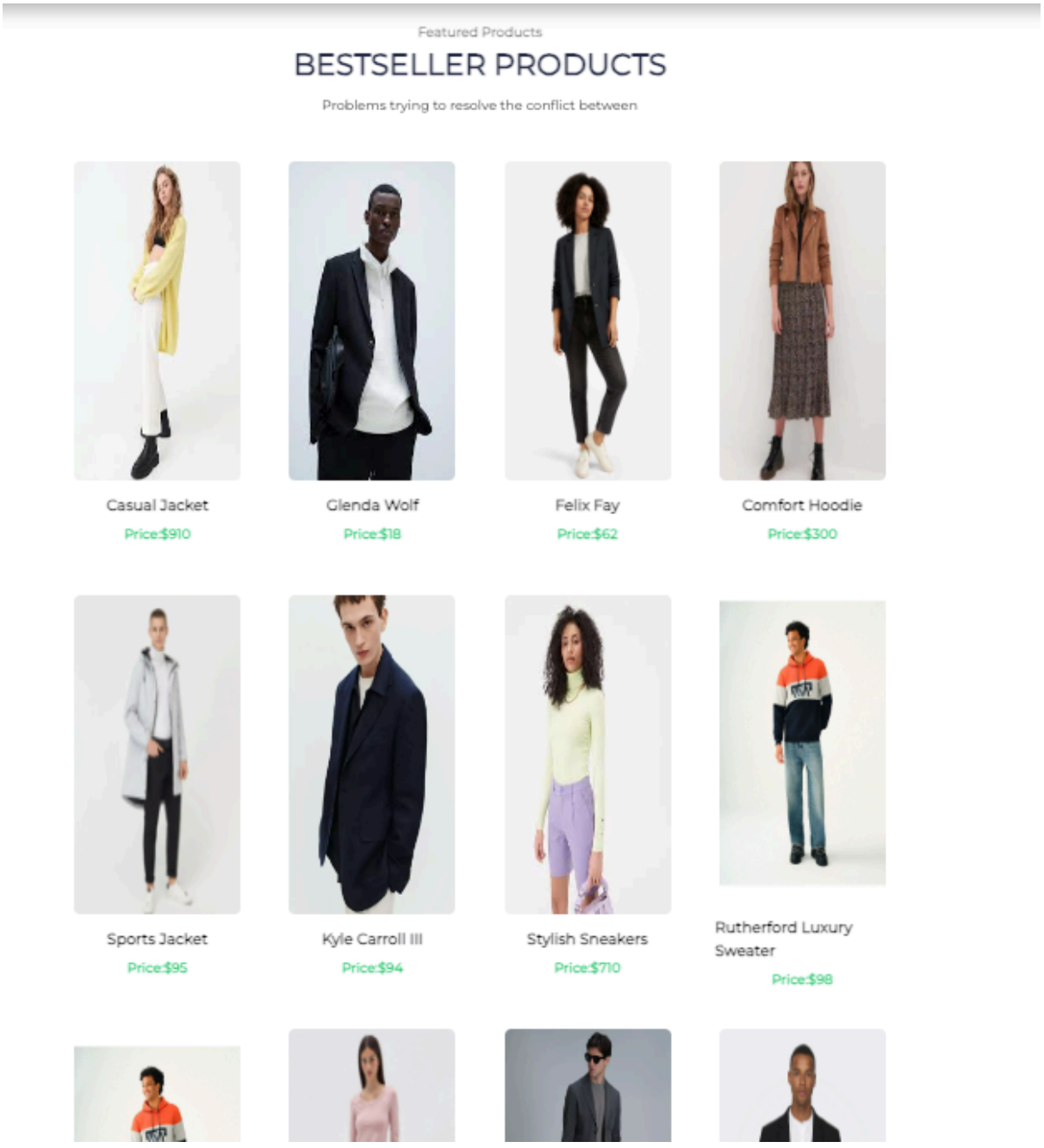
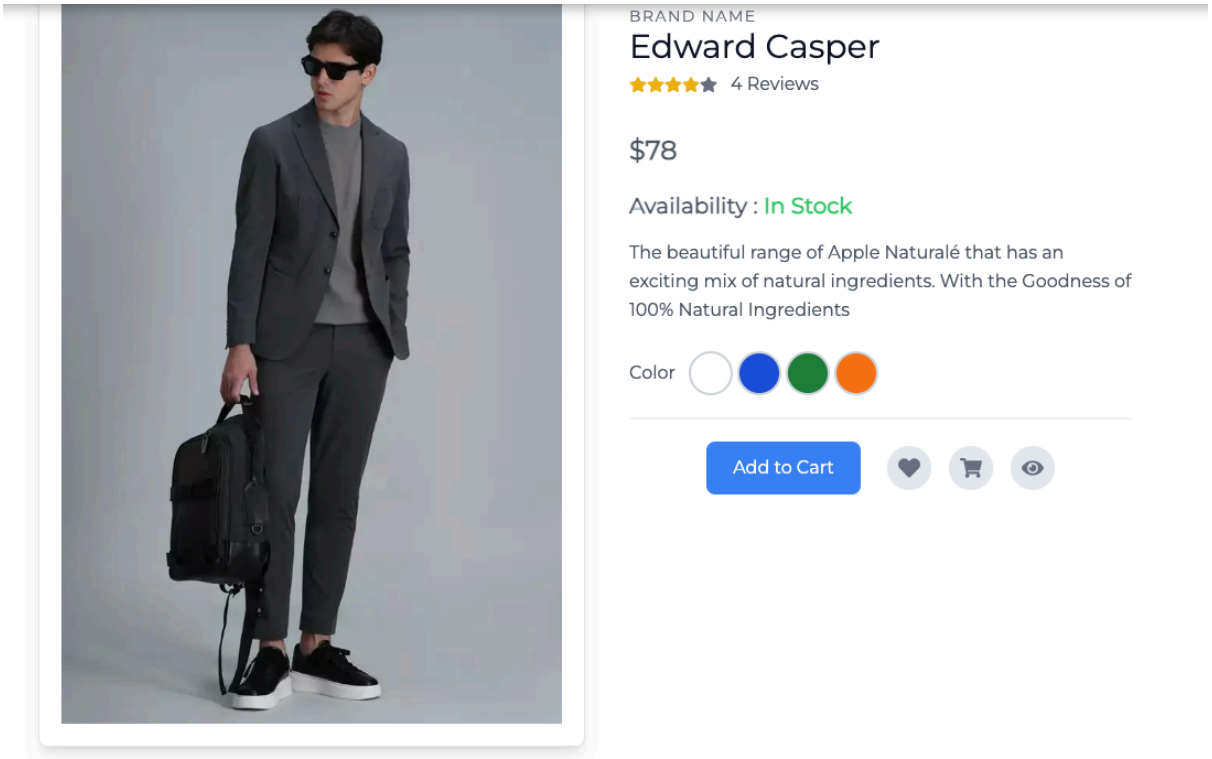
Day 4 - Dynamic Frontend Components [Bandage]

1. Functional Deliverables:

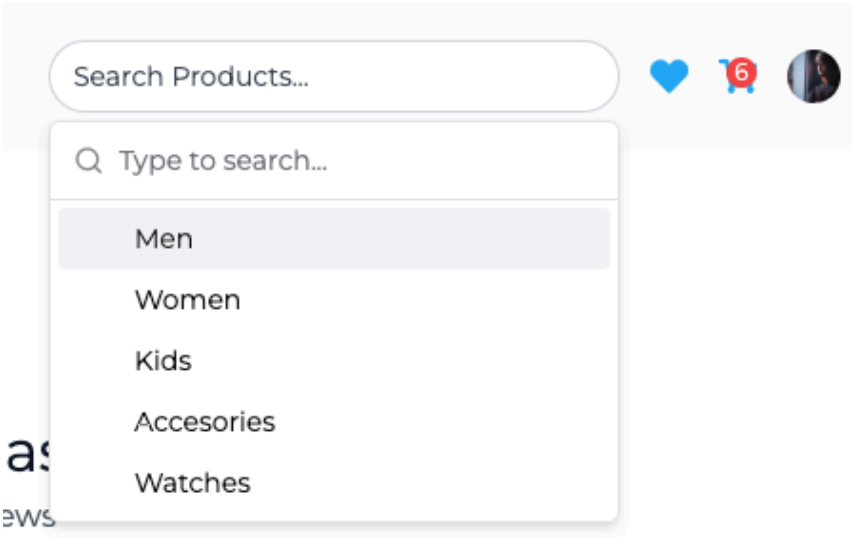
1. **Product Listing Page with Dynamic Data:**

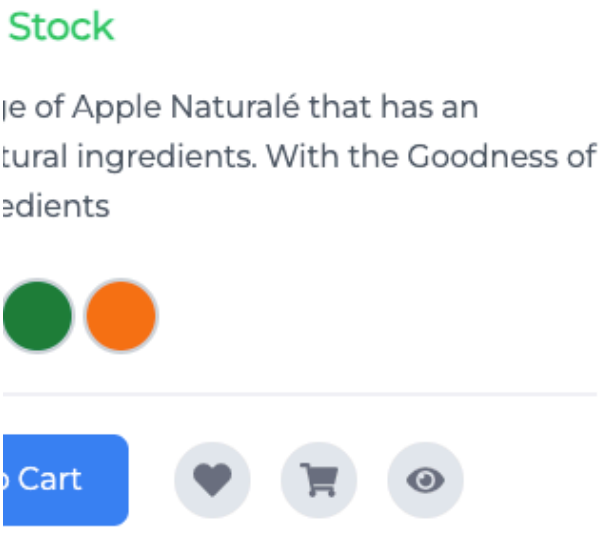


2. **Individual Product Detail Pages:**



3. **Working Search Bar**

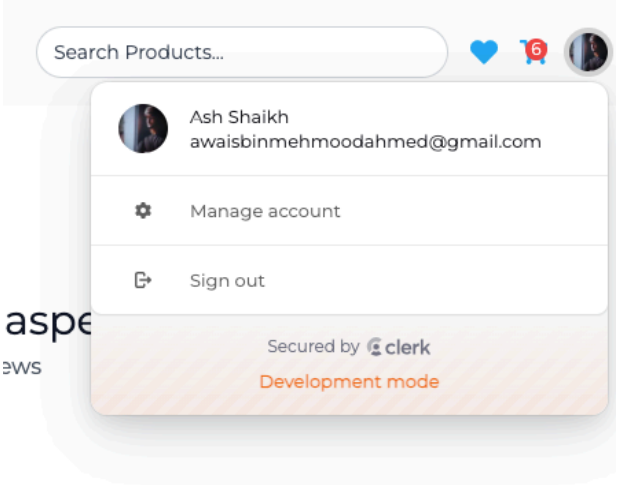




4. **Additional Features Implemented:**

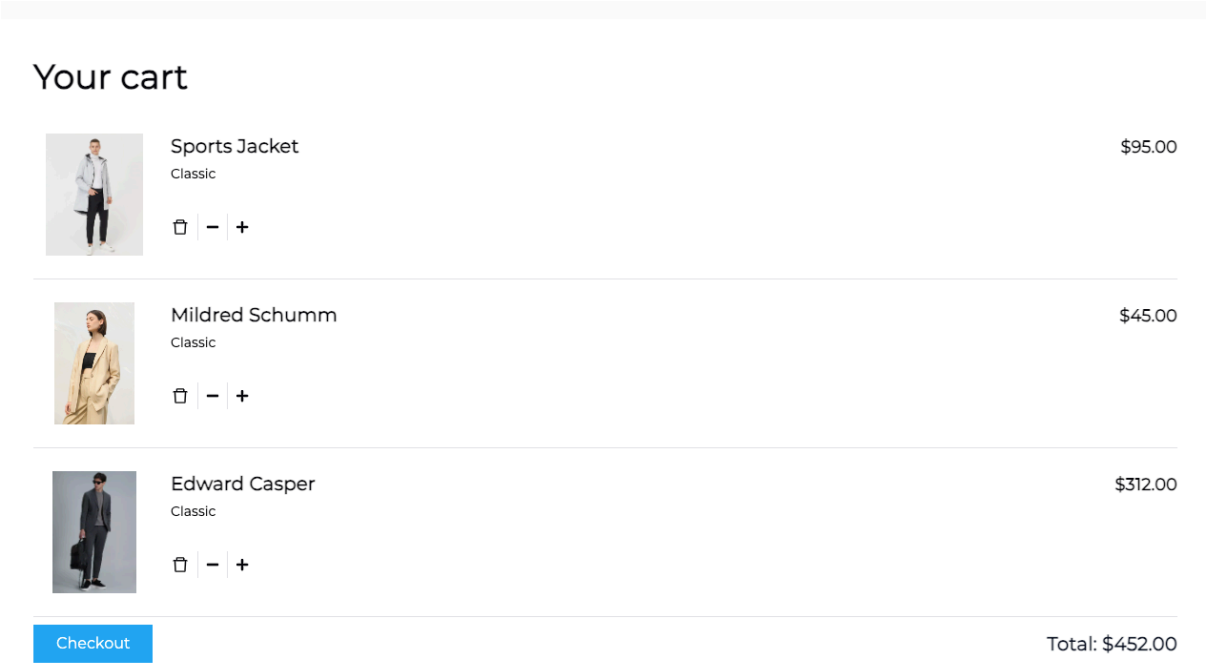
   - *Authentication using clerk*

     Users can sign-in , signout and manage thier account



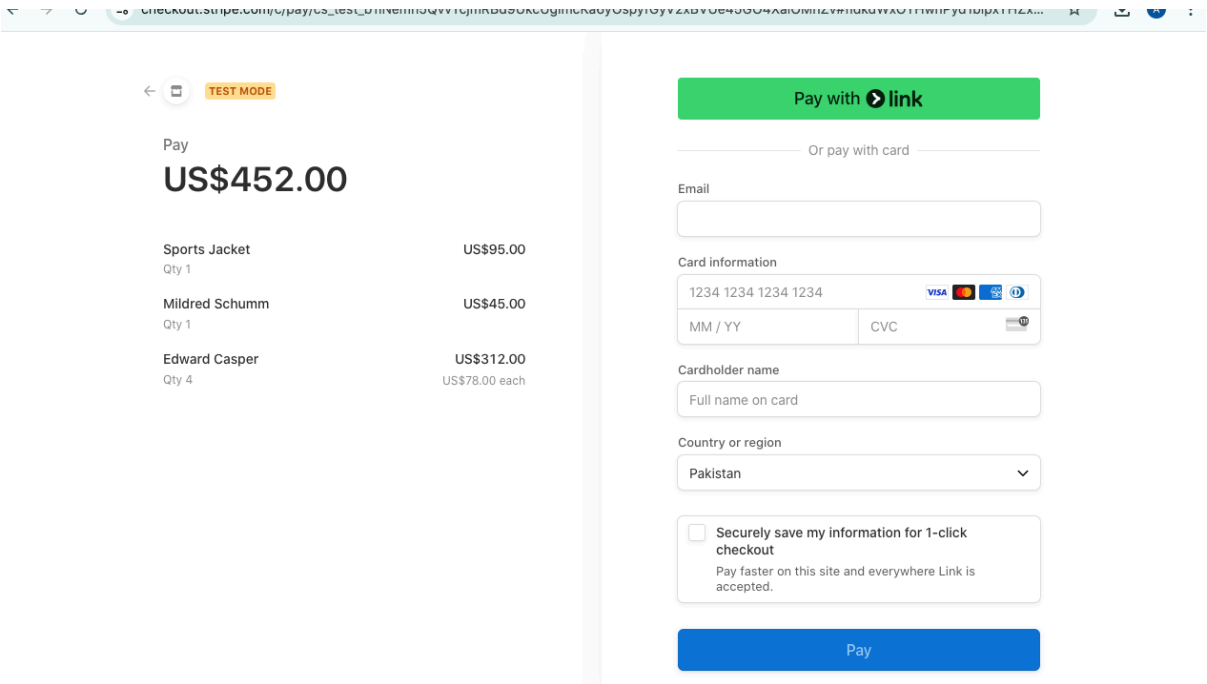
- *Add to cart functionality*

User can increment - decrement and delete product . Prodct will save in ther cart | in local storage until user delete by temselve

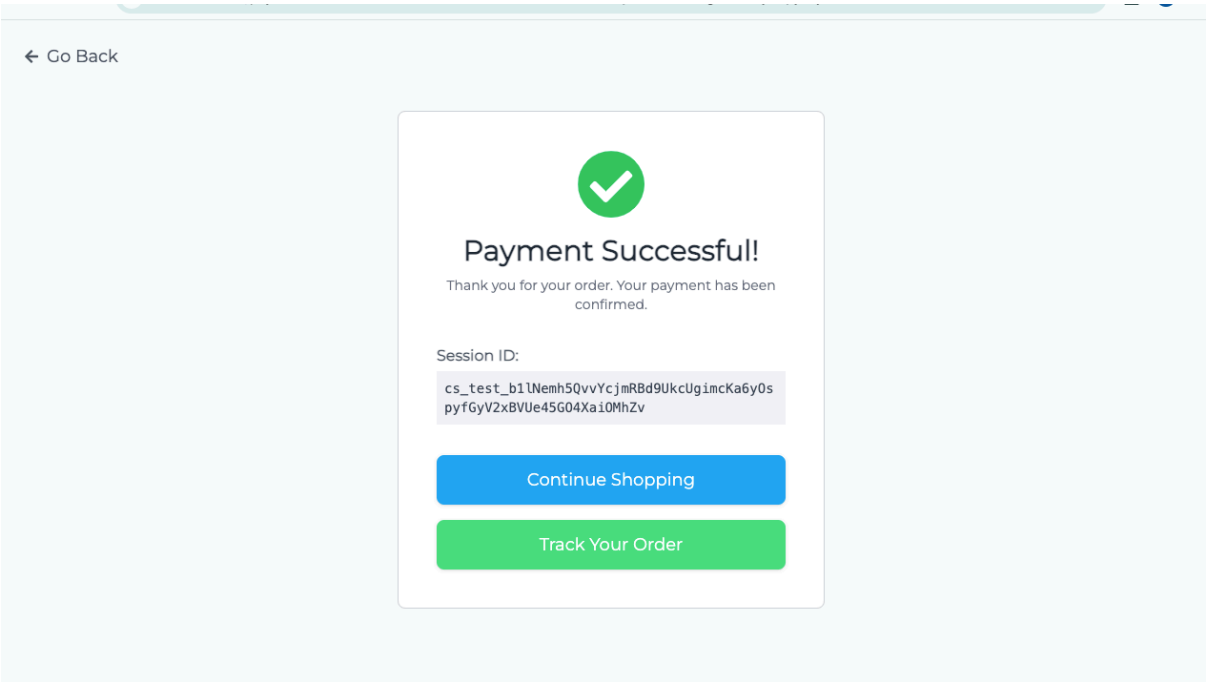


- *Checkout process using stripe*

  User can buy product with ease by paying through thier bank cards



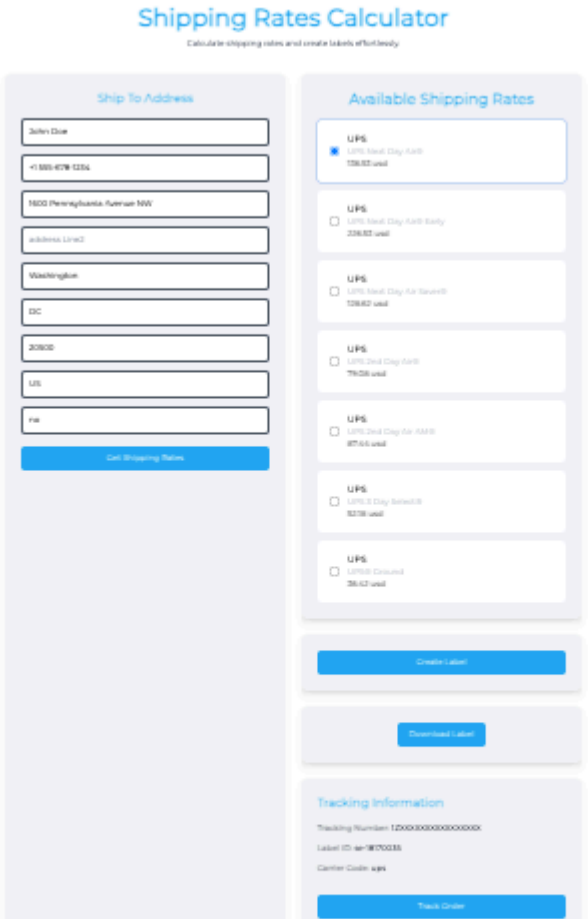
After paynment confirmation, user will navigate to order confirmtiona page and he would see options like Go back (to cart) , continue shopping (shop) and **track order**



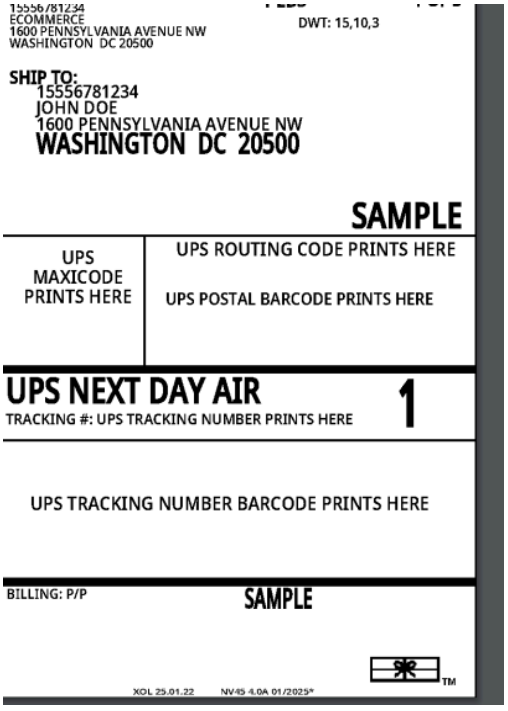
- Order tracking [Integrated using ship engine]

  User can get live location of his order and generte label id to track thier order

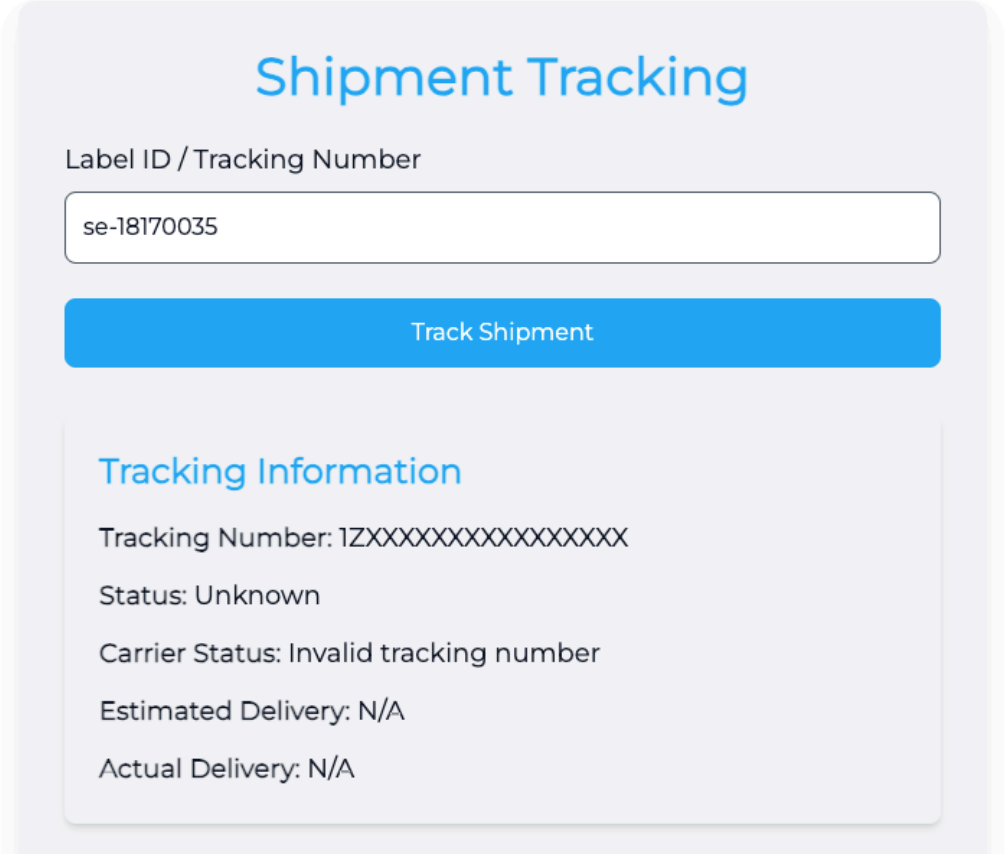
Note : Shipengine does not provide live tracking of order for free, client need to buy subcription



Label:



Order tracking:



2. Code Deliverables:

Key Components:

- **ProductCard Component:**

```tsx
"use client";

import { useCart } from "../cartContext";
import Image from "next/image";
import { FaPlus, FaMinus } from "react-icons/fa";
import { AiOutlineDelete } from "react-icons/ai";
import Navbar from "../components/Navbar";
import { useState, useEffect } from "react";
import { getProductById } from "../utils/api";
import { useRouter } from "next/navigation";

const Cart = () => {
  const { cartItems, removeFromCart, updateQuantity, setCartItems } = useCart();
  const [loading, setLoading] = useState(false);
  const router = useRouter();

  useEffect(() => {
    // Load cart items from localStorage on component mount
    const storedCart = localStorage.getItem("cartItems");
    if (storedCart) {
      setCartItems(JSON.parse(storedCart));
    }
  }, [setCartItems]);

  useEffect(() => {
    // Update localStorage whenever cartItems change
    localStorage.setItem("cartItems", JSON.stringify(cartItems));
  }, [cartItems]);

  const getTotal = () =>
    cartItems.reduce((total, item) => total + item.price * item.quantity, 0);

  const handleIncrement = (id: number) => {
    const item = cartItems.find((item) => item.id === id);
    if (item) {
      updateQuantity(id, item.quantity + 1);
    }
  };

  const handleDecrement = (id: number) => {
    const item = cartItems.find((item) => item.id === id);
    if (item && item.quantity > 1) {
      updateQuantity(id, item.quantity - 1);
    }
  };
```

```tsx
  const handleDecrement = (id: number) => {
    const item = cartItems.find((item) => item.id === id);
    if (item && item.quantity > 1) {
      updateQuantity(id, item.quantity - 1);
    }
  };

  const handleDelete = (id: number) => {
    removeFromCart(id);
  };

  const handleCheckout = async () => {
    setLoading(true);
    try {
      const products = await Promise.all(
        cartItems.map(async (item) => {
          const product = await getProductById(item.id.toString());
          return {
            name: product.name,
            amount: product.price * 100,
            currency: product.currency || "usd",
            quantity: item.quantity,
          };
        })
      );

      const response = await fetch("/api/checkout_sessions", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ products }),
      });

      if (response.ok) {
        const { url } = await response.json();
```

```
const handleCheckout = async () => {
    } else {
      const error = await response.json();
      console.error("Checkout Error:", error.message);
    }
  } catch (error) {
    console.error("Error during checkout:", error);
  } finally {
    setLoading(false);
  }
};

return (
  <>
    <Navbar />
    <div className="flex justify-center flex-col  p-6 space-y-4 sm:p-10 dark:bg-gray-50 dark:text-gray-800">
      <h2 className="text-4xl font-semibold">Your cart</h2>
      {cartItems.length === 0 ? (
        <p className="text-center text-gray-500 h-[60dvh] flex justify-center items-center text-3xl font-bold">
          Your cart is empty
        </p>
      ) : (
        <ul className="flex flex-col divide-y dark:divide-gray-300">
          {cartItems.map((item) => (
            <li
              key={item.id}
              className="flex flex-col py-6 sm:flex-row sm:justify-between"
            >
              <div className="flex w-full space-x-2 sm:space-x-4">
                <Image
                  src={item.image}
                  alt={item.name}
                  width={120}
                  height={150}
                  unoptimized
                  className="flex-shrink-0 object-contain w-20 h-20 dark:border- rounded outline-none sm:w-32 sm:h-32 dark:bg-gray-500"
                />
                <div className="flex flex-col justify-between w-full pb-4">
                  <div className="flex justify-between w-full pb-2 space-x-2">
                    <div className="space-y-1">
                      <h3 className="text-xl font-semibold leading-snug sm:pr-8">
                        {item.name}
                        </h3>
                        <p className="text-sm dark:text-gray-600">Classic</p>
                      </div>
                      <div className="text-right">
                        <p className="text-lg font-semibold">
                          ${(item.price * item.quantity).toFixed(2)}
                        </p>
                      </div>
                    </div>
                    <div className="flex text-sm divide-x">
                      <button
                        onClick={() => handleDelete(item.id)}
                        className="flex items-center px-2 py-1 pl-0 space-x-1"
                      >
                        <AiOutlineDelete size={20} />
                      </button>
                      <button
                        onClick={() => handleDecrement(item.id)}
                        className="flex items-center px-2 py-1 space-x-1"
                      >
                        <FaMinus />
                      </button>
                      <button
                        onClick={() => handleIncrement(item.id)}
                        className="flex items-center px-2 py-1 space-x-1"
                      >
                        <FaPlus />
                      </button>
                    </div>
                  </div>
                </div>
              </li>
            ))}

            <div className="flex items-center flex-wrap justify-between pt-2">
              <button
                className={`bg-Color py-2 px-6 text-white cursor-pointer hover:opacity-90 flex items-center justify-center ${
                  loading ? "opacity-70 cursor-not-allowed" : ""
                }`}
                onClick={handleCheckout}
                disabled={loading}
              >
                {loading ? (
                  <div className="flex items-center space-x-2">
                    <span className="spinner-border animate-spin inline-block w-4 h-4 border-4 rounded-full border-white border-t-transparent"></span>
                    <span>Processing...</span>
                  </div>
                ) : (
                  "Checkout"
                )}
              </button>
              <p className="text-xl font-bold text-gray-900">
                Total: ${getTotal().toFixed(2)}
              </p>
            </div>
          </ul>
        )}
      </div>
    </>
  );
};
```

ProductList Component:

```tsx
import React from "react";
import Heading from "./Heading";
import Image from "next/image";

;
import { client } from "@/sanity/lib/client";
import Link from "next/link";

interface type {
  _id: number;
  name: string;
  subtext: string;
  price: number;
  image: string;
}



async function getData(){
  const res = client.fetch(
    `*[_type == "product"]{
      _id,
      name,
      subtext,
      price,
      "image": image.asset->url
    }`
  );
  return res;
}


async function Features() {
  const data:type[] = await getData()


  return (
    <div className="container mx-auto p-4 max-w-[1050px]">
      {/* Heading */}
      <div>
        <Heading
          title="BESTSELLER PRODUCTS"
          subtitle="Featured Products"
          paragraph="Problems trying to resolve the conflict between"
          className="para"
        />
      </div>

      {/* Product Grid */}
      <div className="grid grid-cols-1 sm:grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4 mt-6">
        {data.map((product, index) => (
          <div
            key={index}
            className="rounded-lg p-4 flex flex-col items-center justify-center w-full"
          >
            {/* Product Image */}
            <Link href={`/add/${product._id}`}>
            <div className="flex justify-center mb-4 text-center group cursor-pointer">

                <Image
                  src={product.image}
                  alt={product.name}
                  width={200}
                  height={200}
                  unoptimized
                  className="rounded-lg h-96 object-fit cursor-pointer"
                />
            </div>
          </Link>
```

- **SearchBar Component:**

```tsx
"use client"

import * as React from "react"
import { Check } from "lucide-react"

import { cn } from "@/lib/utils"
import {
  Command,
  CommandEmpty,
  CommandGroup,
  CommandInput,
  CommandItem,
  CommandList,
} from "@/components/ui/command"
import {
  Popover,
  PopoverContent,
  PopoverTrigger,
} from "@/components/ui/popover"

const Search = [
  {
    value: "men",
    label: "Men",
  },
  {
    value: "Women",
    label: "Women",
  },
  {
    value: "Kids",
    label: "Kids",
  },
  {
    value: "Accesories",
    label: "Accesories",
  },
  {
    value: "Watches",
    label: "Watches",
  },
]
```

```
return (
  <Popover open={open} onOpenChange={setOpen}>
    <PopoverTrigger asChild>
      <div
        className={cn(
          "flex w-[300px] items-center rounded-full border border-gray-300 bg-white px-3 py-2 text-sm",
          "focus-within:ring focus-within:ring-blue-500 focus-within:ring-opacity-50"
        )}
        role="combobox"
        aria-expanded={open}
        tabIndex={0}
        onClick={() => setOpen((prev) => !prev)}
      >
        <span className="flex-grow text-gray-700">
          {value
            ? Search.find((Search) => Search.value === value)
                ?.label
            : "Search Products..."}
        </span>
      </div>
    </PopoverTrigger>
    <PopoverContent className="w-[300px] p-0">
      <Command>
        <CommandInput placeholder="Type to search..." />
        <CommandList>
          <CommandEmpty>No Product found.</CommandEmpty>
          <CommandGroup>
            {Search.map((Option) => (
              <CommandItem
                key={Option.value}
                value={Option.value}
                onSelect={(currentValue) => {
                  setValue(currentValue === value ? "" : currentValue)
                  setOpen(false)
                }}
              >
                <Check
                  className={cn(
                    "mr-2 h-4 w-4",
                    value === Option.value ? "opacity-100" : "opacity-0"
                  )}
                />
                {Option.label}
              </CommandItem>
            ))}
          </CommandGroup>
        </CommandList>
      </Command>
    </PopoverContent>
  </Popover>
)
}
```

Note : The searchbar is not working , i will make it functional later

API Integration & Dynamic Routing:

- **API Integration Logic:**

Stripe:(api/checkout_sessions/route.ts)

```ts
import { stripe } from "../../utils/getStripe";
import { NextResponse } from "next/server";

export async function POST(request: Request) {
  try {
    const { products } = await request.json();

    if (!products || !Array.isArray(products) || products.length === 0) {
      return NextResponse.json(
        { message: "Invalid product data" },
        { status: 400 }
      );
    }

    const lineItems = products.map((product) => ({
      price_data: {
        currency: product.currency,
        product_data: {
          name: product.name,
        },
        unit_amount: product.amount,
      },
      quantity: product.quantity,
    }));

    const session = await stripe.checkout.sessions.create({
      payment_method_types: ["card"],
      mode: "payment",
      line_items: lineItems,
      success_url: `${request.headers.get("origin")}/payment-confirmation?session_
      cancel_url: `${request.headers.get("origin")}/payment-cancelled`,
    });

    return NextResponse.json({ url: session.url });
  } catch (error: any) {
    console.error("Stripe Checkout Error:", error.message);
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}
```

Ship Engine:
(api/shipengine/get-rates/route.ts)

```ts
import { shipengine } from "../../../helper/shipEngine";
import { Address, Package } from "../../../type";
import { NextRequest } from "next/server";

export async function POST(req: NextRequest) {
  try {
    const {
      shipeToAddress,
      packages,
    }: { shipeToAddress: Address; packages: Package[] } = await req.json();

    // Validate required fields
    if (!shipeToAddress || !packages) {
      return new Response(
        JSON.stringify({
          error: "Missing required fields: shipeToAddress and packages",
        }),
        { status: 400 }
      );
    }

    const shipFromAddress: Address = {
      name: "John Doe",
    phone: "+1 555-678-1234",
    addressLine1: "1600 Pennsylvania Avenue NW",
    addressLine2: "",
    cityLocality: "Washington",
    stateProvince: "DC",
    postalCode: "20500",
    countryCode: "US",
    addressResidentialIndicator: "no",
    };

    // Fetch shipping rates from ShipEngine
    const shipmentDetails = await shipengine.getRatesWithShipmentDetails({
      shipment: {
        shipTo: shipeToAddress,
        shipFrom: shipFromAddress,
        packages: packages,
      },
      rateOptions: {
        carrierIds: [
          process.env.SHIPENGINE_FIRST_COURIER || "",
          process.env.SHIPENGINE_SECOND_COURIER || "",
          process.env.SHIPENGINE_THIRD_COURIER || "",
          process.env.SHIPENGINE_FOURTH_COURIER || "",
        ].filter(Boolean), // Remove empty strings
      },
    });

    // Log details for debugging
    console.log("Ship To Address:", shipeToAddress);
    console.log("Packages:", packages);
    console.log("Shipment Details:", shipmentDetails);

    // Return the response with shipment details
    return new Response(
      JSON.stringify({ shipeToAddress, packages, shipmentDetails }),
      { status: 200 }
    );
  } catch (error) {
    console.log("Error fetching shipping rates:", error)
    return new Response(JSON.stringify({ error: error }), {
      status: 500,
    });
  }
}
```

(api/shipengine/label/route.ts)

```ts
import { shipengine } from "../../../helper/shipEngine";
import { NextRequest, NextResponse } from "next/server";


export async function POST(req: NextRequest): Promise<NextResponse> {
  try {

    const { rateId } = await req.json();


    if (!rateId) {
      return NextResponse.json(
        { error: "rateId is required" },
        { status: 400 }
      );
    }


    const label = await shipengine.createLabelFromRate({
      rateId,
    });


    console.log("Label created successfully:", label);


    return NextResponse.json(label, { status: 200 });
  } catch (error) {

    console.error("Error creating label:", error);


    return NextResponse.json(
      { error: "An error occurred while creating the label" },
      { status: 500 }
    );
  }
}
```

(api/shipengine/tracikng/ [labelId] /route.ts)

```ts
import { shipengine } from "../../../../helper/shipEngine";
import { NextRequest, NextResponse } from "next/server";

export async function GET(
  req: NextRequest,
  { params }: {
  params: Promise<{ labelId: string }>
}) {
  const labelId = (await params).labelId;
  if (!labelId) {
    return new Response(JSON.stringify({ error: "Missing required fields"
      status: 400,
    });
  }

  try {

    const label = await shipengine.trackUsingLabelId(labelId);


    console.log(label);

    return NextResponse.json(label, { status: 200 });
  } catch (error) {
    console.log(error);
    return new Response(JSON.stringify({ error: error }), {
      status: 500,
    });
  }
}
```

Dynamic Routing Implementation:
(add/ [productId] /page.tsx)

```tsx
'use client'

import { client } from "../../../sanity/lib/client";
import { useParams } from "next/navigation";
import { useEffect, useState } from "react";
import { FaStar, FaHeart, FaShoppingCart, FaEye}from 'react-icons/fa';
import Image from "next/image";
import { useCart } from "../../cartContext";
import Navbar from '../../components/Navbar'
import CartTop from '../../components/CartTop'
import Quick from '../../components/Quick'
import Best from '../../components/BestSeller'
import Sponser from '../../components/Sponser'



interface Product {
  _id: number;
  name: string;
  description: string;
  image: string;
  price: number;
  currency: string;
}

async function getProductById(productId: string) {
  const res = await client.fetch(
    `*[_type == "product" && _id == "${productId}"]{
      _id,
      name,
      price,
      "image": image.asset->url,
      description,
      currency
    }`
  );
  return res[0];
}

const ProductDetails = () => {
  const { productId }:any = useParams();
  const [product, setProduct] = useState<Product | null>(null);
  const { addToCart } = useCart();

  useEffect(() => {
    if (productId) {
      getProductById(productId)
        .then((data) => {
          setProduct(data);
        })
        .catch((error) => {
          console.error("Error fetching product:", error);
        });
    }
  }, [productId]);

  if (!product) {
    return (
      <div className="flex justify-center items-center min-h-[80vh] text-5xl font-bold">
        Loading...
      </div>
    );
  }

  const handleAddToCart = () => {
    addToCart({
      id: product._id,
      name: product.name,
      price: product.price,
```

```jsx
const ProductDetails = () => {
  const handleAddToCart = () => {
    });
    alert(`${product.name} added to cart`);
  };

  return (
    <>
    <Navbar />
     <section className="text-gray-600 body-font overflow-hidden">
                <div className="container px-5 py-24 mx-auto">
            <div className="lg:w-4/5 mx-auto flex flex-wrap">
              <div className="lg:w-1/2 w-full flex flex-col items-center relative">
                <Image
                  alt="ecommerce"
                  className="rounded-lg  object-fit border p-5 cursor-pointer shadow-lg"
                  src={product.image} //
                  width={500}
                  height={500}
                />
              </div>
              <div className="lg:w-1/2 w-full lg:pl-10 lg:py-6 mt-6 lg:mt-0">
                <h2 className="text-sm title-font text-gray-500 tracking-widest">BRAND NAME</h2>
                <h1 className="text-gray-900 text-3xl title-font font-medium mb-1">{product.name}</h1>
                <div className="flex mb-4">
                  <span className="flex items-center">
                    <FaStar className="w-4 h-4 text-yellow-500" />
                    <FaStar className="w-4 h-4 text-yellow-500" />
                    <FaStar className="w-4 h-4 text-yellow-500" />
                    <FaStar className="w-4 h-4 text-yellow-500" />
                    <FaStar className="w-4 h-4 text-gray-500" />
                    <span className="text-gray-600 ml-3">4 Reviews</span>
                  </span>
                </div>
                <div className="py-4">
                  <span className="font-bold text-2xl">${product.price}</span>
                  <div className="font-bold mt-5 text-xl">
                    Availability : <span className="text-green-500">In Stock</span>
                  </div>
                </div>
                <p className="leading-relaxed">
                  {product.description}
                </p>
                <div className="flex mt-6 items-center pb-5 border-b-2 border-gray-100 mb-5">
                  <div className="flex items-center">
                    <span className="mr-3">Color</span>
                    <button className="border-2 border-gray-300 rounded-full w-10 h-10 focus:outline-none" />
                    <button className="border-2 border-gray-300 ml-1 bg-blue-700 rounded-full w-10 h-10 focus:outline-none" />
                    <button className="border-2 border-gray-300 ml-1 bg-green-700 rounded-full w-10 h-10 focus:outline-none" />
                    <button className="border-2 border-gray-300 ml-1 bg-orange-500 rounded-full w-10 h-10 focus:outline-none" />
                  </div>
                </div>
                <div className="flex justify-center items-center">
                  <button
                  onClick={handleAddToCart}
                  className="bg-blue-500 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition">
                   Add to Cart
                  </button>
                  <div className="flex items-center space-x-4 ml-6">
                    <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
                      <FaHeart className="w-5 h-5" />
                    </button>
                    <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
                      <FaShoppingCart className="w-5 h-5" />
                    </button>
                    <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
                      <FaEye className="w-5 h-5" />
                    </button>
                  </div>
```

```jsx
const ProductDetails = () => {
                  onClick={handleAddToCart}
                  className="bg-blue-500 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition">
                   Add to Cart
                  </button>
                  <div className="flex items-center space-x-4 ml-6">
                    <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
                      <FaHeart className="w-5 h-5" />
                    </button>
                    <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
                      <FaShoppingCart className="w-5 h-5" />
                    </button>
                    <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
                      <FaEye className="w-5 h-5" />
                    </button>
                  </div>
                </div>
              </div>
            </div>
          </div>

    </section>
    <CartTop />
      <Quick />
      <Best />
      <Sponser />
    </>
  );
};

export default ProductDetails;
```

Challenges Faced and Solutions Implemented

When integrating Clerk into my project, everything initially worked fine. However, I encountered an error when trying to navigate to my Sanity Studio. The error message read:

[Server] Error: Clerk: auth() was called but Clerk can't detect usage of clerkMiddleware(). Please ensure the following: - clerkMiddleware() is used in your Next.js Middleware. - Your Middleware matcher is configured to match this route or page. - If you are using the src directory, make sure the Middleware file is inside of it.

I spent nearly 5 hours troubleshooting this issue. Despite following all the steps outlined in the error message and related documentation, the issue persisted.

After extensive research and reviewing documentation, I discovered that the solution was to create separate layout pages. This not only resolved the issue but also brought additional benefits such as:

- Better Organization and Maintainability: The codebase became easier to manage with clear separation of concerns.
- Improved Performance: By structuring the layouts properly, unnecessary re-renders were minimized.

This experience reinforced the importance of reading documentation thoroughly and considering architectural improvements to address integration issues.

Best practices followed during development.

During development with Next.js and Tailwind CSS, I focused on creating a clean and user-friendly experience. I built reusable components like buttons and form elements to maintain consistency and save time. Tailwind's utility classes helped me design responsive layouts effortlessly, ensuring the site looked great on all devices. I prioritized easy navigation with a clear structure and accessible design, using semantic HTML and keyboard-friendly interactions. Customizing Tailwind's config for colors and fonts added a personal touch, while optimizing performance with Next.js features like lazy loading and dynamic imports kept the site fast and efficient.

And with that i have completed day 4! I'm super excited for the next hackathon task and can't wait to dive in. I'll keep pushing myself to create something amazing that meets marketplace standards. Let's make it happen!