Project Report
On
E-commerce Application using MERN stack

<u>Submitted By:</u>

C Narasimhulu	**R170574**
T Sai Kumar	**R170575**
G Prasad	**R170649**

<u>Under the guidance of:</u>

Ms.Udaya Sree,

Assistant professor,

Department of CSE.

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge Technologies

(Catering the Educational Needs of Gifted Rural Youth of A.P)

R.K Valley, Y.S.R Kadapa (Dist.)-516330

Rajiv Gandhi University of Knowledge Technologies RK
Valley, Kadapa (Dist), Andhra Pradesh, 516330

DECLARATION

We hereby declare that the report of the B.Tech Mini Project Work entitled with the “E-Commmerce Application Using MERN Stack” which is being submitted to Rajiv Gandhi University of Knowledge Technologies, RK Valley, in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide report of the work carried out by me. The material contained in this report has not been submitted to any university or institution for award of any degree.

C Narasimhulu	**R170574**
T Sai Kumar	**R170575**
G Prasad	**R170649**

Rajiv Gandhi University of Knowledge Technologies RK
Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled "Ecommerce Application Using MERN Stack " is a bonafied project work submitted by C Narasimhulu,G.Prasad,T.Sai kumar in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2022-2023 carried out the work under the supervision

Guide	**Head of the Department**
MS.UDAYA SREE	**MR.SATYANANDARAM**
Assistant professor,	HOD OF CSE,
Department of CSE.	RGUKT RK Valley.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director,Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr.SATYANANDARAM for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project. I would like to convey thanks to our guide at college MS.UDAYA SREE for his guidance, encouragement, co-operation and kindness during the entire duration of the course and academics. My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

Abstract

In today's generation, most people are using technology for leading their lives and fulfilling their daily needs. In this generation most of us using E-commerce websites for shopping for clothes, groceries, and electronics. We have developed one E-commerce web application by using MERN stack technology as it contains MongoDB, Express.JS framework, React.JS library, Node.JS platform. This application is fully functional with different views for user and admin and it also has integrated with payment gateway for checkout. By using this website we can buy different types of t-shirts and we can choose different styles of t-shirts based upon customer interests. In this project, we can add different products and can delete them also. We have developed administrative functions for the website such as create a product, create categories, Admin dashboard, Manage products, Manage categories. For customers, they can quickly add their items to the cart. Based on the items in the cart then the bill gets generate and the customer can pay by using stripe.

KEYWORDS: JavaScript, React.js, MongoDB, Node.js, Express.js,BootStrap

INDEX:

Introduction

1.1 Overview

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

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

1.2 Scope of project

Purchasing and selling products and services over the internet without the need of going physically to the market is what online shopping all about. Online shopping is just like a retail store shopping that we do by going to the market, but it is done through the internet. Online shopping has made shopping painless and added more fun. Online stores offer product description, pictures, comparisons, price and much more. Few examples of these are Amazon.com, ebay.com, framt.com and the benefits of online shopping is that by having direct access to consumer ,the online stores can offer products that cater to the needs of consumer ,cookies can be used for tracking the customer selection over the internet or what is of their interest when they visit the site again . Online shopping makes use of digital technology for managing the flow of information, products, and payment between consumer, site owners and suppliers.

1.3 Purpose of project

The purpose of online shopping is to save time, save money. Through online shopping one can save his valuable time. One can watch and select things he wanna to buy. Through online shopping we can save our money because prices are less than market prices and we receive our bought things at our home. No need to go anywhere and do shopping. We can get different varieties of things online and we can choose which one we want.

1.4 Proposed System

The development of this new system contains the following activities, which try to develop online application by keeping the entire process in the view of database integration approach.

• Secure registration and profile management facilities for Customers.

• Browsing through the e-Mall to see the items that are there in each category of products like Apparel, Kitchen accessories, Bath accessories, Food items etc.

• Creating a Shopping cart so that customer can Shoppe 'n' no. of items and checkout finally with the entire shopping cart

• Customers should be able to mail the Shop about the items they would like to see in the Shop

• Secured mechanism for checking out from the Shop(Credit card verification mechanism)

• Updates to customers about the Recent Items in the Shop.

• Uploading 'Most Purchased' Items in each category of products in the Shop like Apparel, Kitchen accessories, Bath accessories, Food items etc.

2. Feasibility Analysis

Feasibility study is the process of determination of whether or not a project is worth doing. Feasibility studies are undertaken within tight time constraints and normally culminate in a written and oral feasibility report. I have taken two weeks in feasibility study with my co-developer.The contents and recommendations of this feasibility study helped us as a sound basis for deciding how to proceed the project. It helped in taking decisions such as which software to use, hardware combinations, etc.

1)Technical Feasibility 2)Economical Feasibility 3) Operational Feasibility

2.1 TECHNICAL FEASIBILITY: Technical feasibility determines whether the work for the project can be done with theexisting equipment, software technology and available personnel. Technical feasibility is concernedwith specifying equipment and software that will satisfy the user requirement. This project is feas ible on technic al remarks als o, as the propos ed sys tem is more beneficiary in terms of having a sound proof system with new technical components installed on thesystem. The proposed system can run on any machines supporting Windows and Internet services andworks on the best software and hardware that had been used while designing the system so it would befeasible in all technical terms of feasibility.

2.2 ECONOMICAL FEASIBIBLITY: Economical feasibility determines whether there are sufficient benefits in creating tomake the cost acceptable, or is the cost of the system too high. As this signifies cost-benefit analysisand s avin gs . O n the behalf of the cos t-be nefit analys is , the propos ed system is feas ible and is economical regarding its pre-assumed cost for making a system. We classified the costs of eSHOP according to the phase in which they occur. As weknow that the system development costs are usually one-time costs that will not recur after the projecthas been completed.

2.3 OPERATIONAL FEASIBILITY: It is mainly related to human organization and political aspects. The points to be considered are: •What changes will be brought with the system?

3. Software and Hardware Requirements

3.1 Software Requirements :

Operating System : Windows XP/7 or Linux
User Interface : React,BootStrap
Front End : React
Back End : Node Js,Express Js
Databse:Mongo Db
Browser : Chrome

Hardware Requirements:

Processor : Intel Pentium IV
Hard Disk : 1.44 MB FDD
RAM : 512MB or more
Monitor : 1 5”Colour monitor
Proccessor Speed : 1.7 GHZ
CD Drive : 52-X CD ROM
Keyboard : Mercury 1 10 keys
Mouse : Logtech mouse

4. System Study

4.1 DEFINITION OF THE SYSTEM:

A system is an orderly grouping of independent components linked together according to a plan to achieve a specific objective. Its main characteristics are organization, interaction, independent, integration and central objective a system does not necessarily mean to a computer system. It may be a manual system or any other names.

4.2 NEEDS OF THE SYSTEM:

Social and economic factor: a wave of social and economic changes often follows in the wake of the new technology. New opportunities may arise to improve on a production process or to do something that was not previously possible. Changes in the ways individuals are organized into groups may then be necessary, and the new groups may complete for economic resources with established units. Technological factor: people have never before in a time when the scope of scientific inquiry was so broad, so when the speed of applying the new technology accounts for many changes in the organization. High level decisions and operating processes: in response to technological, socio-economical factors, top level managers may decide to recognize operations and introduce new products. To deal with these needs, people commonly seek new modified information to support the decision. When that happens, then they obtain turn to a computer system for help the information users and data processing specialist then work together to complete a series of steps in a system study to produce output results to satisfy information needs.

4.3 SYSTEM ANALYSIS

System Analysis is a process by which we attribute process or goals to a human activity, determine how well those purpose are being achieved and specify the requirements of the various tools and techniques that are to be used within the system if the system performances are to be achieved.

5. MERN STACK

JavaScript

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

Node Js

Node.js is an open source, a system application and furthermore is an environment for servers. Nodejs is an independent development platform built on Chrome's JavaScript Runtime that we can build network applications quickly and easily. Google V8 JavaScript engine is used by Node.js to execute code. Moreover, a huge proportion of essential modules are written in JavaScript.

Applications using NodeJS:

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

NodeJS should be used when:

- Building RESTful API (JSON).

You can use Node.js in building RESTful API (JSON). They handle JSON very easily, even more than JavaScript. API servers when using Node.js usually do not have to perform heavy processing, but the number of concurrent requests is high. • Applications that demand alternative connection protocols, not just http. With TCP protocol backing, any custom protocol can be built easily.

• Real-time applications.

• Stateful websites.

Every request on the invariable procedure is handled by Node.js, therefore building caching is simpler: store it to a comprehensive variable then all requests can approach the cache. The status of one client can be 9 saved and shared with other clients and do not have to go through external memory.

Express Js

Express is a framework purely based on Node JS. Express offers strong structures to develop an efficient backend. Express support HTTP and middleware method which makes the APIs made using Express JS relatively more robust and simpler to use. Express offers many tools for us to create a better backend environment. We must also take note that famous NodeJS frameworks like JS Sails and MEAN includes Express JS as a core function. We break down the simple route into two parts as controller and routes. This makes managing the routes easier because complex functions are in a different file. Also, we can reuse the same function again and again which is increasing the code reusability.

Mongo Db

MongoDB is an open source database; it is also the leading NoSQL (*) database currently used by millions of people. It is written in one of the most popular programming languages today. In addition, MongoDB is cross-platform data that operates on the concepts of Collections and Documents, providing high performance with high availability and ease of expansion

React Js

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

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

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

Pros of ReactJS:

- Update data changes quickly.
- React is not a framework so it offloads the constraints of libraries together.
- Easy access to who understands JS.

Cons of ReactJS:

- ReactJS only serves the View tier, but the library size is on par with Angular while Angular is a complete framework.
- Incorporating ReactJS within common MVC frameworks demands reconfiguration.
- Hard to reach for beginners on website developing

MERN Stack in Website Development

Concept of Stack technology:

The technical stack is a combination of technologies / frameworks / programming languages, etc. to create a complete software. With current software, there are usually two parts: client side and server side, also known as frontend and backend. Therefore, people also split the backend stack, the frontend stack as well. We often use the first letter to name the technical stack: LAMP (Linux, Apache, MySQL, PHP), MEAN (MongoDB, Express, Angular, NodeJS).

Concept Of MERN Stack:

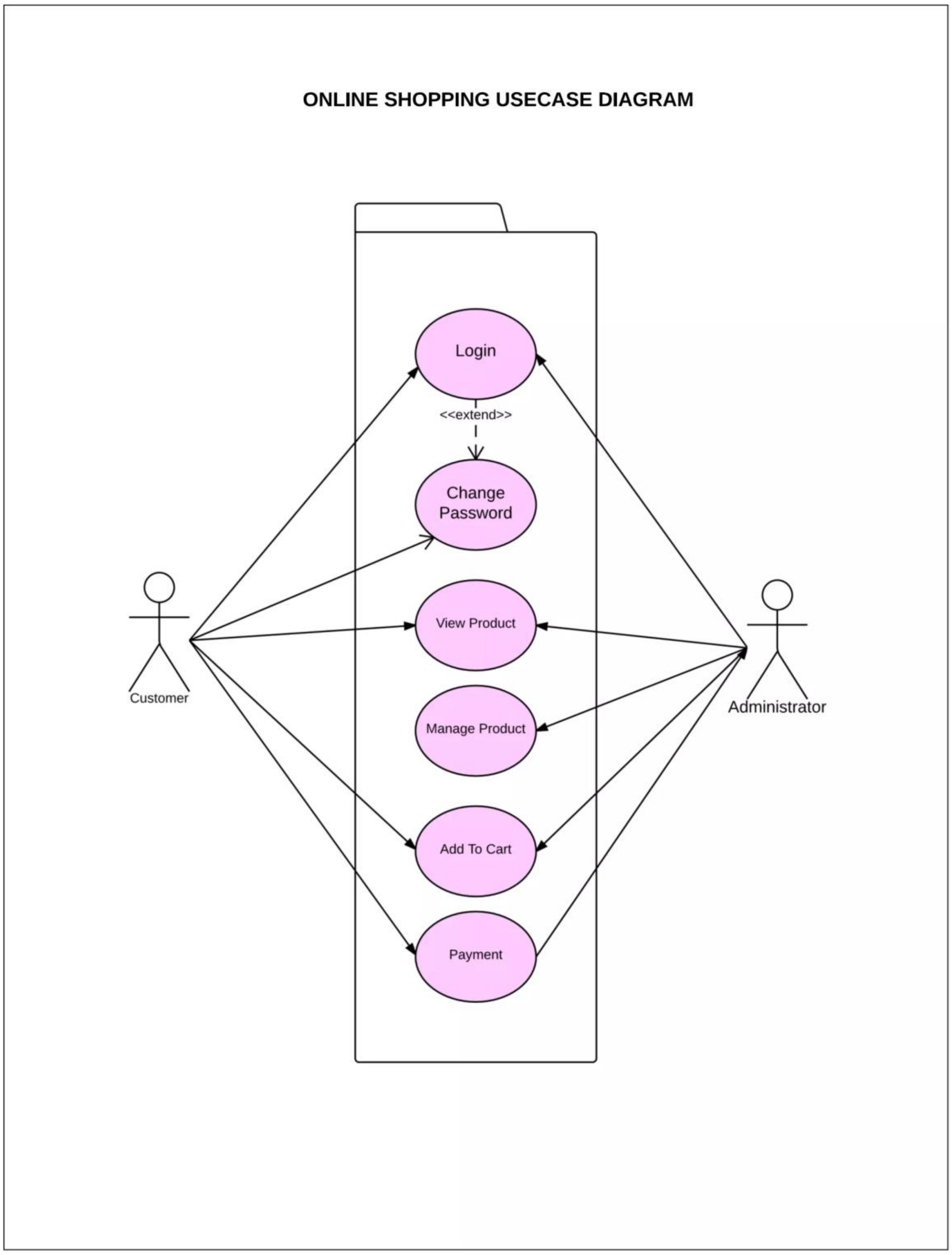
The MERN stack is a complete open source combination, all JavaScript-related technologies are also the hottest today: MongoDB, Express.js, React / React Native, NodeJS. People use the MERN stack to build Universal React App.

Highlights in MERN Stack:

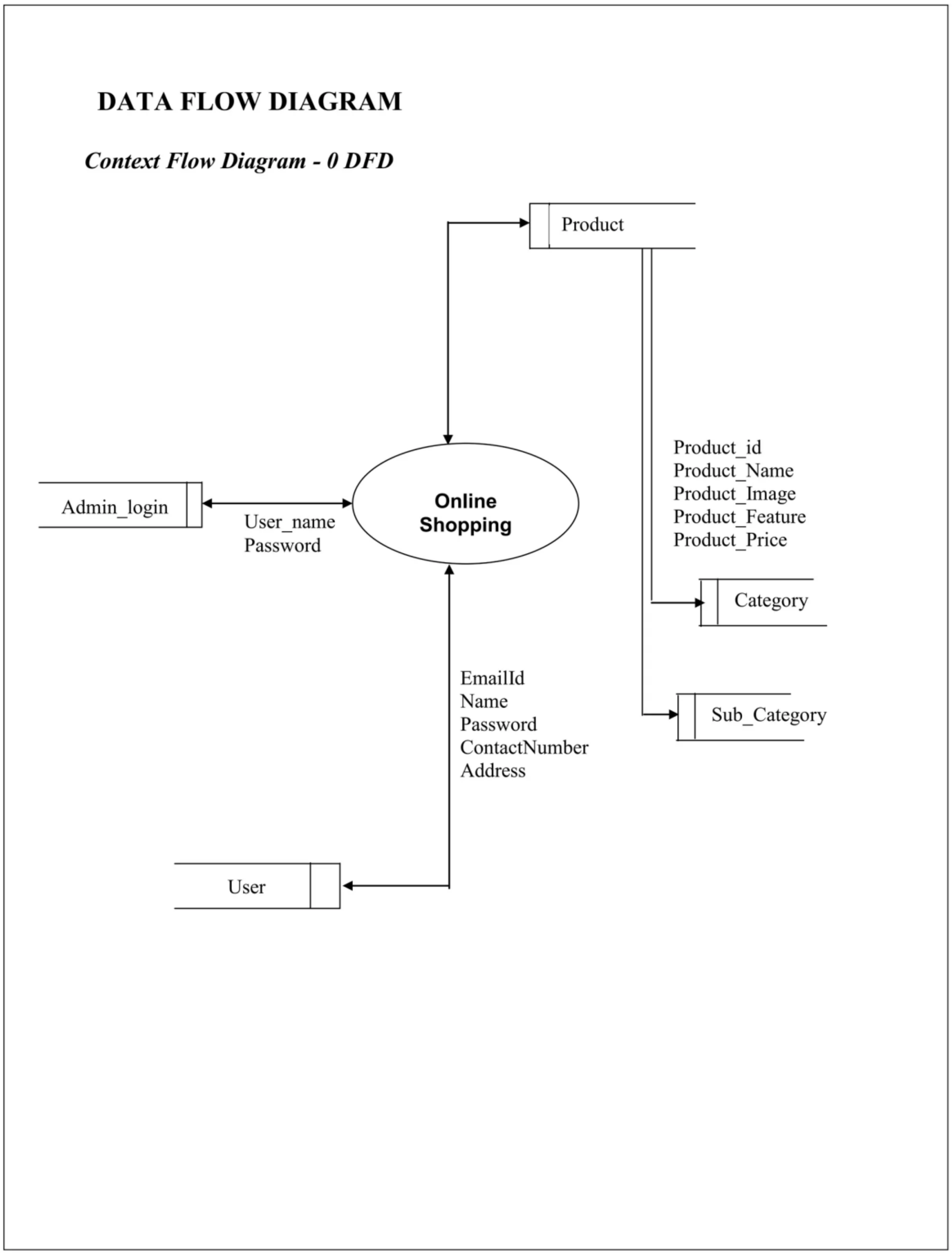
- Hot Reloading for React Components.
- The code snippets are divided by React Router.
- Multi-language support.
- Generate code support.
- Modular structure.
- Docker support.
- Can customize the MERN version for individual users

6. System Design

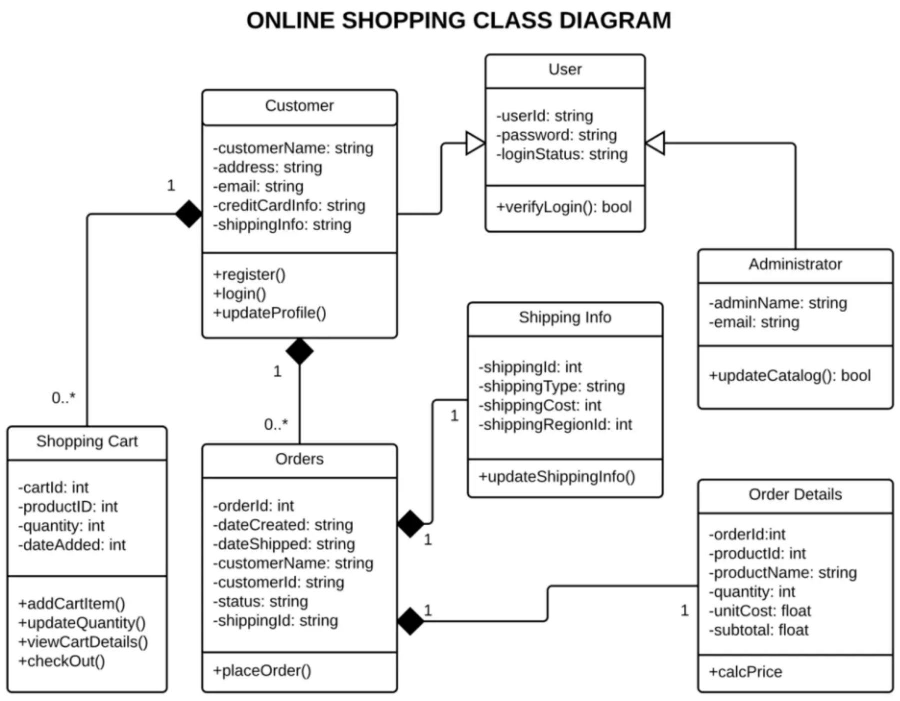
6.1 Use Case Diagram



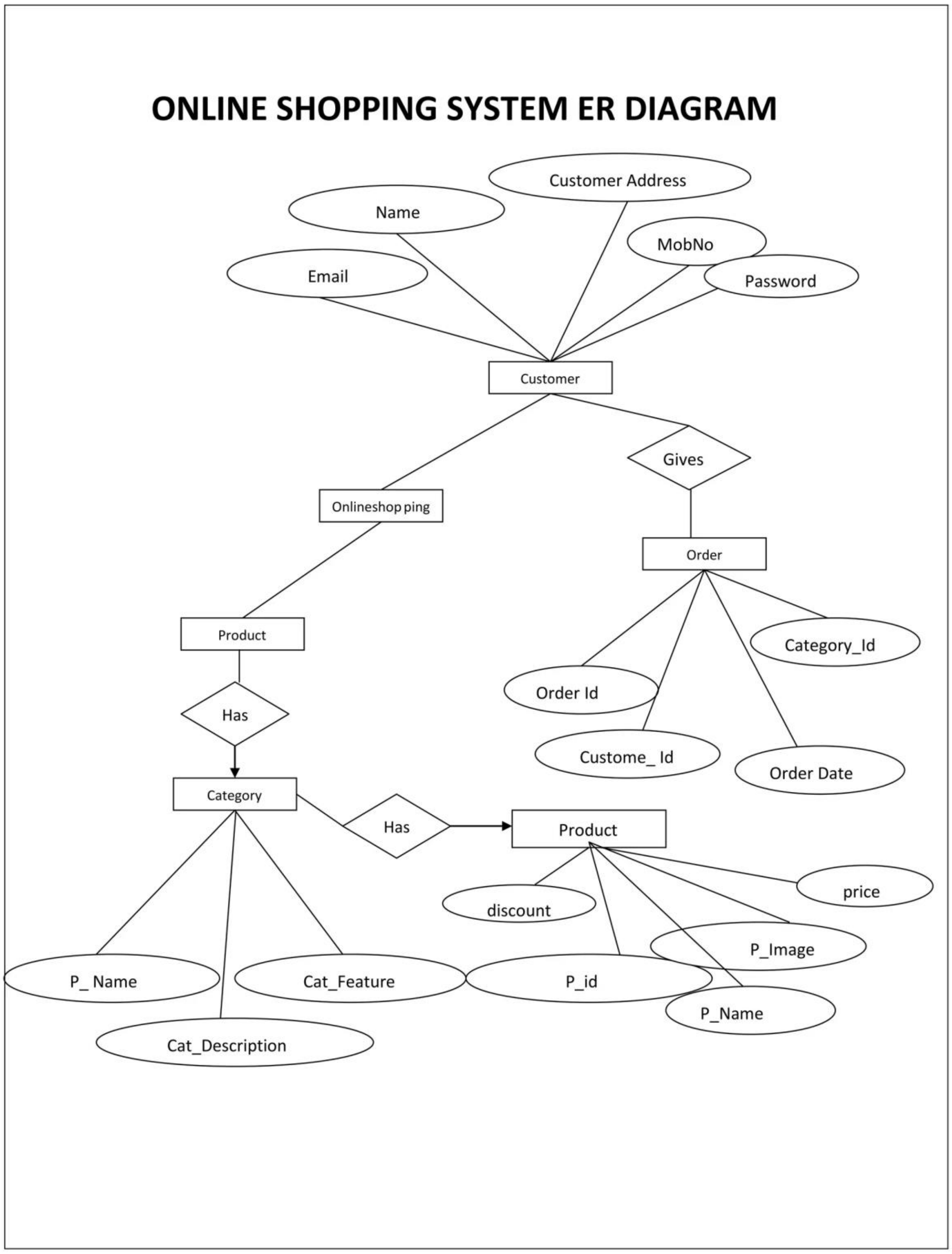
6.1 Data Flow Diagram



CLASS DIAGRAM



ER-DIAGRAM

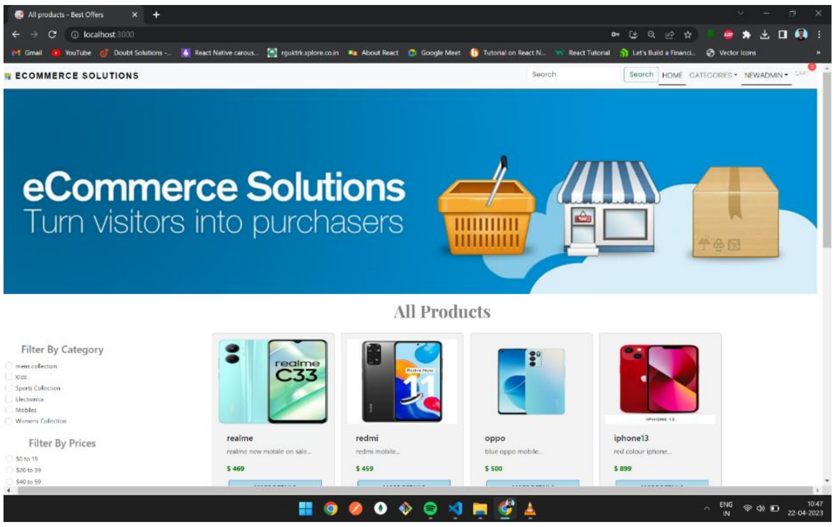

ONLINE SHOPPING SYSTEM ER DIAGRAM
Customer Address
Name
MobNo
Email
Password
Customer
Gives
Onlineshop ping
Order
Product
Category_Id
Order Id
Has
Custome_ Id
Order Date
Category
Has
Product
price
discount
P_Image
P_ Name
Cat_Feature
P_id
P_Name
Cat_Description

7. SOURCE CODE

Home Page:

```
import React, { useState, useEffect } from 'react';
import Layout from './../components/layout/Layout';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import { Checkbox, Radio } from 'antd';
import { Prices } from '../components/Prices';
import { useCart } from '../context/cart';
import toast from 'react-hot-toast';
import "../styles/Homepage.css";
const Homepage = () => {
  const navigate = useNavigate();
  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);
  //get All Category
  const getAllCategory = async () => {
    try {
      const { data } = await axios.get("/api/v1/category/get-category");
      if (data?.success) {
        setCategories(data?.category);
      }
    } catch (error) {
      console.log(error);
    }
  };
  useEffect(() => {
    getAllCategory();
    getTotal();
  }, []);
  // get products
  const getAllProducts = async () => {
    try {
      setLoading(true)
      const { data } = await axios.get(`/api/v1/product/product-list/${page}`);
      setLoading(false)
      setProducts(data.products);
    } catch (error) {
      setLoading(false)
      console.log(error)
    }
  };
  //get total count
  const getTotal = async () => {
    try {
      const { data } = await axios.get('/api/v1/product/product-count')
      setTotal(data?.total)
    } catch (error) {
      console.log(error)
    }
  };
  useEffect(() => {
    if (page === 1) return
    loadMore();
  }, [page]);
  //load more
  const loadMore = async () => {
    try {
        const { data } = await axios.get(`/api/v1/product/product-list/${page}`)
        setLoading(false)
        setProducts([...products, ...data?.products])
    } catch (error) {
        console.log(error)
        setLoading(false}

//filter by cattegory
const handleFilter = (value, id) => {
    let all = [...checked]
    if (value) {
        all.push(id)
    } else {
        all = all.filter(c => c !== id)
    }
    setChecked(all);
};
useEffect(() => {
    if (!checked.length || !radio.length) getAllProducts();
    //eslint-disable-next-line
}, [checked.length, radio.length]);

useEffect(() => {
    if (checked.length || radio.length) filterProduct();
}, [checked, radio]);
//get filtered product
const filterProduct = async () => {
    try {
        const { data } = await axios.post('/api/v1/product/product-filters', { checked, radio })
        setProducts(data?.products)
    } catch (error) {
        console.log(error)
    }
}
return (
    <Layout title={'All products - Best Offers'}>
        <img
            src="/images/e-com.jpg"
            className="banner-img"
            alt="bannerimage"
            width={"100%"}
            style={{ marginTop: 0 }}
        />
        <div className='row mt-3'>
            <div className='col-md-2'>
                <h4 className='text-center' style={{ marginTop: 100, color: "gray" }}>Filter By Category</h4>
                <div className='d-flex flex-column'>
                    {categories?.map((c) => (
                        <Checkbox key={c._id} onChange={(e) => handleFilter(e.target.checked, c._id)} style={{ marginLeft: 17 }}>
                            {c.name}
                        </Checkbox>
                    ))}
                </div>
                {/* price filter */}
                <h4 className='text-center mt-4'
                    style={{ marginTop: 100, color: "gray" }}
                >Filter By Prices</h4>
                <div className='d-flex flex-column'>
                    <Radio.Group onChange={(e) => setRadio(e.target.value)} style={{ marginBottom: 5, marginLeft: 8 }}>
                        {Prices?.map((p) => (
                            <div key={p._id}>
                                <Radio value={p.array}>{p.name}</Radio>
                            </div>
                        ))}
                    </Radio.Group>
                </div>
                <div className='d-flex flex-column' style={{ paddingLeft: 15 }}>
                    <button className='btn btn-danger' onClick={() => window.location.reload()} style={{ backgroundColor: "black", width: 300,
border: "none", borderRadius: 0, marginTop: 20, padding: 15 }}> RESET FILTERS </button>
                </div>
            </div>
            <div className='col-md-9' >
                <h1 className='text-center' style={{ color: "gray", fontFamily: "Playfair Display" }}> All Products </h1>
                <div className='d-flex flex-wrap' style={{ paddingLeft: 170 }}>
                    {products?.map((p) => (

                        <div className="card m-2" style={{ width: "18rem", backgroundColor: "rgba(128, 128, 128, 0.097)", padding: 14 }}>
                            <img
                                src={`/api/v1/product/product-photo/${p._id}`}
                                className="card-img-top"
                                alt={p.name}
                                style={{ height: 200 }}
                            />
                            <div className="card-body" style={{ padding: 20 }}>
                                <h5 className="card-title">{p.name}</h5>
                                <p className="card-text" style={{ color: "rgb(90, 89, 89)" }}>{p.description.substring(0, 30)}...
                                </p>
                                <p className="card-text" style={{ color: "green", fontWeight: "bold" }}>$ {p.price}</p>
                                <button class="btn btn-primary ms-1"
                                    onClick={() => navigate(`/product/${p.slug}`)
                                    }
                                    style={{ borderRadius: 0, width: "100%", borderTopLeftRadius: 5, borderBottomRightRadius: 5, fontSize: 14,
fontWeight: "bold", marginBottom: 5, backgroundColor: "lightblue", color: "black" }}
                                >MORE DETAILS</button>
                                <button class="btn btn-secondary ms-1"
                                    onClick={() => {
                                        setCart([...cart, p])
                                        localStorage.setItem('cart', JSON.stringify([...cart, p]))
                                        toast.success('Item added to cart')
                                    }}
                                    style={{ borderRadius: 0, width: "100%", borderTopLeftRadius: 5, borderBottomRightRadius: 5, fontSize: 14,
fontWeight: "bold", backgroundColor: "#002D62", color: "white" }}
                                >ADD TO CART</button>
                            </div>
                        </div>
                    ))}
                </div>
                <div className='m-2 p-3' >
                    {products && products.length < total && (
                        <button className='btn btn-warning'
                            style={{

                                color: "white",
                                fontweight: 30,
                                fontsize: 20,
                                marginLeft: 540
                            }}
                            onClick={(e) => {
                                e.preventDefault();
                                setPage(page + 1);
                            }}
                        >
                            {loading ? "loading..." : "Loadmore"}
                        </button>
                    )}
                </div>
            </div>
        </div>
    </Layout>
  )
}
```

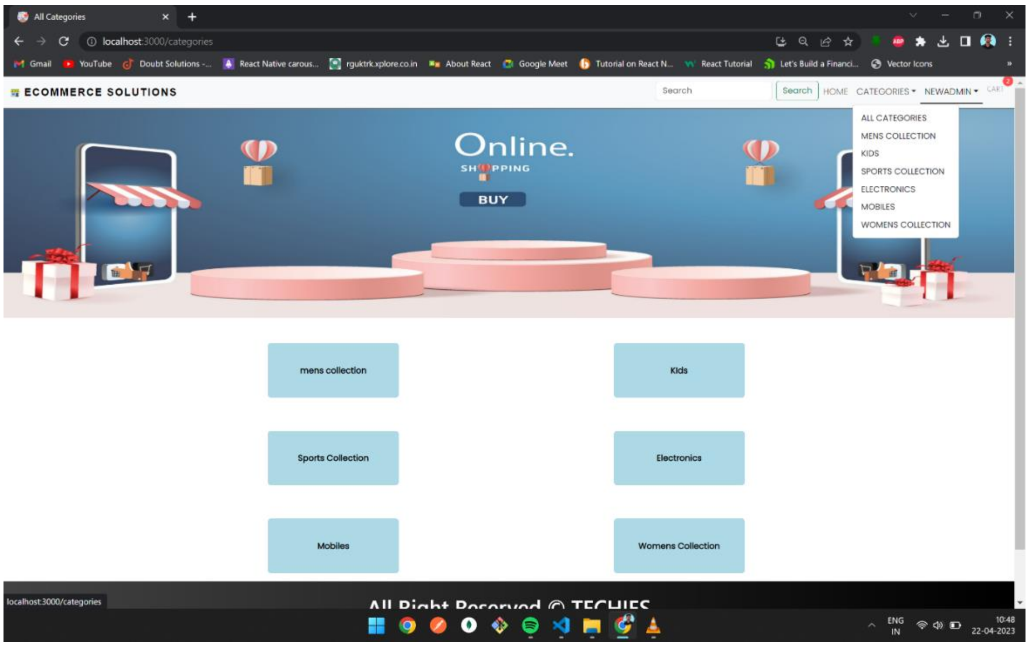
OUTPUT:



Category Page:

```
import React, { useState, useEffect } from 'react'
import { Link } from 'react-router-dom';
import useCategory from '../hooks/useCategory'
import Layout from '../components/layout/Layout'
const Categories = () => {
  const categories = useCategory()
  return (
    <Layout title={'All Categories'}>
      <img
        src="/images/shopping.jpg"
        className="banner-img"
        alt="bannerimage"
        width={"100%"}
        style={{ marginTop: 0, height: 400 }}
      />
      <div className='container' >
        <div className='row'  >
          {categories.map(c => (
            <div  className='col-md-6  mt-5  mb-3  gx-3  gy-3'  key={c._id}  style={{  paddingLeft:  200,  paddingBottom:  0, paddingTop: 0, marginBottom: 10 }} >
              <Link to={`/category/${c.slug}`} className='btn btn-primary'
                style={{ padding: 40, width: "55.3%", backgroundColor: "lightblue", border: 0, color: "black", fontFamily: "poppins", fontWeight: "bold" }}
              >{c.name}
              </Link>
            </div>
          ))}

        </div>
      </div>
    </Layout>
  )
}
export default Categories
```

OUTPUT:



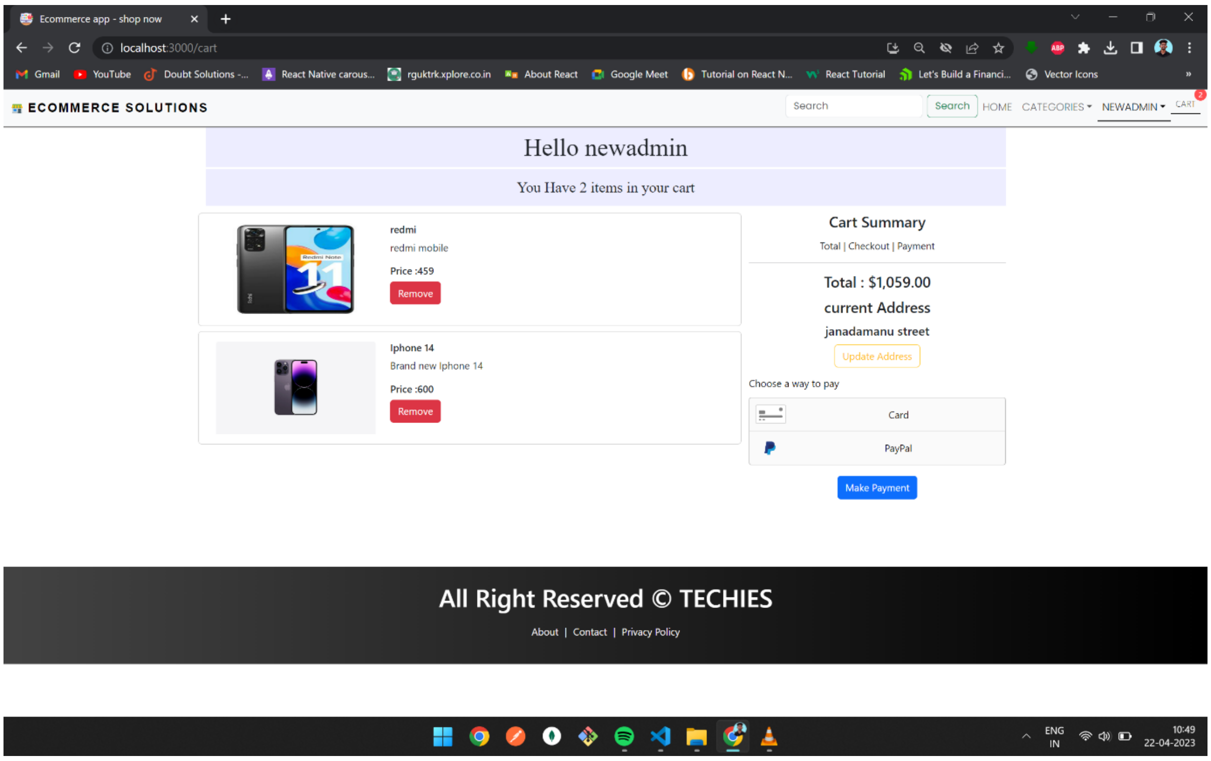
Cart & Payment Page:

```
import React, { useEffect, useState } from 'react';
import Layout from '../components/layout/Layout';
import { useCart } from '../context/cart';
import { useAuth } from '../context/Auth';
import { useNavigate } from 'react-router-dom';
import DropIn from "braintree-web-drop-in-react";
import axios from 'axios';
import toast from 'react-hot-toast';
import "../styles/CartStyles.css";
const CartPage = () => {
   const [auth, setAuth] = useAuth()
   const [cart, setCart] = useCart()
   const [clientToken, setClientToken] = useState("")
   const [instance, setInstance] = useState("")
   const [loading, setLoading] = useState(false)
   const navigate = useNavigate()
   //total price
   const totalPrice = () => {
      try {
         let total = 0
         cart?.map((item) => {
            total = total + item.price;
         });
         return total.toLocaleString('en-US', {
            style: 'currency',
            currency: 'USD',
         });
      } catch (error) {
         console.log(error)
      }
   }
```

```
  //delete item
  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));
    } catch (error) {
      console.log(error)
    }}
  //get payment gateway token
  const getToken = async () => {
    try {
      const { data } = await axios.get("/api/v1/product/braintree/token");
      setClientToken(data?.clientToken);
    } catch (error) {
      console.log(error);
    }
  };
  useEffect(() => {
    getToken();
  }, [auth?.token]);

  //handle payments
  const handlePayment = async () => {
    try {
      setLoading(true);
      const { nonce } = await instance.requestPaymentMethod();
      const { data } = await axios.post("/api/v1/product/braintree/payment", {
        nonce,
        cart,
      });
      setLoading(false);
      localStorage.removeItem("cart");
      setCart([]);
      navigate("/dashboard/user/orders");
      toast.success("Payment Completed Successfully ");
    } catch (error) {
      console.log(error);
      setLoading(false);
    }
  };
  return (
    <Layout>
      <div className='container' style={{ marginTop: 0, marginBottom: 20, }}>
        <div className='row'>
          <div className='col-md-12'>
            <h1 className='text-center bt-light p-2 mb-1' style={{ padding: 15, fontFamily: "roboto", fontWeight: 
"normal", backgroundColor: "rgba(0, 0, 255, 0.072)" }}>
              {`Hello ${auth?.token && auth?.user?.name}`}

            </h1>
            <h4 className='text-center' style={{ padding: 15, fontFamily: "roboto", fontWeight: "normal", 
backgroundColor: "rgba(0, 0, 255, 0.072)", marginTop: 0 }}>
              {cart?.length
                ? `You Have ${cart.length} items in your cart ${auth?.token ? "" : "please login to checkout"}` : "your cart 
is empty"}
```

```
          </h4>
        </div>
      </div>
      <div className='row'>
        <div className='col-md-8'>
          {
            cart?.map((p) => (
              <div className='row mb-2 p-3 card flex-row'>
                <div className='col-md-4'>
                  <img
                    src={`/api/v1/product/product-photo/${p._id}`}
                    className="card-img-top"
                    alt={p.name}
                    width="100px"
                    height={'150px'}
                  />
                </div>
                <div className='col-md-8'>
                  <h6>{p.name}</h6>
                  <p>{p.description.substring(0, 30)}</p>
                  <h6>Price :{p.price}</h6>
                  <button className='btn btn-danger'
                    onClick={() => removeCartItem(p._id)}
                  >Remove</button>
                </div>
              </div>
            ))}
        </div>
        <div className='col-md-4 text-center'>
          <h4>Cart Summary</h4>
          <p>Total | Checkout | Payment</p>
          <hr />
          <h4>Total : {totalPrice()} </h4>
          {auth?.user?.address ? (
            <>
              <div className='mb-3'>
                <h4>current Address</h4>
                <h5>{auth?.user?.address}</h5>
                <button className='btn btn-outline-warning'
                  onClick={() => navigate('/dashboard/user/profile')}
                >Update Address</button>
              </div>
            </>
          ) : (
            <div className='mn-3'>
              {auth?.token ? (
                <button className='btn btn-outline-warning'
                  onClick={() => navigate('/dashboard/user/profile')}>Update Address</button>
              ) : (
                <button className='btn btn-outline-warning'
                  onClick={() => navigate('/login', {
                    state: "/cart"
                  })}>please login to checkout</button>
              )}
            </div>
          )}
          <div className="mt-2">
            {!clientToken || !auth?.token || !cart?.length ? (
                ""
              ) : (
                <>
                  <DropIn
                    options={{
                      authorization: clientToken,
                      paypal: {
                        flow: "vault",
                      },
                    }}
                    onInstance={(instance) => setInstance(instance)}
                  />

                  <button
                    className="btn btn-primary"
                    onClick={handlePayment}
                    disabled={loading || !instance || !auth?.user?.address}
                  >
                    {loading ? "Processing ...." : "Make Payment"}
                  </button>
                </>
              )}
            </div>
          </div>

        </div>
      </div>
    </Layout>
  )
}

export default CartPage;
```

OUTPUT:



Ecommerce app - shop now

localhost:3000/cart

ECOMMERCE SOLUTIONS

Hello newadmin

You Have 2 items in your cart

redmi
redmi mobile
Price :459

Iphone 14
Brand new Iphone 14
Price :600

Cart Summary
Total | Checkout | Payment
Total : $1,059.00
current Address
janadamanu street

Choose a way to pay

All Right Reserved © TECHIES

About | Contact | Privacy Policy

BACKEND

Product Controllers.js

```
import productModel from "../models/productModel.js";
import categoryModel from "../models/categoryModel.js";
import orderModel from "../models/orderModel.js";
import fs from 'fs';
import slugify from "slugify";
import braintree from "braintree";
import dotenv from 'dotenv';

dotenv.config();


//payment gateway
var gateway = new braintree.BraintreeGateway({
  environment: braintree.Environment.Sandbox,
  merchantId: process.env.BRAINTREE_MARCHANT_ID,
  publicKey: process.env.BRAINTREE_PUBLIC_KEY,
  privateKey: process.env.BRAINTREE_PRIVATE_KEY,
});


export const createProductController = async (req, res) => {
  try {
    const { name, description, price, category, quantity, shippingg } = req.fields;
    const { photo } = req.files;
    //Validation
    switch (true) {
      case !name:
        return res.status(500).send({ error: "Name is Required" });
      case !description:
        return res.status(500).send({ error: "Description is Requied" });

      case !price:
        return res.status(500).send({ error: "Price  is Requied" });
      case !category:
        return res.status(500).send({ error: "Category is Requied" });

      case !quantity:
        return res.status(500).send({ error: "Quantityy is Requied" });
      case photo && photo.size > 1000000:
        return res.status(500).send({ error: "Photo is Required and should be Leass than 1mb" })


    }
    const products = new productModel({ ...req.fields, slug: slugify(name) });
    if (photo) {
      products.photo.data = fs.readFileSync(photo.path);
      products.photo.contentType = photo.type;
    }
    await products.save();
    res.status(201).send({
      success: true,
      message: "Products Created Successfully",
      products,
        })

    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            error,
            message: "Error  while creating product",

        });
    }
};

//get All Products
export const getProductController = async (req, res) => {
    try {
        const products = await productModel.find({}).populate("category").select("-photp").limit(12).sort({ createdAt: -1 });
        res.status(200).send({
            success: true,
            countTotal: products.length,
            message: "All Products",
            products,
        });
    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            message: "Error in getting Products",
            error: error.message,


        });
    }
};


//get single Product
export const getSingleProductController = async (req, res) => {
    try {
        const product = await productModel.findOne({ slug: req.params.slug }).populate("category").select("-photo");
        res.status(200).send({
            success: true,
            message: "Single Product Fetched",
            product,
        });
    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            message: "Error in While  getting  SingleProducts",
            error,


        });
    }
};

//get Photo
```

```
export const productPhotoController = async (req, res) => {
  try {
    const product = await productModel.findById(req.params.pid).select("photo");
    if (product.photo.data) {
      res.set("Content-type", product.photo.contentType);
      return res.status(200).send(product.photo.data);

    }
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error  While  getting  Photo",
      error,


    });
  }
};

//delete Controller

export const deleteProductController = async (req, res) => {
  try {
    await productModel.findByIdAndDelete(req.params.pid).select("-photo");
    return res.status(200).send({
      success: true,
      message: "Product Deleted Successfully",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error  While  deleting  Photo",
      error,


    });
  }
};


//update Products

export const updateProductController = async (req, res) => {
  try {
    const { name, description, price, category, quantity, shippingg } = req.fields;
    const { photo } = req.files;
    //Validation
    switch (true) {
      case !name:
        return res.status(500).send({ error: "Name is Required" });
      case !description:
        return res.status(500).send({ error: "Description is Requied" });

      case !price:
        return res.status(500).send({ error: "Price  is Requied" });
      case !category:
```

```
                return res.status(500).send({ error: "Category is Requied" });

            case !quantity:
                return res.status(500).send({ error: "Quantityy is Requied" });
            case photo && photo.size > 1000000:
                return res.status(500).send({ error: "Photo is Required and should be Leass than 1mb" })


        }
        const products = await productModel.findByIdAndUpdate(req.params.pid, { ...req.fields, slug: slugify(name) }, { new:
true });
        if (photo) {
            products.photo.data = fs.readFileSync(photo.path);
            products.photo.contentType = photo.type;
        }
        await products.save();
        res.status(201).send({
            success: true,
            message: "Products Updated Successfully",
            products,
        });

    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            error,
            message: "Error  while  Updating  products",

        });
    }
};
//filters
export const productFiltersController = async (req, res) => {
    try {
        const { checked, radio } = req.body
        let args = {}
        if (checked.length > 0) args.category = checked
        if (radio.length) args.price = { $gte: radio[0], $lte: radio[1] }
        const products = await productModel.find(args)
        res.status(200).send({
            success: true,
            products,
        });
    } catch (error) {
        console.log(error);
        res.status(400).send({
            success: false,
            message: 'Error while Filtering Products',
            error
        });
    }
};
///product count
export const productCountController = async (req, res) => {
    try {
        const total = await productModel.find({}).estimatedDocumentCount()
        res.status(200).send({
            success: true,
            total,
        });
    } catch (error) {
        console.log(error)
        res.status(400).send({
            message: 'Error in product count',
            error,
            success: false
        });
    }
};

//Product list base on page
export const productListController = async (req, res) => {
    try {
        const perPage = 6
        const page = req.params.page ? req.params.page : 1
        const products = await productModel
            .find({}).select("-photo")
            .skip((page - 1) * perPage)
            .limit(perPage)
            .sort({ createdAt: -1 });
        res.status(200).send({
            success: true,
            products,
        });
    } catch (error) {
        console.log(error)
        res.status(400).send({
            success: false,
            message: 'error in per page ctrl',
            error
        })
    }
}

//search product
export const searchProductController = async (req, res) => {
    try {
        const { keyword } = req.params
        const results = await productModel
            .find({
                $or: [
                    { name: { $regex: keyword, $options: "i" } },
                    { description: { $regex: keyword, $options: "i" } }
                ]
            }).select("-photo");
        res.json(results);
    } catch (error) {
        console.log(error)
        res.status(400).send({
            success: false,
            message: `Error In search product API`,
            error,
        });
    }
};
```

```
//similar products
export const relatedProductController = async (req, res) => {
   try {
      const { pid, cid } = req.params
      const products = await productModel.find({
         category: cid,
         _id: { $ne: pid }
      }).select("-photo").limit(3).populate("category")
      res.status(200).send({
         success: true,
         products,
      });
   } catch (error) {
      console.log(error)
      res.status(400).send({
         success: false,
         message: 'error while getting related data',
         error
      })
   }
}


//get product by category
export const productCategoryController = async (req, res) => {
   try {
      const category = await categoryModel.findOne({ slug: req.params.slug });
      const products = await productModel.find({ category }).populate("category");
      res.status(200).send({
         success: true,
         category,
         products,
      });

   } catch (error) {
      console.log(error)
      res.status(400).send({
         success: false,
         error,
         message: 'Error while Getting Products'
      })
   }
}


//payment gateway api
//token
export const braintreeTokenController = async (req, res) => {
   try {
      gateway.clientToken.generate({}, function (err, response) {
         if (err) {
            res.status(500).send(err)
         } else {
            res.send(response);
         }
      })
   } catch (error) {
        console.log(error)
    }
}

//payment
export const braintreePaymentController = async (req, res) => {
    try {
        const { cart, nonce } = req.body
        let total = 0
        cart.map((i) => {
            total += i
        });
        let newTransaction = gateway.transaction.sale({
            amount: total,
            paymentMethodNonce: nonce,
            options: {
                submitForSettlement: true
            }
        },
            function (error, result) {
                if (result) {
                    const order = new orderModel({
                        products: cart,
                        payment: result,
                        buyer: req.user._id
                    }).save()
                    res.json({ ok: true })
                } else {
                    res.status(500).send(error)
                }
            }
        )
    } catch (error) {
        console.log(error)
    }
}
```

CategoryController.js

```
import categoryModel from "../models/categoryModel.js";
import slugify from "slugify";
export const createCategoryController = async (req, res) => {
    try {
        const { name } = req.body
        if (!name) {
            return res.status(401).send({ message: 'Name is required' })
        }
        const existingCategory = await categoryModel.findOne({ name })
        if (existingCategory) {
            return res.status(200).send({
                success: true,
                message: 'Category Already exists'
            })
        }
        const category = await new categoryModel({ name, slug: slugify(name) }).save()
        res.status(201).send({
            success: true,
            message: 'new category created',
            category,
        })
    } catch (eroor) {
        console.log(error)
        res.status(500).send({
            error,
            message: 'Error in Category'
        })}
};
//Update Category Controller
export const updateCategoryController = async (req, res) => {
    try {
        const { name } = req.body;
        const { id } = req.params;
        const category = await categoryModel.findByIdAndUpdate(
            id,
            { name, slug: slugify(name) },
            { new: true }
        );
        res.status(200).send({
            success: true,
            message: "Category Updated Successfully",
            category,
        });
    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            error,
            message: "Error Whiile updating category",
        })
    }
};
//get all Category
export const categoryControlller = async (req, res) => {
    try {
        const category = await categoryModel.find({});
        res.status(200).send({
            success: true,
            message: "All Categories List",
            category,
        });
    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            error,
            message: "Error While getting All Categories"
        });
    }
}
//Single Category
export const singleCategoryController = async (req, res) => {
    try {
        const category = await categoryModel.findOne({ slug: req.params.slug });
        res.status(200).send({
            success: true,
            message: "Get Single Category Successfully",
      category,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While Getting Single Category"
    });
  }
};
//delete Catgory
export const deleteCategoryController = async (req, res) => {
  try {
    const { id } = req.params;
    await categoryModel.findByIdAndDelete(id);
    res.status(200).send({
      success: true,
      message: "Category Deleted Successfully",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While deleting  Category"
    });
  }
}
```

AuthController.js

```
import userModel from "../models/usermodel.js";
import orderModel from "../models/orderModel.js";
import { comparePassword, hashPassword } from "../helper/authHelper.js";
import JWT from "jsonwebtoken";
import { compare } from "bcrypt";
export const registerController = async (req, res) => {
  try {
    const { name, email, password, phone, address, answer } = req.body
    //validation
    if (!name) {
      return res.send({ message: 'name is required' });
    }
    if (!email) {
      return res.send({ message: 'email is required' });
    }
    if (!password) {
      return res.send({ message: 'password is required' });
    }
    if (!phone) {
      return res.send({ message: 'phone is required' });
    }
    if (!address) {
      return res.send({ message: 'address is required' });
    }
    if (!answer) {
      return res.send({ message: 'answer is required' });
    }
    //check user
```

```
        const existingUser = await userModel.findOne({ email })
        //existing user
        if (existingUser) {
            return res.status(200).send({
                success: false,
                message: "Already register please login",
            })
        }
        //register user
        const hashedPassword = await hashPassword(password);
        //save
        const user = await new userModel({ name, email, phone, address, password: hashedPassword, answer }).save()

        res.status(201).send({
            success: true,
            message: "User Register successfully",
            user,
        })

    } catch (error) {
        console.log(error);
        res.status(500).send({
            success: false,
            message: "error in registration",
            error
        })
    }
};
//POST LOGIN
export const loginController = async (req, res) => {
    try {
        const { email, password } = req.body
        //validation
        if (!email || !password) {
            return res.status(404).send({
                success: false,
                message: "Invalid email or password"
            })
        }
        //check user
        const user = await userModel.findOne({ email })
        if (!user) {
            return res.status(404).send({
                success: false,
                message: 'Email not Registered'
            })
        }
        const match = await comparePassword(password, user.password)
        if (!match) {
            return res.status(200).send({
                success: false,
                message: "invalid password"
            })
        }
        //token
        const token = await JWT.sign({ _id: user._id }, process.env.JWT_SECRET, {
            expiresIn: "7d",
        });
        res.status(200).send({
            success: true,
            message: 'login successful',
            user: {
                name: user.name,
                email: user.email,
                phone: user.phone,
                address: user.address,
                role: user.role,
            },
            token,
        })
    } catch (error) {
        console.log(error)
        res.status(500).send({
            success: false,
            message: "error in login",
            error
        });
    }
};
//forgotPasswordController
export const forgotPasswordController = async (req, res) => {
    try {
        const { email, answer, newPassword } = req.body
        if (!email) {
            res.status(400).send({ message: 'Email is required' })
        }
        if (!answer) {
            res.status(400).send({ message: 'answer is required' })
        }
        if (!newPassword) {
            res.status(400).send({ message: 'New Password  is required' })
        }
        //check User
        const user = await userModel.findOne({ email, answer })
        //validation
        if (!user) {
            return res.status(404).send({
                success: false,
                message: 'Wrong Email or Answer'
            });
        }
        const hashed = await hashPassword(newPassword)
        await userModel.findByIdAndUpdate(user._id, { password: hashed })
        res.status(200).send({
            success: true,
            message: "Password Reset SuccessfUlly",
        });
    } catch (error) {
        console.log(error)
        res.status(500).send({
            success: false,
            message: 'Something went wrong',
            error
        });
    }
};
```

```
//test controller
export const testController = (req, res) => {
  try {
    res.send('protected route');
  } catch (error) {
    console.log(error);
    res.send({ error });
  }
//update profile
export const updateProfileController = async (req, res) => {
  try {
    const { name, email, password, address, phone } = req.body
    const user = await userModel.findById(req.user._id)
    //password
    if (password && password.length < 6) {
      return res.json({ error: `password is required and 6 character long` })
    }
    const hashedPassword = password ? await hashPassword(password) : undefined
    const updatedUser = await userModel.findByIdAndUpdate(req.user._id, {
      name: name || user.name,
      password: hashedPassword || user.password,
      phone: phone || user.phone,
      address: address || user.address,
    }, { new: true })
    res.status(200).send({
      success: true,
      message: 'profile updated successfully',
      updatedUser
    })
  } catch (error) {
    console.log(error)
    res.status(400).send({
      success: false,
      message: 'Error While Update Profile',
      error
    })
  }
}
//orders
export const getOrderController = async (req, res) => {
  try {
    const orders = await orderModel.find({ buyer: req.user._id }).populate("products", "-photo").populate("buyer", "name")
    res.json(orders);
  } catch (error) {
    console.log(error)
    res.status(500).send({
      success: false,
      message: 'Error While Getting Orders',
      error
    })
  }
}
//orders
export const getAllOrderController = async (req, res) => {
  try {
    const orders = await orderModel.find({}).populate("products", "-photo").populate("buyer", "name").sort({ createdAt: "-
1" })
```

```
        res.json(orders);
    } catch (error) {
        console.log(error)
        res.status(500).send({
            success: false,
            message: 'Error While Getting Orders',
            error
        })
    }
}
//order status
export const orderStatusController = async (req, res) => {
    try {
        const { orderId } = req.params;
        const { status } = req.body;
        const orders = await orderModel.findByIdAndUpdate(orderId, { status }, { new: true })
        res.json(orders)
    } catch (error) {
        console.log(error)
        res.status(500).send({
            success: false,
            message: "Error While updating Order",
            error
        })
    }
}
```

AuthMiddleWare.js

```
import JWT from 'jsonwebtoken';
import userModel from '../models/usermodel.js';

//protected Routes token base
export const requireSignIn = async (req, res, next) => {
    try {
        const decode = JWT.verify(
            req.headers.authorization,
            process.env.JWT_SECRET
        );
        req.user = decode;
        next();
    } catch (error) {
        console.log(error);
    }
};
//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).send({
                success: false,
                message: "UnAuthorized Access",
            });
        } else {
            next();
        }
    } catch (error) {
        console.log(error);
        res.status(401).send({
          success: false,
          error,
          message: "error in admin middleware",
        });
    }
};
```

ProductModel.js

```
import mongoose from "mongoose";
const productSchema = new mongoose.Schema(
    {
      name: {
        type: String,
        required: true,
      },
      slug: {
        type: String,
        required: true,
      },
      description: {
        type: String,
        required: true,
      },
      price: {
        type: Number,
        required: true,
      },
      category: {
        type: mongoose.ObjectId,
        ref: "Category",
        required: true,
      },
      quantity: {
        type: Number,
        required: true,
      },
      photo: {
        data: Buffer,
        contentType: String,
      },
      shipping: {
        type: Boolean,
      },
    },
    { timestamps: true }
);
export default mongoose.model("Products", productSchema);
```

CategoryModel.js

```
import mongoose from "mongoose";
const categorySchema = new mongoose.Schema({
    name: {
      type: String,
      //required: true,
      //unique: true,
    },
    slug: {
      type: String,
      lowercase: true,
    },
});

export default mongoose.model("Category", categorySchema);
```

OrderModel.js

```
import mongoose from "mongoose";
const orderSchema = new mongoose.Schema({
  products: [
    {
      type: mongoose.ObjectId,
      ref: "Products",},],
  payment: {},
  buyer: {
    type: mongoose.ObjectId,
    ref: 'users'},
  status: {
    type: String,
    default: 'Not Process',
    enum: ["Not Process", "Processing", "Shipped", "Delivered", "cancel"]},},
  { timestamps: true });
export default mongoose.model("Order", orderSchema);
```

ProductRoute.js

```
import express from "express";
import { createProductController, getProductController, productPhotoController, deleteProductController, updateProductController, productFiltersController, productCountController, productListController, searchProductController, relatedProductController, productCategoryController, braintreeTokenController, braintreePaymentController } from "../controller/productController.js";
import { isAdmin, requireSignIn } from './../middleware/authMiddleware.js';
import ExpressFormidable from "express-formidable";
import { getSingleProductController } from './../controller/productController.js';
const router = express.Router()
//routes
router.post('/create-product', requireSignIn, isAdmin, ExpressFormidable(), createProductController);
//routes
router.put('/update-product/:pid', requireSignIn, isAdmin, ExpressFormidable(), updateProductController);
//get all products
router.get("/get-product", getProductController);
//get Single Product
router.get("/get-product/:slug", getSingleProductController);
//getbphoto
router.get("/product-photo/:pid", productPhotoController);
//delete product
router.delete("/delete-product/:pid", deleteProductController);
//filter product
router.post('/product-filters', productFiltersController)
//product count
router.get('/product-count', productCountController)
//product per page
router.get('/product-list/:page', productListController)
//search product
```

```
router.get('/search/:keyword', searchProductController);
//similar product
router.get('/related-product/:pid/:cid', relatedProductController)
//category wise product
router.get('/product-category/:slug', productCategoryController)
//payment routes
//token
router.get('/braintree/token', braintreeTokenController)
//payments
router.post('/braintree/payment', requireSignIn, braintreePaymentController)
export default router;
```

CategoryRoute.js

```
import express from "express";
import { createProductController, getProductController, productPhotoController, deleteProductController, updateProductController, productFiltersController, productCountController, productListController, searchProductController, relatedProductController, productCategoryController, braintreeTokenController, braintreePaymentController } from "../controller/productController.js";
import { isAdmin, requireSignIn } from './../middleware/authMiddleware.js';
import ExpressFormidable from "express-formidable";
import { getSingleProductController } from './../controller/productController.js';
const router = express.Router()
//routes
router.post('/create-product', requireSignIn, isAdmin, ExpressFormidable(), createProductController);
//routes
router.put('/update-product/:pid', requireSignIn, isAdmin, ExpressFormidable(), updateProductController);
//get all products
router.get("/get-product", getProductController);
//get Single Product
router.get("/get-product/:slug", getSingleProductController);
//getbphoto
router.get("/product-photo/:pid", productPhotoController);
//delete product
router.delete("/delete-product/:pid", deleteProductController);
//filter product
router.post('/product-filters', productFiltersController)
//product count
router.get('/product-count', productCountController)
//product per page
router.get('/product-list/:page', productListController)
//search product
router.get('/search/:keyword', searchProductController);
//similar product
router.get('/related-product/:pid/:cid', relatedProductController)
//category wise product
router.get('/product-category/:slug', productCategoryController)
//payment routes
//token
router.get('/braintree/token', braintreeTokenController)
//payments
router.post('/braintree/payment', requireSignIn, braintreePaymentController)
export default router;
```

Server.js

```
import express from 'express';
import colors from 'colors';
import dotenv from 'dotenv';
import morgan from 'morgan';
import connectDB from './config/db.js';
import authRoutes from './routes/authRoute.js';
import cors from "cors";
import categoryRoutes from './routes/categoryRoutes.js'
import productRoutes from './routes/productRoutes.js'
//configure env
dotenv.config();
//database config
connectDB();
//rest object
const app = express();
//middleware
app.use(cors());
app.use(express.json());
app.use(morgan('dev'));
//routes
app.use('/api/v1/auth', authRoutes);
app.use('/api/v1/category', categoryRoutes);
app.use("/api/v1/product", productRoutes);
//rest api
app.get('/', (req, res) => {
  res.send("<h1>welcome to e commerce app</h1>")})
//port
const PORT = process.env.PORT || 8080;
//run listen
app.listen(PORT, () => {
  console.log(`server Running on ${process.env.DEV_MODE} mode on port ${PORT}`.bgCyan.white);
})
```

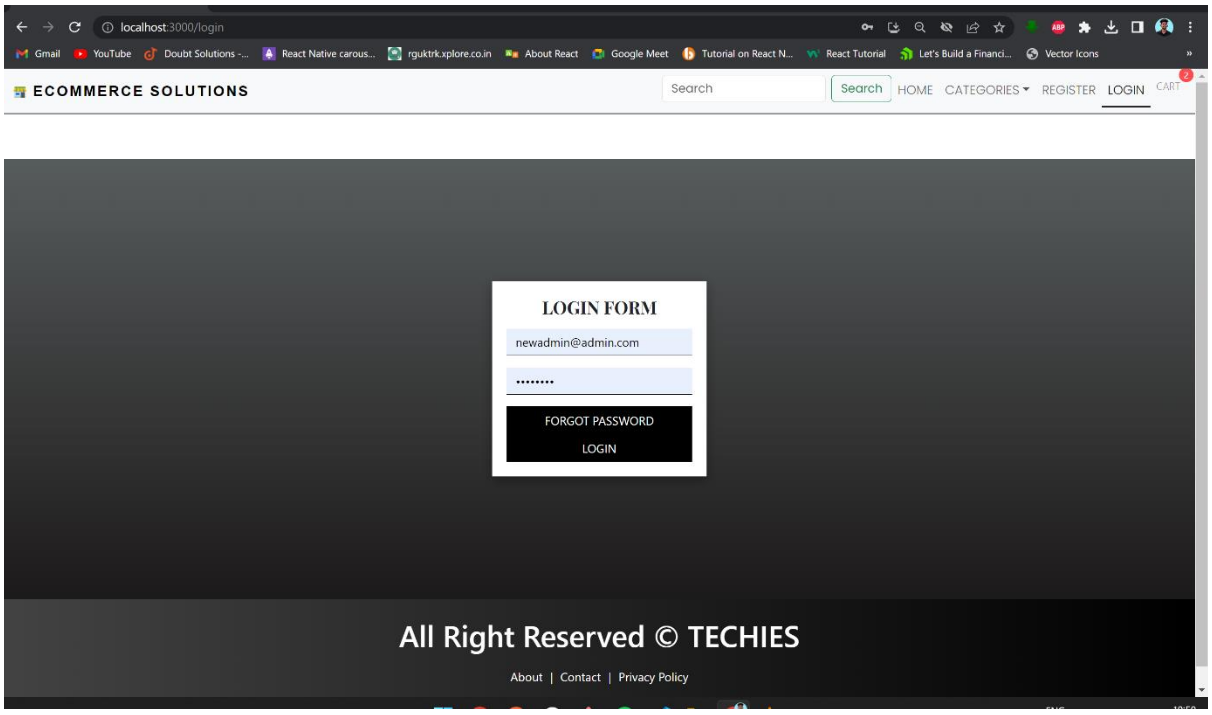
App.js

```
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 { Toast } from 'react-hot-toast';
import Login from './pages/Auth/Login';
import Dashboard from './pages/user/Dashboard';
import PrivateRoute from './components/Routes/Private';
import ForgotPassword from './pages/Auth/ForgotPassword';
import AdminRoute from './components/Routes/AdminRoute';
import { AdminDashboard } from './pages/Admin/AdminDashboard';
import CreateCategory from './pages/Admin/CreateCategory';
import CreateProduct from './pages/Admin/CreateProduct';
import Users 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 Categories from './pages/Categories';
import CategoryProduct from './pages/CategoryProduct';
import CartPage from './pages/CartPage';
import AdminOrders from './pages/Admin/AdminOrders';
function App() {
 return (
   <>
     <Routes>
       <Route path='/' element={<Homepage />} />
       <Route path='/product/:slug' element={<ProductDetails />} />
       <Route path='/categories' element={<Categories />} />
       <Route path='/cart' element={<CartPage />} />
       <Route path='/category/:slug' element={<CategoryProduct />} />
       <Route path='/search' element={<Search />} />
       <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-product" element={<CreateProduct />} />
         <Route path="admin/product/:slug" element={<UpdateProduct />} />
         <Route path="admin/products" element={<Products />} />
         <Route path="admin/users" element={<Users />} />
         <Route path="admin/orders" element={<AdminOrders />} />
       </Route>
       <Route path='/register' element={<Register />} />
       <Route path='/forgot-password' element={<ForgotPassword />} />
       <Route path='/login' element={<Login />} />
       <Route path='/about' element={<About />} />
       <Route path='/contact' element={<Contact />} />
       <Route path='/policy' element={<Policy />} />
       <Route path='*' element={<Pagenotfound />} />
     </Routes>
   </>);}
export default App;
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
import { AuthProvider } from './context/Auth';
import { SearchProvider } from './context/search';
import { CartProvider } from './context/cart';
import "antd/dist/reset.css";
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <AuthProvider>
   <SearchProvider>
     <CartProvider>
       <BrowserRouter>
         <App />
       </BrowserRouter>
     </CartProvider>
   </SearchProvider>
 </AuthProvider>
);
reportWebVitals();
```

Login Form:



LOGIN FORM

newadmin@admin.com

••••••••

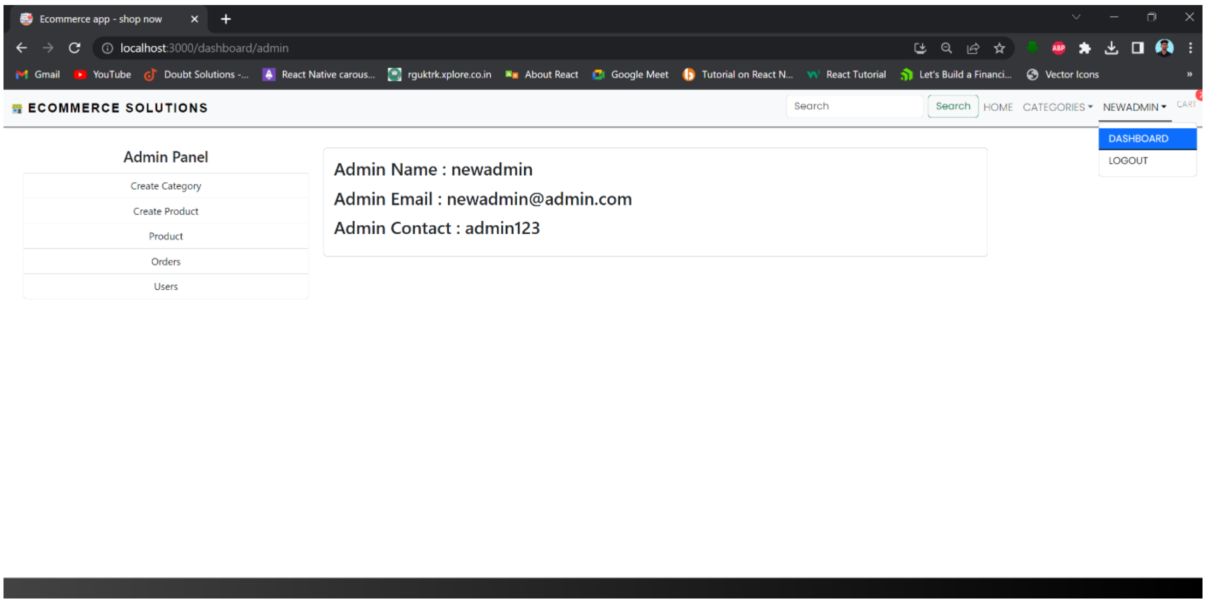
FORGOT PASSWORD

LOGIN

All Right Reserved © TECHIES

About | Contact | Privacy Policy

Admin Dashboard:

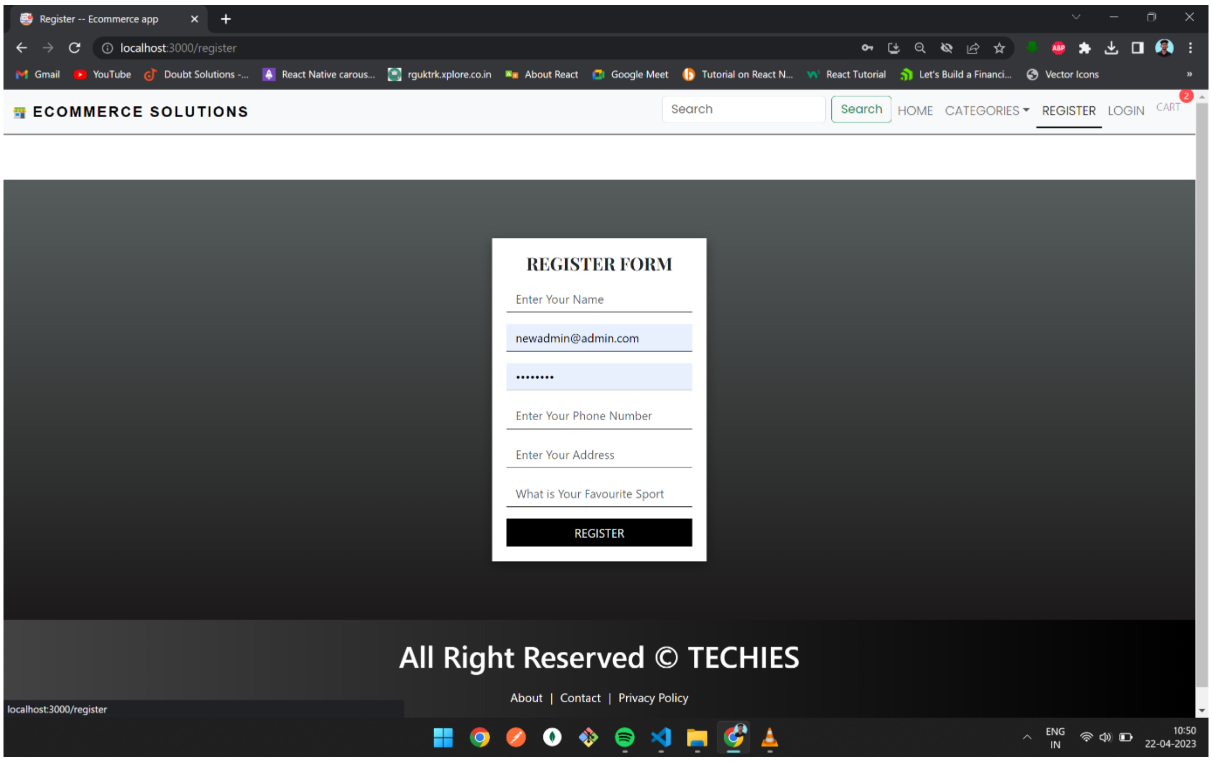


Admin Name : newadmin

Admin Email : newadmin@admin.com

Admin Contact : admin123

Register:


ECOMMERCE SOLUTIONS

REGISTER FORM

Enter Your Name

newadmin@admin.com

••••••••

Enter Your Phone Number

Enter Your Address

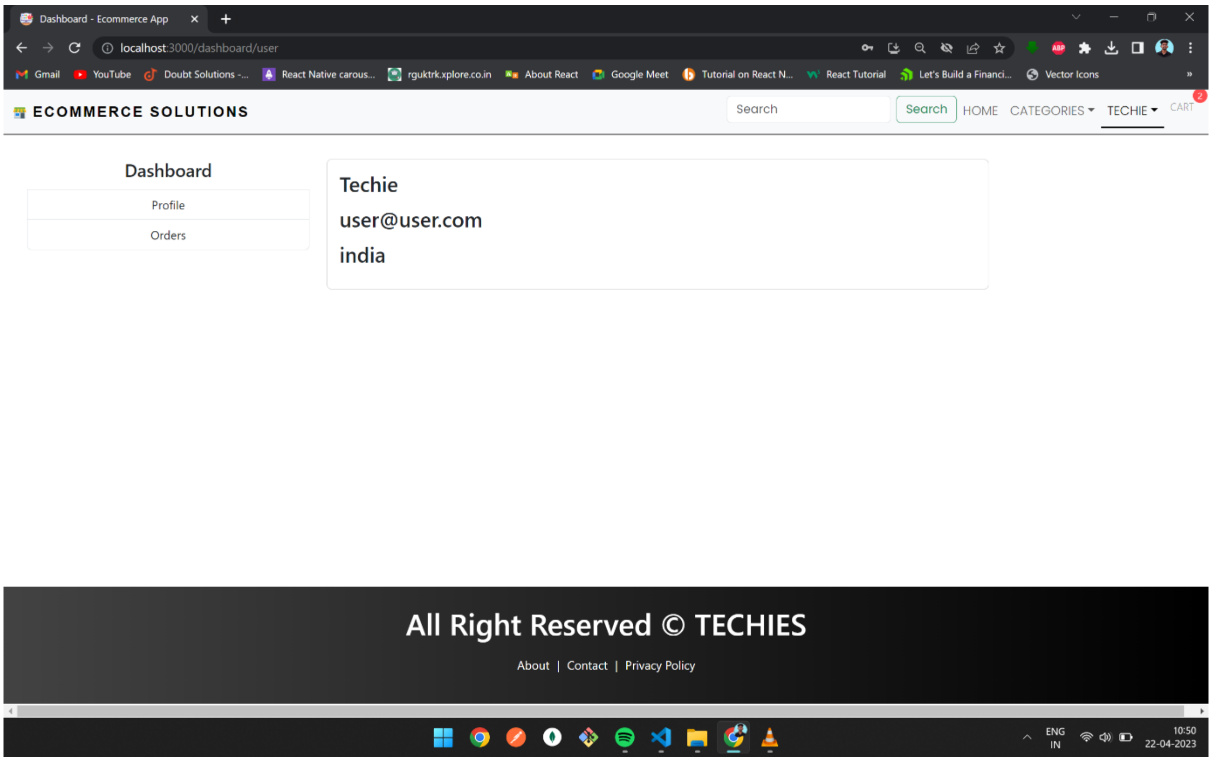
What is Your Favourite Sport

REGISTER

All Right Reserved © TECHIES

About | Contact | Privacy Policy

User Dashboard:


ECOMMERCE SOLUTIONS

Dashboard

Profile

Orders

Techie

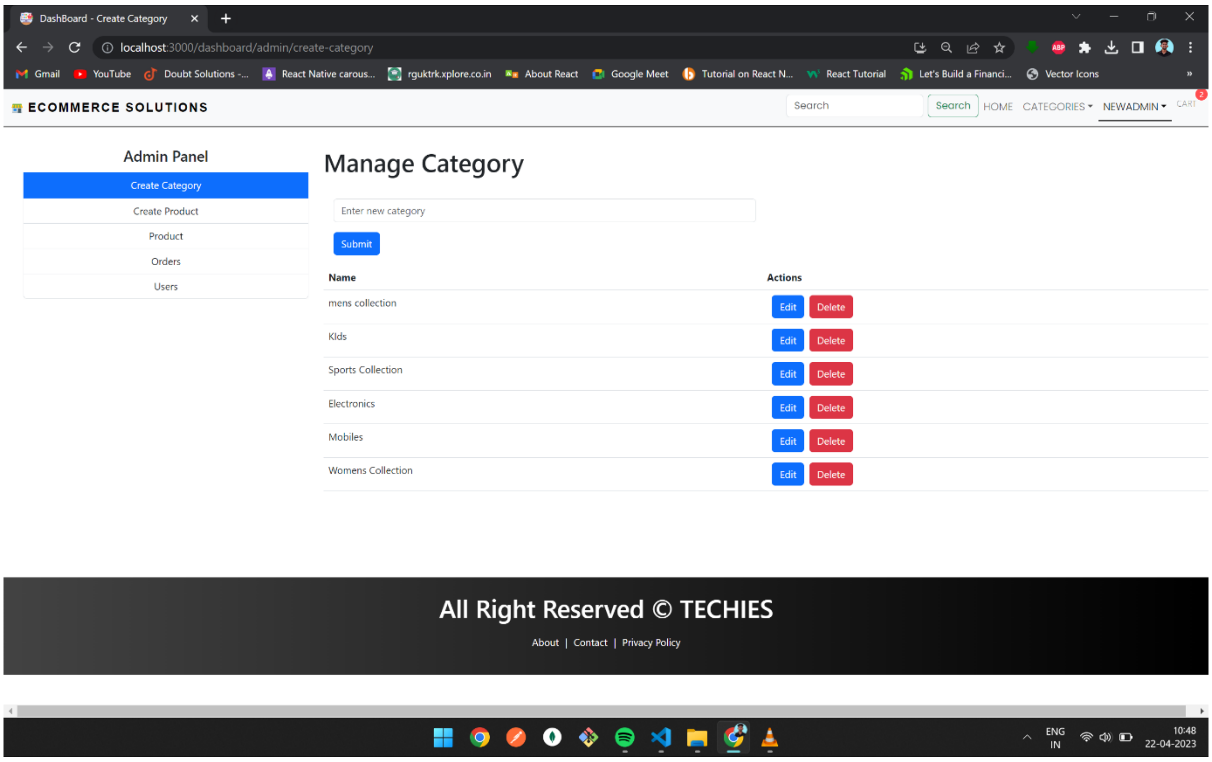
user@user.com

india

All Right Reserved © TECHIES

About | Contact | Privacy Policy

Create Category:



DashBoard - Create Category

localhost:3000/dashboard/admin/create-category

ECOMMERCE SOLUTIONS

Admin Panel

- Create Category
- Create Product
- Product
- Orders
- Users

Manage Category

Enter new category

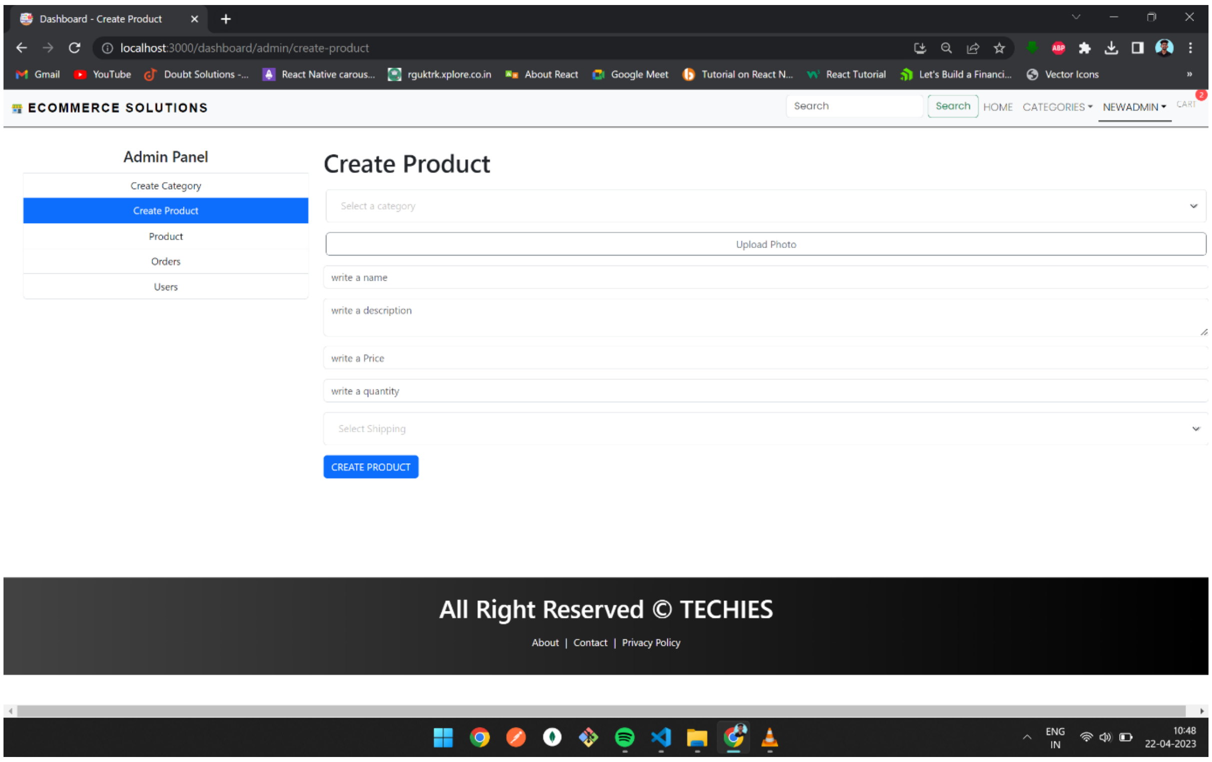
Submit

Name	Actions
mens collection	Edit Delete
Kids	Edit Delete
Sports Collection	Edit Delete
Electronics	Edit Delete
Mobiles	Edit Delete
Womens Collection	Edit Delete

All Right Reserved © TECHIES

About | Contact | Privacy Policy

Create Product:



Dashboard - Create Product

localhost:3000/dashboard/admin/create-product

ECOMMERCE SOLUTIONS

Admin Panel

- Create Category
- Create Product
- Product
- Orders
- Users

Create Product

Select a category

Upload Photo

write a name

write a description

write a Price

write a quantity

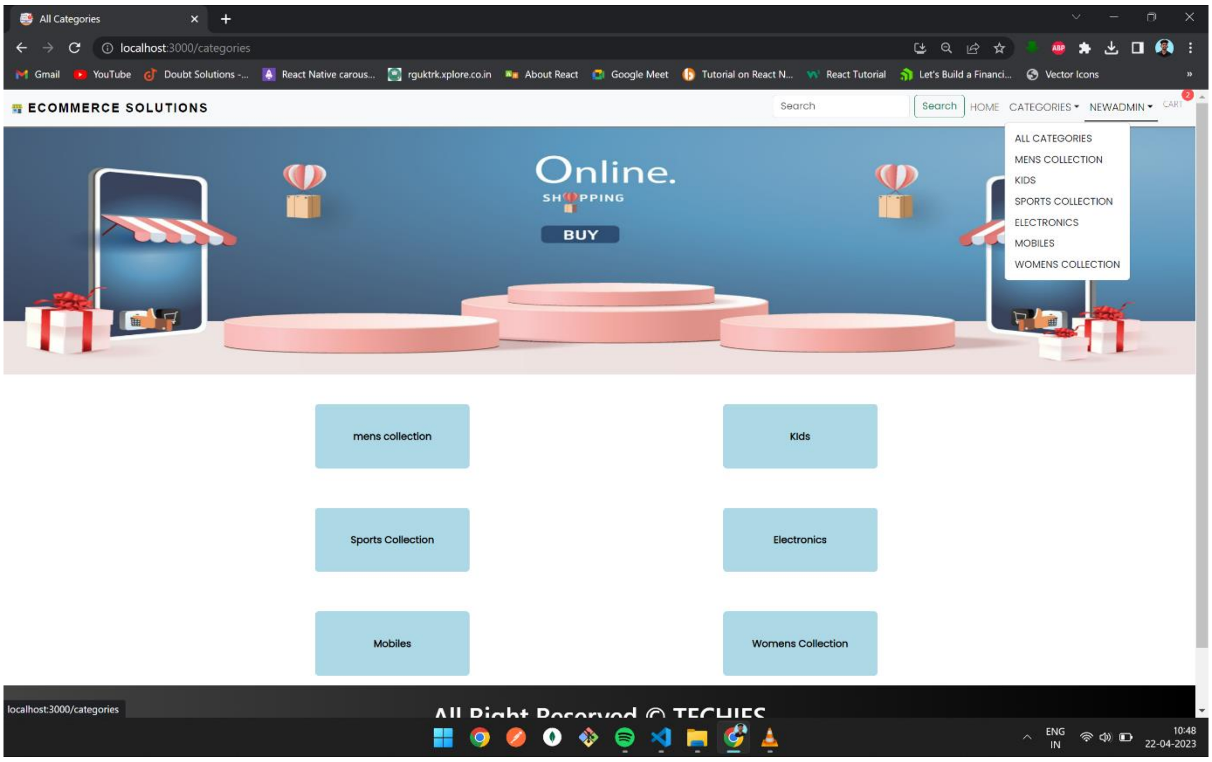
Select Shipping

CREATE PRODUCT

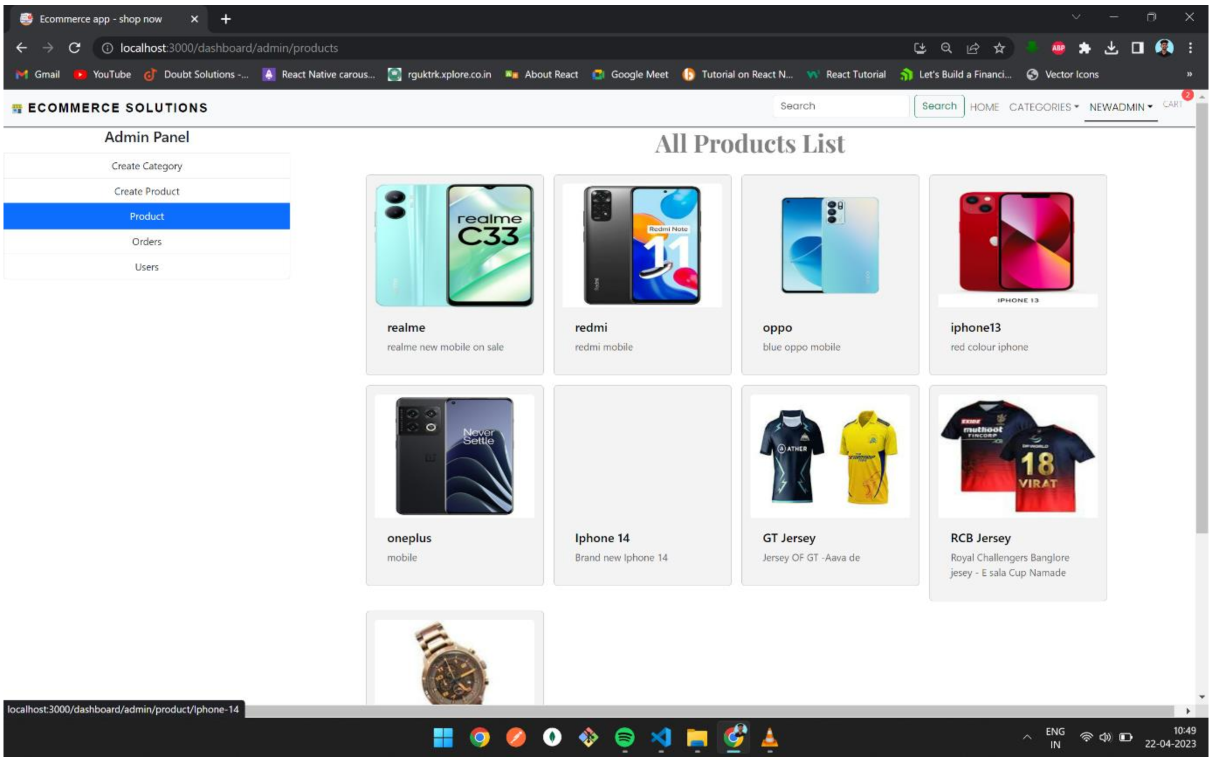
All Right Reserved © TECHIES

About | Contact | Privacy Policy

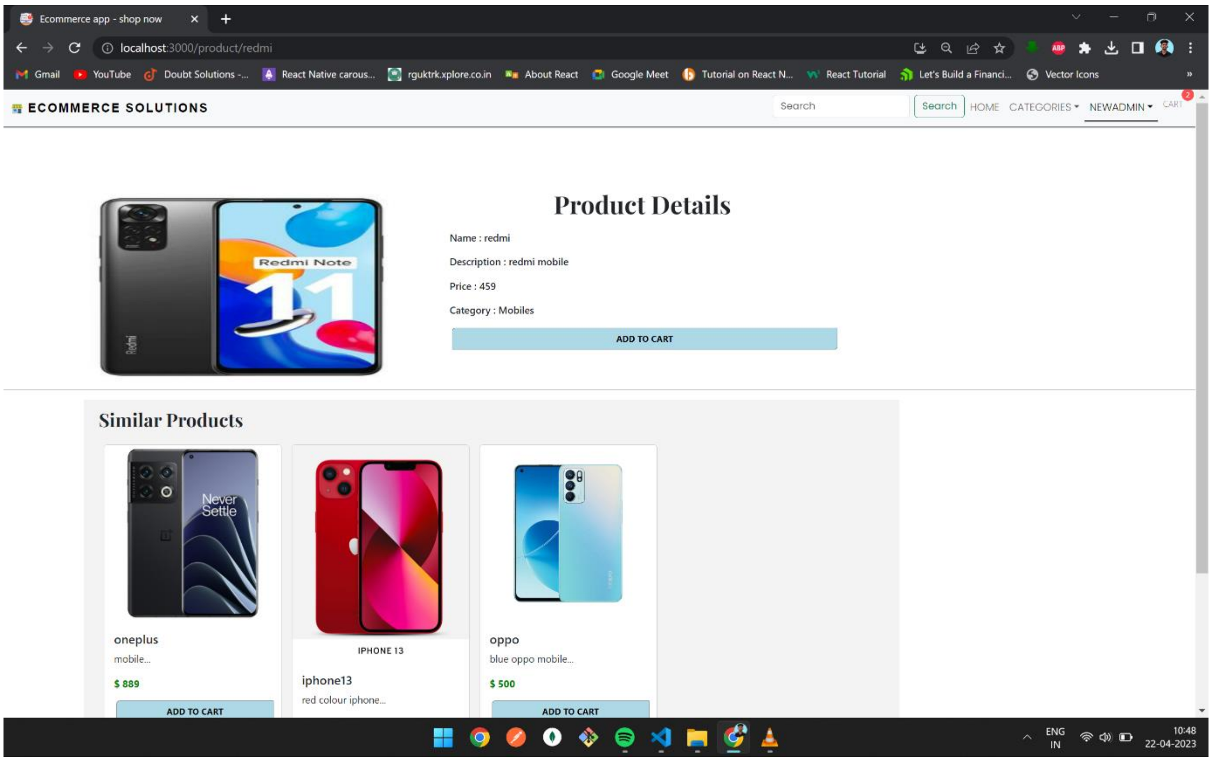
All Categories:



All Products List:



More Details 1:



Product Details

Name : redmi

Description : redmi mobile

Price : 459

Category : Mobiles

ADD TO CART

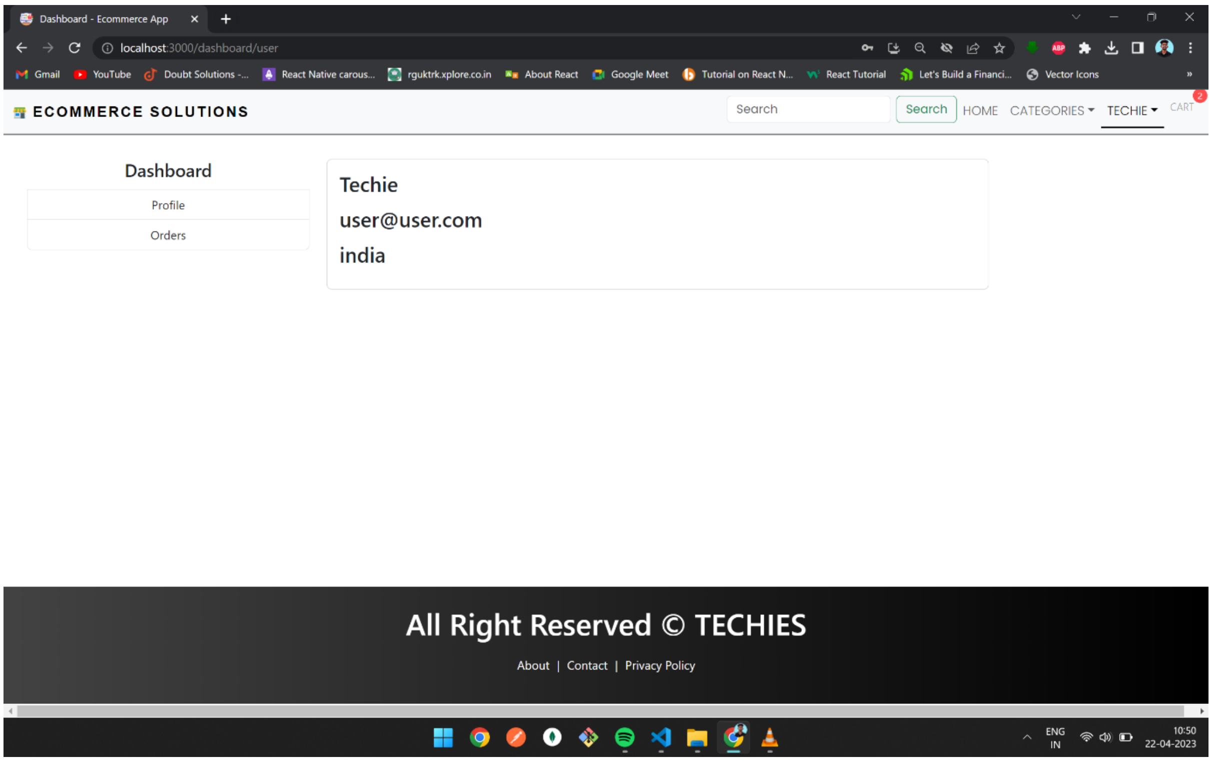
Similar Products

oneplus
mobile...
$ 889
ADD TO CART

iphone13
red colour iphone...

oppo
blue oppo mobile...
$ 500
ADD TO CART

User Dashboard:



Dashboard

Profile

Orders

Techie

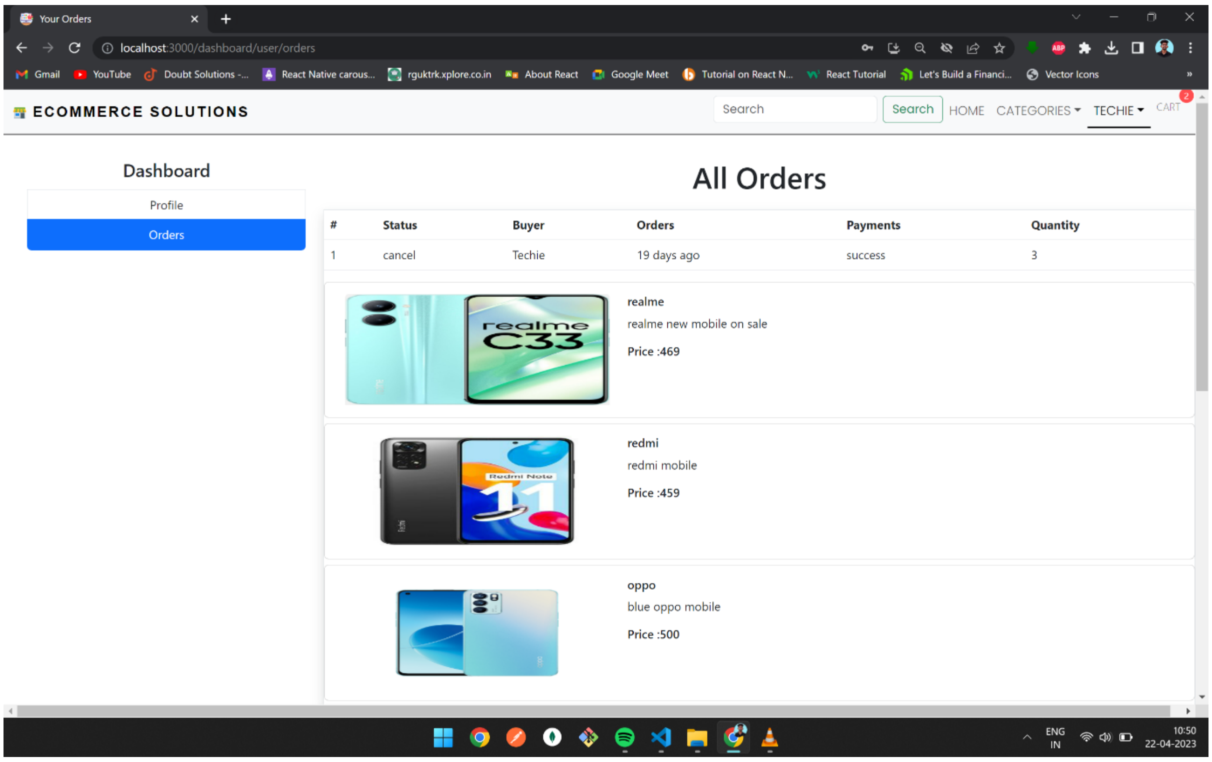
user@user.com

india

All Right Reserved © TECHIES

About | Contact | Privacy Policy

User Order Status:


localhost:3000/dashboard/user/orders

ECOMMERCE SOLUTIONS

Dashboard

Profile

Orders

All Orders

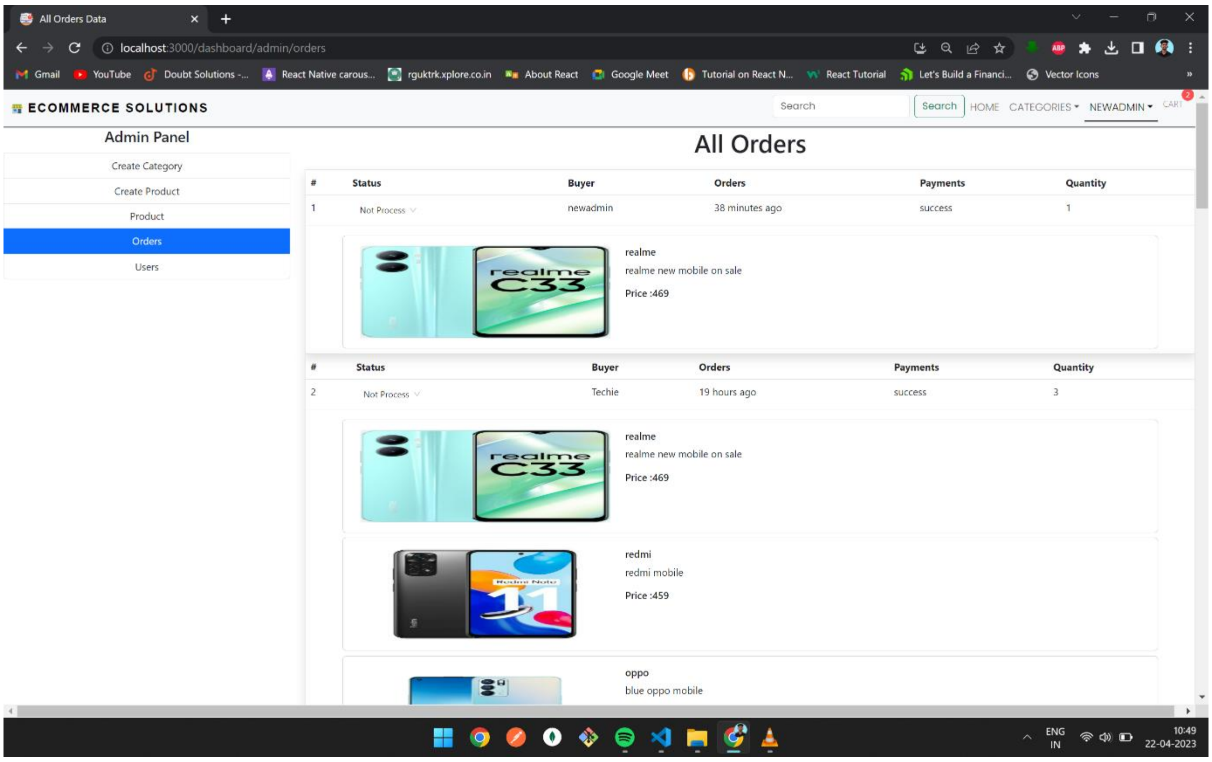
#	Status	Buyer	Orders	Payments	Quantity
1	cancel	Techie	19 days ago	success	3

realme
realme new mobile on sale
Price :469

redmi
redmi mobile
Price :459

oppo
blue oppo mobile
Price :500

Order Stats:


localhost:3000/dashboard/admin/orders

ECOMMERCE SOLUTIONS

Admin Panel

Create Category

Create Product

Product

Orders

Users

All Orders

#	Status	Buyer	Orders	Payments	Quantity
1	Not Process	newadmin	38 minutes ago	success	1

realme
realme new mobile on sale
Price :469

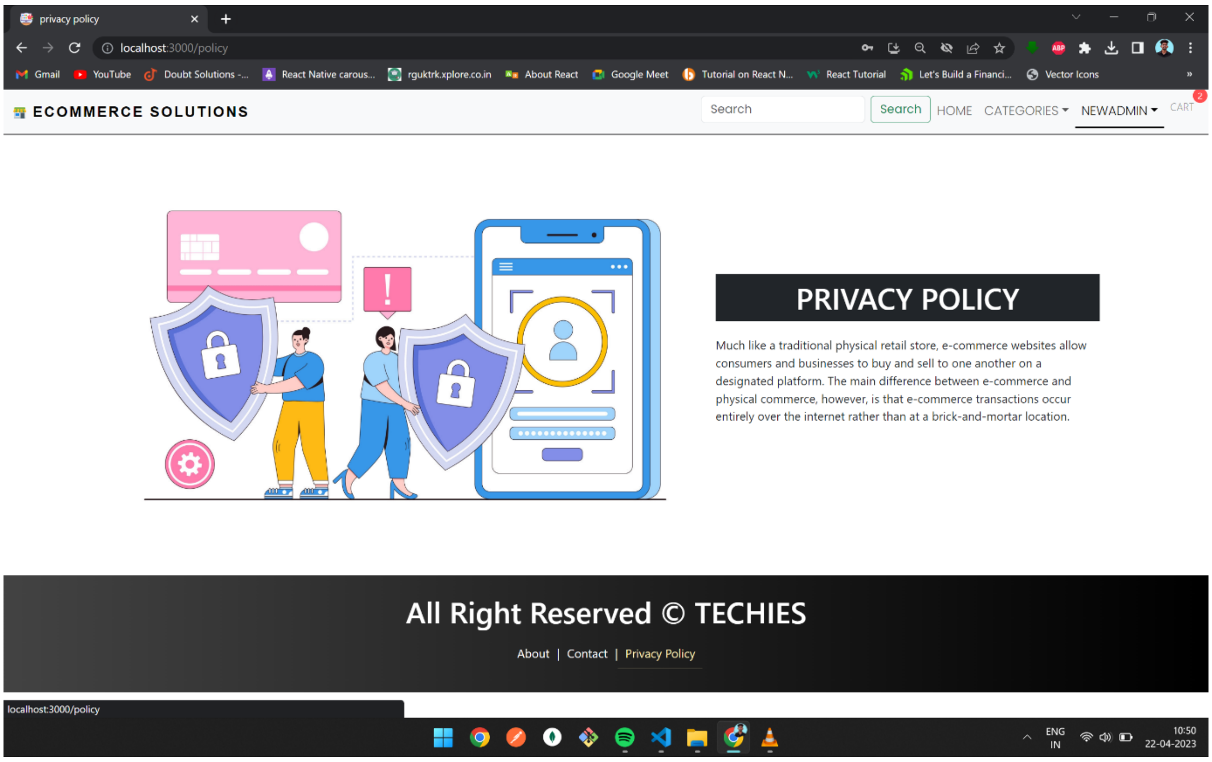
#	Status	Buyer	Orders	Payments	Quantity
2	Not Process	Techie	19 hours ago	success	3

realme
realme new mobile on sale
Price :469

redmi
redmi mobile
Price :459

oppo
blue oppo mobile

Privacy Policy:

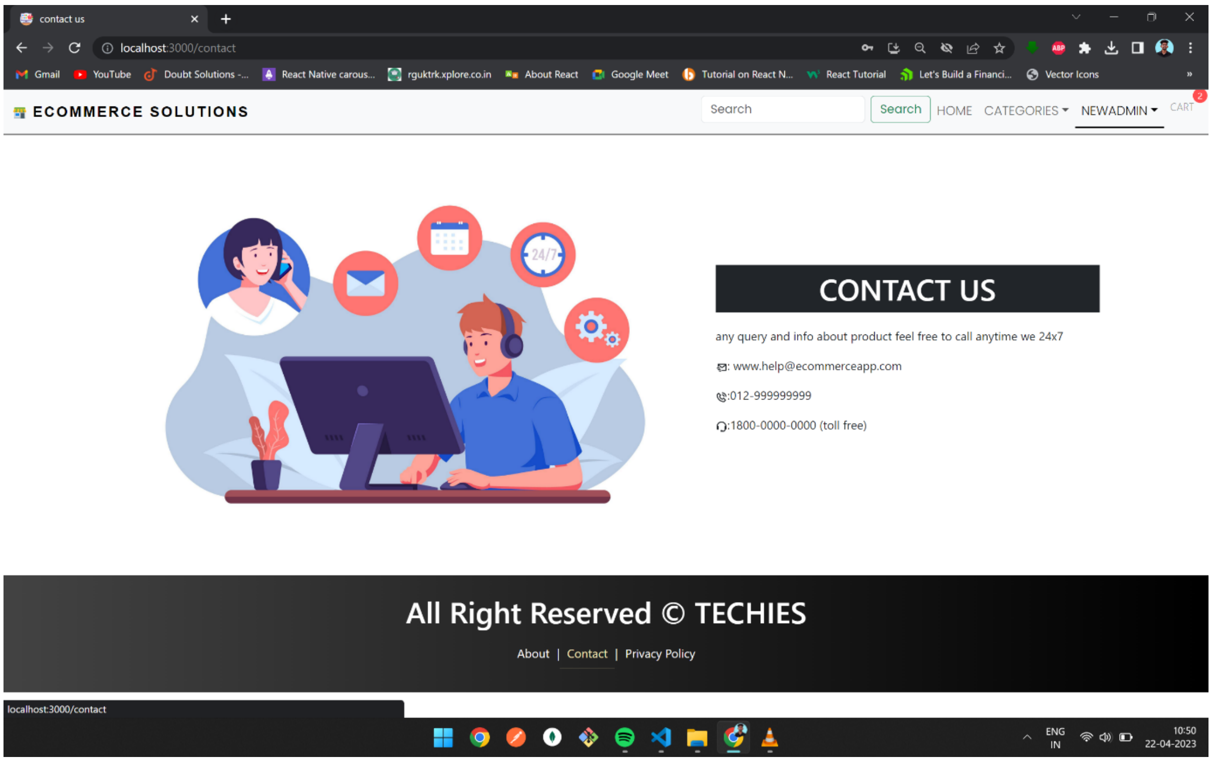


PRIVACY POLICY

Much like a traditional physical retail store, e-commerce websites allow consumers and businesses to buy and sell to one another on a designated platform. The main difference between e-commerce and physical commerce, however, is that e-commerce transactions occur entirely over the internet rather than at a brick-and-mortar location.

All Right Reserved © TECHIES

Contact Us:



CONTACT US

any query and info about product feel free to call anytime we 24x7

✉: www.help@ecommerceapp.com

☏:012-999999999

🎧:1800-0000-0000 (toll free)

All Right Reserved © TECHIES

Interpretation Of The Project:

Our purpose was to find out whether the efficiency of an e-commerce app will increase with MERN stack and usage of modern tools. What we found out was it actually increases the efficiency. React itself is making use of the same code again and again, adding Redux, Material UI and Axios libraries to it not only increases the efficiency, but it also makes the development process much easier. Paired with using JSON for data transmission and robust APIs from Express JS backend make the application more efficient. Using GitHub made the merging of code much easier and efficient because it will automatically merge and points out if there are any conflicts there. It was also helpful to find many issues we had with our code. Changes made after that also increased the code efficiency. Using selenium to automate the testing of our web application saved us much time because otherwise it will take a long time to write test cases. And it helped us find many bugs and errors in our application which helped us to increase the application efficiency. Looking at above mentioned details we can see that using modern tools with MERN stack we can make an Ecommerce web application more efficient.

Conclusion:

The main goal is to create an E-commerce web application that sells chocolatier items online using the frontend, backend, and database. This website is a completely functional website, starting with login authentication, admin authorization, adding things to the shopping cart, and using the payment gateway. Any branding or category sector can be used, whether on a small or large scale. They can easily use the web application, and without much effort, they can add products and establish new categories. The buyer will find it quite appealing to view the products while seated at home or at work. Smallscale businesses will benefit greatly from being able to sell products directly to consumers rather than through wholesalers or huge retail intermediaries which make both parties happy. And as we have used more modern tools boards for the project management and GitHub for version controlling that make much easier in the development stage and also for the planning stage.

REFERENCES:

1. https://react.dev/
2. https://nodejs.org/
3. https://www.mongodb.com/
4. https://www.postman.com/
5. https://expressjs.com/
6. https://getbootstrap.com/
7. https://blog.logrocket.com/mern-stack-tutorial/
8. https://www.geeksforgeeks.org/mern-stack/
9. https://www.youtube.com/watch?v=VsUzmlZfYNg