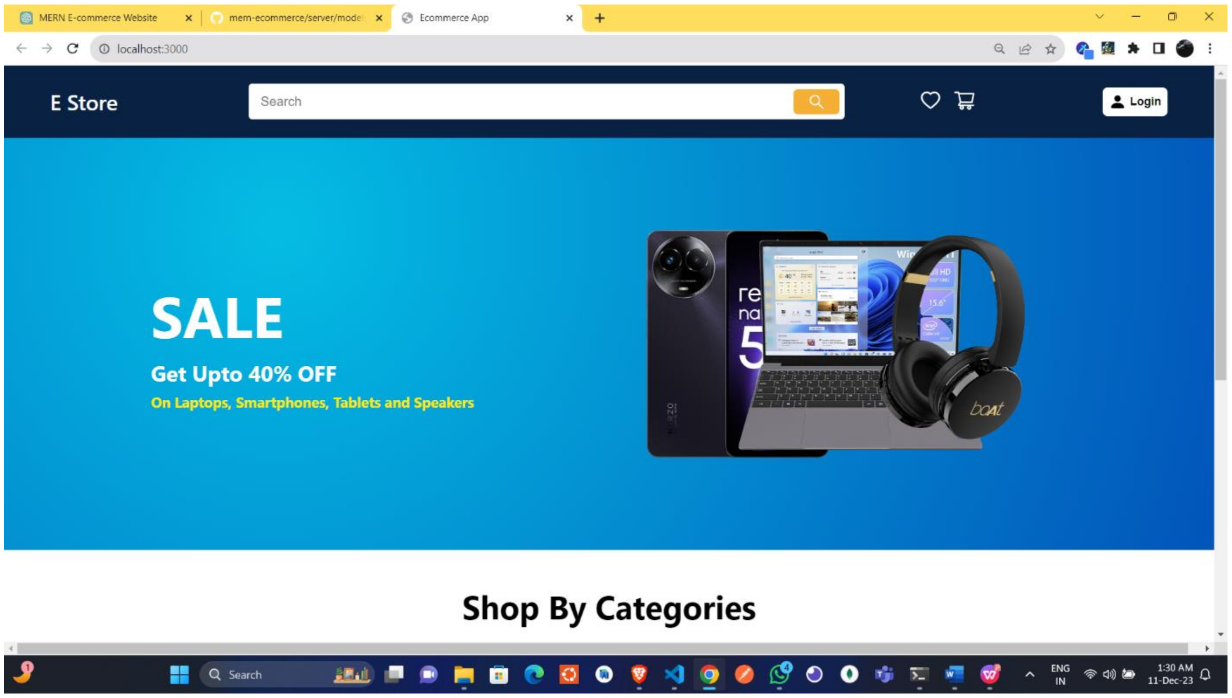
CartItem

Field	Type
_id	String
Product	Object Id(Ref:`Product`)
Quantity	Number
Price	Number

Product

Field	Type
_id	String
Title	String
Price	Number
Description	String
Category	String
Image	String
Brand	String
Bestselling	String

4.2 Home page



Main Home Page.

We can see the home page of the app. It includes the indispensable components of an online store like the Navbar that includes the Search bar, Login, Wishlist and Cart Button.

Code:

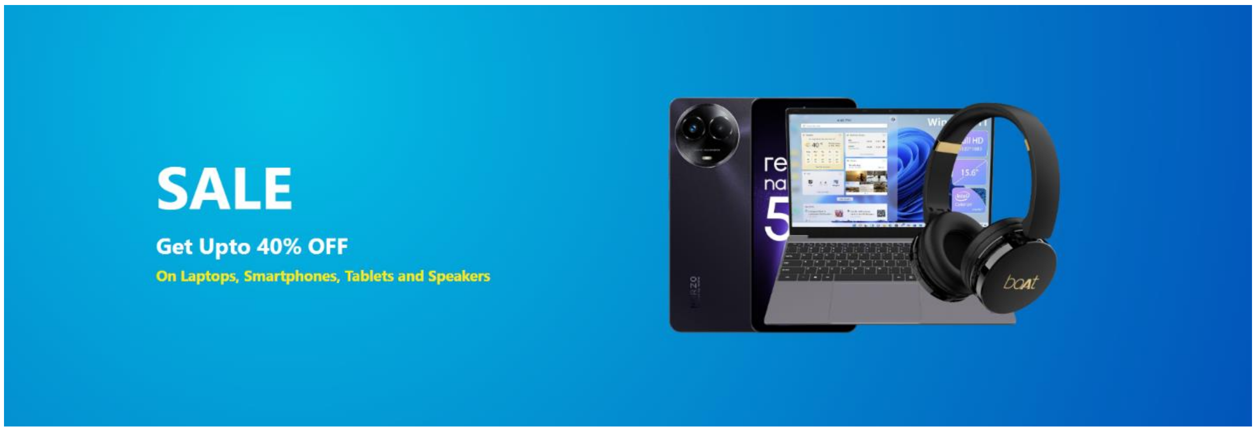
```
export default function Navbar() {
   const { isLoggedIn, setIsLoggedIn, setUserData, setShowCart, cart, setCart,
setWishListedProducts } = useContext(Context);

   const [scrolled, setScrolled] = useState(false);

   const navigate = useNavigate();

   const handleScroll = () => {
       const offset = window.scrollY;  //it counts by how many pixels the screen is getting scrolled in y
direction
       if (offset > 200) {
           setScrolled(true)
       } else {
           setScrolled(false)
       }
   }
```

```
const handleLogout = () => {
    localStorage.removeItem("authToken");
    navigate("/");
    setWishListedProducts([]);
    setIsLoggedIn(false);
    setUserData({});
    toast.success("Logged Out")
}

const openWishListPage = () => {
    if (!isLoggedIn) {
        toast.error("You need to login first");
    } else {
        navigate("/wishlistPage")
    }
}
const openShoppingCart = () => {
    if (!isLoggedIn) {
        toast.error("You need to login first");
    } else {
        setShowCart(true);
    }
}

useEffect(() => {
    window.addEventListener("scroll", handleScroll)
}, [])
return (
    <>
        <div className={`navbar-container ${scrolled ? "sticky_navbar" : ""}`}>
            <div className="left">
                <h1 onClick={() => navigate("/")}> E Store</h1>
            </div>
            <div className="middle">
                <div className="searchBox">
                    <input type="text" placeholder='Search' />
                    <AiOutlineSearch className='searchIcon' />
                </div>
            </div>
            <div className="right">
                <div className='icons'>
```

```
                    <div className='icon' onClick={openWishListPage} ><AiOutlineHeart
className='right_icons' /></div>
                    <div className='icon' onClick={openShoppingCart}>
                        <AiOutlineShoppingCart className='right_icons' />
                        {cart.length !== 0 && <p className='number'>{cart.length}</p>}
                    </div>
                </div>
                {!isLoggedIn ?
                    <button className="login" onClick={() => navigate("/login")}>
                        <BsFillPersonFill className='right_icons' />
                        <p>Login</p>
                    </button> : <button className='login2' onClick={handleLogout}>
                        Logout
                    </button>
                }
            </div>
        </div>
    </>
  )
}
```

The Home Page contains 3 components: Banner, Categories and BestSelling.

Banner:



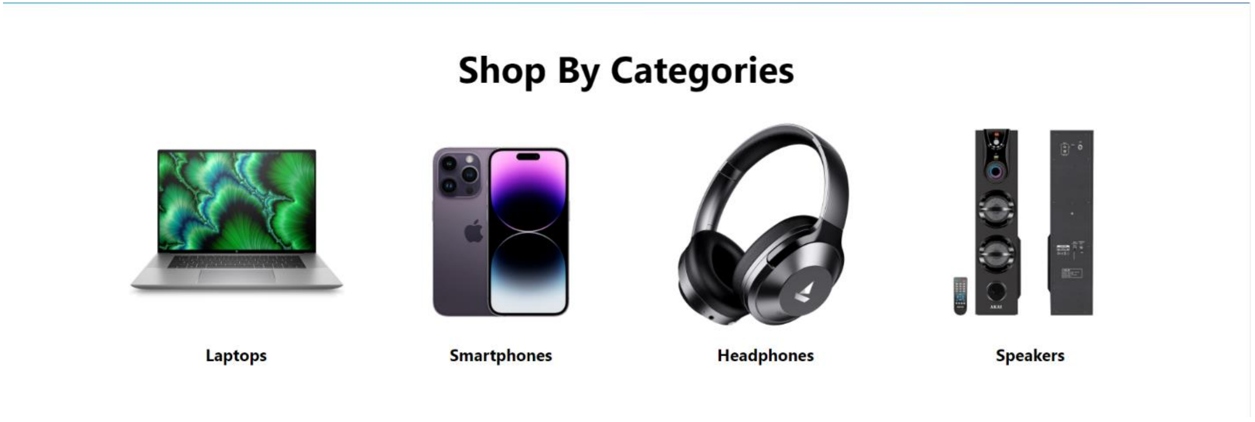
The above figure contains the Banner.

<u>**Code:**</u>

```
export default function Banner() {
  return (
    <div className='Banner_Container'>
      <div className="left">
          <h1>SALE</h1>
```

```
            <h2>Get Upto 40% OFF</h2>
            <h3>On Laptops, Smartphones, Tablets and Speakers</h3>
      </div>
      <div className="right">
            <img src={Banner_Smartphone} id='img1' alt="" />
            <img src={BannerLaptop} id='img2' alt="" />
            <img src={Banner_Headphone} id='img3' alt="" />
      </div>
    </div>
  )
}
```

Shop By Category:

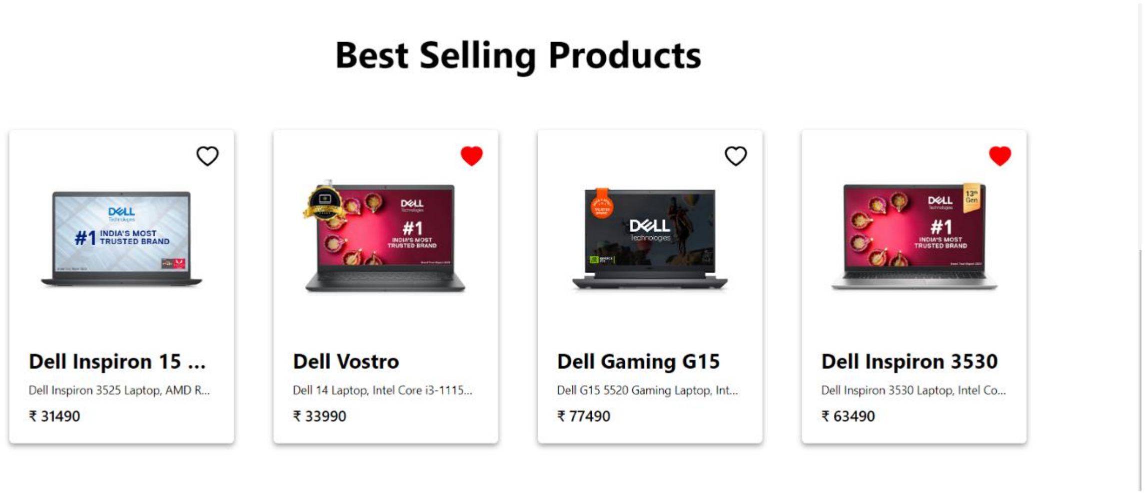


Users can click on the Category of Product they are looking for.

Code:

```
Below is the code for Shop By Category:
export default function Categories() {

    const navigate = useNavigate();
    const { setSelectedCategory, pageNumber } = useContext(Context);
    return (
        <>
            <div className="categories-container">
                <h1>Shop By Categories</h1>
                <div className="img-container">
                    <div className="img-container-items" onClick={() => {
                        setSelectedCategory("laptops");
                        navigate(`/categoryProduct/:pageNumber`);
                    }}>
            <img src={Laptop} alt="" />
            <p>Laptops</p>
          </div>
          <div className="img-container-items" onClick={() => {
            setSelectedCategory("smartphones");
            navigate(`/categoryProduct/:pageNumber`);
          }}>
            <img src={Smartphone} alt="" />
            <p>Smartphones</p>
          </div>
          <div className="img-container-items" onClick={() => {
            setSelectedCategory("headphones");
            navigate(`/categoryProduct/:pageNumber`);
          }}>
            <img src={HeadPhone} alt="" />
            <p>Headphones</p>
          </div>
          <div className="img-container-items" onClick={() => {
            setSelectedCategory("speakers");
            navigate(`/categoryProduct/:pageNumber`);
          }}>
            <img src={Speaker} alt="" />
            <p>Speakers</p>
          </div>
        </div>
      </div>
    </>
  )
}
```

BestSelling:


Best Selling Products

Dell Inspiron 15 ...
Dell Inspiron 3525 Laptop, AMD R...
₹ 31490

Dell Vostro
Dell 14 Laptop, Intel Core i3-1115...
₹ 33990

Dell Gaming G15
Dell G15 5520 Gaming Laptop, Int...
₹ 77490

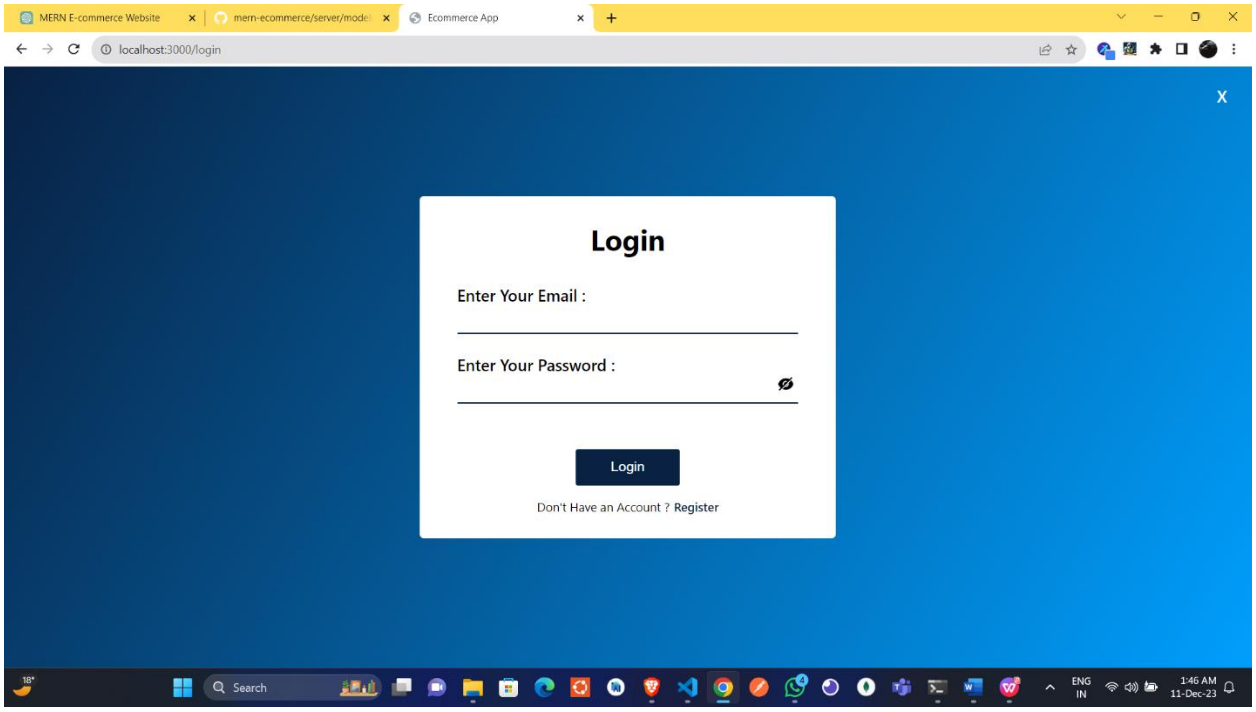
Dell Inspiron 3530
Dell Inspiron 3530 Laptop, Intel Co...
₹ 63490

This Component shows all the best selling products.

Code:

```
export default function BestSelling() {
    const [bestSellingProducts, setBestSellingProducts] = useState([]);
    const [loading, setLoading] = useState(false);
    const [error, setError] = useState(false);

    useEffect(() => {
        const fetchBestSellingProducts = async (req, res) => {
            setLoading(true);
            try {
                const response = await axios.get("/api/products/products/bestSelling")

                setBestSellingProducts(response.data);
                setLoading(false);
            } catch (error) {
                console.log(error);
                setLoading(false);
                setError("Some Error Occured");
            }
        }
        fetchBestSellingProducts();
    }, [])
    return (
        <div className='BestSelling_Container'>
            <h1 className='bestSelling_Heading'>Best Selling Products</h1>
            <div className="card_Container">
            {loading ? (<h1 className="loading_text">Loading ...</h1>) : error ?
              (<h1 className='error_text'>Some Error Occured ...</h1>) :
              <>
                {Array.isArray(bestSellingProducts) && bestSellingProducts.length > 0 ? (
                  <div className='bestSelling_Container'>
                    {bestSellingProducts.map((item) => (
                      <Link to={`/productPage/${item._id}`} key={item._id}
className='card_To_Product_Link'>
                        <Card key={item._id} post={item} />
                      </Link>
                    ))}
                  </div>
                ) : (
                  <h1 className='error_text'>No Best Selling Products Available</h1>
                )}

              </>
            }
          </div>
        </div>
    )
}
```

4.3 Login Page



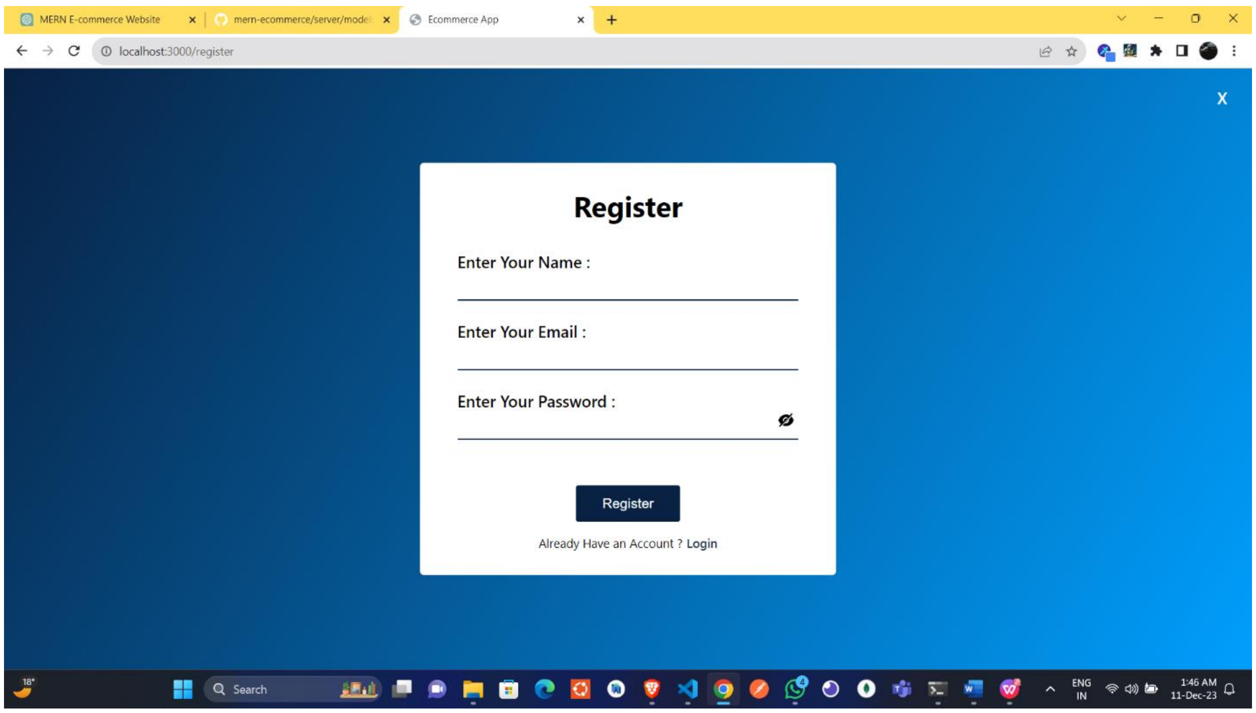
The login page of the app will ask the user to provide the username and password to login to their account and continue shopping.

Code:

```
export default function Register() {
  const { userData, setUserData, isLoggedIn, setIsLoggedIn, setWishListedProducts } =
useContext(Context);
  const [formData, setFormData] = useState({
    Email: "",
    Password: ""
  });
  const [showPassword, setShowPassword] = useState(false);
  const [errors, setErrors] = useState({
    Email: "",
    Password: ""
  });

  const toggleShowPassword = () => {
    setShowPassword(!showPassword);
  }
```

```
const handleChange = (e) => {
  setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
};

const navigate = useNavigate();

const validateForm = () => {
  let isValid = true;
  const newErrors = {
    Email: "",
    Password: ""
  };

  // Email validation
  if (formData.Email.length === 0) {
    newErrors.Email = "*Email field can't be empty";
    isValid = false;
  } else if (!formData.Email.match(/^.+@gmail\.com$/)) {
    newErrors.Email = "*Enter a valid gmail address";
    isValid = false;
  }

  // Password validation
  if (formData.Password.length < 8) {
    newErrors.Password = "*Password must have atleast 8 characters.";
    isValid = false;
  }

  setErrors(newErrors);
  return isValid;
};

const loginUser = async () => {
  try {
    const response = await axios.post("/api/users/login", formData);
    const { userResponse, token } = response.data.data;
    const message = response.data.message;

    const userID = response.data.data.userResponse._id;
    const response2 = await axios.post("/api/products/getProducts/wishlisted", { userId: userID });
    setWishListedProducts(response2.data.wishlistedProducts);
    setUserData({ userResponse, token })
    localStorage.setItem("authToken", token);
    toast.success(message);
    setIsLoggedIn(true);
    navigate("/");
  } catch (error) {
    console.error("Registration failed", error);
    toast.error(error.response.data.message);
  }
};

const handleSubmit = async (e) => {
  e.preventDefault();
  if (validateForm()) {
    await loginUser();
  }
};

return (
  <div className='login_Container'>
    <h1 className="close_mark_btn" onClick={() => navigate("/")} >X</h1>
    <form className='login_form' onSubmit={handleSubmit}>
      <h1 className="formHeading">Login</h1>
      <div className="labelInput">
        <label htmlFor="email">Enter Your Email : </label>
        <input
          type="text"
          id='email'
          name='Email'
          value={formData.Email}
          onChange={handleChange}
        />
        <p className="error">{errors.Email}</p>
      </div>
      <div className="labelInput">
        <label htmlFor="password">Enter Your Password : </label>
        <div className="password_container">
          <input
            type={showPassword ? "text" : "password"}
            className='password_input'
            id='password'
            name='Password'
            value={formData.Password}
            onChange={handleChange}
          />
          {showPassword ? <AiFillEye className='icon' onClick={toggleShowPassword} /> :
<AiFillEyeInvisible className='icon' size={22} onClick={toggleShowPassword} />}
        </div>
        <p className="error">{errors.Password}</p>
      </div>
      <button type='submit' className='login_btn'>
        Login
      </button>
      <div className='registerLink_Box'>
        Don't Have an Account ? <Link to={"/register"} className='register_Link'>Register</Link>
      </div>
    </form>
  </div>
 )
}
```

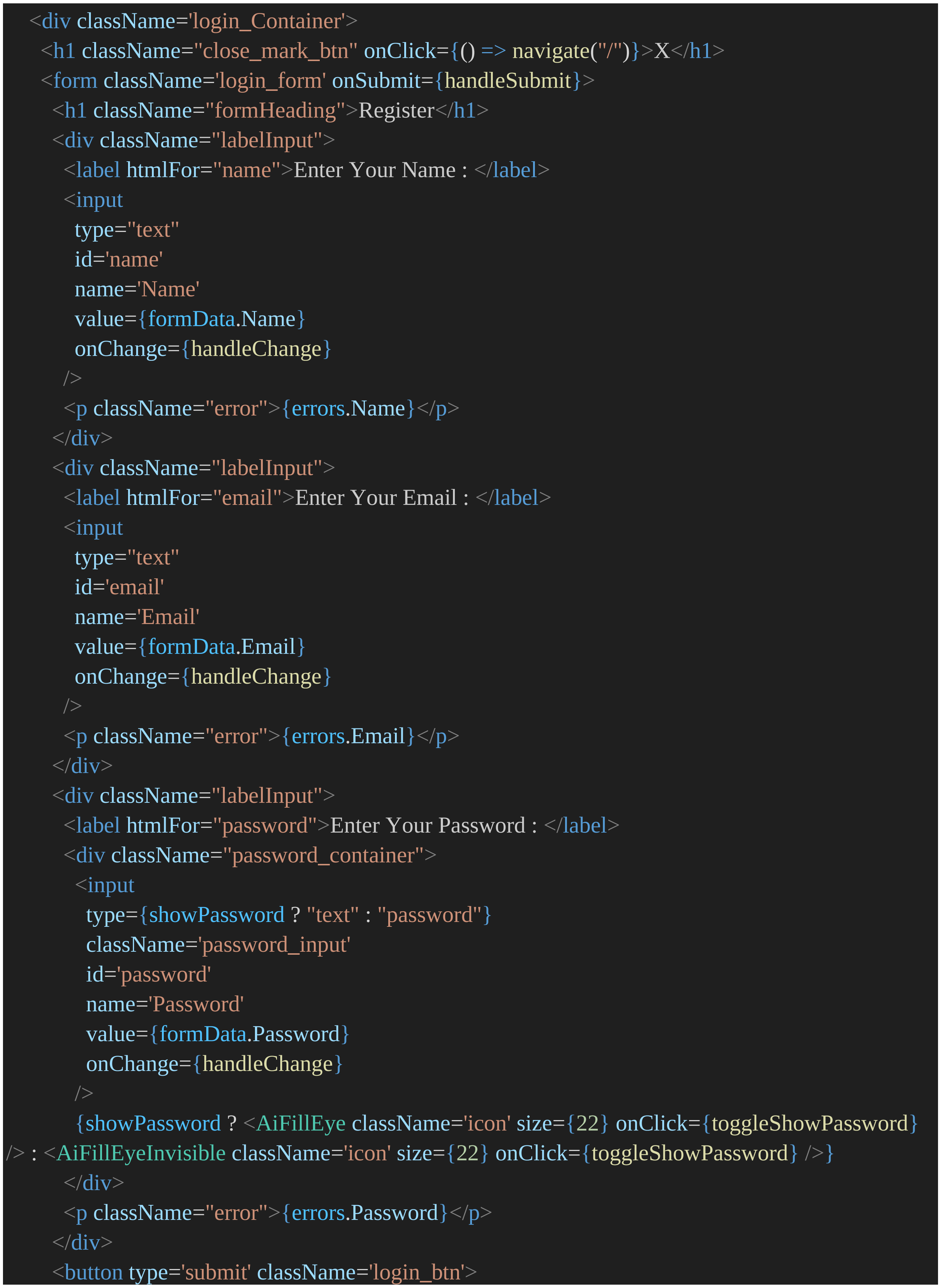
4.4 Register



Users login into the website need to sign in otherwise they have to register a new account. The Register screen only has 3 fields that require users to fill in is Name, Email and Password.

Code:

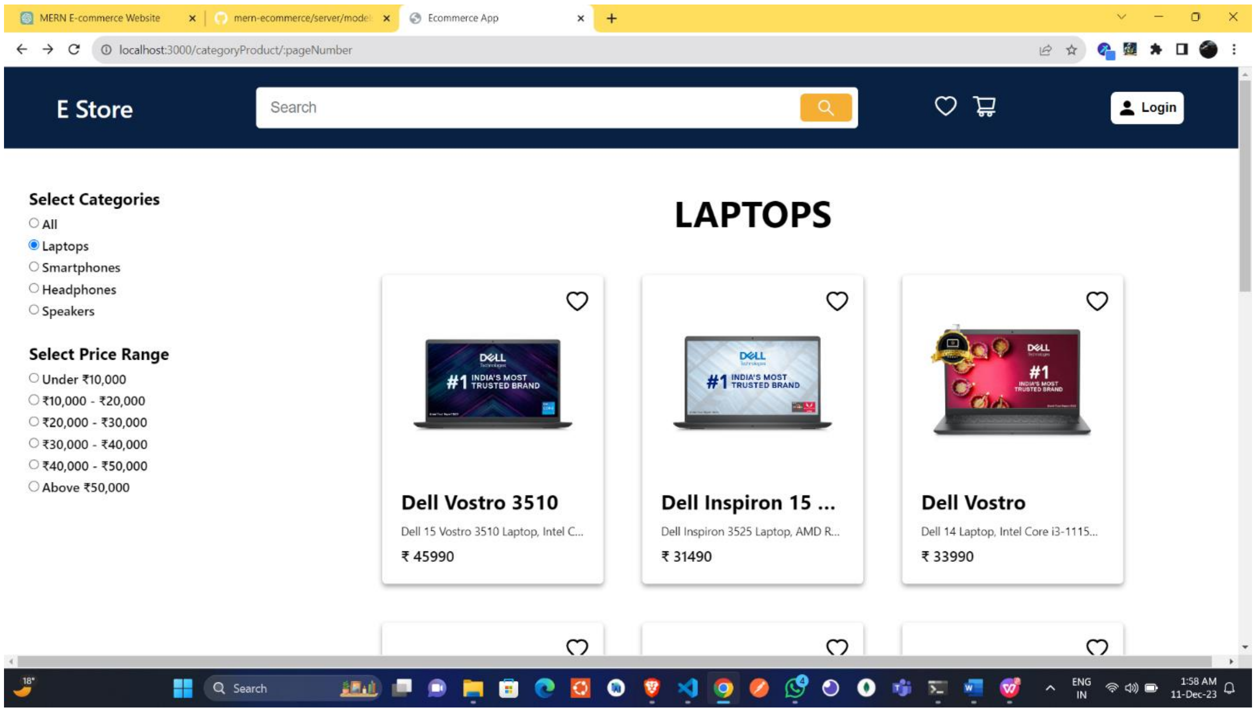
```
export default function Register() {
  const { userData, setUserData, isLoggedIn, setIsLoggedIn, setWishListedProducts } =
useContext(Context);
  const [formData, setFormData] = useState({
    Name: "",
    Email: "",
    Password: ""
  });
  const [showPassword, setShowPassword] = useState(false);
  const [errors, setErrors] = useState({
    Name: "",
    Email: "",
    Password: ""
  });

  const toggleShowPassword = () => {
    setShowPassword(!showPassword);
  }

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const navigate = useNavigate();

  const validateForm = () => {
    let isValid = true;
    const newErrors = {
      Name: "",
      Email: "",
      Password: ""
    };

    // Name validation
    if (formData.Name.length < 3) {
      newErrors.Name = "*Name must have atleast 3 characters";
      isValid = false;
    } else if (!/^[a-zA-Z\s]+$/.test(formData.Name)) {
      newErrors.Name = "*Name must contain only alphabets and spaces";
      isValid = false;
    }
    // Email validation
    if (formData.Email.length === 0) {
      newErrors.Email = "*Email field can't be empty";
      isValid = false;
    } else if (!formData.Email.match(/^.+@gmail\.com$/)) {
      newErrors.Email = "*Enter a valid gmail address";
      isValid = false;
    }

    // Password validation
    if (formData.Password.length < 8) {
      newErrors.Password = "*Password must have atleast 8 characters.";
      isValid = false;
    }

    setErrors(newErrors);
    return isValid;
  };

  const registerUser = async () => {
    try {
      const response = await axios.post("/api/users/register", formData);
      const { userResponse, token } = response.data.data;
      const message = response.data.message;
      setUserData({ userResponse, token })
      localStorage.setItem("authToken", token);
      toast.success(message);
      setIsLoggedIn(true);
      navigate("/");
    } catch (error) {  //for error code above 400 it goes to catch block
      toast.error(error.response.data.message)
      console.error("Registration failed", error);
    }
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (validateForm()) {
      await registerUser();
    }
  };

  return (
```



```
<div className='login_Container'>
  <h1 className="close_mark_btn" onClick={() => navigate("/")}>X</h1>
  <form className='login_form' onSubmit={handleSubmit}>
    <h1 className="formHeading">Register</h1>
    <div className="labelInput">
      <label htmlFor="name">Enter Your Name : </label>
      <input
        type="text"
        id='name'
        name='Name'
        value={formData.Name}
        onChange={handleChange}
      />
      <p className="error">{errors.Name}</p>
    </div>
    <div className="labelInput">
      <label htmlFor="email">Enter Your Email : </label>
      <input
        type="text"
        id='email'
        name='Email'
        value={formData.Email}
        onChange={handleChange}
      />
      <p className="error">{errors.Email}</p>
    </div>
    <div className="labelInput">
      <label htmlFor="password">Enter Your Password : </label>
      <div className="password_container">
        <input
          type={showPassword ? "text" : "password"}
          className='password_input'
          id='password'
          name='Password'
          value={formData.Password}
          onChange={handleChange}
        />
        {showPassword ? <AiFillEye className='icon' size={22} onClick={toggleShowPassword}
/> : <AiFillEyeInvisible className='icon' size={22} onClick={toggleShowPassword} />}
      </div>
      <p className="error">{errors.Password}</p>
    </div>
    <button type='submit' className='login_btn'>
```

```
        Register
      </button>
      <div className='registerLink_Box'>
        Already Have an Account ? <Link to={"/login"} className='register_Link'>Login</Link>
      </div>
    </form>
  </div>
 )
}
```

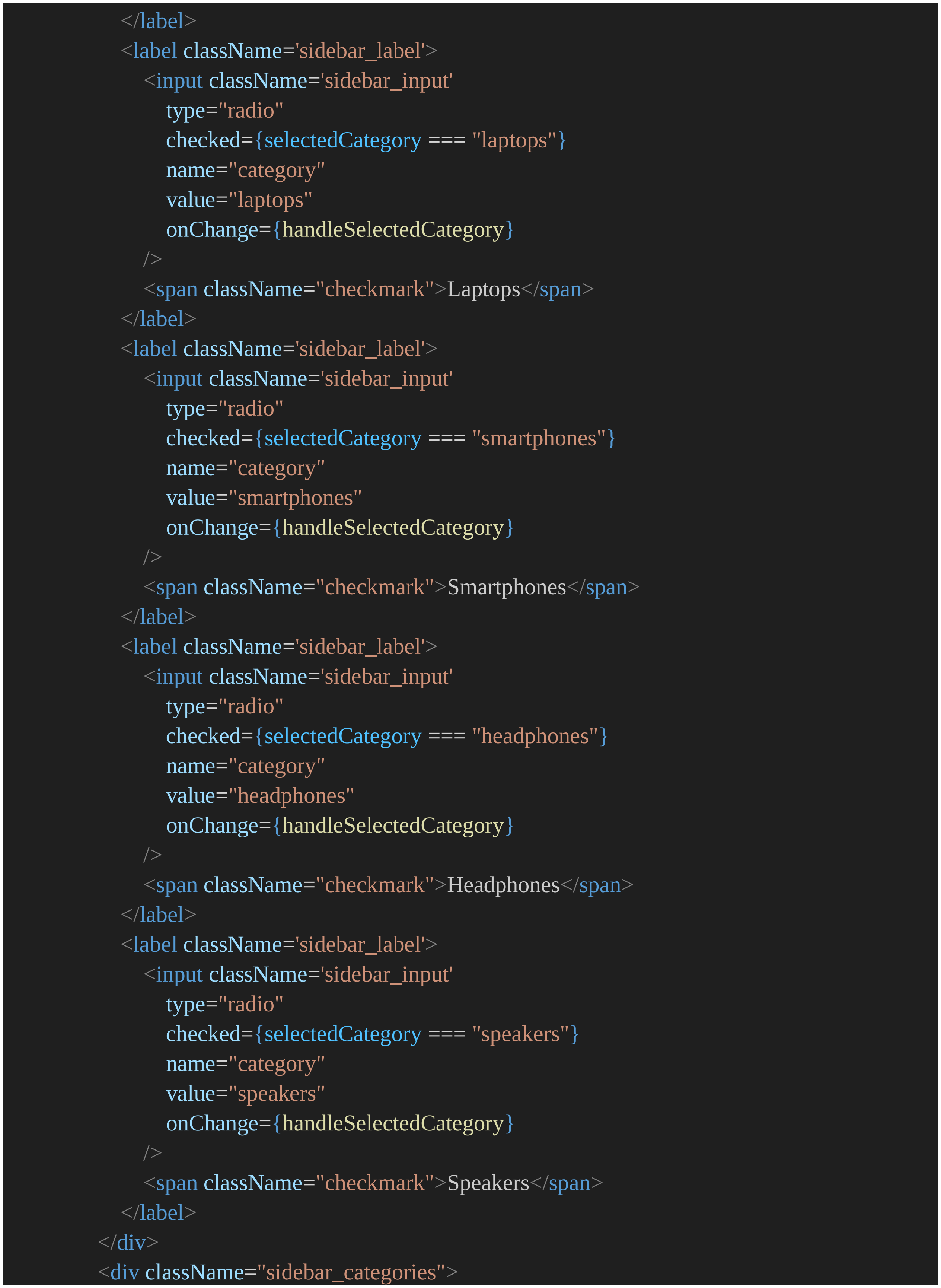
4.5 Shop Page



The main shop page that customers can select the products that they want to add to cart. User can scroll down the mouse to browse all the products, or filter by categories or filter by price range.

Code:

```
export default function CategoryProduct() {
    const {selectedCategory, setSelectedCategory} = useContext(Context);
    const [selectedPriceRange, setSelectedPriceRange] = useState(null);
    const [minPrice, setMinPrice] = useState(1000);
    const [maxPrice, setMaxPrice] = useState(1000000);

    const location = useLocation();

    useEffect(() => {
        if (selectedPriceRange === null) {
            setMinPrice(1000);
            setMaxPrice(1000000);
        }
    }, [selectedPriceRange]);

    const handleSelectedCategory = (event) => {
        setSelectedCategory(event.target.value);
    }

    const handleSelectedPriceRange = (value) => {
        if (selectedPriceRange === value) {
            setSelectedPriceRange(null);
        } else {
            setSelectedPriceRange(value);
        }
    };

    return (
        <div className='CategoryProduct_Container'>
            <div className="left">
                <div className='sidebar_categories'>
                    <h1 className='select_categories_heading'>Select Categories</h1>
                    <label className='sidebar_label'>
                        <input className='sidebar_input'
                            type="radio"
                            checked={selectedCategory === "all"}
                            name="category"
                            value="all"
                            onChange={handleSelectedCategory}
                        />
                        <span className="checkmark">All</span>
```

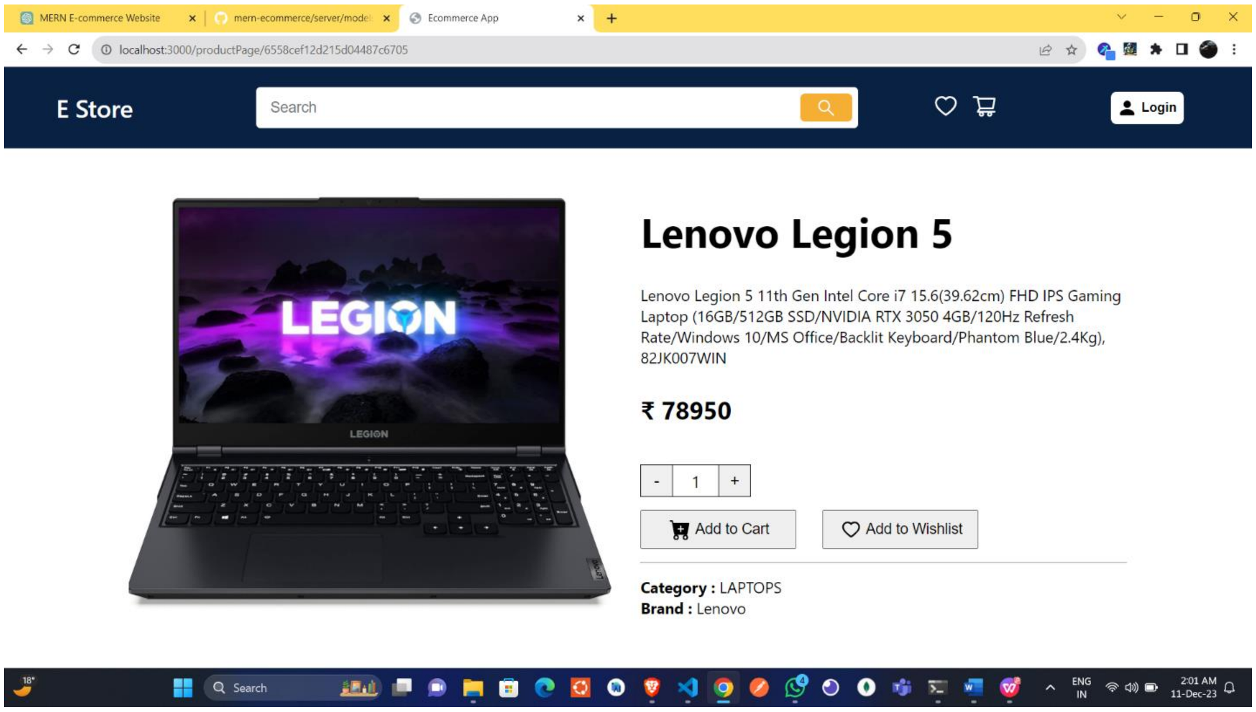


```
          </label>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              checked={selectedCategory === "laptops"}
              name="category"
              value="laptops"
              onChange={handleSelectedCategory}
            />
            <span className="checkmark">Laptops</span>
          </label>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              checked={selectedCategory === "smartphones"}
              name="category"
              value="smartphones"
              onChange={handleSelectedCategory}
            />
            <span className="checkmark">Smartphones</span>
          </label>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              checked={selectedCategory === "headphones"}
              name="category"
              value="headphones"
              onChange={handleSelectedCategory}
            />
            <span className="checkmark">Headphones</span>
          </label>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              checked={selectedCategory === "speakers"}
              name="category"
              value="speakers"
              onChange={handleSelectedCategory}
            />
            <span className="checkmark">Speakers</span>
          </label>
        </div>
        <div className="sidebar_categories">
```

```
          <h1 className="select_categories_heading">Select Price Range</h1>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              name="price_range"
              checked={selectedPriceRange === "Under &#8377;10,000"}
              value="Under &#8377;10,000"
              onClick={() => { handleSelectedPriceRange("Under &#8377;10,000") }}
              onChange={() => {
                setMinPrice(1000);
                setMaxPrice(10000);
              }}
            />
            <span className="checkmark">Under &#8377;10,000</span>
          </label>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              name="price_range"
              checked={selectedPriceRange === "&#8377;10,000 - &#8377;20,000"}
              value="&#8377;10,000 - &#8377;20,000"
              onClick={() => { handleSelectedPriceRange("&#8377;10,000 - &#8377;20,000")
}}
              onChange={() => {
                setMinPrice(10000);
                setMaxPrice(20000);
              }}
            />
            <span className="checkmark">&#8377;10,000 - &#8377;20,000</span>
          </label>
          <label className='sidebar_label'>
            <input className='sidebar_input'
              type="radio"
              name="price_range"
              checked={selectedPriceRange === "&#8377;20,000 - &#8377;30,000"}
              value="&#8377;20,000 - &#8377;30,000"
              onClick={() => { handleSelectedPriceRange("&#8377;20,000 - &#8377;30,000")
}}
              onChange={() => {
                setMinPrice(20000);
                setMaxPrice(30000);
              }}
            />
          <span className="checkmark">&#8377;20,000 - &#8377;30,000</span>
        </label>
        <label className='sidebar_label'>
          <input className='sidebar_input'
            type="radio"
            name="price_range"
            checked={selectedPriceRange === "&#8377;30,000 - &#8377;40,000"}
            value="&#8377;30,000 - &#8377;40,000"
            onClick={() => { handleSelectedPriceRange("&#8377;30,000 - &#8377;40,000")
}}
            onChange={() => {
              setMinPrice(30000);
              setMaxPrice(40000);
            }}
          />
          <span className="checkmark">&#8377;30,000 - &#8377;40,000</span>
        </label>
        <label className='sidebar_label'>
          <input className='sidebar_input'
            type="radio"
            name="price_range"
            checked={selectedPriceRange === "&#8377;40,000 - &#8377;50,000"}
            value="&#8377;40,000 - &#8377;50,000"
            onClick={() => { handleSelectedPriceRange("&#8377;40,000 - &#8377;50,000")
}}
            onChange={() => {
              setMinPrice(40000);
              setMaxPrice(50000);
            }}
          />
          <span className="checkmark">&#8377;40,000 - &#8377;50,000</span>
        </label>
        <label className='sidebar_label'>
          <input className='sidebar_input'
            type="radio"
            name="price_range"
            checked={selectedPriceRange === "Above &#8377;50,000"}
            value="Above &#8377;50,000"
            onClick={() => { handleSelectedPriceRange("Above &#8377;50,000") }}
            onChange={() => {
              setMinPrice(50000);
              setMaxPrice(100000)
            }}
```

```
              />
              <span className="checkmark">Above &#8377;50,000</span>
            </label>
          </div>
        </div>
        <div className="right">
          <h1 className='heading'>{selectedCategory && selectedCategory.toUpperCase()}</h1>
          <div className="ProductsContainer">
            <ProductsByPage
            selectedCategory={selectedCategory}
            minPrice={minPrice}
            maxPrice={maxPrice}
             />
          </div>
        </div>
      </div>
  )
}
```

4.6 Product Page



MERN E-commerce Website | mern-ecommerce/server/model | Ecommerce App

localhost:3000/productPage/6558cef12d215d04487c6705

Lenovo Legion 5

Lenovo Legion 5 11th Gen Intel Core i7 15.6(39.62cm) FHD IPS Gaming Laptop (16GB/512GB SSD/NVIDIA RTX 3050 4GB/120Hz Refresh Rate/Windows 10/MS Office/Backlit Keyboard/Phantom Blue/2.4Kg), 82JK007WIN

₹ 78950

- 1 +

Add to Cart   Add to Wishlist

Category : LAPTOPS
Brand : Lenovo

This Product page contains Title, Price, Description and Image of the Product. It also shows the category and the brand of the Product. There are two buttons: “Add To Cart” and “Add To Wishlist” buttons with which user can either add the product to cart or to wishlist.

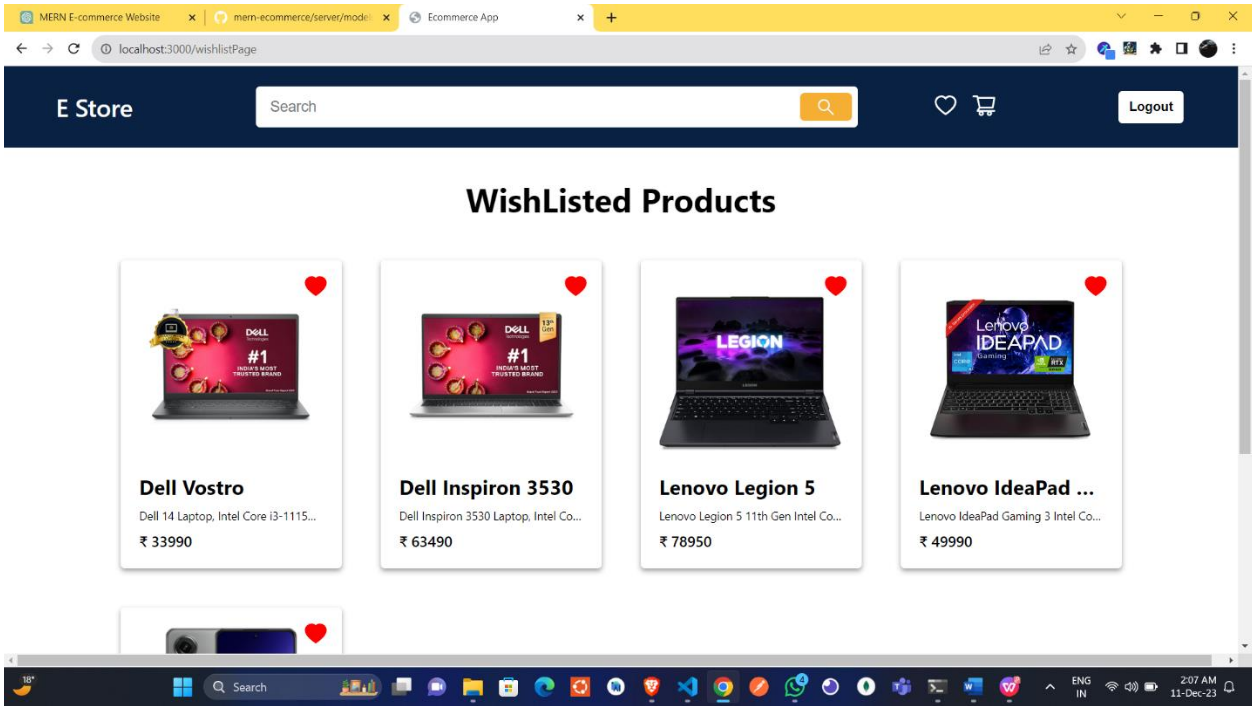
Code:

```
export default function ProductPage() {
  const { userData, isLoggedIn, wishListedProducts, setWishListedProducts, cartItems, setCartItems,
cart, setCart } = useContext(Context);
  const [singleProduct, setSingleProduct] = useState("");
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(false);
  const { id } = useParams();
  const [productLiked, setProductLiked] = useState(false);
  const [quantity, setQuantity] = useState(1);

  const decrement = () => {
    setQuantity((prevState) => {
      if (prevState === 1) return 1;
      return prevState - 1;
    });
  };
  const increment = () => {
    setQuantity((prevState) => prevState + 1);
  };

  useEffect(() => {

    const fetchProductDetails = async () => {
      setLoading(true);
      try {
        const response = await axios.get(`/api/products/getProductById/${id}`)

        setSingleProduct(response.data);
        setLoading(false);
      } catch (error) {
        console.log(error);
        setLoading(false);
        setError("Some Error Occured");
      }
    }
    fetchProductDetails();
  }, [])
```

```
const productId = singleProduct._id;

useEffect(() => {
  const isProductLiked = wishListedProducts.some(product => product._id === productId);
  setProductLiked(isProductLiked);
}, [wishListedProducts, productId]);

const addProductToWishList = async () => {
  try {
    const response = await axios.post(
      '/api/products/addProducts/toWishlist',
      {
        productId: singleProduct._id,
        userId: userData.userResponse._id
      },
      {
        headers: {
          Authorization: `Bearer ${userData.token}`
        },
      }
    );

    // Toggle the productLiked state
    setProductLiked(prevProductLiked => !prevProductLiked);
    setWishListedProducts(response.data.wishlistedProducts);
  } catch (error) {
    console.log(error);
    toast.error('Error adding product to wishlist');
  }
};

const toggleLike = async (e) => {
  e.preventDefault();
  if (!isLoggedIn) {
    toast.error("You need to login first");
  } else {
    if (productLiked === false) {
      await addProductToWishList();
    }
  }
}
```

```
const toggleAddToCart = () => {
  if (!isLoggedIn) {
    toast.error('You need to login first');
  } else {
    const existingCartItemIndex = cart.findIndex((item) => item.product._id === singleProduct._id);

    if (existingCartItemIndex !== -1) {
      // If the product is already in the cart, update its quantity
      const updatedCart = [...cart];
      updatedCart[existingCartItemIndex].productQuantity += quantity;
      setCart(updatedCart);
    } else {
      // If the product is not in the cart, add it
      const newCartItem = {
        product: singleProduct,
        productQuantity: quantity,
      };
      setCart((prevCart) => [...prevCart, newCartItem]);
    }

    setCartItems({
      product: singleProduct,
      productQuantity: quantity,
    });

    toast.success('Product added to cart');
  }
};

console.log('From Product Page:', singleProduct);

return (
  <div className='ProductPageContainer'>
    {loading ? (<h3 className="loading_text">Loading...</h3>) :
      error ? (<h3 className="error_text">{error}</h3>) :
        (
          <>
            <div className="left">
              <img src={singleProduct.Image} alt="" />
            </div>
            <div className="right">
              <h1 className="productTitle">{singleProduct.Title}</h1>
              <p className="productDesc">{singleProduct.Description}</p>
        <p className="productPrice">&#8377; {singleProduct.Price}</p>
        <div className="quantity_container">
         <button className="quantity_btn" onClick={decrement}>-</button>
         <p className="quantity">{quantity}</p>
         <button className="quantity_btn" onClick={increment}>+</button>
        </div>
        <div className="btn_container">
         <button className="addToCart" onClick={toggleAddToCart}>
          <BsFillCartPlusFill className='icon' />
          Add to Cart
         </button>
         <button className="wishlist" onClick={toggleLike}>
          {productLiked ? <>
           <AiFillHeart className='icon' /> Wishlisted
          </> : <>
           <AiOutlineHeart className='icon' />
           Add to Wishlist
          </>
          }</button>
        </div>
        <hr className='division_line' />
        <p className="labels"><strong>Category :</strong> {singleProduct.Category && singleProduct.Category.toUpperCase()}</p>
        <p className="labels"><strong>Brand :</strong> {singleProduct.Brand}</p>
       </div>
      </>
     )
   }
  </div>
 )
}
```

4.7 Wishlist Page


localhost:3000/wishlistPage
E Store
Search
Logout
WishListed Products
Dell Vostro
Dell 14 Laptop, Intel Core i3-1115...
₹ 33990
Dell Inspiron 3530
Dell Inspiron 3530 Laptop, Intel Co...
₹ 63490
Lenovo Legion 5
Lenovo Legion 5 11th Gen Intel Co...
₹ 78950
Lenovo IdeaPad ...
Lenovo IdeaPad Gaming 3 Intel Co...
₹ 49990

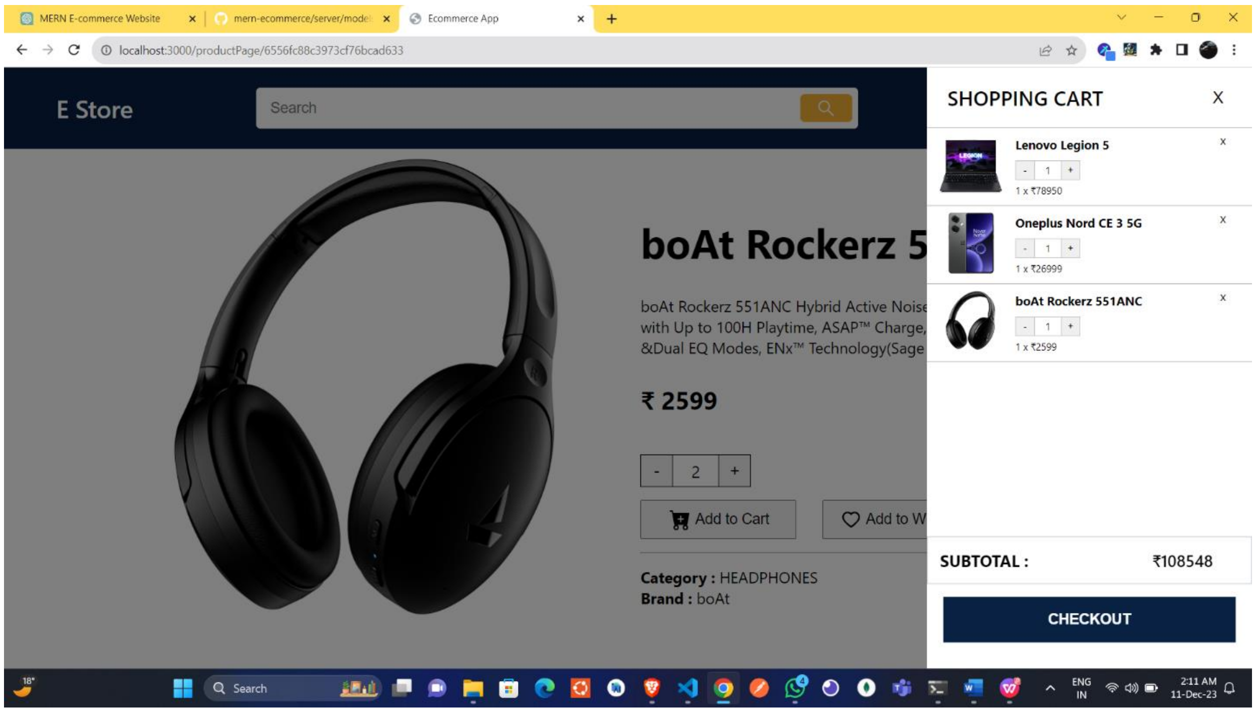
The wishlist page contains all the products that the user has wishlisted.

Code:

```
export default function WishListPage() {
  const { userData } = useContext(Context);
  const { wishListedProducts, setWishListedProducts } = useContext(Context);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(false);

  return (
    <div className='WishListPage_Container'>
      <h1 className='wishListedProductsHeading'>WishListed Products</h1>
      {loading ? (<h3 className="loading_text">Loading...</h3>) :
        error ? (<h3 className="error_text">{error}</h3>) :
          (
            <>
              <div className="wishListedProducts_Container">
                {
                  wishListedProducts.length !== 0 ? (<>
                    {wishListedProducts && (wishListedProducts?.map((item) => (
                      <Link to={`/productPage/${item._id}`} key={item._id}
className='card_To_Product_Link'>
```

```
                <Card key={item._id} post={item} />
              </Link>
            )))}
          </>) : (
            <h1 className='noProducts_head'>No Products available</h1>
          )
        }
      </div>
    </>
  )
}
</div>
);
}
```

4.8 Cart Component



The cart component contains all the products that the user has added to cart to purchase and at the bottom of this component it shows the total price of all the products that were added to cart.

Code:

```
export default function Cart() {
    const { showCart, setShowCart, cart } = useContext(Context);
    const [cartTotalPrice, setCartTotalPrice] = useState(0);

    // Update cart total price whenever the cart changes
    useEffect(() => {
        calculateCartTotal();
    }, [cart]);

    const calculateCartTotal = () => {
        const total = cart.reduce((acc, cartItem) => {
            return acc + cartItem.product.Price * cartItem.productQuantity;
        }, 0);
        setCartTotalPrice(total);
    };

    return (
        showCart && (
            <div className="shopping_cart">
                <div className="dark_background"></div>
                <div className="cart_container">
                    <div className="cartContainerHead">
                        <h1 className='header'>SHOPPING CART</h1>
                        <button className='close_btn' onClick={() => setShowCart(false)}>X</button>
                    </div>
                    <div className="cart">
                        {cart.length === 0 ?
                            <div className="empty_cart">
                                <BsCartX className='empty_cart_icon' />
                                <h1 className="empty_cart_head">No products in the Cart</h1>
                                <button className='returnToShop' onClick={() => setShowCart(false)}>Return
To Shop</button>
                            </div> :
                            <div>
                                {cart && (cart?.map((item) => (
                                    <CartItem key={item._id} cartItem={item} />
                                )))}
                            </div>
                        }
                    </div>
                    <div className="cartContainerBottom">
                        {cart.length !== 0 &&
                            <>
                                <div className="subtotal_box">
                                    <h1 className='subtotal'>SUBTOTAL :</h1>
                                    <p className='subtotal_price'> &#8377;{cartTotalPrice}</p>
                                </div>
                                <button className="checkoutbtn">CHECKOUT</button>
                            </>
                        }
                    </div>
                </div>
            </div>
        )
    );
}
```

5. Summary

The achievement of the thesis is researching the basic components of MERN stack technology: MongoDB, ExpressJS framework, ReactJS library and NodeJS platform. Using MERN stack technology in conjunction with Braintree to build an e-commerce web application with payment gateway.

The advantages are performing the basic functions of a product search website for customers, making it easy for customers to find categories that have the products they are looking for. Gives small stores a platform to store information and promote their products.

Password data of accounts when logging in to the system is stored in a secure database. The management interface, statistics of the user and admin are easy to use for everyone. The disadvantages are online chat functions between shop owners and customers are not yet supported as well as between shop owners and administrators. The current product search algorithm locates by the user location that is not really optimal, needs to be improved to speed up the search even more.

Since the purpose of the thesis is the e-commerce application, the understanding about MERN technologies and applying it to this app is the most important. Overcome current shortcomings, listen to customers' comments and making improvements, helping users have a great experience in the future.35

References:

1. E-commerce Definition – What is E-commerce? [Internet]. Shopify.com. Available from:
https://www.shopify.com/encyclopedia/what-is-ecommerce
2. Advantages of E-commerce [Internet]. Thebalancesmb.com 2019 [cited 20 November 2019].
Available from: https://www.thebalancesmb.com/ecommerce
pros-and-cons-1141609
3. JavaScript [Internet]. Mozilla.org. Available from:
https://devel
oper.mozilla.org/en-US/docs/Web/JavaScript
4. NodeJS
Introduction
[Internet].
Tutorialspoint.com. Available from:
https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm
5. NodeJS Pros and Cons [Internet]. Mindinventory.com. Available from:
https://www.mindinventory.com/blog/pros-and-cons-of-node-js-web-app-development/
6. NodeJS use cases [Internet]. Credencys.com. Available from:
https://www.credencys.com/blog/node-js-development-use-cases/
7. Express.js Introduction [Internet]. Mozilla.org. Available from:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
8. MongoDB [Internet]. Mongodb.com. Available from:
https://docs.mongodb.com/manual/introduction/
9. Virtual-DOM [Internet]. Reactjs.org. Available from: https://reactjs.org/docs/faqinternals.html
10. Component [Internet]. Reactjs.org. Available from: https://reactjs.org/docs/components-and-props.html36
11. Props and State [Internet]. Flaviocopes.com. Available from:
https://reactjs.org/docs/components-and-props.html
12. Pros and Cons of ReactJS [Internet]. Javatpoint.com. Available from:
https://www.javatpoint.com/pros-and-cons-of-react
13. Stack technology [Internet]. Stackshare.io. Available from: https://stackshare.io/stacks
14. MERN stack concept [Internet]. Mongodb.com. Available from:
https://www.mongodb.com/mern-stack