PROJECT REPORT

On

ELEVATING ONLINE SHOPPING USING MERN

Submitted for Partial Fulfilment of Award of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

(2024)

By

Ayush Kumar Singh (Roll No: 2001220100038)

Harish Kushwaha (Roll No: 2001220100051)

Under the Guidance

of

Er. Sarika Singh

SHRI RAMSWAROOP MEMORIAL GROUP OF PROFESSIONAL COLLEGES, LUCKNOW

Affiliated to

Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

DEPARTMENT OF COMPUTER SCIENECE AND ENGINEERING SRMCEM

CERTIFICATE

Certified that the project entitled "**ELEVATING ONLINE SHOPPING USING MERN STACK**" submitted by **AYUSH KUMAR SINGH (2001220100038)** and **HARISH KUSHWAHA (2001220100051)** in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering) of Dr. APJ Abdul Kalam Technical University, is a record of student's own work carried under our supervision and guidance. The project report embodies results of original work and studies carried out by students and the contents do not forms the basis for the award of any other degree to the candidate or to anybody else.

Er. Sarika Singh	**Dr. Pankaj Kumar**
Assistant Professor	**(Head of Department)**
(Project Guide)	

DEPARTMENT OF COMPUTER SCIENECE AND ENGINEERING
SRMCEM

DECLARTION

We hereby declare that the project entitled "**ELEVATING ONLINE SHOPPING USING MERN STACK"** submitted by **AYUSH KUMAR SINGH (2001220100038)** and **HARISH KUSHWAHA (2001220100051)** in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering) of Dr. APJ Abdul Kalam Technical University, is record of our own work carried under the supervision and guidance of **Er. Sarika Singh**.
To the best of our knowledge this project has been submitted to Dr. APJ Abdul Kalam Technical University or Institute for the award of any degree.

Name: AYUSH KUMAR SINGH	**Name: HARISH KUSHWAHA**
Roll No: 2001220100038	**Roll No:2001220100051**

DEPARTMENT OF COMPUTER SCIENECE AND ENGINEERING
SRMCEM

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our guide **Er. Sarika Singh** and our coordinator **Er. Sarika Singh** for their exemplary guidance, monitoring and constant encouragement. The blessing, help and guidance given by them time to time shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to **Computer Science and Engineering Department, SRMCEM, Lucknow** for their cordial support, valuable information and guidance, which helped us in this task through various stages.

We are obliged to staff members of **Computer Science and Engineering Department, SRMGPC**, for the valuable information provided by them in their respective fields. Weare grateful for their cooperation. We would like to express my special gratitude and thanks to them for giving us such attention and time.

Last but definitely not least, we would like to thank our mother, father, family member and friends for the constant encouragement and constant support they showed usthroughout our entire period as a college student which helped me to keep going and never give up.

Name: AYUSH KUMAR SINGH	**Name: HARISH KUSHWAHA**
Roll No: 2001220100038	**Roll No:2001220100051**

PREFACE

In an era characterized by digital transformation and the ever-growing importance of e-commerce, the need for innovative and user-friendly online shopping experiences has never been greater. The MERN stack, a powerful combination of MongoDB, Express.js, React, and Node.js, has emerged as a leading technology stack for building robust and scalable web applications. This project, "Elevating Online Shopping with the MERN Stack," aims to leverage the capabilities of the MERN stack to revolutionize the online shopping experience. The traditional online shopping process often leaves room for improvement, such as slow page load times, clunky interfaces, and limited personalization.

When an Online business to buy from another business, the process is called business -to-business(B2B) online shopping. A typical online cloth store enables the customer to browse the Firm's range of products and services, view photos or images of the products, along with information about the product specifications, features and prices. Online stores typically enable shoppers to use “search” features to find specific models, brands or items Online customers must have access to the Internet and a valid method of payment in order to complete a transaction.

This report delves into the objectives, scope, technical details, challenges encountered, and the implementation process of the Algorithm Visualizer project. It also discusses the unique features and functionalities of each algorithm module, shedding light on the algorithms' underlying principles and their visual representation. By providing an in-depth exploration of the online shopping website, this report aims to showcase the value and significance of MERN in comprehending their complexities and fostering a deeper appreciation for the field of computer science.

The project report has been divided into multiple chapters. The topics covered under each areas follows:

INTRODUCTION: This chapter gives our problem definition along with the aims and objectives. This part also includes a section on objectives, project analysis which gives information about our project.

LITERATURE REVIEW: This chapter explains the takes the form of making important summaries from these sources that are of relevance from the entire work.

PROPOSED METHODOLOGY: This chapter describes the way in which prediction system works, about the modules in our projects and the model used. It also defines the software/hardware requirements and specifications along with programming codes used inproject.

RESULT ANALYSIS AND DISCUSSION: This section is used to describe the significance of the system and providesnew insights about overall system.

CONCLUSION: This section covers the various inferences that were drawn after the completion of the entire project.

FUTURE SCOPE: This section gives the future enhancements that can be made in the project idea and its implementation.

REFERENCES: This section lists all the sources we have used in our project so that readerscan easily find what we have cited.

ABSTRACT

This project report presents the design, development, and implementation of an online shopping platform aimed at enhancing the retail experience for both consumers and businesses. The primary objective of this project is to create a user-friendly, secure, and efficient e-commerce website that facilitates seamless transactions and offers a comprehensive shopping experience.

The development process utilized a combination of front-end and back-end technologies, including HTML, CSS, JavaScript, React for the user interface, and Node.js with Express for the server-side application. MongoDB was chosen for the database to ensure scalability and flexibility in handling large volumes of data. The implementation also prioritized security measures such as SSL encryption, user authentication, and data privacy protocols to protect sensitive user information.

The platform's performance was evaluated through rigorous testing, including functionality tests, usability assessments, and security audits. The results demonstrated that the online shopping platform effectively meets the needs of users, providing a reliable and enjoyable shopping experience while ensuring data security and privacy.

This report concludes with a discussion on potential future enhancements, such as integrating artificial intelligence for smarter product recommendations, expanding payment options, and incorporating augmented reality for virtual try-ons. The successful development of this online shopping platform underscores its potential to significantly impact the retail sector by offering a modern, efficient, and secure shopping solution.

Keywords: Online shopping, e-commerce, user-friendly interface, secure payment gateway, product catalog, real-time order tracking, recommendation systems, web development, data security.

TABLE OF CONTENT

List of Figures

CHAPTER – 1

INTRODCTION

1.1 About the project

In an era characterized by digital transformation and the ever-growing importance of e-commerce, the need for innovative and user-friendly online shopping experiences has never been greater. The MERN stack, a powerful combination of MongoDB, Express.js, React, and Node.js, has emerged as a leading technology stack for building robust and scalable web applications. This project, "Elevating Online Shopping with the MERN Stack," aims to leverage the capabilities of the MERN stack to revolutionize the online shopping experience. The traditional online shopping process often leaves room for improvement, such as slow page load times, clunky interfaces, and limited personalization.

When an Online business to buy from another business, the process is called business -to-business(B2B) online shopping. A typical online cloth store enables the customer to browse the Firm’s range of products and services, view photos or images of the products, along with information about the product specifications, features and prices. Online stores typically enable shoppers to use “search” features to find specific models, brands or items Online customers must have access to the Internet and a valid method of payment in order to complete a transaction.

In summary, "Elevating Online Shopping with the MERN Stack" is poised to transform the way people shop online by harnessing the power of modern web development technologies. This project aims to set new standards in online shopping, providing users with a delightful, personalized, and efficient shopping journey.

1.2 Project Objectives and Goals

- The objective of the project is to develop a user friendly and efficient e-commerce platform that enables customers to browse, search for, select and purchase products or services online.
- It is time and energy saving.
- Person can easily login to site while sitting at home.

- Person can buy more than one product without going anywhere.
- The transactions are executed in offline mode, hence on-line data for shopping, Internet capture and modification is not possible.
- Manage the information of internet.
- Shows the information and description of the shopping.
- It deals with monitoring the information and transactions of bills.

1.3 Scope of the Project

The future of ecommerce website holds immense potential with continued technological advancements. AI-driven personalization, augmented reality shopping experiences, and seamless mobile integration will play pivotal roles. In conclusion, e-commerce will remain a dominant force in the global economy, offering convenience and limitless opportunities for business and consumers alike.

The future of e-commerce is expected to continue evolving with several key trends and opportunities.

- Mobile commerce
- AI and Personalization
- Voice Commerce
- Augmented Reality (AR) and Virtual Reality (VR)
- Blockchain for Supply Chain and Payment
- Global expansion
- Data Security
- Health and Wellness

The future of e-commerce will be shaped by technology, consumer preferences and market dynamics.

CHAPTER – 2
LITERATURE SURVEY

The design of the paper is influenced by the paper “future of e-commerce in India” by Chanana, N., & Goele, S. (2012) published in International Journal of Computing &Business Research, 8. The paper “e-commerce Application using MERN stack” by B.Mai (2020)[1].

“Developing an E-commerce website” helped to get the idea of the components included into the software solutions was taken. The MERN stack, a powerful combination of MongoDB, Express.js, React, and Node.js, has emerged as a leading technology stack for building robust and scalable web applications. This project, "Elevating Online Shopping with the MERN Stack," aims to leverage the capabilities of the MERN stack to revolutionize the online shopping experience.

The paper “Analysis of e-commerce and m-commerce: advantages, limitations and security issues” published in International Journal of Advanced Research in Computer and Communication Engineering, 2(6), 2360-2370 by Niranjanamurthy, M., Kavyashree, N., Jagannath, S., & Chahar, D.(2013).

The paper “Developing an e-commerce platform using MERN Stack” at Research Gate by Kritika Desai and Jinan Fidaihi. Bakos, Yannis (2001). "The Emerging Landscape for Retail E-Commerce". Journal of Economic Perspectives. 15 (1): 69–80. CiteSeerX 10.1.1.4.9128. doi:10.1257/jep.15.1.69. J. Kumar and V. Garg, "Security analysis of unstructured data in NOSQL MongoDB database," 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), Gurgaon, 2017, pp. 300-305. MongoDB official website for understanding what is MERN stack:https://www.mongodb.com/mern-stack

CHAPTER – 3
PROPOSED METHODOLOGY

The project “Elevating Online Shopping using MERN stack” is developed by using Iterative Waterfall model which is a popular version of development life cycle model for software engineering. It describes a development model that is linear and sequential.

The Iterative Waterfall Model is a traditional and linear approach to software development in the field of software engineering. It is a sequential and well-structured methodology that divides the software development process into distinct phases, where each phase must be completed before the next one begins. The Iterative Waterfall Model follows a strict top-down approach and is characterized by the following key phases:

1.Requirement Gathering and Analysis

2.System Design

3.Implementation (Coding)

4.Testing

5.Deployment

6.Maintenance and Support

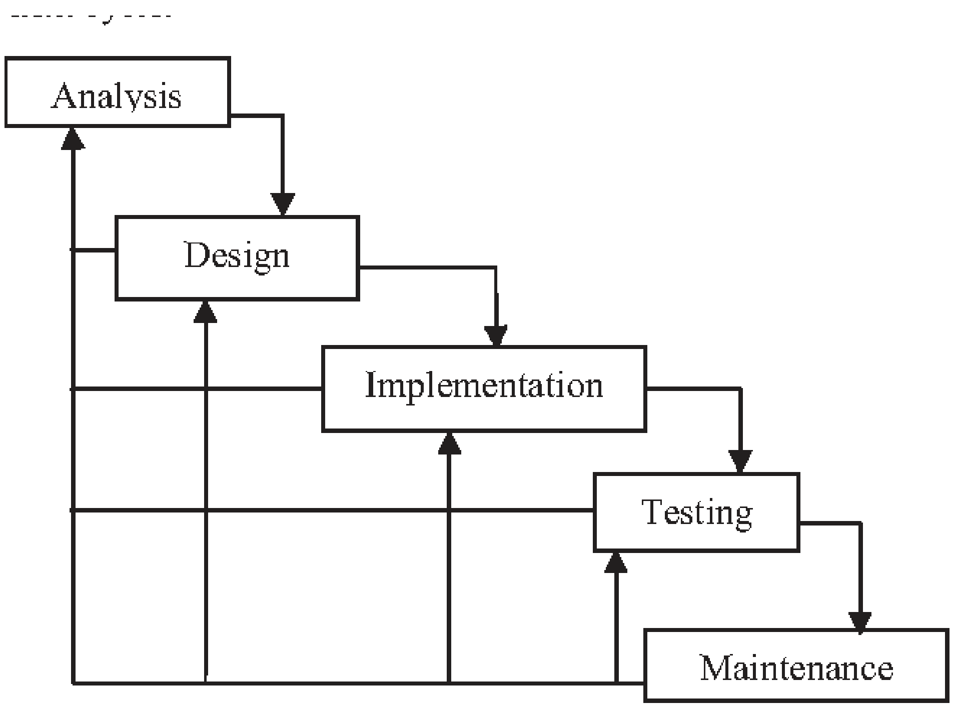


Figure 1

Reasons for choosing this Model

- **Clarity**: The well-defined phases and requirements documentation ensure a clear understanding of the project's scope and objectives.
- **Stability**: Once a phase is complete, it is difficult to make changes to it, which can help prevent scope creep.
- **Documentation**: Extensive documentation is produced at each stage, which can be beneficial for future reference and maintenance.
- **Suitability**: The Waterfall Model is often used for projects with well-understood and stable requirements, where changes are expected to be minimal.

System Architecture

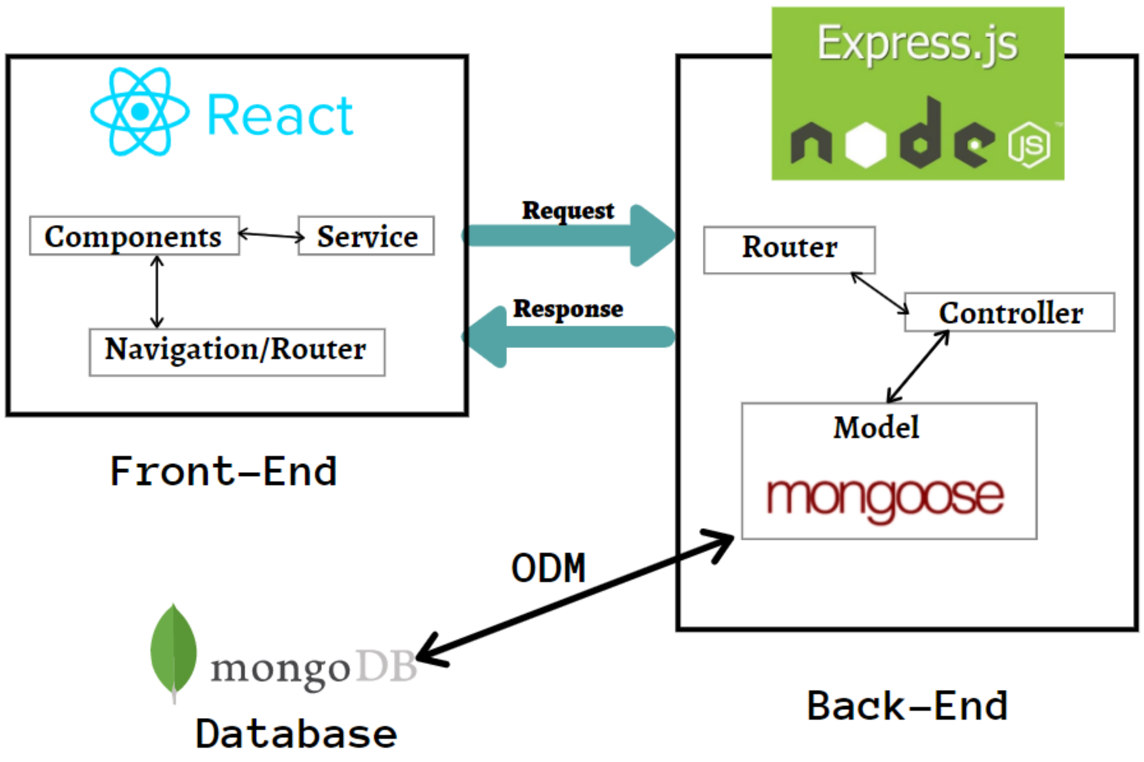


Figure 2

In a MERN project, the user interacts with the front-end built using React.js, which dynamically updates the UI based on user actions and application state. React components send HTTP requests to the Express.js server, which runs on Node.js, to perform CRUD operations. The Express.js server processes these requests, interacts with MongoDB through Mongoose to retrieve or modify data, and then sends the appropriate responses back to the React front-end. This seamless integration allows for a responsive, real-time web application where the front-end, back-end, and database work together to handle user interactions and data management efficiently.

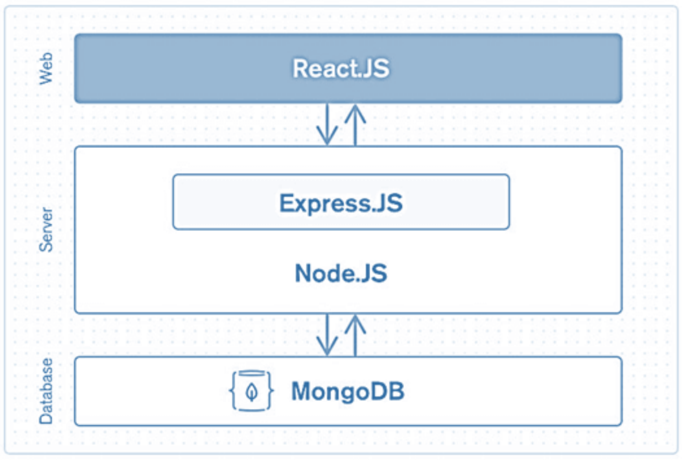


Figure 3

Our proposed work consists of following major parts – Frontend (HTML,CSS, JavaScript) and React as a frontend framework. Node JS and Express JS as the backend framework and the MongoDB as a database to provide access to real time data.

A. HTML

Hypertext Markup Language revision 5 (HTML5) is markup language for the structure and presentation of World Wide Web contents. It is client-side markup language and is platform independent. HTML5 consists of various tags which allows us to structure data in a definite way. Some of the most used tags are heading tags, paragraph tags, division tags, anchor tags etc. One of the helpful aspects of HTML is, it can embed programs written in a scripting language like JavaScript and CSS. HTML is not considered a programming language as it can't create dynamic functionality. Instead, with HTML, web users can create and structure sections, paragraphs, and links using elements, tags, and attribute

B. CSS3

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users. CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language**.**

There are 3 ways to write CSS in our HTML file.

- Inline CSS
- Internal CSS
- External CSS

C. JavaScript

JavaScript is a cross-platform, object-oriented language used to interact withweb pages (e.g., complex animation, clickable buttons, pop-up menus, etc.). There are also more advanced JavaScript versions such as Node.js, which allow you to add more functionality to a website rather than downloading files (such as real-time interaction between multiple computers). Within the host environment (for example, a web browser), JavaScript can be linked to its nativeobjects to provide system control over them.

D. React:

React is widely recognized as one of the most popular JavaScript libraries for building user interfaces. It was developed by Facebook and open-sourced in 2013, quickly gaining popularity among developers due to its efficiency, reusability, and component-based architecture.
One of the key features of React is its ability to create reusable UI components. Components in React are self-contained and independent entities that encapsulate the logic and rendering of a specific part of the user interface. This modular approach makes it easier to manage and maintain complex applications by breaking them down into smaller, more manageable pieces.
React utilizes a virtual DOM (Document Object Model) for efficient updates. The virtual DOM is a lightweight representation of the actual DOM, which is the browser's internal representation of the web page structure. React efficiently updates only the necessary parts of the virtual DOM and then synchronizes those changes with the actual DOM, reducing the number of expensive DOM operations and enhancing performance.
Another distinctive feature of React is JSX (JavaScript XML), a syntax extension that allows developers to write HTML-like code within JavaScript. JSX makes it easier to define the structure and appearance of components, combining HTML markup and

JavaScript logic in a single file. It provides a declarative way of defining user interfaces, where developers describe "what" they want to render, and React takes care of the "how" by efficiently updating the UI based on changes in application state. React is not a full-fledged framework, but rather a library focused on the view layer of an application. It can be used in combination with other libraries or frameworks such as Redux for managing application state, React Router for handling navigation, or Material-UI for ready-to-use UI components. This flexibility allows developers to choose and integrate various tools that best fit their project requirements.

E. Node JS

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to execute JavaScript code on the server side. Built on Chrome's V8 JavaScript engine, Node.js is designed for building scalable network applications due to its non-blocking, event-driven architecture.

- Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. This architecture is well-suited for I/O-heavy operations, such as handling multiple network connections or reading/writing to a database.
- While Node.js operates on a single thread, it uses asynchronous programming to handle multiple operations concurrently. This prevents the server from being blocked by long-running tasks, enhancing performance.
- Node.js comes with npm, the largest ecosystem of open-source libraries in the world. Developers can easily share, discover, and reuse code packages, which accelerates development and promotes code reusability.
- The vibrant Node.js community contributes to a vast array of libraries and frameworks, such as Express.js for web applications, enabling developers to quickly build and deploy applications.
- With Node.js, developers can use JavaScript for both client-side and server-side development, streamlining the development process and allowing for full-stack development within a single language.
- Node.js’s performance is enhanced by the V8 engine, which compiles JavaScript to native machine code, and its event-driven architecture, which optimizes handling of multiple simultaneous connections.
- Node.js is designed for scalable network applications. Its architecture handles

numerous concurrent connections efficiently, making it ideal for real-time applications like chat applications, online gaming, and collaborative tools.

- Node.js runs on various platforms, including Windows, Linux, and macOS, ensuring flexibility and broad compatibility for deployment.
- Node.js is often used to build microservices and RESTful APIs due to its lightweight and efficient nature. This architecture allows for modular and maintainable application design.

- Node.js is supported by a strong community and major corporations, ensuring continuous improvement and a wealth of resources for developers.

F. Express JS

Express.js is a minimalistic, flexible, and widely-used web application framework for Node.js, designed to simplify the development of server-side applications and APIs. It provides a robust set of features to build single-page, multi-page, and hybrid web applications. Here are some key aspects and features of Express.js:

- Express.js uses a series of middleware functions to handle requests and responses. Middleware functions can perform tasks such as parsing request bodies, logging, authentication, and error handling. This modular approach allows for a clean separation of concerns and reusability.
- Express.js provides a powerful and flexible routing system that allows developers to define routes for handling various HTTP methods and URL patterns. This makes it easy to organize the application's endpoints and manage complex URL structures.
- With its straightforward and expressive syntax, Express.js makes it easier to build RESTful APIs. It provides methods to define routes and handle HTTP requests, making API development faster and more efficient.
- Express.js can easily integrate with various template engines (such as Pug, EJS, and Handlebars) to generate dynamic HTML pages. This feature is useful for rendering views in server-side applications.
- Express.js can serve static files such as images, CSS, and JavaScript directly from the server. This is useful for delivering client-side assets efficiently.
- Express.js is highly extensible, allowing developers to use a wide range of third-party middleware and plugins available via npm. This extensibility makes it possible

to add functionality such as authentication, database integration, and more.

- Express.js provides robust configuration options and helpful debugging tools to streamline the development process. Environment-specific configurations and error-handling middleware enhance the development and maintenance of applications.
- Express.js supports the development of RESTful services and MVC (Model-View-Controller) architecture, promoting clean code organization and separation of concerns.
- As one of the most popular Node.js frameworks, Express.js has a large and active community. There is extensive documentation, numerous tutorials, and a wealth of open-source libraries and middleware that can be easily integrated into Express.js applications.

G.Mongo DB

MongoDB is a widely-used, open-source NoSQL database designed for scalability, performance, and ease of development. It stores data in a flexible, JSON-like format called BSON (Binary JSON), which allows for the representation of complex data structures and hierarchical relationships within a single record. Here are some key features and aspects of MongoDB:

- MongoDB uses a document-oriented data model, where data is stored in collections of documents. Each document is a JSON-like object consisting of key-value pairs. This format is highly flexible, allowing for varying data structures in different documents within the same collection.
- Unlike traditional relational databases, MongoDB does not require a predefined schema. Fields can vary from document to document, and new fields can be added without affecting existing documents. This schema flexibility allows for rapid development and iteration.
- MongoDB is designed to scale horizontally through a process called sharding, which distributes data across multiple servers. This allows the database to handle large volumes of data and high-throughput operations by adding more nodes to the cluster.
- MongoDB supports various types of indexes, including single-field, compound, geospatial, and text indexes. Indexes improve query performance by allowing the database to quickly locate and retrieve relevant documents.

- MongoDB provides a powerful aggregation framework for performing complex data processing and transformations. It includes operations such as filtering, grouping, sorting, and reshaping data, similar to SQL's GROUP BY and JOIN operations.
- MongoDB ensures high availability and data redundancy through replica sets. A replica set consists of a primary node and multiple secondary nodes that replicate the data from the primary. If the primary node fails, one of the secondary nodes is automatically promoted to primary, ensuring continued operation.
- MongoDB supports multi-document ACID transactions, allowing for atomic operations across multiple documents and collections. This ensures data integrity and consistency, particularly in complex applications requiring transactional support.
- MongoDB includes native support for geospatial data and queries. It can store location data and perform spatial queries, such as finding documents within a certain radius or area.
- MongoDB's design and architecture are optimized for high performance, with features such as in-memory computing, efficient storage engines, and support for various data models. This makes it suitable for applications requiring fast read and write operations.

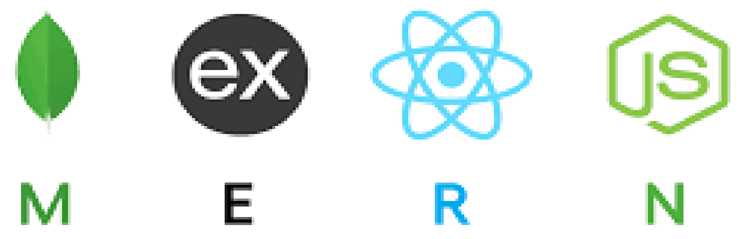


Figure 4

3.1. Software and Hardware Requirements

3.1.1. SOFTWARE REQUIREMENTS:

- Frontend Frameworks: HTML5, CSS3, JavaScript, ReactJS, Bootstrap.
- Backend Frameworks: NodeJS, ExpressJS.

- Database Tool: MongoDB.
- IDE: Visual Studio Code.
- Runtime Environment: Chrome, Mozilla Firefox.

3.1.2. HARDWARE REQUIREMENTS

- Processor: Intel core i3 / AMD Ryzen 3 or above.
- RAM: 8 GB DDR4 or above.
- HDD: 5 GB minimum free space.
- OS: Windows 7 or above (preferred Win11).
- Screen: VGA Monitor or Laptop Screen.
- Mouse: Standard Mouse of Laptop pointer.
- Keyboard: Standard Keyboard (QWERTY)
- Backup Device: CD or Pen Drive (PD) minimum 4GB.
- Power Backup: UPS / 4 Cell Batteries.

3.2. Process Description

The approach to solve the problem is 2-fold. First part is to create the front-end part and add the rules and logic to the shopping site. The second part is to integrate it with the database and enable a seamless online shopping experience. The frontend part will be done using HTML, CSS, JavaScript and React framework. The backend part will require the libraries of NodeJS and ExpressJS which will integrate with our database (MongoDB). Our Software comprises of 2 zones – Admin Zone and General Zone.

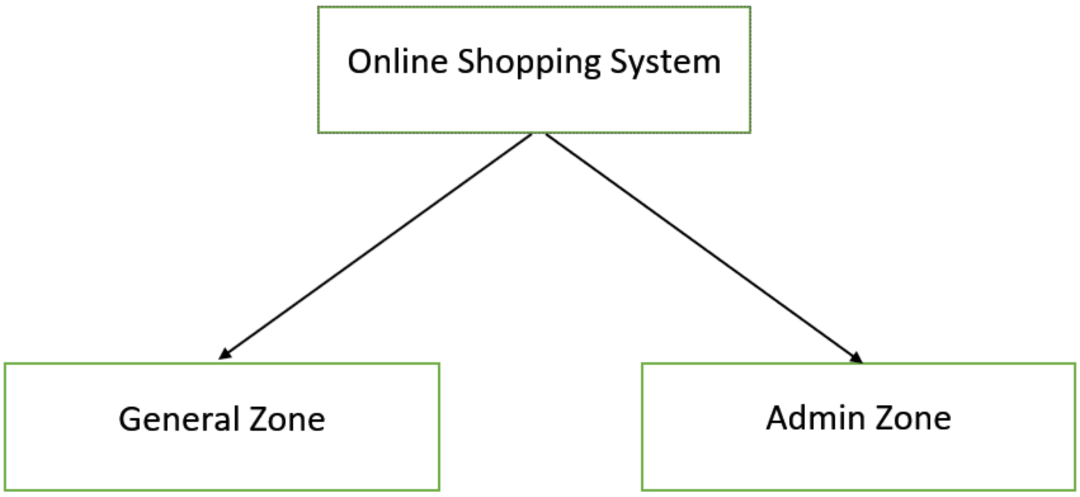


Figure 5

3.2.1.Working with General Zone

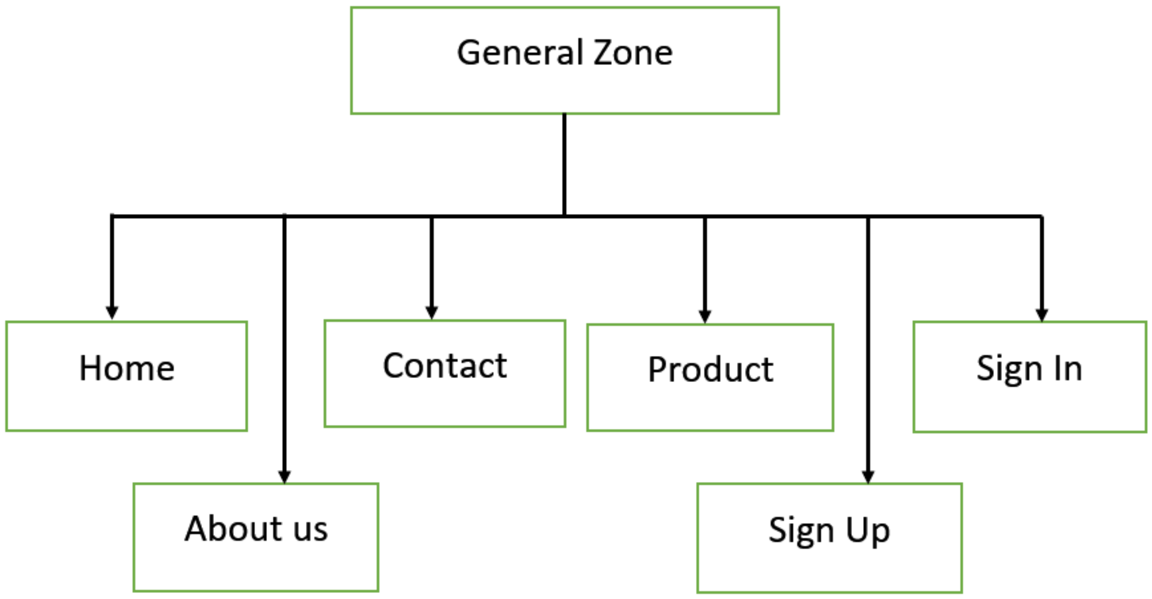


Figure 6

HOME- The homepage of a shopping website typically features a layout and elements designed to engage visitors, showcase products, and encourage them to explore the online store. It consists of Header, Hero section, Featured Products, Deals and Promotions, Customer Reviews/Testimonials, Featured Brands, Trending or Recommended Products, Additional Information, Footer.

ABOUT US- The "About Us" page of an online shopping website provides valuable information about the company or brand, its mission, values, history, and team. This page serves to build trust and establish a connection with visitors.

PRODUCT- The product page of an online shopping website is a critical part of the e-commerce experience. It provides detailed information about a specific product and aims to persuade visitors to make a purchase.

CONTACT US- The "Contact Us" page on an online shopping website is crucial for establishing communication between the business and its customers. It should make it easy for visitors to get in touch with the company for inquiries, support, or feedback.

SIGN UP- The "sign-up" or "Registration" page on an online shopping website is a critical part of the user experience. It allows visitors to create accounts, which can lead to increased engagement, personalized shopping experiences, and order history tracking.

SIGN IN- The "Sign-In" or "Login" page on an online shopping website allows registered users to access their accounts and begin shopping. The "Sign-In" page should be user-friendly, with clear instructions and an uncomplicated layout. It's essential to prioritize security and privacy, especially when handling sensitive login information.

3.2.2. Working with Admin Zone

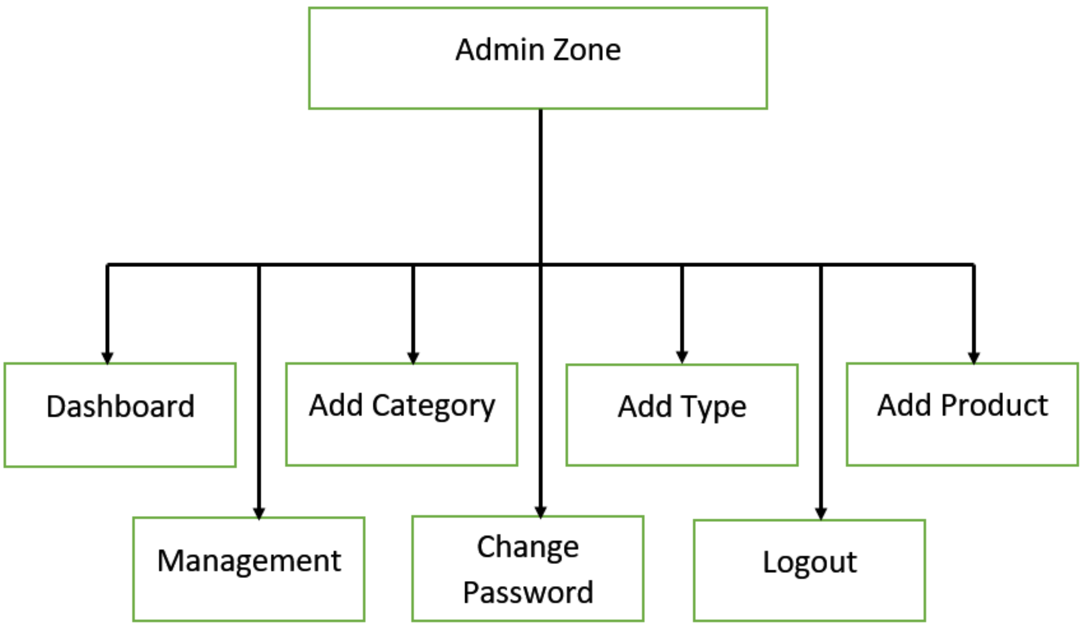


Figure 7

DASHBORAD- Dashboard of the Master Zone is a static page containing a well displayed menu of Admin Zone with active hyperlink to related page. You can use some special effects and animation to it more interactive.

ADD CATEGORY- This can be used to add the categories in the product. The categories section of an eCommerce website is a crucial component that helps organize and display products in a user-friendly manner. It typically appears in the website's navigation menu or on the homepage and serves as a way for customers to quickly find the types of products they are interested in.

ADD TYPE- From here we can add type on behalf of Category. (Category and Type).

ADD PRODUCT- Adding a product section to an eCommerce website is a fundamental step in setting up an online store. This section is where you showcase your products, provide detailed information, and enable customers to make purchases. This section involves Product Listing, Product details, Images and Media, Pricing and Availability, Add to cart functionality, Related Products, Search and Filter, Security and Customer Support.

MANAGEMENT- This can be further divided into 4 parts-

- View orders made by the user.
- Products added by the admin and its quantity.
- User details who are registered from the general zone.
- Records the enquiries from the contact page.

CHANGE PASSWORD- From here the admin can change password, provide previous. This password will be used for next time login.

LOGOUT- Logout page will destroy the created session at the time of login in admin zone and redirect at login page.

3.3. Use Case Diagram

A use case diagram is a visual representation in the field of software engineering that illustrates the interactions between various actors (users or systems) and a system (software application or process). It helps in defining and understanding the functional requirements of the system by showcasing the specific actions, or "use cases," that the system performs in response to interactions with these actors.

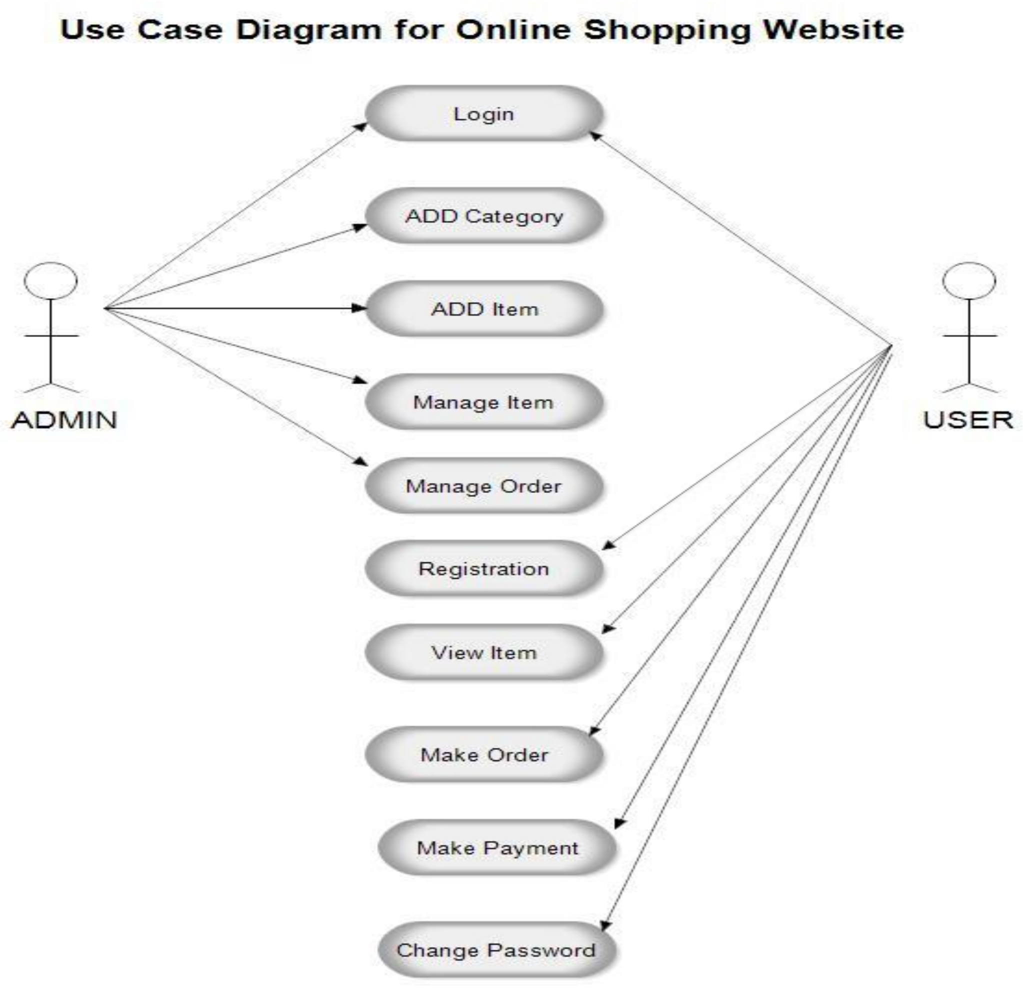


Figure 8

3.4. Activity Diagram

An activity diagram is a type of UML (Unified Modelling Language) diagram used in software engineering to model the dynamic aspects of a system or process. It visually represents the workflow or flow of activities within a system, process, or use case.

In an activity diagram, we use various symbols and notations to represent activities, actions, decisions, forks, joins, and the flow of control. Arrows or transitions indicate the order in which these activities occur, and decision points (diamond shapes) represent conditions that determine the flow of control.

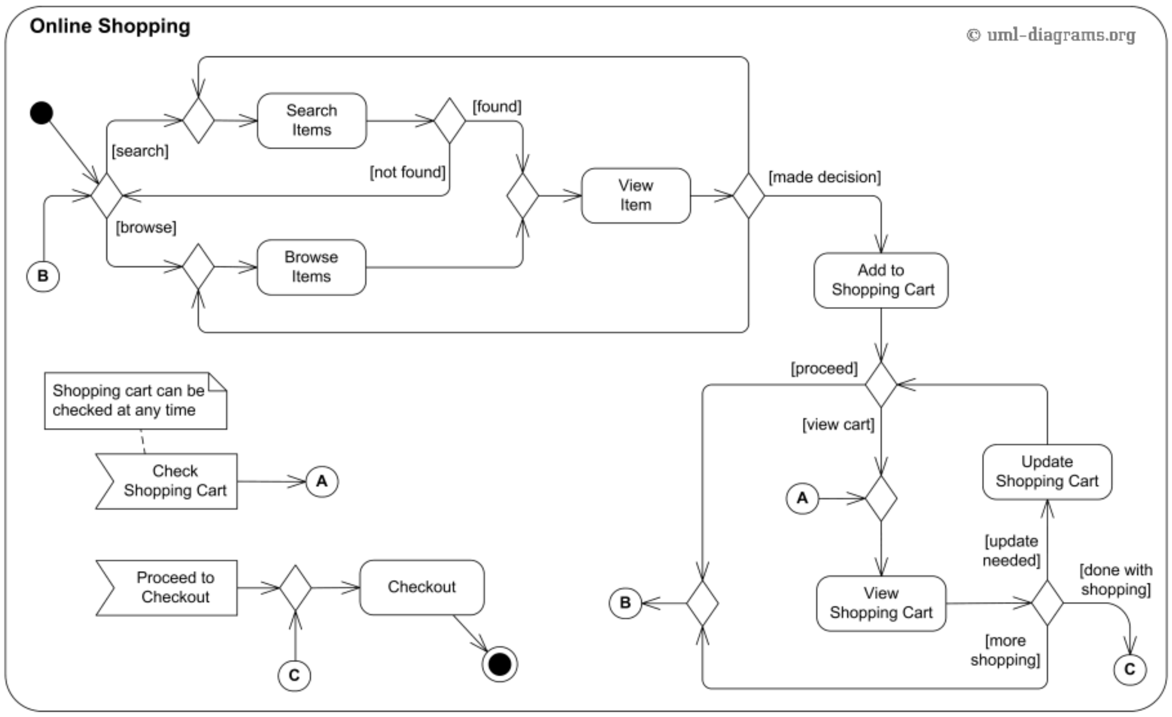

Online Shopping
© uml-diagrams.org
Search Items
[found]
[search]
[not found]
View Item
[made decision]
[browse]
Browse Items
B
Add to Shopping Cart
[proceed]
Shopping cart can be checked at any time
[view cart]
Update Shopping Cart
Check Shopping Cart
A
A
[update needed]
[done with shopping]
Proceed to Checkout
Checkout
B
View Shopping Cart
[more shopping]
C
C

Figure 9

CHAPTER 4

IMPLEMENTATION OVERVIEW

The code of the project is divided into 3 sections

- Frontend section
- Backend section
- Admin section

4.1. Frontend section

Index.html file

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
       <link   href="https://fonts.googleapis.com/css2?family=Poppins&display=swap"
rel="stylesheet">
    <title>Shopper</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

App.js

```
import Navbar from "./Components/Navbar/Navbar";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Shop from "./Pages/Shop";
import Cart from "./Pages/Cart";
import Product from "./Pages/Product";
import Footer from "./Components/Footer/Footer";
import ShopCategory from "./Pages/ShopCategory";
import women_banner from "./Components/Assets/banner_women.png";
import men_banner from "./Components/Assets/banner_mens.png";
import kid_banner from "./Components/Assets/banner_kids.png";
import LoginSignup from "./Pages/LoginSignup";

function App() {

  return (
    <div>
      <Router>
        <Navbar />
        <Routes>
          <Route path="/" element={<Shop gender="all" />} />
          <Route path="/mens" element={<ShopCategory banner={men_banner} category="men" />} />
          <Route path="/womens" element={<ShopCategory banner={women_banner} category="women" />} />
          <Route path="/kids" element={<ShopCategory banner={kid_banner} category="kid" />} />
          <Route path='/product' element={<Product />}>
            <Route path=':productId' element={<Product />} />
          </Route>
          <Route path="/cart" element={<Cart />} />
          <Route path="/login" element={<LoginSignup/>} />
        </Routes>
        <Footer />
      </Router>
    </div>
  );
}

export default App;
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import ShopContextProvider from './Context/ShopContext';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <ShopContextProvider>
      <App />
    </ShopContextProvider>
);
```

The project is divided into various components such as Navbar, Item, Footer, Description Box, Cart Items, Related Products etc

Navbar.jsx

```
import React, { useContext, useRef, useState } from 'react'
import './Navbar.css'
import { Link } from 'react-router-dom'
import logo from '../Assets/logo.png'
import cart_icon from '../Assets/cart_icon.png'
import { ShopContext } from '../../Context/ShopContext'
import nav_dropdown from '../Assets/nav_dropdown.png'
```

```
const Navbar = () => {

  let [menu,setMenu] = useState("shop");
  const {getTotalCartItems} = useContext(ShopContext);

  const menuRef = useRef();

  const dropdown_toggle = (e) => {
    menuRef.current.classList.toggle('nav-menu-visible');
    e.target.classList.toggle('open');
  }

  return (
    <div className='nav'>
      <Link to='/' onClick={()=>{setMenu("shop")}} style={{ textDecoration: 'none' }}
className="nav-logo">
        <img src={logo} alt="logo" />
        <p>SHOPPER</p>
      </Link>
                <img onClick={dropdown_toggle} className='nav-dropdown'
src={nav_dropdown} alt="" />
      <ul ref={menuRef} className="nav-menu">
          <li onClick={()=>{setMenu("shop")}}><Link to='/' style={{ textDecoration:
'none' }}>Shop</Link>{menu==="shop"?<hr/>:<></>}</li>
              <li onClick={()=>{setMenu("mens")}}><Link to='/mens' style={{
textDecoration: 'none' }}>Men</Link>{menu==="mens"?<hr/>:<></>}</li>
            <li onClick={()=>{setMenu("womens")}}><Link to='/womens' style={{
textDecoration: 'none' }}>Women</Link>{menu==="womens"?<hr/>:<></>}</li>
        <li onClick={()=>{setMenu("kids")}}><Link to='/kids' style={{ textDecoration:
'none' }}>Kids</Link>{menu==="kids"?<hr/>:<></>}</li>
      </ul>
      <div className="nav-login-cart">
        {localStorage.getItem('auth-token')
```

```
                    ?<button    onClick={()=>{localStorage.removeItem('auth-token');window.location.replace("/");}}>Logout</button>
                  :<Link    to='/login'    style={{    textDecoration:    'none' }}><button>Login</button></Link>}
        <Link to="/cart"><img src={cart_icon} alt="cart"/></Link>
        <div className="nav-cart-count">{getTotalCartItems()}</div>
      </div>
    </div>
  )
}

export default Navbar
```

Item.jsx

```
import React from 'react'
import './Item.css'
import { Link } from 'react-router-dom'

const Item = (props) => {
  return (
    <div className='item'>
        <Link  to={`/product/${props.id}`}  style={{  textDecoration:  'none'  }}><img onClick={window.scrollTo(0, 0)} src={props.image} alt="products" /></Link>
      <p>{props.name}</p>
      <div className="item-prices">
        <div className="item-price-new">${props.new_price}</div>
        <div className="item-price-old">${props.old_price}</div>
      </div>
    </div>
  )
}

export default Item
```

CartItems.jsx

```
import React, { useContext } from "react";
import "./CartItems.css";
import cross_icon from "../Assets/cart_cross_icon.png";
import { ShopContext } from "../../Context/ShopContext";

const CartItems = () => {
  const {products} = useContext(ShopContext);
        const        {cartItems,removeFromCart,getTotalCartAmount}        =
useContext(ShopContext);

  return (
    <div className="cartitems">
      <div className="cartitems-format-main">
        <p>Products</p>
        <p>Title</p>
        <p>Price</p>
        <p>Quantity</p>
        <p>Total</p>
        <p>Remove</p>
      </div>
      <hr />
      {products.map((e)=>{

        if(cartItems[e.id]>0)
        {
          return  <div>
                    <div className="cartitems-format-main cartitems-format">
                      <img className="cartitems-product-icon" src={e.image} alt="" />
                      <p cartitems-product-title>{e.name}</p>
                      <p>${e.new_price}</p>
                      <button className="cartitems-quantity">{cartItems[e.id]}</button>
```

```
                  <p>${e.new_price*cartItems[e.id]}</p>
                  <img onClick={()=>{removeFromCart(e.id)}} className="cartitems-
remove-icon" src={cross_icon} alt="" />
                </div>
                <hr />
              </div>;
          }
          return null;
        })}

        <div className="cartitems-down">
          <div className="cartitems-total">
            <h1>Cart Totals</h1>
            <div>
              <div className="cartitems-total-item">
                <p>Subtotal</p>
                <p>${getTotalCartAmount()}</p>
              </div>
              <hr />
              <div className="cartitems-total-item">
                <p>Shipping Fee</p>
                <p>Free</p>
              </div>
              <hr />
              <div className="cartitems-total-item">
                <h3>Total</h3>
                <h3>${getTotalCartAmount()}</h3>
              </div>
            </div>
            <button>PROCEED TO CHECKOUT</button>
          </div>
          <div className="cartitems-promocode">
            <p>If you have a promo code, Enter it here</p>
            <div className="cartitems-promobox">
              <input type="text" placeholder="promo code" />
              <button>Submit</button>
            </div>
          </div>
        </div>
      </div>
     );
    };

    export default CartItems;
```

DescriptionBox.jsx

```
import React from "react";
import "./DescriptionBox.css";

const DescriptionBox = () => {
 return (
  <div className="descriptionbox">
   <div className="descriptionbox-navigator">
    <div className="descriptionbox-nav-box">Description</div>
    <div className="descriptionbox-nav-box fade">Reviews (122)</div>
   </div>
   <div className="descriptionbox-description">
    <p>
     An e-commerce website is an online platform that facilitates the
     buying and selling of products or services over the internet. It
     serves as a virtual marketplace where businesses and individuals can
     showcase their products, interact with customers, and conduct
     transactions without the need for a physical presence. E-commerce
     websites have gained immense popularity due to their convenience,
     accessibility, and the global reach they offer.
    </p>
    <p>
```

```
          E-commerce websites typically display products or services along with
          detailed descriptions, images, prices, and any available variations
          (e.g., sizes, colors). Each product usually has its own dedicated page
          with relevant information.
        </p>
      </div>
    </div>
  );
};

export default DescriptionBox;
```

ProductDisplay.jsx

```
import React, { useContext } from "react";
import "./ProductDisplay.css";
import star_icon from "../Assets/star_icon.png";
import star_dull_icon from "../Assets/star_dull_icon.png";
import { ShopContext } from "../../Context/ShopContext";

const ProductDisplay = (props) => {

  const {product} = props;
  const {addToCart} = useContext(ShopContext);

  return (
    <div className="productdisplay">
      <div className="productdisplay-left">
        <div className="productdisplay-img-list">
          <img src={product.image} alt="img" />
          <img src={product.image} alt="img" />
          <img src={product.image} alt="img" />
          <img src={product.image} alt="img" />
        </div>
        <div className="productdisplay-img">
          <img className="productdisplay-main-img" src={product.image} alt="img"
/>
        </div>
      </div>
      <div className="productdisplay-right">
        <h1>{product.name}</h1>
        <div className="productdisplay-right-stars">
          <img src={star_icon} alt="" />
          <img src={star_icon} alt="" />
          <img src={star_icon} alt="" />
          <img src={star_icon} alt="" />
          <img src={star_dull_icon} alt="" />
          <p>(122)</p>
        </div>
        <div className="productdisplay-right-prices">
          <div className="productdisplay-right-price-old">${product.old_price}</div>
                                  <div          className="productdisplay-right-price-
new">${product.new_price}</div>
        </div>
        <div className="productdisplay-right-description">
          A lightweight, usually knitted, pullover shirt, close-fitting and with
          a round neckline and short sleeves, worn as an undershirt or outer
          garment.
        </div>
        <div className="productdisplay-right-size">
          <h1>Select Size</h1>
          <div className="productdisplay-right-sizes">
            <div>S</div>
            <div>M</div>
            <div>L</div>
            <div>XL</div>
            <div>XXL</div>
          </div>
        </div>
        <button onClick={()=>{addToCart(product.id)}}>ADD TO CART</button>
            <p  className="productdisplay-right-category"><span>Category  :</span>
Women, T-shirt, Crop Top</p>
         <p className="productdisplay-right-category"><span>Tags :</span> Modern,
Latest</p>
      </div>
    </div>
  );
};

export default ProductDisplay;
```

RelatedProducts.jsx

```
import React from 'react'
import './RelatedProducts.css'
import Item from '../Item/Item'
import data_product from '../Assets/data'

const RelatedProducts = () => {
  return (
    <div className='relatedproducts'>
      <h1>Related Products</h1>
      <hr />
      <div className="relatedproducts-item">
        {data_product.map((item)=>{
                              return    <Item    id={item.id}    name={item.name}
image={item.image}  new_price={item.new_price} old_price={item.old_price}/>
        })}
      </div>
    </div>
  )
}
```

```
export default RelatedProducts
```

Footer.jsx

```
import React from 'react'
import './Footer.css'

import footer_logo from '../Assets/logo_big.png'
import instagram_icon from '../Assets/instagram_icon.png'
import pintrest_icon from '../Assets/pintester_icon.png'
import whatsapp_icon from '../Assets/whatsapp_icon.png'

const Footer = () => {
  return (
    <div className='footer'>
      <div className="footer-logo">
        <img src={footer_logo} alt="" />
        <p>SHOPPER</p>
      </div>
      <ul className="footer-links">
        <li>Company</li>
        <li>Products</li>
        <li>Offices</li>
        <li>About</li>
        <li>Contact</li>
      </ul>
      <div className="footer-social-icons">
        <div className="footer-icons-container">
            <img src={instagram_icon} alt="" />
        </div>
        <div className="footer-icons-container">
            <img src={pintrest_icon} alt="" />
        </div>
        <div className="footer-icons-container">
            <img src={whatsapp_icon} alt="" />
        </div>
      </div>
      <div className="footer-copyright">
        <hr />
        <p>Copyright @ 2023 - All Right Reserved.</p>
      </div>
    </div>
  )
}

export default Footer
```

Various pages to fetch data to frontend from the database.

LoginSignup.jsx

```
import React, { useState } from "react";
import "./CSS/LoginSignup.css";

const LoginSignup = () => {

  const [state,setState] = useState("Login");
  const [formData,setFormData] = useState({username:"",email:"",password:""});

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

  const login = async () => {
    let dataObj;
    await fetch('http://localhost:4000/login', {
      method: 'POST',
      headers: {
        Accept:'application/form-data',
        'Content-Type':'application/json',
      },
      body: JSON.stringify(formData),
    })
      .then((resp) => resp.json())
      .then((data) => {dataObj=data});
      console.log(dataObj);
      if (dataObj.success) {
        localStorage.setItem('auth-token',dataObj.token);
        window.location.replace("/");
      }
      else
      {
        alert(dataObj.errors)
      }
  }

  const signup = async () => {
    let dataObj;
    await fetch('http://localhost:4000/signup', {
      method: 'POST',
      headers: {
        Accept:'application/form-data',
        'Content-Type':'application/json',
      },
      body: JSON.stringify(formData),
    })
      .then((resp) => resp.json())
      .then((data) => {dataObj=data});

      if (dataObj.success) {
        localStorage.setItem('auth-token',dataObj.token);
```

```
        window.location.replace("/");
      }
      else
      {
        alert(dataObj.errors)
      }
  }

  return (
    <div className="loginsignup">
      <div className="loginsignup-container">
        <h1>{state}</h1>
        <div className="loginsignup-fields">
              {state==="Sign Up"?<input type="text" placeholder="Your name" name="username" value={formData.username} onChange={changeHandler}/>:<></>}
              <input type="email" placeholder="Email address" name="email" value={formData.email} onChange={changeHandler}/>
              <input type="password" placeholder="Password" name="password" value={formData.password} onChange={changeHandler}/>
        </div>

        <button onClick={()=>{state==="Login"?login():signup()}}>Continue</button>

        {state==="Login"?
              <p className="loginsignup-login">Create an account? <span onClick={()=>{setState("Sign Up")}}>Click here</span></p>
            :<p className="loginsignup-login">Already have an account? <span onClick={()=>{setState("Login")}}>Login here</span></p>}

        <div className="loginsignup-agree">
          <input type="checkbox" name="" id="" />
          <p>By continuing, i agree to the terms of use & privacy policy.</p>
        </div>
      </div>
    </div>
  );
};

export default LoginSignup;
```

Shop.jsx

```
import React, { useEffect, useState } from 'react'
import Hero from '../Components/Hero/Hero'
import Popular from '../Components/Popular/Popular'
import Offers from '../Components/Offers/Offers'
import NewCollections from '../Components/NewCollections/NewCollections'
import NewsLetter from '../Components/NewsLetter/NewsLetter'

const Shop = () => {

  const [popular, setPopular] = useState([]);
  const [newcollection, setNewCollection] = useState([]);

  const fetchInfo = () => {
    fetch('http://localhost:4000/popularinwomen')
            .then((res) => res.json())
            .then((data) => setPopular(data))
    fetch('http://localhost:4000/newcollections')
            .then((res) => res.json())
            .then((data) => setNewCollection(data))
    }

    useEffect(() => {
      fetchInfo();
    }, [])
  return (
    <div>
      <Hero/>
      <Popular data={popular}/>
      <Offers/>
      <NewCollections data={newcollection}/>
      <NewsLetter/>
    </div>
  )
}

export default Shop
```

ShopCategory.jsx

```
import React, { useEffect, useState } from "react";
import "./CSS/ShopCategory.css";
import dropdown_icon from '../Components/Assets/dropdown_icon.png'
import Item from "../Components/Item/Item";
import { Link } from "react-router-dom";

const ShopCategory = (props) => {

  const [allproducts, setAllProducts] = useState([]);

  const fetchInfo = () => {
    fetch('http://localhost:4000/allproducts')
            .then((res) => res.json())
            .then((data) => setAllProducts(data))
    }

    useEffect(() => {
      fetchInfo();
    }, [])

  return (
    <div className="shopcategory">
      <img src={props.banner} className="shopcategory-banner" alt="" />
      <div className="shopcategory-indexSort">
        <p><span>Showing 1 - 12</span> out of 54 Products</p>
            <div className="shopcategory-sort">Sort by  <img src={dropdown_icon}
alt="" /></div>
      </div>
      <div className="shopcategory-products">
        {allproducts.map((item,i) => {
            if(props.category===item.category)
            {
                        return <Item id={item.id} key={i} name={item.name}
image={item.image}  new_price={item.new_price} old_price={item.old_price}/>;
            }
            else
            {
              return null;
            }
        })}
      </div>
      <div className="shopcategory-loadmore">
      <Link to='/' style={{ textDecoration: 'none' }}>Explore More</Link>
      </div>
    </div>
  );
};

export default ShopCategory;
```

ShopContext.jsx

```
import React, { createContext, useEffect, useState } from "react";

export const ShopContext = createContext(null);

const ShopContextProvider = (props) => {

  const [products,setProducts] = useState([]);

  const getDefaultCart = () => {
    let cart = {};
    for (let i = 0; i < 300; i++) {
      cart[i] = 0;
    }
    return cart;
  };

  const [cartItems, setCartItems] = useState(getDefaultCart());

  useEffect(() => {
    fetch('http://localhost:4000/allproducts')
          .then((res) => res.json())
          .then((data) => setProducts(data))

    if(localStorage.getItem("auth-token"))
    {
      fetch('http://localhost:4000/getcart', {
      method: 'POST',
      headers: {
        Accept:'application/form-data',
        'auth-token':`${localStorage.getItem("auth-token")}`,
        'Content-Type':'application/json',
      },
      body: JSON.stringify(),
    })
```

```
      .then((resp) => resp.json())
      .then((data) => {setCartItems(data)});
    }

}, [])

  const getTotalCartAmount = () => {
    let totalAmount = 0;
    for (const item in cartItems) {
      if (cartItems[item] > 0) {
        let itemInfo = products.find((product) => product.id === Number(item));
        totalAmount += cartItems[item] * itemInfo.new_price;
      }
    }
    return totalAmount;
  };

  const getTotalCartItems = () => {
    let totalItem = 0;
    for (const item in cartItems) {
      if (cartItems[item] > 0) {
        totalItem += cartItems[item];;
      }
    }
    return totalItem;
  };

  const addToCart = (itemId) => {
    setCartItems((prev) => ({ ...prev, [itemId]: prev[itemId] + 1 }));
    if(localStorage.getItem("auth-token"))
    {
      fetch('http://localhost:4000/addtocart', {
      method: 'POST',
      headers: {
        Accept:'application/form-data',
        'auth-token':`${localStorage.getItem("auth-token")}`,
        'Content-Type':'application/json',
      },
      body: JSON.stringify({"itemId":itemId}),
    })
      .then((resp) => resp.json())
      .then((data) => {console.log(data)});
    }
  };

  const removeFromCart = (itemId) => {
    setCartItems((prev) => ({ ...prev, [itemId]: prev[itemId] - 1 }));
    if(localStorage.getItem("auth-token"))
    {
      fetch('http://localhost:4000/removefromcart', {
      method: 'POST',
      headers: {
        Accept:'application/form-data',
        'auth-token':`${localStorage.getItem("auth-token")}`,
        'Content-Type':'application/json',
      },
      body: JSON.stringify({"itemId":itemId}),
    })
      .then((resp) => resp.json())
      .then((data) => {console.log(data)});
    }
  };

    const contextValue = {products, getTotalCartItems, cartItems, addToCart,
removeFromCart, getTotalCartAmount };
  return (
    <ShopContext.Provider value={contextValue}>
      {props.children}
    </ShopContext.Provider>
  );
};

export default ShopContextProvider;
```

4.2. Admin Section

App.js

```
import { BrowserRouter } from "react-router-dom";
import Footer from "./Components/Footer/Footer";
import Navbar from "./Components/Navbar/Navbar";
import Admin from "./Pages/Admin";

function App() {
  return (
    <BrowserRouter>
      <div>
        <Navbar />
        <Admin />
        <Footer />
      </div>
    </BrowserRouter>
  );
}

export default App;
```

Various components in the admin panel are Navbar, Add Product, List Product, Sidebar and Footer.

Navbar.jsx

```
import React from 'react'
import './Navbar.css'
import navlogo from '../Assets/nav-logo.svg'
import navprofileIcon from '../Assets/nav-profile.svg'

const Navbar = () => {
  return (
    <div className='navbar'>
```

```
      <img src={navlogo} className='nav-logo' alt="" />
    </div>
  )
}

export default Navbar
```

AddProduct.jsx

```
import React, { useState } from "react";
import "./AddProduct.css";
import upload_area from "../Assets/upload_area.svg";

const AddProduct = () => {

  const[image,setImage] = useState(false);
  const [productDetails,setProductDetails] = useState({
      name:"",
      image:"",
      category:"women",
      new_price:"",
      old_price:""
  });

  const AddProduct = async () => {

    let dataObj;
    let product = productDetails;

    let formData = new FormData();
    formData.append('product', image);

    await fetch('http://localhost:4000/upload', {
      method: 'POST',
    headers: {
      Accept:'application/json',
    },
    body: formData,
  })
    .then((resp) => resp.json())
    .then((data) => {dataObj=data});

  if (dataObj.success) {
    product.image = dataObj.image_url;
    console.log(product);
    await fetch('http://localhost:4000/addproduct', {
    method: 'POST',
    headers: {
      Accept:'application/json',
      'Content-Type':'application/json',
    },
    body: JSON.stringify(product),
  })
    .then((resp) => resp.json())
    .then((data) => {data.success?alert("Product Added"):alert("Failed")});

  }
}

const changeHandler = (e) => {
  console.log(e);
  setProductDetails({...productDetails,[e.target.name]:e.target.value});
  }

const imageHandler = (e) => {
  setImage(e.target.files[0]);
  }
  return (
    <div className="addproduct">
      <div className="addproduct-itemfield">
        <p>Product title</p>
                <input type="text" name="name" value={productDetails.name}
onChange={(e)=>{changeHandler(e)}} placeholder="Type here" />
      </div>
      <div className="addproduct-price">
        <div className="addproduct-itemfield">
          <p>Price</p>
              <input type="text" name="old_price" value={productDetails.old_price}
onChange={(e)=>{changeHandler(e)}} placeholder="Type here" />
        </div>
        <div className="addproduct-itemfield">
          <p>Offer Price</p>
             <input type="text" name="new_price" value={productDetails.new_price}
onChange={(e)=>{changeHandler(e)}} placeholder="Type here" />
        </div>
      </div>
      <div className="addproduct-itemfield">
        <p>Product category</p>
          <select value={productDetails.category} name="category" className="add-
product-selector" onChange={changeHandler}>
          <option value="women">Women</option>
          <option value="men">Men</option>
          <option value="kid">Kid</option>
        </select>
      </div>
      <div className="addproduct-itemfield">
        <p>Product title</p>
        <label for="file-input">
                              <img className="addproduct-thumbnail-img"
src={!image?upload_area:URL.createObjectURL(image)} alt="" />
        </label>
        <input onChange={(e)=>{imageHandler(e)}} type="file" name="image" id="file-input" hidden />
      </div>
      <button className="addproduct-btn" onClick={()=>{AddProduct()}}>ADD</button>
    </div>
  );
};

export default AddProduct;
```

ListProduct.jsx

```
import React, { useEffect, useState } from "react";
import "./ListProduct.css";
import cross_icon from '../Assets/cross_icon.png'

const ListProduct = () => {
  const [allproducts, setAllProducts] = useState([]);

  const fetchInfo = () => {
    fetch('http://localhost:4000/allproducts')
            .then((res) => res.json())
            .then((data) => setAllProducts(data))
    }

    useEffect(() => {
      fetchInfo();
    }, [])

    const removeProduct = async (id) => {
      await fetch('http://localhost:4000/removeproduct', {
      method: 'POST',
      headers: {
```

```
      Accept:'application/json',
      'Content-Type':'application/json',
    },
    body: JSON.stringify({id:id}),
  })

  fetch('http://localhost:4000/allproducts')
  .then((res) => res.json())
  .then((data) => setAllProducts(data))

  }

return (
  <div className="listproduct">
    <h1>All Products List</h1>
    <div className="listproduct-format-main">
        <p>Products</p>
        <p>Title</p>
        <p>Old Price</p>
        <p>New Price</p>
        <p>Category</p>
        <p>Remove</p>
      </div>
    <div className="listproduct-allproducts">
      <hr />
      {allproducts.map((e) => {
        return (
          <div>
            <div className="listproduct-format-main listproduct-format">
              <img className="listproduct-product-icon" src={e.image} alt="" />
              <p cartitems-product-title>{e.name}</p>
              <p>${e.old_price}</p>
              <p>${e.new_price}</p>
              <p>{e.category}</p>
```

```
                                    <img      className="listproduct-remove-icon"
onClick={()=>{removeProduct(e.id)}} src={cross_icon} alt="" />
              </div>
              <hr />
            </div>
          );
        })}
      </div>
    </div>
  );
};

export default ListProduct;
```

Sidebar.jsx

```
import React from 'react'
import './Sidebar.css'
import add_product_icon from '../Assets/Product_Cart.svg'
import list_product_icon from '../Assets/Product_list_icon.svg'
import { Link } from 'react-router-dom'

const Sidebar = () => {
  return (
    <div className='sidebar'>
      <Link to='/addproduct' style={{ textDecoration: 'none' }}>
        <div className="sidebar-item">
          <img src={add_product_icon} alt="" />
          <p>Add Product</p>
        </div>
      </Link>
      <Link to='/listproduct' style={{ textDecoration: 'none' }}>
        <div className="sidebar-item">
          <img src={list_product_icon} alt="" />
          <p>Product List</p>
        </div>
      </Link>

    </div>
  )
}

export default Sidebar
```

Footer.jsx

```
import React from 'react'
import './Footer.css'
const Footer = () => {
  return (
    <div className='footer'>
      <hr />
      <p>Copyright @ 2023 - All Right Reserved.</p>
    </div>
  )
}

export default Footer
```

Various Pages used are Admin and Login Page

Admin.jsx

```
import React from "react";
import "./CSS/Admin.css";
import Sidebar from "../Components/Sidebar/Sidebar";
import AddProduct from "../Components/AddProduct/AddProduct";
import { Route, Routes } from "react-router-dom";
import ListProduct from "../Components/ListProduct/ListProduct";
```

```
const Admin = () => {

  return (
    <div className="admin">
      <Sidebar />
      <Routes>
        <Route path="/addproduct" element={<AddProduct />} />
        <Route path="/listproduct" element={<ListProduct />} />
      </Routes>
    </div>
  );
};

export default Admin;
```

Login.jsx

```
import React from 'react'

const Login = () => {
  return (
    <div className='login'>

    </div>
  )
}

export default Login
```

4.3. Backend Section

Index.js

```
const port = 4000;
const express = require("express");
const app = express();
const mongoose = require("mongoose");
const jwt = require("jsonwebtoken");
const multer = require("multer");
const path = require("path");
const cors = require("cors");

app.use(express.json());
app.use(cors());
```

Database Connection With MongoDB

```
mongoose.connect("mongodb+srv://weareayush:weareayush123@cluster0.gn1tvf1.
mongodb.net/e-commerce");

//Image Storage Engine
const storage = multer.diskStorage({
    destination: './upload/images',
    filename: (req, file, cb) => {
      console.log(file);
                                                  return                    cb(null,
`${file.fieldname}_${Date.now()}${path.extname(file.originalname)}`)
    }
})
const upload = multer({storage: storage})
app.post("/upload", upload.single('product'), (req, res) => {
    res.json({
        success: 1,
        image_url: `http://localhost:4000/images/${req.file.filename}`
    })
})
app.use('/images', express.static('upload/images'));
```

MiddleWare to fetch user from database

```
const fetchuser = async (req, res, next) => {
  const token = req.header("auth-token");
  if (!token) {
    res.status(401).send({ errors: "Please authenticate using a valid token" });
  }
  try {
    const data = jwt.verify(token, "secret_ecom");
    req.user = data.user;
    next();
  } catch (error) {
    res.status(401).send({ errors: "Please authenticate using a valid token" });
  }
};
```

Schema for creating user model

```
const Users = mongoose.model("Users", {
  name: {
    type: String,
  },
  email: {
    type: String,
    unique: true,
  },
  password: {
    type: String,
  },
  cartData: {
    type: Object,
  },
  date: {
    type: Date,
    default: Date.now,
  },
});
```

Schema for creating Product

```
const Product = mongoose.model("Product", {
  id: {
    type: Number,
    required: true,
  },
  name: {
    type: String,
    required: true,
  },
  image: {
    type: String,
    required: true,
  },
  category: {
    type: String,
    required: true,
  },
  new_price: {
    type: Number
  },
  old_price: {
    type: Number
  },
  date: {
    type: Date,
    default: Date.now,
  },
  avilable: {
    type: Boolean,
    default: true,
  },
});

app.get("/", (req, res) => {
  res.send("Root");
});
```

Create an endpoint at ip/login for login the user and giving auth-token

```
app.post('/login', async (req, res) => {
  console.log("Login");
    let success = false;
    let user = await Users.findOne({ email: req.body.email });
    if (user) {
        const passCompare = req.body.password === user.password;
        if (passCompare) {
            const data = {
                user: {
                    id: user.id
                }
            }
      success = true;
      console.log(user.id);
      const token = jwt.sign(data, 'secret_ecom');
      res.json({ success, token });
        }
        else {
             return res.status(400).json({success: success, errors: "please try with correct
email/password"})
        }
    }
    else {
        return res.status(400).json({success: success, errors: "please try with correct email/password"})
    }
})
```

Create an endpoint at ip/auth for regestring the user in data base & sending token

```
app.post('/signup', async (req, res) => {
  console.log("Sign Up");
        let success = false;
        let check = await Users.findOne({ email: req.body.email });
        if (check) {
            return res.status(400).json({ success: success, errors: "existing user found with this email" });
        }
        let cart = {};
          for (let i = 0; i < 300; i++) {
          cart[i] = 0;
        }
        const user = new Users({
            name: req.body.username,
            email: req.body.email,
            password: req.body.password,
            cartData: cart,
        });
        await user.save();
        const data = {
            user: {
                id: user.id
            }
        }

        const token = jwt.sign(data, 'secret_ecom');
        success = true;
        res.json({ success, token })
```

```
    })

app.get("/allproducts", async (req, res) => {
  let products = await Product.find({});
  console.log("All Products");
    res.send(products);
});

app.get("/newcollections", async (req, res) => {
  let products = await Product.find({});
  let arr = products.slice(1).slice(-8);
  console.log("New Collections");
  res.send(arr);
});

app.get("/popularinwomen", async (req, res) => {
  let products = await Product.find({});
  let arr = products.splice(0,  4);
  console.log("Popular In Women");
  res.send(arr);
});
```

Create an endpoint for saving the product in cart

```
app.post('/addtocart', fetchuser, async (req, res) => {
  console.log("Add Cart");
    let userData = await Users.findOne({_id:req.user.id});
    userData.cartData[req.body.itemId] += 1;
    await Users.findOneAndUpdate({_id:req.user.id}, {cartData:userData.cartData});
    res.send("Added")
  })
```

Create an endpoint for saving the product in cart

```
app.post('/removefromcart', fetchuser, async (req, res) => {
  console.log("Remove Cart");
  let userData = await Users.findOne({_id:req.user.id});
  if(userData.cartData[req.body.itemId]!=0)
  {
    userData.cartData[req.body.itemId] -= 1;
  }
  await Users.findOneAndUpdate({_id:req.user.id}, {cartData:userData.cartData});
  res.send("Removed");
 })
```

Create an endpoint for saving the product in cart

```
app.post('/getcart', fetchuser, async (req, res) => {
 console.log("Get Cart");
 let userData = await Users.findOne({_id:req.user.id});
 res.json(userData.cartData);

 })

app.post("/addproduct", async (req, res) => {
 let products = await Product.find({});
 let id;
 if (products.length>0) {
   let last_product_array = products.slice(-1);
   let last_product = last_product_array[0];
   id = last_product.id+1;
 }
 else
 { id = 1; }
 const product = new Product({
   id: id,
   name: req.body.name,
   image: req.body.image,
   category: req.body.category,
   new_price: req.body.new_price,
   old_price: req.body.old_price,
  });
  console.log(product);
  await product.save();
  console.log("Saved");
  res.json({success:true,name:req.body.name})
});

app.post("/removeproduct", async (req, res) => {
  const product = await Product.findOneAndDelete({ id: req.body.id });
  console.log("Removed");
  res.json({success:true,name:req.body.name})
});

app.listen(port, (error) => {
  if (!error) console.log("Server Running on port " + port);
  else console.log("Error : ", error);
});
```

CHAPTER 5

RESULT ANALYSIS AND DISCUSSION

An eCommerce website is designed with specific purposes and goals in mind, which shape its overall strategy and execution. The primary purpose could range from selling physical products, digital goods, or services, to a combination of these offerings. Understanding this purpose helps define the website's goals, which may include increasing sales, expanding market reach, improving customer experience, or building brand recognition. These goals drive the site's design, functionality, and marketing efforts.

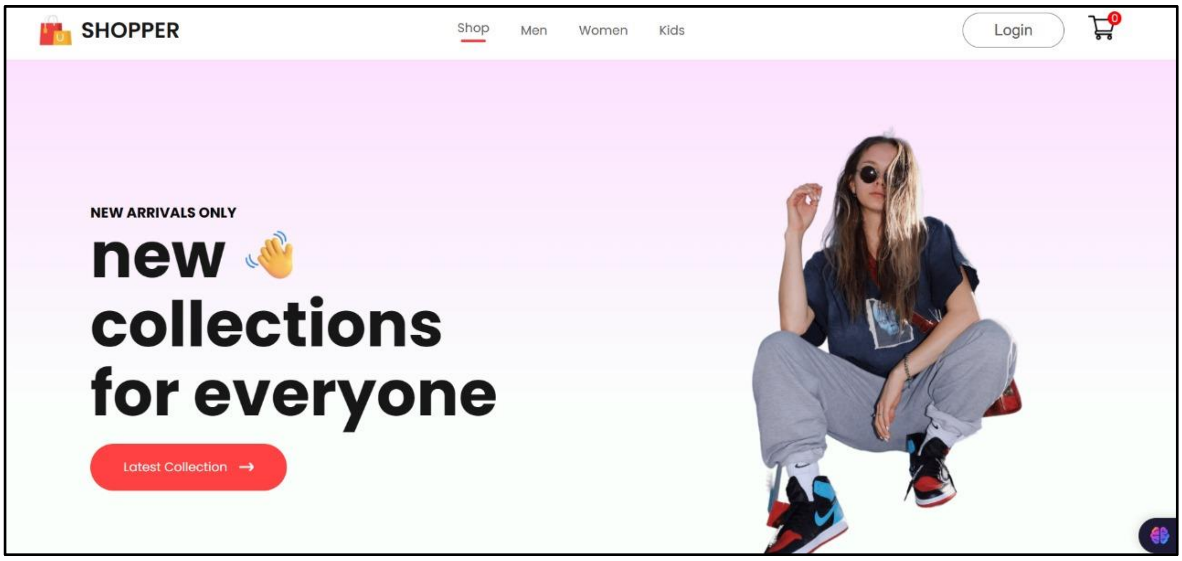
5.1. Results Screenshots

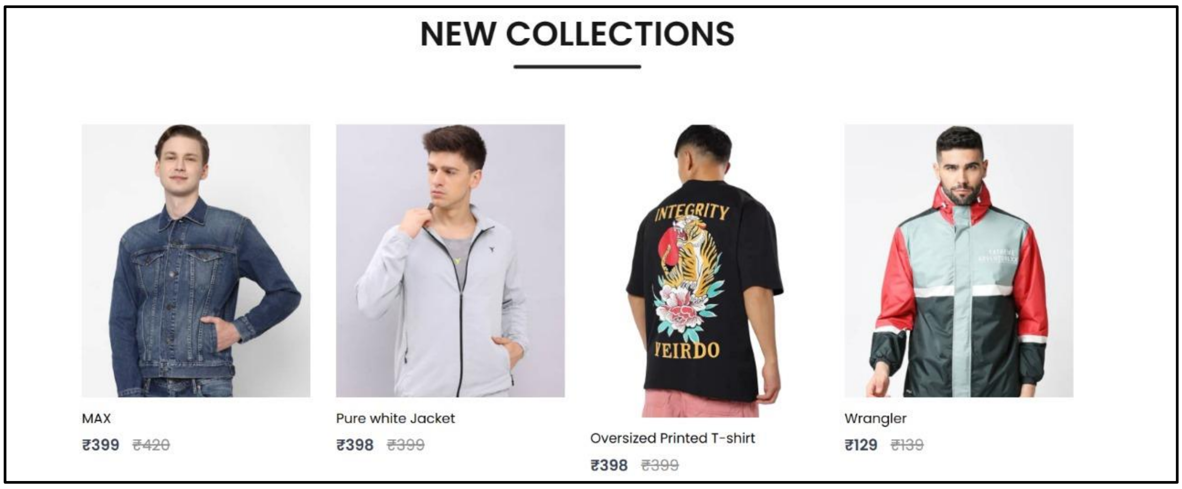
The below page shows the General Zone of the project. Using this page the user and buy clothes that he/she wants.

5.1.1. General Zone

The below given diagram shows the home page of the website. It comprises of various tabs such as mens, womens, kids, login and the cart tab.
In addition it also comprise of new collections portion which are newly added to the website.

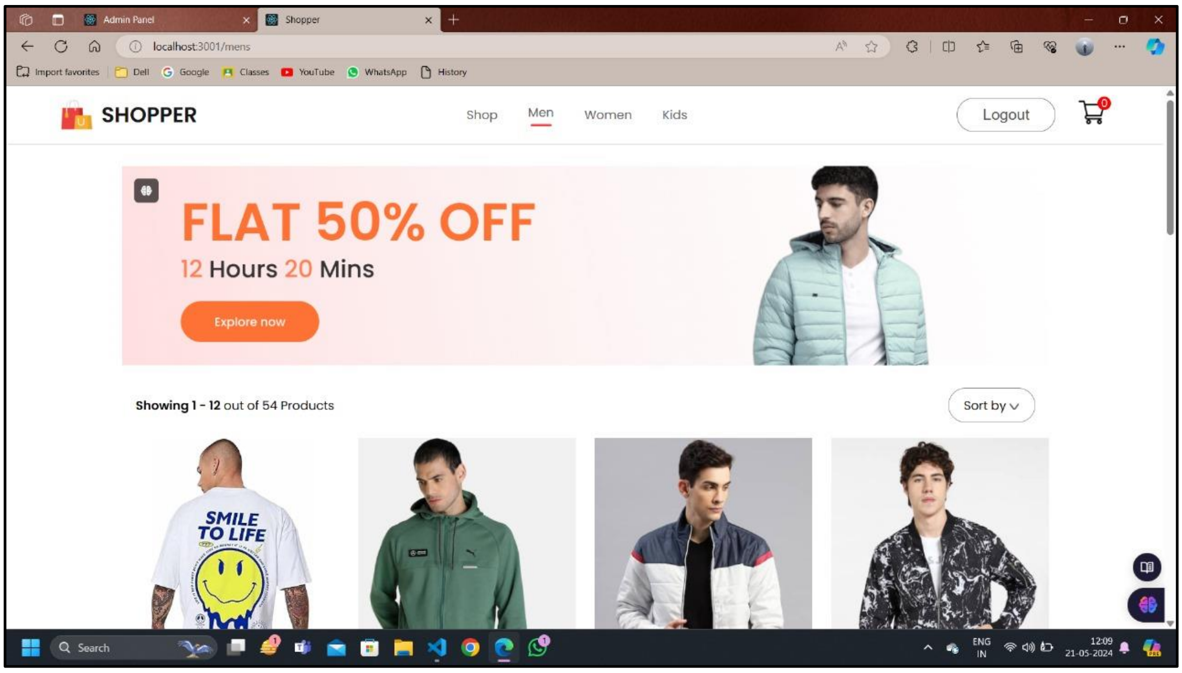
Home Page

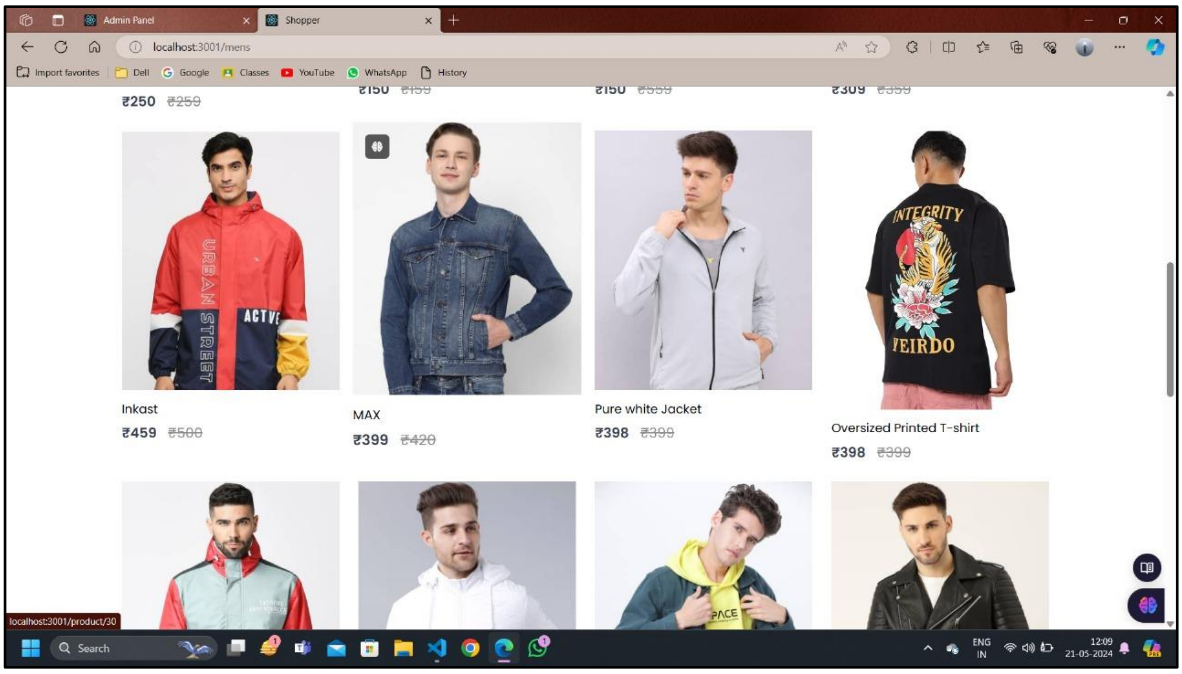




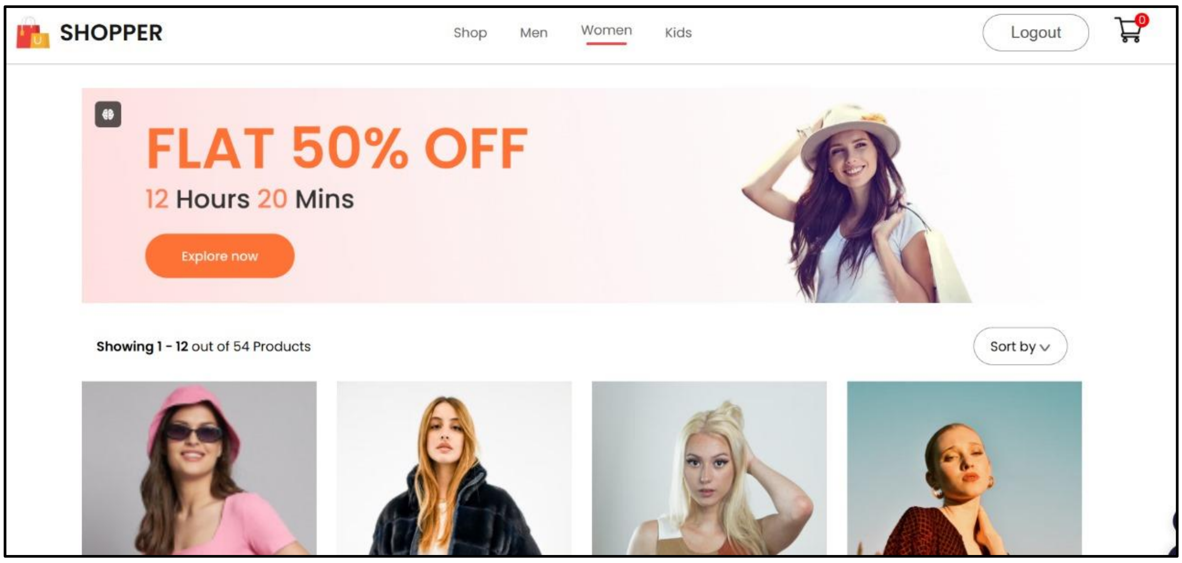
The website comprises of various sections such as mens, womens and the kids.

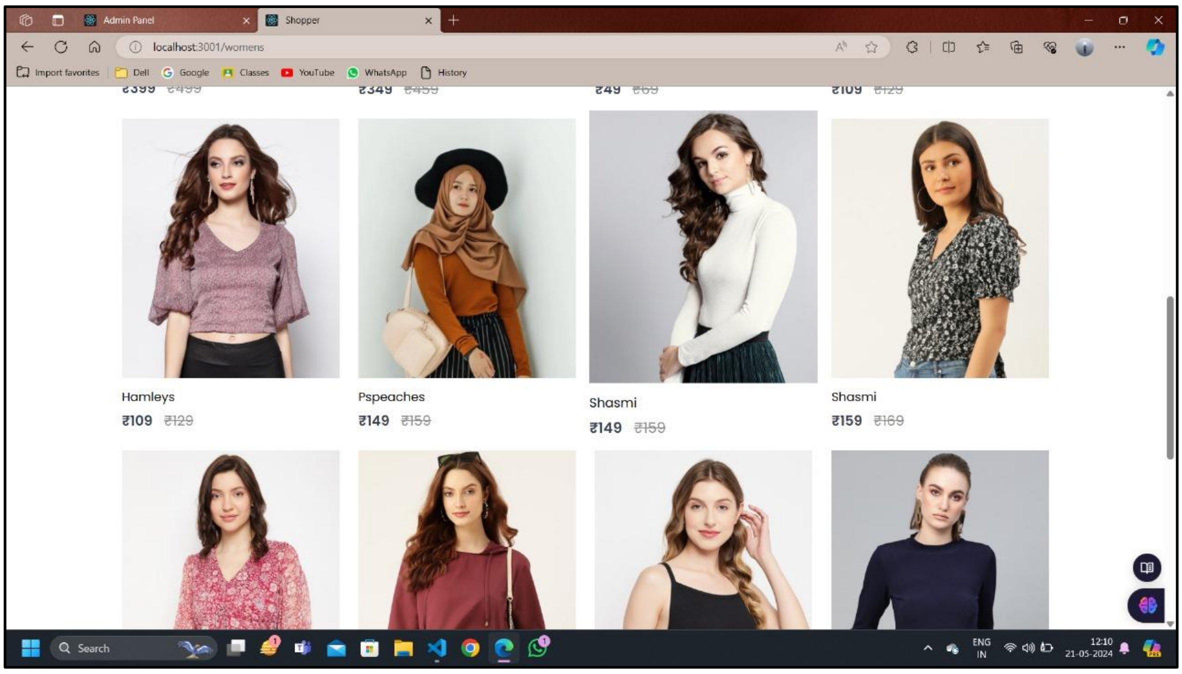
Men’s section

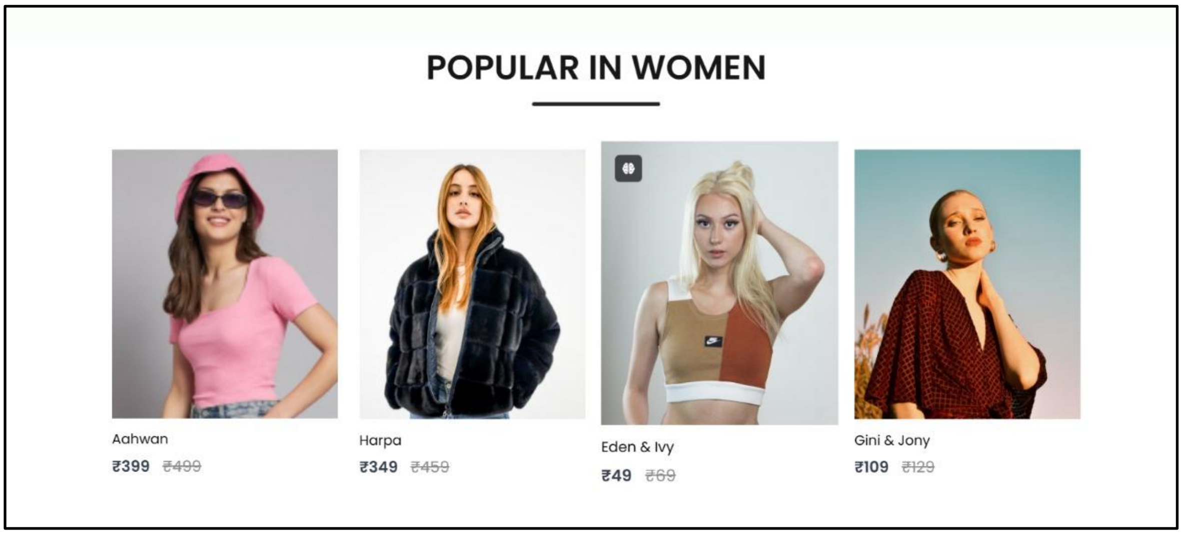




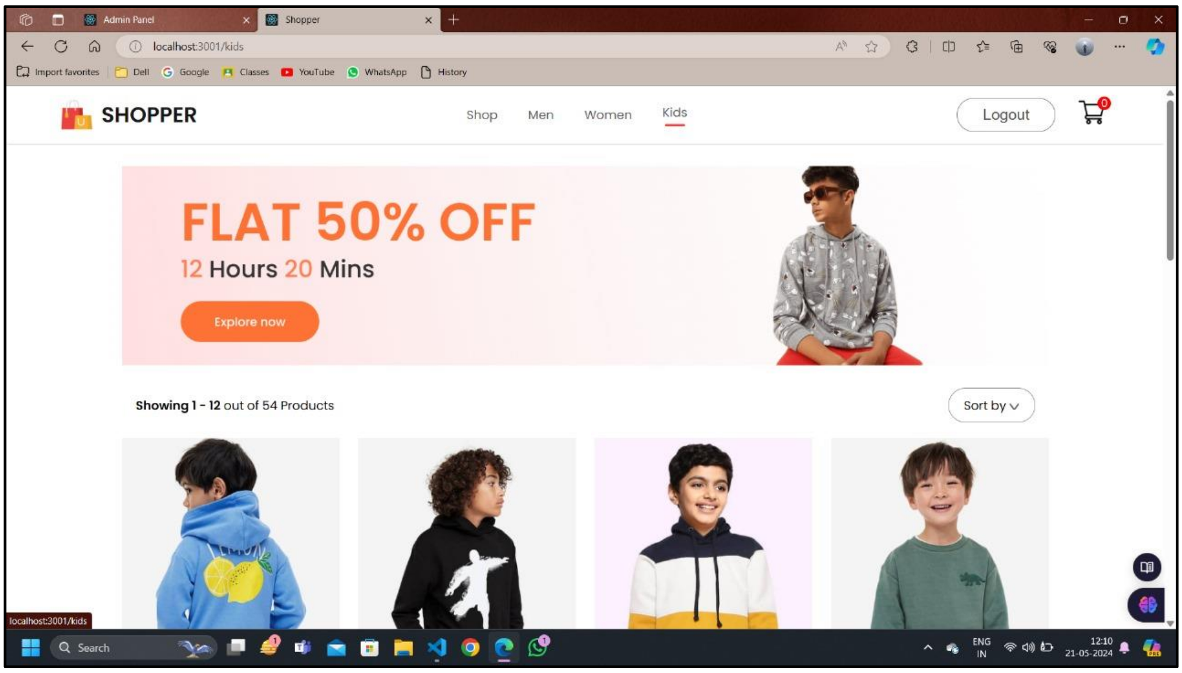
Women’s section

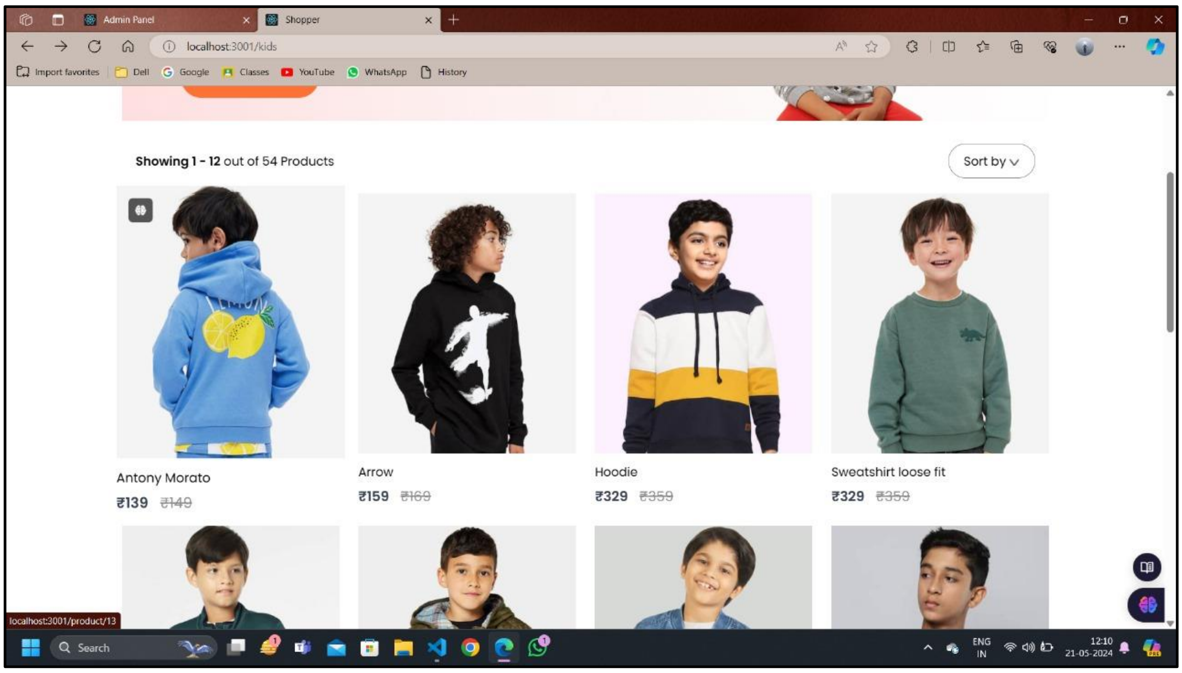



Admin Panel
Shopper
localhost:3001/womens
Import favorites
Dell
Google
Classes
YouTube
WhatsApp
History
Hamleys
₹109 ~~₹129~~
Pspeaches
₹149 ~~₹159~~
Shasmi
₹149 ~~₹159~~
Shasmi
₹159 ~~₹169~~
Search
ENG IN
12:10
21-05-2024

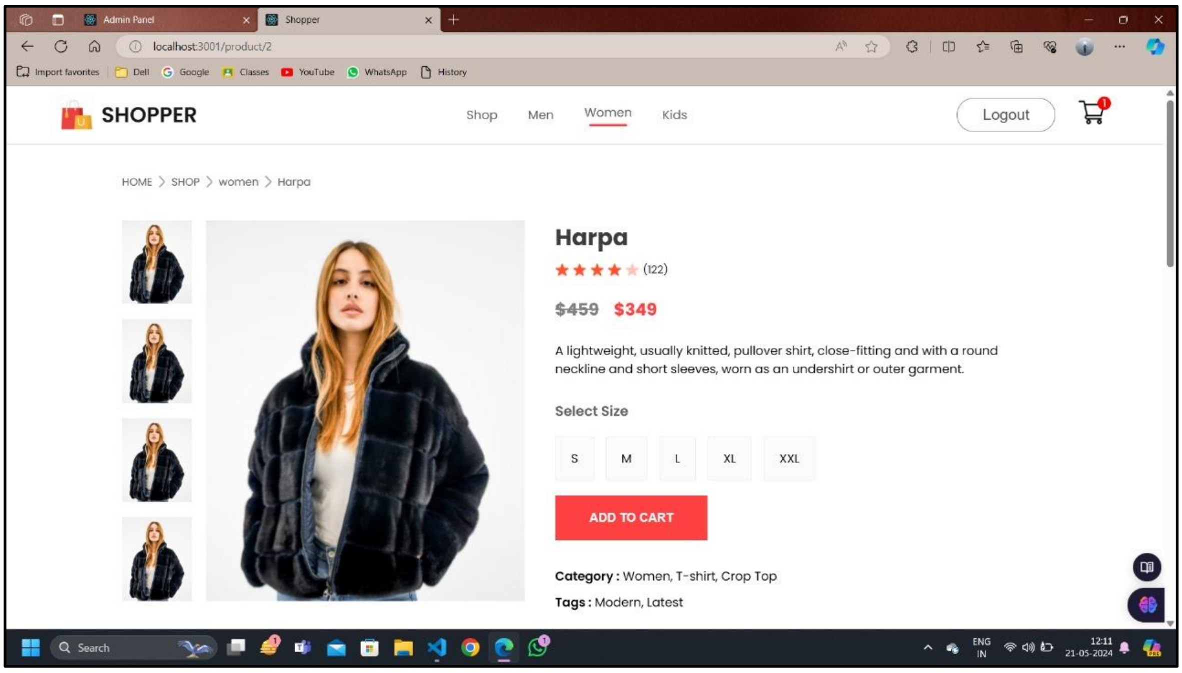

POPULAR IN WOMEN
Aahwan
₹399 ~~₹499~~
Harpa
₹349 ~~₹459~~
Eden & Ivy
₹49 ~~₹69~~
Gini & Jony
₹109 ~~₹129~~

Kids section





Product details



Harpa

★★★★☆ (122)

~~$459~~ $349

A lightweight, usually knitted, pullover shirt, close-fitting and with a round neckline and short sleeves, worn as an undershirt or outer garment.

Select Size

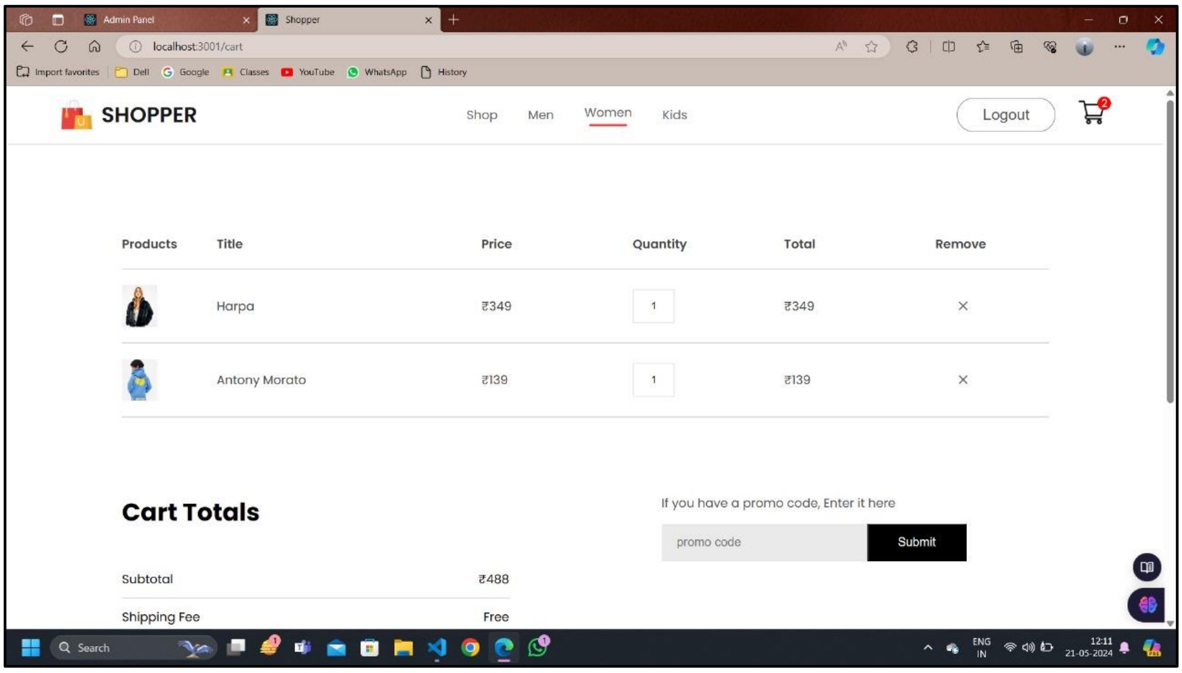
S M L XL XXL

ADD TO CART

Category : Women, T-shirt, Crop Top

Tags : Modern, Latest

Cart section



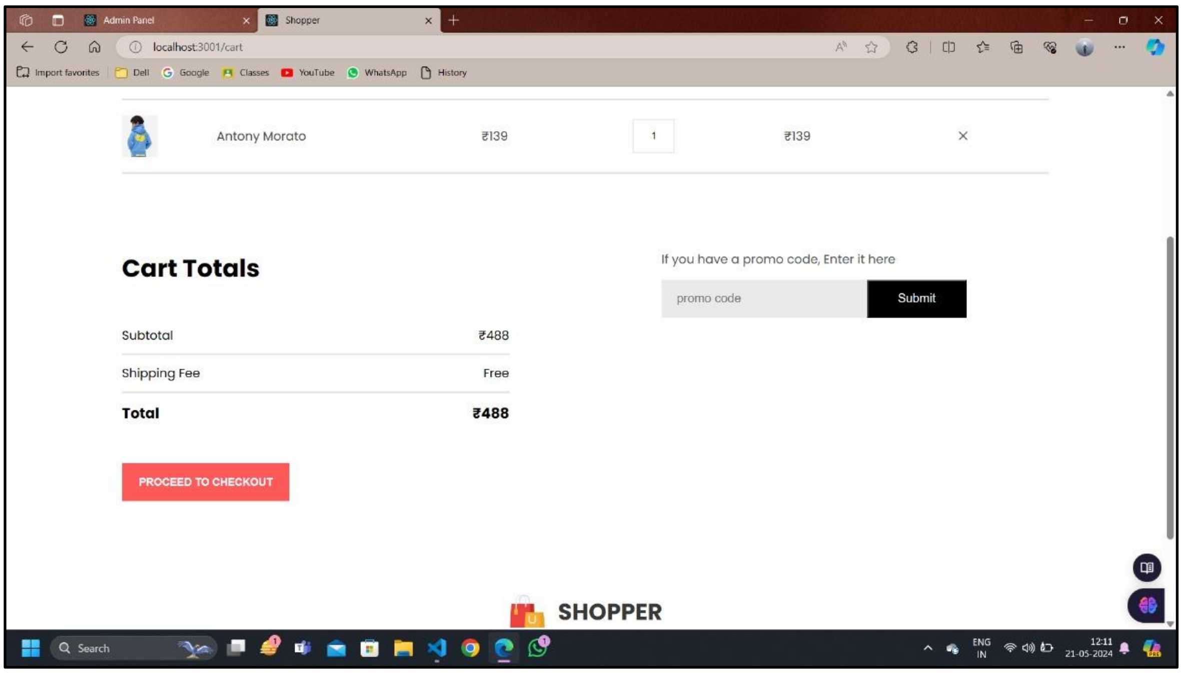
Products	Title	Price	Quantity	Total	Remove
	Harpa	₹349	1	₹349	×
	Antony Morato	₹139	1	₹139	×

Cart Totals

Subtotal	₹488
Shipping Fee	Free

If you have a promo code, Enter it here

promo code | Submit



Antony Morato	₹139	1	₹139	×

Cart Totals

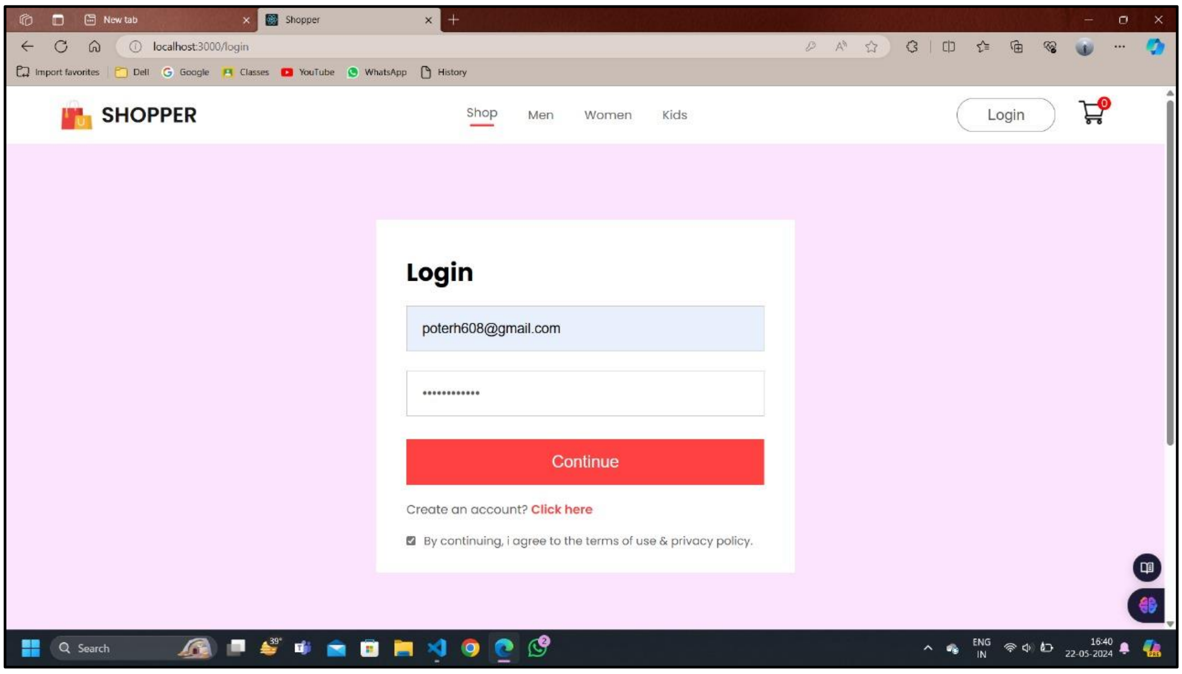
Subtotal	₹488
Shipping Fee	Free
Total	**₹488**

PROCEED TO CHECKOUT

If you have a promo code, Enter it here

promo code | Submit

Login page



Login

poterh608@gmail.com

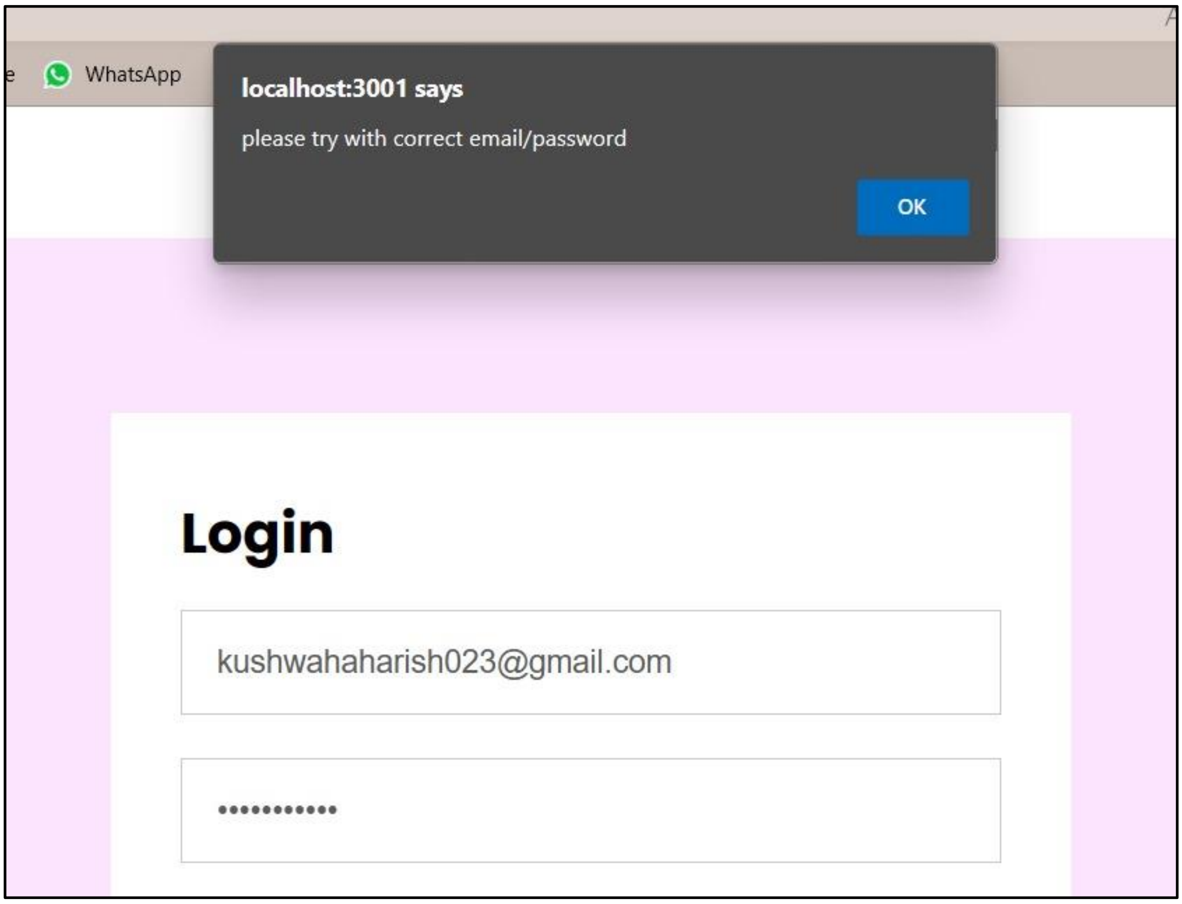
••••••••••••

Continue

Create an account? Click here

By continuing, i agree to the terms of use & privacy policy.

If error occurs while login


WhatsApp

localhost:3001 says

please try with correct email/password

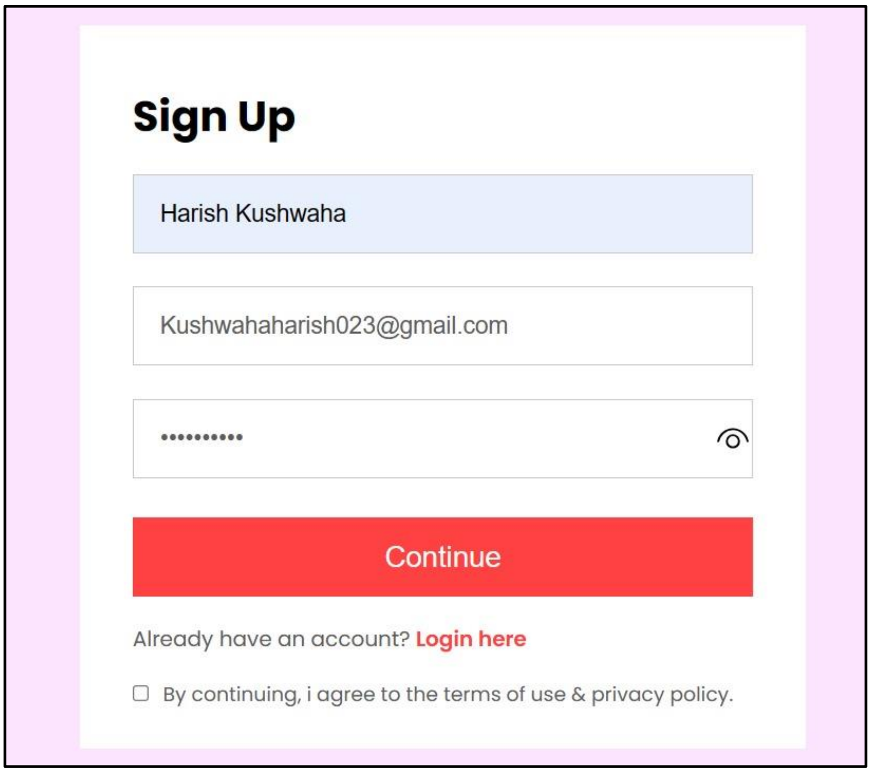
OK

Login

kushwahaharish023@gmail.com

•••••••••••

Signup page


Sign Up

Harish Kushwaha

Kushwahaharish023@gmail.com

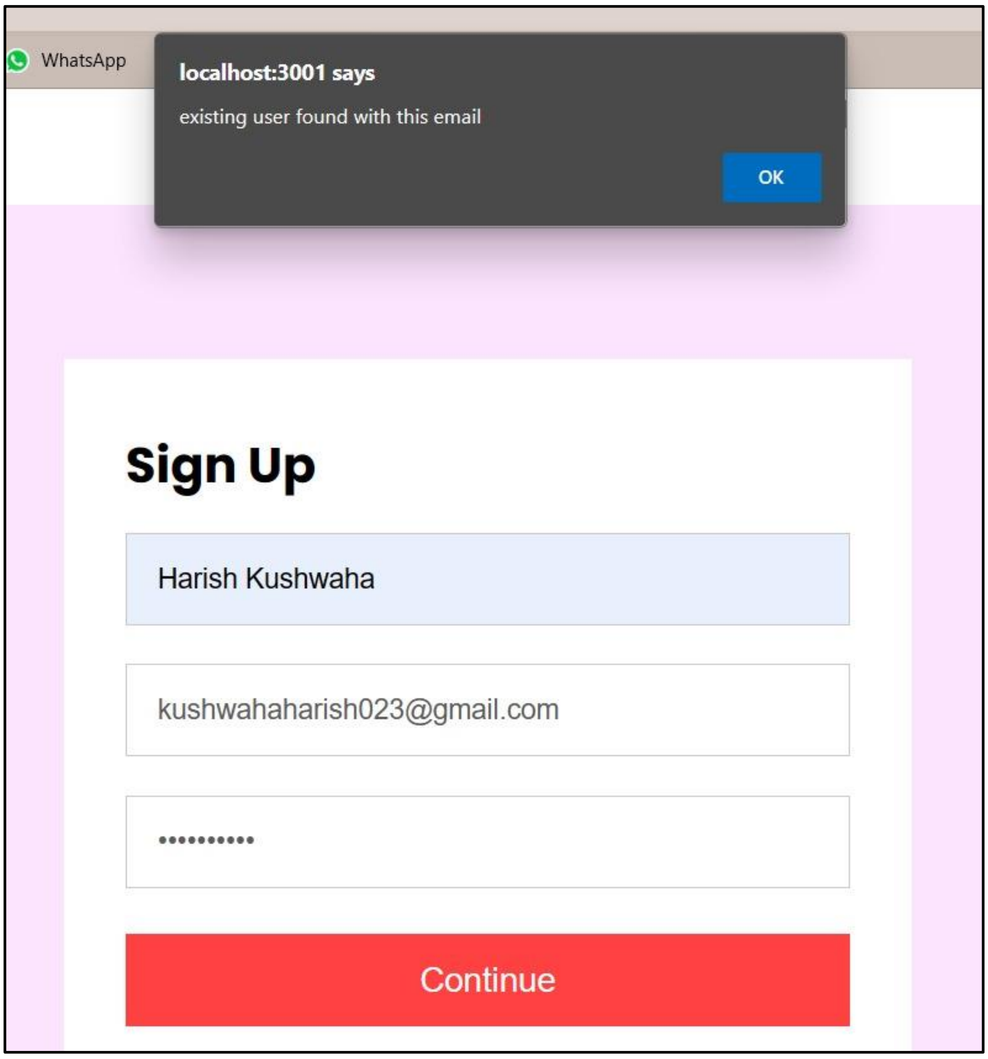
••••••••••

Continue

Already have an account? Login here

☐ By continuing, i agree to the terms of use & privacy policy.

If the email id is already registered



WhatsApp

localhost:3001 says

existing user found with this email

OK

Sign Up

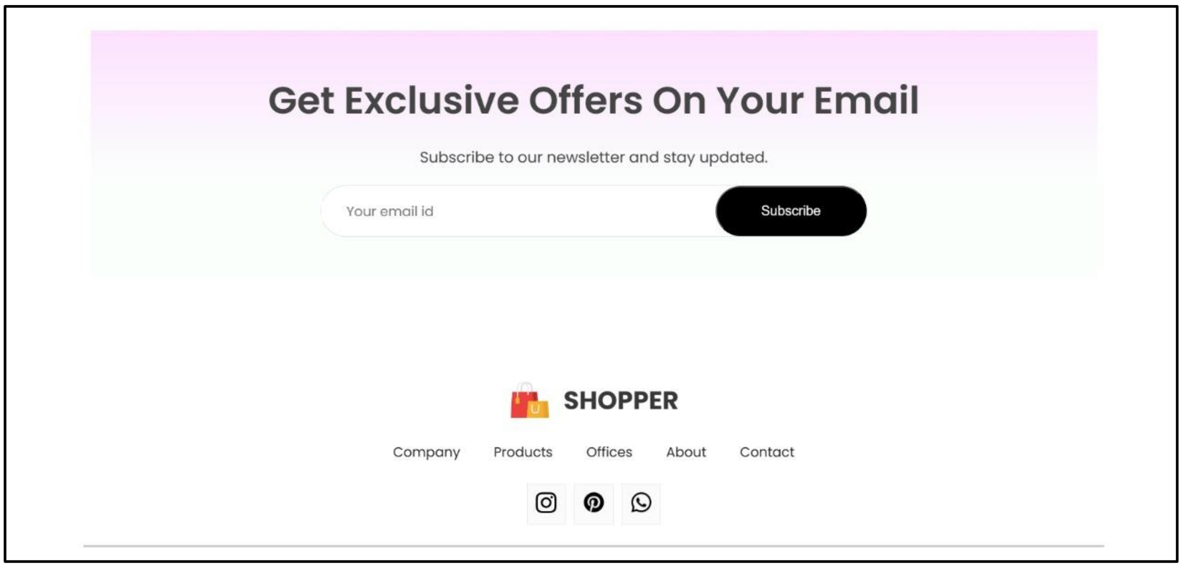
Harish Kushwaha

kushwahaharish023@gmail.com

••••••••••

Continue

Footer of General Zone



Get Exclusive Offers On Your Email

Subscribe to our newsletter and stay updated.

Your email id

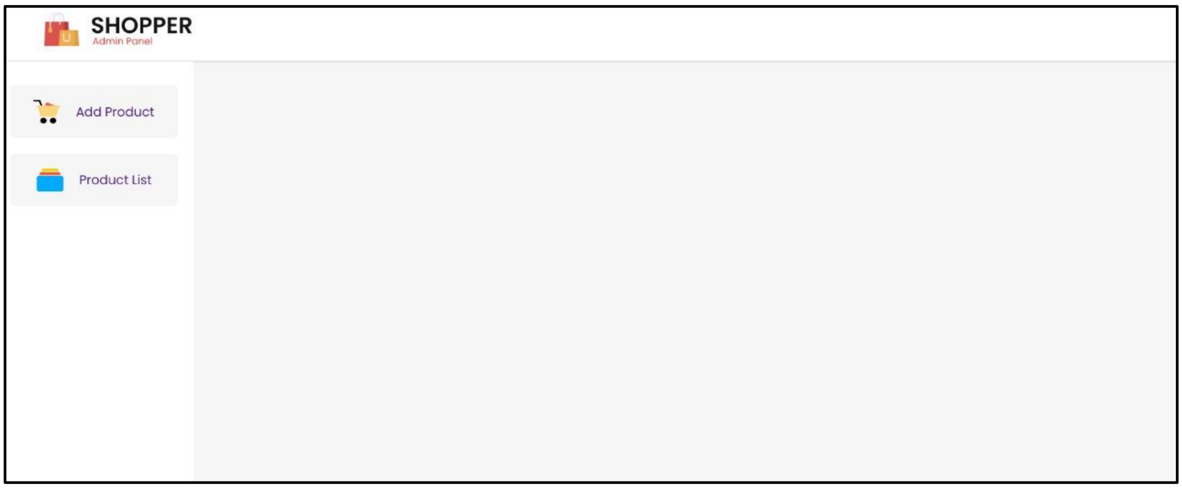
Subscribe

SHOPPER

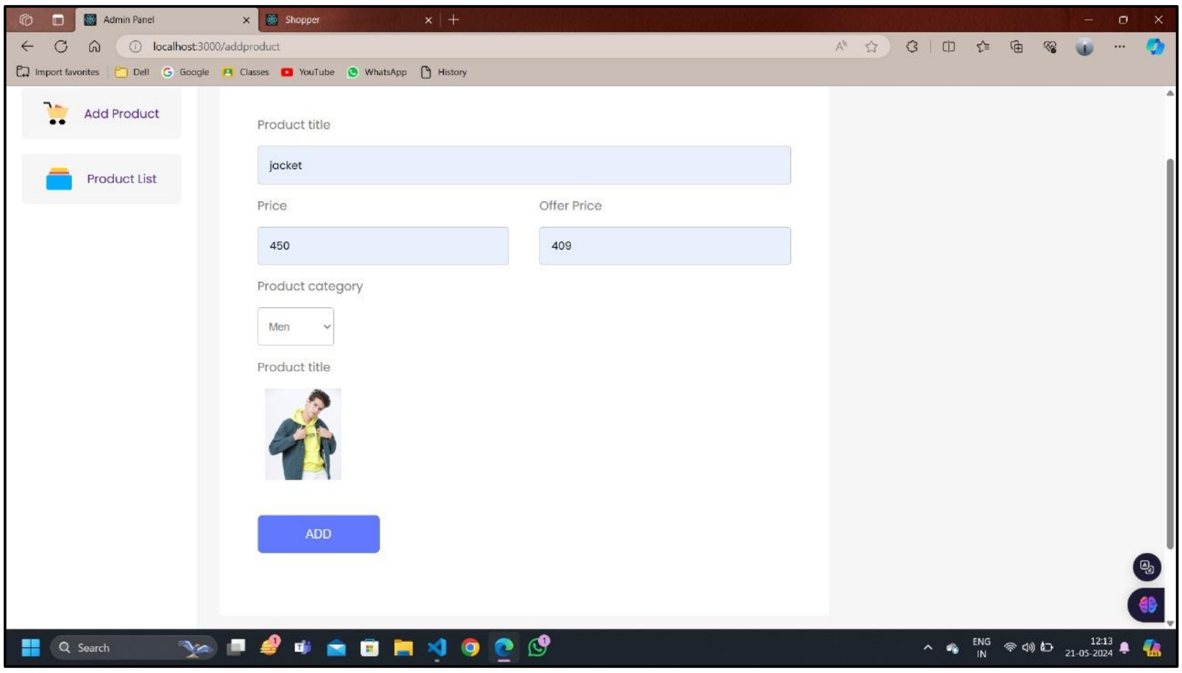
Company Products Offices About Contact

5.1.2. Admin Section

Section to add product and view product



Add product section



Product title

jacket

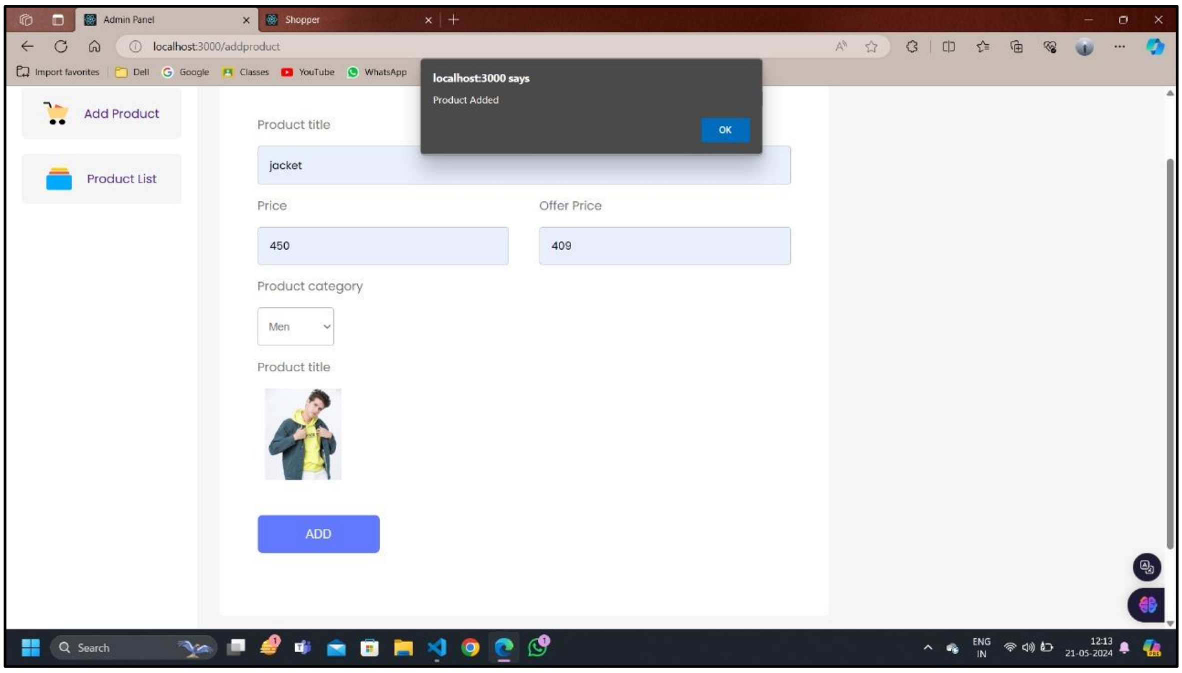
Price: 450

Offer Price: 409

Product category: Men

Product title

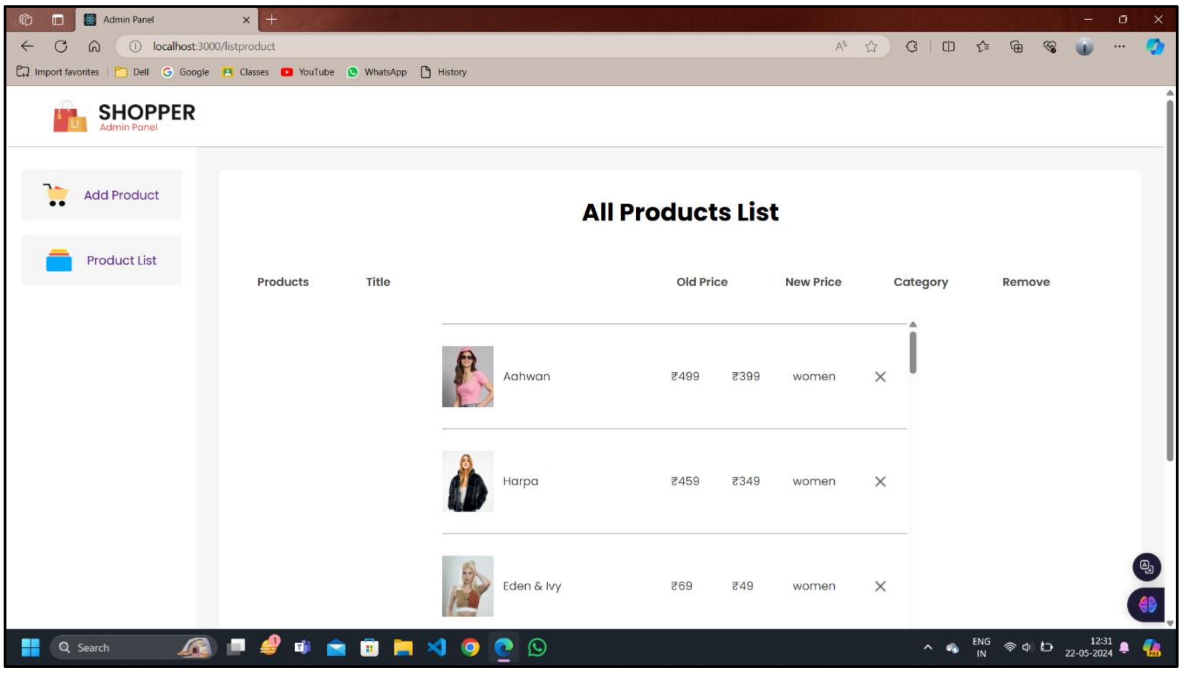
Notification on successful addition of product



localhost:3000 says

Product Added

Product List section



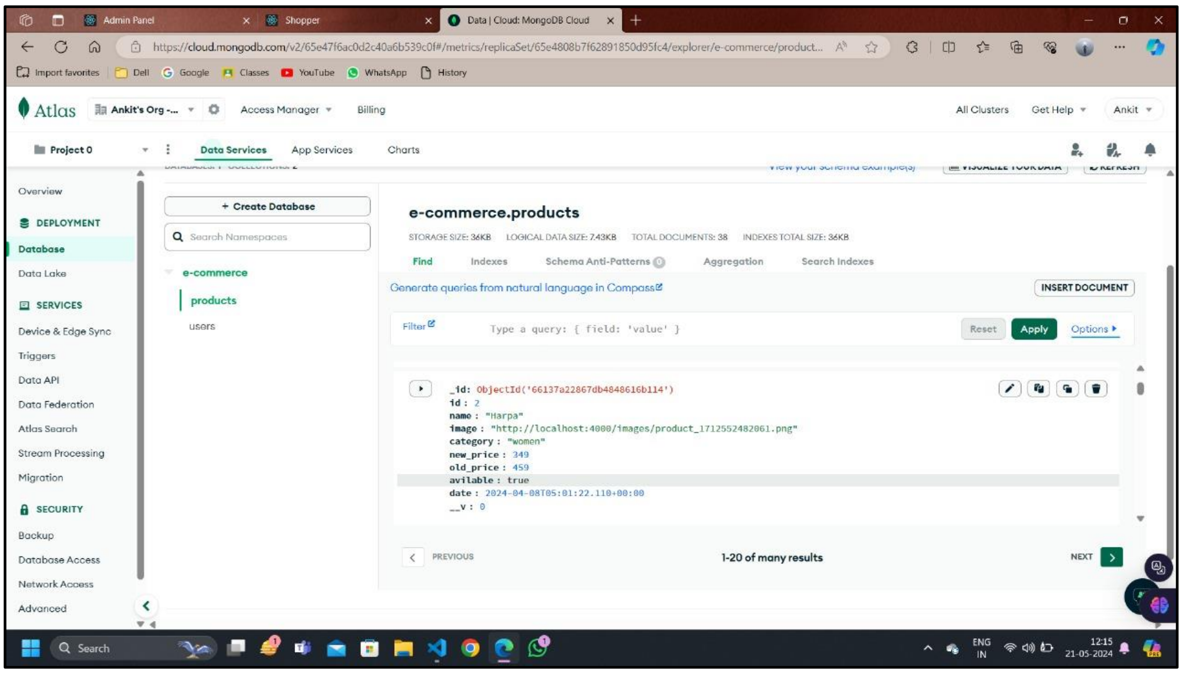
All Products List

Products	Title	Old Price	New Price	Category	Remove
	Aahwan	₹499	₹399	women	×
	Harpa	₹459	₹349	women	×
	Eden & Ivy	₹69	₹49	women	×

5.1.3. Database storage

Use of MongoDB atlas

Data of products



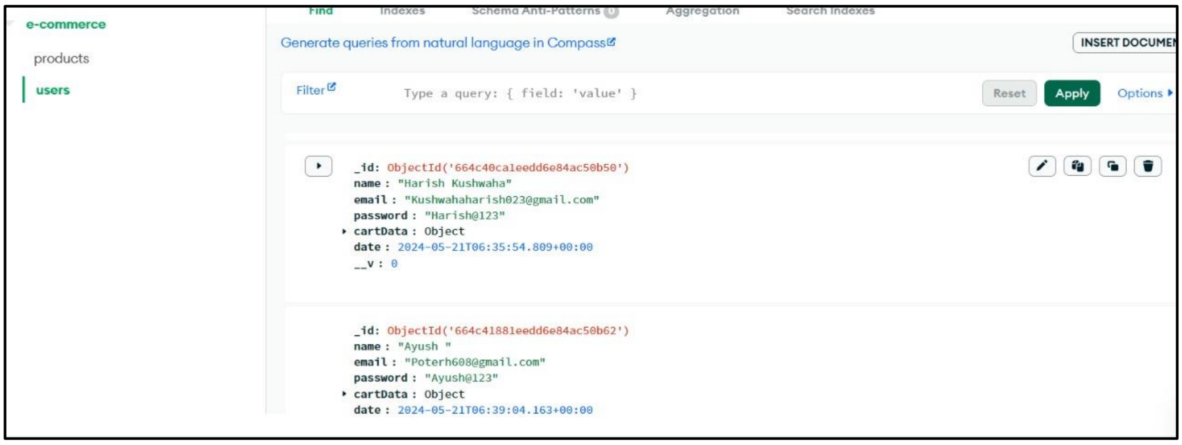
e-commerce.products

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 7.43KB TOTAL DOCUMENTS: 38 INDEXES TOTAL SIZE: 36KB

```
_id: ObjectId('66137a22867db4848616b114')
id : 2
name : "Harpa"
image : "http://localhost:4000/images/product_1712552482061.png"
category : "women"
new_price : 349
old_price : 459
avilable : true
date : 2024-04-08T05:01:22.110+00:00
__v : 0
```

1-20 of many results

Data of users



```
_id: ObjectId('664c40ca1eedd6e84ac50b50')
name : "Harish Kushwaha"
email : "Kushwahaharish023@gmail.com"
password : "Harish@123"
cartData : Object
date : 2024-05-21T06:35:54.809+00:00
__v : 0

_id: ObjectId('664c41881eedd6e84ac50b62')
name : "Ayush "
email : "Poterh608@gmail.com"
password : "Ayush@123"
cartData : Object
date : 2024-05-21T06:39:04.163+00:00
```

CHAPTER 6

CONCLUSION

Creating an e-commerce website can be a highly rewarding venture, providing businesses with an opportunity to reach a broader audience and increase sales. However, it requires careful planning and execution to overcome inherent challenges and limitations. Here are the key points to consider for a successful e-commerce website:

1. **User Experience:** Prioritize a user-friendly design with intuitive navigation, fast loading times, and mobile responsiveness to ensure a seamless shopping experience.

2. **Product Presentation:** Invest in high-quality images, detailed descriptions, and accurate product information to bridge the gap created by the lack of physical interaction.

3. **Security:** Implement robust security measures to protect customer data and build trust. This includes SSL certificates, secure payment gateways, and regular security audits.

4. **Customer Service:** Offer efficient and accessible customer support through various channels like live chat, email, and phone to assist customers promptly.

5. **Returns and Refunds:** Simplify the return and refund process to enhance customer satisfaction and loyalty. Clearly communicate return policies and streamline the procedure.

6. **Logistics and Delivery:** Partner with reliable logistics providers to ensure timely and cost-effective delivery. Consider offering multiple shipping options to meet diverse customer needs.

7. **Marketing and SEO:** Develop a comprehensive marketing strategy that includes SEO, social media marketing, email campaigns, and paid advertising to drive traffic and conversions.

8**. Continuous Improvement:** Regularly update and improve the website based on user feedback, analytics, and market trends to stay competitive and relevant.

9. **Sustainability:** Address environmental concerns by adopting eco-friendly packaging and exploring sustainable delivery options to appeal to environmentally conscious consumers.

In conclusion, while an e-commerce website opens up new business opportunities, its success depends on addressing its limitations and enhancing the overall customer experience. By focusing on these critical areas, businesses can build a robust online presence that attracts and retains customers, ultimately driving growth and profitability.

CHAPTER 7

LIMITATIONS

Online shopping websites, despite their many advantages, come with several limitations that can affect the overall shopping experience. Here are some of the key limitations:

1. Lack of Physical Interaction:
    - Inability to Try Before Buying: Customers cannot physically inspect or try on products before purchasing. This is particularly problematic for items like clothing, shoes, and furniture, where fit and feel are important.
    - Touch and Feel: The tactile experience of products is missing, which can be crucial for items like fabrics, gadgets, and other tangible goods.
2. Delivery Issues:
    - Shipping Delays: Unexpected delays in shipping can occur due to various reasons such as logistical problems, weather conditions, or high demand periods.
    - Shipping Costs: Sometimes, the cost of shipping can be high, especially for international deliveries or expedited shipping options.
3. Returns and Refunds:
    - Complicated Return Processes: Returning items can be a hassle with strict return policies, lengthy procedures, and potential shipping costs.
    - Refund Delays: Getting a refund can take time, and customers may need to wait weeks to get their money back.
4. Security Concerns:
    - Data Privacy: Sharing personal and payment information online carries the risk of data breaches and identity theft.
    - Fraud: There is a risk of encountering fraudulent websites or sellers that scam customers by not delivering products or delivering counterfeit goods.

5. Limited Customer Service:
    - Lack of Immediate Assistance: Unlike physical stores where assistance is readily available, online customer service might be slow, and resolving issues can take time.
    - Automated Responses: Many online platforms rely on automated responses, which can be frustrating when dealing with specific or complex issues.
6. Environmental Concerns:
    - Packaging Waste: Online shopping often involves excessive packaging, contributing to environmental waste.
    - Carbon Footprint: The logistics involved in shipping products contribute to the carbon footprint, particularly with long-distance deliveries.

CHAPTER 8
FUTURE SCOPE OF THE PROJECT

The future of ecommerce website holds immense potential with continued technological advancements. AI-driven personalization, augmented reality shopping experiences, and seamless mobile integration will play pivotal roles. In conclusion, e-commerce will remain a dominant force in the global economy, offering convenience and limitless opportunities for business and consumers alike.

The future of e-commerce is expected to continue evolving with several key trends and opportunities.

- Mobile commerce
- AI and Personalization
- Voice Commerce
- Augmented Reality (AR) and Virtual Reality (VR)
- Blockchain for Supply Chain and Payment
- Global expansion
- Data Security
- Health and Wellness

The future of e-commerce will be shaped by technology, consumer preferences and market dynamics.

CHAPTER 9

REFERENCES

[1] Chanana, N., & Goele, S. (2012). Future of e-commerce in India. International Journal of Computing & Business Research, 8.

[2] Mai, N. (2020). E-commerce Application using MERN stack.

[3] Ullah, S. E., Alauddin, T., & Zaman, H. U. (2016, January). eveloping an E-commerce website. In 2016 International Conference on Microelectronics, Computing and Communications (Microcosm) (pp.1-4). IEEE.

[4] King, DN., & King, D.N. (2004). Introduction to e-commerce. Prentice Hall.

[5] Nemat, R. (2011). Taking a look at different types of e-commerce. World Applied Programming, 1(2), 100-104.

[6] Niranjanamurthy, M., Kavyashree, N., Jagannath, S., & Chahar, D. (2013). Analysis of e-commerce and m-commerce: advantages, limitations and security issues. International Journal of Advanced Research in Computer and Communication Engineering, 2(6), 2360-2370.

[7] Thi Thu Hien Tran. (2022). THE DEVELOPMENT OF AN ECOMMERCE WEB APPLICATIONUSING MERN STACK.

[8] Monika Mehra, Manish Kumar, Anjali Maurya, Charu Sharma, Shanu. (2021). MERN Stack WebDevelopment. Annals of the Romanian Society for Cell Biology, 25(6), 11756–11761

[9] Nagothu Diwakar Naidu., Pentapati Adarsh., Sabharinadh Reddy., Gumpula Raju., Uppu Sai Kiran & Vikash Sharma. E-Commerce web Application by using MERN Technology. International Journal for Modern Trends in Science and Technology 7, 1–5 (2021).

[10] Subramanian, V. (2019). MongoDB. In: Pro MERN Stack. Apress, Berkeley, CA.44.

[11] Naidu, N. D., Adarsh, P., Reddy, S., Raju, G., Kiran, U. S., Sharma, V., ... & Sharma, V. (2021).E-Commerce web Application by using MERN Technology. International Journal for Modern Trends inScience and Technology, 7 , 1-5.

[12] Hirenkumar Pravinbhai Vacchani. (2018). A Critical Study of E-Commerce Market of India. Vidhyayana- An International n Multidisciplinary Peer-Reviewed E-Journal - ISSN 2454-8596 , 4(2).

[13] Hoque, S. (2020). Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node. js. Packet Publishing Ltd.

[14] Rossi, G., Schwabe, D., & Lyardet, F. (1999, November). Web application models are more than conceptual models.

[15] Brown , Jeff ", E-commerce strategies and practices "Editor Jill McKenna. Ciwf (2001)

[16] Khurana, Ajeet (25 November 2019). "Did You Know That There Are 4 Types of Ecommerce?". The Balance Small Business. Dotdash. Archived from the original on 22 January 2021. Retrieved 4 May 2021

[17] Millward, Steven (18 August 2016). "Asia's e-commerce spending to hit a record $1 trillion this year – but most of that is China". Tech in Asia. Archived from the original on 19 August 2016. Retrieved 4 May 2021.

[18] Bakos, Yannis (2001). "The Emerging Landscape for Retail E-Commerce". Journal of Economic Perspectives. 15 (1): 69–80. CiteSeerX 10.1.1.4.9128. doi:10.1257/jep.15.1.69.

[19] J. Kumar and V. Garg, "Security analysis of unstructured data in NOSQL MongoDB database," 2017 International Conference on
Computing and Communication Technologies for Smart Nation (IC3TSN), Gurgaon, 2017, pp. 300-305.

[20] IEEE.Dyl, T. and Przeorski, K., 2017. Mastering Full-Stack React Web Development. Packt Publishing. [12]Ambler, T. and Cloud, N.,
2015. Javascript Frameworks For Modern Web Dev. Apress

[21] Velliangiri, S., & Karunya, P. K. (2020, January). Blockchain Technology: Challenges and Security issues in Consensus algorithm. In 2020
International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-8).

[22] MongoDB official website for understanding what is MERN stack: https://www.mongodb.com/mern-stack [Access Date: 10-02-23]

[23] Official Documentation for ReactJS: https://react.dev/learn [Access Date: 13-02-23]

[24] NodeJS official documentation: https://nodejs.org/en/docs [Access Date: 15-02-23]

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

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