BRAC UNIVERSITY

Inspiring Excellence

Project Title: Papier Factory

Table of Content

Sl. no	Content	Page no.
01	Introduction	03
02	Project Requirements	04
03	User Manual	05
04	Front End Development	27
05	Back End Development	50
06	Technology (Framework, Languages)	67
07	GitHub Repository Link	67
08	Individual Contributions	67

Introduction

The advent of digital platforms has revolutionized the way we engage with commerce, bridging distances and fostering accessibility. In this context, 'Papier Factory' emerges as a comprehensive e-commerce solution tailored to cater to the diverse needs of customers seeking quality stationery and reading materials. This report delves into the meticulous design and development of Papier Factory, an intuitive and feature-rich online platform aimed at enhancing the shopping experience for users across the spectrum.

Project Overview

Papier Factory embodies a multifaceted approach, seamlessly integrating modules to ensure a robust and user-friendly interface. Divided into distinct yet interconnected modules, this e-commerce website caters to various stakeholders, encompassing User Authentication and Authorization, Product Management, Seller Management, and Reviews and Ratings.

Module Highlights

Central to fostering trust and security, this module incorporates user registration, login/logout functionalities, comprehensive user profiles, authorization, access control, and essential features like password reset and recovery. Upholding user privacy and control is paramount within Papier Factory's framework.

Efficiently organized into categories, the product management module encapsulates the essence of Papier Factory, providing users with a seamless browsing experience. Detailed product pages, shopping cart integration, streamlined checkout processes, event showcases, wish lists, related product suggestions, and quick view options collectively enhance the purchasing journey.

Empowering sellers is at the core of Papier Factory's ethos. This module facilitates seamless onboarding through seller registration, product and event creation, discount code management, seller profile visibility, order confirmation, and shop updates, ensuring a symbiotic relationship between sellers and the platform.

Additional features such as sorting, filtering, messaging to sellers, discount code applications, and order tracking enhance user engagement and satisfaction.

Conclusion

Papier Factory's development underscores a commitment to combining functionality with user-centric design, culminating in an immersive and convenient online shopping experience. This report chronicles the intricacies of each module, underscoring the dedication and innovation invested in bringing the Papier Factory to fruition.

Project Requirements:

Module 1: User Authentication and Authorization

1. User Registration
2. Login/ Logout
3. User Profiles
4. Authorization and Access Control
5. Password Reset and Recovery
6. Update Profile

Module 2: Product Management

1. Product Category
2. Product Detail Page
3. Shopping Cart
4. Checkout Process
5. Events
6. Product wishlist
7. Related Product Suggestion
8. Product Quick View

Module 3: Seller Management

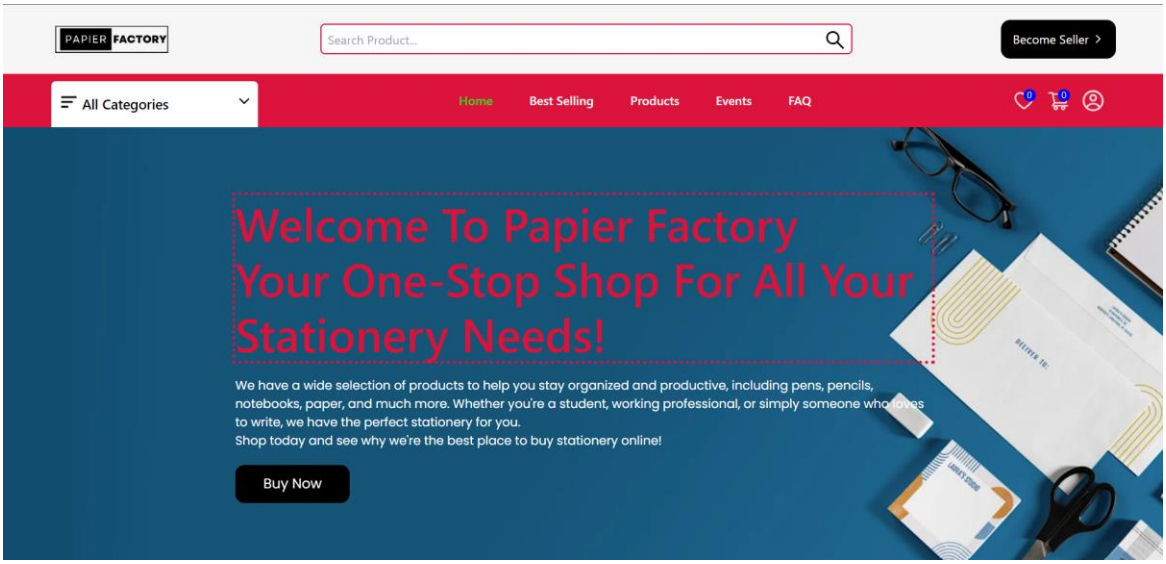
1. Seller Registration
2. Create & View Products
3. Create & View Events
4. Create Discount Code
5. View Seller Profile
6. Order Confirm
7. Update Shop

Module 4: Reviews and Ratings

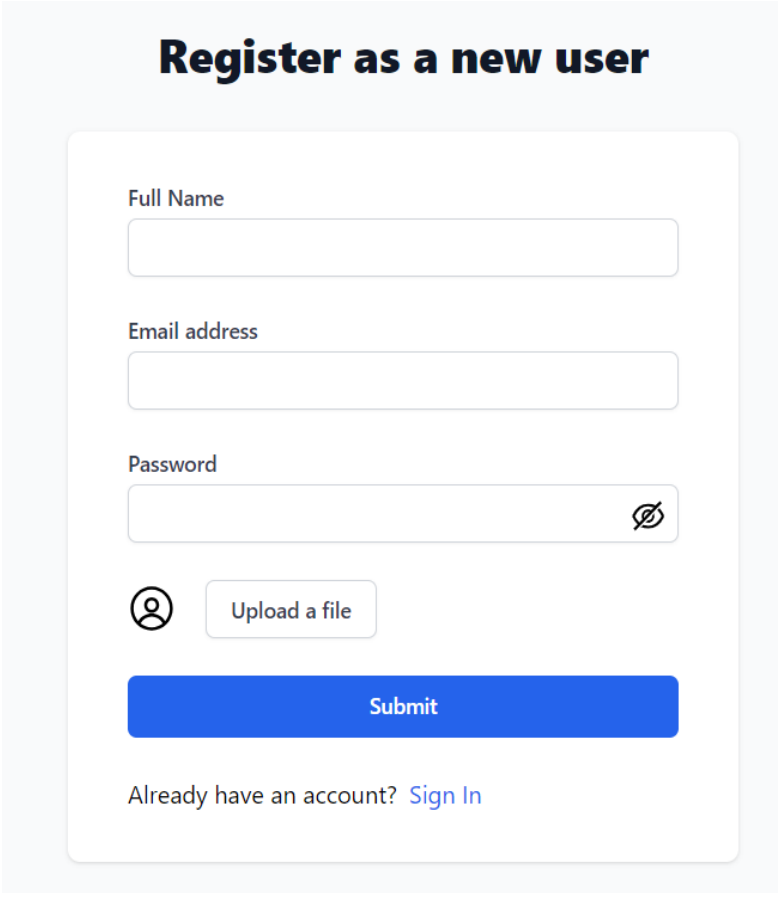
1. Sort and Filter
2. Message to Seller
3. Apply Discount Code
4. Trace & Track Order

USER MANUAL

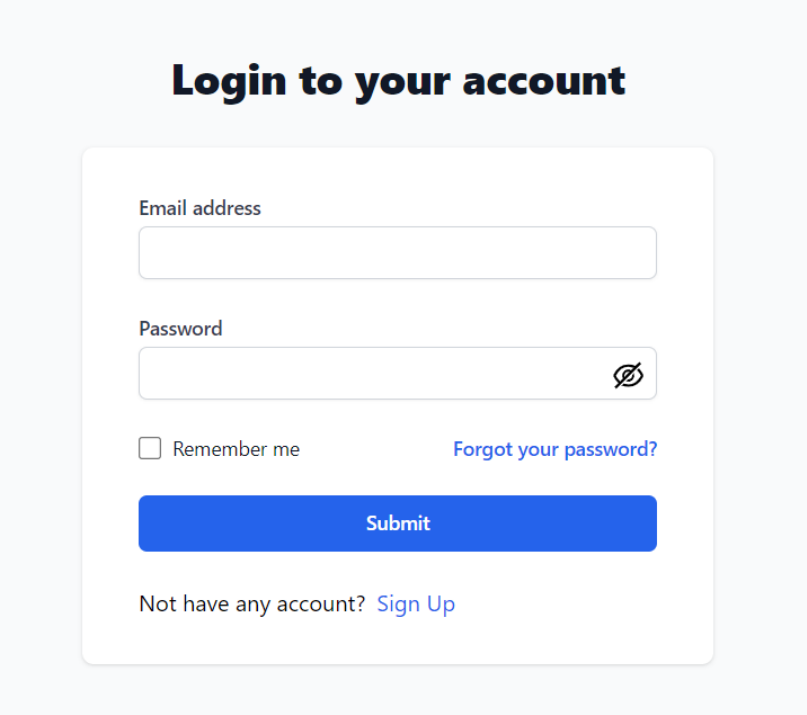
Homepage: This is the homepage view



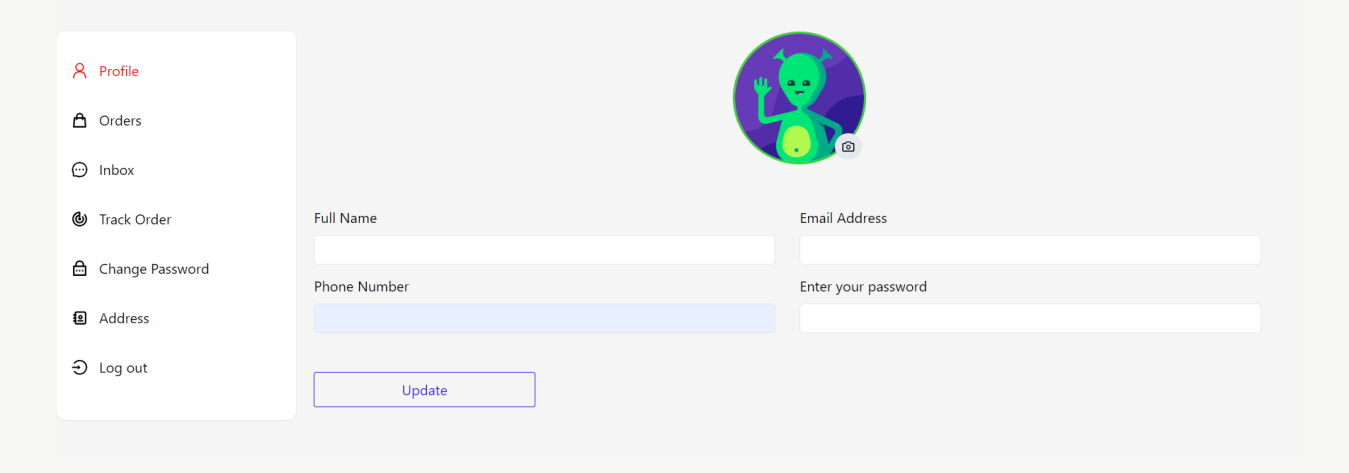
Signup: If they're not registered yet, they can sign up by clicking the sign-up button.



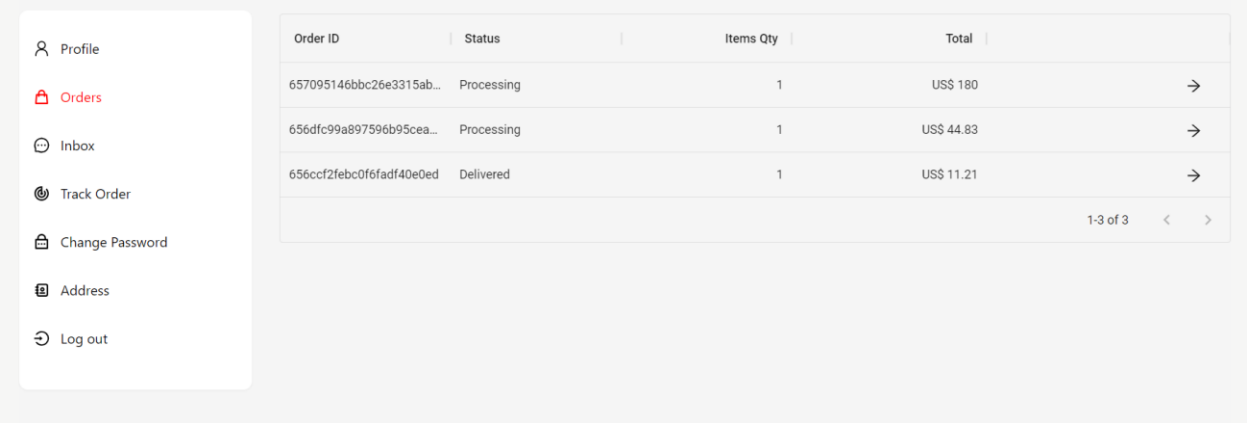
Sign-in: Registered Users can log in using their credentials here.



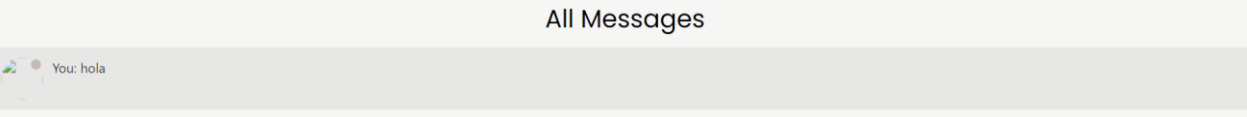
Profile: Here one can see their profile.



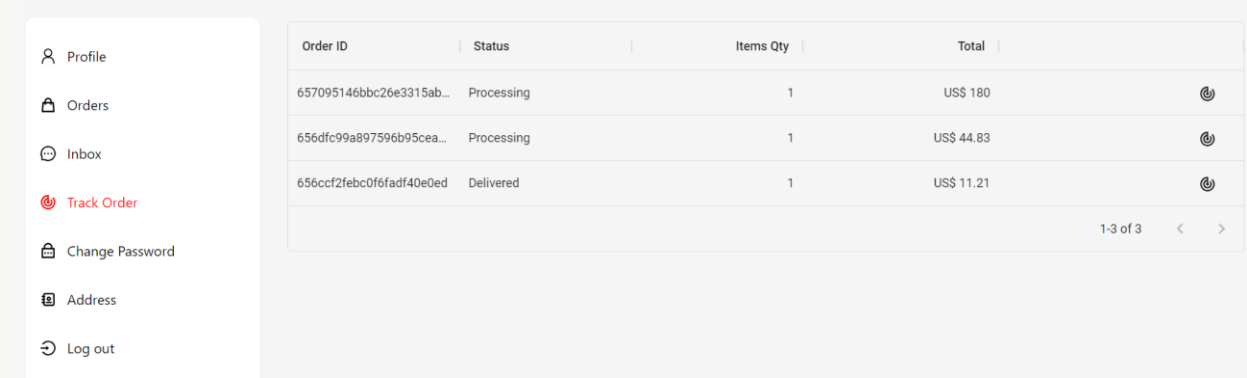
Orders: From here the customer can see the products he/she ordered.



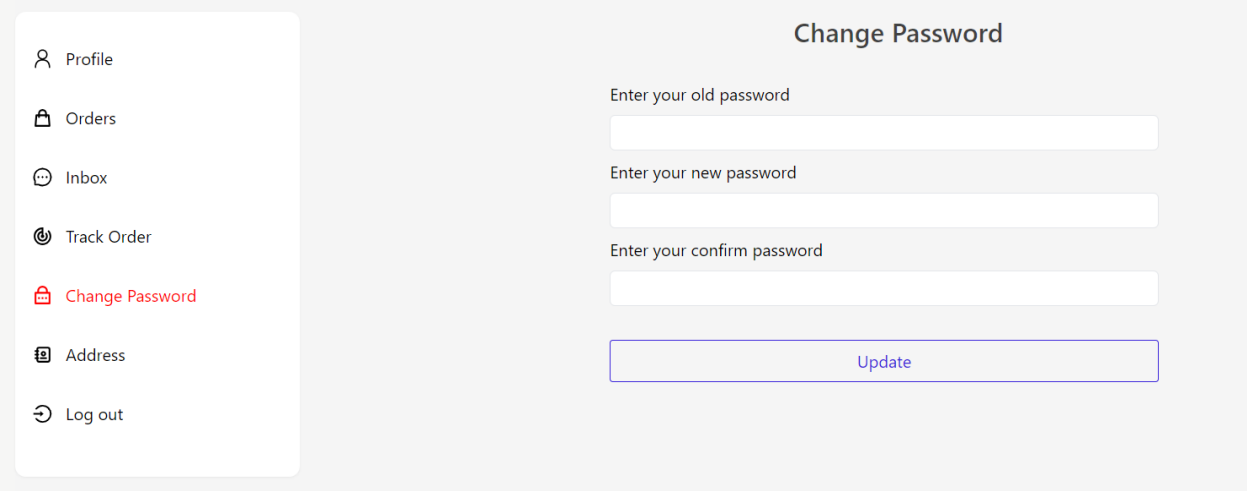
Inbox: From here the customer can see his inbox for any messages.



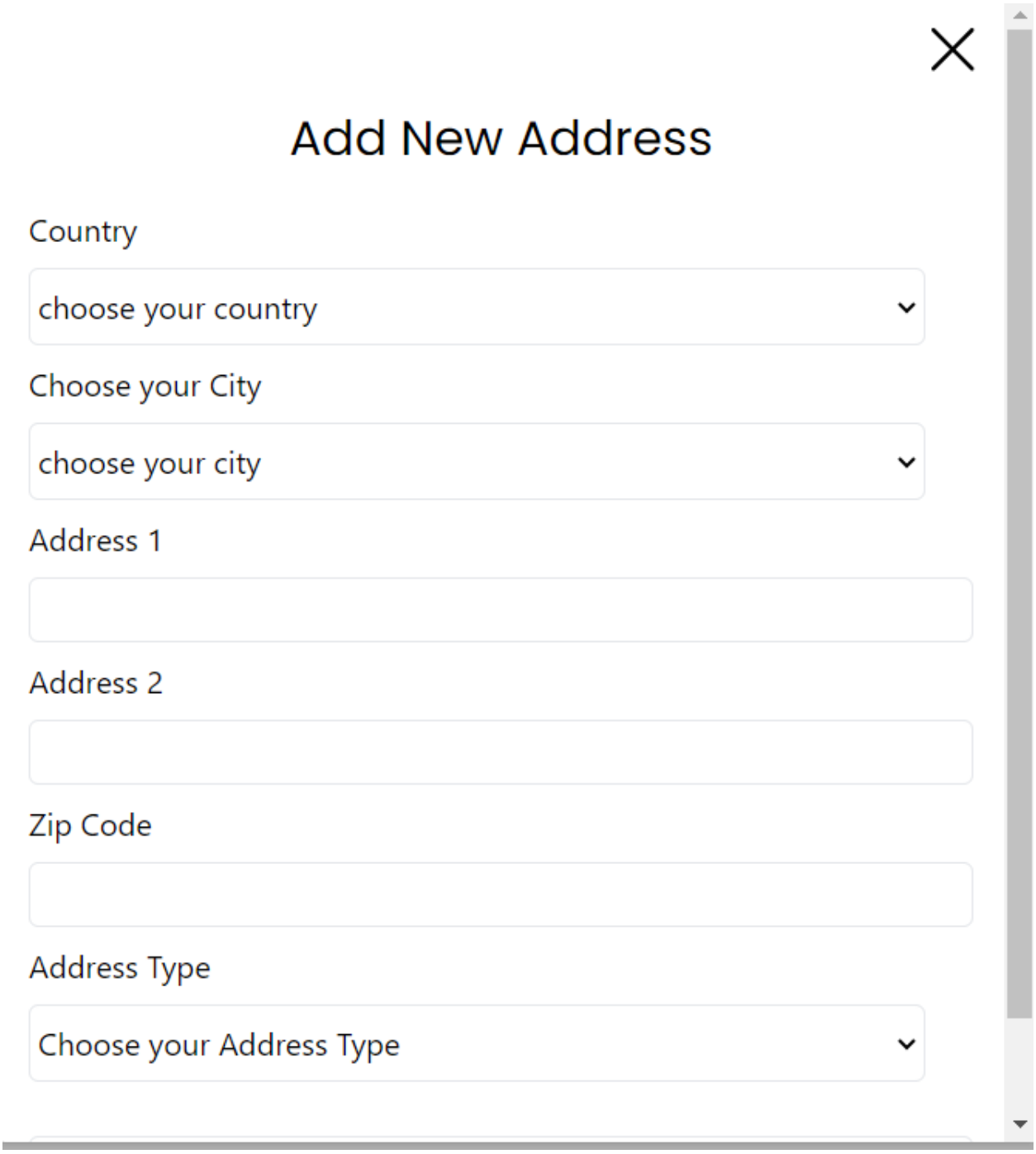
Track Order: From here the customer can track his orders.



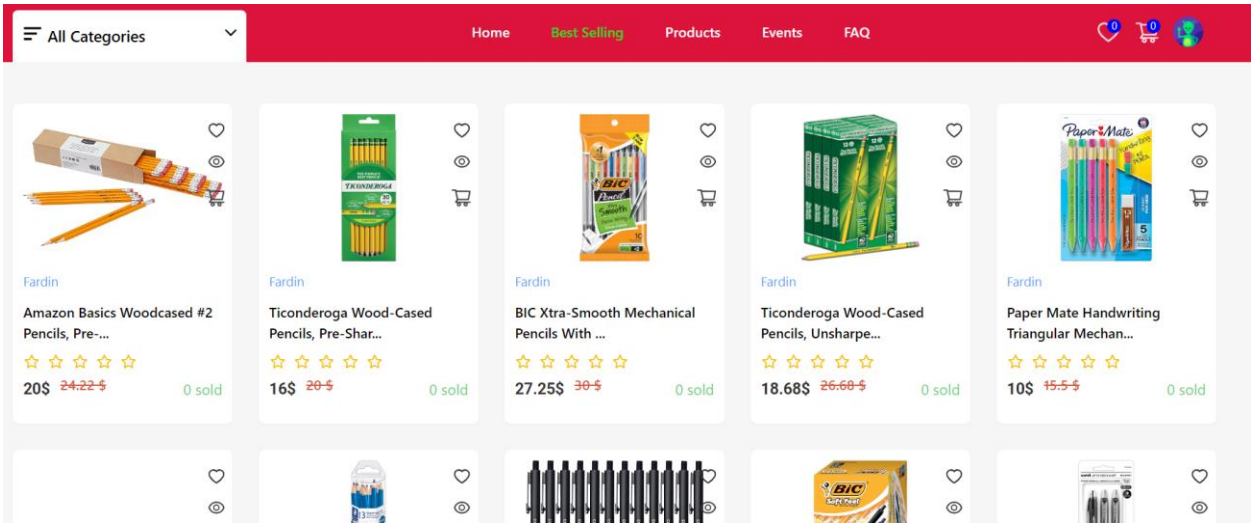
Change Password: The customer can change the password for his account from here.



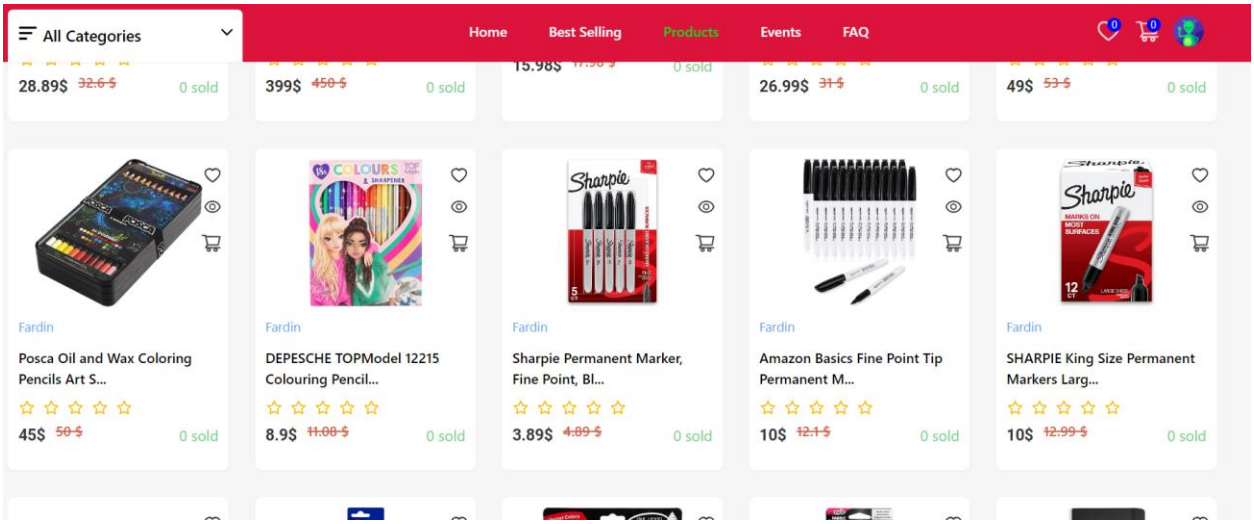
Address: The customer can view his address



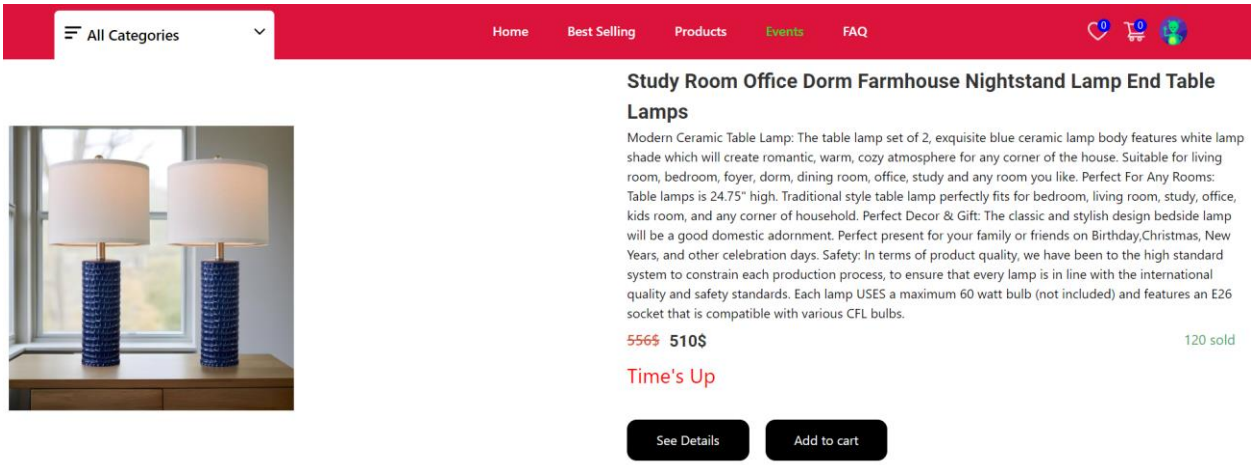
Best Selling: From here the customer can see the current best-selling products on this website.



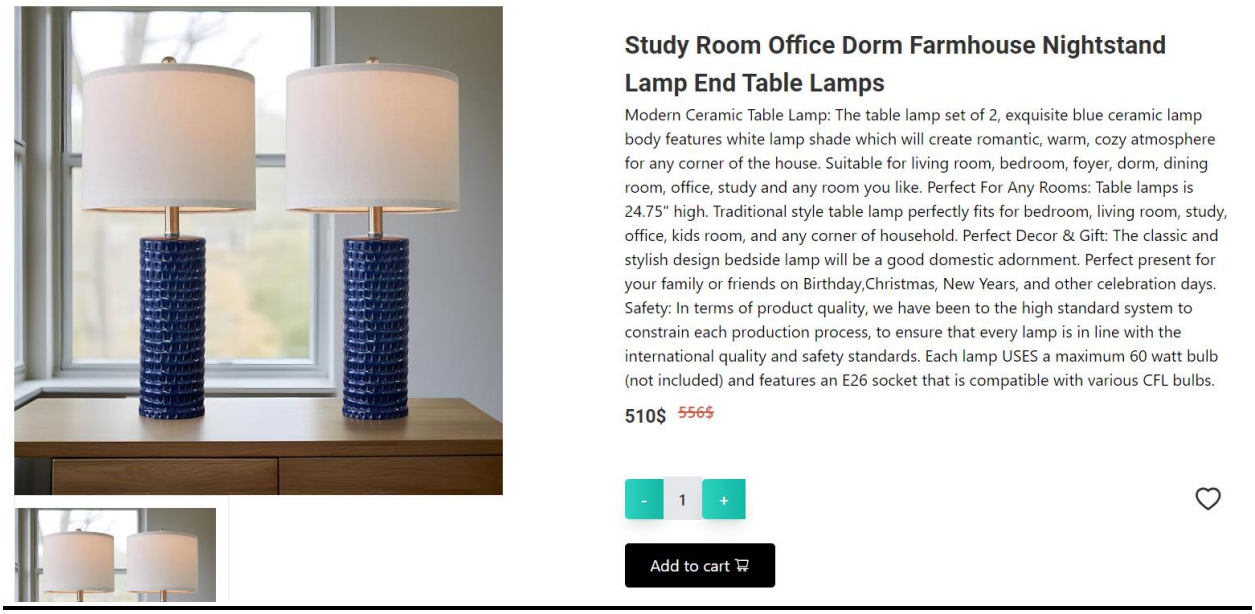
Product: From here the customer can see the products which are available currently.



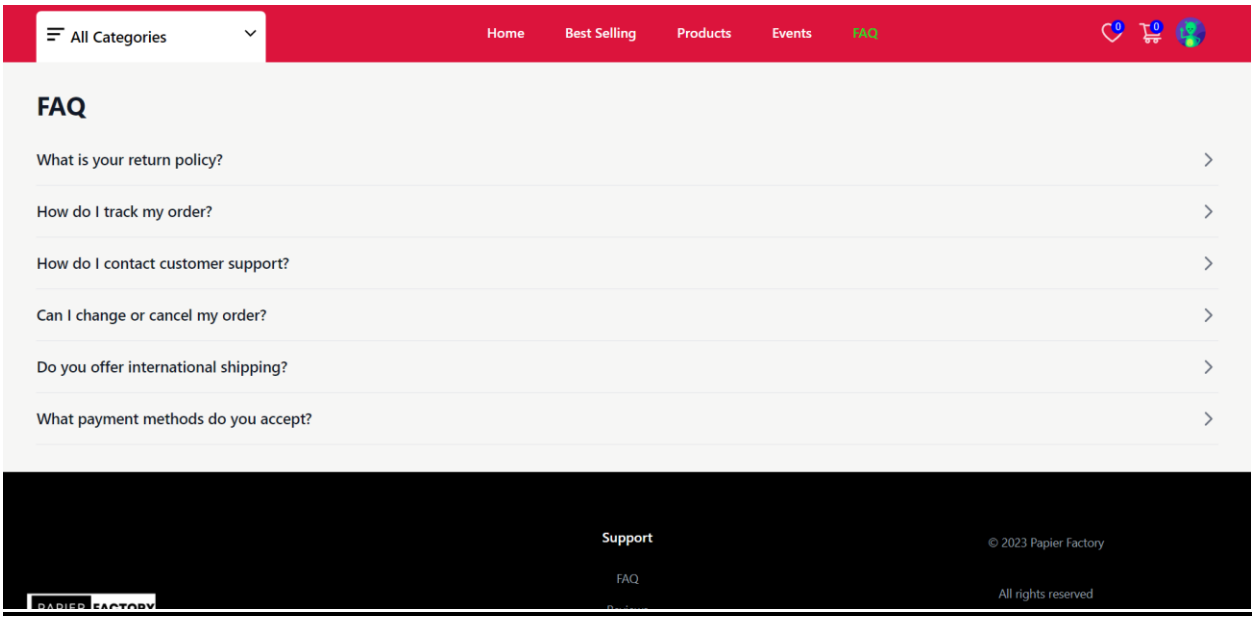
Events: Any current events that are happening right now.



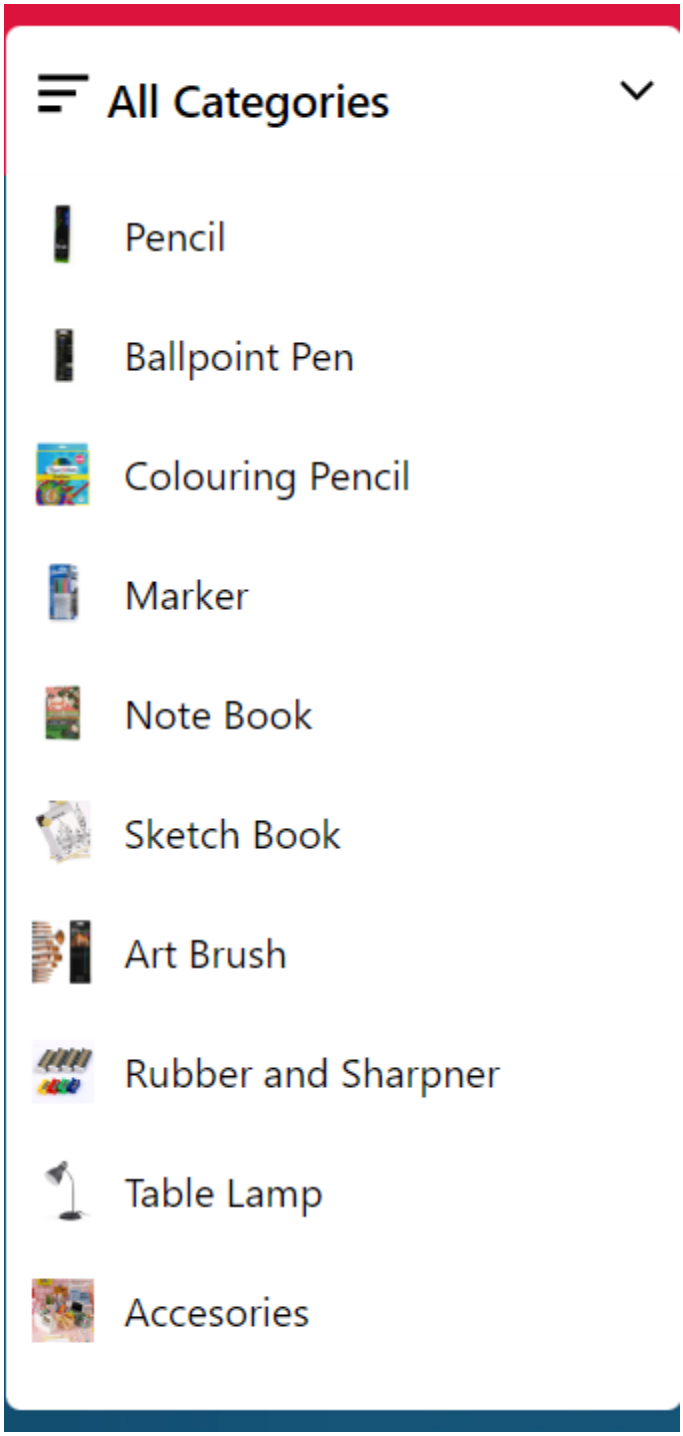
Event Product Details:



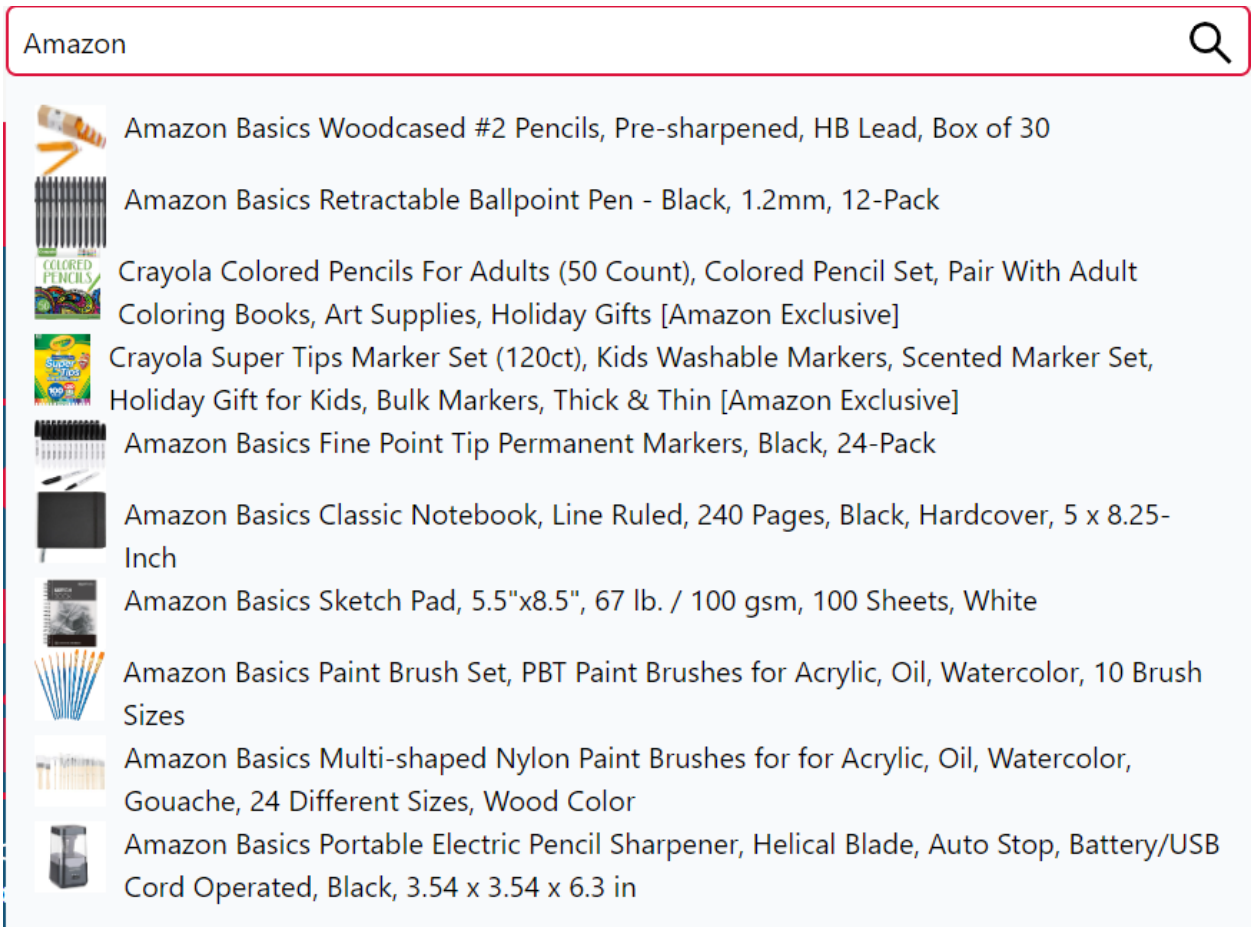
FAQ:


All Categories
Home
Best Selling
Products
Events
FAQ
FAQ
What is your return policy?
How do I track my order?
How do I contact customer support?
Can I change or cancel my order?
Do you offer international shipping?
What payment methods do you accept?
Support
FAQ
© 2023 Papier Factory
All rights reserved

All Catagories: Users can click categories from the shop dropdown to get a view of the categories.

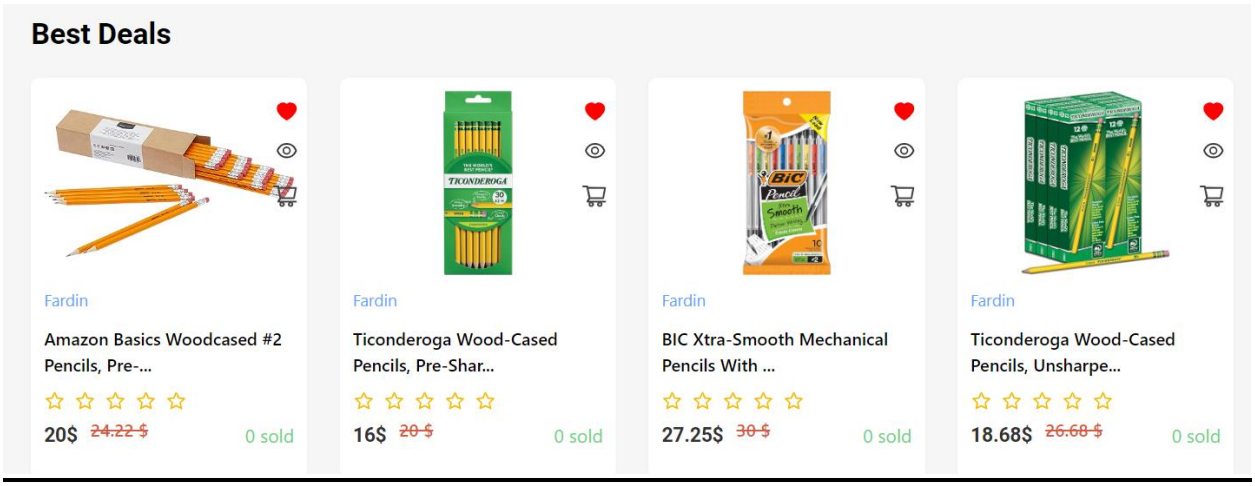


Search Box: From here the customer can search his desired products.

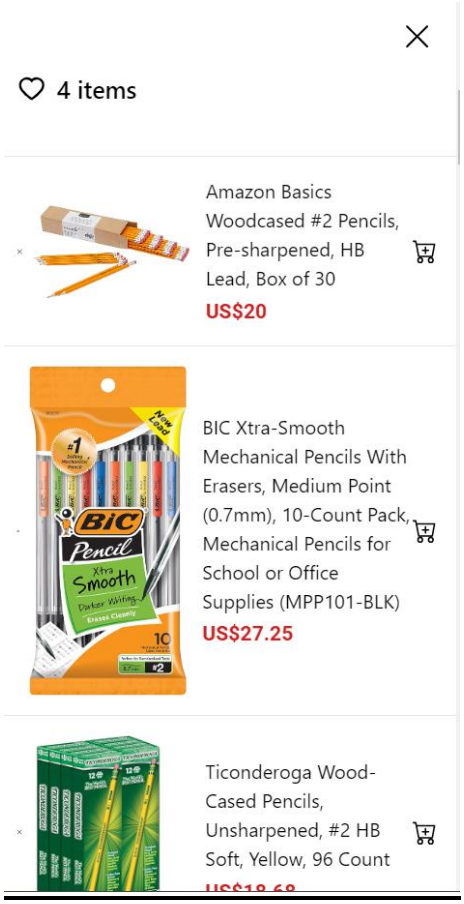


Quick View: Customers can tap on a product to see the full description of the product.

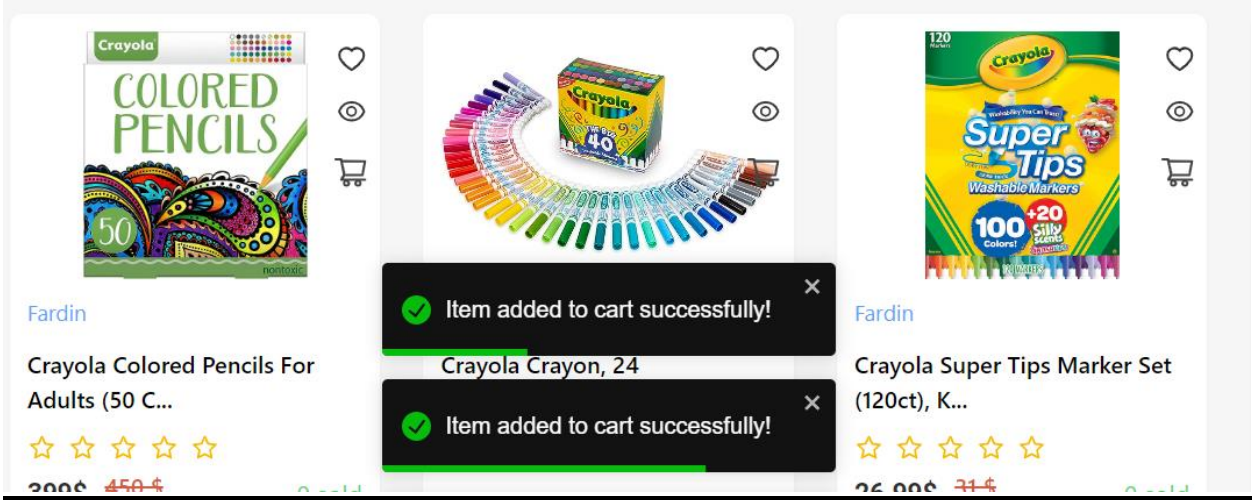
Wishlist: The customer can add their desired products to the wishlist.



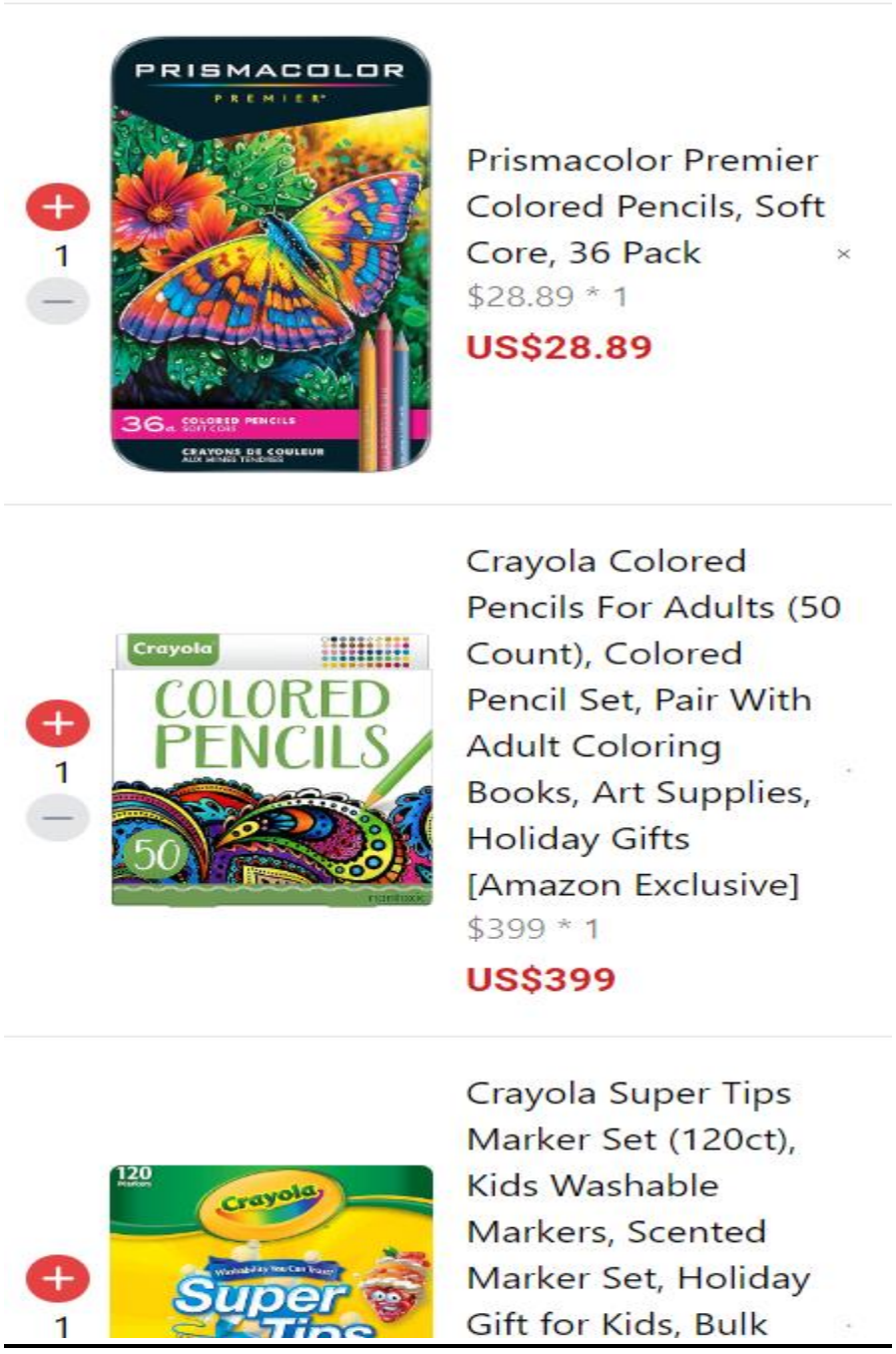
Wishlist add: After adding their products to the wishlist they can see their products in this feature.



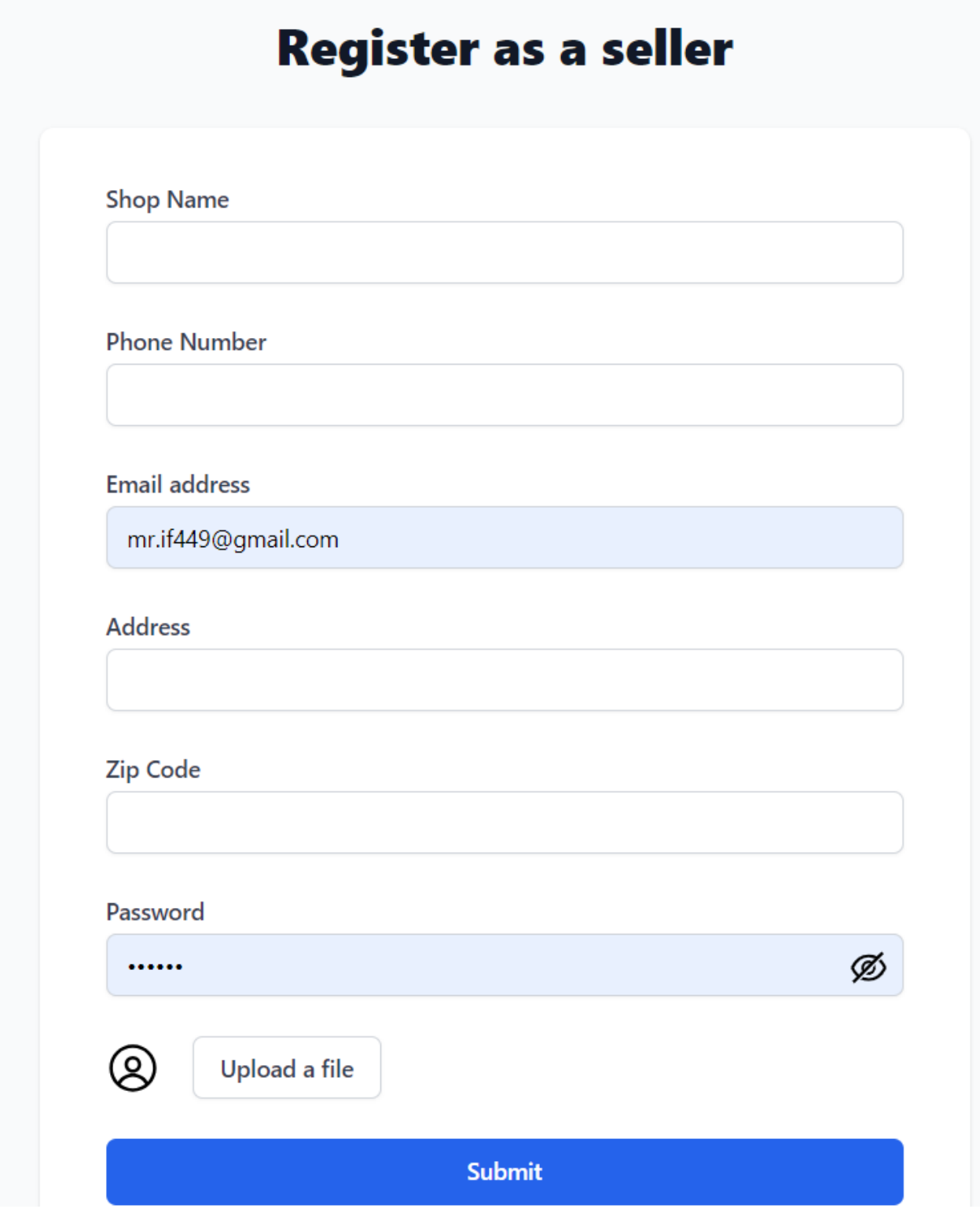
Product Cart: The customer can add their desired products in the cart to buy.



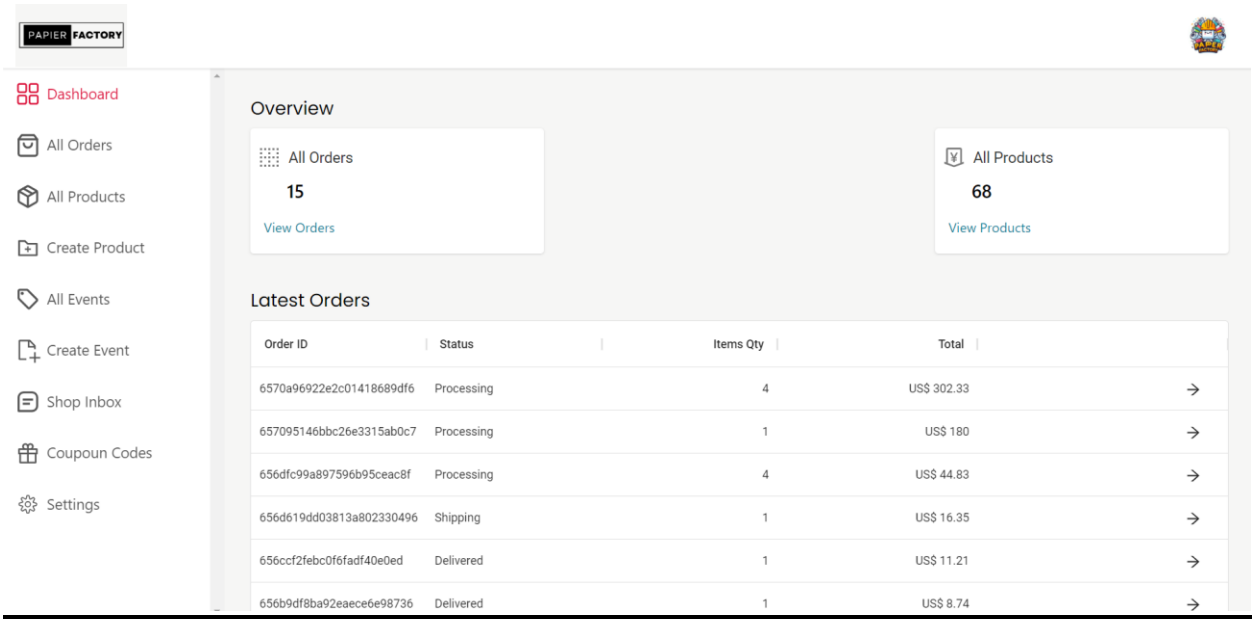
Product Cart Add: After adding their products to the cart, they can see their products in this feature.



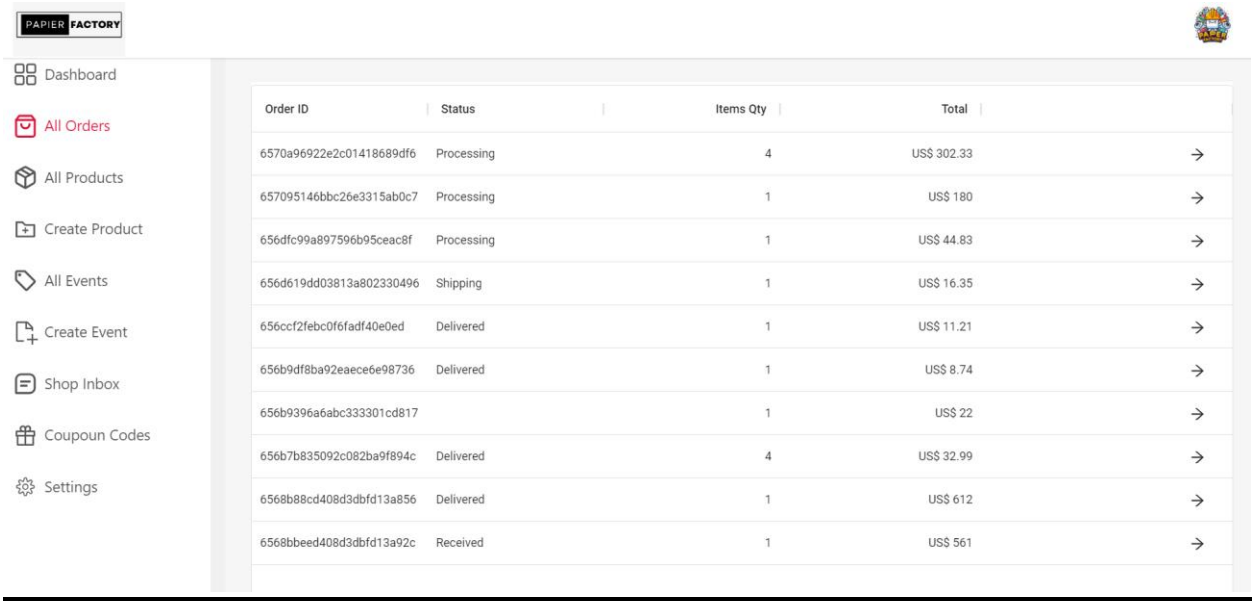
Seller Signup: This process is for sellers who want to sell their products through this website.



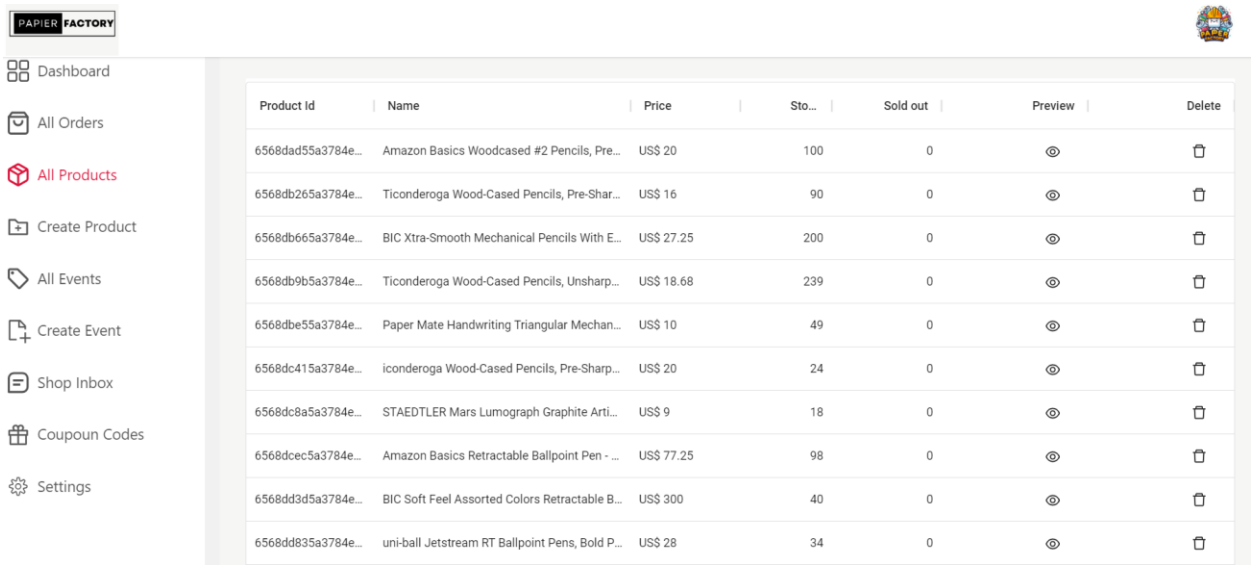
Dashboard: This is the dashboard for the seller to use.



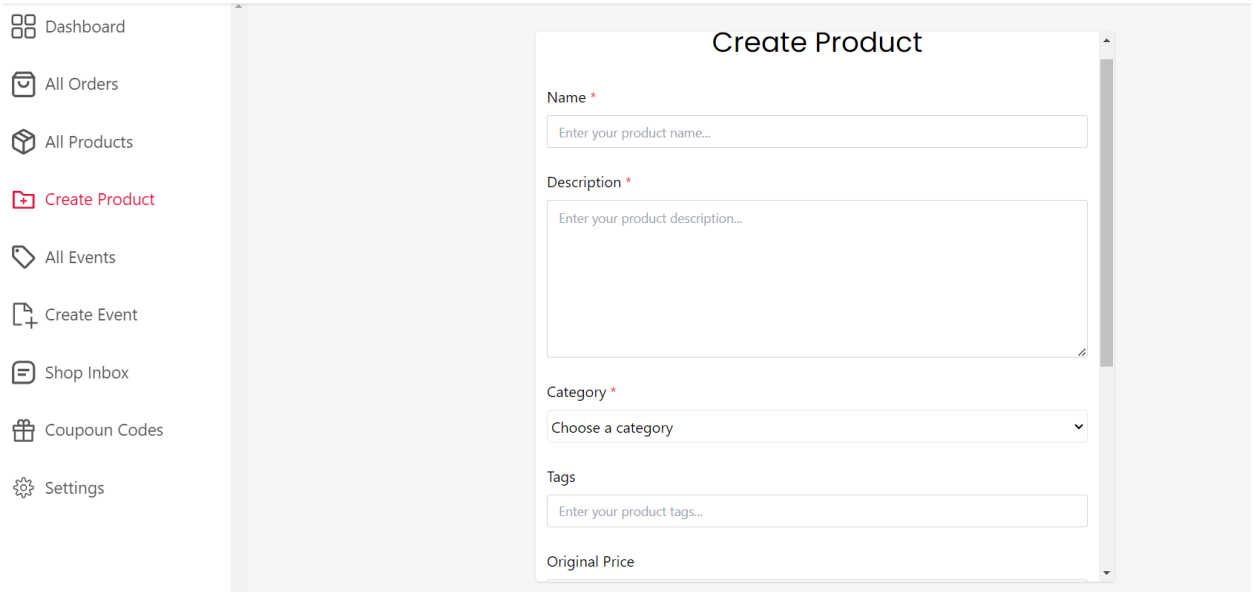
Seller All Orders: The seller can see the products that the customers ordered.



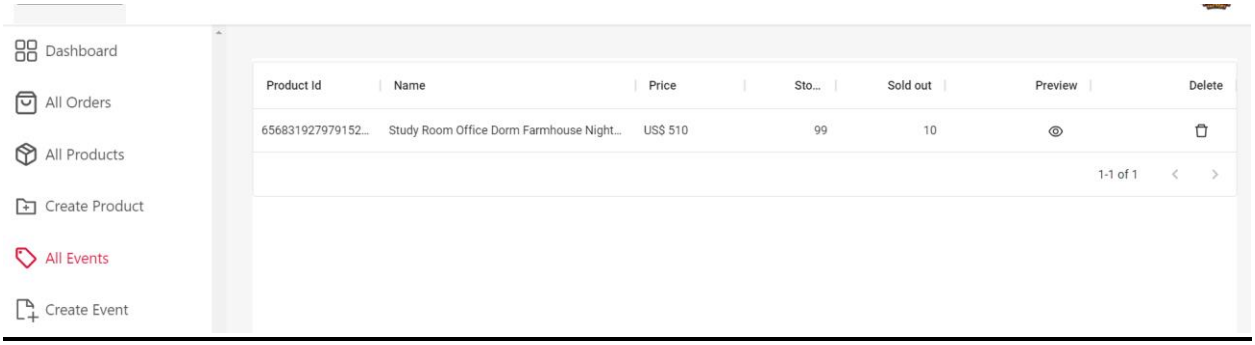
Seller All Products: The seller can see all the products he enlisted.



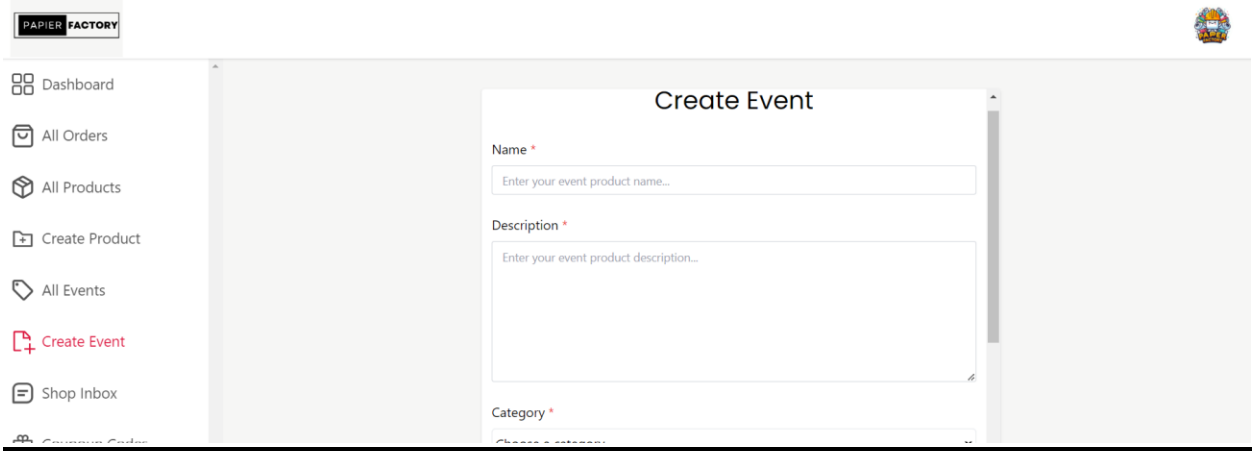
Seller Create Product: The Seller can create new products from this site.



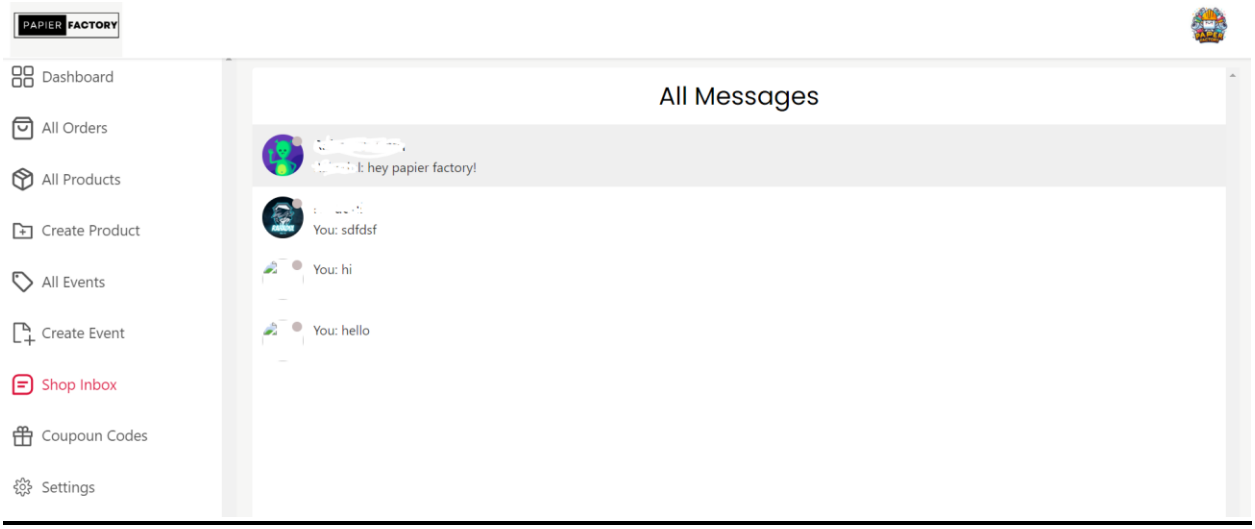
Seller All Events: The seller can see the events that other sellers are giving.



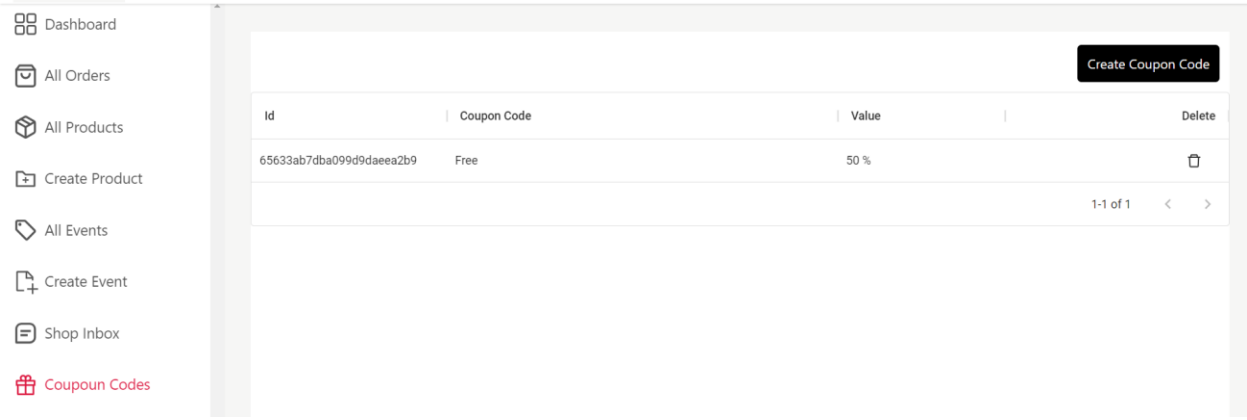
Seller Create Event: The seller can create events for his products.



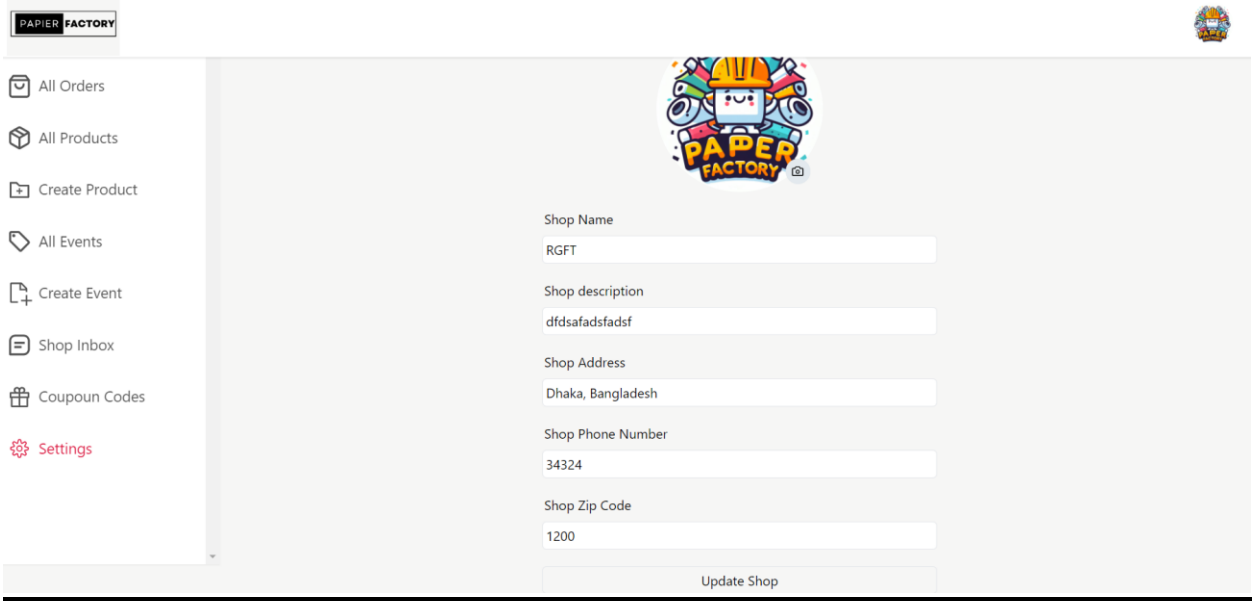
Seller Shop Inbox: The seller can see the messages from customers that was sent to the seller to know a particular item.



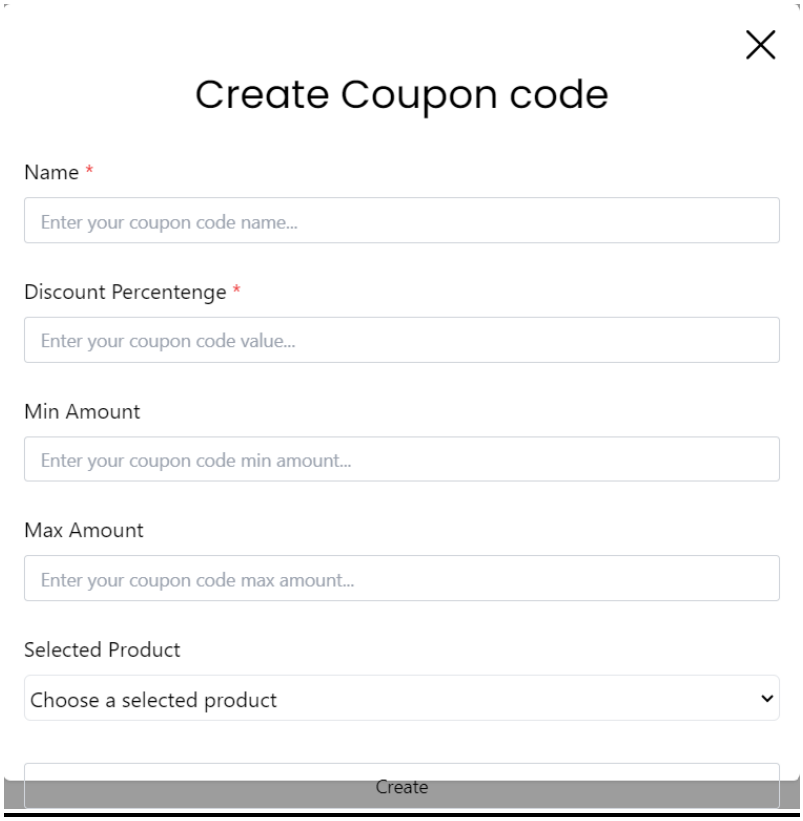
Seller Coupon Codes: The seller can see the coupon codes from here.



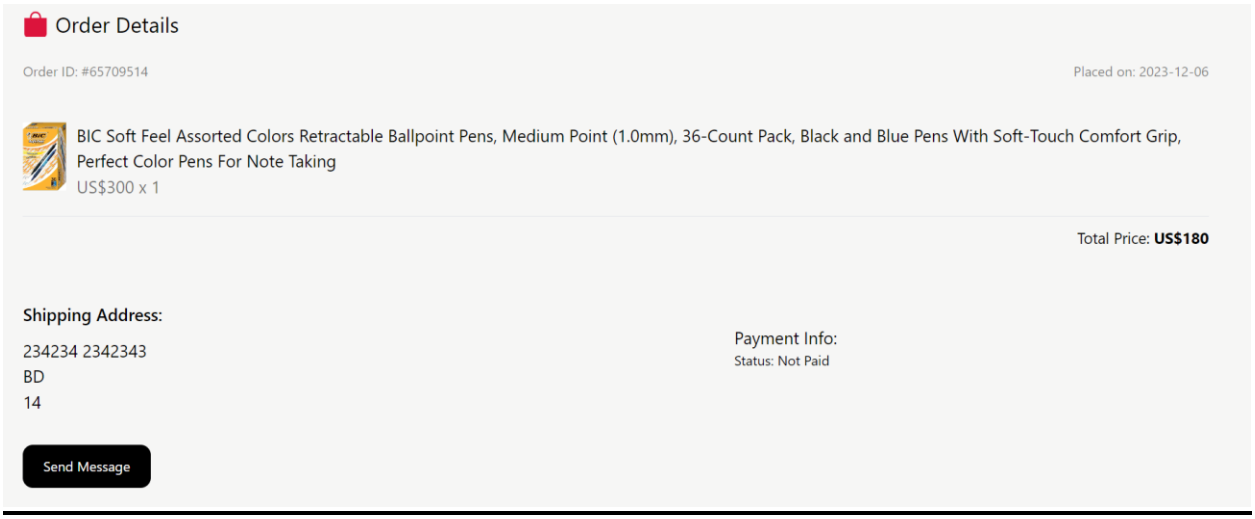
Settings: From here the seller change his shop name or others.



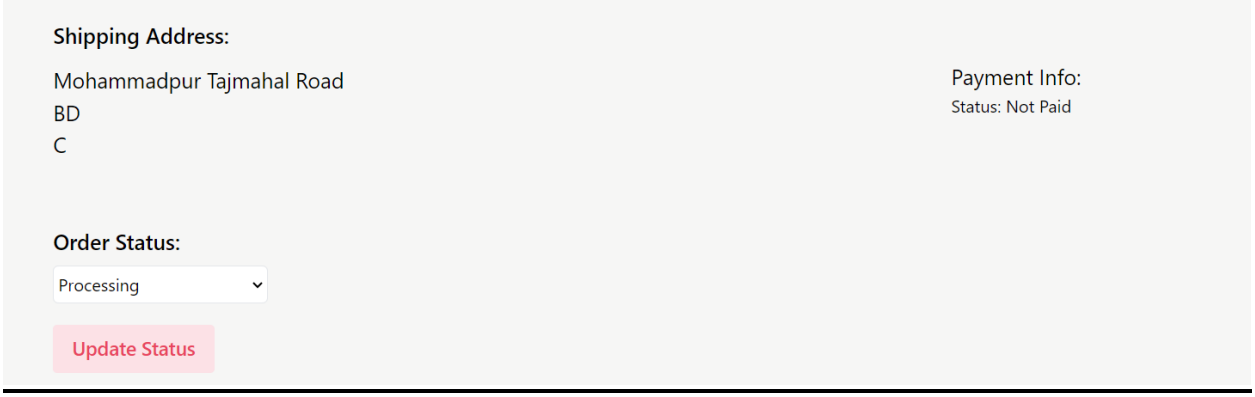
Seller Create Coupon: The seller can create new coupons for his products



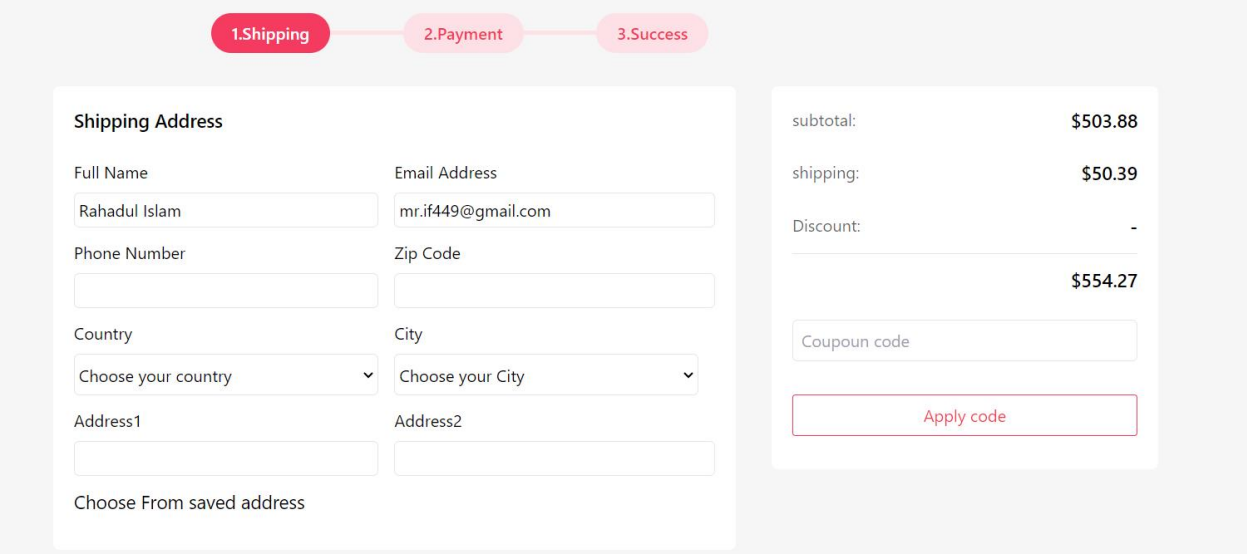
All Orders: The seller can see the orders for his products.



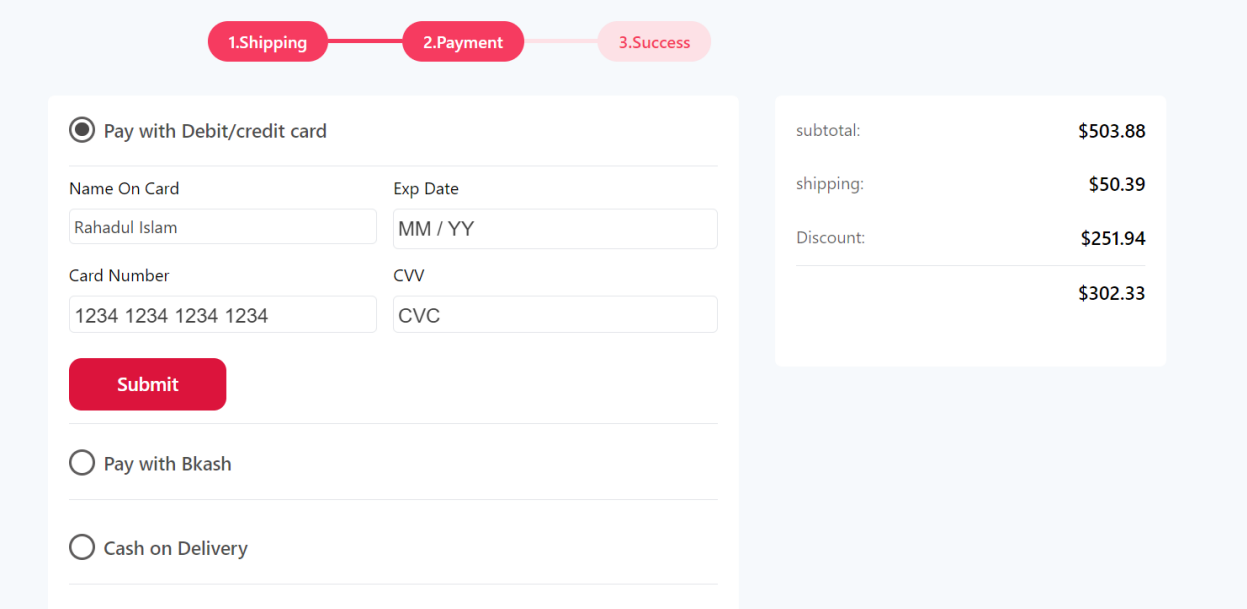
Order Status: From here, the seller can update the order status



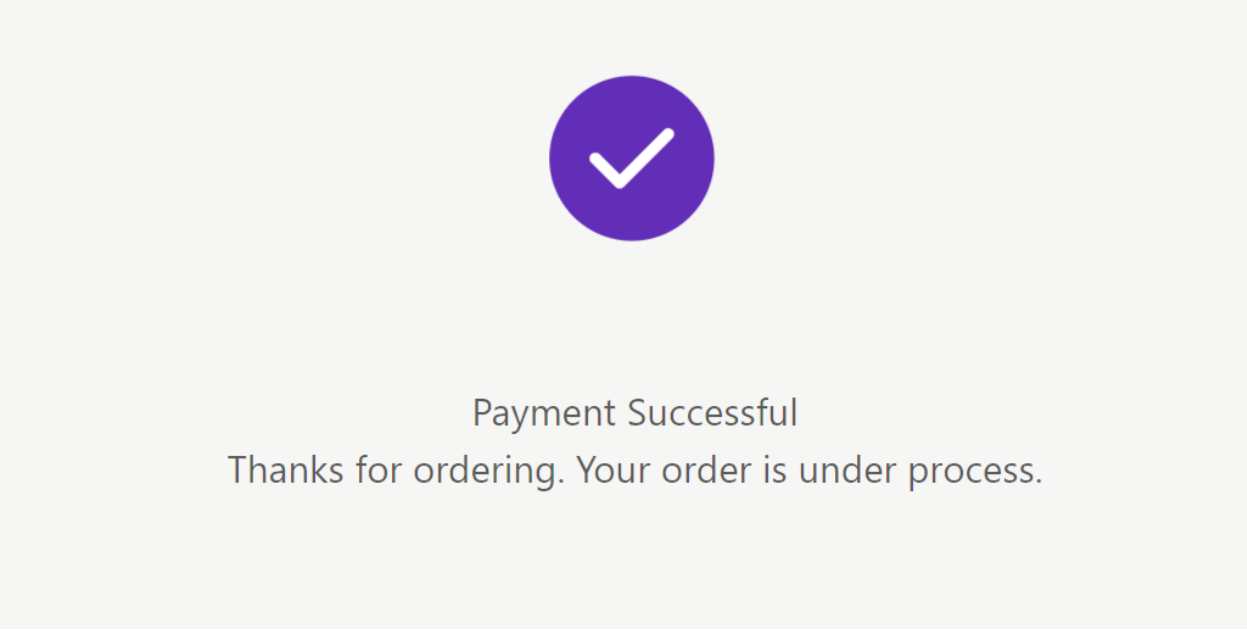
Checkout: This is the part where the customers can see and edit their information for the delivery.



Proceed to Payment: Here the customers have to add their payment method to pay for their desired products.

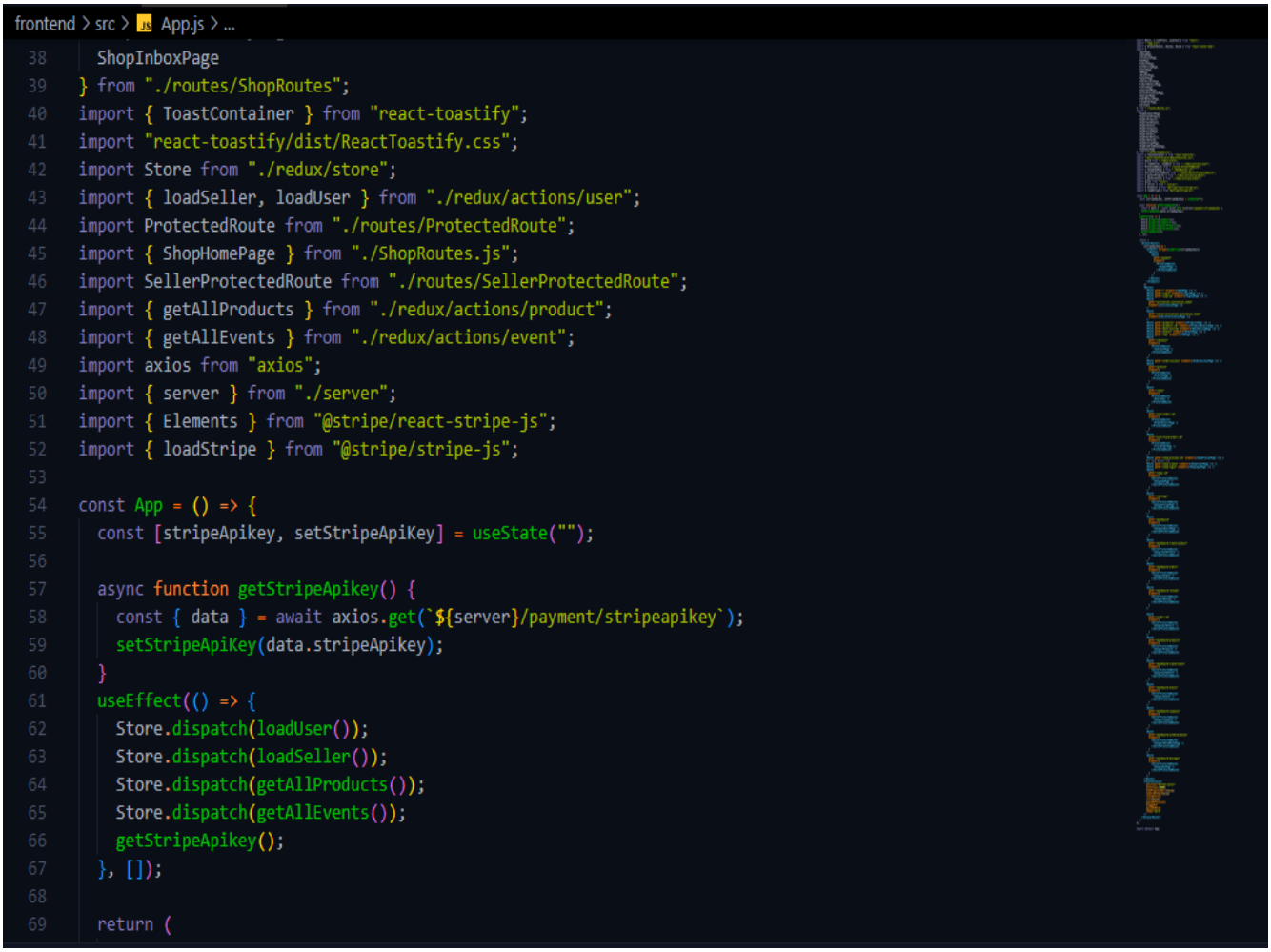


Confirm: They will be shown this page if their payment is successful and their order will be confirmed.



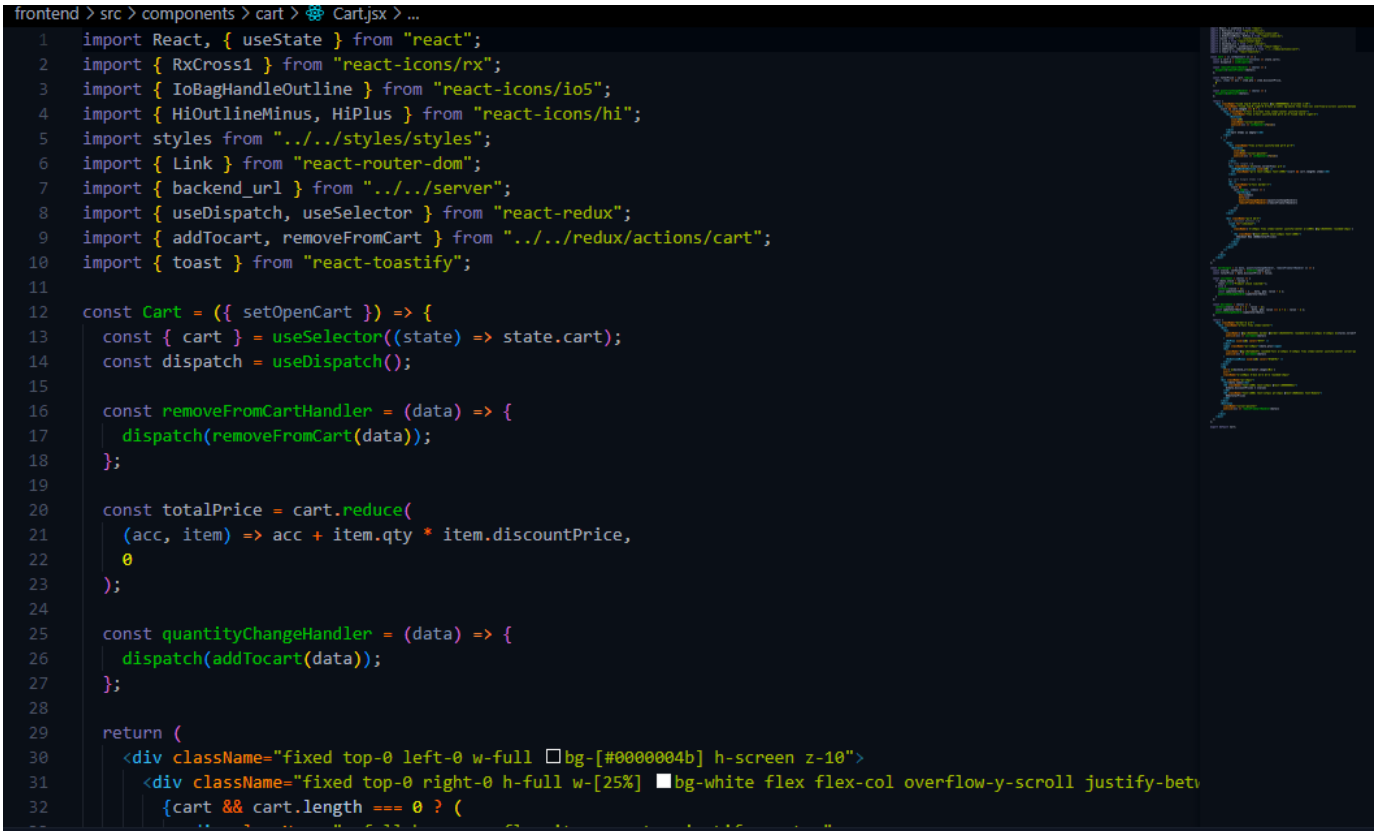
CODE SNIPPETS:

Front end App.js



```
frontend > src > App.js > ...
38      ShopInboxPage
39    } from "./routes/ShopRoutes";
40    import { ToastContainer } from "react-toastify";
41    import "react-toastify/dist/ReactToastify.css";
42    import Store from "./redux/store";
43    import { loadSeller, loadUser } from "./redux/actions/user";
44    import ProtectedRoute from "./routes/ProtectedRoute";
45    import { ShopHomePage } from "./ShopRoutes.js";
46    import SellerProtectedRoute from "./routes/SellerProtectedRoute";
47    import { getAllProducts } from "./redux/actions/product";
48    import { getAllEvents } from "./redux/actions/event";
49    import axios from "axios";
50    import { server } from "./server";
51    import { Elements } from "@stripe/react-stripe-js";
52    import { loadStripe } from "@stripe/stripe-js";
53
54    const App = () => {
55      const [stripeApikey, setStripeApiKey] = useState("");
56
57      async function getStripeApikey() {
58        const { data } = await axios.get(`${server}/payment/stripeapikey`);
59        setStripeApiKey(data.stripeApikey);
60      }
61      useEffect(() => {
62        Store.dispatch(loadUser());
63        Store.dispatch(loadSeller());
64        Store.dispatch(getAllProducts());
65        Store.dispatch(getAllEvents());
66        getStripeApikey();
67      }, []);
68
69      return (
```

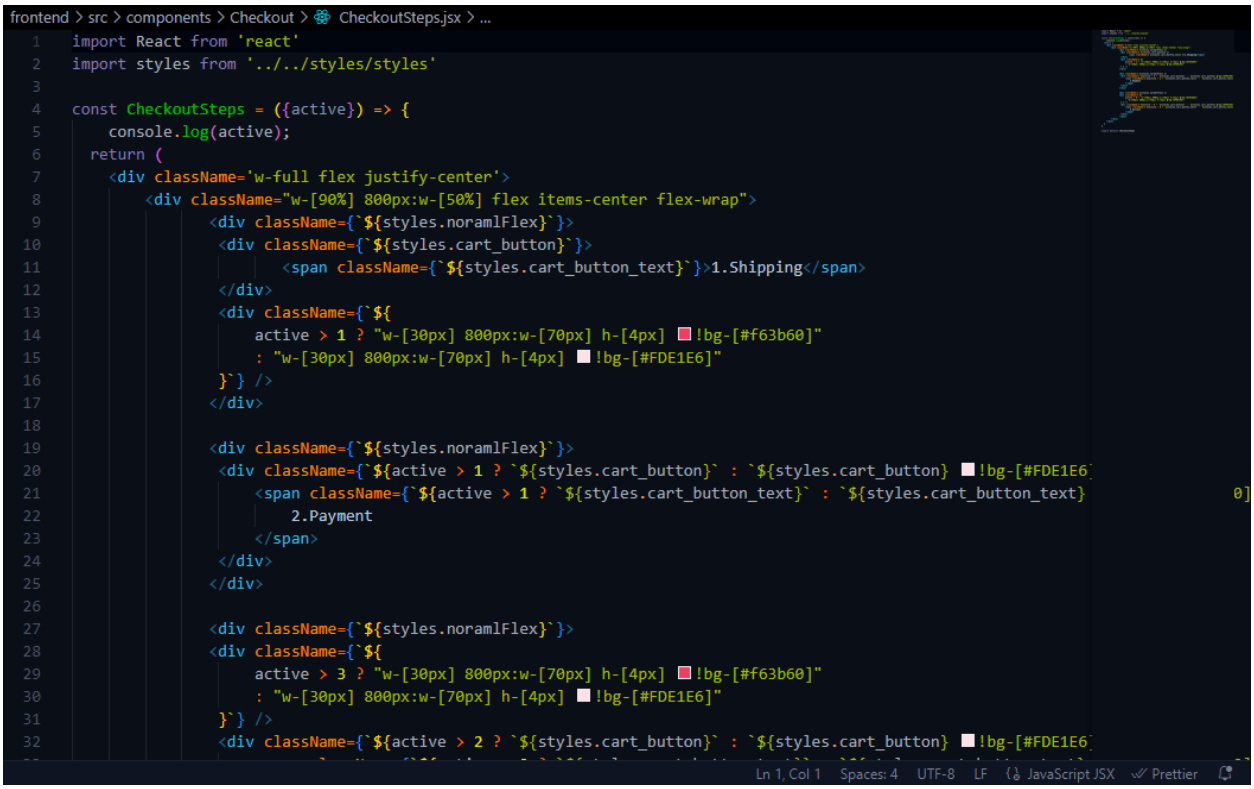
Frontend component codes



```
frontend > src > components > cart > Cart.jsx > ...
import React, { useState } from "react";
import { RxCross1 } from "react-icons/rx";
import { IoBagHandleOutline } from "react-icons/io5";
import { HiOutlineMinus, HiPlus } from "react-icons/hi";
import styles from "../../styles/styles";
import { Link } from "react-router-dom";
import { backend_url } from "../../server";
import { useDispatch, useSelector } from "react-redux";
import { addTocart, removeFromCart } from "../../redux/actions/cart";
import { toast } from "react-toastify";

const Cart = ({ setOpenCart }) => {
  const { cart } = useSelector((state) => state.cart);
  const dispatch = useDispatch();

  const removeFromCartHandler = (data) => {
    dispatch(removeFromCart(data));
  };

  const totalPrice = cart.reduce(
    (acc, item) => acc + item.qty * item.discountPrice,
    0
  );

  const quantityChangeHandler = (data) => {
    dispatch(addTocart(data));
  };

  return (
    <div className="fixed top-0 left-0 w-full bg-[#0000004b] h-screen z-10">
      <div className="fixed top-0 right-0 h-full w-[25%] bg-white flex flex-col overflow-y-scroll justify-betv
        {cart && cart.length === 0 ? (
```

checkout.jsx

checkoutSteps.jsx



countdown.jsx

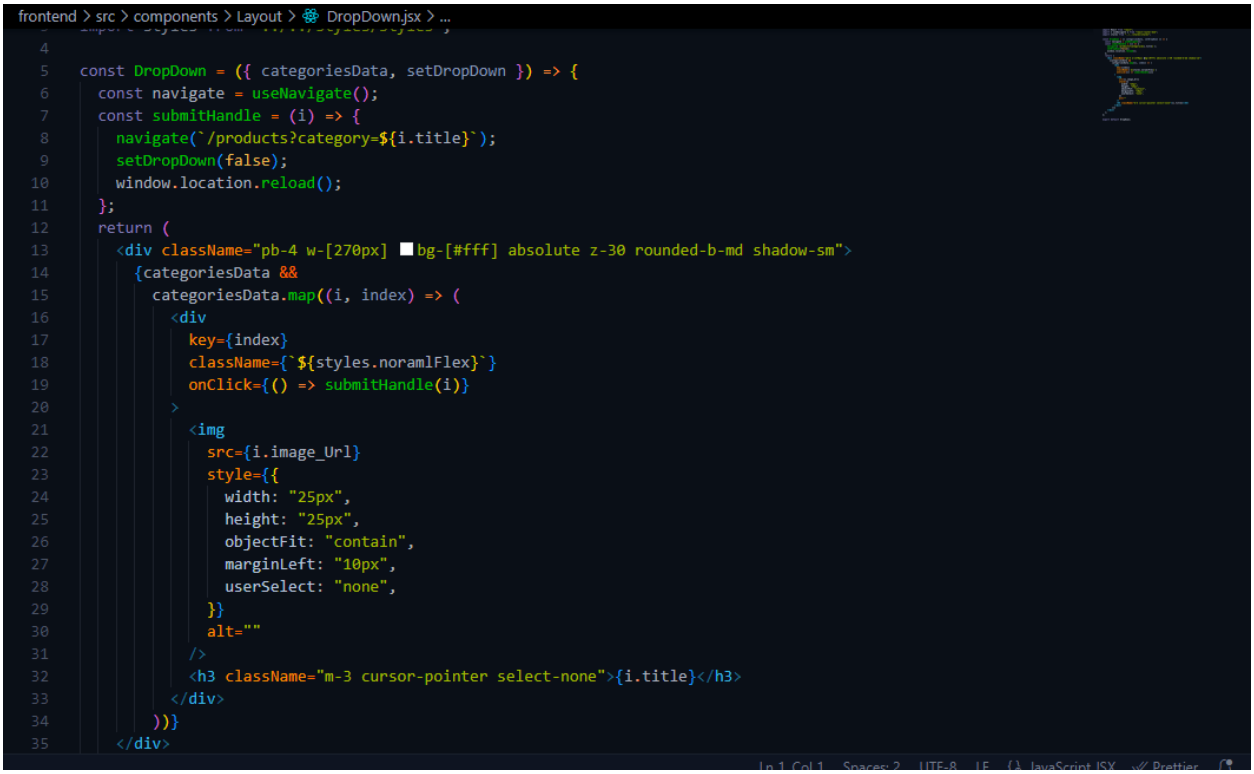
```
frontend > src > components > Events > CountDown.jsx > ...
import React, { useEffect, useState } from "react";

const CountDown = ({data}) => {
  const [timeLeft, setTimeLeft] = useState(calculateTimeLeft());

  useEffect(() => {
    const timer = setTimeout(() => {
      setTimeLeft(calculateTimeLeft());
    }, 1000);
    return () => clearTimeout(timer);
  });

  function calculateTimeLeft() {
    const difference = +new Date(data.Finish_Date) - +new Date();
    let timeLeft = {};

    if (difference > 0) {
      timeLeft = {
        days: Math.floor(difference / (1000 * 60 * 60 * 24)),
        hours: Math.floor((difference / (1000 * 60 * 60)) % 24),
        minutes: Math.floor((difference / 1000 / 60) % 60),
        seconds: Math.floor((difference / 1000) % 60),
      };
    }

    return timeLeft;
  }

  const timerComponents = Object.keys(timeLeft).map((interval) => {
    if (!timeLeft[interval]) {
      return null;
    }
```

eventcard.jsx

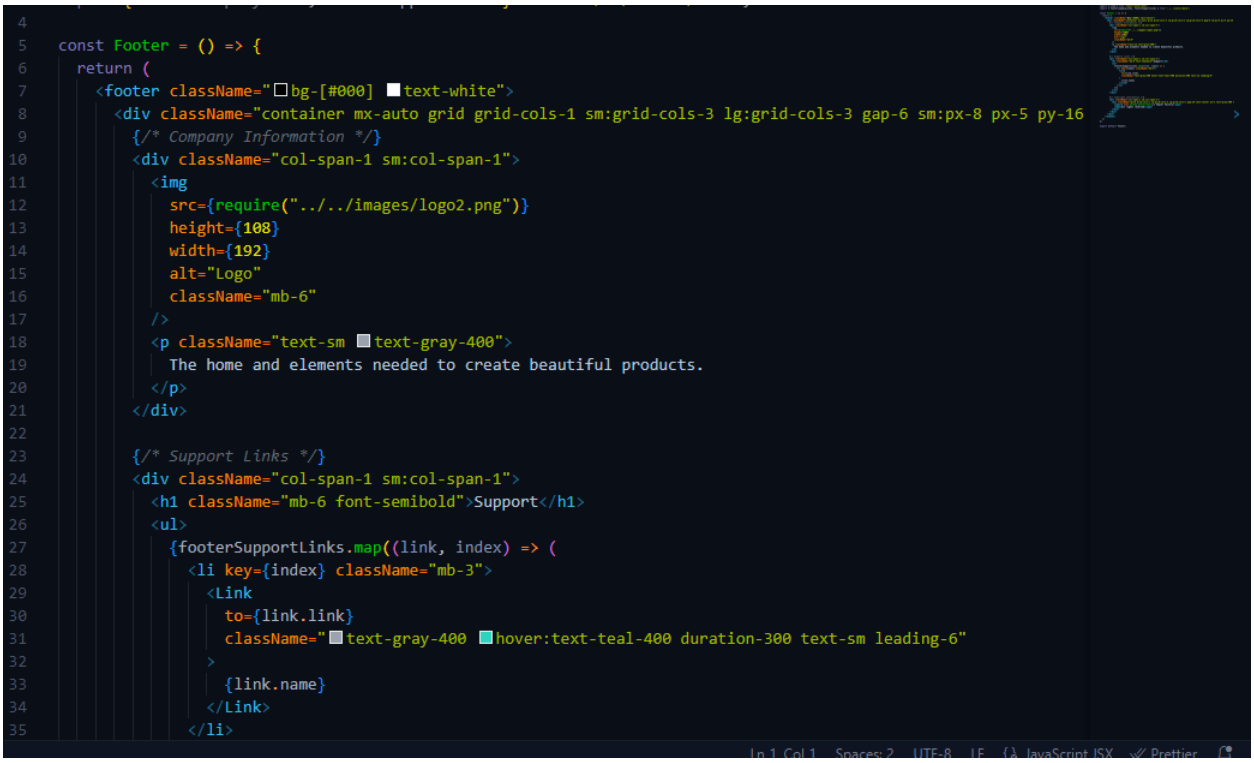
```
frontend > src > components > Events > EventCard.jsx > ...
import React from "react";
import { backend_url } from "../../server";
import styles from "../../styles/styles";
import CountDown from "./CountDown";
import { Link } from "react-router-dom";
import { useDispatch, useSelector } from "react-redux";
import { addTocart } from "../../redux/actions/cart";
import { toast } from "react-toastify";

const EventCard = ({ active, data }) => {
  const { cart } = useSelector((state) => state.cart);
  const dispatch = useDispatch();

  const addToCartHandler = (data) => {
    const isItemExists = cart && cart.find((i) => i._id === data._id);
    if (isItemExists) {
      toast.error("Item already in cart!");
    } else {
      if (data.stock < 1) {
        toast.error("Product stock limited!");
      } else {
        const cartData = { ...data, qty: 1 };
        dispatch(addTocart(cartData));
        toast.success("Item added to cart successfully!");
      }
    }
  }
  return (
    <div
      className={`w-full block bg-white rounded-lg ${
        active ? "unset" : "mb-12"
      } lg:flex p-2`}
```

event.jsx

dropdown.jsx



Footer.jsx



```jsx

const Footer = () => {
  return (
    <footer className="bg-[#000] text-white">
      <div className="container mx-auto grid grid-cols-1 sm:grid-cols-3 lg:grid-cols-3 gap-6 sm:px-8 px-5 py-16
        {/* Company Information */}
        <div className="col-span-1 sm:col-span-1">
          <img
            src={require("../../images/logo2.png")}
            height={108}
            width={192}
            alt="Logo"
            className="mb-6"
          />
          <p className="text-sm text-gray-400">
            The home and elements needed to create beautiful products.
          </p>
        </div>

        {/* Support Links */}
        <div className="col-span-1 sm:col-span-1">
          <h1 className="mb-6 font-semibold">Support</h1>
          <ul>
            {footerSupportLinks.map((link, index) => (
              <li key={index} className="mb-3">
                <Link
                  to={link.link}
                  className="text-gray-400 hover:text-teal-400 duration-300 text-sm leading-6"
                >
                  {link.name}
                </Link>
              </li>
```

header.jsx

```jsx
  const [open, setOpen] = useState(false);

  const handleSearchChange = (e) => {
    const term = e.target.value;
    setSearchTerm(term);

    const filteredProducts = allProducts && allProducts.filter((product) =>
      product.name.toLowerCase().includes(term.toLowerCase())
    );
    setSearchData(filteredProducts);
  };

  window.addEventListener("scroll", () => {
    if (window.scrollY > 70) {
      setActive(true);
    } else {
      setActive(false);
    }
  });

  return (
    <>
      <div className={`${styles.section}`}>
        <div className="hidden 800px:h-[50px] 800px:my-[20px] 800px:flex items-center justify-between">
          <div>
            <Link to="/">
              <img
                src={require("../../images/logo.png")}
                alt=""
                height={280}
                width={157.5}
              />
```

loader.jsx

frontend > src > components > Layout > Loader.jsx > ...

```jsx
import React from "react";
import Lottie from "react-lottie";

const Loader = () => {
  const defaultOptions = {
    loop: true,
    autoplay: true,
    rendererSettings: {
      preserveAspectRatio: "xMidYMid slice",
    },
  };
  return (
    <div className="w-full h-screen flex items-center justify-center">
      <Lottie options={defaultOptions} width={300} height={300} />
    </div>
  );
};

export default Loader;

```

navbar.jsx

frontend > src > components > Layout > Navbar.jsx > ...

```jsx
import React from 'react'
import { Link } from 'react-router-dom'
import { navItems } from '../../static/data'
import styles from '../../styles/styles'

const Navbar = ({active}) => {
  return (
    <div className={`block 800px:${styles.noramlFlex}`}>
          {
             navItems && navItems.map((i,index) => (
                 <div className="flex">
                     <Link to={i.url}
                     className={`${active === index + 1 ? "text-[#17dd1f]" : "text-black 800px:text-[#fff]
                     >
                     {i.title}
                     </Link>
                 </div>
             ))
          }
    </div>
  )
}

export default Navbar
```

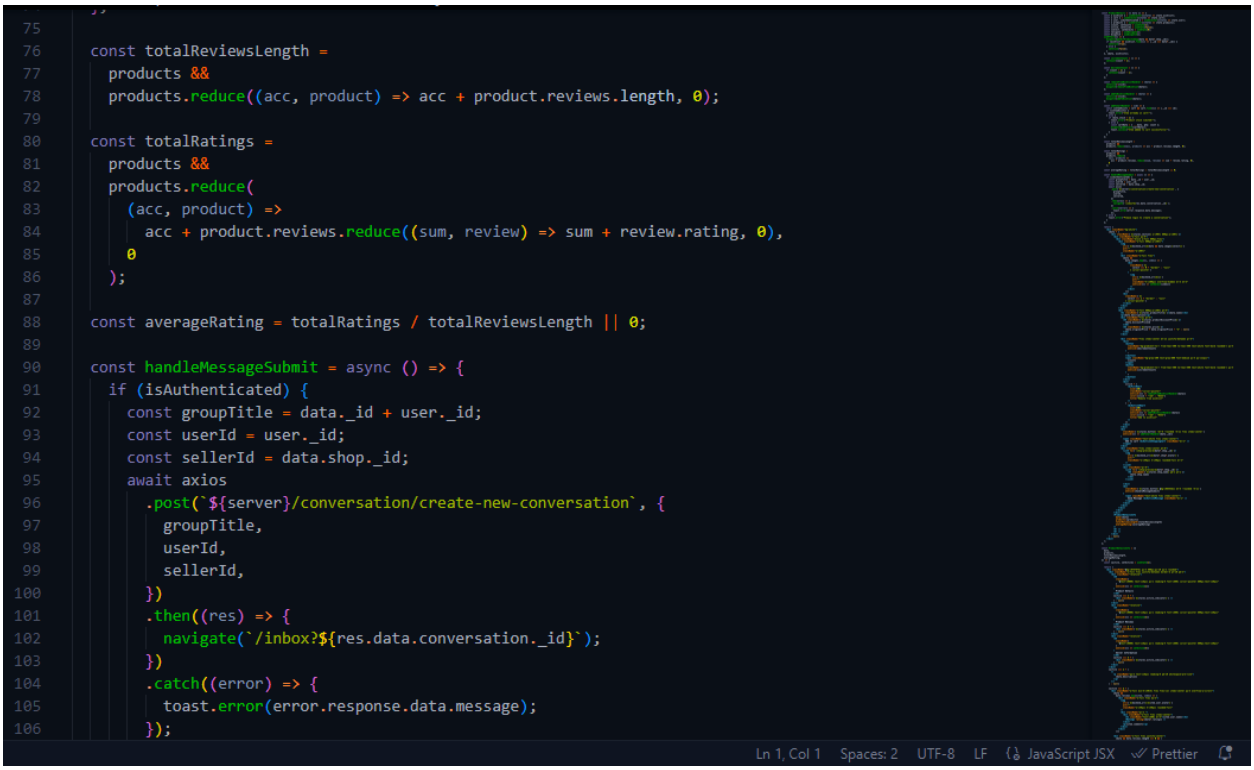
login.jsx

```jsx
const Login = () => {
  const navigate = useNavigate();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [visible, setVisible] = useState(false);

  const handleSubmit = async (e) => {
    e.preventDefault();

    await axios
      .post(
        `${server}/user/login-user`,
        {
          email,
          password,
        },
        { withCredentials: true }
      )
      .then((res) => {
        toast.success("Login Success!");
        navigate("/");
        window.location.reload(true);
      })
      .catch((err) => {
        toast.error(err.response.data.message);
      });
  };

  return (
    <div className="min-h-screen bg-gray-50 flex flex-col justify-center py-12 sm:px-6 lg:px-8">
      <div className="sm:mx-auto sm:w-full sm:max-w-md">
        <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">
```

payment.jsx

```jsx
      const client_secret = data.client_secret;

      if (!stripe || !elements) return;
      const result = await stripe.confirmCardPayment(client_secret, {
        payment_method: {
          card: elements.getElement(CardNumberElement),
        },
      });

      if (result.error) {
        toast.error(result.error.message);
      } else {
        if (result.paymentIntent.status === "succeeded") {
          order.paymnentInfo = {
            id: result.paymentIntent.id,
            status: result.paymentIntent.status,
            type: "Credit Card",
          };

          await axios
            .post(`${server}/order/create-order`, order, config)
            .then((res) => {
              setOpen(false);
              navigate("/order/success");
              toast.success("Order successful!");
              localStorage.setItem("cartItems", JSON.stringify([]));
              localStorage.setItem("latestOrder", JSON.stringify([]));
              window.location.reload();
            });
        }
      }
    } catch (error) {
```

productDetails.jsx



```
75
76  const totalReviewsLength =
77    products &&
78    products.reduce((acc, product) => acc + product.reviews.length, 0);
79
80  const totalRatings =
81    products &&
82    products.reduce(
83      (acc, product) =>
84        acc + product.reviews.reduce((sum, review) => sum + review.rating, 0),
85      0
86    );
87
88  const averageRating = totalRatings / totalReviewsLength || 0;
89
90  const handleMessageSubmit = async () => {
91    if (isAuthenticated) {
92      const groupTitle = data._id + user._id;
93      const userId = user._id;
94      const sellerId = data.shop._id;
95      await axios
96        .post(`${server}/conversation/create-new-conversation`, {
97          groupTitle,
98          userId,
99          sellerId,
100       })
101       .then((res) => {
102         navigate(`/inbox?${res.data.conversation._id}`);
103       })
104       .catch((error) => {
105         toast.error(error.response.data.message);
106       });
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript JSX Prettier

Ratings.jsx

```
6   const stars = [];
7
8   for (let i = 1; i <= 5; i++) {
9     if (i <= rating) {
10      stars.push(
11        <AiFillStar
12          key={i}
13          size={20}
14          color="#f6b100"
15          className="mr-2 cursor-pointer"
16        />
17      );
18    } else if (i === Math.ceil(rating) && !Number.isInteger(rating)) {
19      stars.push(
20        <BsStarHalf
21          key={i}
22          size={17}
23          color="#f6ba00"
24          className="mr-2 cursor-pointer"
25        />
26      );
27    } else {
28      stars.push(
29        <AiOutlineStar
30          key={i}
31          size={20}
32          color="#f6ba00"
33          className="mr-2 cursor-pointer"
34        />
35      );
36    }
37  }
```

SuggestedProduct.jsx

```jsx
const SuggestedProduct = ({ data }) => {
  const {allProducts} = useSelector((state) => state.products);
  const [productData,setProductData] = useState();

  useEffect(() => {
    const d =
    allProducts && allProducts.filter((i) => i.category === data.category);
    setProductData(d);
  }, []);

  return (
    <div>
      {data ? (
        <div className={`p-4 ${styles.section}`}>
          <h2
            className={`${styles.heading} text-[25px] font-[500] border-b mb-5`}
          >
            Related Product
          </h2>
          <div className="grid grid-cols-1 gap-[20px] md:grid-cols-2 md:gap-[25px] lg:grid-cols-4 lg:gap-[25px]
            {
              productData && productData.map((i,index) => (
                  <ProductCard data={i} key={index} />
              ))
            }
        </div>
        </div>
      ) : null}
    </div>
  );
};
```

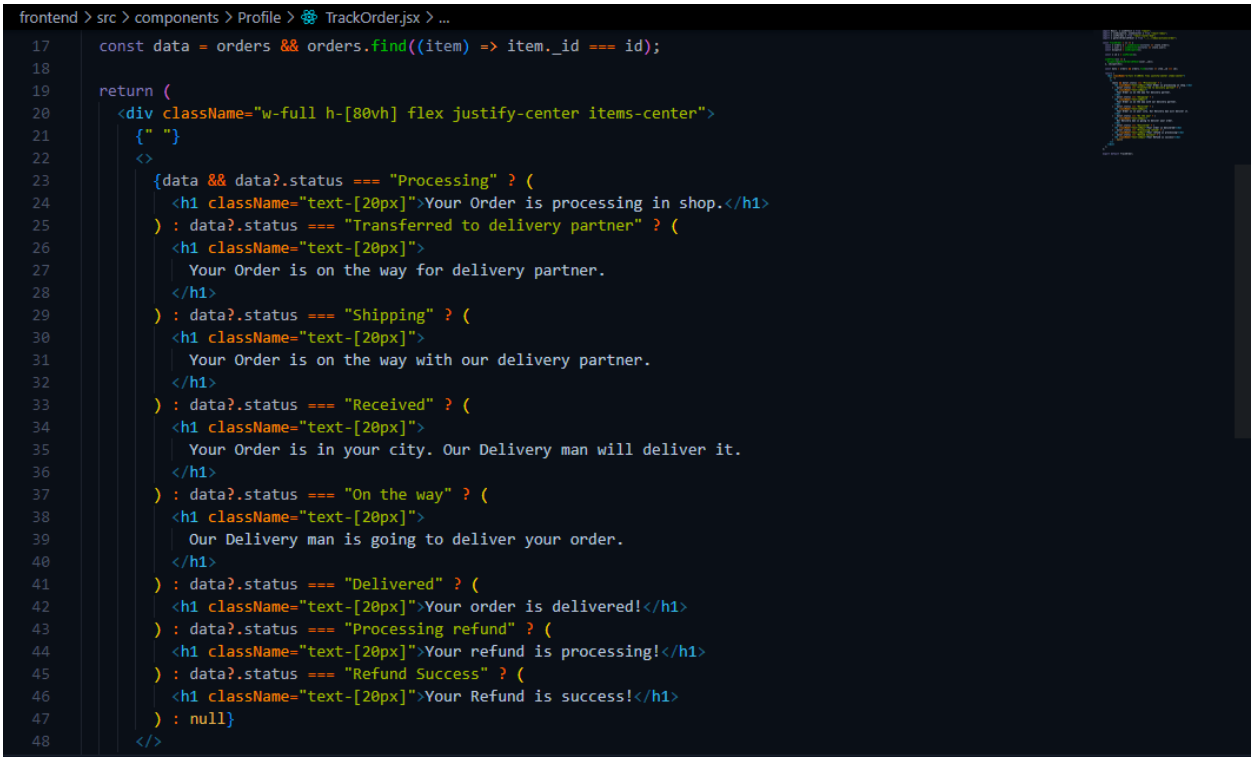
profileContent.jsc

```jsx
const ProfileContent = ({ active }) => {
  const { user, error, successMessage } = useSelector((state) => state.user);
  const [name, setName] = useState(user && user.name);
  const [email, setEmail] = useState(user && user.email);
  const [phoneNumber, setPhoneNumber] = useState(user && user.phoneNumber);
  const [password, setPassword] = useState("");
  const [avatar, setAvatar] = useState(null);
  const dispatch = useDispatch();

  useEffect(() => {
    if (error) {
      toast.error(error);
      dispatch({ type: "clearErrors" });
    }
    if (successMessage) {
      toast.success(successMessage);
      dispatch({ type: "clearMessages" });
    }
  }, [error, successMessage]);

  const handleSubmit = (e) => {
    e.preventDefault();
    dispatch(updateUserInformation(name, email, phoneNumber, password));
  };

  const handleImage = async (e) => {
    const file = e.target.files[0];
    setAvatar(file);

    const formData = new FormData();
```

profileSidebar.jsx

```jsx
const ProfileSidebar = ({ setActive, active }) => {
  const navigate = useNavigate();

  const logoutHandler = () => {
    axios
      .get(`${server}/user/logout`,{ withCredentials: true })
      .then((res) => {
        toast.success(res.data.message);
        window.location.reload(true);
        navigate("/login");
      })
      .catch((error) => {
        console.log(error.response.data.message);
      });
  };
  return (
    <div className="w-full bg-white shadow-sm rounded-[10px] p-4 pt-8">
      <div
        className="flex items-center cursor-pointer w-full mb-8"
        onClick={() => setActive(1)}
      >
        <RxPerson size={20} color={active === 1 ? "red" : ""} />
        <span className={`pl-3 ${active === 1 ? "text-[red]" : ""} 800px:block hidden`}>
          Profile
        </span>
      </div>
      <div
        className="flex items-center cursor-pointer w-full mb-8"
        onClick={() => setActive(2)}
      >
```

trackorder.jsx



```jsx
frontend > src > components > Profile > TrackOrder.jsx > ...
17    const data = orders && orders.find((item) => item._id === id);
18
19    return (
20      <div className="w-full h-[80vh] flex justify-center items-center">
21        {" "}
22        <>
23          {data && data?.status === "Processing" ? (
24            <h1 className="text-[20px]">Your Order is processing in shop.</h1>
25          ) : data?.status === "Transferred to delivery partner" ? (
26            <h1 className="text-[20px]">
27              Your Order is on the way for delivery partner.
28            </h1>
29          ) : data?.status === "Shipping" ? (
30            <h1 className="text-[20px]">
31              Your Order is on the way with our delivery partner.
32            </h1>
33          ) : data?.status === "Received" ? (
34            <h1 className="text-[20px]">
35              Your Order is in your city. Our Delivery man will deliver it.
36            </h1>
37          ) : data?.status === "On the way" ? (
38            <h1 className="text-[20px]">
39              Our Delivery man is going to deliver your order.
40            </h1>
41          ) : data?.status === "Delivered" ? (
42            <h1 className="text-[20px]">Your order is delivered!</h1>
43          ) : data?.status === "Processing refund" ? (
44            <h1 className="text-[20px]">Your refund is processing!</h1>
45          ) : data?.status === "Refund Success" ? (
46            <h1 className="text-[20px]">Your Refund is success!</h1>
47          ) : null}
48        </>
```

bestdeals.jsx

Route codes:

featuredProduct.jsx

frontend > src > components > Route > FeaturedProduct > FeaturedProduct.jsx > ...

```jsx
import React, { useEffect } from "react";
import { useSelector } from "react-redux";
import styles from "../../../styles/styles";
import ProductCard from "../ProductCard/ProductCard";

const FeaturedProduct = () => {
  const {allProducts} = useSelector((state) => state.products);

  return (
    <div>
      <div className={`${styles.section}`}>
        <div className={`${styles.heading}`}>
          <h1>Featured Products</h1>
        </div>
        <div className="grid grid-cols-1 gap-[20px] md:grid-cols-2 md:gap-[25px] lg:grid-cols-4 lg:gap-[25px] xl                :g
          {allProducts &&
            allProducts.map((i, index) => <ProductCard data={i} key={index} />)}
        </div>
      </div>
    </div>
  );
};

export default FeaturedProduct;

```

Productcard.jsx

```jsx
import Ratings from "../../Products/Ratings";

const ProductCard = ({ data,isEvent }) => {
  const { wishlist } = useSelector((state) => state.wishlist);
  const { cart } = useSelector((state) => state.cart);
  const [click, setClick] = useState(false);
  const [open, setOpen] = useState(false);
  const dispatch = useDispatch();

  useEffect(() => {
    if (wishlist && wishlist.find((i) => i._id === data._id)) {
      setClick(true);
    } else {
      setClick(false);
    }
  }, [wishlist]);

  const removeFromWishlistHandler = (data) => {
    setClick(!click);
    dispatch(removeFromWishlist(data));
  };

  const addToWishlistHandler = (data) => {
    setClick(!click);
    dispatch(addToWishlist(data));
  };

  const addToCartHandler = (id) => {
    const isItemExists = cart && cart.find((i) => i._id === id);
    if (isItemExists) {
```

Sponsoder.jsx

allcoupons.jsx

allevent.jsx

```jsx
const AllEvents = () => {
  const { events, isLoading } = useSelector((state) => state.events);
  const { seller } = useSelector((state) => state.seller);

  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllEventsShop(seller._id));
  }, [dispatch]);

  const handleDelete = (id) => {
    dispatch(deleteEvent(id));
    window.location.reload();
  }

  const columns = [
    { field: "id", headerName: "Product Id", minWidth: 150, flex: 0.7 },
    {
      field: "name",
      headerName: "Name",
      minWidth: 180,
      flex: 1.4,
    },
    {
      field: "price",
      headerName: "Price",
      minWidth: 100,
      flex: 0.6,
    },
```

allorders.jsx

```
frontend > src > components > Shop > AllOrders.jsx > ...
```

```jsx
    const { seller } = useSelector((state) => state.seller);

    const dispatch = useDispatch();

    useEffect(() => {
      dispatch(getAllOrdersOfShop(seller._id));
    }, [dispatch]);

    const columns = [
      { field: "id", headerName: "Order ID", minWidth: 150, flex: 0.7 },

      {
        field: "status",
        headerName: "Status",
        minWidth: 130,
        flex: 0.7,
        cellClassName: (params) => {
          return params.getValue(params.id, "status") === "Delivered"
            ? "greenColor"
            : "redColor";
        },
      },
      {
        field: "itemsQty",
        headerName: "Items Qty",
        type: "number",
        minWidth: 130,
        flex: 0.7,
      },

      {
        field: "total",
```

Allproducst.jsx

```jsx

const AllProducts = () => {
  const { products, isLoading } = useSelector((state) => state.products);
  const { seller } = useSelector((state) => state.seller);

  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllProductsShop(seller._id));
  }, [dispatch]);

  const handleDelete = (id) => {
    dispatch(deleteProduct(id));
    window.location.reload();
  };

  const columns = [
    { field: "id", headerName: "Product Id", minWidth: 150, flex: 0.7 },
    {
      field: "name",
      headerName: "Name",
      minWidth: 180,
      flex: 1.4,
    },
    {
      field: "price",
      headerName: "Price",
      minWidth: 100,
      flex: 0.6,
    },
    {
```


```jsx

const AllProducts = () => {
  const { products, isLoading } = useSelector((state) => state.products);
  const { seller } = useSelector((state) => state.seller);

  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllProductsShop(seller._id));
  }, [dispatch]);

  const handleDelete = (id) => {
    dispatch(deleteProduct(id));
    window.location.reload();
  };

  const columns = [
    { field: "id", headerName: "Product Id", minWidth: 150, flex: 0.7 },
    {
      field: "name",
      headerName: "Name",
      minWidth: 180,
      flex: 1.4,
    },
    {
      field: "price",
      headerName: "Price",
      minWidth: 100,
      flex: 0.6,
    },
    {
```

```jsx
const AllRefundOrders = () => {
  const { orders, isLoading } = useSelector((state) => state.order);
  const { seller } = useSelector((state) => state.seller);

  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllOrdersOfShop(seller._id));
  }, [dispatch]);

  const refundOrders = orders && orders.filter((item) => item.status === "Processing refund"  || item.status ===

  const columns = [
    { field: "id", headerName: "Order ID", minWidth: 150, flex: 0.7 },

    {
      field: "status",
      headerName: "Status",
      minWidth: 130,
      flex: 0.7,
      cellClassName: (params) => {
        return params.getValue(params.id, "status") === "Delivered"
          ? "greenColor"
          : "redColor";
      },
    },
    {
      field: "itemsQty",
      headerName: "Items Qty",
      type: "number",
      minWidth: 130,
```

```jsx
const CreateEvent = () => {
  const { seller } = useSelector((state) => state.seller);
  const { success, error } = useSelector((state) => state.events);
  const navigate = useNavigate();
  const dispatch = useDispatch();

  const [images, setImages] = useState([]);
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");
  const [category, setCategory] = useState("");
  const [tags, setTags] = useState("");
  const [originalPrice, setOriginalPrice] = useState();
  const [discountPrice, setDiscountPrice] = useState();
  const [stock, setStock] = useState();
const [startDate,setStartDate] = useState(null);
const [endDate,setEndDate] = useState(null);

const handleStartDateChange = (e) => {
    const startDate = new Date(e.target.value);
    const minEndDate = new Date(startDate.getTime() + 3 * 24 * 60 * 60 * 1000);
    setStartDate(startDate);
    setEndDate(null);
    document.getElementById("end-date").min = minEndDate.toISOString.slice(0,10);
}

const handleEndDateChange = (e) => {
    const endDate = new Date(e.target.value);
     setEndDate(endDate);
};

const today = new Date().toISOString().slice(0,10);
```

```
const CreateProduct = () => {
  const { seller } = useSelector((state) => state.seller);
  const { success, error } = useSelector((state) => state.products);
  const navigate = useNavigate();
  const dispatch = useDispatch();

  const [images, setImages] = useState([]);
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");
  const [category, setCategory] = useState("");
  const [tags, setTags] = useState("");
  const [originalPrice, setOriginalPrice] = useState();
  const [discountPrice, setDiscountPrice] = useState();
  const [stock, setStock] = useState();

  useEffect(() => {
    if (error) {
      toast.error(error);
    }
    if (success) {
      toast.success("Product created successfully!");
      navigate("/dashboard");
      window.location.reload();
    }
  }, [dispatch, error, success]);

  const handleImageChange = (e) => {
    e.preventDefault();

    let files = Array.from(e.target.files);
    setImages((prevImages) => [...prevImages, ...files]);
```

```
const DashboardMessages = () => {
  const { seller } = useSelector((state) => state.seller);
  const [conversations, setConversations] = useState([]);
  const [arrivalMessage, setArrivalMessage] = useState(null);
  const [currentChat, setCurrentChat] = useState();
  const [messages, setMessages] = useState([]);
  const [userData, setUserData] = useState(null);
  const [newMessage, setNewMessage] = useState("");
  const [onlineUsers, setOnlineUsers] = useState([]);
  const [activeStatus, setActiveStatus] = useState(false);
  const [open, setOpen] = useState(false);
  const scrollRef = useRef(null);

  useEffect(() => {
    socketId.on("getMessage", (data) => {
      setArrivalMessage({
        sender: data.senderId,
        text: data.text,
        createdAt: Date.now(),
      });
    });
  }, []);

  useEffect(() => {
    arrivalMessage &&
      currentChat?.members.includes(arrivalMessage.sender) &&
      setMessages((prev) => [...prev, arrivalMessage]);
  }, [arrivalMessage, currentChat]);

  useEffect(() => {
    const getConversation = async () => {
```

```jsx
const OrderDetails = () => {
  const { orders, isLoading } = useSelector((state) => state.order);
  const { seller } = useSelector((state) => state.seller);
  const dispatch = useDispatch();
  const [status, setStatus] = useState("");
  const navigate = useNavigate();

  const { id } = useParams();

  useEffect(() => {
    dispatch(getAllOrdersOfShop(seller._id));
  }, [dispatch]);

  const data = orders && orders.find((item) => item._id === id);

  const orderUpdateHandler = async (e) => {
    await axios
      .put(
        `${server}/order/update-order-status/${id}`,
        {
          status,
        },
        { withCredentials: true }
      )
      .then((res) => {
        toast.success("Order updated!");
        navigate("/dashboard-orders");
      })
      .catch((error) => {
        toast.error(error.response.data.message);
```

shopcreate.jsx

```jsx
const ShopCreate = () => {
  const navigate = useNavigate();
  const [email, setEmail] = useState("");
  const [name,setName] = useState("");
  const [phoneNumber,setPhoneNumber] = useState();
  const [address,setAddress] = useState("");
  const [zipCode,setZipCode] = useState();
  const [avatar,setAvatar] = useState();
  const [password, setPassword] = useState("");
  const [visible, setVisible] = useState(false);

  const handleSubmit = async (e) => {
    e.preventDefault();
    const config = { headers: { "Content-Type": "multipart/form-data" } };

    const newForm = new FormData();

    newForm.append("file", avatar);
    newForm.append("name", name);
    newForm.append("email", email);
    newForm.append("password", password);
    newForm.append("zipCode", zipCode);
    newForm.append("address", address);
    newForm.append("phoneNumber", phoneNumber);
    axios
      .post(`${server}/shop/create-shop`, newForm, config)
      .then((res) => {
        toast.success(res.data.message);
        setName("");
```

ShopInfo.jsx

```jsx
const ShopInfo = ({ isOwner }) => {
  const [data,setData] = useState({});
  const {products} = useSelector((state) => state.products);
  const [isLoading,setIsLoading] = useState(false);
  const {id} = useParams();
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllProductsShop(id));
    setIsLoading(true);
    axios.get(`${server}/shop/get-shop-info/${id}`).then((res) => {
     setData(res.data.shop);
     setIsLoading(false);
    }).catch((error) => {
      console.log(error);
      setIsLoading(false);
    })
  }, [])


  const logoutHandler = async () => {
    axios.get(`${server}/shop/logout`,{
      withCredentials: true,
    });
    window.location.reload();
  };

  const totalReviewsLength =
    products &&
    products.reduce((acc, product) => acc + product.reviews.length, 0);
```

ShopLogin.jsx

```jsx
const ShopLogin = () => {
  const navigate = useNavigate();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [visible, setVisible] = useState(false);

  const handleSubmit = async (e) => {
    e.preventDefault();

    await axios
      .post(
        `${server}/shop/login-shop`,
        {
          email,
          password,
        },
        { withCredentials: true }
      )
      .then((res) => {
        toast.success("Login Success!");
        navigate("/dashboard");
        window.location.reload(true);
      })
      .catch((err) => {
        toast.error(err.response.data.message);
      });
  };

  return (
    <div className="min-h-screen bg-gray-50 flex flex-col justify-center py-12 sm:px-6 lg:px-8">
      <div className="sm:mx-auto sm:w-full sm:max-w-md">
        <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">
```

ShopProfileData.jsx

```jsx
const ShopProfileData = ({ isOwner }) => {
  const { products } = useSelector((state) => state.products);
  const { events } = useSelector((state) => state.events);
  const { seller } = useSelector((state) => state.seller);
  const { id } = useParams();
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllProductsShop(id));
    dispatch(getAllEventsShop(seller._id));
  }, [dispatch]);

  const [active, setActive] = useState(1);

  const allReviews =
    products && products.map((product) => product.reviews).flat();

  return (
    <div className="w-full">
      <div className="flex w-full items-center justify-between">
        <div className="w-full flex">
          <div className="flex items-center" onClick={() => setActive(1)}>
            <h5
              className={`font-[600] text-[20px] ${
                active === 1 ? "text-red-500" : "text-[#333]"
              } cursor-pointer pr-[20px]`}
            >
              Shop Products
            </h5>
          </div>
```

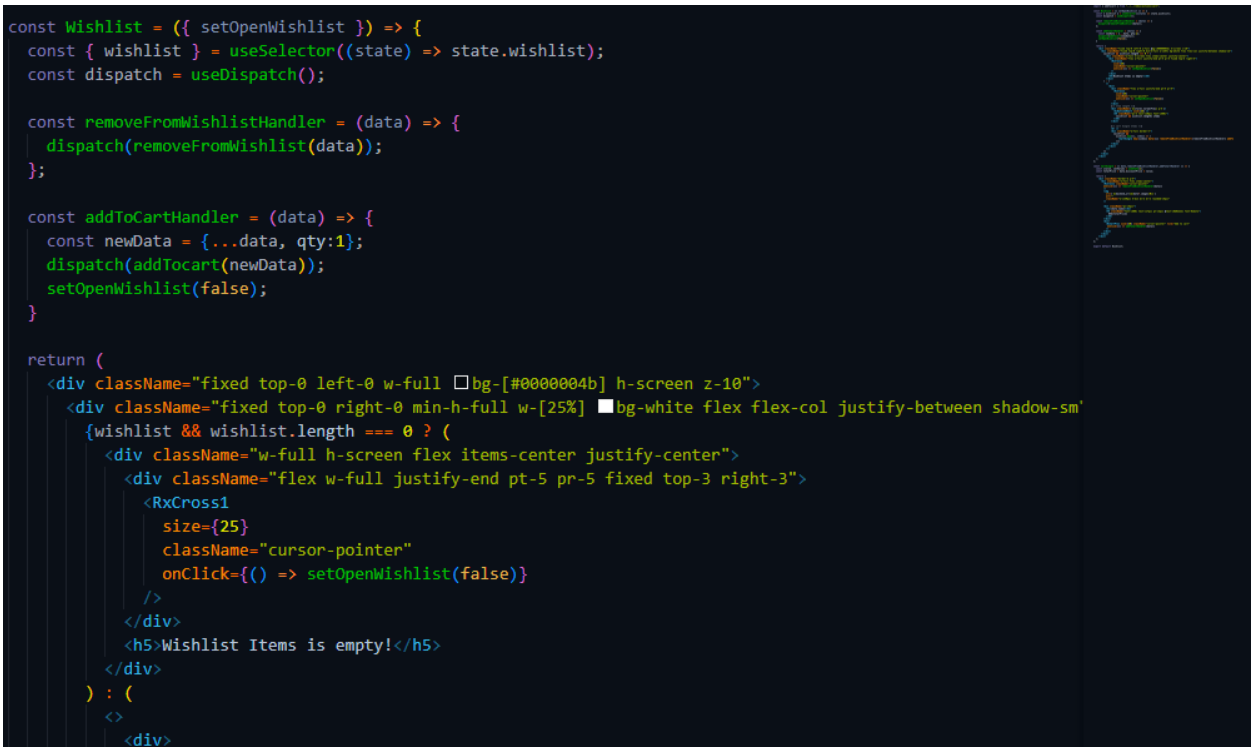
Shopsettings.jsx

```jsx
const ShopSettings = () => {
  const { seller } = useSelector((state) => state.seller);
  const [avatar,setAvatar] = useState();
  const [name,setName] = useState(seller && seller.name);
  const [description,setDescription] = useState(seller && seller.description ? seller.description : "");
  const [address,setAddress] = useState(seller && seller.address);
  const [phoneNumber,setPhoneNumber] = useState(seller && seller.phoneNumber);
  const [zipCode,setZipcode] = useState(seller && seller.zipCode);


  const dispatch = useDispatch();

  const handleImage = async (e) => {
    e.preventDefault();
    const file = e.target.files[0];
    setAvatar(file);

    const formData = new FormData();

    formData.append("image", e.target.files[0]);

    await axios.put(`${server}/shop/update-shop-avatar`, formData,{
        headers: {
            "Content-Type": "multipart/form-data",
        },
        withCredentials: true,
    }).then((res) => {
        dispatch(loadSeller());
        toast.success("Avatar updated successfully!")
    }).catch((error) => {
        toast.error(error.response.data.message);
```

Signup.jsx

```jsx
const Singup = () => {
  const [email, setEmail] = useState("");
  const [name, setName] = useState("");
  const [password, setPassword] = useState("");
  const [visible, setVisible] = useState(false);
  const [avatar, setAvatar] = useState(null);

  const handleFileInputChange = (e) => {
    const file = e.target.files[0];
    setAvatar(file);
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    const config = { headers: { "Content-Type": "multipart/form-data" } };

    const newForm = new FormData();

    newForm.append("file", avatar);
    newForm.append("name", name);
    newForm.append("email", email);
    newForm.append("password", password);

    axios
      .post(`${server}/user/create-user`, newForm, config)
      .then((res) => {
        toast.success(res.data.message);
        setName("");
        setEmail("");
        setPassword("");
        setAvatar();
```

Wishlist.jsx



```jsx
const Wishlist = ({ setOpenWishlist }) => {
  const { wishlist } = useSelector((state) => state.wishlist);
  const dispatch = useDispatch();

  const removeFromWishlistHandler = (data) => {
    dispatch(removeFromWishlist(data));
  };

  const addToCartHandler = (data) => {
    const newData = {...data, qty:1};
    dispatch(addTocart(newData));
    setOpenWishlist(false);
  }

  return (
    <div className="fixed top-0 left-0 w-full bg-[#0000004b] h-screen z-10">
      <div className="fixed top-0 right-0 min-h-full w-[25%] bg-white flex flex-col justify-between shadow-sm
        {wishlist && wishlist.length === 0 ? (
          <div className="w-full h-screen flex items-center justify-center">
            <div className="flex w-full justify-end pt-5 pr-5 fixed top-3 right-3">
              <RxCross1
                size={25}
                className="cursor-pointer"
                onClick={() => setOpenWishlist(false)}
              />
            </div>
            <h5>Wishlist Items is empty!</h5>
          </div>
        ) : (
          <>
            <div>
```

UserOrderDetails.jsx

```jsx
const UserOrderDetails = () => {
  const { orders } = useSelector((state) => state.order);
  const { user } = useSelector((state) => state.user);
  const dispatch = useDispatch();
  const [open, setOpen] = useState(false);
  const [comment, setComment] = useState("");
  const [selectedItem, setSelectedItem] = useState(null);
  const [rating, setRating] = useState(1);

  const { id } = useParams();

  useEffect(() => {
    dispatch(getAllOrdersOfUser(user._id));
  }, [dispatch]);

  const data = orders && orders.find((item) => item._id === id);

  const reviewHandler = async (e) => {
    await axios
      .put(
        `${server}/product/create-new-review`,
        {
          user,
          rating,
          comment,
          productId: selectedItem?._id,
          orderId: id,
        },
        { withCredentials: true }
      )
      .then((res) => {
```

Routes.js

frontend > src > routes > Routes.js

```jsx
import LoginPage from "../pages/Login";
import SignupPage from "../pages/SignupPage";
import ActivationPage from "../pages/ActivationPage";
import HomePage from "../pages/HomePage";
import ProductsPage from "../pages/ProductsPage";
import BestSellingPage from "../pages/BestSellingPage";
import EventsPage from "../pages/EventsPage";
import FAQPage from "../pages/FAQPage";
import CheckoutPage from "../pages/CheckoutPage";
import PaymentPage from "../pages/PaymentPage";
import OrderSuccessPage from "../pages/OrderSuccessPage";
import ProductDetailsPage from "../pages/ProductDetailsPage";
import ProfilePage from "../pages/ProfilePage";
import ShopCreatePage from "../pages/ShopCreate";
import SellerActivationPage from "../pages/SellerActivationPage";
import ShopLoginPage from "../pages/ShopLoginPage";
import OrderDetailsPage from "../pages/OrderDetailsPage";
import TrackOrderPage from "../pages/TrackOrderPage";
import UserInbox from "../pages/UserInbox.jsx";

export {
    LoginPage,
    SignupPage,
    ActivationPage,
    HomePage,
    ProductsPage,
    BestSellingPage,
    EventsPage,
    FAQPage,
    CheckoutPage,
    PaymentPage,
    OrderSuccessPage,
```

Conversation.js

backend > model > conversation.js > ...

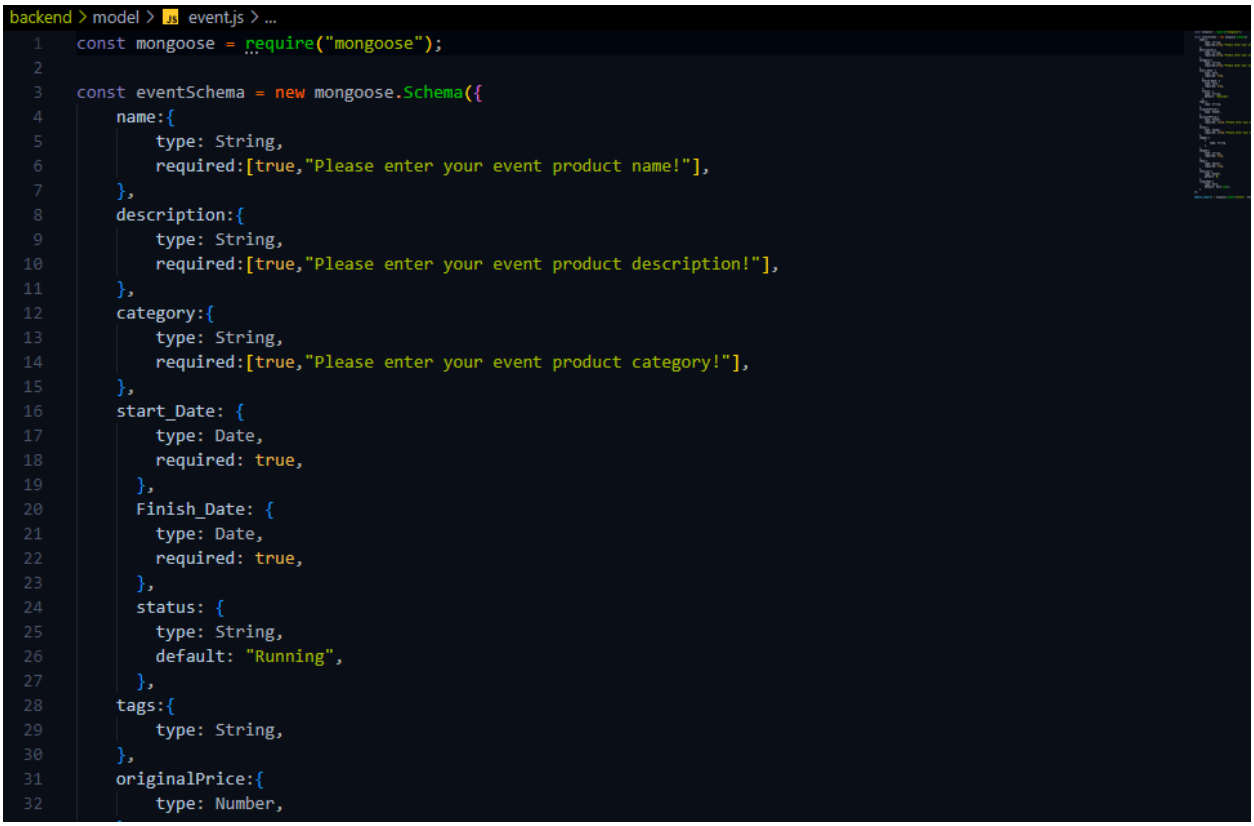
```
const mongoose = require("mongoose");

const conversationSchema = new mongoose.Schema(
  {
    groupTitle:{
        type: String,
    },
    members: {
      type: Array,
    },
    lastMessage: {
      type: String,
    },
    lastMessageId: {
      type: String,
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Conversation", conversationSchema);

```

CouponCode.js

backend > model > JS coupounCode.js > ...

```
const mongoose = require("mongoose");

const coupounCodeSchema = new mongoose.Schema({
    name:{
        type: String,
        required:[true,"Enter Coupoun Code Here"],
        unique: true,
    },
    value:{
        type: Number,
        required: true,
    },
    minAmount:{
        type: Number,
    },
    maxAmount:{
        type: Number,
    },
    shopId:{
     type: String,
     required: true,
    },
    selectedProduct:{
     type: String,
    },
    createdAt:{
        type: Date,
        default: Date.now(),
    }
});

module.exports = mongoose.model("CoupounCode", coupounCodeSchema);
```

Event.js


backend > model > event.js > ...
const mongoose = require("mongoose");
const eventSchema = new mongoose.Schema({
name:{
type: String,
required:[true,"Please enter your event product name!"],
},
description:{
type: String,
required:[true,"Please enter your event product description!"],
},
category:{
type: String,
required:[true,"Please enter your event product category!"],
},
start_Date: {
type: Date,
required: true,
},
Finish_Date: {
type: Date,
required: true,
},
status: {
type: String,
default: "Running",
},
tags:{
type: String,
},
originalPrice:{
type: Number,

Messages.js

backend > model > messages.js > ...

```
const mongoose = require("mongoose");

const messagesSchema = new mongoose.Schema(
  {
    conversationId: {
      type: String,
    },
    text:{
      type: String,
    },
    sender: {
      type: String,
    },
    images: [
     {
        type: String,
     }
    ],
  },
  { timestamps: true }
);

module.exports = mongoose.model("Messages", messagesSchema);

```

Order.js

backend > model > order.js > ...

```
const mongoose = require("mongoose");

const orderSchema = new mongoose.Schema({
    cart:{
        type: Array,
        required: true,
    },
    shippingAddress:{
        type: Object,
        required: true,
    },
    user:{
        type: Object,
        required: true,
    },
    totalPrice:{
        type: Number,
        required: true,
    },
    status:{
        type: String,
        default: "Processing",
    },
    paymentInfo:{
        id:{
            type: String,
        },
        status: {
            type: String,
        },
        type:{
            type: String,
```

Product.js

backend > model > product.js > ...

```
const mongoose = require("mongoose");

const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your product name!"],
  },
  description: {
    type: String,
    required: [true, "Please enter your product description!"],
  },
  category: {
    type: String,
    required: [true, "Please enter your product category!"],
  },
  tags: {
    type: String,
  },
  originalPrice: {
    type: Number,
  },
  discountPrice: {
    type: Number,
    required: [true, "Please enter your product price!"],
  },
  stock: {
    type: Number,
    required: [true, "Please enter your product stock!"],
  },
  images: [
    {
      type: String,
```

Shop.js

backend > model > shop.js > ...

```
const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

const shopSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your shop name!"],
  },
  email: {
    type: String,
    required: [true, "Please enter your shop email address"],
  },
  password: {
    type: String,
    required: [true, "Please enter your password"],
    minLength: [6, "Password should be greater than 6 characters"],
    select: false,
  },
  description: {
    type: String,
  },
  address: {
    type: String,
    required: true,
  },
  phoneNumber: {
    type: Number,
    required: true,
  },
  role: {
    type: String,
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF

User.js

backend > model > user.js > ...

```
const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your name!"],
  },
  email: {
    type: String,
    required: [true, "Please enter your email address"],
  },
  password: {
    type: String,
    required: [true, "Please enter your password"],
    minLength: [6, "Password should be greater than 6 characters"],
    select: false,
  },
  phoneNumber: {
    type: Number,
  },
  addresses: [
    {
      country: {
        type: String,
      },
      city: {
        type: String,
      },
      address1: {
        type: String,
```

event.js

```
const AllEvents = () => {
  const { events, isLoading } = useSelector((state) => state.events);
  const { seller } = useSelector((state) => state.seller);

  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getAllEventsShop(seller._id));
  }, [dispatch]);

  const handleDelete = (id) => {
    dispatch(deleteEvent(id));
    window.location.reload();
  }

  const columns = [
    { field: "id", headerName: "Product Id", minWidth: 150, flex: 0.7 },
    {
      field: "name",
      headerName: "Name",
      minWidth: 180,
      flex: 1.4,
    },
    {
      field: "price",
      headerName: "Price",
      minWidth: 100,
      flex: 0.6,
    },
```

server.jseer

backend > server.js > ...

```
const app = require("./app");
const connectDatabase = require("./db/Database");

// Handling uncaught Exception
process.on("uncaughtException", (err) => {
  console.log(`Error: ${err.message}`);
  console.log(`shutting down the server for handling uncaught exception`);
});

// config
if (process.env.NODE_ENV !== "PRODUCTION") {
  require("dotenv").config({
    path: "backend/config/.env",
  });
}

// connect db
connectDatabase();

// create server
const server = app.listen(process.env.PORT, () => {
  console.log(
    `Server is running on http://localhost:${process.env.PORT}`
  );
});

// unhandled promise rejection
process.on("unhandledRejection", (err) => {
  console.log(`Shutting down the server for ${err.message}`);
  console.log(`shutting down the server for unhandle promise rejection`);

  server.close(() => {
```

ErrorHandler.js

backend > utils > ErrorHandler.js > ErrorHandler

```
class ErrorHandler extends Error{
    constructor(message,statusCode){
        super(message);
        this.statusCode = statusCode

        Error.captureStackTrace(this,this.constructor);

    }

}
module.exports = ErrorHandler
```

Jwttoken.js

backend > utils > jwtToken.js > ...

```
// create token and saving that in cookies
const sendToken = (user, statusCode, res) => {
  const token = user.getJwtToken();

  // Options for cookies
  const options = {
    expires: new Date(Date.now() + 90 * 24 * 60 * 60 * 1000),
    httpOnly: true,
    Secure: true,
  };

  res.status(statusCode).cookie("token", token, options).json({
    success: true,
    user,
    token,
  });
};

module.exports = sendToken;
```

SendMail.js

backend > utils > JS sendMail.js > ...

```
const nodemailer = require("nodemailer");

const sendMail = async (options) => {
    const transporter = nodemailer.createTransport({
        host: process.env.SMPT_HOST,
        port: process.env.SMPT_PORT,
        service: process.env.SMPT_SERVICE,
        auth:{
            user: process.env.SMPT_MAIL,
            pass: process.env.SMPT_PASSWORD,
        },
    });

    const mailOptions = {
        from: process.env.SMPT_MAIL,
        to: options.email,
        subject: options.subject,
        text: options.message,
    };

    await transporter.sendMail(mailOptions);
};

module.exports = sendMail;
```

ShopToken.js

backend > utils > shopToken.js > ...

```
// create token and saving that in cookies
const sendShopToken = (user, statusCode, res) => {
    const token = user.getJwtToken();

    // Options for cookies
    const options = {
      expires: new Date(Date.now() + 90 * 24 * 60 * 60 * 1000),
      httpOnly: true,
      Secure: true,
    };

    res.status(statusCode).cookie("seller_token", token, options).json({
      success: true,
      user,
      token,
    });
  };

  module.exports = sendShopToken;

```

Database.js

backend > db > Database.js > ...

```
const mongoose = require("mongoose");

const connectDatabase = () => {
  mongoose
    .connect(process.env.DB_URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    })
    .then((data) => {
      console.log(`mongod connected with server: ${data.connection.host}`);
    });
};

module.exports = connectDatabase;

```

conversation.js

```
const router = express.Router();

// create a new conversation
router.post(
  "/create-new-conversation",
  catchAsyncErrors(async (req, res, next) => {
    try {
      const { groupTitle, userId, sellerId } = req.body;

      const isConversationExist = await Conversation.findOne({ groupTitle });

      if (isConversationExist) {
        const conversation = isConversationExist;
        res.status(201).json({
          success: true,
          conversation,
        });
      } else {
        const conversation = await Conversation.create({
          members: [userId, sellerId],
          groupTitle: groupTitle,
        });

        res.status(201).json({
          success: true,
          conversation,
        });
      }
    } catch (error) {
      return next(new ErrorHandler(error.response.message), 500);
    }
  })
```

couponCode.js

```
// create coupoun code
router.post(
  "/create-coupon-code",
  isSeller,
  catchAsyncErrors(async (req, res, next) => {
    try {
      const isCoupounCodeExists = await CoupounCode.find({
        name: req.body.name,
      });

      if (isCoupounCodeExists.length !== 0) {
        return next(new ErrorHandler("Coupoun code already exists!", 400));
      }

      const coupounCode = await CoupounCode.create(req.body);

      res.status(201).json({
        success: true,
        coupounCode,
      });
    } catch (error) {
      return next(new ErrorHandler(error, 400));
    }
  })
```

CreateEvent.js

```
// create event
router.post(
  "/create-event",
  upload.array("images"),
  catchAsyncErrors(async (req, res, next) => {
    try {
      const shopId = req.body.shopId;
      const shop = await Shop.findById(shopId);
      if (!shop) {
        return next(new ErrorHandler("Shop Id is invalid!", 400));
      } else {
        const files = req.files;
        const imageUrls = files.map((file) => `${file.filename}`);

        const eventData = req.body;
        eventData.images = imageUrls;
        eventData.shop = shop;

        const product = await Event.create(eventData);

        res.status(201).json({
          success: true,
```

Message.js

```
// create new message
router.post(
  "/create-new-message",
  upload.array("images"),
  catchAsyncErrors(async (req, res, next) => {
    try {
      const messageData = req.body;

      if (req.files) {
        const files = req.files;
        const imageUrls = files.map((file) => `${file.fileName}`);
        messageData.images = imageUrls;
      }

      messageData.conversationId = req.body.conversationId;
      messageData.sender = req.body.sender;
      messageData.text = req.body.text;

      const message = new Messages({
        conversationId: messageData.conversationId,
        text: messageData.text,
        sender: messageData.sender,
        images: messageData.images ? messageData.images : undefined,
      });
```

order.js

```
// create new order
router.post(
  "/create-order",
  catchAsyncErrors(async (req, res, next) => {
    try {
      const { cart, shippingAddress, user, totalPrice, paymentInfo } = req.body;

      //   group cart items by shopId
      const shopItemsMap = new Map();

      for (const item of cart) {
        const shopId = item.shopId;
        if (!shopItemsMap.has(shopId)) {
          shopItemsMap.set(shopId, []);
        }
        shopItemsMap.get(shopId).push(item);
      }

      // create an order for each shop
      const orders = [];

      for (const [shopId, items] of shopItemsMap) {
        const order = await Order.create({
          cart: items,
          shippingAddress,
          user,
          totalPrice,
```

payment.js

```
router.post(
  "/process",
  catchAsyncErrors(async (req, res, next) => {
    const myPayment = await stripe.paymentIntents.create({
      amount: req.body.amount,
      currency: "inr",
      metadata: {
        company: "Becodemy",
      },
    });
    res.status(200).json({
      success: true,
      client_secret: myPayment.client_secret,
    });
  })
);

router.get(
  "/stripeapikey",
  catchAsyncErrors(async (req, res, next) => {
    res.status(200).json({ stripeApikey: process.env.STRIPE_API_KEY });
  })
);


module.exports = router;
```

product.js

```
// create product
router.post(
  "/create-product",
  upload.array("images"),
  catchAsyncErrors(async (req, res, next) => {
    try {
      const shopId = req.body.shopId;
      const shop = await Shop.findById(shopId);
      if (!shop) {
        return next(new ErrorHandler("Shop Id is invalid!", 400));
      } else {
        const files = req.files;
        const imageUrls = files.map((file) => `${file.filename}`);

        const productData = req.body;
        productData.images = imageUrls;
        productData.shop = shop;

        const product = await Product.create(productData);

        res.status(201).json({
          success: true,
          product,
        });
      }
    } catch (error) {
      return next(new ErrorHandler(error, 400));
    }
  })
```

Shop.js

```
// create shop
router.post("/create-shop", upload.single("file"), async (req, res, next) => {
  try {
    const { email } = req.body;
    const sellerEmail = await Shop.findOne({ email });
    if (sellerEmail) {
      const filename = req.file.filename;
      const filePath = `uploads/${filename}`;
      fs.unlink(filePath, (err) => {
        if (err) {
          console.log(err);
          res.status(500).json({ message: "Error deleting file" });
        }
      });
      return next(new ErrorHandler("User already exists", 400));
    }

    const filename = req.file.filename;
    const fileUrl = path.join(filename);

    const seller = {
      name: req.body.name,
      email: email,
      password: req.body.password,
      avatar: fileUrl,
      address: req.body.address,
      phoneNumber: req.body.phoneNumber,
      zipCode: req.body.zipCode,
    };

    const activationToken = createActivationToken(seller);
```

user.js

```
router.post("/create-user", upload.single("file"), async (req, res, next) => {
  try {
    const { name, email, password } = req.body;
    const userEmail = await User.findOne({ email });

    if (userEmail) {
      const filename = req.file.filename;
      const filePath = `uploads/${filename}`;
      fs.unlink(filePath, (err) => {
        if (err) {
          console.log(err);
          res.status(500).json({ message: "Error deleting file" });
        }
      });
      return next(new ErrorHandler("User already exists", 400));
    }

    const filename = req.file.filename;
    const fileUrl = path.join(filename);

    const user = {
      name: name,
      email: email,
      password: password,
      avatar: fileUrl,
    };

    const activationToken = createActivationToken(user);
```

Technology (Framework, Languages)

Framework: MERN Stack

GitHub Repository
Link: https://github.com/mrif449/Papier-Factory