DHARMSINH DESAI UNIVERSITY,NADIAD

FACULTY OF TECHNOLOGY

Department of Computer Engineering

B. Tech. CE Semester – V

Subject: (CE – 520) Advanced Technologies

Online Grocery Store
(Group no - 1)

Prepared by:	**Prepared by:**
Name: Chudasama Akshayrajsinh Roll No: CE020 ID: 21CEUOG108	Name: Darji Mit Roll No: CE025 ID: 21CEUON151

Dharamsinh Desai University,Nadiad

Faculty of Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of **Advanced Technology** and Recorded in this Journal is the bonafide work of

Chudasama Akshayrajsinh(CE020) (21CEUOG108)

Darji Mit(CE025) (21CEUON151)

Of B.Tech semester V in the branch of computer Engineering during the academic year 2021-2022

Prof. Siddharth Shah	Dr. C.K. Bhensdadia
Dept of computer engg.	Head of Dept. of Computer engg.
Faculty of technology	Faculty of Technolog

3) Contents

4) Abstract

"Development of an Online Grocery Store Using the MERN Stack"

Objective:
This project aims to create a user-friendly online grocery store platform that provides a seamless shopping experience for customers, offers a wide range of products, and addresses the increasing demand for convenient and safe grocery shopping solutions.

Online Grocery Shopping is a way of buying food and other household necessities using a web-based shopping service.

Online Grocery Store is a supermarket that allows online purchasing of fruits, vegetables, cooking oils and food grains etc. The user can conveniently use your computer to place your order online. Users can select the choice of grocery items he wants at the grocery store. Once the user has finalized the order, then he can add the items to the shopping cart. And after the payment the products will be shipped to the mentioned address.

One of the main benefits of online grocery shopping is the convenience it provides. With online grocery shopping, customers can place their order from the comfort of their own home, and have it delivered to their door or picked up at a convenient location.

This is particularly useful for people who are short on time, have mobility issues, or live in remote areas.

Additionally, many online grocery stores offer a wider range of products than physical stores, including organic, specialty and international items.

So, it becomes easy for the customer to order from a wide variety of products and save time by not going to the supermarket.

5) Introduction

In an era characterized by digital transformation and changing consumer preferences, the development of an online grocery store stands as a testament to the evolving landscape of e-commerce. This project represents an exploration into the world of online retail, specifically focusing on the grocery sector, which has witnessed a surge in demand for digital solutions to streamline the shopping experience. By leveraging cutting-edge technologies and a robust technology stack, this online grocery store aims to provide customers with a convenient, secure, and efficient means of purchasing their daily essentials.

The contemporary consumer seeks not only convenience but also a seamless and secure online shopping environment. This project endeavors to address these evolving needs. By leveraging cutting-edge technologies and a robust technology stack, this online grocery store aims to provide customers with a convenient, secure, and efficient means of purchasing their daily essentials.

-Technology/Platform/Tools Used

The creation of this online grocery store has relied on a range of modern technologies, platforms, and tools to ensure a seamless and user-friendly experience. The key components and tools used in the development of this project include:

MERN Stack: The project is built on the MERN stack, a popular and powerful combination of technologies consisting of MongoDB (for the database), Express.js (for the server), React (for the front-end user interface), and Node.js (for server-side JavaScript).

Front-End Technologies: React, along with HTML, CSS, and JavaScript, forms the foundation of the front-end user interface, enabling dynamic and responsive web pages.

Back-End Technologies: Node.js and Express.js are employed on the server side to handle data processing, user authentication, and the seamless interaction between the front-end and the database.

Database: MongoDB, a NoSQL database, is used to efficiently store and manage product listings, user data, and transaction records.

User Authentication: The project incorporates secure user authentication using technologies like JWT (JSON Web Tokens) to ensure data privacy and security.

Responsive Design: The project is designed to be responsive, ensuring a seamless user experience across various devices, including desktops, tablets, and smartphones.

6) Software Requirement Specifications (SRS)

1. Introduction

1.1 Purpose

The purpose of an online grocery store is to document and clearly define the functional and non-functional requirements for the development of the online grocery store system. The SRS serves as a guide for the development team and stakeholders, outlining the features and functionality of the system, as well as any constraints or limitations that must be considered during the development process.It also helps to keep the development process aligned with the overall business goals and objectives.

1.2 Document conventions

Standard IEEE Conventions

1.3 Intended Audience and Reading Suggestions

We are making this document by keeping in mind different types of readers. This document will be useful for different audiences in various ways.

Developers:- They will use this document for guidance for the design and implementation phase.

Managers:- They will see the constraints all cover properly. Time and cost is within limits or not.

Marketing Staff:- They can use this document to make advertisements for this web store because by reading this document they will know what the system will do? How this system is different from others.

User:- They can ensure there self by reading the SRS that their needs being met in the web store or not

Testers:- They will test the implementation of the project according to the SRS base.

DocumentationWriter:- They will use this document during the documentation of the project. It will be really helpful for them.

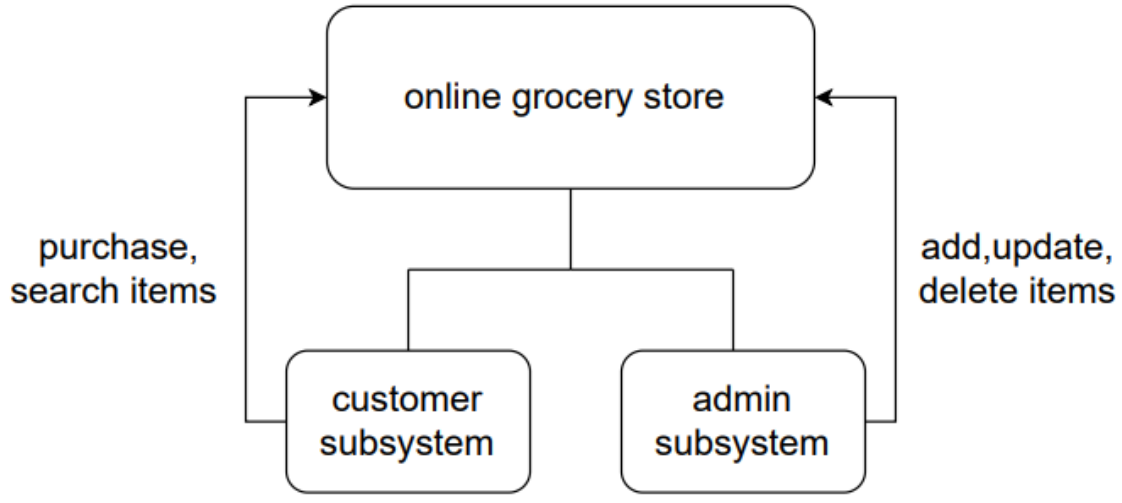
1.4 Product Scope

The system should allow customers to create an account and log in to access the store's features and functionalities.Customers should be able to browse products by category, search for specific products, and view product details such as images, descriptions, and pricing.The system should include a shopping cart feature that allows customers to add and remove products from their cart and to view the total cost of their order.Customers should be able to complete their purchase by providing their shipping and payment information and submitting the order.The system should include a product management feature for store administrators to add, update, and remove products from the store's inventory.The system should include features for store administrators to create and manage promotions and discounts to attract customers.The online store should be mobile-friendly and responsive to different screen sizes and devices.Customers should be able to contact customer service for assistance with their order or for general inquiries.

2. Overall Description

2.1 Product Perspective

The system includes the user subsystem as well the admin subsystem. The online grocery system provides an outstanding way of bringing customers on an online platform to make purchases in an efficient and secure manner irrespective of the distance between the two. It is a platform for customers to shop items online without having to visit a store. This system is a one stop for customers to shop from millions of products online**.**



2.2 Product Functions

- **Register:** for customers
- **Login:** for customers and admin
- **Logout:** for customers and admin
- **Edit Account Details:** for customers and admin
- **Search item:** for customers
- **View item:** for customers
- **Add item to cart:** for customers
- **View shopping cart:** for customers
- **Change items in cart:** for customers
- **Proceed to buy:** for customers
- **Delivery & payment:** for customers
- **Place order:** for customers
- **Cancel order:** for customers
- **Return item:** for customers
- **Add items:** for admin
- **Modify items:** for admin
- **Remove items:** for admin

2.3 User Classes and Characteristics

Customer - He/she is a verified user of the system who is intended to buy a product using the platform. The functions used by customers are register, view account, login, browse item, view item, buy item now, add to cart, view cart, proceed to buy, enter delivery address, enter mode of payment, make payment, place order, view orders,cancel order, return item, logout.

Admin - He/she is a verified user of the system. The product functions used by admin are view account, login,add items,modify items,remove items from system.

2.4 Operating Environment

There is only one mode of using the software - web applications. Web applications can be run on Windows 10: Google Chrome, Mozilla Firefox,Internet Explorer,Microsoft Edge, Mac OS X: Apple Safari. The Internet is a basic necessity for the system to be accessed.

2.5 Design and Implementation Constraints

The system should be designed to protect customer's personal and financial information, and comply with relevant security standards.The system should be designed to provide fast response times and handle a large number of concurrent users.The system should be compatible with the most popular web browsers, including Chrome, Firefox, Safari, and Internet Explorer.The system should be designed to be mobile-friendly and responsive to different screen sizes and devices.The system should be designed to be accessible to users with disabilities, in compliance with relevant accessibility standards, such as WCAG 2.0.The system should be developed using current web technologies, such as HTML and CSS.

2.6 User Documentation

Any user of the software system is the target audience for user documentation generated about the software system. A range of short document types (e.g., guidelines, tutorials, frequently asked questions, user manuals, on-line help, and trouble-shooting manuals) in HyperText Markup Language (HTML) and/or Portable Document Format (PDF) format must describe the use of the software system.

2.7 Assumptions and Dependencies

Under the assumption that a Windows/iOS/ Linux based operating system is available with C++/Python working along with WSGI Server as a dev server with Sqlite database, designing a modular view of the system is smooth. The recommendation models are assumed to be dependent on the server and its functionalities though relevant to customers will be more clearly defined by the server

3. External Interface Requirements

3.1 Hardware Interfaces

The system is compatible with a wide range of hardware, including desktop computers, laptops, and mobile devices. The system should be hosted on one or more web servers capable of handling the expected traffic and processing power requirements.The system should use a RDBMS to store and manage customer and product data.The system should be connected to a high-speed network, providing a fast and reliable connection to customers. RAM:- 4GB, Hard Drive Storage Needed:- 2GB

Other Hardware Requirement: None

3.2 Software Interfaces

The system should be compatible with the most popular operating systems, including Windows, MacOS, and Linux and also compatible with the most popular web browsers, including Chrome, Firefox, Safari. The system should be built using a programming language such as JavaScript, Python.The system should use RDBMS such as MySQL,MongoDB to store and manage customer and product data. The system should integrate with a payment gateway, such as Stripe or PayPal.

3.3 Communications Interfaces

The user can access the online grocery system through the internet by searching the system's name on the web browser. The system shall use the HTTP protocol for communication over the internet.The system shall give a confirmation to the customer that their order is placed by sending a message to the customer's email id.

4. System Features

4.1 admin and customer

R.1 :- User Accounts
Description :- Users from both subsystems- customers and admin must have an account on the grocery system.Customers accounts will hold information about their name, email id or password. Both users can view and login to their accounts and even edit account details in the future.

R.1.1 :- Sign up
User :- Customers
Input :- Enters name, email-id,password
Output :- Successfully registered, the login page is displayed

R.1.2 :- Sign in/Login
User :- Customers and Admin
Input :- Enters registered email-id,password
Output :- Successfully logged in,the home page is displayed

R.1.3 :- Logout
User :- Customers and Admin
Input :- Click on ‘logout’ in home page
Output :- User is logged out of the account, Login page will be displayed

R.1.4 :- Edit account details
User :- Customers and Admin
Input :- Re-Enter the details which need to be updated
Output :- Successfully updated List

4.2 Customer

R.2 :- The search facility
Description :- Customers can search for an item from the large catalogue of items in the grocery system by two methods.They can either search for a product using keywords related to the product or by providing the category of the product.

R.2.1 :- Search item
Input :- The customer can request the name of the product he/she is interested in.
Output :- List of products related to the item searched for.

R.2.2 :- Search category
Input :- The customer can request the category.
Output :- List of products related to the category searched for.

R.3 :- The shopping cart facility
Description :- Once the customer views and selects an item that he/she wishes to purchase, one must add the item to cart using the add to cart button. The shopping cart contains all the items that the customer intends to buy, there is one shopping cart associated with one customer account.

R.3.1 :- Add item to cart
Input :- Select the interested item to be added to cart.
Output :- Added item to cart, will remain in the same page

R.3.2 :- View shopping cart
Input :- After there are few items in the cart, it can be seen.
Output :- List of items that were added to cart is displayed along with item details and total cost

R.3.3 :- Change items in cart
Input :- select on delete item present next to the item to delete the item from the cart and to change the quantity of items click on ‘+’ to increase quantity by 1, or ‘-’ to decrease quantity by 1

Output :- The item will be removed from the page if it is deleted

R.3.4 :- Proceed to buy
Input :- Buy the selected item and payment details can be entered.
Output :- Payment page is displayed

R.4 :- Payment
Description :- Once the customer proceeds to buy from the cart page, the delivery and payment page is visible where the total cost of all the items in the card is visible and the customer has to fill in the details necessary such as address and payment method.

R.4.1 :- Delivery details
Input :- Enter the delivery address
Output :- The payment option is displayed

R.4.2 :- Payment options
Input :- select the payment option
Output :- show the details of the selected payment option

R.4.3 :- Payment details
Input :- Enter the payment details
Output :- Payment is successful, the place order is displayed

R.4.4 :- Order Placed
Input :- select place order to conform the order
Output :- Order successfully placed, the home page is displayed

R.5 :- Orders and returns
Description :- The customer can view and manage orders that are yet to be delivered to the customer. He/she can cancel the order that is yet to be delivered or return the item that is delivered and get the paid amount back, or view past orders and returns.

R.5.1 :- View orders
Input :- Select the view order to see all the orders placed
Output :- the past and existing orders and past returns are displayed

R.5.3 :- Return item
Input :- Select the item to be returned
Output :- The item is successfully booked for return

4.3 Admin

R.6 :- Admin features
Description :- Admin can add the items one by one in the Grocery Store. Admin can also remove the item and modify items by name,price and image.Admin also views new orders placed by the customers.
R.6.1 :- Add items
Input :- Enter details of all the items to be added
Output :- The items successfully added, customers can view these items

R.6.2 :- Remove Items
Input :- Search the item by the name and id of the item to be removed
Output :- The item successfully removed

R.6.3 :- Modify Items
Input :- Modify the items such as name, price, image, etc.
Output :- The item successfully modified

R.6.4 :- View new orders
Input :- Select View new orders placed by the customers.
Output :- Displays the orders for his/her items from the customers

5. Other Nonfunctional Requirements

5.1 Performance Requirements
You will be signed in within 20 sec.If anybody makes a new account then he will

receive confirmation email within 5 minutes.Search results shown within 15 sec.Credit cart validate within 5 sec.Web support 200 customers logged at the same time.The server will be working 24X7 time.

5.2 Safety Requirements

The system should clearly indicate to the user any potential risks associated with certain actions.The system should validate user input and prevent any data injection attacks.

5.3 Security Requirements

The system should use encryption to protect sensitive data, such as user passwords.The system should have regular vulnerability assessments and penetration tests to identify and remediate any security weaknesses.

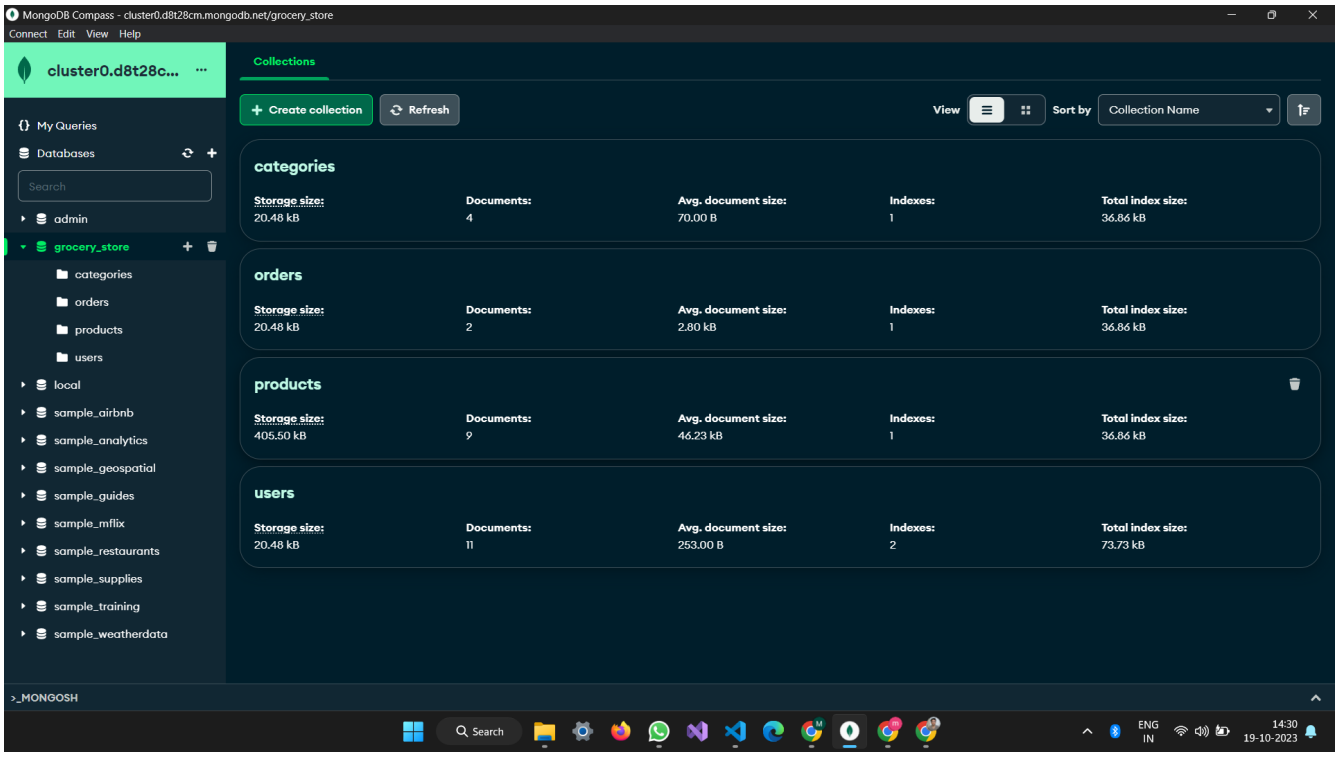
5.4 Software Quality Attributes

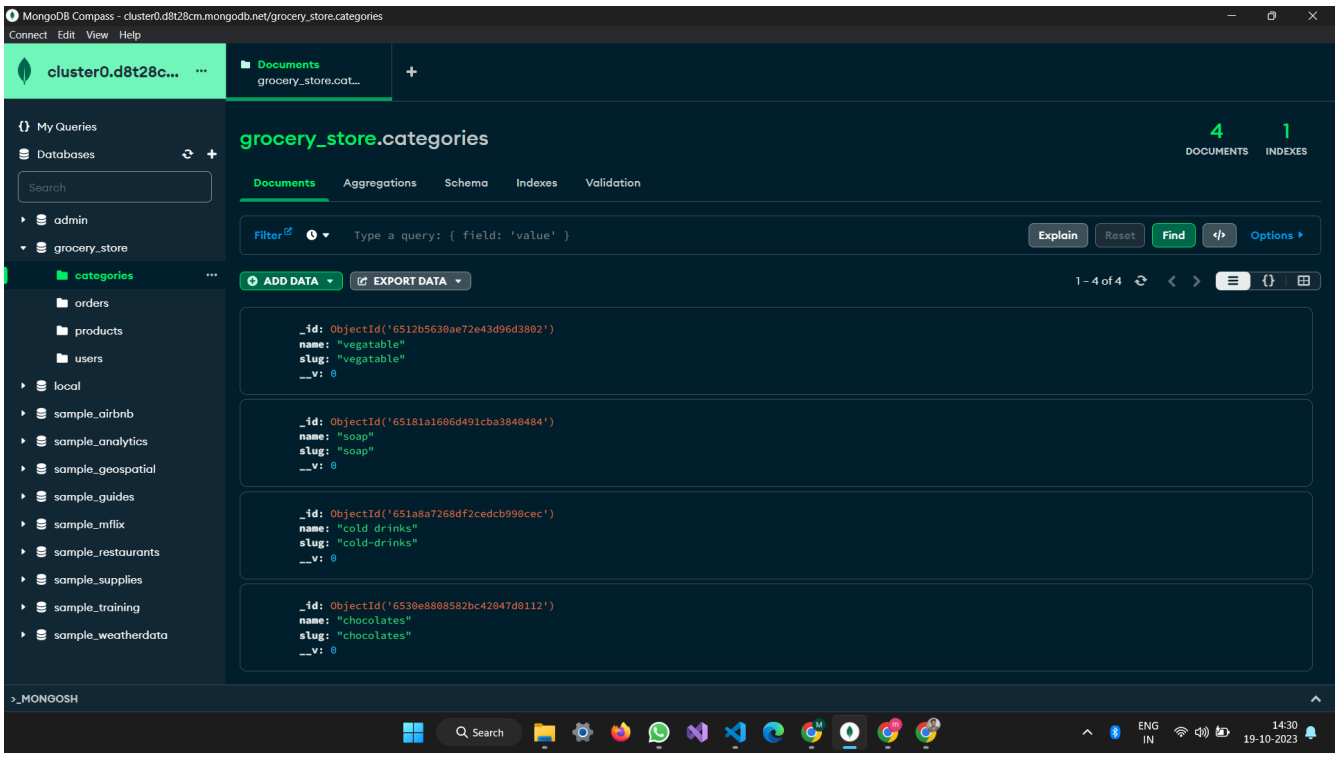
Security: The application is password protected and also any update of new product entries and order processing is done by only privileged users.
Maintainability: The application is to be designed so that it is easily maintained.Also it should allow incorporating new requirements in any module of the system.
Reliability: The application will be able to handle two orders.When a user confirms his/her order the database will be updated immediately and the next user will not face problems in ordering.
Portability: The application will be easily portable on any window based system.

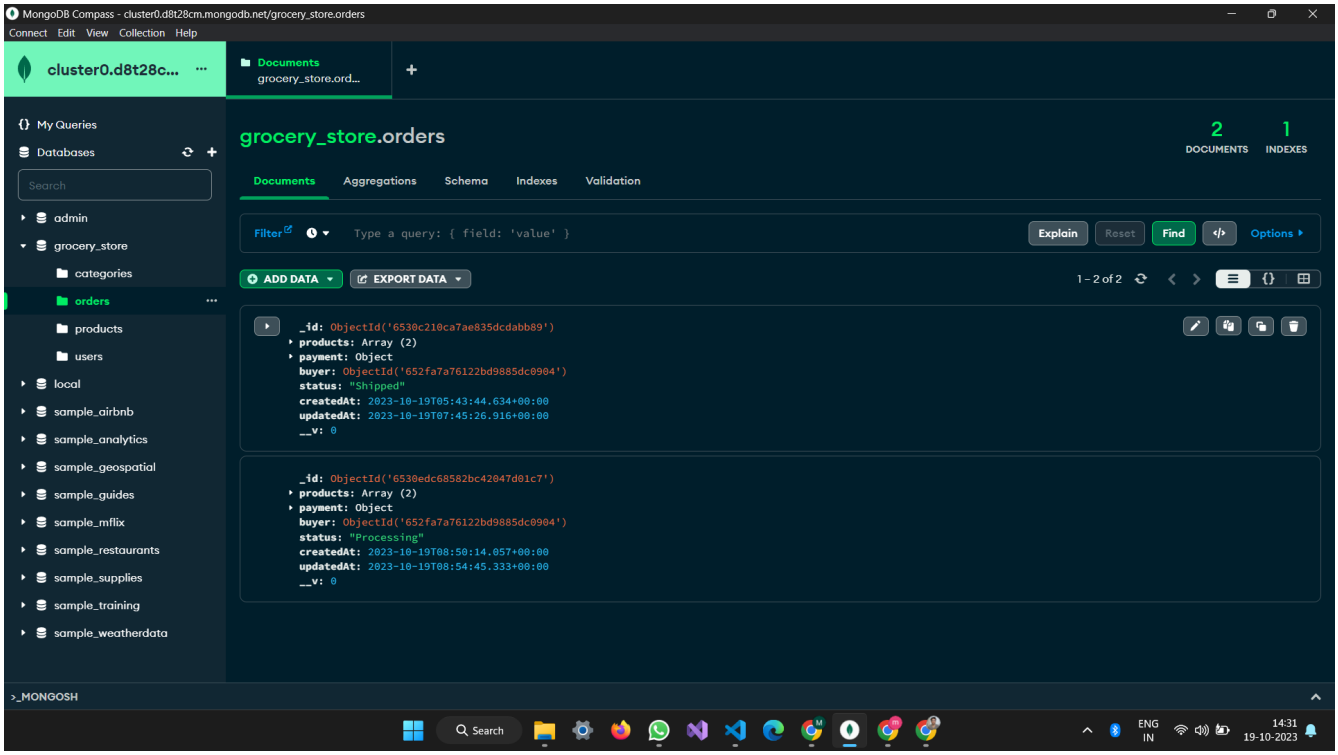
5.5 Business Rules

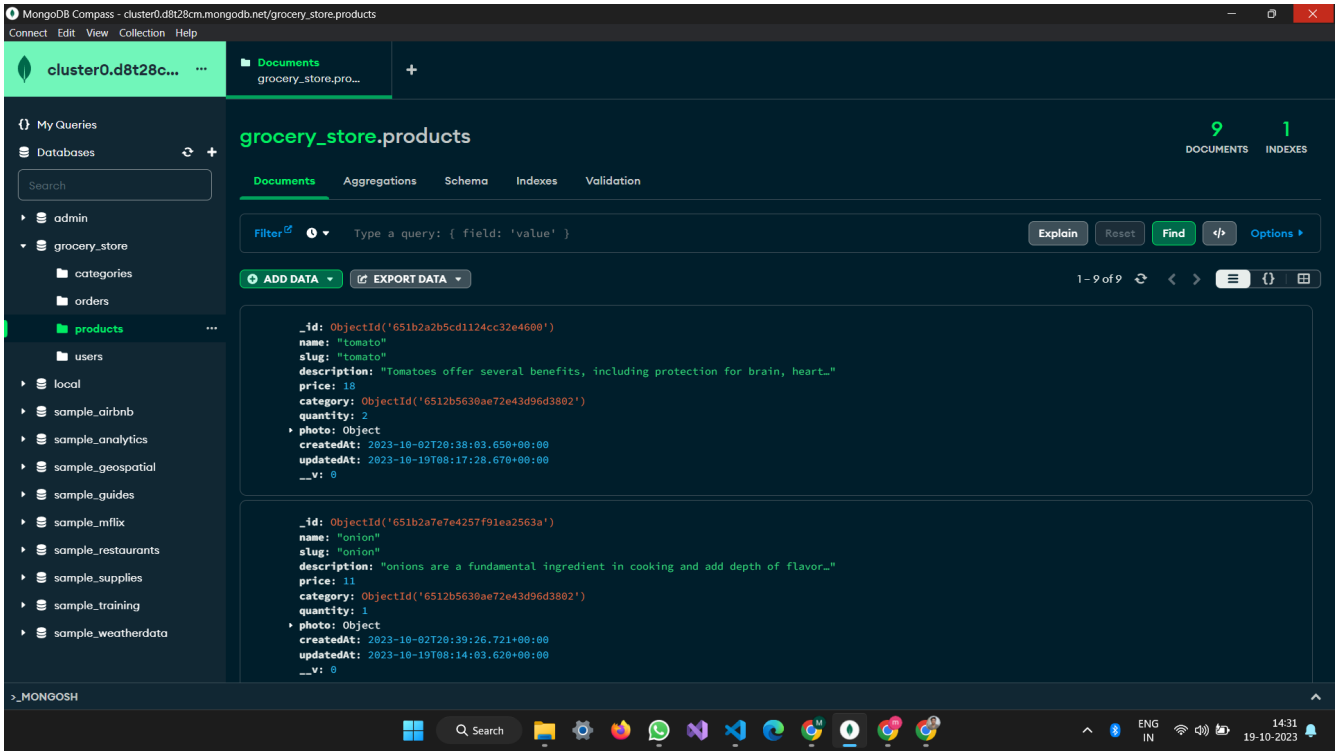
Given the presence of two subsystems for the customers and admin, the two types of users have different levels of privileges - including functionalities.Some of the functionalities common to the two subsystems are- login, viewing account details.Some functionalities specific to admin subsystems are add,modify and remove items. Customers also have browsing features, cart features, delivery details, payment options.

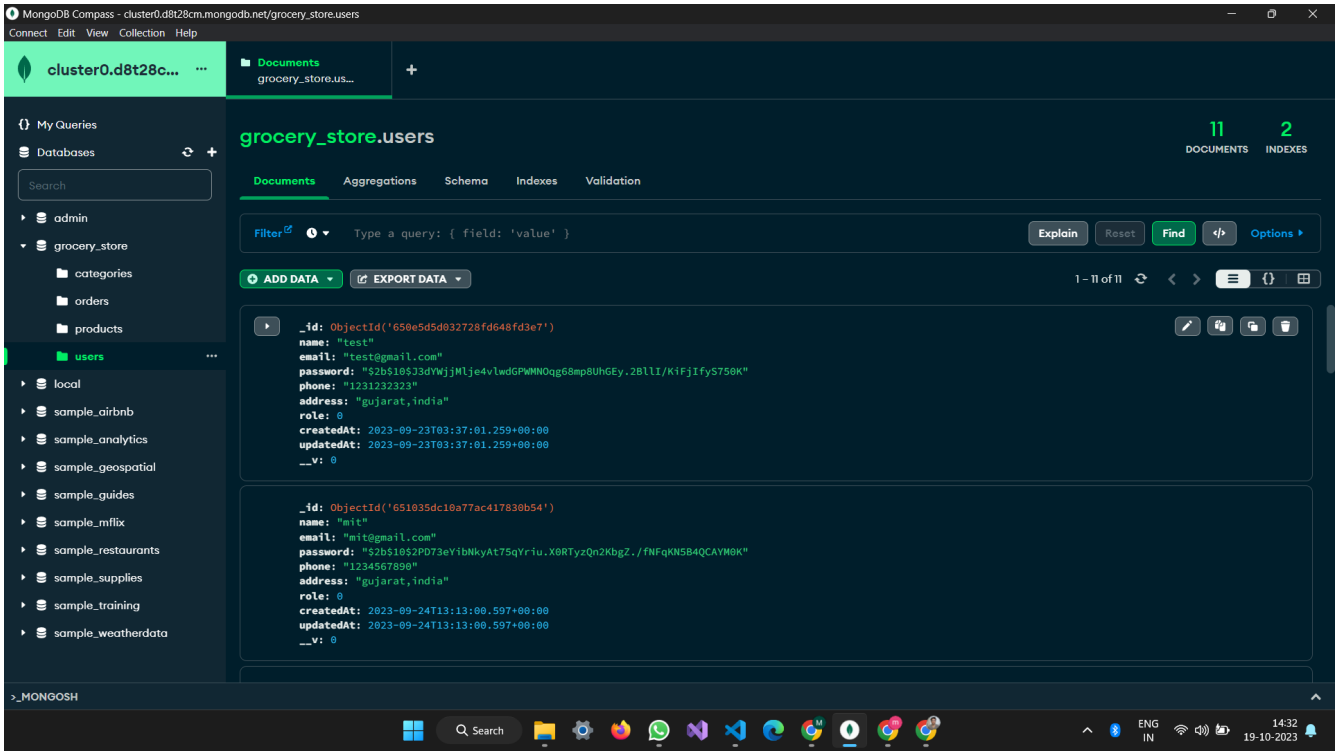
7) Database Design






MongoDB Compass - cluster0.d8t28cm.mongodb.net/grocery_store.orders
Connect Edit View Collection Help
cluster0.d8t28c...
Documents
grocery_store.ord...
My Queries
Databases
Search
admin
grocery_store
categories
orders
products
users
local
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
grocery_store.orders
2
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' }
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 2 of 2
_id: ObjectId('6530c210ca7ae835dcdabb89')
products: Array (2)
payment: Object
buyer: ObjectId('652fa7a76122bd9885dc0904')
status: "Shipped"
createdAt: 2023-10-19T05:43:44.634+00:00
updatedAt: 2023-10-19T07:45:26.916+00:00
__v: 0
_id: ObjectId('6530edc68582bc42047d01c7')
products: Array (2)
payment: Object
buyer: ObjectId('652fa7a76122bd9885dc0904')
status: "Processing"
createdAt: 2023-10-19T08:50:14.057+00:00
updatedAt: 2023-10-19T08:54:45.333+00:00
__v: 0
>_MONGOSH
Search
ENG
IN
14:31
19-10-2023


MongoDB Compass - cluster0.d8t28cm.mongodb.net/grocery_store.products
Connect Edit View Collection Help
cluster0.d8t28c...
Documents
grocery_store.pro...
My Queries
Databases
Search
admin
grocery_store
categories
orders
products
users
local
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
grocery_store.products
9
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' }
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 9 of 9
_id: ObjectId('651b2a2b5cd1124cc32e4600')
name: "tomato"
slug: "tomato"
description: "Tomatoes offer several benefits, including protection for brain, heart…"
price: 18
category: ObjectId('6512b5630ae72e43d96d3802')
quantity: 2
photo: Object
createdAt: 2023-10-02T20:38:03.650+00:00
updatedAt: 2023-10-19T08:17:28.670+00:00
__v: 0
_id: ObjectId('651b2a7e7e4257f91ea2563a')
name: "onion"
slug: "onion"
description: "onions are a fundamental ingredient in cooking and add depth of flavor…"
price: 11
category: ObjectId('6512b5630ae72e43d96d3802')
quantity: 1
photo: Object
createdAt: 2023-10-02T20:39:26.721+00:00
updatedAt: 2023-10-19T08:14:03.620+00:00
__v: 0
>_MONGOSH
Search
ENG
IN
14:31
19-10-2023


MongoDB Compass - cluster0.d8t28cm.mongodb.net/grocery_store.users
Connect Edit View Collection Help
cluster0.d8t28c...
Documents
grocery_store.us...
My Queries
Databases
Search
admin
grocery_store
categories
orders
products
users
local
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
grocery_store.users
11
DOCUMENTS
2
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' }
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 11 of 11
_id: ObjectId('650e5d5d032728fd648fd3e7')
name: "test"
email: "test@gmail.com"
password: "$2b$10$J3dYWjjMlje4vlwdGPWMNOqg68mp8UhGEy.2BllI/KiFjIfyS758K"
phone: "1231232323"
address: "gujarat,india"
role: 0
createdAt: 2023-09-23T03:37:01.259+00:00
updatedAt: 2023-09-23T03:37:01.259+00:00
__v: 0
_id: ObjectId('651035dc10a77ac417830b54')
name: "mit"
email: "mit@gmail.com"
password: "$2b$10$2PD73eYibNkyAt75qYriu.X0RTyzQn2KbgZ./fNFqKN5B4QCAYM0K"
phone: "1234567890"
address: "gujarat,india"
role: 0
createdAt: 2023-09-24T13:13:00.597+00:00
updatedAt: 2023-09-24T13:13:00.597+00:00
__v: 0
>_MONGOSH
Search
ENG
IN
14:32
19-10-2023

8) Implementation Details

i) Modules created and brief description of each module.

1.Login Module - Lets the user login into the system by checking credentials provided through the login form and checking it against the user table.

2.Register Module - Lets the user register into the system by giving the credential into the register form. The user gets registered depending on whether they are already registered into the system.

3.Managing products Module - This module is responsible for providing the crud operations on the products in the database so that the user can issue the book by searching.

4. Manage Category Module - The admin can modify, add, delete the categories of the grocery items.

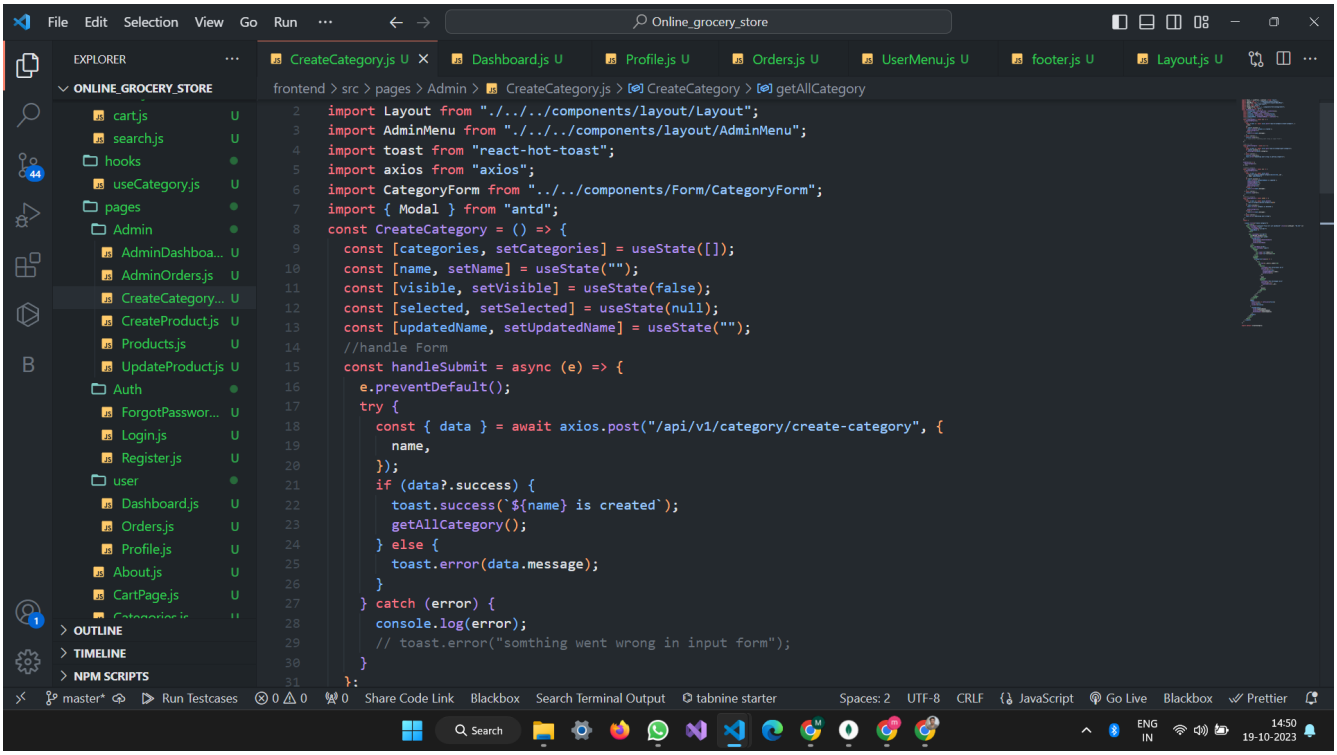
5. Search Product Module- This helps the user to search any product which he wishes to buy.

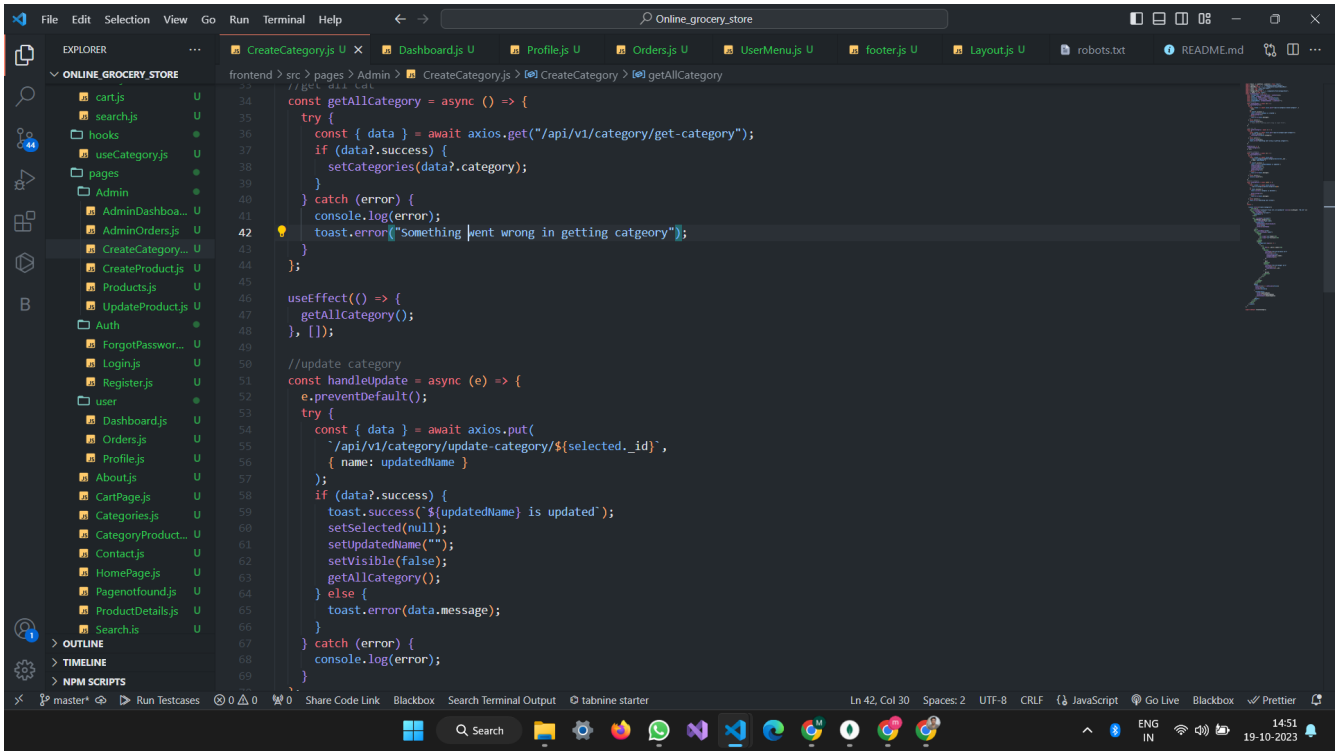
6. Payment Module - It is used by the user to buy the products added by him in the cart.

ii) Function prototypes which implements major functionality.

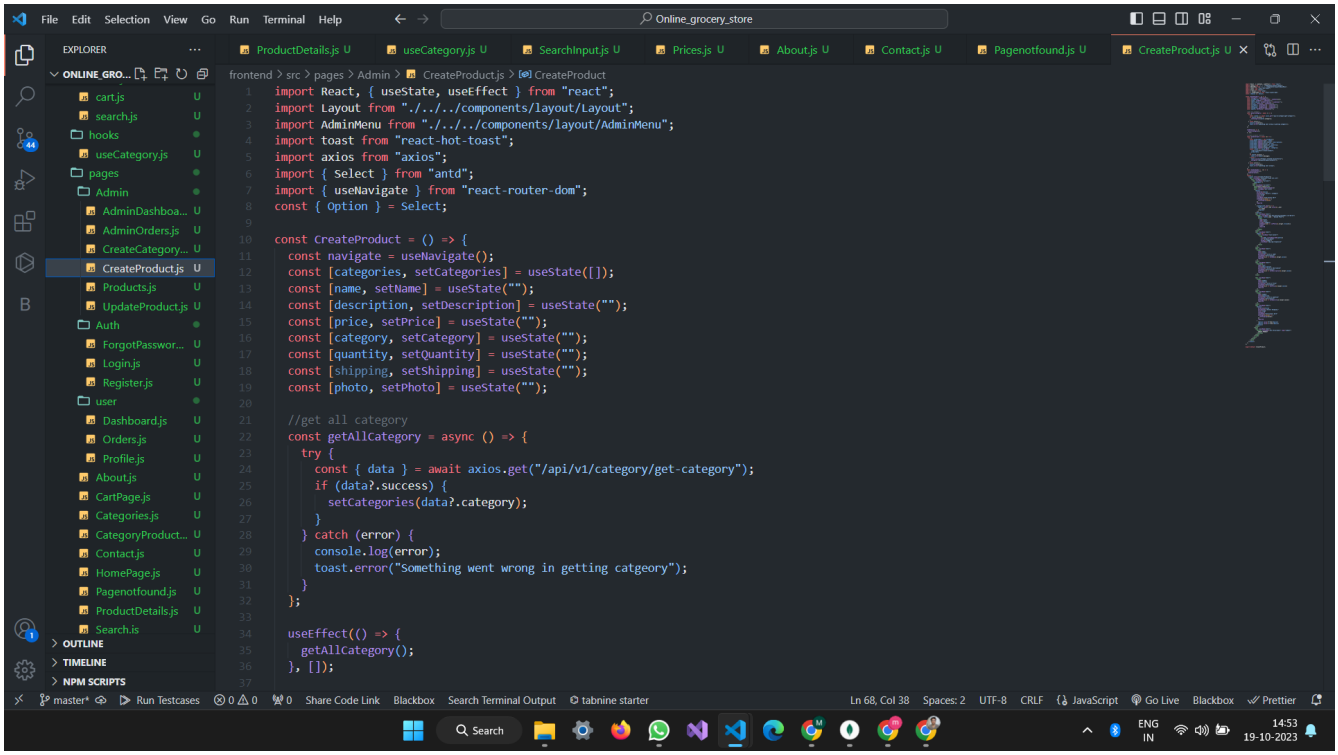
Admin functionalities -

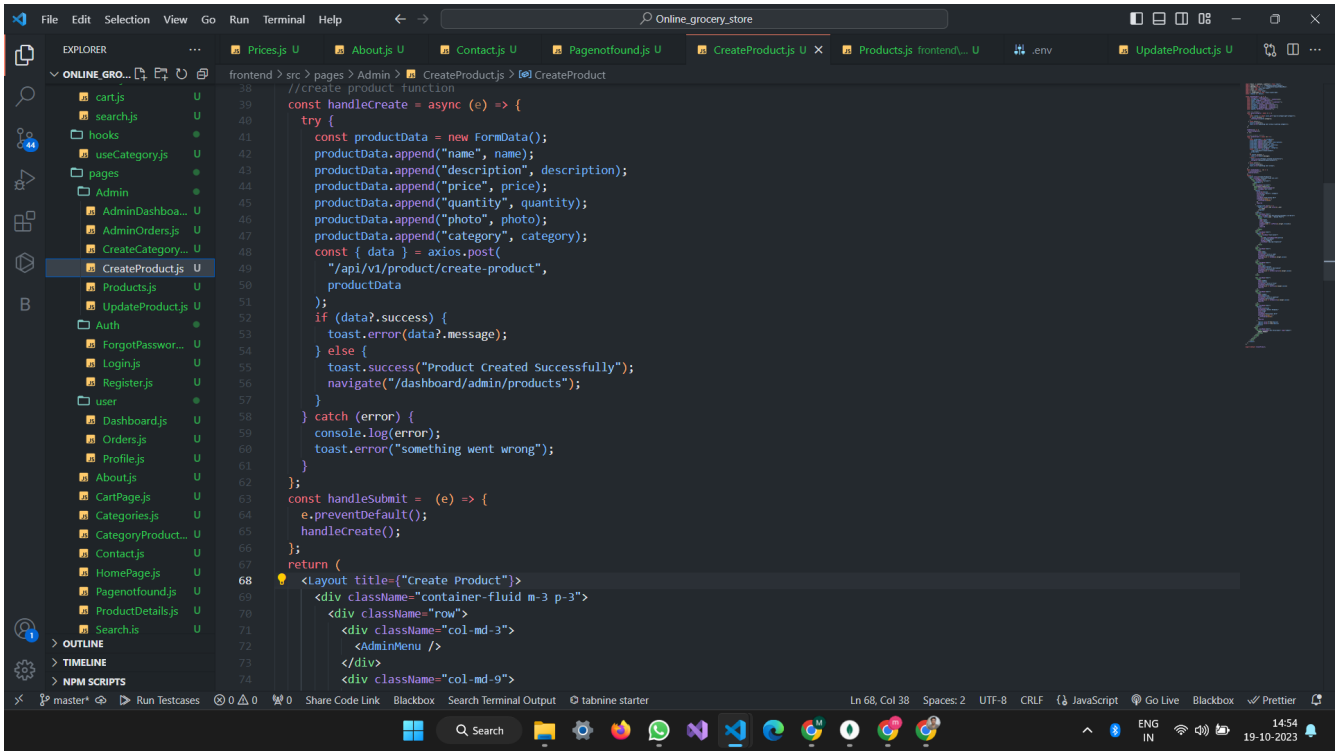
Manage Category





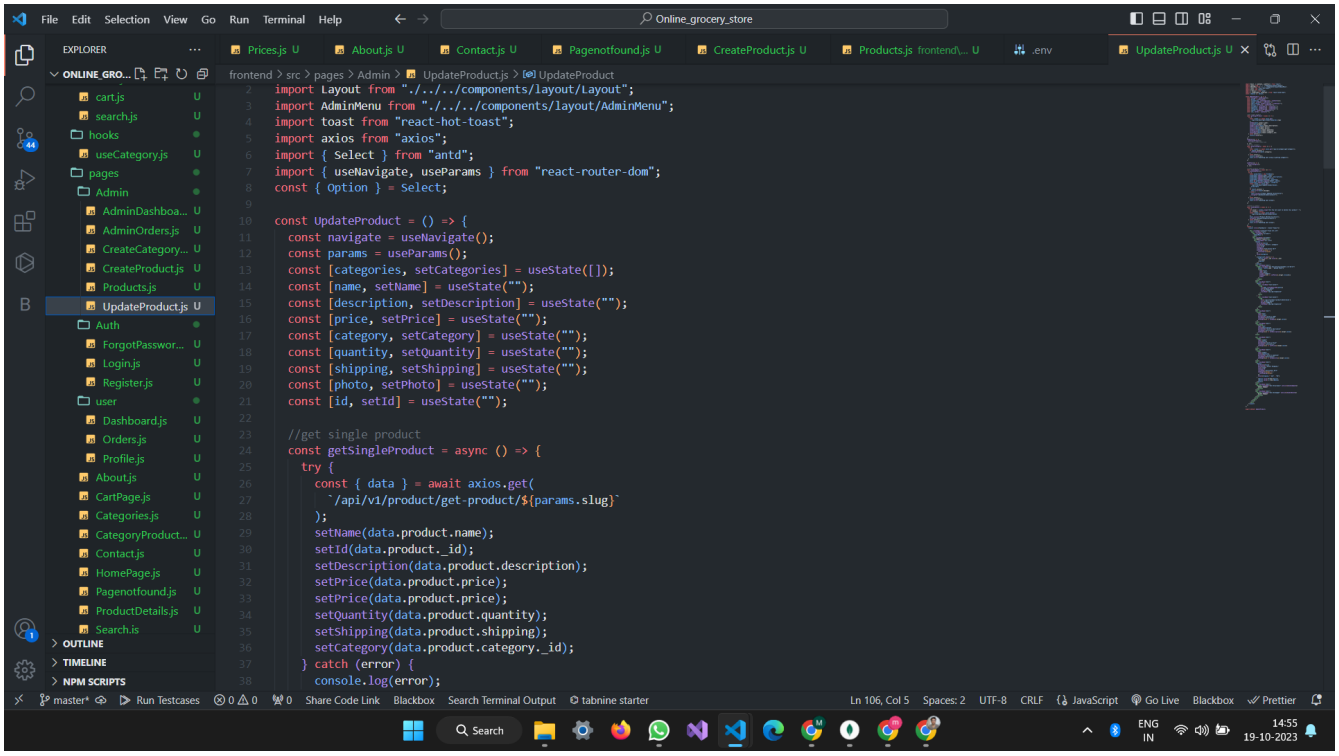
Manage Products

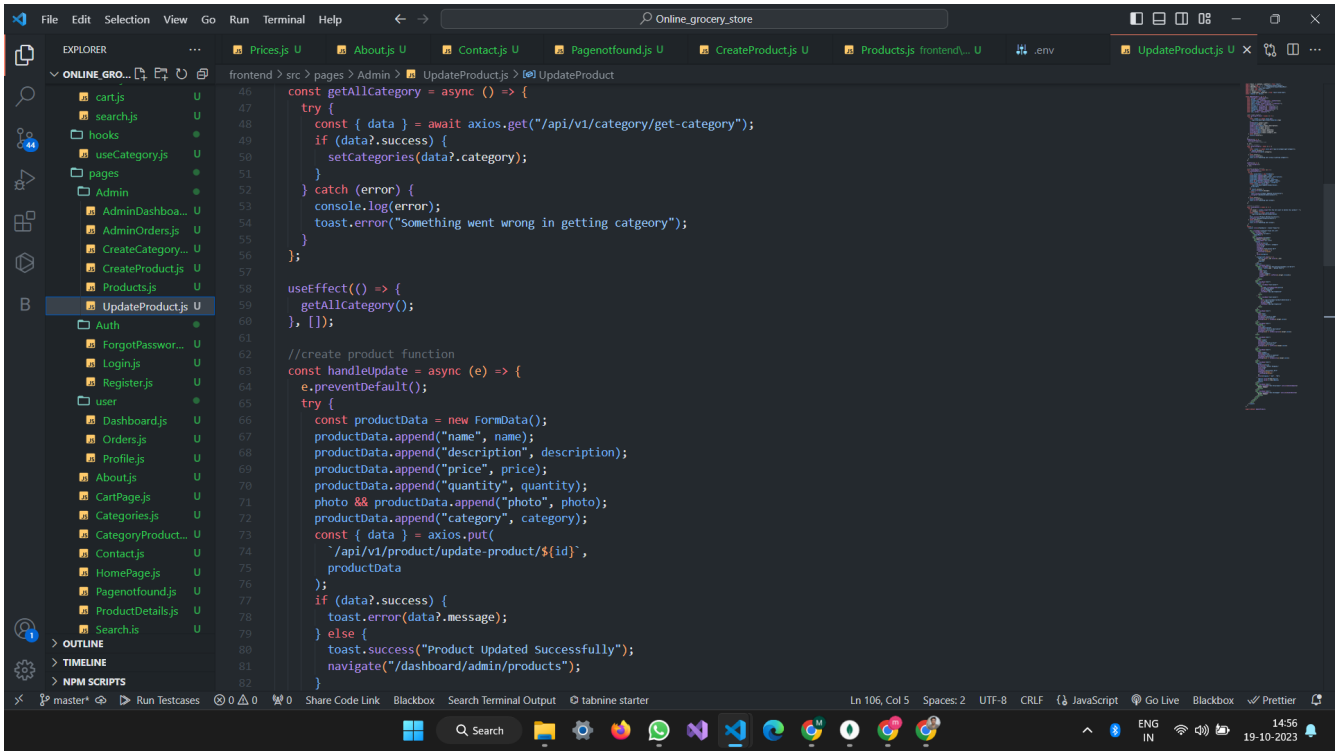



File Edit Selection View Go Run Terminal Help
Online_grocery_store
EXPLORER
Prices.js U
About.js U
Contact.js U
Pagenotfound.js U
CreateProduct.js U
Products.js frontend\... U
.env
UpdateProduct.js U
frontend > src > pages > Admin > CreateProduct.js > CreateProduct
//create product function
const handleCreate = async (e) => {
try {
const productData = new FormData();
productData.append("name", name);
productData.append("description", description);
productData.append("price", price);
productData.append("quantity", quantity);
productData.append("photo", photo);
productData.append("category", category);
const { data } = axios.post(
"/api/v1/product/create-product",
productData
);
if (data?.success) {
toast.error(data?.message);
} else {
toast.success("Product Created Successfully");
navigate("/dashboard/admin/products");
}
} catch (error) {
console.log(error);
toast.error("something went wrong");
}
};
const handleSubmit = (e) => {
e.preventDefault();
handleCreate();
};
return (
<Layout title={"Create Product"}>

<AdminMenu />

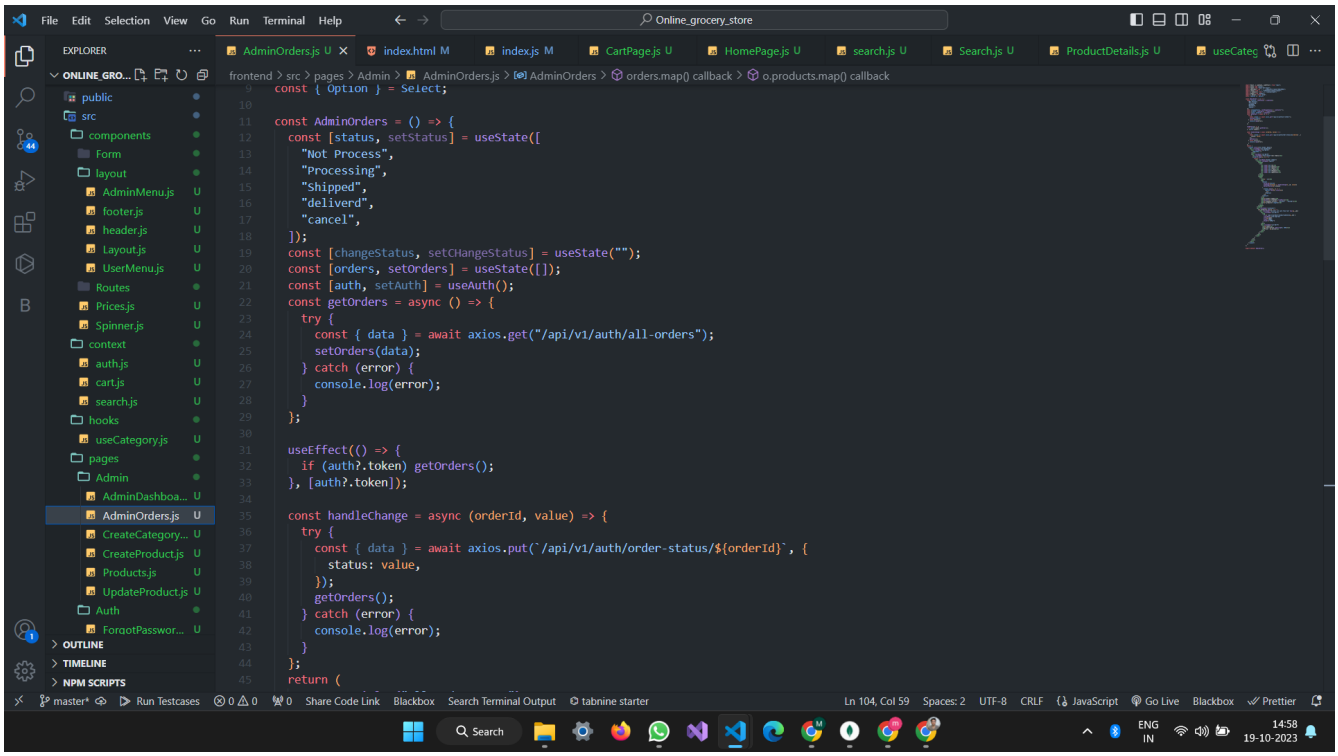
OUTLINE
TIMELINE
NPM SCRIPTS
master*
Run Testcases
Share Code Link
Blackbox
Search Terminal Output
tabnine starter
Ln 68, Col 38
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live
Blackbox
Prettier
Search
ENG
IN
14:54
19-10-2023


File Edit Selection View Go Run Terminal Help
Online_grocery_store
EXPLORER
Prices.js U
About.js U
Contact.js U
Pagenotfound.js U
CreateProduct.js U
Products.js frontend\... U
.env
UpdateProduct.js U
frontend > src > pages > Admin > UpdateProduct.js > UpdateProduct
import Layout from "./../../components/layout/Layout";
import AdminMenu from "./../../components/layout/AdminMenu";
import toast from "react-hot-toast";
import axios from "axios";
import { Select } from "antd";
import { useNavigate, useParams } from "react-router-dom";
const { Option } = Select;
const UpdateProduct = () => {
const navigate = useNavigate();
const params = useParams();
const [categories, setCategories] = useState([]);
const [name, setName] = useState("");
const [description, setDescription] = useState("");
const [price, setPrice] = useState("");
const [category, setCategory] = useState("");
const [quantity, setQuantity] = useState("");
const [shipping, setShipping] = useState("");
const [photo, setPhoto] = useState("");
const [id, setId] = useState("");
//get single product
const getSingleProduct = async () => {
try {
const { data } = await axios.get(
`/api/v1/product/get-product/${params.slug}`
);
setName(data.product.name);
setId(data.product._id);
setDescription(data.product.description);
setPrice(data.product.price);
setPrice(data.product.price);
setQuantity(data.product.quantity);
setShipping(data.product.shipping);
setCategory(data.product.category._id);
} catch (error) {
console.log(error);
OUTLINE
TIMELINE
NPM SCRIPTS
master*
Run Testcases
Share Code Link
Blackbox
Search Terminal Output
tabnine starter
Ln 106, Col 5
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live
Blackbox
Prettier
Search
ENG
IN
14:55
19-10-2023



```
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);
    toast.error("Something went wrong in getting catgeory");
  }
};

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

//create product function
const handleUpdate = async (e) => {
  e.preventDefault();
  try {
    const productData = new FormData();
    productData.append("name", name);
    productData.append("description", description);
    productData.append("price", price);
    productData.append("quantity", quantity);
    photo && productData.append("photo", photo);
    productData.append("category", category);
    const { data } = axios.put(
      `/api/v1/product/update-product/${id}`,
      productData
    );
    if (data?.success) {
      toast.error(data?.message);
    } else {
      toast.success("Product Updated Successfully");
      navigate("/dashboard/admin/products");
    }
```

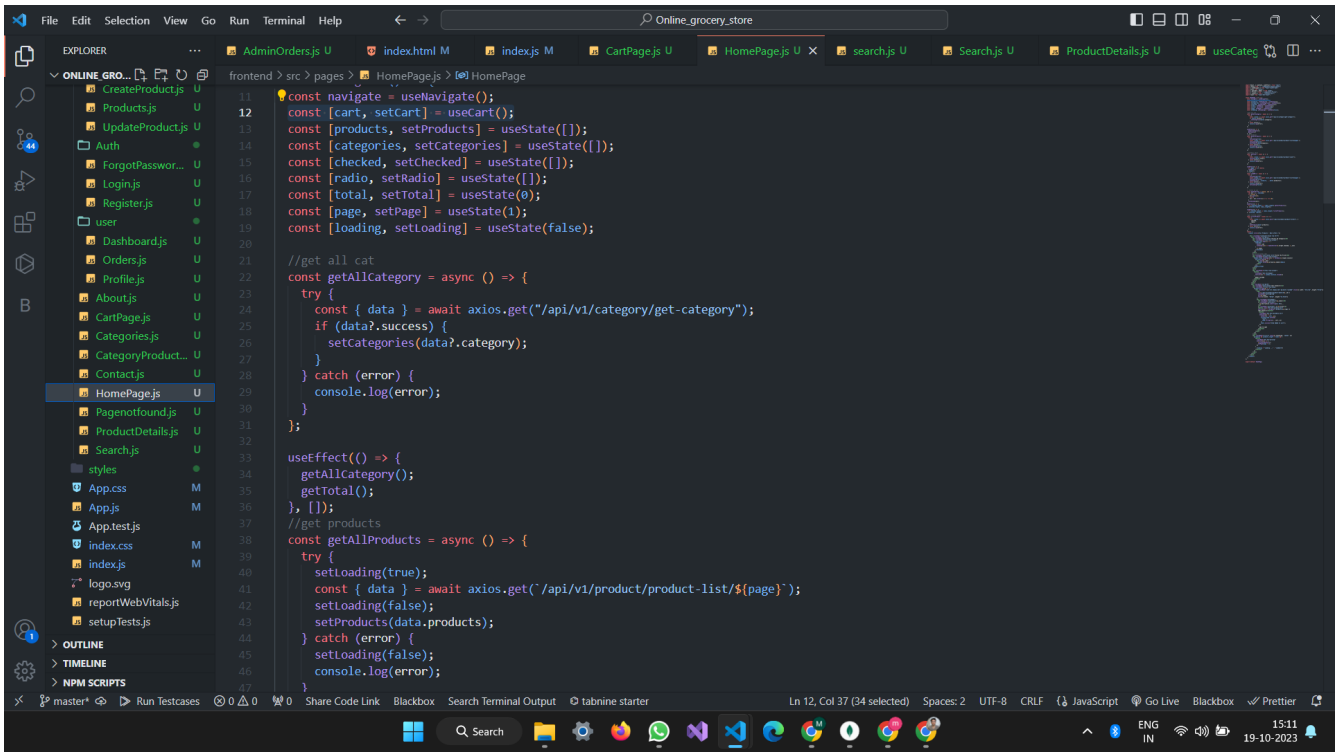
Manage Order



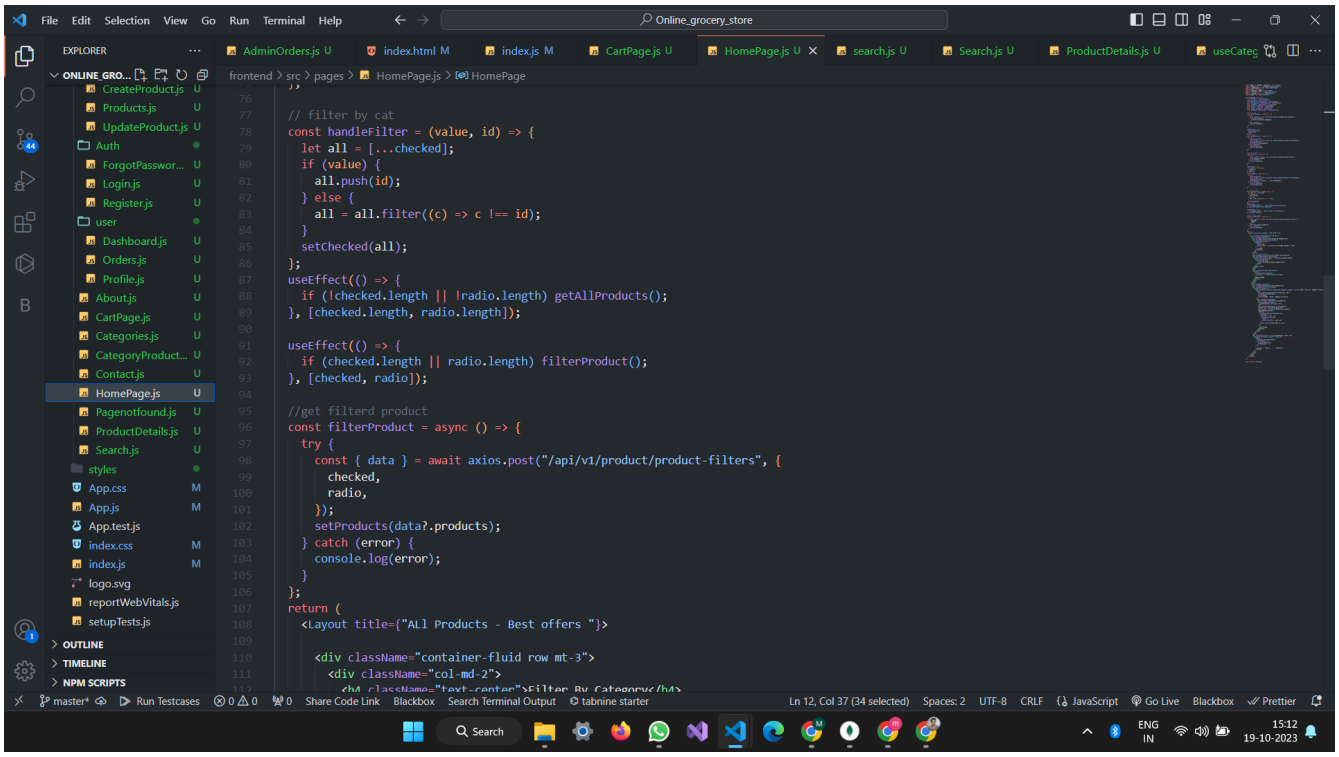
```
const { Option } = Select;

const AdminOrders = () => {
  const [status, setStatus] = useState([
    "Not Process",
    "Processing",
    "Shipped",
    "deliverd",
    "cancel",
  ]);
  const [changeStatus, setCHangeStatus] = useState("");
  const [orders, setOrders] = useState([]);
  const [auth, setAuth] = useAuth();
  const getOrders = async () => {
    try {
      const { data } = await axios.get("/api/v1/auth/all-orders");
      setOrders(data);
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {
    if (auth?.token) getOrders();
  }, [auth?.token]);

  const handleChange = async (orderId, value) => {
    try {
      const { data } = await axios.put(`/api/v1/auth/order-status/${orderId}`, {
        status: value,
      });
      getOrders();
    } catch (error) {
      console.log(error);
    }
  };
  return (
```

User functionalities :-

Filter The Product

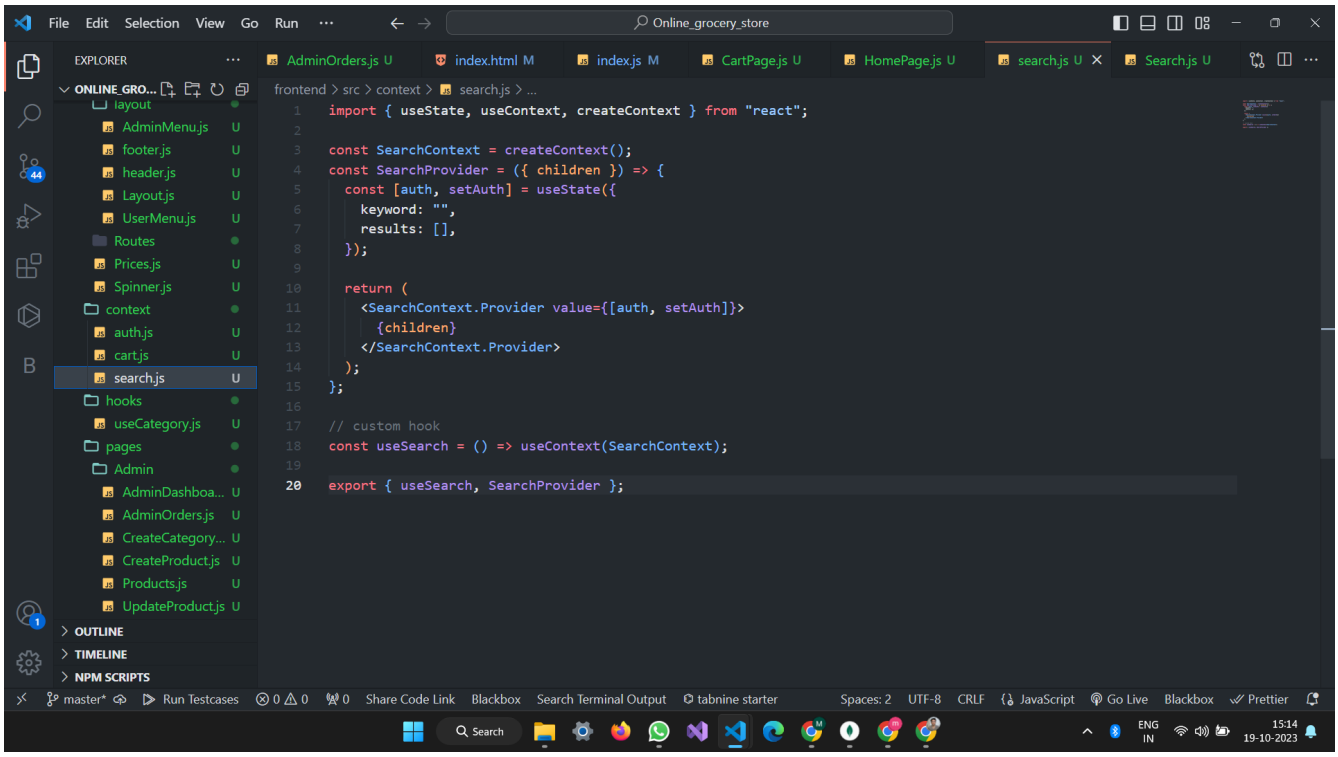


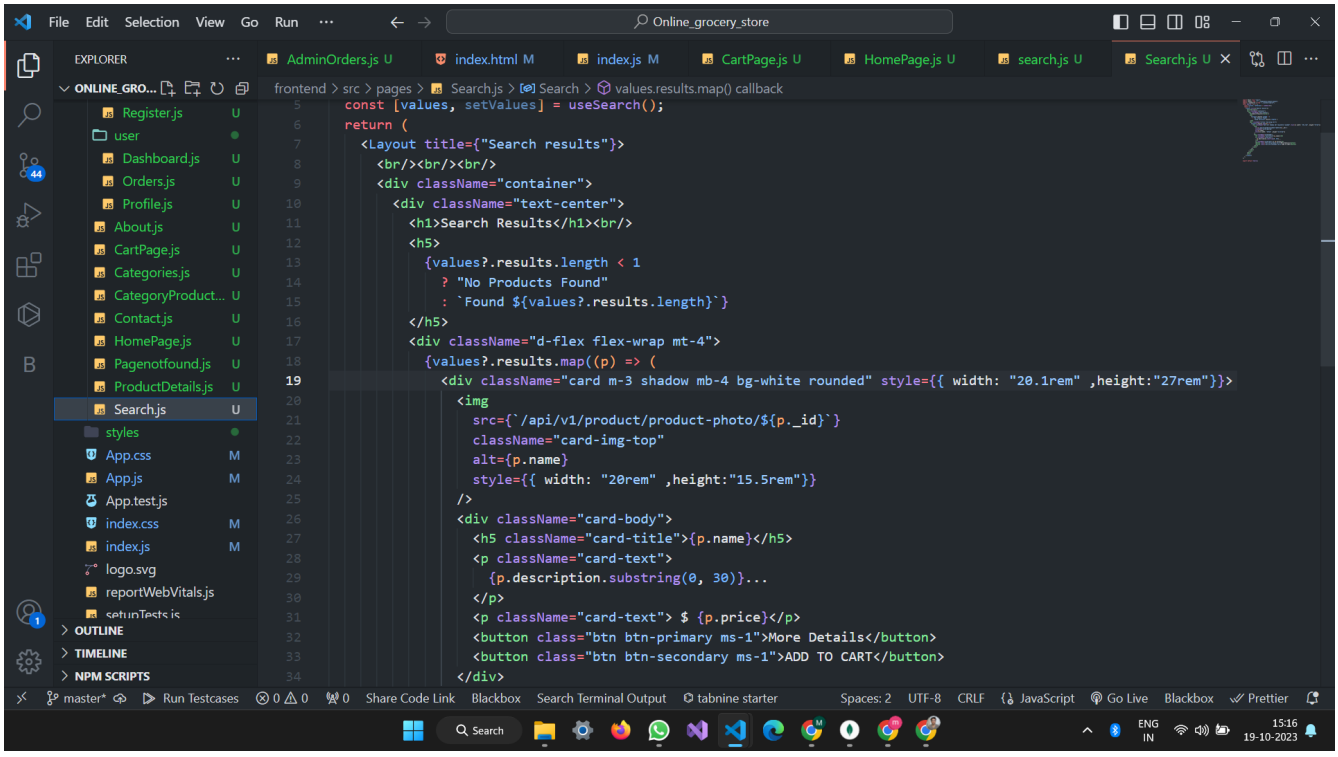
```
const navigate = useNavigate();
const [cart, setCart] = useCart();
const [products, setProducts] = useState([]);
const [categories, setCategories] = useState([]);
const [checked, setChecked] = useState([]);
const [radio, setRadio] = useState([]);
const [total, setTotal] = useState(0);
const [page, setPage] = useState(1);
const [loading, setLoading] = useState(false);

//get all cat
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);
  }
};
```



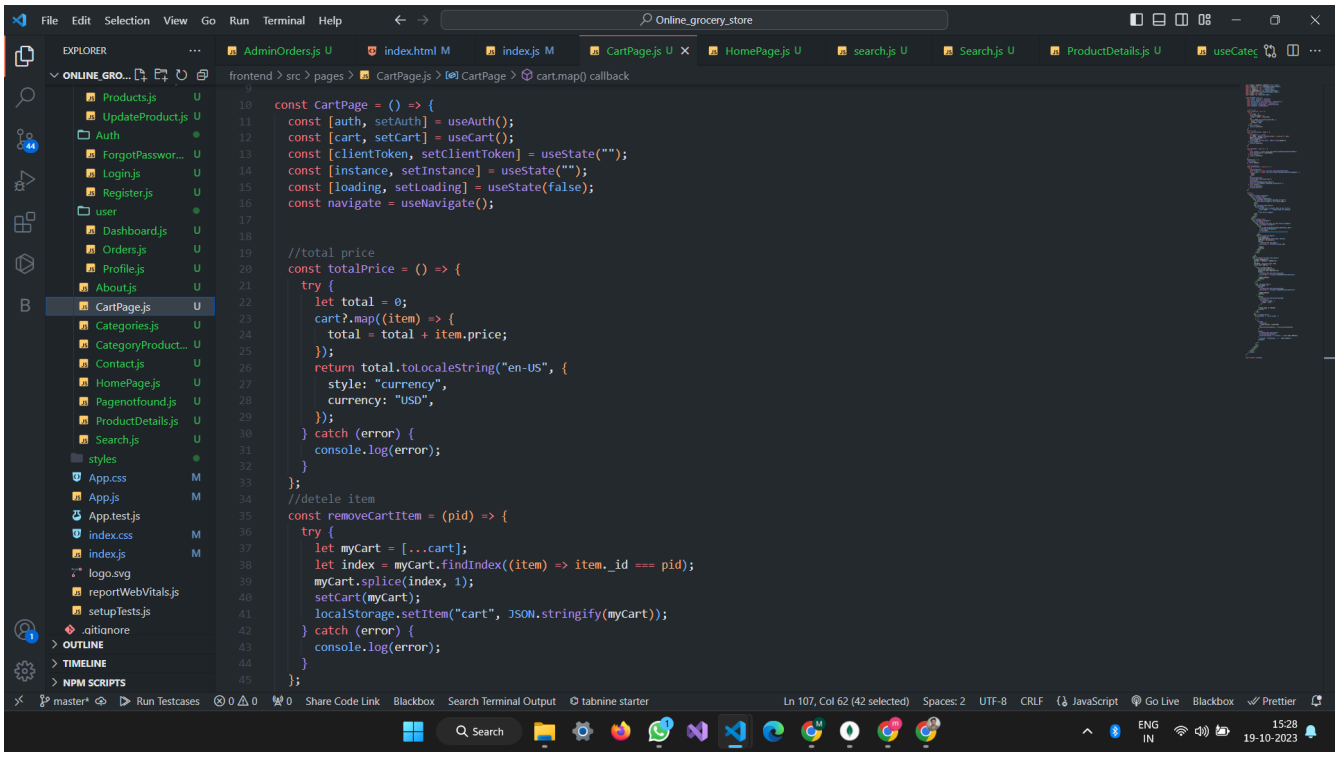
```
// filter by cat
const handleFilter = (value, id) => {
  let all = [...checked];
  if (value) {
    all.push(id);
  } else {
    all = all.filter((c) => c !== id);
  }
  setChecked(all);
};
useEffect(() => {
  if (!checked.length || !radio.length) getAllProducts();
}, [checked.length, radio.length]);

useEffect(() => {
  if (checked.length || radio.length) filterProduct();
}, [checked, radio]);

//get filterd 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 "}>

    <div className="container-fluid row mt-3">
      <div className="col-md-2">
```

Search Product

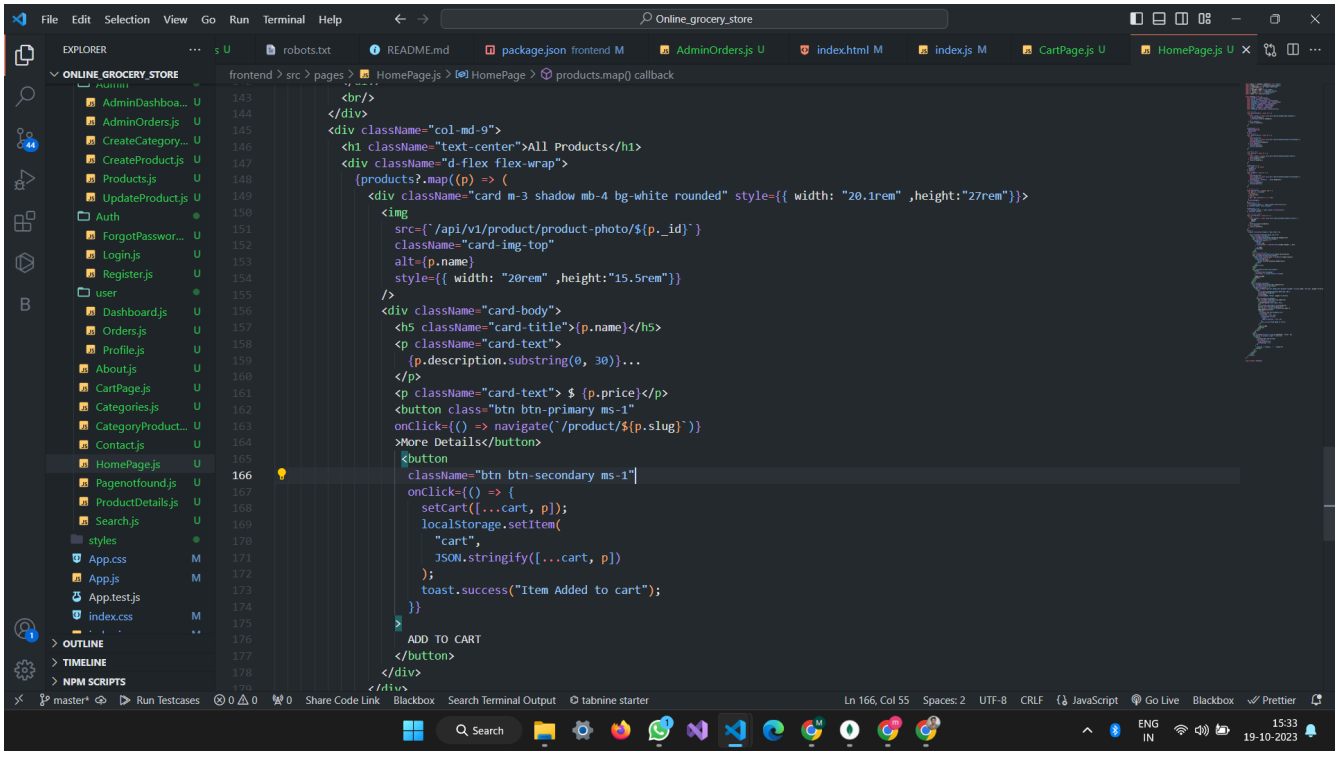




Manage Cart

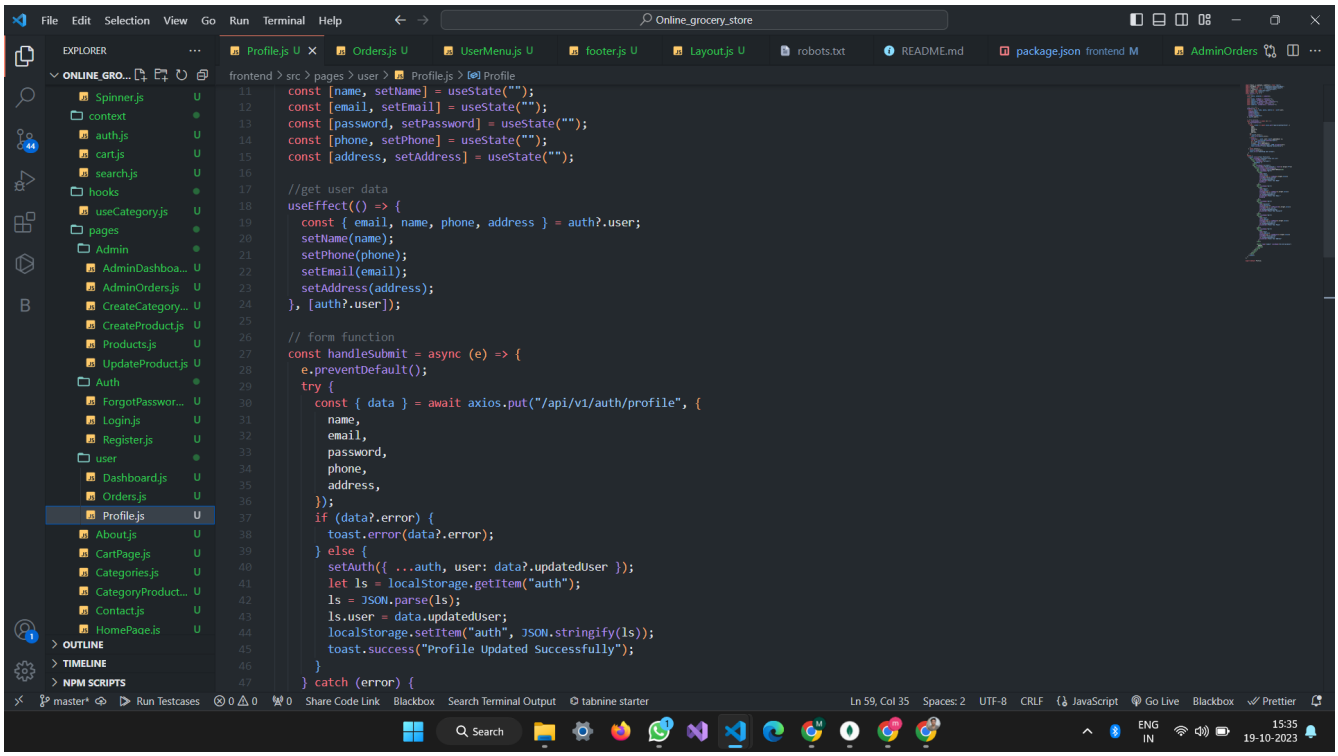


```
const CartPage = () => {
  const [auth, setAuth] = useAuth();
  const [cart, setCart] = useCart();
  const [clientToken, setClientToken] = useState("");
  const [instance, setInstance] = useState("");
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();


  //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);
    }
  };
  //detele 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);
    }
  };
```



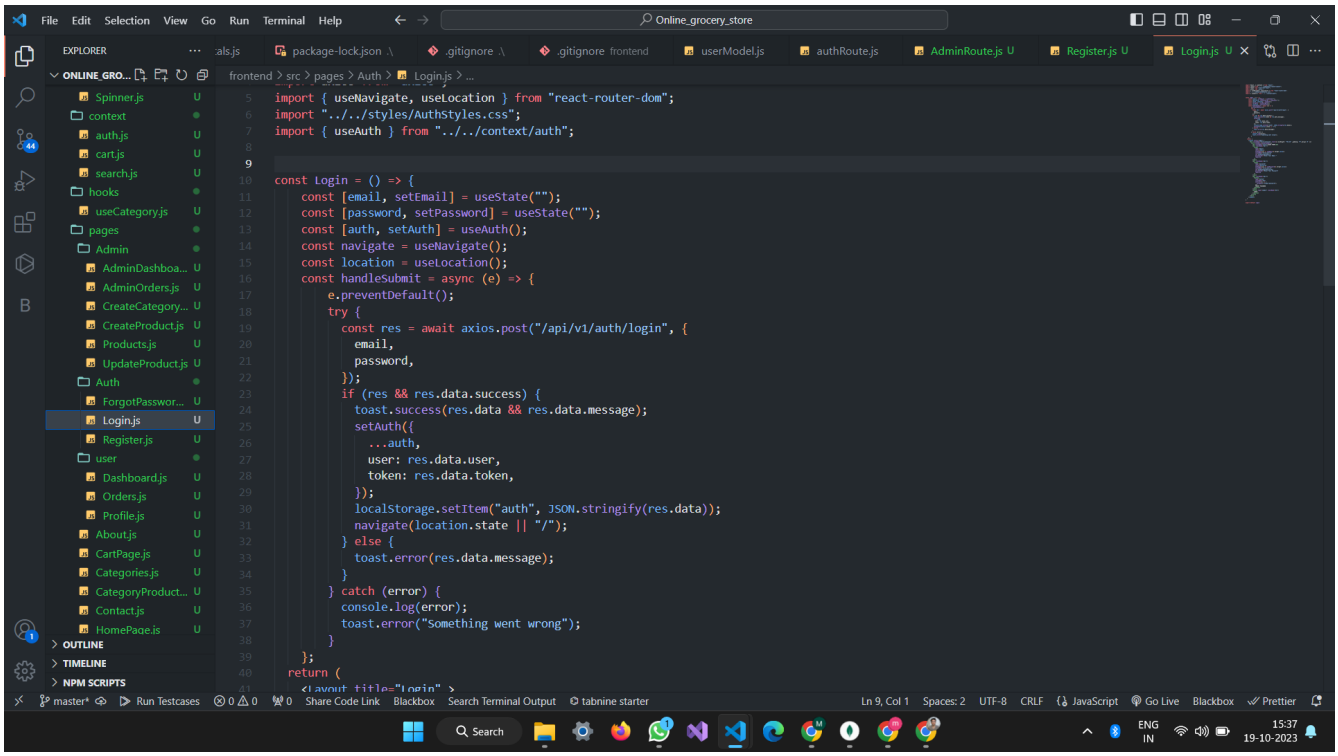
```
            <br/>
          </div>
          <div className="col-md-9">
            <h1 classNme="text-center">All Products</h1>
            <div className="d-flex flex-wrap">
              {products?.map((p) => (
                <div className="card m-3 shadow mb-4 bg-white rounded" style={{ width: "20.1rem" ,height:"27rem"}}>
                  <img
                    src={`/api/v1/product/product-photo/${p._id}`}
                    className="card-img-top"
                    alt={p.name}
                    style={{ width: "20rem" ,height:"15.5rem"}}
                  />
                  <div className="card-body">
                    <h5 className="card-title">{p.name}</h5>
                    <p className="card-text">
                      {p.description.substring(0, 30)}...
                    </p>
                    <p className="card-text"> $ {p.price}</p>
                    <button class="btn btn-primary ms-1"
                    onClick={() => navigate(`/product/${p.slug}`)}
                    >More Details</button>
                    <button
                      className="btn btn-secondary ms-1"
                      onClick={() => {
                        setCart([...cart, p]);
                        localStorage.setItem(
                          "cart",
                          JSON.stringify([...cart, p])
                        );
                        toast.success("Item Added to cart");
                      }}
                    >
                      ADD TO CART
                    </button>
                  </div>
```

Manage Profile

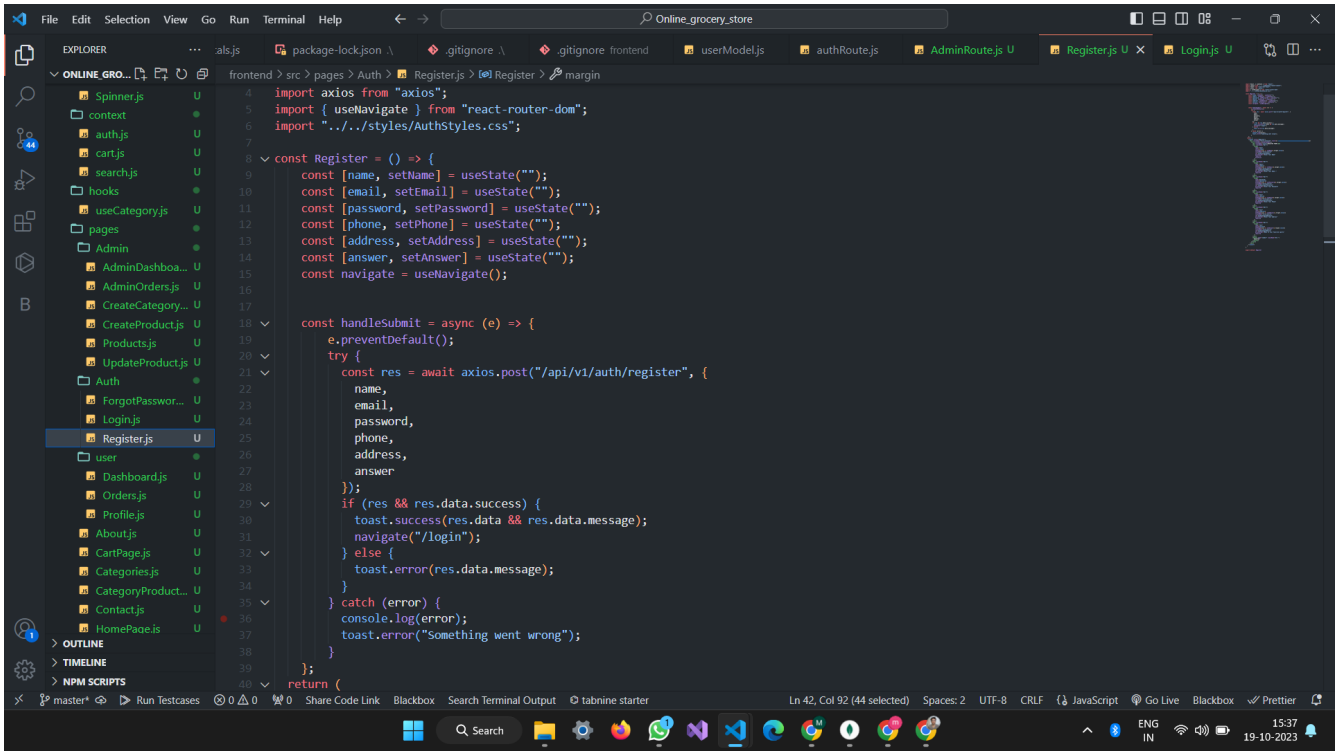


```
const [name, setName] = useState("");
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const [phone, setPhone] = useState("");
const [address, setAddress] = useState("");

//get user data
useEffect(() => {
  const { email, name, phone, address } = auth?.user;
  setName(name);
  setPhone(phone);
  setEmail(email);
  setAddress(address);
}, [auth?.user]);

// form function
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const { data } = await axios.put("/api/v1/auth/profile", {
      name,
      email,
      password,
      phone,
      address,
    });
    if (data?.error) {
      toast.error(data?.error);
    } else {
      setAuth({ ...auth, user: data?.updatedUser });
      let ls = localStorage.getItem("auth");
      ls = JSON.parse(ls);
      ls.user = data.updatedUser;
      localStorage.setItem("auth", JSON.stringify(ls));
      toast.success("Profile Updated Successfully");
    }
  } catch (error) {
```

Login and Register

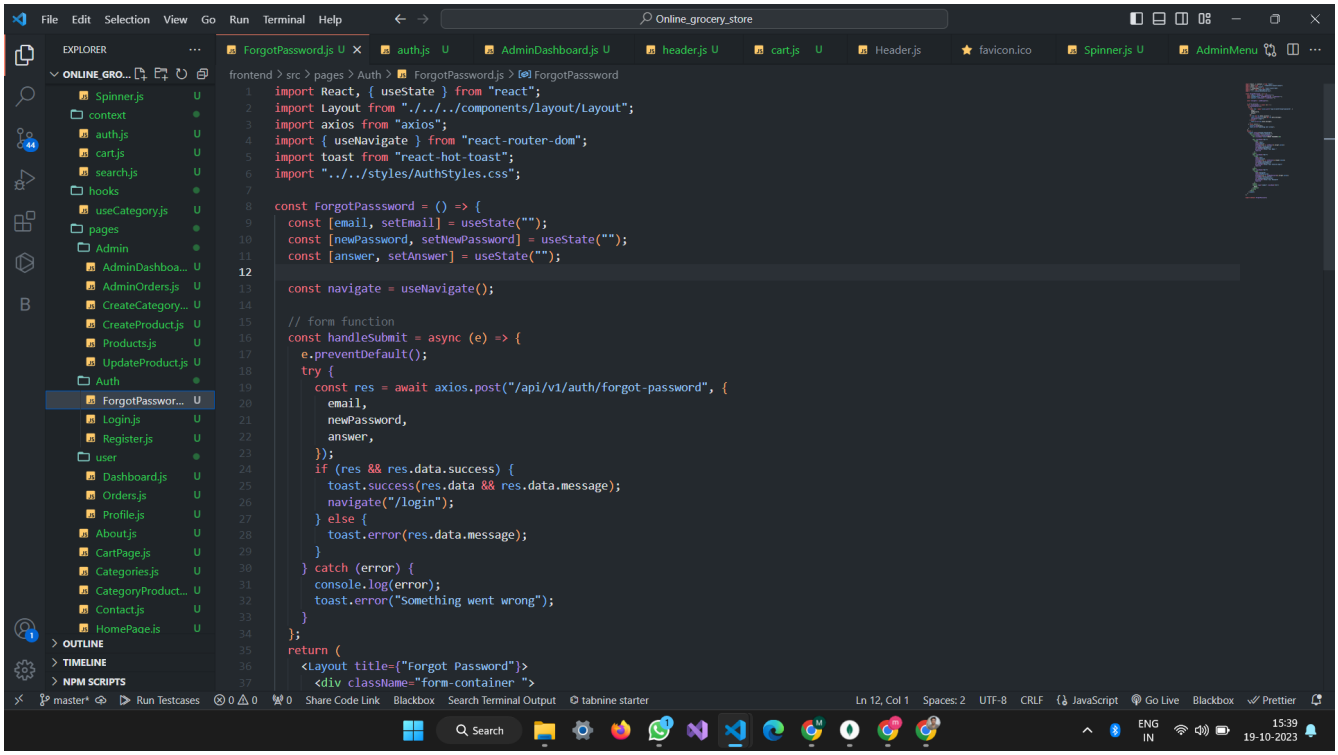


```
import { useNavigate, useLocation } from "react-router-dom";
import "../../styles/AuthStyles.css";
import { useAuth } from "../../context/auth";


const Login = () => {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [auth, setAuth] = useAuth();
    const navigate = useNavigate();
    const location = useLocation();
    const handleSubmit = async (e) => {
        e.preventDefault();
        try {
          const res = await axios.post("/api/v1/auth/login", {
            email,
            password,
          });
          if (res && res.data.success) {
            toast.success(res.data && res.data.message);
            setAuth({
              ...auth,
              user: res.data.user,
              token: res.data.token,
            });
            localStorage.setItem("auth", JSON.stringify(res.data));
            navigate(location.state || "/");
          } else {
            toast.error(res.data.message);
          }
        } catch (error) {
          console.log(error);
          toast.error("Something went wrong");
        }
    };
  return (
```



```
import axios from "axios";
import { useNavigate } from "react-router-dom";
import "../../styles/AuthStyles.css";

const Register = () => {
    const [name, setName] = useState("");
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [phone, setPhone] = useState("");
    const [address, setAddress] = useState("");
    const [answer, setAnswer] = useState("");
    const navigate = useNavigate();


    const handleSubmit = async (e) => {
        e.preventDefault();
        try {
          const res = await axios.post("/api/v1/auth/register", {
            name,
            email,
            password,
            phone,
            address,
            answer
          });
          if (res && res.data.success) {
            toast.success(res.data && res.data.message);
            navigate("/login");
          } else {
            toast.error(res.data.message);
          }
        } catch (error) {
          console.log(error);
          toast.error("Something went wrong");
        }
    };
    return (
```

Forgot Password



```
import React, { useState } from "react";
import Layout from "./../../components/layout/Layout";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import toast from "react-hot-toast";
import "../../styles/AuthStyles.css";

const ForgotPassword = () => {
  const [email, setEmail] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [answer, setAnswer] = useState("");

  const navigate = useNavigate();

  // form function
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await axios.post("/api/v1/auth/forgot-password", {
        email,
        newPassword,
        answer,
      });
      if (res && res.data.success) {
        toast.success(res.data && res.data.message);
        navigate("/login");
      } else {
        toast.error(res.data.message);
      }
    } catch (error) {
      console.log(error);
      toast.error("Something went wrong");
    }
  };
  return (
    <Layout title={"Forgot Password"}>
      <div className="form-container ">
```

9) Testing

i) Describe the testing framework and testing method used. List down the test cases used to test your system.

For testing our application, a mixed approach of integration testing and regression testing is used.

Integration testing : each main part of the application is tested after each small small function is tested first and then combine them and test that main part.

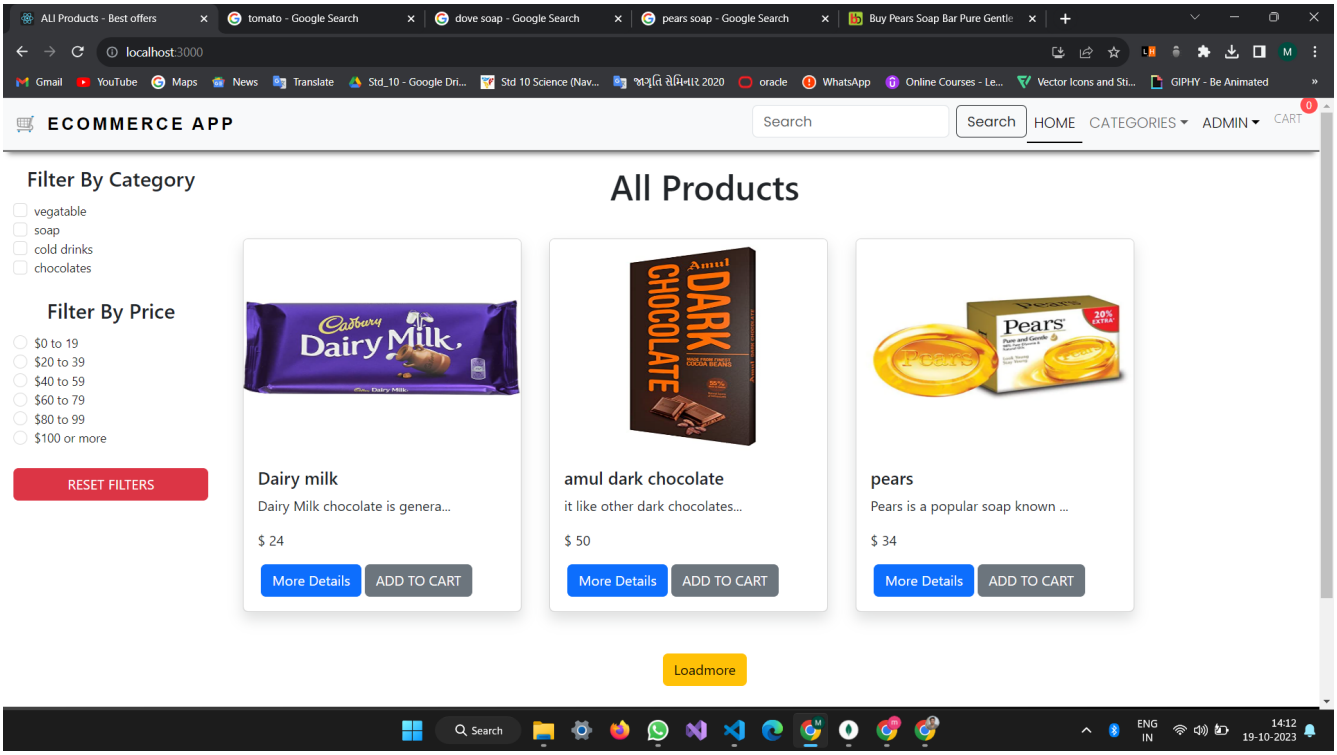
Regression testing : After the main part of the application is created, add them in the whole system, and then test the whole system to make sure the whole system is working fine after adding some main part.

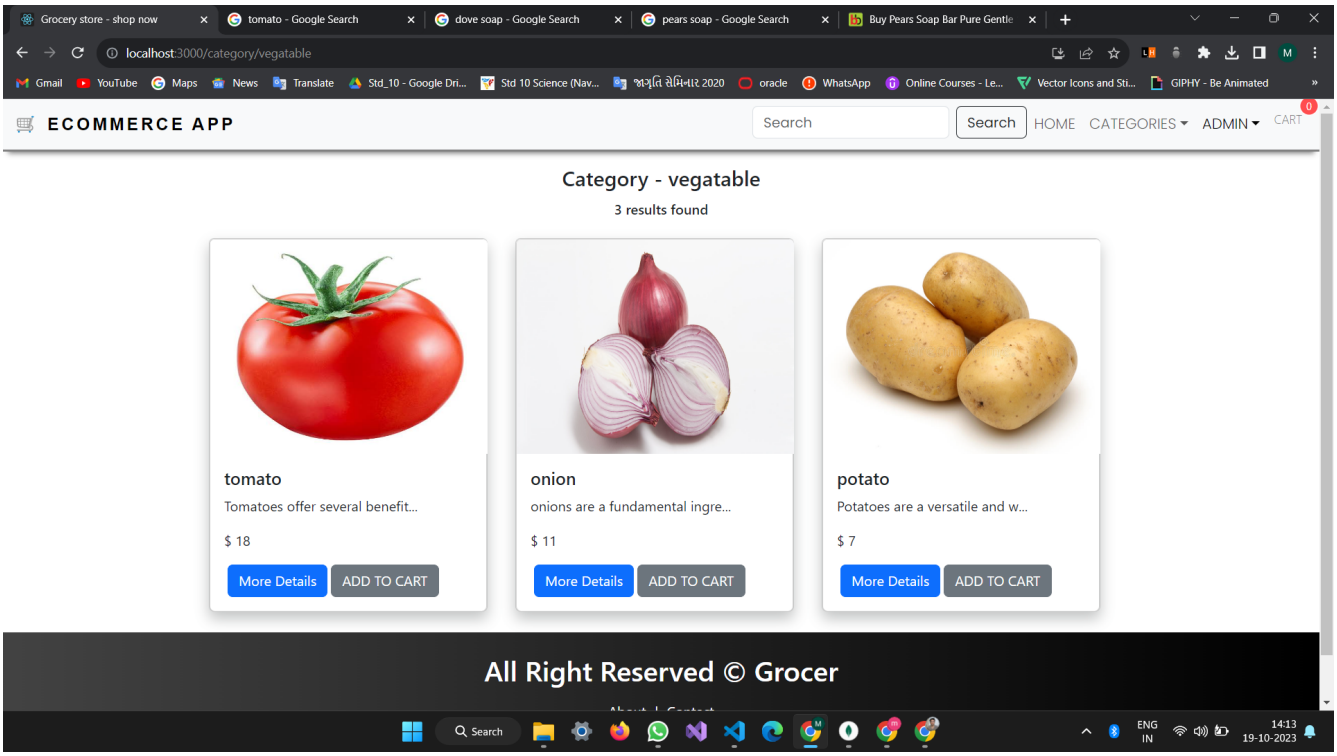
Manual Testing:- Manual testing is used to find and fix the bug in our application.

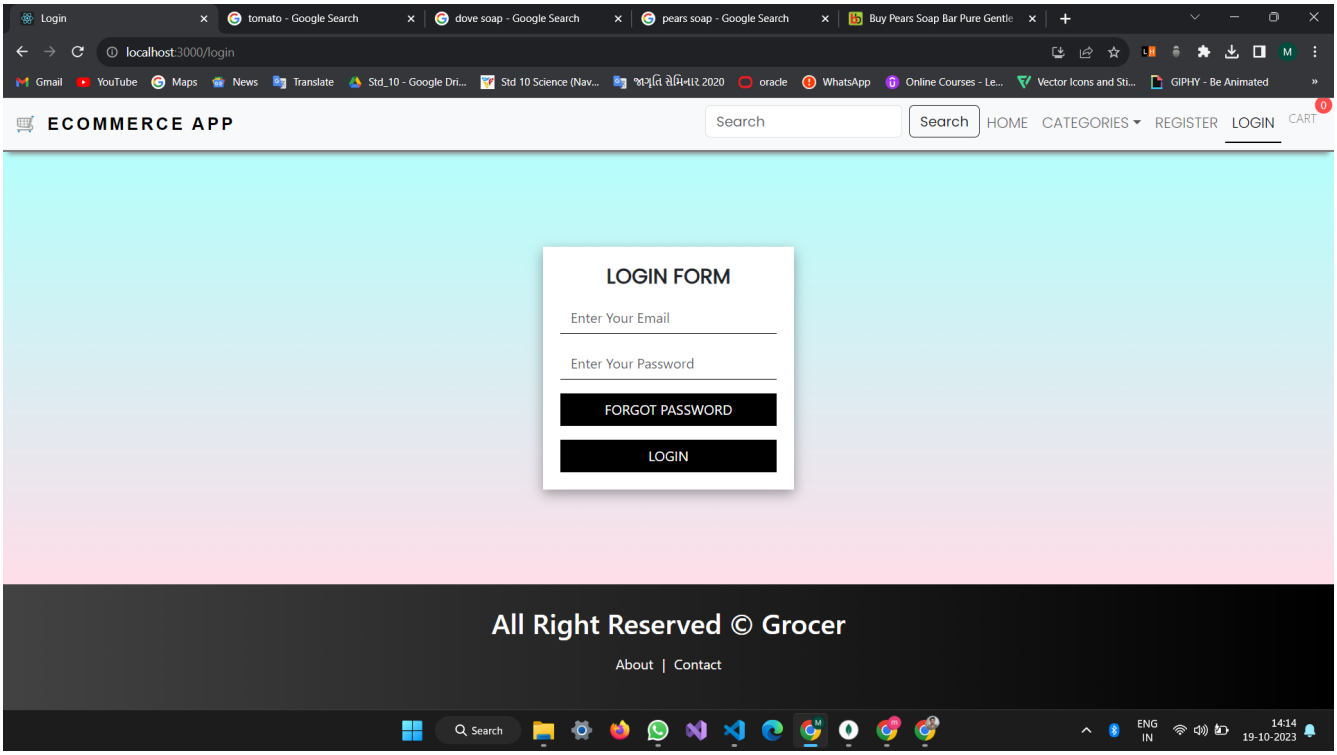
Sr No.	Test Scenario	Expected Result	Actual Result	Status
1	Login	The user or admin should be able to login	The user and the admin is able to login	Success
2	Register	The user should be able to register to the system	The user is able to register to the system	Success
3	Add Product to the cart	Product should be added into cart	Product added successfully	Success
4	Delete product from cart	Product Should be deleted from the cart	Product Deleted Successfully	Success
5	Search Product	Only that Product is to be displayed which are searched by the user	Product Is Displayed Successfully	Success

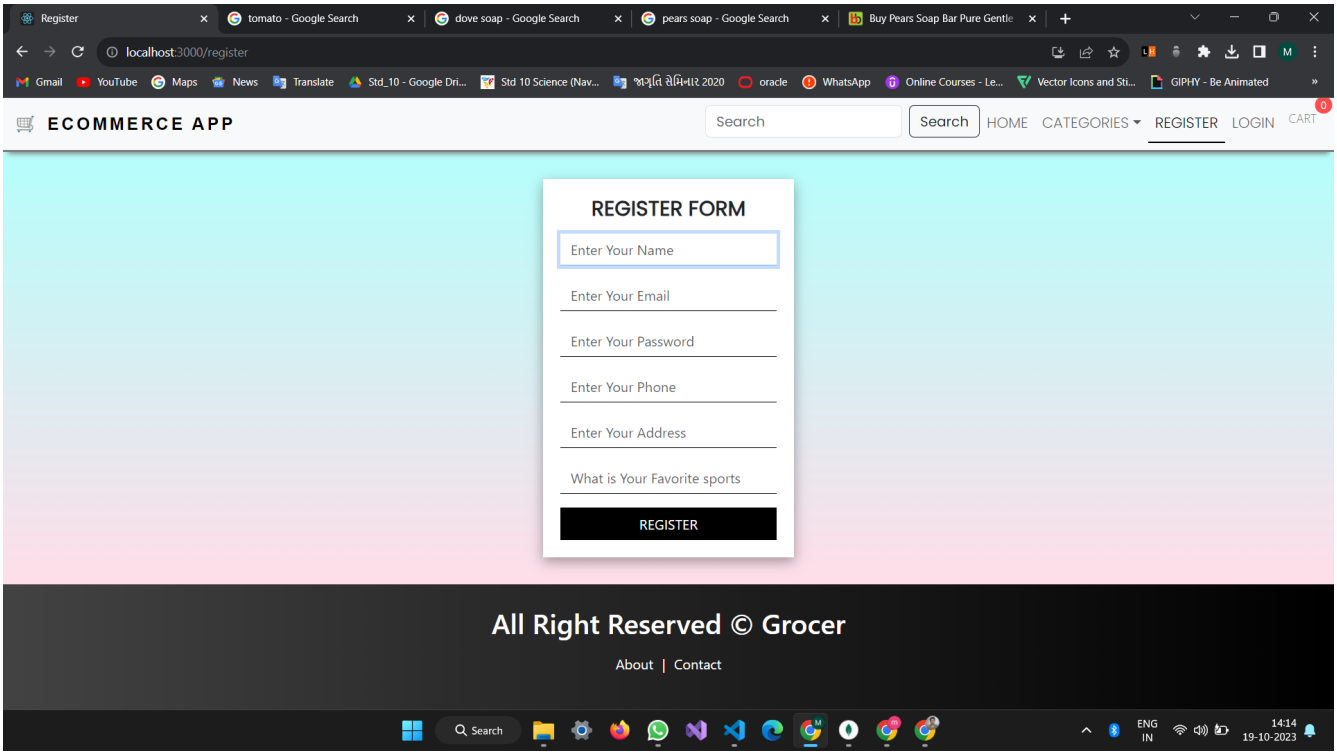
10) Screen-shots

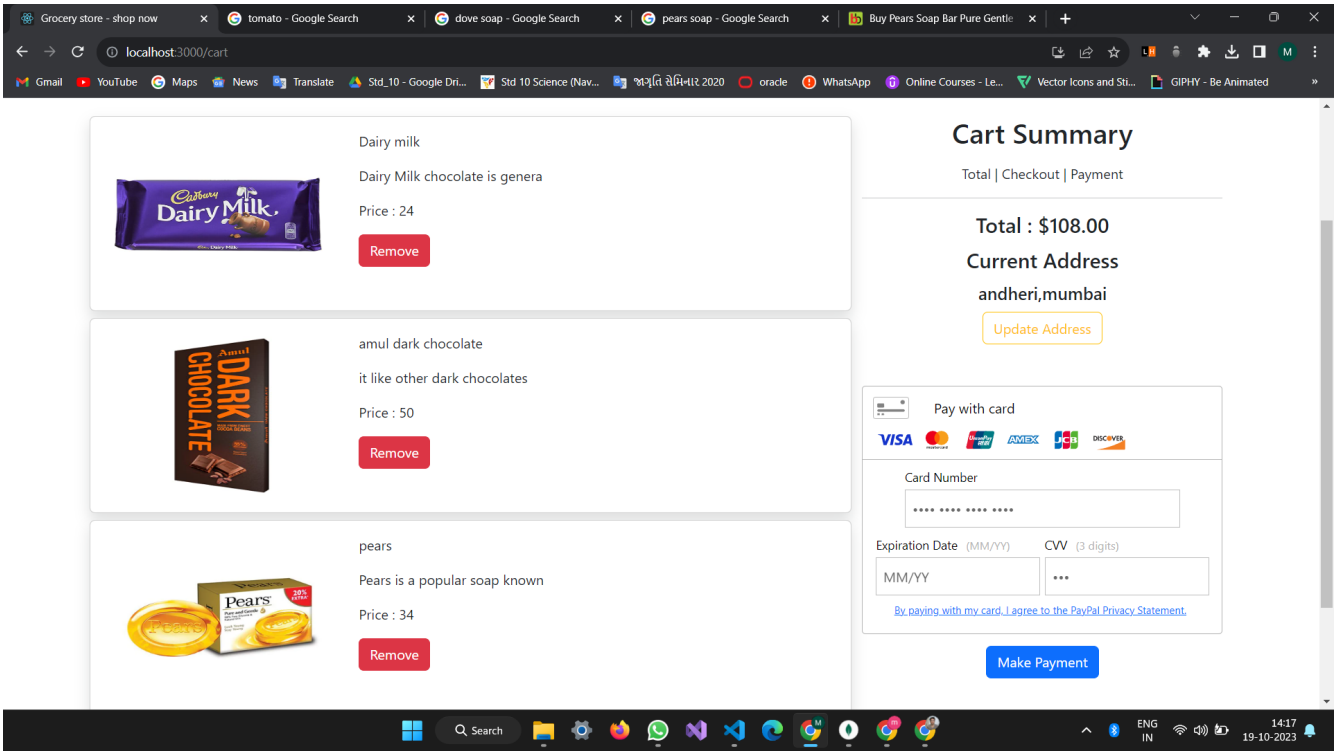
i) Important screen-shots of the system -

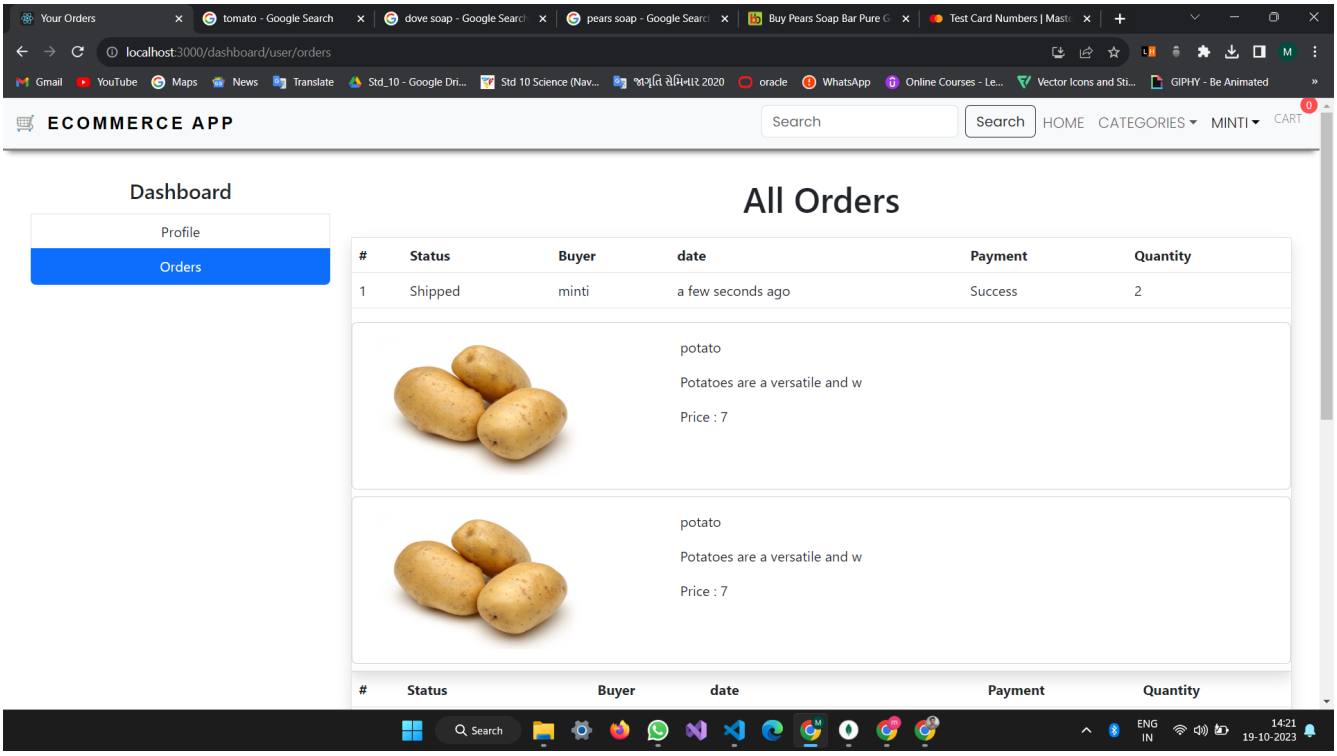


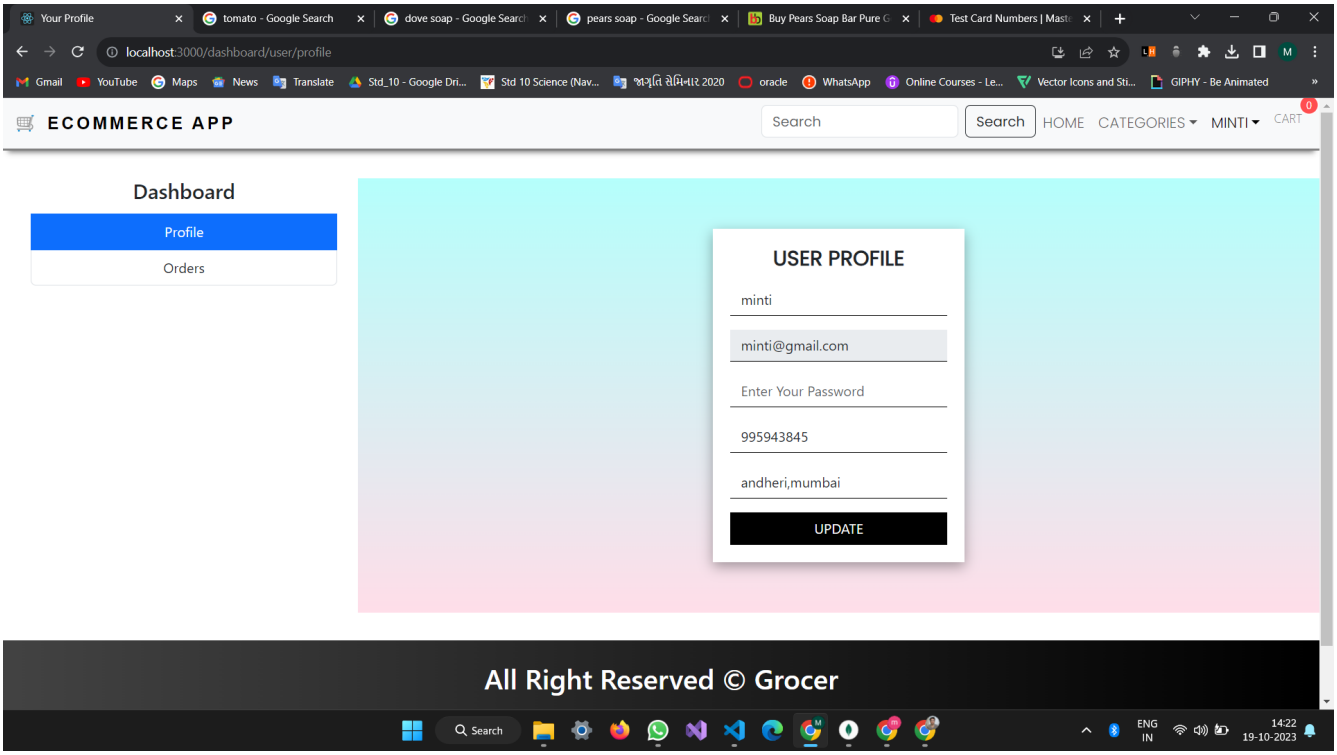


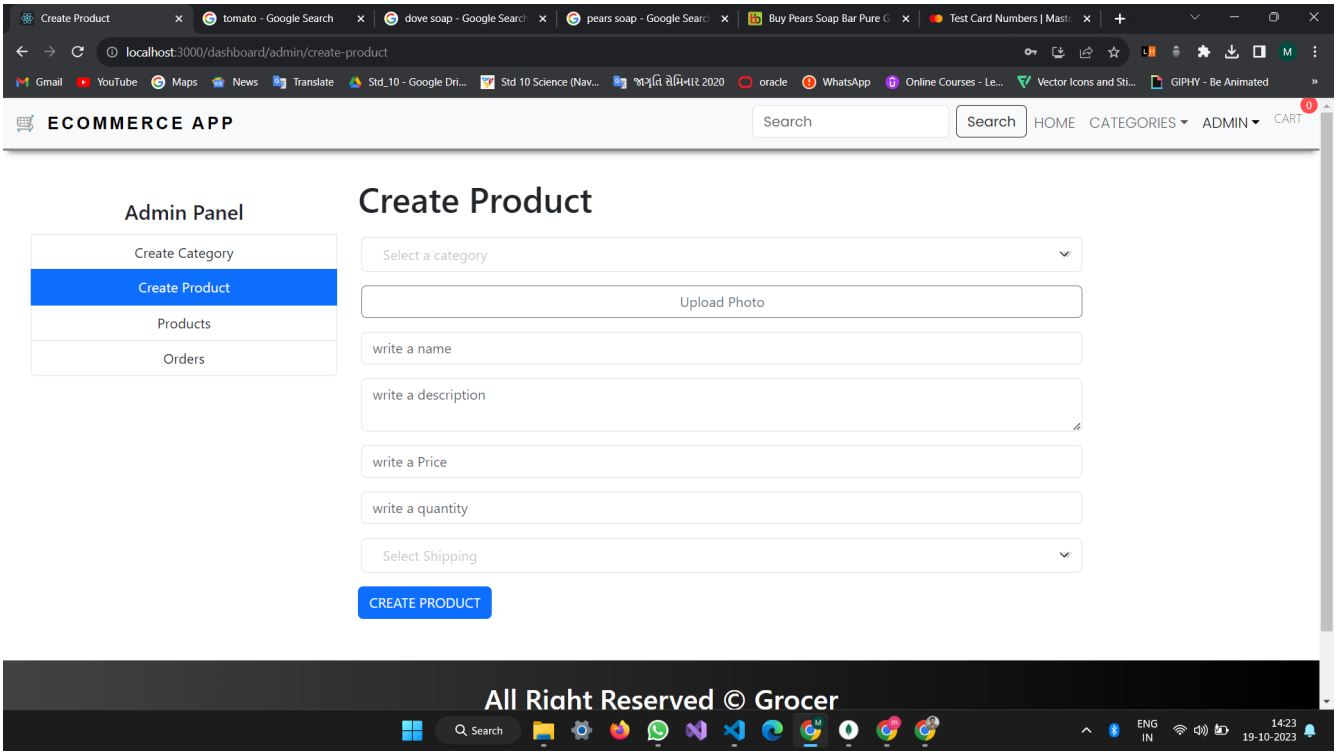

Login
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure Gentle
localhost:3000/login
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
REGISTER
LOGIN
CART
LOGIN FORM
Enter Your Email
Enter Your Password
FORGOT PASSWORD
LOGIN
All Right Reserved © Grocer
About | Contact

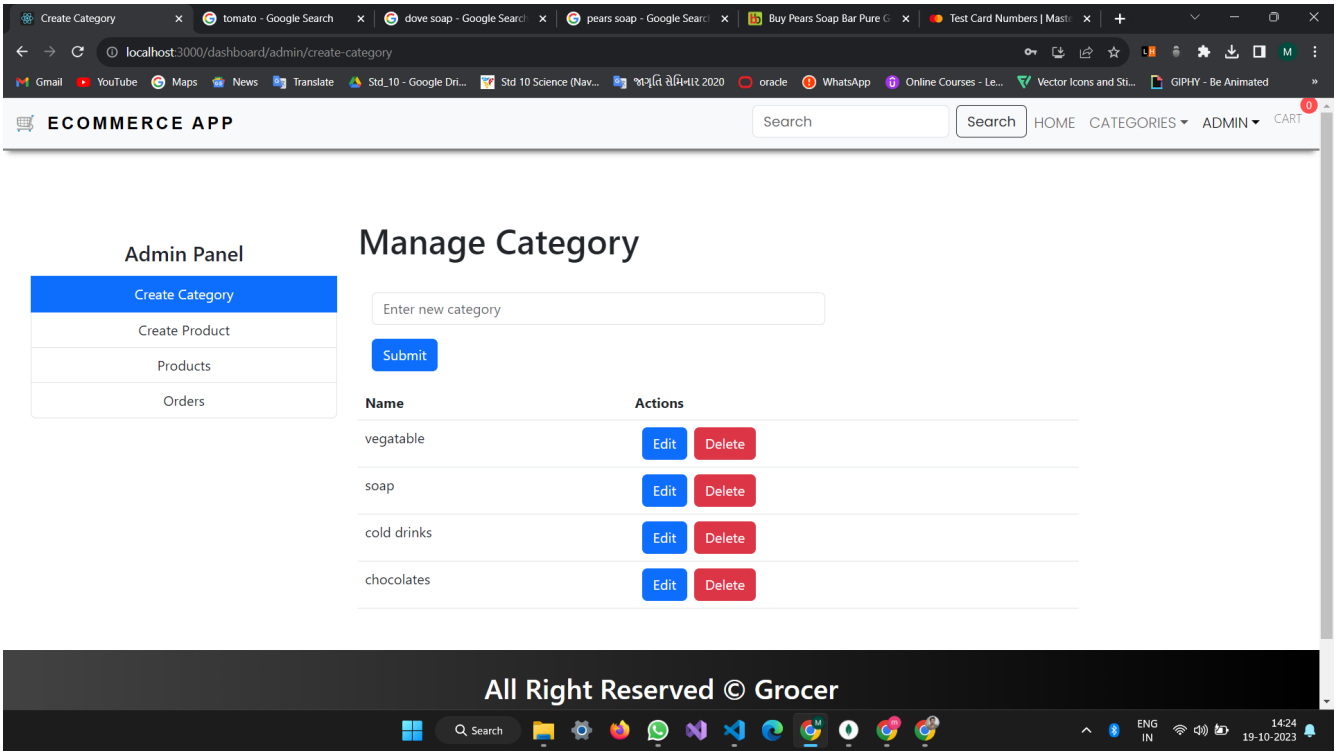

Register
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure Gentle
localhost:3000/register
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
REGISTER
LOGIN
CART
REGISTER FORM
Enter Your Name
Enter Your Email
Enter Your Password
Enter Your Phone
Enter Your Address
What is Your Favorite sports
REGISTER
All Right Reserved © Grocer
About | Contact

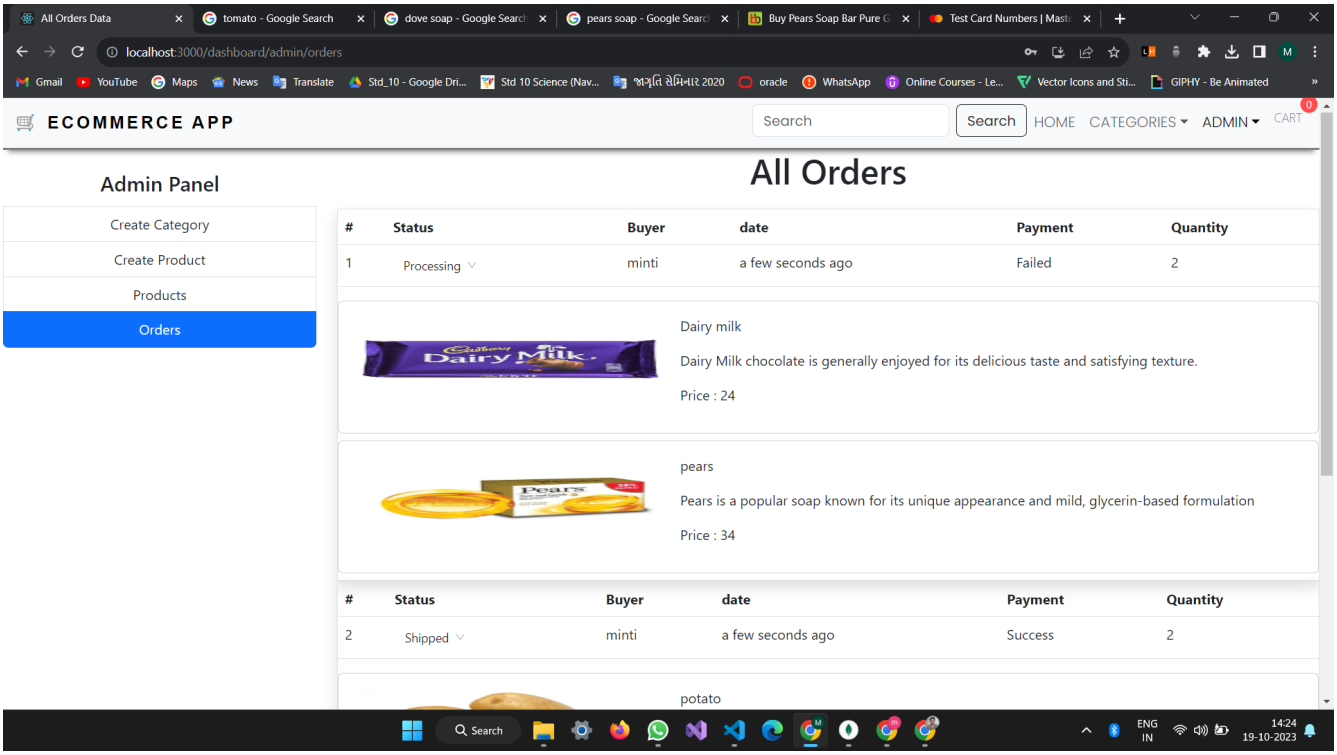

Grocery store - shop now
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure Gentle
localhost:3000/cart
Dairy milk
Dairy Milk chocolate is genera
Price : 24
Remove
amul dark chocolate
it like other dark chocolates
Price : 50
Remove
pears
Pears is a popular soap known
Price : 34
Remove
Cart Summary
Total | Checkout | Payment
Total : $108.00
Current Address
andheri,mumbai
Update Address
Pay with card
Card Number
Expiration Date (MM/YY)
CVV (3 digits)
MM/YY
By paying with my card, I agree to the PayPal Privacy Statement.
Make Payment


Your Orders
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure G
Test Card Numbers | Mast
localhost:3000/dashboard/user/orders
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
MINTI
CART
Dashboard
Profile
Orders
All Orders
Status Buyer date Payment Quantity
1 Shipped minti a few seconds ago Success 2
potato
Potatoes are a versatile and w
Price : 7
potato
Potatoes are a versatile and w
Price : 7
Status Buyer date Payment Quantity


Your Profile
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure G
Test Card Numbers | Mast
localhost:3000/dashboard/user/profile
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
MINTI
CART
Dashboard
Profile
Orders
USER PROFILE
minti
minti@gmail.com
Enter Your Password
995943845
andheri,mumbai
UPDATE
All Right Reserved © Grocer
Search
ENG
IN
14:22
19-10-2023


Create Product
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure G
Test Card Numbers | Mast
localhost:3000/dashboard/admin/create-product
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
ADMIN
CART
Admin Panel
Create Category
Create Product
Products
Orders
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 © Grocer
Search
ENG
IN
14:23
19-10-2023


Create Category
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure
Test Card Numbers | Mast
localhost:3000/dashboard/admin/create-category
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
ADMIN
CART
Admin Panel
Create Category
Create Product
Products
Orders
Manage Category
Enter new category
Submit
Name
Actions
vegatable
Edit
Delete
soap
Edit
Delete
cold drinks
Edit
Delete
chocolates
Edit
Delete
All Right Reserved © Grocer
Search
ENG
IN
14:24
19-10-2023


All Orders Data
tomato - Google Search
dove soap - Google Search
pears soap - Google Search
Buy Pears Soap Bar Pure
Test Card Numbers | Mast
localhost:3000/dashboard/admin/orders
ECOMMERCE APP
Search
Search
HOME
CATEGORIES
ADMIN
CART
Admin Panel
Create Category
Create Product
Products
Orders
All Orders
Status Buyer date Payment Quantity
1 Processing minti a few seconds ago Failed 2
Dairy milk
Dairy Milk chocolate is generally enjoyed for its delicious taste and satisfying texture.
Price : 24
pears
Pears is a popular soap known for its unique appearance and mild, glycerin-based formulation
Price : 34
Status Buyer date Payment Quantity
2 Shipped minti a few seconds ago Success 2
potato
Search
ENG
IN
14:24
19-10-2023

11)Conclusion

- The functionality successfully implemented are -

User Functionality -

- The user can filter the products by category or by price.
- The user can search for the desired products.
- The user can add items in the cart which he wishes to buy.
- The User can change his/her profile.
- User can add card payment details

Admin Functionality -

- Admin can add and remove products
- Admin can see all the orders of different users
- Admin can create and remove categories
- Admin can change the order shipping status

12)Limitation and Future Extension

- The limitations of your project

- The user cannot track their own product and where it is right now.
- The user can not increase the quantity of product in their cart with the help of + or – buttons which are not implemented by our system.
- Payment method is not fully functional
- The user can not return the product

- The functionality which was not implemented (if any)

- No tracking functionality implemented
- No functionality to increase or decrease the product quantity in cart.

- List of future extension to your project

- We would add + and – buttons in the cart to increase or decrease the quantity of product.
- We would add product tracking functionality for the users for their convenience.

13) Bibliography:- https://www.w3schools.com/
https://reactjs.org/
https://www.mongodb.com/
https://expressjs.com/
https://www.youtube.com/