SnapBuy – Ecommerce Web App

Mid term progress report

Submitted to:

Dr. Hemant Verma

Assistant Professor

CSE Department

Submitted by:

Gettanjali

230000613022

4th Semester

CONTENTS

CERTIFICATE

This is to certify that the project entitled “SnapBuy– Ecommerce Web App” submitted by “gettanjali” to Chaudhary Bansilal University Bhiwani, Haryana in partial fulfilment of the requirement for the award of year 2nd of the degree of Msc in Computer Science Engineering. This project work carried out by me. The project fulfils the requirement as per the regulation of the institute of university and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university to the best of my knowledge and belief.

Date: Mentor Signature:

DECLARATION

This is to certify that the work being presented in the minor project entitled “SnapBuy Ecommerce Web-App” submitted by undersigned student of Msc in Computer Science to Chaudhary Bansilal University (Bhiwani), Haryana in the fulfilment for award of semester 4th is a record of my own work carried out by me under guidance and supervision of concerned faculty and staff of the college. I certify the content of this work has not submitted elsewhere for award of any other degree.

Date:

Signature of Student:

Geetanjali

230000613022

Msc (cs) 4th sem

ACKNOWLEDGMENT

I am thankful to my faculty and coordinators, for their most valuable and significant guidance throughout the course of the project and in elaborating our view of studying the project's details and getting the right vision for its implementation. Also, I would like to thank the Department of Computer Science Engineering, that has played an important role in strengthening my career by providing full cooperation and encouragement throughout the tenure of the project.

INTRODUCTION

Time is an integral part of our daily lives, and clocks serve not only as functional tools but also as aesthetic and personal style statements. **SnapBuy** is an innovative e-commerce platform dedicated to offering a wide range of clocks, catering to diverse customer preferences. From stylish wristwatches for men and women to colorful and engaging designs for kids, as well as elegant wall clocks that enhance home and office décor, **SnapBuy** aims to be a one-stop destination for all timekeeping needs.

Built using the **MERN (MongoDB, Express.js, React.js, Node.js) stack**, the platform ensures a seamless, efficient, and userfriendly shopping experience. The **frontend**, developed with React.js, provides an intuitive and responsive interface that allows customers to browse through categories, apply filters, and make informed purchasing decisions.

The **backend**, powered by Node.js and Express.js, ensures smooth server-side operations, including product management, user authentication, and order processing, while MongoDB efficiently handles the storage of user data, product listings, and order history. With an easy-to-use interface, secure checkout process, and advanced search functionality, **SnapBuy** ensures that users can effortlessly find and purchase the perfect clock to suit their needs.

The platform is designed to be scalable, responsive, and accessible across multiple devices, ensuring a consistent shopping experience for all users.

Whether customers are looking for a premium luxury watch, a budget-friendly everyday timepiece, or a statement wall clock for interior decoration, **SnapBuy** offers a well-curated selection that meets every requirement. With a commitment to quality, convenience, and customer satisfaction, **SnapBuy** aims to redefine the way people shop for clocks online by providing a reliable, engaging, and feature-rich e-commerce experience.

4.1 Aim:

- The aim of **SnapBuy** is to develop a fully functional and userfriendly e-commerce platform dedicated to selling a wide variety of clocks, including wristwatches for men, women, and kids, as well as decorative wall clocks for homes and offices.
- The goal is to provide a seamless and efficient shopping experience by integrating modern web technologies to ensure smooth navigation, secure transactions, and an intuitive product catalog. The platform is designed to offer a visually appealing interface that allows users to browse products effortlessly, apply advanced search filters, and make secure purchases with ease.

- Using the **MERN (MongoDB, Express.js, React.js, Node.js) stack**, the project aims to create a scalable and highperformance system that can handle large volumes of product listings, user data, and transactions efficiently. The focus is on building a secure authentication system, enabling safe login and payment methods to protect user information.

- Additionally, the platform is structured to support seamless product management, ensuring that new products can be easily added, modified, or removed. The website is designed to be fully responsive, making it accessible on multiple devices, including smartphones, tablets, and desktops, ensuring a smooth shopping experience for users from any location.

Performance optimization is also a key objective, ensuring fast page loading times and smooth interactions to enhance user engagement. By developing this platform, **SnapBuy** aims to provide a reliable and engaging online shopping destination where customers can find high-quality clocks with convenience and confidence..

4.2 Objectives:

The **SnapBuy** e-commerce platform is designed to provide a seamless and user-friendly shopping experience for customers looking to purchase a variety of clocks. The key objectives of this project are:

1.**User Authentication & Security**–Implement secure user

authentication using JWT for login, signup, and authorization to ensure data privacy and account security.

2.**Product Management**–Enable the addition, modification, and removal of clock products, ensuring an organized and up-to-date inventory.

3.**Advanced Search & Sorting**–Allow users to search for specific products and apply sorting filters based on price, category, and popularity to enhance the browsing experience.
4.**Cart Functionality**–Provide users with the ability to add items to the cart, update quantities, and remove products before checkout.

5.**Order Management**–Facilitate order placement, tracking, and history management for a seamless purchasing process.

6.**User Account & Profile Management**–Allow users to create new accounts, manage their profiles, view order history, and save favourite items for future purchases.

7.**Secure Payment Integration**–Implement a secure and reliable payment system to ensure smooth transactions.

8.**Responsive & Scalable Design**–Build a mobile-friendly and scalable platform using the MERN stack to provide a fast and efficient user experience across devices.

9.**Wishlist Feature**–Enable users to save products for later purchases, improving engagement and convenience.

10.**Performance Optimization**–Ensure fast loading times and efficient database queries for a smooth shopping experience. By achieving these objectives, **Snapbuy** aims to offer a reliable and engaging e-commerce platform that makes shopping for clocks easy, secure, and enjoyable.

4.3 Need:

The **Snapbuy** e-commerce platform is developed to meet the growing demand for an efficient and specialized online store dedicated to clocks. The key needs of this project are:

1.**Diverse Product Availability**–Ensure a wide range of clocks, including wristwatches for men, women, and kids, as well as wall clocks, catering to different customer preferences.

2.**User-Friendly Shopping Experience**–Provide a seamless and intuitive interface that allows users to browse, search, and purchase products with ease.

3.**Secure Authentication & Data Protection**–Implement JWT-based authentication to ensure user data security and protect transactions from unauthorized access.

4.**Advanced Search & Sorting**–Enable users to quickly find specific products by applying search filters such as category, price, and popularity.

5.**Efficient Cart & Order Management**–Allow users to add products to the cart, update quantities, remove items, and track their orders for a smooth purchasing process.

6.**Mobile Responsiveness**–Ensure the platform is fully optimized for various devices, including smartphones, tablets, and desktops, to provide a consistent shopping experience.

7.**Secure Payment Integration**–Implement a reliable and encrypted payment system to facilitate hassle-free transactions while ensuring customer trust.

8.**Scalability & Performance Optimization**–Develop a high-performance backend with optimized database queries to handle increasing user traffic and ensure fast loading times.

9.**Wishlist Feature**–Allow users to save favorite products for future purchases, enhancing customer engagement and retention.

10.**Efficient Product Management**–Provide an easy-to-use admin system for adding, updating, and removing products, ensuring a wellmaintained and updated catalog.

By fulfilling these needs, **Snapbuy** aims to create a reliable, feature-rich, and user-centric platform that enhances the online shopping experience for clock enthusiasts.

4.4 Functionalities:

The Snapbuy e-commerce platform is designed to provide a seamless and feature-rich shopping experience for users. The key functionalities of this project are:

1.User Authentication & Authorization–Secure login and signup system using JWT authentication to protect user accounts and ensure safe access.

2.Product Listing & Management–Display a wide variety of clocks with detailed descriptions, images, and pricing, while allowing admins to add, update, and delete products.

3.Search & Sorting Features–Enable users to search for specific clocks and apply sorting options based on price, category, and popularity to refine their shopping experience.

4.Shopping Cart System–Allow users to add products to the cart, update quantities, and remove items before proceeding to checkout.

5.Order Processing & Tracking–Enable users to place orders, view order details, and track their purchases in real time.

6.Wishlist Feature–Allow users to save products for future reference and easy access.

7.Responsive & Interactive UI–Provide a mobile-friendly and visually appealing interface that adapts to different screen sizes for an optimized shopping experience.

8.Secure Payment Gateway–Integrate a secure payment system to facilitate smooth and safe transactions.

9.Profile & Account Management–Allow users to update their profiles, manage addresses, and view order history for a personalized experience.

10.Admin Dashboard–Provide an admin panel to manage products, track sales, and monitor user activities efficiently.

11.Performance Optimization–Ensure fast loading times, smooth navigation, and an optimized backend to handle high traffic.

12.Review & Rating System–Enable users to leave reviews and rate products to help other buyers make informed decisions. By implementing these functionalities, Snapbuy ensures an efficient, secure, and userfriendly online shopping experience tailored to clock enthusiasts.

5.1 Project Definition:

The **Snapbuy** e-commerce platform is a web-based application designed to facilitate the seamless buying and selling of clocks. Built using the **MERN (MongoDB, Express.js, React.js, Node.js) stack**, the platform provides users with an interactive and feature-rich environment where they can browse, search, and purchase a wide range of clocks, including wristwatches for men, women, and kids, as well as decorative wall clocks.

The project focuses on creating a secure and scalable system that ensures smooth user authentication, efficient product management, and a hassle-free shopping experience.

The core functionality of **Snapbuy** includes **user authentication and authorization** using JWT, enabling customers to create accounts, log in securely, and manage their profiles.

The platform also features **an advanced search and sorting system**, allowing users to find products based on categories, price range, and popularity. A **dynamic shopping cart** ensures users can add, remove, and update items before checkout, while an **order tracking system** keeps them informed about their purchases. The integration of a **secure payment gateway** ensures safe transactions, enhancing customer trust and satisfaction.

For **administrators**, the platform includes an **admin dashboard** for managing product listings, tracking sales, and monitoring user

activity. Additionally, a **wishlist feature** allows users to save products for future purchases, and a **review and rating system** helps enhance product credibility by enabling customer feedback.

The project is designed to be **fully responsive**, ensuring accessibility across various devices such as desktops, tablets, and smartphones.

With a strong focus on **performance optimization, security, and user experience**, **Snapbuy** aims to provide a reliable and efficient ecommerce solution tailored specifically for clock enthusiasts, making online shopping easy, secure, and engaging.

5.2 Solution:

The **Snapbuy** e-commerce platform provides a **comprehensive and efficient solution** for buying and selling clocks online, ensuring a seamless shopping experience through a robust and scalable system. Built using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, it integrates modern technologies to offer a **secure, user-friendly,**

and high-performance platform that caters to a wide range of customers.

To address the need for a **diverse clock shopping experience**, **Snapbuy** features a well-organized product catalog, allowing users to browse and purchase wristwatches for men, women, and kids, along with decorative wall clocks. An **advanced search and sorting system** ensures users can efficiently find products based on categories, price, and popularity.

To enhance **user engagement and personalization**, the platform offers **secure user authentication** using JWT, allowing customers to create accounts, log in securely, and manage their profiles.

A **shopping cart system** enables users to add, update, and remove products before checkout, while an **order tracking feature** keeps them informed about their purchases. Additionally, a **wishlist functionality** lets users save products for future reference.
For **secure and hassle-free transactions**, **Snapbuy** integrates a **payment gateway**, ensuring smooth and encrypted payments. The platform also includes an **admin dashboard** for efficient product management, sales tracking, and user monitoring. A **review and rating system** allows customers to share feedback, enhancing product credibility.

To ensure **optimal performance and scalability**, **Snapbuy** is designed with **responsive UI components**, making it accessible across all devices, including desktops, tablets, and smartphones. The backend is optimized for **fast loading speeds**, ensuring seamless navigation and a smooth shopping experience.

By implementing these solutions, **Snapbuy** successfully bridges the gap between clock sellers and buyers, creating a **secure, efficient, and user-centric** e-commerce platform that enhances convenience and reliability in online clock shopping.

TOOLS AND TECHNIQUE USED

The code for this project is written using the language known as HTML, CSS, JS, Node js, ReactJs, Express js , MongoDB & BootStrap using Visual Studio Code.

6.1 HTML (Hypertext Markup Language):

HTML is the standard markup language used for creating web pages and applications. It serves as one of the core building blocks of the World Wide Web, alongside CSS and JavaScript. HTML structures content on the web, allowing browsers to

render text, images, and multimedia as part of a cohesive web page. HTML tags represent different elements, such as headings, paragraphs, links, and forms, giving semantic meaning to the content. By organizing the structure of a web page, HTML enables other technologies like CSS and JavaScript to style and add interactivity to the content.

6.2 CSS (Cascading Style Sheets):

CSS is a stylesheet language used to define the visual presentation of HTML documents. It allows developers to separate content from design, specifying elements like layout, colors, and fonts. CSS is a foundational technology of the Web, working with HTML and JavaScript. By storing CSS in an external file, multiple pages can share the same styling, making it easier to maintain and update the website's appearance. CSS also enables responsive design, allowing websites to adapt to different screen sizes, ensuring accessibility and a better user experience across devices.

6.3 JavaScript (JS):

JavaScript is a dynamic programming language essential for creating interactive and responsive web applications. As a core technology of the Web, alongside HTML and CSS, JavaScript allows developers to manipulate HTML elements in real-time,

enhancing user engagement through features like form validation, animations, and event handling. JavaScript is often used with libraries and frameworks to build complex, front-end applications and provide a seamless user experience.

6.4 Node.js:

Node.js is a powerful, open-source runtime environment that allows JavaScript to be executed on the server side. It is built on Chrome's V8 JavaScript engine and is widely used for developing scalable and high-performance web applications. Node.js is particularly suitable for handling real-time data and API requests, making it an ideal choice for back-end development in this project.

6.5 Express.js:

Express.js is a lightweight and flexible web application framework for Node.js, designed for building robust and scalable web applications and APIs. It simplifies the development of server-side applications by providing built-in functionalities for routing, middleware, and handling HTTP requests. Express.js enables the development of fast, modular back-end systems that seamlessly integrate with front-end components.

6.6 MongoDB:

MongoDB is a NoSQL database known for its flexibility and scalability. Instead of using tables and rows, MongoDB stores data in collections of JSON-like documents, making it ideal for handling unstructured data. Its document-oriented structure allows for dynamic schemas, which makes it a perfect fit for projects with changing or complex data requirements. MongoDB is often used with Node.js applications to provide a scalable, high-performance back-end.

6.7 Bootstrap:

Bootstrap is a popular open-source CSS framework used for designing responsive and mobile-first websites. It includes a collection of ready-made design templates, grid layouts, components, and utilities, allowing developers to create professional-looking websites efficiently. Bootstrap simplifies styling and layout, enabling a consistent design across web pages and making the website responsive on various devices and screen sizes.

6.8 React Js:

React.js is a popular **JavaScript library** used for building fast and interactive **user interfaces (UIs)**, particularly for **single-page applications (SPAs)**. It was developed by **Facebook (now Meta)** and is widely used in modern web development due to its efficiency and flexibility.

Key Features of React.js

1. **Component-Based Architecture** – UI is broken into reusable components, making development modular and maintainable.
2. **Virtual DOM** – React updates only the changed parts of the UI, improving performance.

3. **JSX (JavaScript XML)** – Allows writing HTML-like syntax within JavaScript, making code more readable.

4. **Unidirectional Data Flow** – Ensures predictable state management and easier debugging.

5. **Hooks & State Management** – Provides built-in hooks like useState and useEffect for managing component logic.

6.9 React Js:

JWT is used for user authentication and authorization. It helps in securely transmitting user data between the client and the server by generating encrypted tokens that verify identity and access permissions.

6.10 Bcrypt.js

Bcrypt.js is a **password-hashing library** that ensures secure storage of user passwords. It uses **salted hashing techniques** to prevent unauthorized access and protect user credentials.

6.11. Razorpay

Razorpay is a **payment gateway API** that facilitates secure online transactions. It allows users to make payments using various methods like **credit/debit cards, UPI, net banking, and wallets**.

6.12. CORS (Cross-Origin Resource Sharing)

CORS is a security feature that enables or restricts access to resources from different origins. It is essential for handling API requests between the frontend and backend when they run on separate domains or ports.

9. Express Formidable

Express Formidable is used for handling **multipart form data**, allowing users to upload images and other files easily when managing products or user profiles.

10. Slugify

Slugify is a utility that **converts text into URL-friendly slugs**, ensuring SEO-friendly product URLs for better search engine ranking and user navigation.

Code :

```
import React from "react";
import { Routes, Route } from "react-router-dom";
import Homepage from "./pages/HomePage";
import About from "./pages/About";
import Contact from "./pages/Contact";
import Policy from "./pages/Policy";
import PageNotFound from "./pages/PageNotFound";
import Register from "./pages/Auth/Register";
import Login from "./pages/Auth/Login";
import Dashboard from "./pages/user/Dashboard";
import PrivateRoute from "./component/Routes/PrivateRoute";
import ForgotPassword from "./pages/Auth/ForgotPassword";
import AdminRoute from "./component/Routes/AdminRoute";
import AdminDashBoard from "./pages/Admin/AdminDashboard";
import CreateCategory from "./pages/Admin/CreateCategory";
import CreateProduct from "./pages/Admin/CreateProduct";
import User from "./pages/Admin/Users";
import Orders from "./pages/user/Orders";
import Profile from "./pages/user/Profile";
import Products from "./pages/Admin/Products";
import UpdateProduct from "./pages/Admin/UpdateProduct";
import Search from "./pages/Search";
import ProductDetails from "./pages/ProductDetails";
import CartPage from "./pages/user/CartPage";
import Success from "./pages/Success";

export default function App() {
  return (
    <Routes>
      <Route path="/" element={<Homepage />} />
      <Route path="/product/:slug" element={<ProductDetails />} />
```

```
    <Route path="/search" element={<Search />} />
    <Route path="/cart" element={<CartPage />} />
    <Route path="/dashboard" element={<PrivateRoute />}>
      <Route path="user" element={<Dashboard />} />
      <Route path="user/orders" element={<Orders />} />
      <Route path="user/profile" element={<Profile />} />
    </Route>
    <Route path="/dashboard" element={<AdminRoute />}>
      <Route path="admin" element={<AdminDashBoard />} />
      <Route path="admin/create-category" element={<CreateCategory />} />
      <Route path="admin/create-products" element={<CreateProduct />} />
      <Route path="admin/product/:slug" element={<UpdateProduct />} />
      <Route path="admin/products" element={<Products />} />
      <Route path="admin/users" element={<User />} />
    </Route>
    <Route path="/forgot-password" element={<ForgotPassword />} />
    <Route path="/about" element={<About />} />
    <Route path="/register" element={<Register />} />
    <Route path="/contact" element={<Contact />} />
    <Route path="/policy" element={<Policy />} />
    <Route path="/login" element={<Login />} />
    <Route path="/success" element={<Success />} />
    <Route path="*" element={<PageNotFound />} />
   </Routes>
  );
}
```

```
//HomePage

import React, { useState, useEffect } from "react";
import Layout from "../component/layout/Layout";
import { useAuth } from "../context/auth";
import axios from "axios";
import { Checkbox, Radio } from "antd";
import { useNavigate } from "react-router-dom";
import { useCart } from "../context/cart";
import toast from "react-hot-toast";
import { Badge } from "antd";

export default function HomePage() {
  const navigate = useNavigate();
  const [auth, setAuth] = useAuth();
  const [cart, setCart] = useCart();
  const [products, setProducts] = useState([]);
  const [categories, setCategories] = useState([]);
  const [checked, setChecked] = useState([]);
  const [radio, setRadio] = useState([]);
  const [total, setTotal] = useState(0);
  const [page, setPage] = useState(1);
  const [loading, setLoading] = useState(false);

  // Price options
  const priceOptions = [
    { _id: 1, name: "Under ₹500", array: [0, 500] },
    { _id: 2, name: "₹500 - ₹1000", array: [500, 1000] },
    { _id: 3, name: "₹1000 - ₹2000", array: [1000, 2000] },
    { _id: 4, name: "Above ₹2000", array: [2000, 10000] },
  ];

  // Get all categories
```

```
const getAllCategories = async () => {
  try {
    setLoading(true);
    const { data } = await axios.get(
      `${import.meta.env.VITE_API_URL}/api/v1/category/get-category`
    );
    setCategories(data.category);
    setLoading(false);
  } catch (error) {
    setLoading(false);
    console.log(error);
  }
};

// Get total product count
const getTotal = async () => {
  try {
    const { data } = await axios.get(
      `${import.meta.env.VITE_API_URL}/api/v1/product/product-count`
    );
    if (data?.success) {
      setTotal(data.total);
    }
  } catch (error) {
    console.log(error);
  }
};

// Filter products based on category and price
const filterProduct = async () => {
  try {
    setLoading(true);
    const { data } = await axios.post(
      `${import.meta.env.VITE_API_URL}/api/v1/product/product-filters`,
```

```
      { checked, radio }
    );
    setProducts(data?.products || []);
    setLoading(false);
  } catch (error) {
    setLoading(false);
    console.log(error);
  }
};

// Handle category filter change
const handleFilter = (value, id) => {
  let updatedChecked = [...checked];
  if (value) {
    updatedChecked.push(id);
  } else {
    updatedChecked = updatedChecked.filter((c) => c !== id);
  }
  setChecked(updatedChecked);
};

// Load more products
const loadMore = async () => {
  try {
    setLoading(true);
    const { data } = await axios.get(
      `${import.meta.env.VITE_API_URL}/api/v1/product/product-list/${page}`
    );
    setProducts((prev) => [...prev, ...data?.products]);
    setLoading(false);
  } catch (error) {
    setLoading(false);
    console.log(error);
  }
```

```
};

// Initial data load
useEffect(() => {
  getAllCategories();
  getTotal();
  filterProduct();
}, []);

// Load more products on page change
useEffect(() => {
  if (page > 1) loadMore();
}, [page]);

// Trigger filter when category/price changes
useEffect(() => {
  filterProduct();
}, [checked, radio]);

return (
  <Layout title={"All Products - Best offers"}>
    <div className="homepage">
      <div className="row">
        {/* ======= Filter Section ======= */}
        <div className="col-md-3 filter-section">
          {/* Filter by Category */}
          <h4 className="filter-title">Filter by Category</h4>
          {categories?.map((c) => (
            <div key={c._id} className="filter-item">
              <Checkbox
                onChange={(e) => handleFilter(e.target.checked, c._id)}
                checked={checked.includes(c._id)}
              >
                {c.name}
```

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

 {/* Filter by Price */}
 <h4 className="filter-title mt-4">Filter by Price</h4>
 <Radio.Group
  value={JSON.stringify(radio)}
  onChange={(e) => setRadio(JSON.parse(e.target.value))}
  className="filter-item"
 >
  {priceOptions.map((p) => (
   <Radio key={p._id} value={JSON.stringify(p.array)}>
    {p.name}
   </Radio>
  ))}
 </Radio.Group>

 {/* Reset Filters */}
 <div className="mt-4">
  <button
   className="btn btn-danger w-100"
   onClick={() => window.location.reload()}
  >
   Reset Filters
  </button>
 </div>
</div>

{/* ======= Product Section ======= */}
<div className="col-md-9 product-section">
 <h2 className="section-title">All Products</h2>
 <div className="row">
  {products.length > 0 ? (
```

```
products.map((p) => (
  <div key={p._id} className="col-md-4 mb-4">
    <div className="card product-card">
      <img
        className="card-img-top"
        src={`${
          import.meta.env.VITE_API_URL
        }/api/v1/product/product-photo/${p._id}`}
        alt={p.name}
      />
      <div className="card-body">
        <h5 className="card-title">{p.name}</h5>
        <p className="card-price">₹{p.price}</p>
        <p className="card-text">
          {p.description.substring(0, 30)}...
        </p>
        <div className="btn-group">
          <button
            className="btn btn-outline-primary"
            onClick={() => navigate(`/product/${p.slug}`)}
          >
            More Details
          </button>
          <button
            className="btn btn-outline-success"
            onClick={() => {
              setCart([...cart, p]);
              localStorage.setItem(
                "cart",
                JSON.stringify([...cart, p])
              );
              toast.success("Item added to cart");
            }}
          >
```

```
                Add to Cart
              </button>
            </div>
          </div>
        </div>
      </div>
    ))
  ) : (
    <p>May Take some time to load...</p>
  )}
</div>

{/* Load More Button */}
{products.length < total && (
  <div className="text-center mt-4">
    <div className="spinner-grow text-primary" role="status">
      <span className="sr-only">
        <br />
        Loading...
      </span>
    </div>
  </div>
)}
</div>
</div>
</div>
</Layout>
);
}
```

```
//ProductDetails.jsx

import React, { useEffect, useState } from "react";
import Layout from "../component/layout/Layout";
import axios from "axios";
import { useParams, useNavigate } from "react-router-dom";
import { useCart } from "../context/cart";
import toast from "react-hot-toast";

export default function ProductDetails() {
  const navigate = useNavigate();
  const params = useParams();
  const [product, setProduct] = useState({});
  const [relatedProducts, setRelatedProducts] = useState([]);
  const [cart, setCart] = useCart();

  // Initial product details
  useEffect(() => {
    if (params?.slug) {
      getProduct();
    }
  }, [params.slug]);

  // Get product
  const getProduct = async () => {
    try {
      const { data } = await axios.get(
        `${import.meta.env.VITE_API_URL}/api/v1/product/get-product/${
          params.slug
        }`
      );
      setProduct(data?.product);
      getSimilarProducts(data?.product._id, data?.product.category._id);
    } catch (error) {
```

```
    console.log(error);
  }
};

// Get similar products
const getSimilarProducts = async (pid, cid) => {
  try {
    const { data } = await axios.get(
      `${
        import.meta.env.VITE_API_URL
      }/api/v1/product/related-product/${pid}/${cid}`
    );
    setRelatedProducts(data?.products);
  } catch (error) {
    console.log(error);
  }
};

// Add to Cart Function
const handleAddToCart = (product) => {
  const updatedCart = [...cart, product];
  setCart(updatedCart);
  localStorage.setItem("cart", JSON.stringify(updatedCart));
  toast.success("Item added to cart");
};

return (
  <Layout>
    <div className="product-details container">
      <div className="row">
        {/* Product Image */}
        <div className="col-md-6">
          <img
            className="product-image"
```

```
    src={`${
      import.meta.env.VITE_API_URL
    }/api/v1/product/product-photo/${product._id}`}
    alt={product.name}
   />
  </div>

  {/* Product Info */}
  <div className="col-md-6">
   <div className="product-info">
    <h1 className="product-title">{product.name}</h1>
    <p className="product-description">{product.description}</p>
    <h4 className="product-price">₹{product.price}</h4>
    <p className="product-category">
     Category: <span>{product.category?.name}</span>
    </p>
    <button
     className="btn btn-primary"
     onClick={() => handleAddToCart(product)}
    >
     Add to Cart
    </button>
   </div>
  </div>
</div>

{/* Similar Products */}
<div className="similar-products">
 <h3>Similar Products</h3>
 <div className="row">
  {relatedProducts.length > 0 ? (
   relatedProducts.map((p) => (
    <div key={p._id} className="col-md-4">
     <div className="card">
```

```
          <img
            className="card-img-top"
            src={`${
              import.meta.env.VITE_API_URL
            }/api/v1/product/product-photo/${p._id}`}
            alt={p.name}
            style={{
              height: "180px",
              objectFit: "cover",
            }}
          />
          <div className="card-body">
            <h5 className="card-title">{p.name}</h5>
            <p className="card-text">₹{p.price}</p>
            <button
              className="btn btn-outline-primary"
              onClick={() => navigate(`/product/${p.slug}`)}
            >
              View Details
            </button>
            <button
              onClick={() => handleAddToCart(p)}
              className="btn btn-primary m-3"
            >
              Add to Cart
            </button>
          </div>
        </div>
      </div>
    ))
  ) : (
    <p>No similar products found.</p>
  )}
</div>
```

```
      </div>
    </div>
  </Layout>
 );
}

//CartPage
import React from "react";
import Layout from "../../component/layout/Layout";
import { useCart } from "../../context/cart";
import { useAuth } from "../../context/auth";
import { useNavigate } from "react-router-dom";
import toast from "react-hot-toast";

export default function CartPage() {
 const [auth] = useAuth(); // No need to setAuth
 console.log("Auth State:", auth);

 const [cart, setCart] = useCart();
 const navigate = useNavigate();

 // Total price function
 const totalPrice = () => {
  try {
   let total = 0;
   cart?.forEach((item) => {
    total += item.price || 0;
   });
   return total;
  } catch (error) {
   console.log(error);
   return 0;
  }
 };
```

```
// Remove item from cart
const removeCartItem = (pid) => {
  try {
    let myCart = [...cart];
    let index = myCart.findIndex((item) => item._id === pid);
    myCart.splice(index, 1);
    setCart(myCart);
    localStorage.setItem("cart", JSON.stringify(myCart));
    toast.success("Item removed successfully");
  } catch (error) {
    console.log(error);
  }
};

// Handle checkout using Razorpay
const handleCheckout = async () => {
  if (!auth?.token) {
    toast.error("Please login to continue");
    navigate("/login");
    return;
  }

  try {
    const response = await fetch(
      `${import.meta.env.VITE_API_URL}/api/v1/product/create-order`,
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
          Authorization: `Bearer ${auth.token}`,
        },
        body: JSON.stringify({ total: totalPrice() }),
      }
```

```
    );

    const data = await response.json();

    if (!data.success) return toast.error(data.message);

    const options = {
      key: import.meta.env.VITE_RAZORPAY_KEY,
      amount: data.order.amount,
      currency: data.order.currency,
      order_id: data.order.id,
      handler: () => {
        toast.success("Payment Successful");
        navigate("/success");
      },
      prefill: {
        name: auth.user.name,
        email: auth.user.email,
        contact: "9991423362",
      },
      theme: { color: "#3399cc" },
    };

    new window.Razorpay(options).open();
  } catch (error) {
    console.error("Payment error:", error);
    toast.error("Failed to process payment");
  }
};

return (
  <Layout title={"Cart"}>
    <div className="container">
      <div className="row">
```

```
  <div className="col-md-12">
   <h1 className="text-center bg-light p-2">
    {auth?.token ? `Hello ${auth.user.name}` : "Welcome to SnapBuy"}
   </h1>
   <h4 className="text-center">
    {cart?.length > 0
     ? auth?.token
      ? `You have ${cart.length} items in your cart`
      : "Please login to check out"
     : "Your cart is empty"}
   </h4>
  </div>
</div>

{/* Main Row */}
<div className="d-flex align-items-start mt-4">
 {/* Product List */}
 <div className="col-md-9">
  {cart?.map((p, index) => (
   <div className="row m-3 p-3 card flex-row" key={index}>
    <div className="col-md-4">
     <img
      className="card-img-top"
      width="100px"
      height="100px"
      src={`${
       import.meta.env.VITE_API_URL
      }/api/v1/product/product-photo/${p._id}`}
      alt={p.name}
     />
    </div>
    <div className="col-md-8">
     <h6>{p.name}</h6>
     <p>
```

```
            {p.description.length > 50
              ? `${p.description.substring(0, 50)}...`
              : p.description}
            </p>
            <p>
              <b>Price:</b> ₹{p.price}
            </p>
            <button
              className="btn btn-danger"
              onClick={() => removeCartItem(p._id)}
            >
              Remove
            </button>
          </div>
        </div>
      ))}
    </div>

    {/* Checkout Section */}
    <div className="col-md-3 text-center">
      <div className="card p-3">
        <h4>Checkout</h4>
        <p>Total Items: {cart.length}</p>
        <p>Total Price: ₹{totalPrice()}</p>
        <button
          className="btn btn-primary w-100"
          onClick={handleCheckout}
        >
          Proceed to Payment
        </button>
      </div>
    </div>
  </div>
</div>
```

```
    </Layout>
  );
}



//PrivateRoutes
import { useState, useEffect } from "react";
import { useAuth } from "../../context/auth";
import { Outlet, useNavigate } from "react-router-dom";
import axios from "axios";
import Spinner from "../Spinner";

export default function PrivateRoute() {
  const [ok, setOk] = useState(false);
  const [auth, setAuth] = useAuth();
  const navigate = useNavigate();

  useEffect(() => {
    const authCheck = async () => {
      const res = await axios.get(
        `${import.meta.env.VITE_API_URL}/api/v1/auth/user-auth `
      );

      if (res.data) {
        setOk(true);
      } else {
        setOk(false);
      }
    };

    if (auth?.token) {
      authCheck();
    }
```

```
  }, [auth?.token, navigate]);

  return ok ? <Outlet /> : <Spinner path="" />;
}


//DashBoard

import React from "react";
import Layout from "../../component/layout/Layout";
import UserMenu from "../../component/layout/UserMenu";
import { useAuth } from "../../context/auth";

export default function Dashboard() {
  const [auth] = useAuth();
  return (
    <Layout title={"DashBoard -All Users"}>
      <div className="container-fluid m-3 p-3">
        <div className="row">
          <div className="col-md-3">
            <UserMenu />
          </div>
          <div className="col-md-9">
            <div className="card w-75 p-3">
              <h3>{auth?.user?.name}</h3>
              <h3>{auth?.user?.email}</h3>
              <h3>{auth?.user?.address}</h3>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
}
```

```
//AdminRoutes

import { useState, useEffect } from "react";
import { useAuth } from "../../context/auth";
import { Outlet, useNavigate } from "react-router-dom";
import axios from "axios";
import Spinner from "../Spinner";

export default function AdminRoute() {
  const [ok, setOk] = useState(false);
  const [auth] = useAuth();
  const navigate = useNavigate();

  useEffect(() => {
    const authCheck = async () => {
      const res = await axios.get(
        `${import.meta.env.VITE_API_URL}/api/v1/auth/admin-auth `
      );

      if (res.data) {
        setOk(true);
      } else {
        setOk(false);
      }
    };

    if (auth?.token) {
      authCheck();
    }
  }, [auth?.token, navigate]);

  return ok ? <Outlet /> : <Spinner />;
}
```

```
                        //Forgot Password.jsx
import React, { useState } from "react";
import Layout from "../../component/layout/Layout";
import { toast } from "react-hot-toast";
import axios from "axios";
import { useNavigate } from "react-router-dom";

export default function ForgotPassword() {
  const [email, setEmail] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [answer, setAnswer] = useState("");
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post(
        `${import.meta.env.VITE_API_URL}/api/v1/auth/forgot-password`,
        { email, newPassword, answer }
      );

      if (res.data.success) {
        toast.success(res.data.message);
        navigate("/login");
      } else {
        toast.error(res.data.message);
      }
    } catch (error) {
      console.error(error);
      toast.error("Something went wrong. Please try again.");
```

```
  }
};

return (
  <Layout title="Forgot Password">
    <div className="container d-flex justify-content-center align-items-center mt-5">
      <div className="col-md-6">
        <div className="card shadow p-4">
          <h2 className="text-center mb-4">Reset Password</h2>
          <form onSubmit={handleSubmit}>
            {/* Email Field */}
            <div className="mb-3">
              <label htmlFor="email" className="form-label">
                Email
              </label>
              <input
                type="email"
                value={email}
                className="form-control"
                onChange={(e) => setEmail(e.target.value)}
                required
                id="email"
                placeholder="Enter your email"
              />
            </div>

            {/* Security Question */}
            <div className="mb-3">
              <label htmlFor="answer" className="form-label">
                Security Answer (Your favorite sport)
              </label>
              <input
                type="text"
                value={answer}
```

```
            className="form-control"
            onChange={(e) => setAnswer(e.target.value)}
            required
            id="answer"
            placeholder="Enter your favorite sport name"
          />
        </div>

        {/* New Password Field */}
        <div className="mb-3">
          <label htmlFor="newPassword" className="form-label">
            New Password
          </label>
          <input
            type="password"
            value={newPassword}
            onChange={(e) => setNewPassword(e.target.value)}
            className="form-control"
            id="newPassword"
            placeholder="Enter your new password"
            required
          />
        </div>

        {/* Submit Button */}
        <button type="submit" className="btn btn-primary w-100">
          Reset Password
        </button>
      </form>
    </div>
  </div>
 </div>
 </Layout>
);
```

```
}


//server.js
import express from "express";
import dotenv from "dotenv";
import morgan from "morgan";
import connectDB from "./config/db.js";
import authRoutes from "./routes/authroute.js";
import categoryRoutes from "./routes/CategoryRoutes.js";
import productRoutes from "./routes/ProductRoutes.js";
import cors from "cors";

const app = express();

dotenv.config();

app.use(
  cors({
    origin: "http://localhost:5173",
    credentials: true,
    methods: ["GET", "POST", "PUT", "DELETE"],
  })
);

// Middleware
app.use(express.json());
app.use(morgan("dev"));

// Connect to database
connectDB();

// Routes
```

```
app.use("/api/v1/auth", authRoutes);
app.use("/api/v1/category", categoryRoutes);
app.use("/api/v1/product", productRoutes);

// Test route
app.get("/", (req, res) => {
  res.send("<h1>Welcome to ecommerce website</h1>");
});

// Start server
const PORT = process.env.PORT || 8080; // Default to 8080 if no .env variable
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});


//authRoute.js

import express from "express";
import {
  loginController,
  registerController,
  testController,
  forgotpasswordController,
  updateProfileController,
} from "../controllers/authContoller.js";

import { isAdmin, requireSignIn } from "../middlewares/authMiddleWare.js";

const router = express.Router();

router.post("/register", registerController);

router.post("/login", loginController);
```

```
router.post("/forgot-password", forgotpasswordController);

router.get("/test", requireSignIn, isAdmin, testController);

router.get("/user-auth", requireSignIn, (req, res) => {
  res.status(200).json({ ok: true });
});

router.get("/admin-auth", requireSignIn, isAdmin, (req, res) => {
  res.status(200).send({ ok: true });
});

//update route
router.put("/profile", requireSignIn, updateProfileController);

export default router;


//categoryRoutes.js

import express from "express";
import { isAdmin, requireSignIn } from "../middlewares/authMiddleWare.js";
import {
  categoryController,
  createCategoryController,
  deleteCategoryController,
  singleCategoryController,
  updateCategoryController,
} from "../controllers/categoryController.js";

const router = express.Router();

// Create category
```

```
router.post(
  "/create-category",
  requireSignIn,
  isAdmin,
  createCategoryController
);

// Update category
router.put(
  "/update-category/:id",
  requireSignIn,
  isAdmin,
  updateCategoryController
);

// Get all categories
router.get("/get-category", categoryController);

// Get single category
router.get("/single-category/:slug", singleCategoryController);

// Delete category
router.delete(
  "/delete-category/:id",
  requireSignIn,
  isAdmin,
  deleteCategoryController
);

export default router;
```

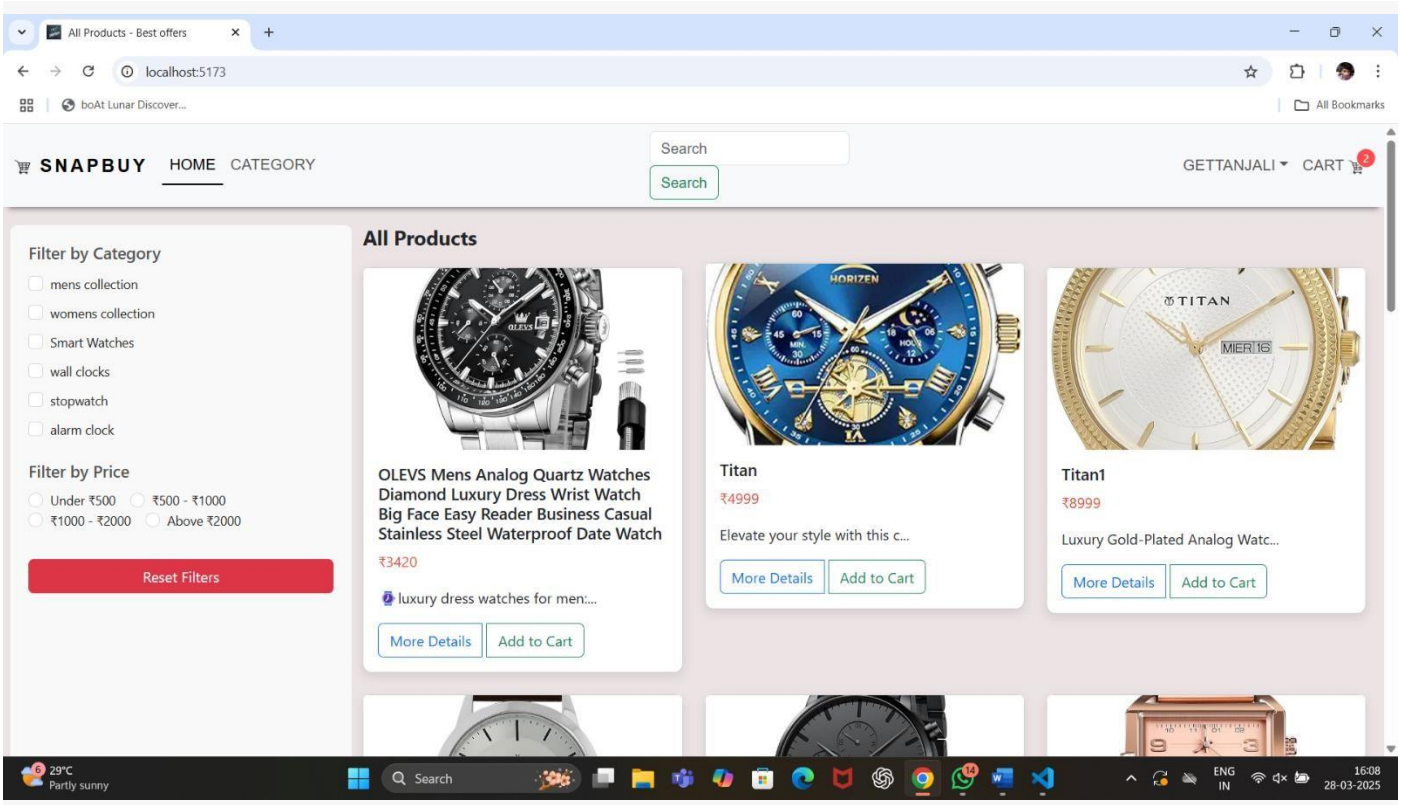
```
//ProductRotes.js

import express from "express";
import { isAdmin, requireSignIn } from "../middlewares/authMiddleWare.js";
import {
  createOrder,
  createProductController,
  deleteController,
  getProductController,
  getSingleProductController,
  ProductCountController,
  productFiltersController,
  productListController,
  productPhotoController,
  relatedProductController,
  searchProductController,
  UpdateProductController,
} from "../controllers/productController.js";
import formidable from "express-formidable";

const router = express.Router();

// Create Product
router.post(
  "/create-product",
  requireSignIn,
  isAdmin,
  formidable(),
  createProductController
);

// Update Product
router.put(
  "/update-product/:pid",
```

```
  requireSignIn,
  isAdmin,
  formidable(),
  UpdateProductController
);

// Get All Products
router.get("/get-product", getProductController);

// Get Single Product
router.get("/get-product/:slug", getSingleProductController);

// Get Product Photo
router.get("/product-photo/:pid", productPhotoController);

// Delete Product
router.delete("/delete-product/:pid", deleteController);

//filter product
router.post("/product-filters", productFiltersController);

//product count
router.get("/product-count", ProductCountController);

//product per page
router.get("/product-list/:page", productListController);

//search product4
router.get("/search/:keyword", searchProductController);

//similar products
router.get("/related-product/:pid/:cid", relatedProductController);

router.post("/create-order", requireSignIn, createOrder);
```

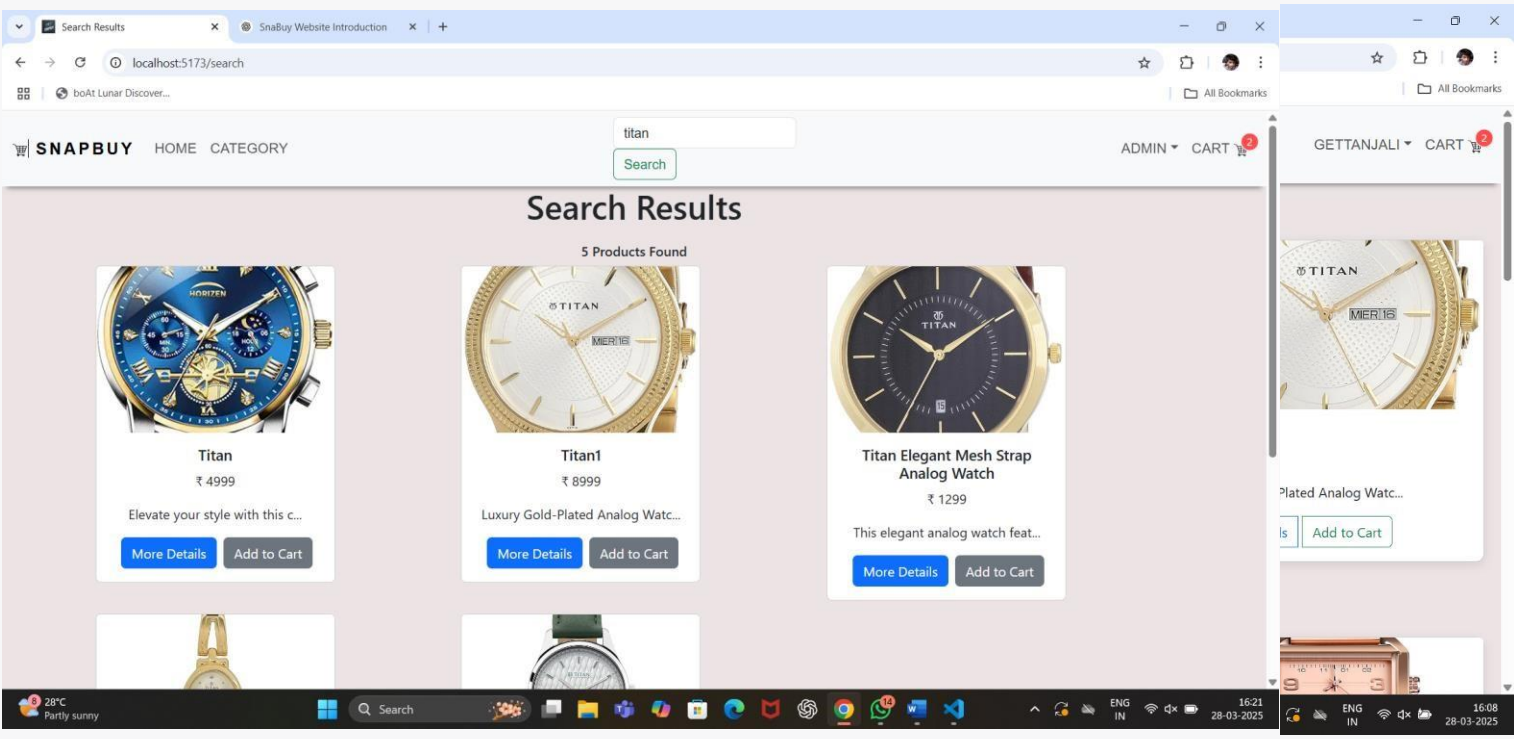
```
export default router;



//isAdmin

import JWT from "jsonwebtoken";
import userModel from "../models/userModel.js";

// Protected Route from base
export const requireSignIn = async (req, res, next) => {
  try {
    const token = req.headers.authorization?.split(" ")[1]; // Extract token
    if (!token) {
      return res
        .status(401)
        .json({ message: "Unauthorized - No token provided" });
    }

    const decode = JWT.verify(token, process.env.JWT_SECRET);
    req.user = decode;
    next();
  } catch (error) {
    console.log("JWT Error:", error);
    return res.status(401).json({ message: "Invalid or expired token" });
  }
};

// Admin Access
export const isAdmin = async (req, res, next) => {
  try {
    const user = await userModel.findById(req.user._id);
    if (user.role !== 1) {
```

```
      return res
        .status(401)
        .json({ success: false, message: "Unauthorized Access" });
    } else {
      next();
    }
  } catch (error) {
    console.log("Admin Middleware Error:", error);
    return res
      .status(401)
      .json({ success: false, message: "Error in admin middleware" });
  }
};
```

OUTPUT

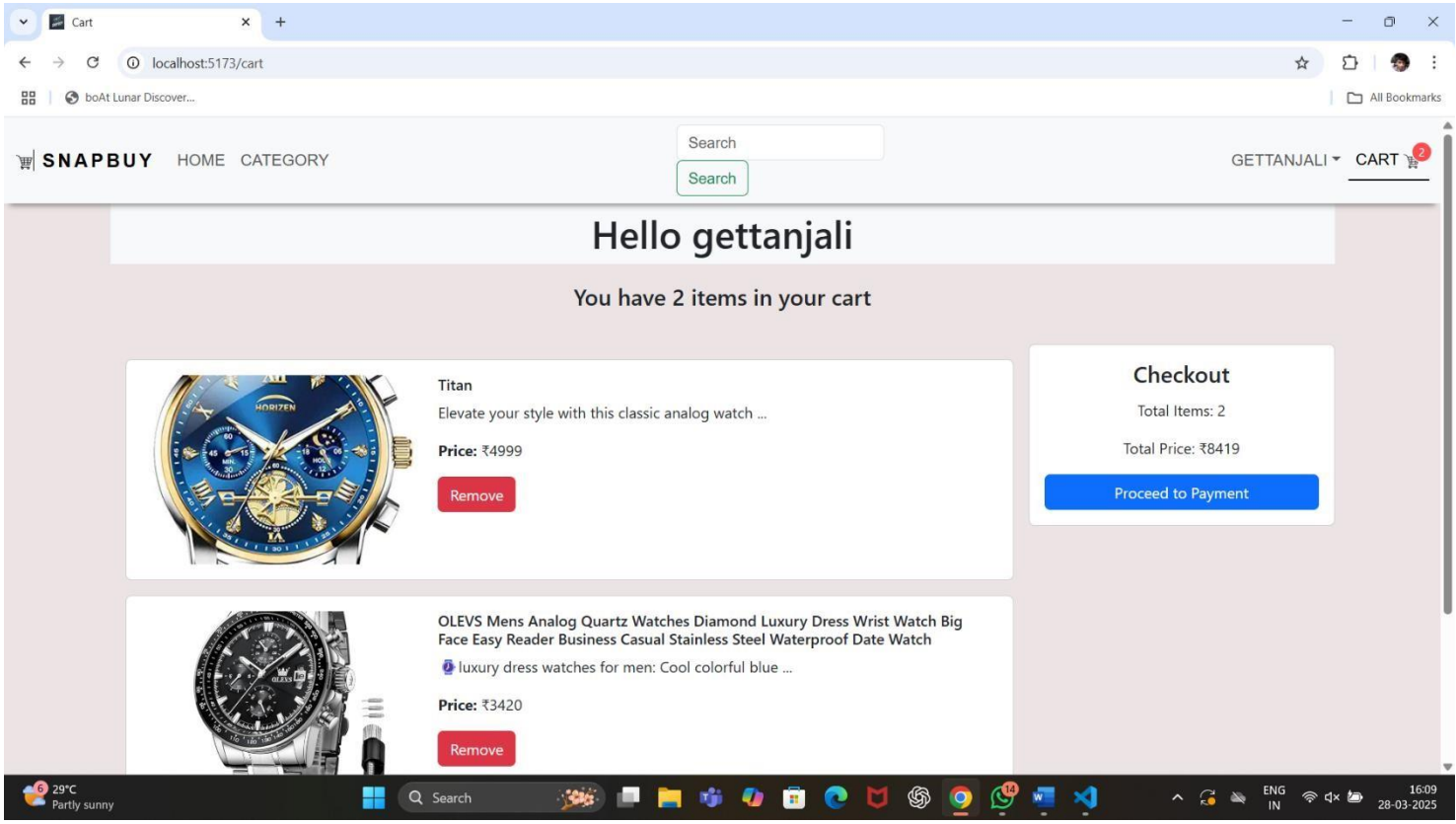


SnapBuy Ecommerce web app

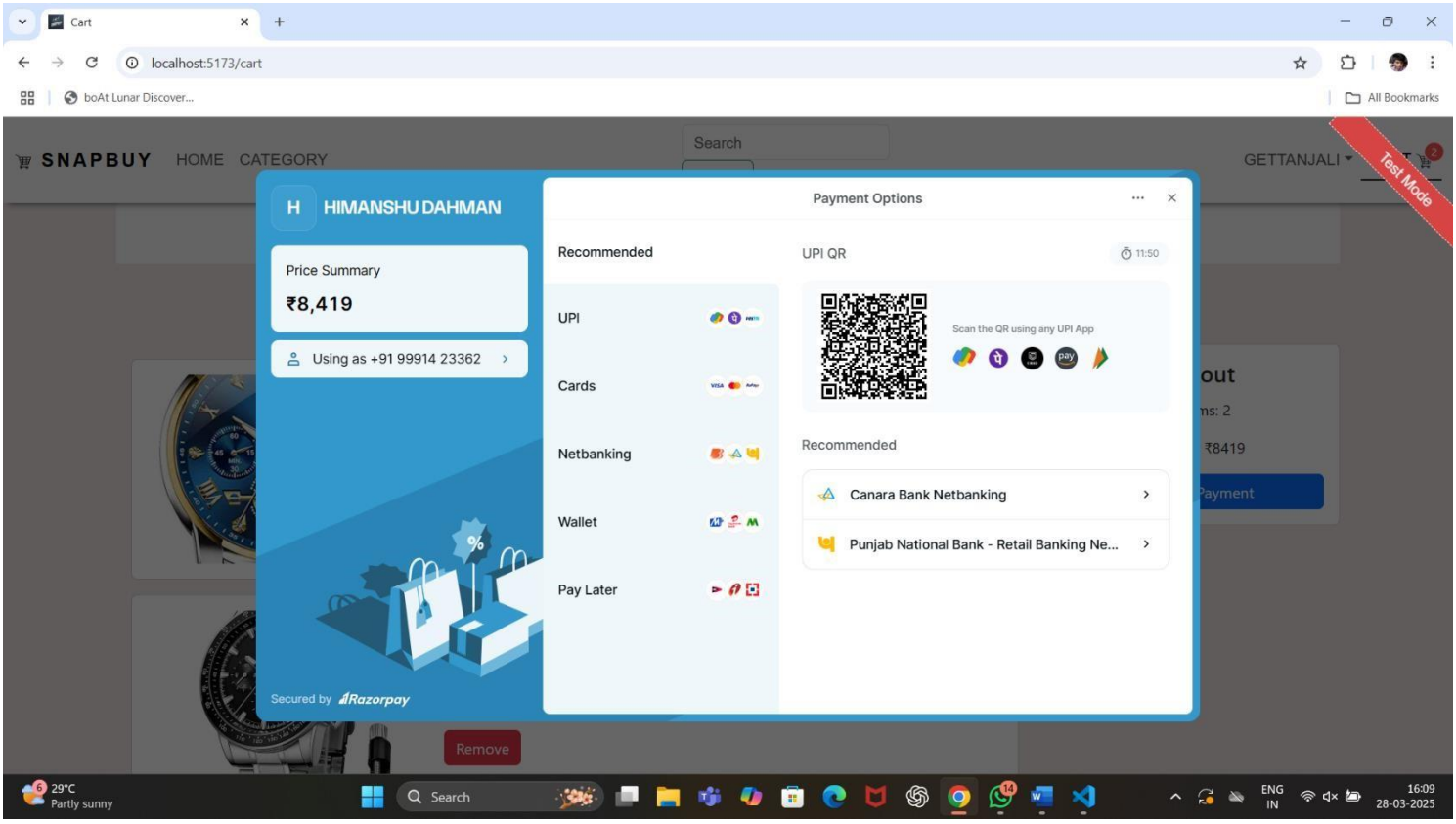


HomePage

Search Function by Backend Api Calling


Cart
localhost:5173/cart
boAt Lunar Discover...
All Bookmarks
SNAPBUY
HOME
CATEGORY
Search
Search
GETTANJALI
CART
2
Hello gettanjali
You have 2 items in your cart
Titan
Elevate your style with this classic analog watch ...
Price: ₹4999
Remove
OLEVS Mens Analog Quartz Watches Diamond Luxury Dress Wrist Watch Big Face Easy Reader Business Casual Stainless Steel Waterproof Date Watch
luxury dress watches for men: Cool colorful blue ...
Price: ₹3420
Remove
Checkout
Total Items: 2
Total Price: ₹8419
Proceed to Payment
29°C
Partly sunny
Search
ENG
IN
16:09
28-03-2025

CartPage



Payment Gateway using Razorpay

CONCLUSION

By utilizing these major technologies, **Snapbuy** ensures a **highly secure, scalable, and feature-rich e-commerce platform** with a **responsive UI,**

efficient data management, and seamless payment processing. These tools collectively provide a **powerful and optimized solution** for both customers and administrators, ensuring a **smooth online shopping experience**.