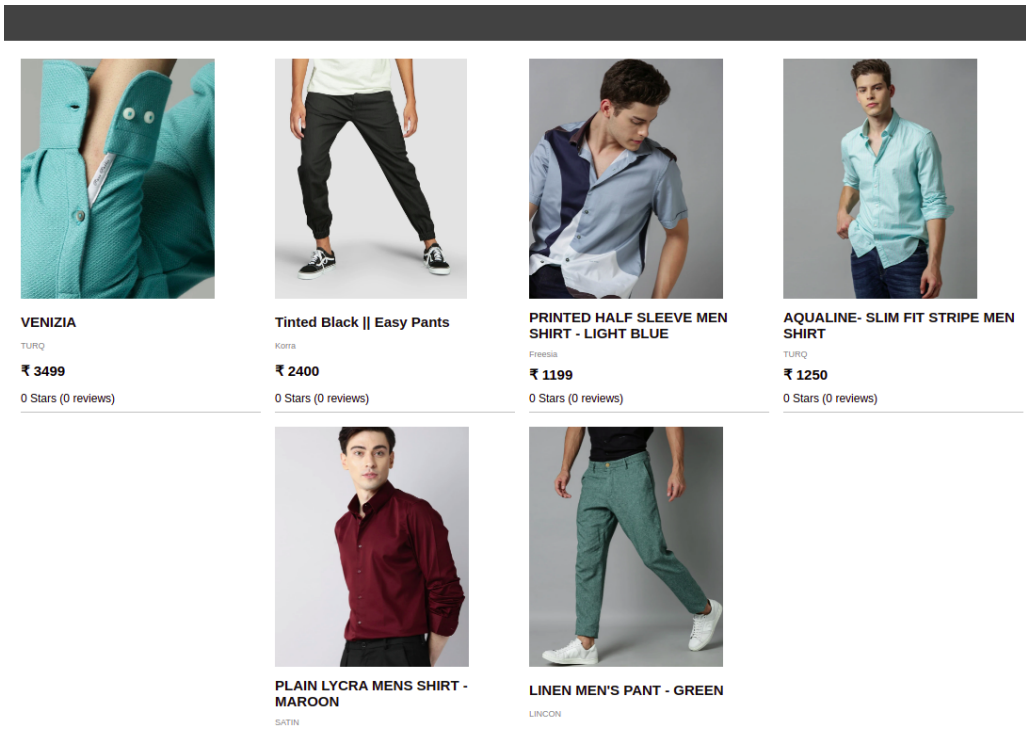
E-Commerce Project REPORT

An e-commerce website - Shopify



Faculty Members

Dr. Manoj Wariya
Mr. Sailendra Kumar Singh
Mr. Saurav Tripathi
Mr. Ataussamad
Dr. D S Kushwaha

Team Members

Ajay Singh (2019CA46)
Shivam Sharma (2019CA58)
Mohd. Azeem (2019CA83)
Vishwam Singh (2019CA55)

13/11/2021

MCA

Acknowledgement :

In preparation of our project, we had to take the help and guidance of some respected persons, who deserve our deepest gratitude. As the completion of this project gave us much pleasure, we would like to show our gratitude towards **Mr. Manoj Wariya Sir**, Course Coordinator, **Motilal Nehru National Institute of Technology** for giving us quick guidelines for the project. We would also like to expand our gratitude to all those who have directly and indirectly guided me in writing this Project.

In addition, a thank you to **Mr. Saurav Tripathi Sir**, **Mr. Ataussamad** and **Mr. Sailendra Kumar Singh** who introduced us to the Methodology of work through numerous consultations, assignments and lectures, and whose passion for the “underlying structures” had lasting effect.

The success and final outcome of this assignment required a lot of guidance and assistance from many people and we were extremely fortunate to have got this all along the completion of our project work. Whatever we have done is only due to such guidance and assistance and we would not forget to thank them.

INTRODUCTION

This e-commerce website was developed as a project for an E-commerce laboratory. It enhances the understanding of software development lifecycle and implementation of hierarchical nature of authority for proper working of the website. Each user is authenticated through proper authentication channels in order to access the website.

Technologies such as ReactJs, NodeJs, MongoDB, are used in order to complete this project. Browser cookies are used in order to store and access the information and data of the logged in user.

Proper strategy was required to complete the whole project. And tasks specified in each assignment helped achieve the proper planning and implementation of the same.

- This project enables one to think about the hierarchical administration of websites and by deploying multiple users in the same project.
- Project planning and strategy played an important role in the completion of the project.
- Enabling team members to collaborate online to create a single website requires a lot of planning and strategy. Only by carefully planning and strategizing one can achieve the proper results.
- Division of Project among the team members is an important task so as not to overload the team members.

Technologies Used :

Front-End:

(i) ReactJs - It combines Javascript and HTML/CSS into one file to provide excellent cross platform support and UI focused design. Templating is easy as components can be rendered once and re-used thereby increasing the overall productivity and modularity of the project.Rich development tools make debugging easier and development faster.

(ii) ReduxJs - Redux is an open-source JavaScript library for managing and centralizing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to Facebook's Flux architecture. It can maintain a store to store the data globally for the running web-application. It enhances the usage of hooks in react. States can be authenticated before updation or retrieval.

(iii) JS-Cookie - To store the information of the current user in the browser cookie to give support for cart items and session variables.

(iv) Babel : Babel is a free and open-source JavaScript transcompiler that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript that can be run by older JavaScript engines. Babel is a popular tool for using the newest features of the JavaScript programming language. To convert JS6 into JS5 and render JSX elements.

(v) React-Router-dom : React Router DOM enables you to implement dynamic routing in a web app. Unlike the traditional routing architecture in which the routing is handled in a configuration outside of a running app, React Router DOM facilitates component-based routing according to the needs of the app and platform. For declaring routes and paths in the front-end.

Back-End:

(i) NodeJs - Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Since NodeJs is an asynchronous and event driven javascript runtime environment, it provides faster execution of codes and massive speed gains which enables it to handle multiple connections single handedly.

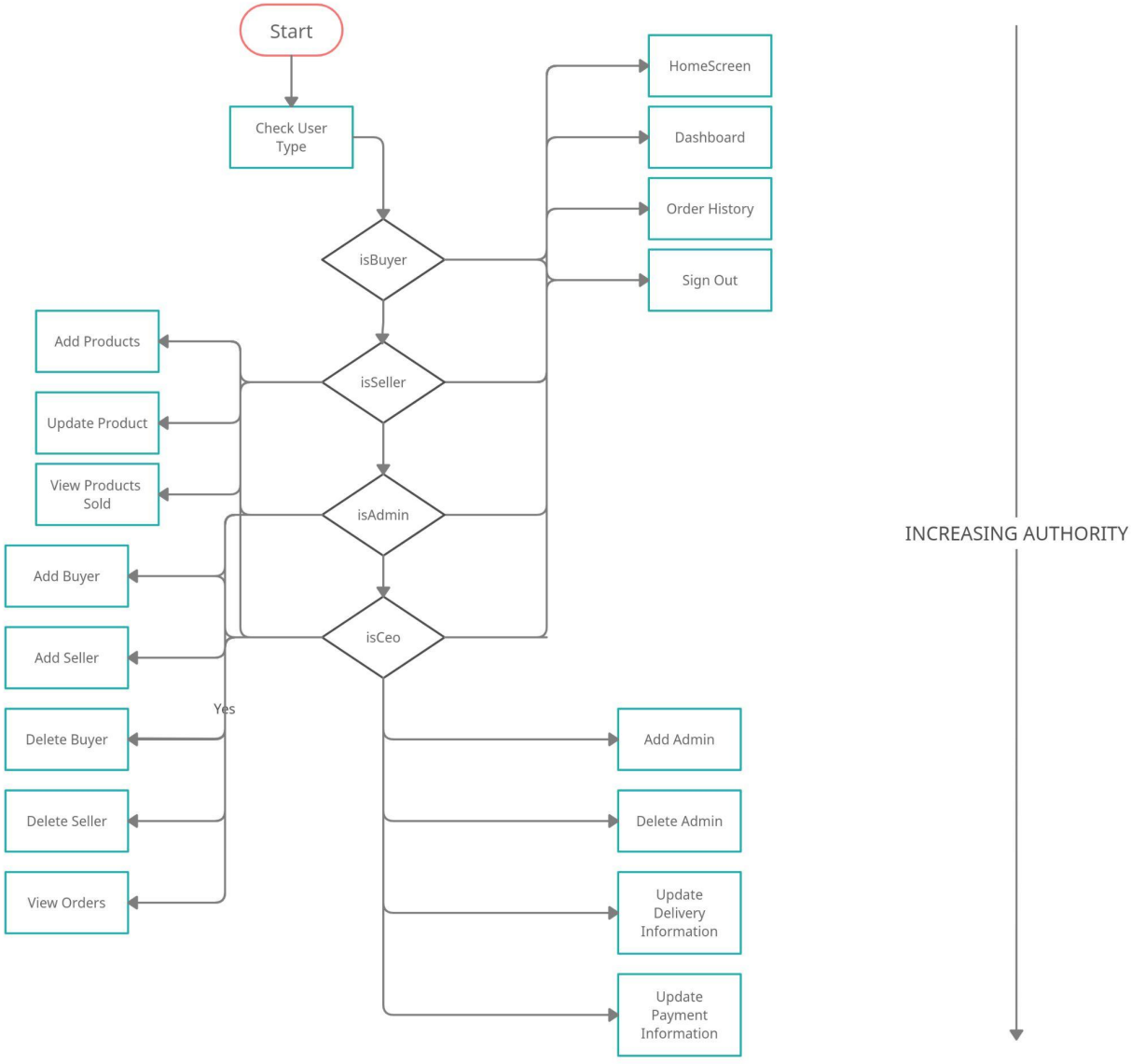
(ii) ExpressJs - Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

(iii) MongoDB - MongoDB uses a similar structure to store data as JSON, therefore it is easier to implement in a development environment. It gives the additional feature of adding fields after the collection has been implemented which enables users to implement additional information based on the object.

(iv) Mongoose - Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.Mongoose has inbuilt codes for all mongoDB queries implemented into functions which makes it easier to work with and efficiently.

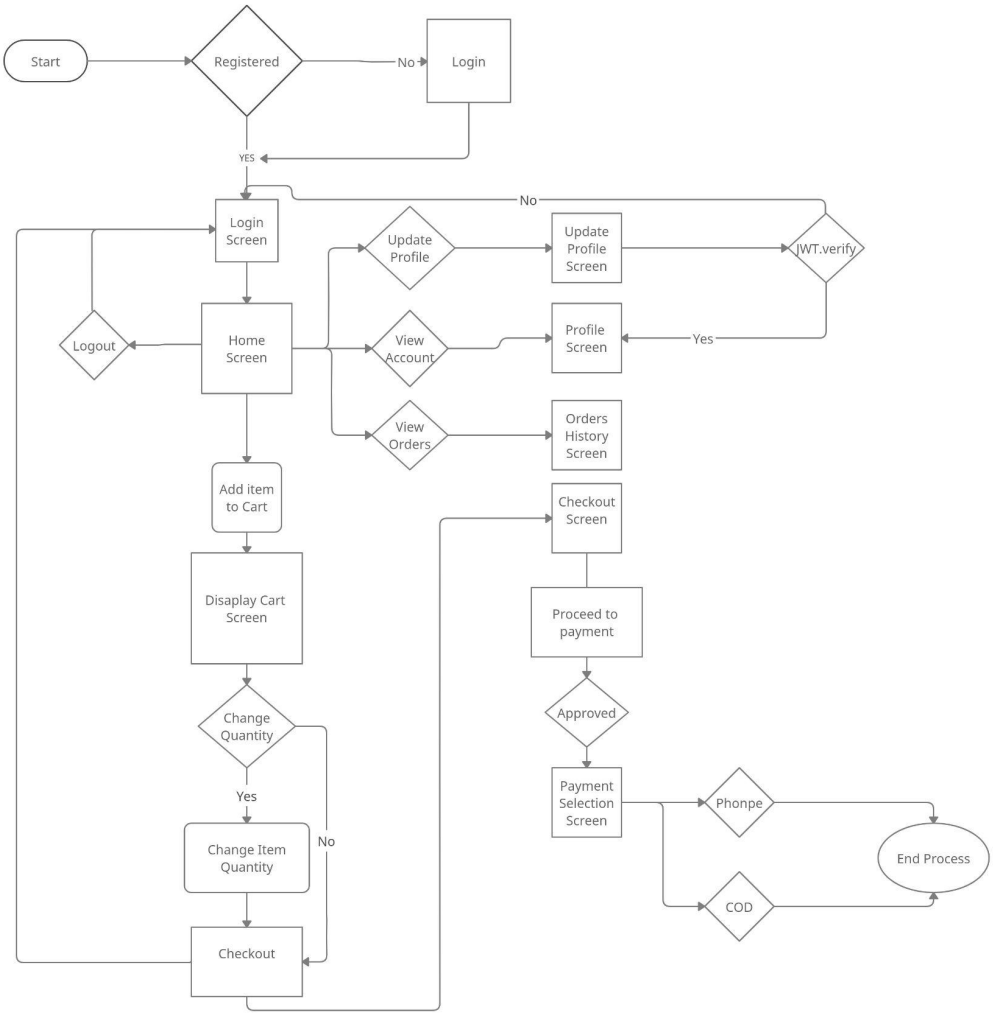
UML Class Diagram:

FlowChart For Users Operation (Event Driven) :



FlowChart for placing order and payment :

◂Yes



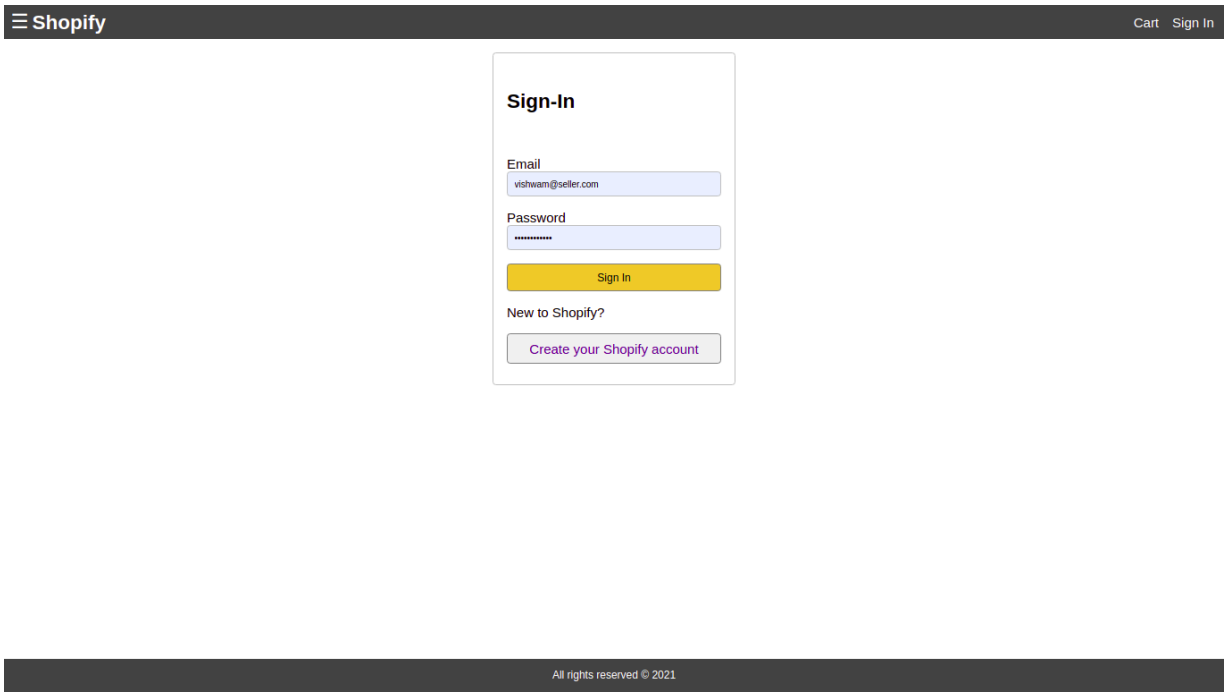
Creating a web page for login and registration with proper validation channels :

LoginScreen (Code) :

```
import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom/cjs/react-router-dom.min";
import { useDispatch, useSelector } from "react-redux";
import { signin } from "../actions/userActions";

function SigninScreen(props) {

    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");

    const userSignin = useSelector(state => state.userSignin);
    const { loading, userInfo, error } = userSignin;
    const dispatch = useDispatch();
    const redirect = props.location.search?props.location.search.split("=")[1]:"/";

    useEffect(() => {
        if(userInfo) {
              redirect === "cart" ? props.history.push(redirect + "/" + userInfo._id) :
props.history.push(redirect);
        }
        //eslint-disable-next-line
    }, [userInfo]);

    const submitHandler = (e) => {
        e.preventDefault();
        dispatch(signin(email, password));
    }

    return <div className="form">
        <form onSubmit={submitHandler}>
            <ul className="form-container">
                <li>
                    <h2>Sign-In</h2>
                </li>
                <li>
                    {loading && <div>Loading...</div>}
                    {error && <div>Invalid Email or Password</div>}
                </li>
                <li>
```

```
<label htmlFor="email">
                    Email
                </label>
                <input
                type="email"
                name="email"
                id="email"
                onChange={(e) => setEmail(e.target.value)}
                placeholder="Enter your email"
                value={email}
                >
                </input>
            </li>
            <li>
                <label htmlFor="password">
                    Password
                </label>
                <input
                type="password"
                name="password"
                id="password"
                onChange={(e) => setPassword(e.target.value)}
                placeholder="Enter your password"
                value={password}
                >
                </input>
            </li>
            <li>
                <button className="button primary">Sign In</button>
            </li>
            <li>
                New to Shopify?
            </li>
            <li>
                    <Link to={redirect === "/" ? "register" : "register?redirect=" +
redirect} className="button secondary text-center">Create your Shopify
account</Link>
            </li>
        </ul>
      </form>
   </div>
}

export default SigninScreen;
```

LoginScreen (ScreenShot) :



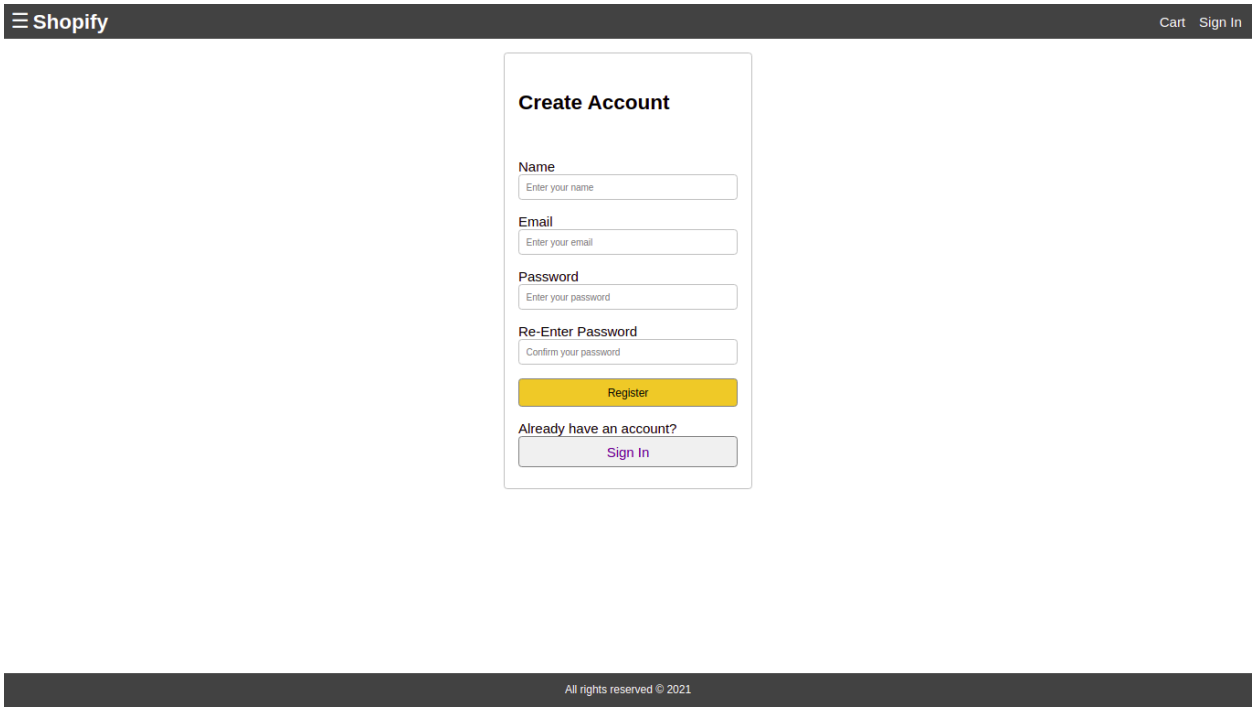
RegistrationScreen (Code) :

```
import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom/cjs/react-router-dom.min";
import { useDispatch, useSelector } from "react-redux";
import { register } from "../actions/userActions";

function RegisterScreen(props) {

    const [name, setName] = useState("");
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [rePassword, setRePassword] = useState("");

    const userRegister = useSelector(state => state.userRegister);
    const { loading, userInfo, error } = userRegister;
    const dispatch = useDispatch();
    const redirect = props.location.search?props.location.search.split("=")[1]:"/";

    useEffect(() => {
        if(userInfo) {
            props.history.push(redirect);
        }
```

```
    //eslint-disable-next-line
}, [userInfo]);

const submitHandler = (e) => {
    e.preventDefault();
    dispatch(register(name, email, password));
}

return <div className="form">
    <form onSubmit={submitHandler}>
        <ul className="form-container">
            <li>
                <h2>Create Account</h2>
            </li>
            <li>
                {loading && <div>Loading...</div>}
                {error && <div>Invalid Email or Password</div>}
            </li>
            <li>
                <label htmlFor="name">
                    Name
                </label>
                <input
                type="text"
                name="name"
                id="name"
                onChange={(e) => setName(e.target.value)}
                placeholder="Enter your name"
                value={name}
                >
                </input>
            </li>
            <li>
                <label htmlFor="email">
                    Email
                </label>
                <input
                type="email"
                name="email"
                id="email"
                onChange={(e) => setEmail(e.target.value)}
                placeholder="Enter your email"
                value={email}
                >
                </input>
            </li>
```

```
            <li>
              <label htmlFor="password">
                Password
              </label>
              <input
              type="password"
              name="password"
              id="password"
              onChange={(e) => setPassword(e.target.value)}
              placeholder="Enter your password"
              value={password}
              >
              </input>
            </li>
            <li>
              <label htmlFor="re-password">
                Re-Enter Password
              </label>
              <input
              type="password"
              name="re-password"
              id="re-password"
              onChange={(e) => setRePassword(e.target.value)}
              placeholder="Confirm your password"
              value={rePassword}
              >
              </input>
            </li>
            <li>
              <button className="button primary">Register</button>
            </li>
            <li>
              Already have an account?
                <Link to={redirect === "/" ? "signin" : "signin?redirect=" + redirect}
className="button secondary text-center">Sign In</Link>
            </li>
          </ul>
        </form>
    </div>
}

export default RegisterScreen;
```

RegistrationScreen (Screenshot) :



Homepage of Shopify :

HomeScreen (Code) :

```
import React, { useEffect } from "react";
import Product from "../components/Product.js";
import { useDispatch, useSelector } from "react-redux";
import { listProducts } from "../actions/productActions";

function HomeScreen(props) {

  const productList = useSelector(state => state.productList);
  const { products, loading, error } = productList;
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(listProducts());
```

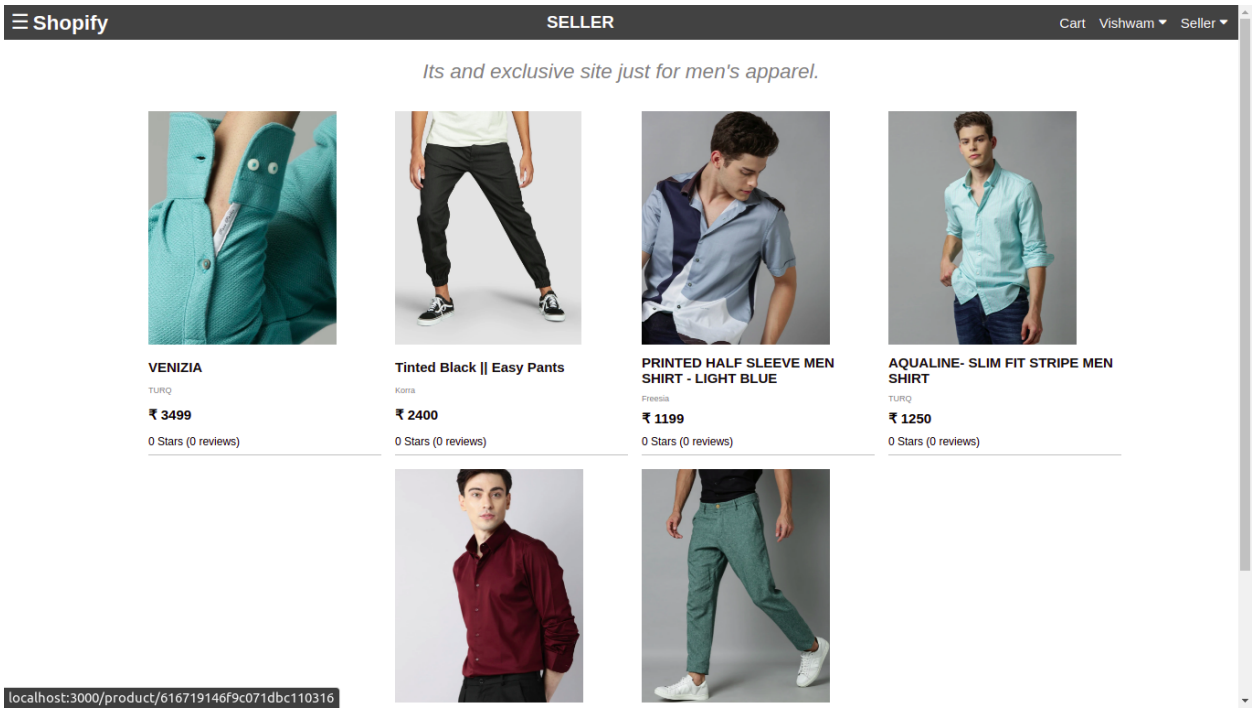
```
      return () => {

      };
      //eslint-disable-next-line
   }, []);

   return loading ? <div>Loading...</div> :
          error ? <div>{error}</div> :
          (<div>
                <h3 className="purpose"><i>Its and exclusive site just for men's
apparel.</i></h3>
          <ul className="products">
             {products.map(product =>
                <Product
                   key={product._id}
                   id={product._id}
                   image={product.image}
                   name={product.name}
                   brand={product.brand}
                   price={product.price}
                   rating={product.rating}
                   numReviews={product.numReviews}
                />
             )}
          </ul>
          </div>);
}

export default HomeScreen;
```

Product Component :

```
import React from "react";
import { Link } from "react-router-dom";

function Product(props) {
    return (
        <li>
            <div className="product">
                <Link to={"/product/" + props.id}>
                    <img className="product-image" src={props.image} alt="Product"
/>
                </Link>
```

```
            <div className="product-name">
                                                <Link to={"/product/" + props.id}
className="link-color">{props.name}</Link>
            </div>
            <div className="product-brand">{props.brand}</div>
            <div className="product-price">&#8377; {props.price}</div>
                                <div  className="product-rating">{props.rating}  Stars
({props.numReviews} reviews)</div>
          </div>
        </li>
    );
}

export default Product;
```

HomeScreen (ScreenShot) :

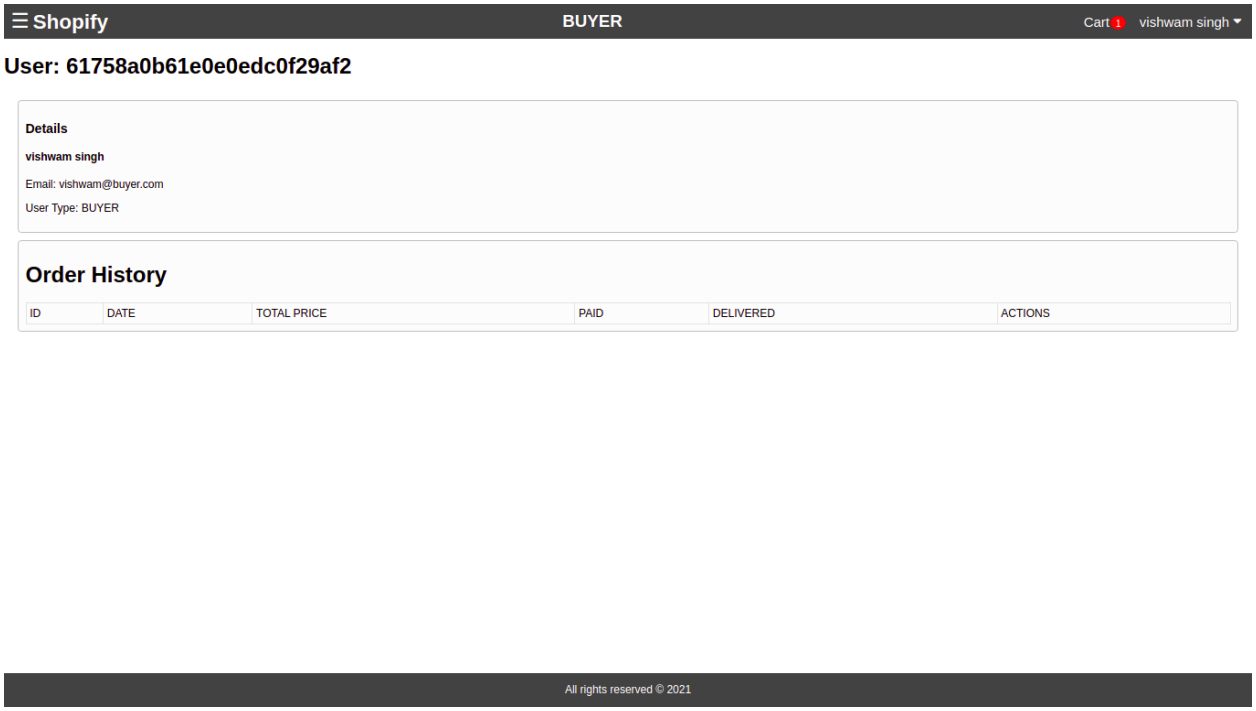


UserProfile (Code) :

```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { listUsers, deleteUser } from "../actions/userActions";

function UsersScreen(props) {

    const userSignin = useSelector(state => state.userSignin);
    const { userInfo } = userSignin;

    const userList = useSelector(state => state.userList);
    const { loading, users, error } = userList;

    const userDelete = useSelector(state => state.userDelete);
    const { success: successDelete } = userDelete;

    const check = (type, userType) => {
        if(type === "ceo" && userType !== "ceo")
            return true;
        if(type === "admin" && userType !== "ceo" && userType !== "admin")
            return true;
        return false;
    }

    const dispatch = useDispatch();

    useEffect(() => {

        dispatch(listUsers());
        //eslint-disable-next-line
    }, [successDelete]);

    const deleteHandler = (user) => {
        dispatch(deleteUser(user._id));
    }

    return(<div className="content content-margined">
        <div className="product-header">
            <h3>Users</h3>
        </div>
        <li>
            { loading && <div>Loading...</div> }
            { error && <div>{error}</div> }
        </li>
```

```
      <div className="product-list">
         <table className="table">
            <thead>
               <tr>
                  <th>ID</th>
                  <th>Name</th>
                  <th>Email</th>
                  <th>User Type</th>
                  <th>Action</th>
               </tr>
            </thead>
            <tbody>
               {users.map(user => {
                  return (
                  <tr key={user._id}>
                     <td>{user._id}</td>
                     <td>{user.name}</td>
                     <td>{user.email}</td>
                     <td>{user.type}</td>
                     <td>
                        {
                           check(userInfo.type, user.type) &&
                                    <button className="button" onClick={() =>
deleteHandler(user)}>Delete</button>
                        }
                     </td>
                  </tr>)
               })}
            </tbody>
         </table>
      </div>
   </div>);
}

export default UsersScreen;
```

UserProfile (ScreenShot) :



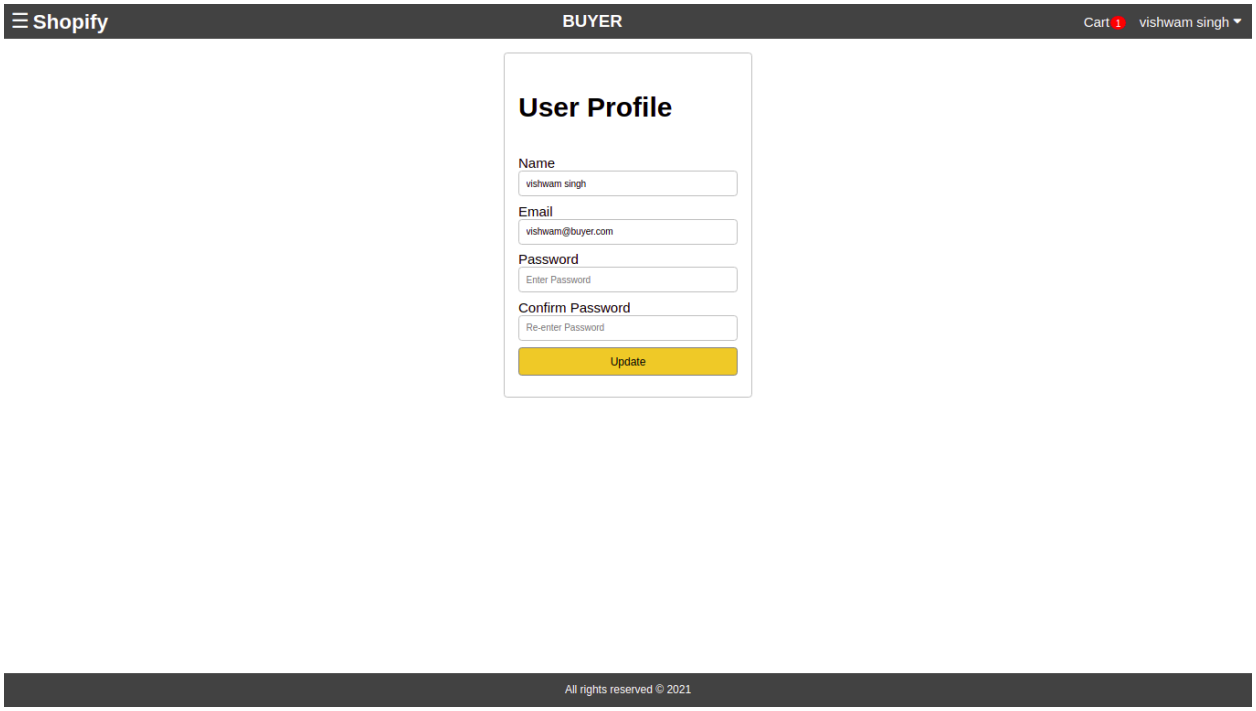
UpdateProfile (Code) :

```
import React, { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { detailsUser, updateUserProfile } from "../actions/userActions";
import { USER_UPDATE_PROFILE_RESET } from "../constants/userConstants";

function ProfileScreen(props) {

   const [name, setName] = useState("");
   const [email, setEmail] = useState("");
   const [password, setPassword] = useState("");
   const [confirmPassword, setConfirmPassword] = useState("");

   const userSignin = useSelector(state => state.userSignin);
   const { userInfo } =  userSignin;

   const userDetails = useSelector(state => state.userDetails);
   const { loading, user, error } = userDetails;

   const userUpdateProfile = useSelector(state => state.userUpdateProfile);
```

```
    const { loading: loadingUpdate, success: successUpdate, error: errorUpdate
} = userUpdateProfile;

    const dispatch = useDispatch();

    useEffect(() => {
        if(!user) {
            dispatch({ type: USER_UPDATE_PROFILE_RESET });
            dispatch(detailsUser(userInfo._id));
        } else {
            setName(user.name);
            setEmail(user.email);
        }
    }, [dispatch, userInfo._id, user]);

    const sumbmitHandler = (e) => {
        e.preventDefault();
        if(password !== confirmPassword) {
            alert("Password and Confirm password do not match");
        }
        dispatch(updateUserProfile({ userId: user._id, name, email, password }));
    }

    return (
        <div className="form">
            <form onSubmit={sumbmitHandler}>
                <ul className="form-container">
                    <li>
                        <h1>User Profile</h1>
                    </li>
                    {loading && <div>Loading...</div>}
                    {error && <div>{error}</div>}
                    {loadingUpdate && <div>Loading...</div>}
                    {errorUpdate && <div>{errorUpdate}</div>}
                    {successUpdate && <div>Profile Updated Successfully</div>}
                    {
                        !loading && (
                        <div>
                            <li>
                                <label htmlFor="name">Name</label>
                                <input
                                    type="text"
                                    id="name"
                                    placeholder="Enter Name"
                                    value={name}
                                    onChange={(e) => setName(e.target.value)}>
```

```
                        </input>
                    </li>
                    <li>
                        <label htmlFor="email">Email</label>
                        <input
                            type="email"
                            id="email"
                            placeholder="Enter Email"
                            value={email}
                            onChange={(e) => setEmail(e.target.value)}>
                        </input>
                    </li>
                    <li>
                        <label htmlFor="password">Password</label>
                        <input
                            type="password"
                            id="password"
                            placeholder="Enter Password"
                            onChange={(e) => setPassword(e.target.value)}>
                        </input>
                    </li>
                    <li>
                                    <label  htmlFor="confirmPassword">Confirm
Password</label>
                        <input
                            type="password"
                            id="confirmPassword"
                            placeholder="Re-enter Password"
                                                        onChange={(e)  =>
setConfirmPassword(e.target.value)}>
                        </input>
                    </li>
                    <li>
                        <label />
                        <button type="submit" className="button primary">
                            Update
                        </button>
                    </li>
                </div>)
            }
        </ul>
    </form>
</div>
);
}
export default ProfileScreen;
```

UpdateProfile (ScreenShot) :

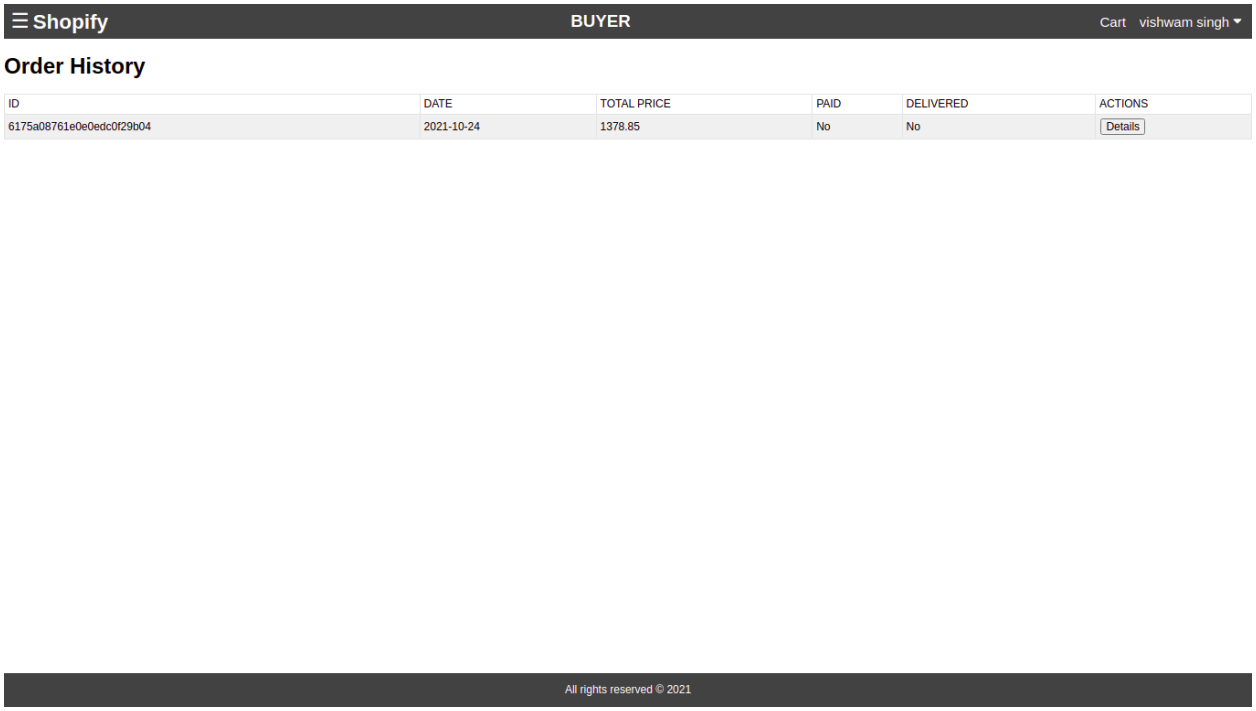


OrderHistory (Code) :

```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { listOrderUser } from "../actions/orderActions";

function OrderHistoryScreen(props) {

  const orderUserList = useSelector(state => state.orderUserList);
  const { loading, orders, error } = orderUserList;
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(listOrderUser());
  }, [dispatch]);

  return (
    <div>
      <h1>Order History</h1>
      {
        loading
        ? <div>Loading...</div>
        : error ? <div>{error}</div>
```

```
            : (
                <table className="table">
                    <thead>
                        <tr>
                            <td>ID</td>
                            <td>DATE</td>
                            <td>TOTAL PRICE</td>
                            <td>PAID</td>
                            <td>DELIVERED</td>
                            <td>ACTIONS</td>
                        </tr>
                    </thead>
                    <tbody>
                        {
                            orders && orders.map(order => {
                                return (
                                    <tr key={order._id}>
                                        <td>{order._id}</td>
                                        <td>{order.createdAt.substring(0, 10)}</td>
                                        <td>{order.totalPrice.toFixed(2)}</td>
                                            <td>{order.isPaid ? order.paidAt.substring(0, 10) :
"No" }</td>
                                                                                <td>{order.isDelivered  ?
order.deliveredAt.substring(0, 10) : "No"}</td>
                                        <td>
                                            <button
                                            type="button"
                                            className="secondary"
                                                onClick={() => props.history.push("/orders/" +
order._id)}>
                                                Details
                                            </button>
                                        </td>
                                    </tr>
                                )
                            })
                        }
                    </tbody>
                </table>
            )
        }
    </div>
  )
}
export default OrderHistoryScreen;
```

OrderHistory (ScreenShot) :



CartScreen (Code) :

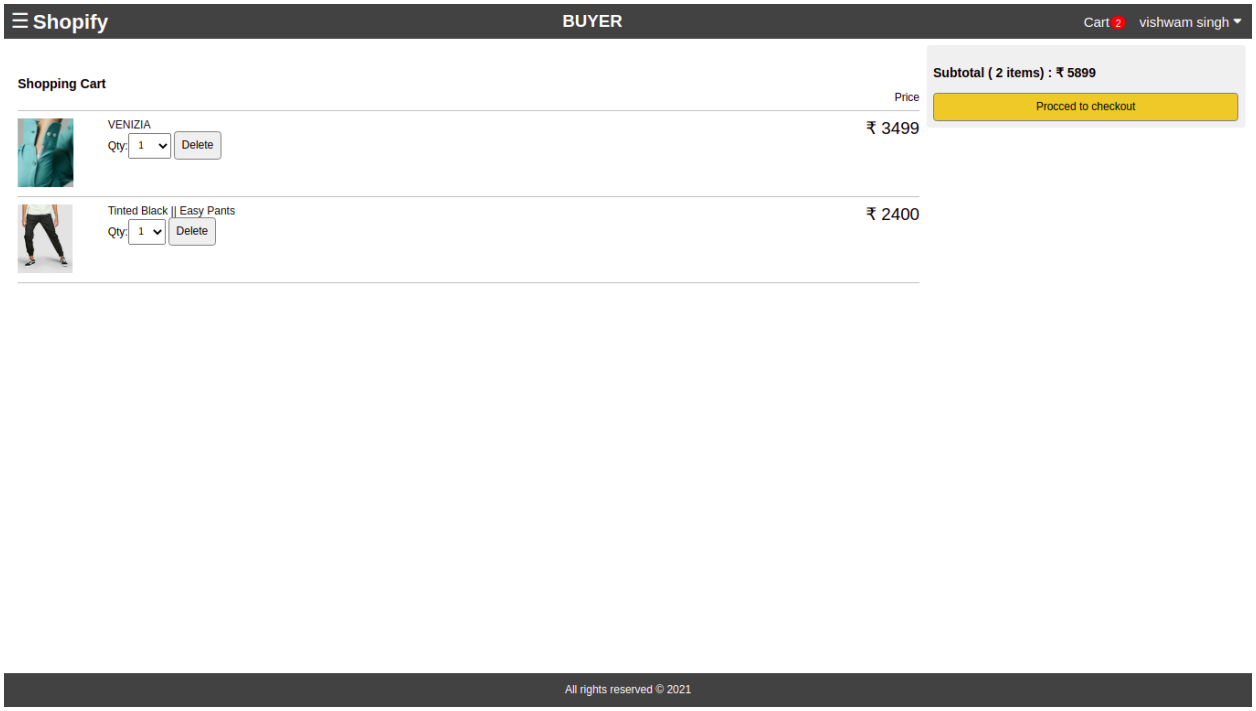
```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { Link } from "react-router-dom/cjs/react-router-dom.min";
import { addToCart, removeFromCart } from "../actions/cartActions";

function CartScreen(props) {

    const cart = useSelector(state => state.cart);
    const { cartItems } = cart;

    const productId = props.match.params.id;
    const search = props.location.search;
    const qty = search ? Number(search.split("=")[1]) : 1;
    const dispatch = useDispatch();

    useEffect(() => {
        if(productId) {
            dispatch(addToCart(productId, qty));
        }
    }, [dispatch, productId, qty]);
```

```
  const removeFromCartHandler = (productId) => {
    dispatch(removeFromCart(productId));
  }

  const checkOutHandler = () => {
    props.history.push("/signin?redirect=shipping");
  }

  return <div className="cart">
    <div className="cart-list">
      <ul className="cart-list-container">
        <li>
          <h3>Shopping Cart</h3>
          <div>
            Price
          </div>
        </li>
        {
          cartItems.length === 0 ?
          <div>
            Your Shopping cart is empty.<br />
            <Link to="/"><h3>Continue Shopping.</h3></Link>
          </div>
          :
          cartItems.map(item => {
            return (
              <li key={item.product}>
                <div className="cart-image">
                                        <Link to={"/product/" + item.product}
className="link-color">
                    <img src={item.image} alt="product" />
                  </Link>
                </div>
                <div className="cart-name">
                  <div>
                                          <Link to={"/product/" + item.product}
className="link-color">
                      {item.name}
                    </Link>
                  </div>
                  <div>
                    Qty:
                                    <select value={item.qty} onChange={(e) =>
dispatch(addToCart(item.product, e.target.value))}>
                      {
```

```
Array.from(Array(item.countInStock).keys()).map((x) => {
                                        return <option key={x} value={x}>{x}</option>
                                    })
                                }
                            </select>
                            {" "}
                            <button
                            type="button"
                            className="button"
                                                                                onClick={() =>
removeFromCartHandler(item.product)}
                            >
                                Delete
                            </button>
                        </div>
                    </div>
                    <div className="cart-price">
                        &#8377; {item.price}
                    </div>
                </li>
            );
        })
    }
</ul>
</div>
<div className="cart-action">
    <h3>
        Subtotal ( { cartItems.reduce((acc, cur) => acc + Number(cur.qty), 0)
} items)
        :
        &#8377; { cartItems.reduce((acc, cur) => acc + cur.price * cur.qty, 0) }
    </h3>
        <button onClick={checkOutHandler} className="button primary
full-width" disabled={cartItems.length === 0}>
        Procced to checkout
    </button>
</div>
</div>
}

export default CartScreen;
```

Sign out redirects the user to HomeScreen.

CartScreen (ScreenShot) :



Designing products and product details screen :

Products Screen :

```
import React, { useEffect } from "react";
import Product from "../components/Product.js";
import { useDispatch, useSelector } from "react-redux";
import { listProducts } from "../actions/productActions";

function HomeScreen(props) {

    const productList = useSelector(state => state.productList);
    const { products, loading, error } = productList;
    const dispatch = useDispatch();

    useEffect(() => {
        dispatch(listProducts());
        return () => {

        };
        //eslint-disable-next-line
    }, []);
```

```
    return loading ? <div>Loading...</div> :
            error ? <div>{error}</div> :
            (<div>
                <h3 className="purpose"><i>Its and exclusive site just for men's apparel.</i></h3>
            <ul className="products">
               {products.map(product =>
                  <Product
                     key={product._id}
                     id={product._id}
                     image={product.image}
                     name={product.name}
                     brand={product.brand}
                     price={product.price}
                     rating={product.rating}
                     numReviews={product.numReviews}
                  />
               )}
            </ul>
            </div>);
}

export default HomeScreen;
```

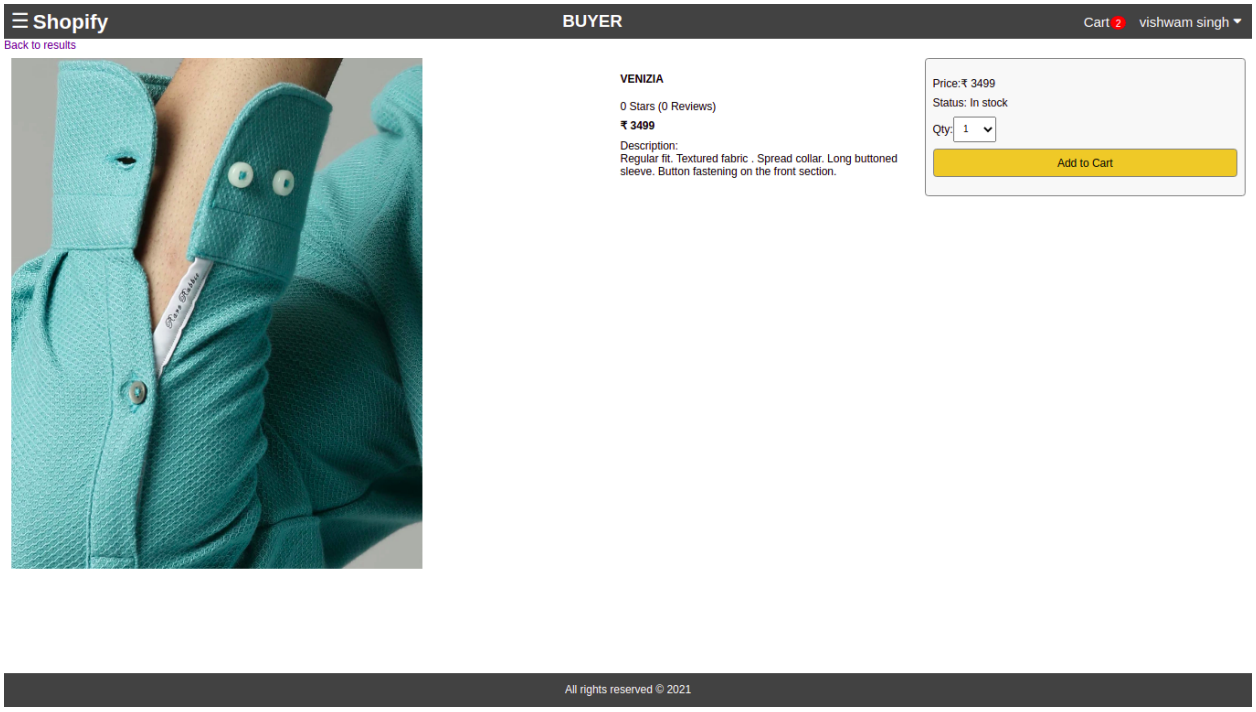
ProductScreen :

```
import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom/cjs/react-router-dom.min";
import { useDispatch, useSelector } from "react-redux";
import { detailsProduct } from "../actions/productActions";

function ProductScreen(props) {

    const [qty, setQty] = useState(1);
    const productDetails = useSelector(state => state.productDetails);
    const { product, loading, error } = productDetails;
    const dispatch = useDispatch();

    useEffect(() => {
        dispatch(detailsProduct(props.match.params.id));
        //eslint-disable-next-line
    }, []);
```

```
  const handleAddToCart = () => {
    props.history.push("/cart/" + props.match.params.id + "?qty=" + qty);
  }

  return <div>
    <div>
      <Link to="/">Back to results</Link>
    </div>
    {loading ? <div>Loading...</div> :
      error ? <div>{error}</div> :
        (<div className="details">
          <div className="details-image">
            <img src={product.image} alt={product.name}/>
          </div>
          <div className="details-info">
            <ul>
              <li>
                <h4>{product.name}</h4>
              </li>
              <li>
                {product.rating} Stars ({product.numReviews} Reviews)
              </li>
              <li>
                <b>&#8377; {product.price}</b>
              </li>
              <li>
                Description:
                <div>
                  {product.description}
                </div>
              </li>
            </ul>
          </div>
          <div className="details-action">
          <ul>
            <li>Price:&#8377; {product.price}</li>
              <li>
                    Status: {product.countInStock > 0 ? "In stock" : "Out of
stock."}
              </li>
              <li>
                Qty:
                                        <select value={qty} onChange={(e) =>
{setQty(e.target.value)}}>
                  {[...Array(product.countInStock).keys()].map(x =>
```

```
                                <option value={x + 1} key={x + 1} >{x + 1}</option>
                            )}
                            </select>
                        </li>
                        <li>
                            {product.countInStock > 0 && <button className="button
primary" onClick={handleAddToCart}>
                                Add to Cart
                            </button>}
                        </li>
                    </ul>
                </div>
            </div>)
        }
    </div>
}

export default ProductScreen;
```

Products Details (ScreenShot) :



Login for all types of Users and Buyer (User Type 1) :

```
import './App.css';
import { BrowserRouter, Route, Link } from "react-router-dom";
import HomeScreen from "./screens/HomeScreen";
import ProductScreen from './screens/ProductScreen';
import CartScreen from "./screens/CartScreen";
import SigninScreen from './screens/SigninScreen';
import { useDispatch, useSelector } from 'react-redux';
import RegisterScreen from './screens/RegisterScreen';
import ProductsScreen from './screens/ProductsScreen';
import ShippingScreen from './screens/ShippingScreen';
import PaymentScreen from './screens/PaymentScreen';
import PlaceOrderScreen from './screens/PlaceOrderScreen';
import OrderScreen from "./screens/OrderScreen";
import { signOut } from './actions/userActions';
import { useEffect, useState } from 'react';
import UsersScreen from './screens/UsersScreen';
import OrderHistoryScreen from './screens/OrderHistoryScreen';
import ProfileScreen from "./screens/ProfileScreen";
import OrdersScreen from "./screens/OrdersScreen";
import AdminMenu from "./components/AdminMenu";
import UserMenu from "./components/UserMenu";
import CeoMenu from "./components/CeoMenu";
import SellerMenu from "./components/SellerMenu";
import CreateUserScreen from './screens/CreateUserScreen';
import DashboardScreen from "./screens/DashboardScreen";

function App() {

    const dispatch = useDispatch();
    const userSignin = useSelector(state => state.userSignin);
    const cart = useSelector(state => state.cart);
    const { cartItems } = cart;
    const { userInfo } = userSignin;
    const [ reload, setReload ] = useState(false);

    useEffect(() => {
        if(reload) {
            window.location.reload(false);
            setReload(false)
        }
        return {

        };
```

```
  }, [reload]);

  const openMenu = () => {
    document.querySelector(".sidebar").classList.add("open");
  }

  const closeMenu = () => {
    document.querySelector(".sidebar").classList.remove("open");
  }

  const signOutHandler = () => {
    dispatch(signOut());
    setReload(true);
  }

  return (
    <BrowserRouter>
      <div className="grid-container">
        <header className="header">
          <div className="brand">
            <button onClick={openMenu}>&#9776;</button>
            <Link to="/">Shopify</Link>
          </div>
          <div className="header">
                                                    {userInfo && <span
classNameadmin-display">{userInfo.type.toUpperCase()}</span>}
          </div>
          <div className="header-links">
            <Link to="/cart">Cart
            {cartItems.length > 0 && (
              <span className="badge">{cartItems.length}</span>
            )}
            </Link>
            {
              userInfo ? ( <UserMenu
                name={userInfo.name}
                signOutHandler={signOutHandler}
                >
              </UserMenu>
              ) : (
                <Link to="/signin">Sign In</Link>
              )
            }
                  { userInfo && userInfo.type === "seller" && <SellerMenu
id={userInfo._id}></SellerMenu>}
                              { userInfo && userInfo.type === "admin" &&
```

```
<AdminMenu></AdminMenu> }
                                { userInfo && userInfo.type === "ceo" &&
<CeoMenu></CeoMenu>}
            </div>
          </header>
          <aside className="sidebar">
            <h3>Shopping Categories</h3>
                                  <button  className="sidebar-close-button"
onClick={closeMenu}>x</button>
            <ul>
              <li>
                <a className="link-color" href="index.html">Pants</a>
              </li>
              <li>
                <a className="link-color" href="index.html">Shirts</a>
              </li>
            </ul>
          </aside>
          <main className="main">
            <div className="content">

              <Route path="/dashboard" component={DashboardScreen} />
              <Route path="/createUser" component={CreateUserScreen} />
              <Route path="/allorders" component={OrdersScreen} />
              <Route path="/profile" component={ProfileScreen} />
                <Route path="/orderhistory" component={OrderHistoryScreen}
/>
              <Route path="/orders/:id" component={OrderScreen} />
              <Route path="/users" component={UsersScreen} />
              <Route path="/signout" component={HomeScreen} />
              <Route path="/placeorder" component={PlaceOrderScreen} />
              <Route path="/payment" component={PaymentScreen} />
              <Route path="/shipping" component={ShippingScreen} />
              <Route path="/products/all" component={ProductsScreen} />
              <Route path="/products/seller/:id" component={ProductsScreen}
/>
              <Route path="/register" component={RegisterScreen} />
              <Route path="/signin" component={SigninScreen} />
              <Route path="/product/:id" component={ProductScreen} />
              <Route path="/" exact={true} component={HomeScreen} />
              <Route path="/cart/:id?" component={CartScreen} />

            </div>
          </main>
          <footer className="footer">
            All rights reserved &copy; 2021
```

```
            </footer>
          </div>
        </BrowserRouter>
    );
}

export default App;
```

Universal Component for all users and the only component containing Buyer Menu :

```
import React from "react";
import { Link } from "react-router-dom";

function UserMenu(props) {
    return (
        <div className="dropdown">
            <Link to="/profile">
                {props.name + " "}
                <i className="fa fa-caret-down"></i>
            </Link>
            <ul className="dropdown-content">
                <li>
                    <Link to="/dashboard">Dashboard</Link>
                </li>
                <li>
                    <Link to="/orderhistory">Order History</Link>
                </li>

                <li>
                    <Link to="/support">Support</Link>
                </li>
                <li>
                    <Link to="/" onClick={props.signOutHandler}>
                        Sign Out
                    </Link>
                </li>
            </ul>
        </div>
    );
}

export default UserMenu;
```

AdminMenuComponent :

```
import React from "react";
import { Link } from "react-router-dom";

function AdminMenu(props) {
    return (
        <div className="dropdown">
                                <Link to="#admin">Admin { " " } <i className="fa
fa-caret-down"></i></Link>
            <ul className="dropdown-content">
                <li>
                    <Link to="/products/all">Products</Link>
                </li>
                <li>
                    <Link to="/users">Users</Link>
                </li>
                <li>
                    <Link to="/allorders">Orders</Link>
                </li>
            </ul>
        </div>
    );
}

export default AdminMenu;
```

Implementation of 2 more types of users other than buyer and admin. Chosen users are CEO and seller :

CEO Screen Component :

```
import React from "react";
import { Link } from "react-router-dom";

function CeoMenu(props) {
    return (
        <div className="dropdown">
```

```
                        <Link to="#admin">CEO { " " } <i className="fa
fa-caret-down"></i></Link>
          <ul className="dropdown-content">
            <li>
              <Link to="/products/all">Products</Link>
            </li>
            <li>
              <Link to="/users">Users</Link>
            </li>
            <li>
              <Link to="/allorders">Orders</Link>
            </li>
            <li>
              <Link to="/createUser">Create User</Link>
            </li>
          </ul>
        </div>
    );
}
export default CeoMenu;
```

Seller Screen Component :

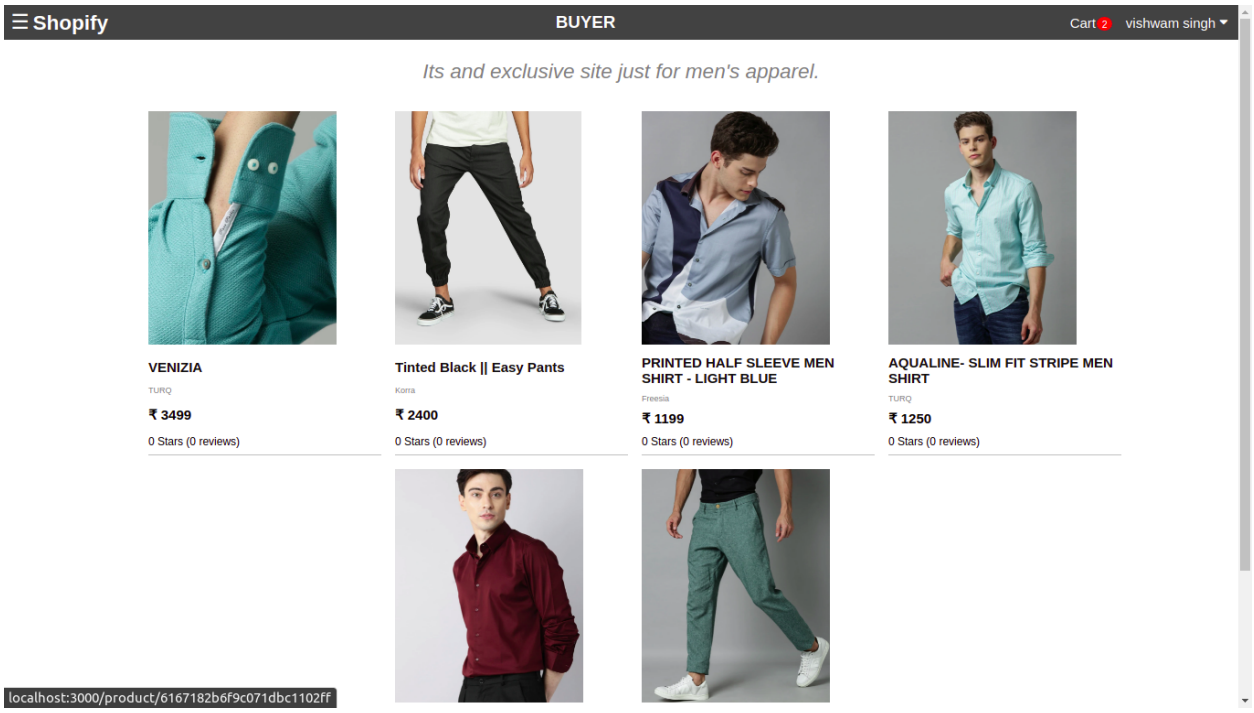
```
import React from "react";
import { Link } from "react-router-dom";

function SellerMenu(props) {
    return (
        <div className="dropdown">
                        <Link to="#admin">Seller { " " } <i className="fa
fa-caret-down"></i></Link>
          <ul className="dropdown-content">
            <li>
              <Link to="/dashboard">Dashboard</Link>
            </li>
            <li>
              <Link to={"/products/seller/" + props.id}>Products</Link>
            </li>
          </ul>
        </div>
    );
}
export default SellerMenu;
```

Authorization for CEO, Admin and Seller :

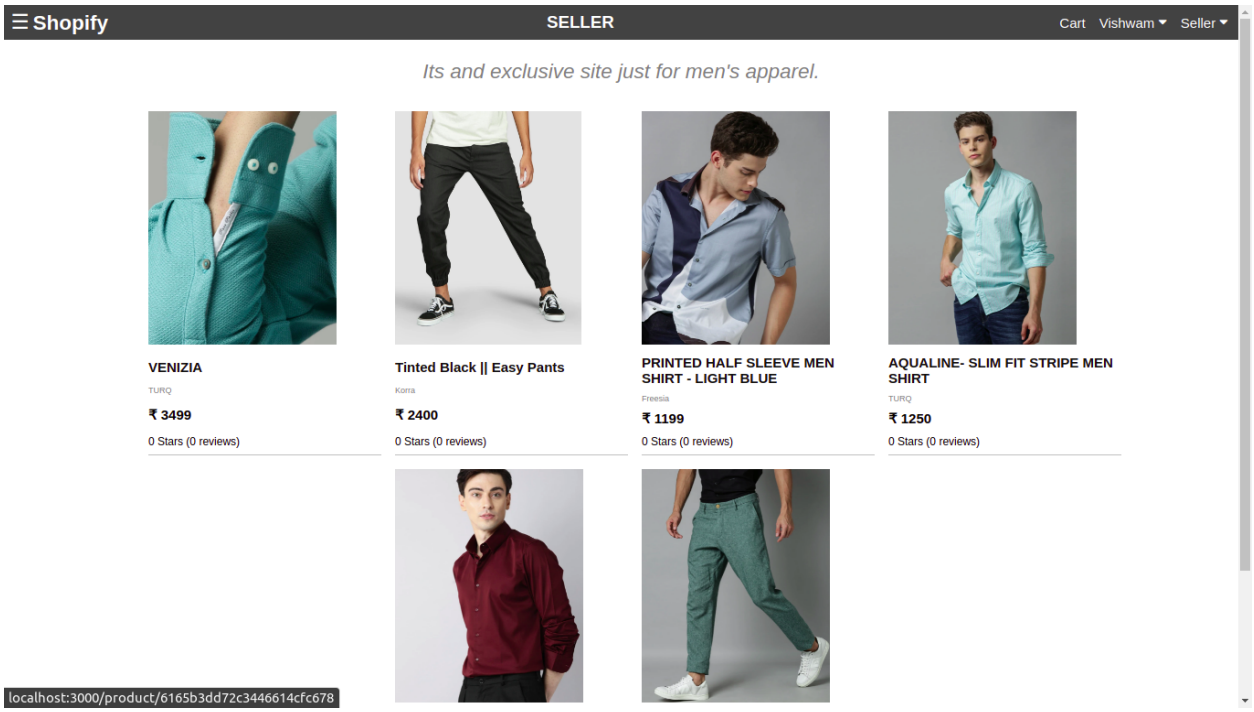
```
import jwt from "jsonwebtoken";
import config from "./config";

const getToken = (user) => {
    return jwt.sign({
        _id: user._id,
        name: user.name,
        email: user.email,
        type: user.type
    }, config.JWT_SECRET_KEY, {
        expiresIn: "48h"
    });
}

const isAuth = (req, res, next) => {
    const token = req.headers.authorization;
    if(token) {
        const onlyToken = token.slice(6, token.length);
        jwt.verify(onlyToken, config.JWT_SECRET_KEY, (error, decode) => {
            if(error) {
                res.status(401).send({msg: "Invalid Token"});
            }
            req.user = decode;
            next();
            return;
        });
    } else {
        return res.status(401).send({msg: "Token is not supplied."});
    }

}

const isAdmin = (req, res, next) => {
    if(req.user && req.user.type === "admin") {
        return next();
    }
    else {
        return res.status(401).send({msg: "Admin token is not valid."});
    }
}
```

```
const isCeo = (req, res, next) => {
    if(req.user && req.user.type === "ceo") {
        return next();
    }
    else {
        return res.status(401).send({msg: "CEO token is not valid."});
    }
}

const isAdminOrCeo = (req, res, next) => {
    if(req.user && (req.user.type === "ceo" || req.user.type === "admin")) {
        return next();
    }
    else {
        return res.status(401).send({msg: "Token is not valid."});
    }
}

const isAdminOrCeoOrSeller = (req, res, next) => {
        if(req.user  &&  (req.user.type  ===  "ceo"  ||  req.user.type  ===  "admin"  ||
req.user.type === "seller")) {
        return next();
    }
    else {
        return res.status(401).send({msg: "Token is not valid."});
    }
}

export {
    getToken,
    isAuth,
    isAdmin,
    isCeo,
    isAdminOrCeo,
    isAdminOrCeoOrSeller,
} ;
```

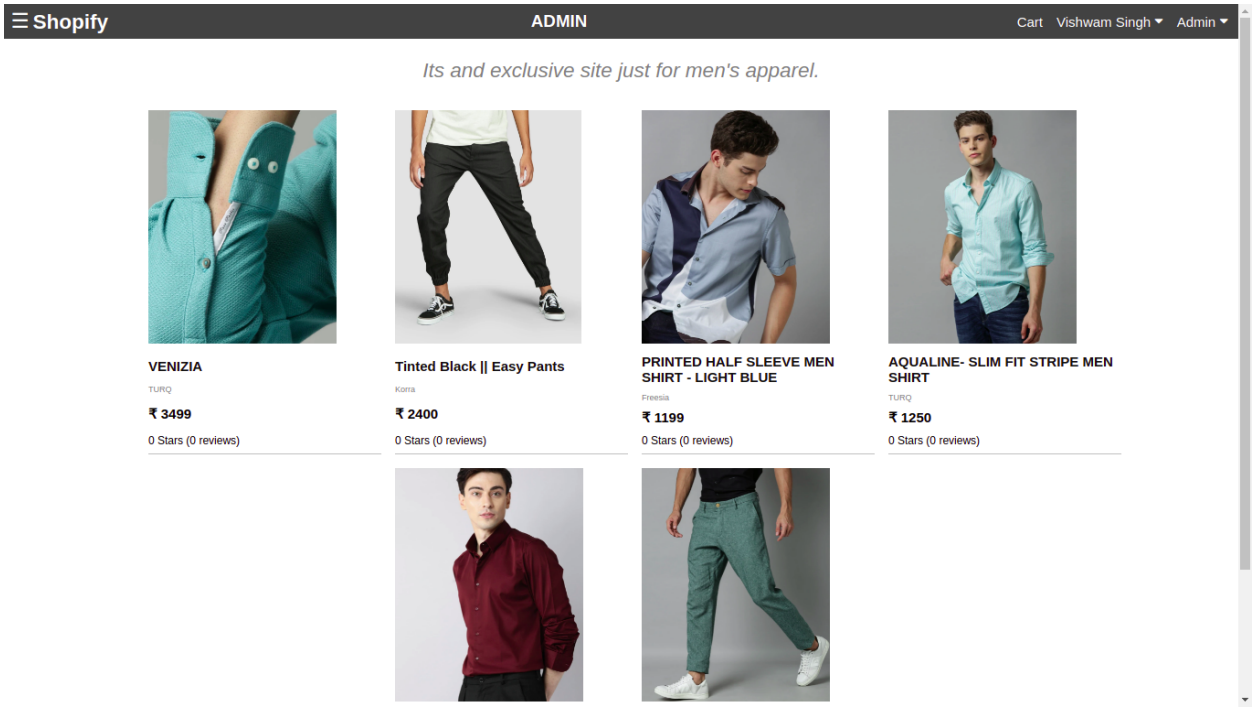
Buyer Login (Screenshot) :



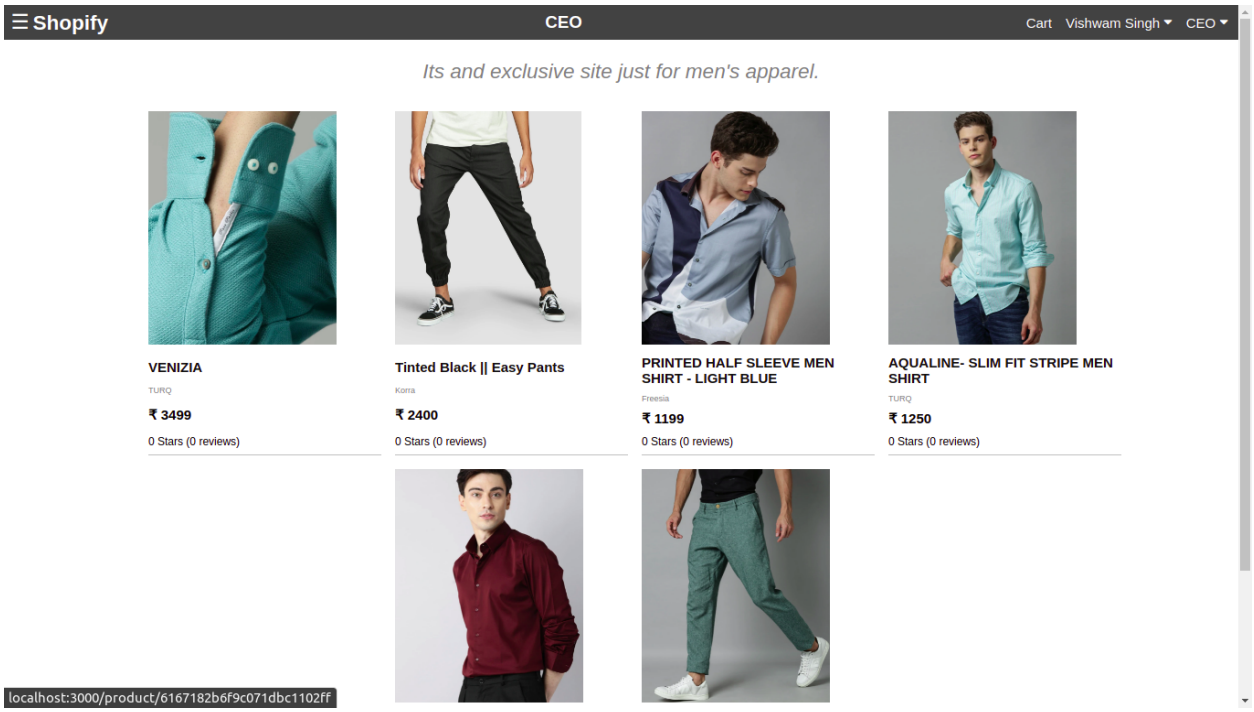
Seller Login (Screenshot) :



Admin Login (Screenshot) :



CEO Login (Screenshot) :



Database Models :

- User Model

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true, dropDups: true },
    password: {type: String, required: true },
    type: { type: String, required: true },
});

const userModel = mongoose.model("user", userSchema);

export default userModel;
```

- Product Model

```
import mongoose from "mongoose";

const productSchema = new mongoose.Schema({
    name: { type: String, required: true },
    image: { type: String, required: true },
    brand: { type: String, required: true },
    category: { type: String, required: true },
    description: { type: String, required: true },
    price: { type: Number, default: 0, required: true },
    rating: { type: Number, default: 0, required: true },
    numReviews: { type: Number, default: 0, required: true },
    countInStock: { type: Number, default: 0, required: true },
    seller: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
        required: true,
    }
});
```

```
const productModel = mongoose.model("product", productSchema);

export default productModel;
```

- Order Model

```
import mongoose from "mongoose";
const orderSchema = mongoose.Schema(
   {
      orderItems: [{
         name: { type: String, requrired: true },
         qty: { type: Number, requrired: true },
         image: { type: String, requrired: true },
         price: { type: Number, required: true },
         product: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Product",
            requrired: true
         },
      }],
      shippingAddress: {
         fullName: { type: String, requrired: true },
         address: { type: String, requrired: true },
         city: { type: String, requrired: true },
         pinCode: {type: Number, require: true },
         country: { type: String, requrired: true },
      },
      paymentMethod: { type: String, required: true },
      itemsPrice: { type: Number, required: true },
      shippingPrice: { type: Number, required: true },
      taxPrice: { type: Number, required: true },
      totalPrice: { type: Number, required: true },
      user: {
         type: mongoose.Schema.Types.ObjectId,
         ref: "User",
         required: true,
      },
      createdAt: { type: Date, required: true },
      isPaid: { type: Boolean, default: false },
      paidAt: { type: Date },
      isDelivered: { type: Boolean, default: false },
      deliveredAt: { type: Date }
   },
   {
```

```
        timestapms: true,
    }
);
const Order = mongoose.model("Order", orderSchema);
export default Order;
```

Backend Routes :

- ### User Routes :

```
import express from "express";
import User from "../models/userModel";
import { getToken, isAdmin, isAuth, isCeo, isAdminOrCeo } from "../util";

const router = express.Router();

router.get("/", async (req, res) => {
    const users = await User.find({});
    res.send(users);
})

router.get("/createadmin", async (req, res) => {
    try {
        const user = await User({
            name: "Vishwam Singh",
            email: "vishwam.mnnit2020@gmail.com",
            password: "123456",
            type: "admin",
        });
        const newUser = await user.save();
        res.send(newUser);

    } catch (error) {
        res.send({msg: error});
    }
});

router.get("/:id", async(req, res) => {
    const user = await User.findById(req.params.id);
    if(user) {
        res.send(user);
    } else {
        res.status(404).send({ msg: "User not found." });
```

```
    }
})

router.post("/signin", async (req, res) => {
    const signinUser = await User.findOne({
        email: req.body.email,
        password: req.body.password
    });

    if(signinUser) {
        res.send({
            _id: signinUser._id,
            email: signinUser.email,
            name: signinUser.name,
            type: signinUser.type,
            token: getToken(signinUser)
        });
    } else {
        res.status(401).send({msg: "Invalid Email or Password."});
    }
});


router.post("/register", async (req, res) => {
    const user = new User({
        name: req.body.name,
        email: req.body.email,
        password: req.body.password,
        type: "buyer"
    });

    const newUser = await user.save();

    if(newUser) {
        res.send({
            _id: newUser._id,
            email: newUser.email,
            name: newUser.name,
            type: newUser.type,
            token: getToken(newUser)
        });
    } else {
        res.status(401).send({msg: "Invalid User Data."});
    }
});
```

```
router.delete("/:id", isAuth, isAdminOrCeo, async (req, res) => {
    const userToDelete = await User.findById(req.params.id);
    if(userToDelete) {
        await userToDelete.remove();
        res.send({ msg: "User Deleted." });
    } else {
        res.send({ msg: "Error in deleting user" });
    }
});

router.put("/profile", isAuth, async (req, res) => {
    const user = await User.findById(req.user._id);
    if(user) {
        user.name = req.body.name || user.name;
        user.email = req.body.email || user.email;
        if(req.body.password) {
            user.password = req.body.password;
        }

        const updatedUser = await user.save()
        res.send({
            _id: updatedUser._id,
            name: updatedUser.name,
            email: updatedUser.email,
            type: updatedUser.type,
            token: getToken(updatedUser),
        });
    }
});

router.post("/createUser", isAuth, isCeo, async (req, res) => {
    const user = new User({
        name: req.body.name,
        email: req.body.email,
        password: req.body.password,
        type: req.body.type,
    });

    const newUser = await user.save();

    if(newUser) {
        res.send({
            _id: newUser._id,
            email: newUser.email,
            name: newUser.name,
            type: newUser.type,
```

```
            token: getToken(newUser)
        });
    } else {
        res.status(401).send({msg: "Invalid User Data."});
    }
});
export default router;
```

- Product Routes :

```
import express from "express";
import Product from "../models/productModel";
import { getToken, isAdmin, isAuth, isAdminOrCeo, isAdminOrCeoOrSeller }
from "../util";

const router = express.Router();

router.get("/all", async (req, res) => {
    console.log(req.params);
    const products = await Product.find({});
    res.send(products);
});

router.get("/seller/:id", async (req, res) => {
    console.log(req.params.id);
    if(req.params.id) {
        const products = await Product.find({ seller: req.params.id });
        res.send(products);
    } else {
        res.send({msg: "Invalid User"});
    }

});

router.get("/:id", async (req, res) => {
    const product = await Product.findOne({_id: req.params.id});
    if(product) {
        res.send(product);
    } else {
        res.status(404).send({msg: "Product not found"});
    }
});

router.post("/", isAuth, isAdminOrCeoOrSeller, async (req, res) => {
```

```
    const product = new Product({
        name: req.body.name,
        image: req.body.image,
        brand: req.body.brand,
        category: req.body.category,
        description: req.body.description,
        price: req.body.price,
        rating: req.body.rating,
        numReviews: req.body.numReviews,
        countInStock: req.body.countInStock,
        seller: req.body.seller,
    });

    const newProduct = await product.save();
    if(newProduct) {
                return res.status(201).send({ msg: "New product created", data: newProduct });
    }
    return res.status(500).send({ msg: "Error in creating the product" });
});

router.put("/:id", isAuth, isAdminOrCeoOrSeller, async (req, res) => {
    const productId = req.params.id;
    const product = await Product.findById(productId);
    if(product) {
        product.name = req.body.name;
        product.image = req.body.image;
        product.brand = req.body.brand;
        product.category = req.body.category;
        product.description = req.body.description;
        product.price = req.body.price;
        product.countInStock = req.body.countInStock;
        const updatedProduct = await product.save();
        if(updatedProduct) {
                        return res.status(200).send({ msg: "Product Updated", data: updatedProduct });
        }
    } else {
        return res.status(500).send({ msg: "Error in updating the product" });
    }
});

router.delete("/:id", isAuth, isAdminOrCeoOrSeller, async (req, res) => {
    const productToDelete = await Product.findById(req.params.id);
    if(productToDelete) {
        await productToDelete.remove();
```

```
        res.send({ msg: "Product Deleted." });
    } else {
        res.send({ msg: "Error in deletion." })
    }
});

export default router;
```

- Order Routes :

```
import express from "express";
import Order from "../models/orderModel";
import mongoose from "mongoose";
import { isAdmin, isAuth, isAdminOrCeo } from "../util";
const router = express.Router();
router.get("/all", isAuth, isAdminOrCeo, async (req, res) => {
    const orders = await Order.find({});
    res.send(orders);
});

router.get("/user", isAuth, async (req, res) => {
    const orders = await Order.find({ user: req.user._id });
    res.send(orders);
});

router.post("/", isAuth, async (req, res) => {
    if(req.body.orderItems.length === 0) {
        res.status(400).send( {msg: "Cart is Empty."} );
    } else {
        const order = new Order({
            orderItems: req.body.orderItems,
            shippingAddress: req.body.shippingAddress,
            paymentMethod: req.body.paymentMethod,
            itemsPrice: req.body.itemsPrice,
            taxPrice: req.body.taxPrice,
            shippingPrice: req.body.shippingPrice,
            totalPrice: req.body.totalPrice,
            createdAt: req.body.createdAt,
            user: req.user._id,
        });
        const createdOrder = await order.save();
        res.status(201).send(createdOrder);
```

```
    }
});
router.get("/:id", isAuth, async (req, res) => {
    const order = await Order.findById(req.params.id);
    if(order) {
        res.status(201).send(order);
    } else {
        res.status(404).send({ msg: "Order not found." });
    }
});
export default router;
```

Back-End Server :

```
import express from "express";
import data from "./data";
import dotenv from "dotenv";
import config from "./config";
import mongoose from "mongoose";
import userRoute from "./routes/userRoute";
import productRoute from "./routes/productRoute";
import orderRoute from "./routes/orderRoute";
import bodyParser from "body-parser";

dotenv.config();

const mongodbUrl = config.MONGODB_URL;
mongoose.connect(mongodbUrl, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
})
.then(() => console.log("DB connected!"))
.catch(error => console.log(error));

const app = express();

app.use(bodyParser.json());

app.use("/api/users", userRoute);
app.use("/api/products", productRoute);
app.use("/api/orders", orderRoute);
```

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

const port = 8080;

app.listen(port, () => {
    console.log(`Server is listening at http://localhost:${port}`);
});
```

Redux Store :

```
import { createStore, combineReducers, applyMiddleware, compose } from "redux";
import thunk from "redux-thunk";
import Cookie from "js-cookie";
import { cartReducer } from "./reducers/cartReducers";
import { productListReducer, productDetailsReducer, productSaveReducer, productDeleteReducer } from "./reducers/productReducers";
import { createUserReducer, userDeleteReducer, userDetailsReducer, userListReducer, userRegisterReducer, userSigninReducer, userUpdateProfileReducer } from "./reducers/userReducers";
import { orderCreateReducer, orderDetailsReducer, orderListReducer, orderUserListReducer } from "./reducers/orderReducers";

const cartItems = (Cookie.get("cartItems") && JSON.parse(Cookie.get("cartItems"))) || [];
const userInfo = (Cookie.get("userInfo") && JSON.parse(Cookie.get("userInfo"))) || "";

const initialState = {
    cart : {
            cartItems,
            shippingAddress : Cookie.get("shippingAddress")
            ? JSON.parse(Cookie.get("shippingAddress"))
            : {},
            paymentMethod: Cookie.get("paymentMethod")
```

```
          ? JSON.parse(Cookie.get("paymentMethod"))
          : "Cash On Delivery"
      },
      userSignin : { userInfo }
  };

const reducer = combineReducers({
  productList: productListReducer,
  productDetails: productDetailsReducer,
  cart: cartReducer,
  userSignin: userSigninReducer,
  userRegister: userRegisterReducer,
  productSave: productSaveReducer,
  productDelete: productDeleteReducer,
  userList: userListReducer,
  userDelete: userDeleteReducer,
  orderCreate: orderCreateReducer,
  orderDetails: orderDetailsReducer,
  orderUserList: orderUserListReducer,
  userDetails: userDetailsReducer,
  userUpdateProfile: userUpdateProfileReducer,
  orderList: orderListReducer,
  createUser: createUserReducer,
});

const composeEnhancer = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;
const store = createStore(reducer, initialState, composeEnhancer(applyMiddleware(thunk)));
export default store;
```

RESULTS :

The requirements of all the assignments were successfully met and the final product still lacks certain web features such as POS integration and order delivery status change.

Features Implemented :

- User can view his profile and update his/her information.
- Sellers can add and update products.

- Admin can add and update products and add and delete sellers or buyers.
- CEOs can add and update products and add and delete sellers or buyers or admins.
- Change the quantity of products in the checkout screen.

Features to be Implemented :

- POS integration.
- Implementation of payment through payment gateways such as paypal and PhonePe.
- Updation of delivery and payment status.
- Adding more categories and subcategories to the apparels for sale.
- Adding api automation for data fetch.
- Deploying on a real web server.

CONCLUSION

- This project enables one think about the immense possibility in the field of POS integration and trading online.
- E-Commerce websites are the future of technology.
- E-commerce is powered by the internet, where customers can access an online store to browse through, and place orders for products or services via their own devices.
- Today there are so many websites where various e-commerce methods are practiced such as Amazon, Flipkart.
- Subscription facilities have grown in recent years due to Covid-virus.
- Trading using crypto currency has a lot of growth with high risks involved, but people still do trading using crypto currency.

REFERENCES :

- NodeJs
- ExpressJs
- ReactJs
- ReduxJs
- React Router Dom
- Babel
- Shopify.in