

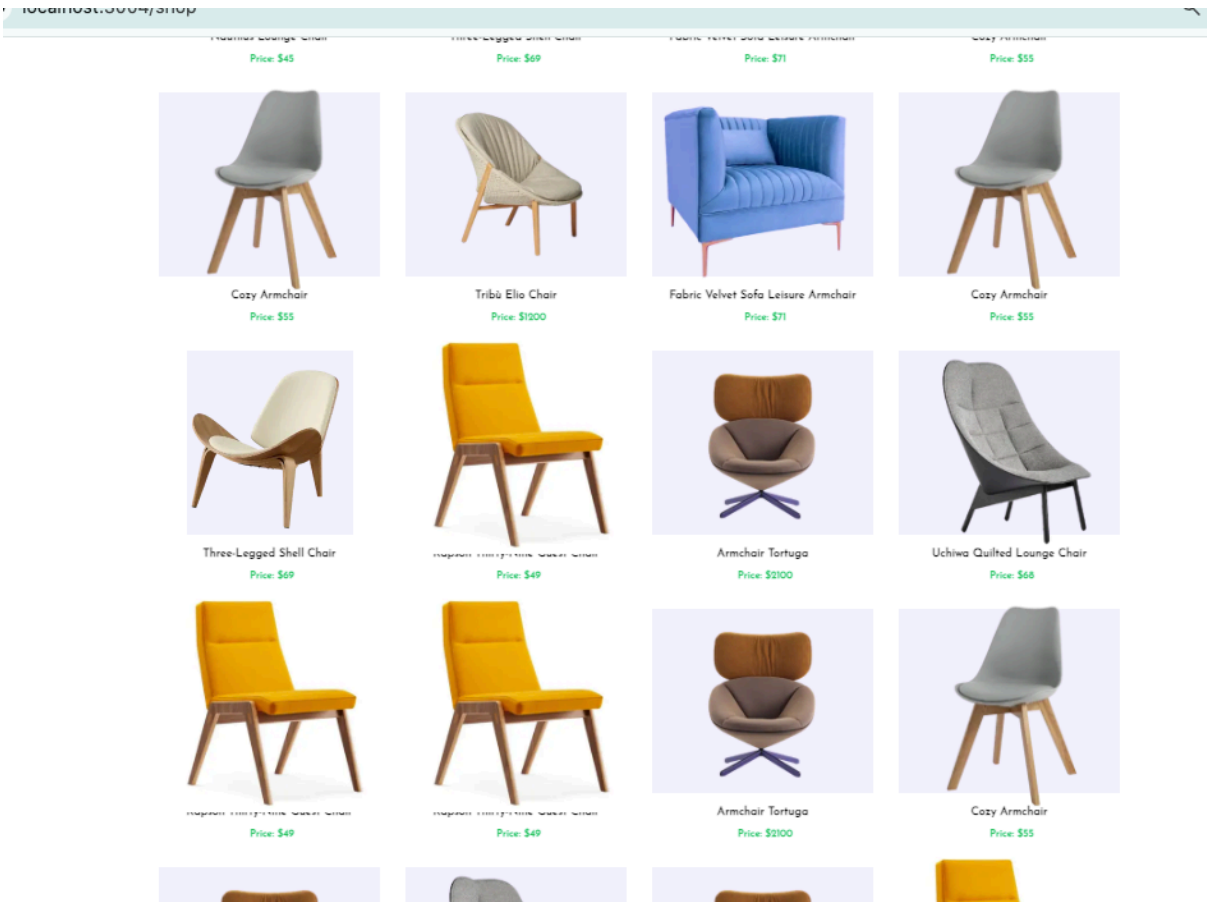
# Day 4 - Dynamic Frontend Components

## [Heckto]

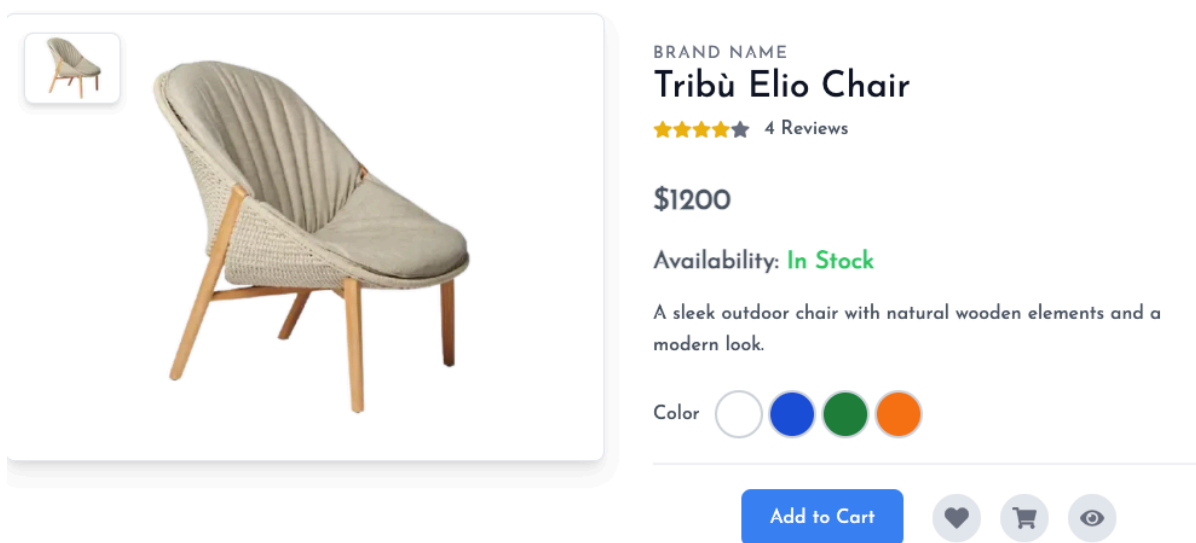
Having successfully integrated Sanity for data management, on Day 4, I plan to implement core frontend functionalities essential for every e-commerce website. This includes features like a dynamic product catalog, advanced filtering options, a responsive shopping cart, user authentication, and seamless interactions to enhance the overall user experience.

### 1. Functional Deliverables:

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



## 2. Individual Product Detail Pages:




## 3. Additional Features Implemented:

- *Authentication using kinde*

Users can sign-in, signout with ease  
heckto ecommerce

### Register

Get started today!

 Continue with Google

Or

First name

Last name

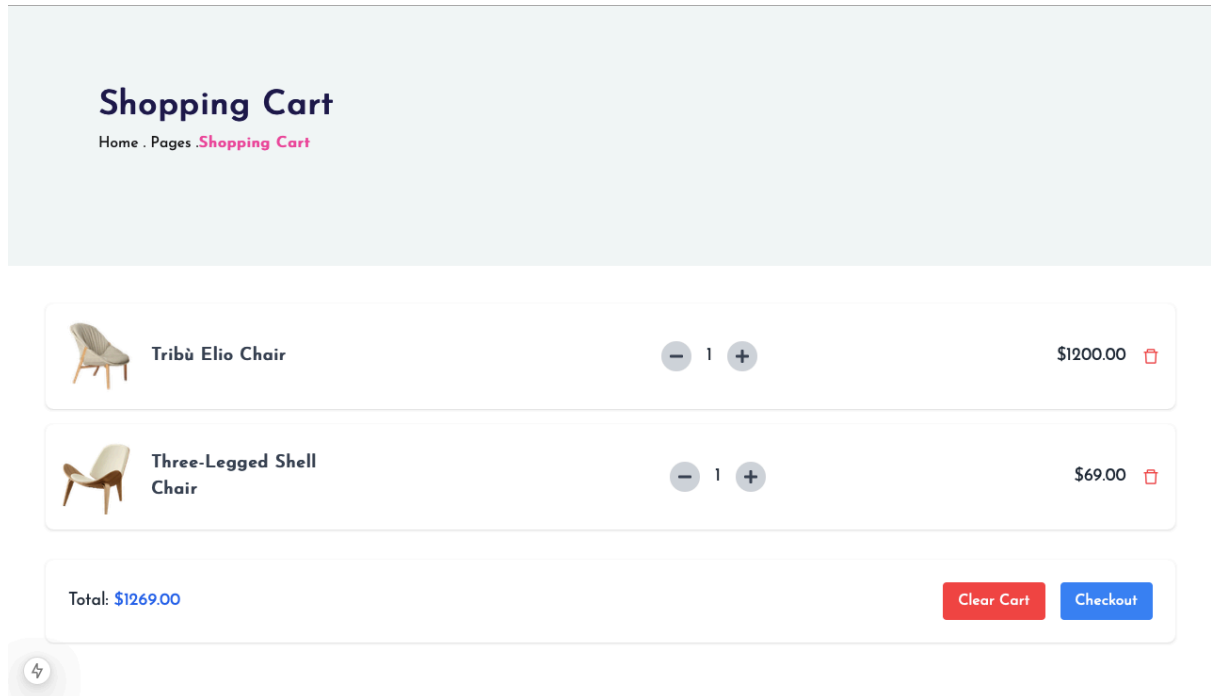
Email

Create your account

Already have an account? [Sign in](#)

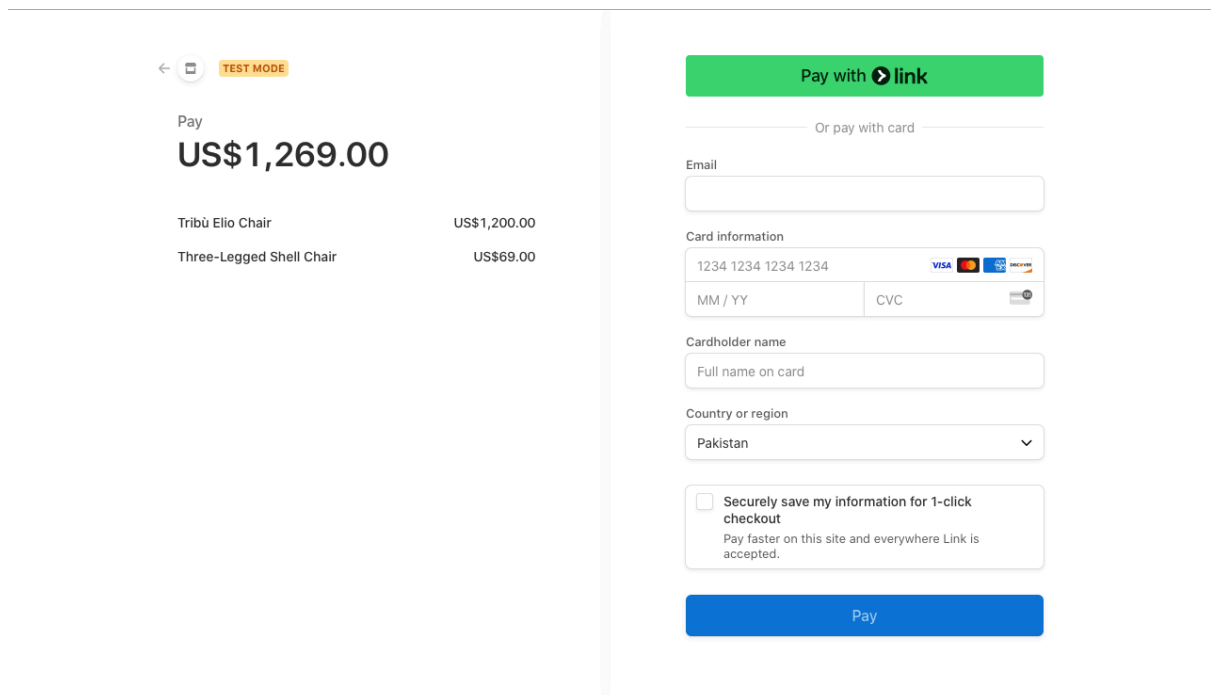
- *Add to cart functionality*

User can increment - decrement and delete product . Product will save in their cart | in local storage until user delete by themselves



- *Checkout process using stripe*

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



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

[← Go Back](#)



### Payment Successful!

Thank you for your order. Your payment has been processed successfully.

Session ID:

cs\_test\_b1wFRCIturQR4tAh3PSKdVQzFGm34R2KBzxSKqiUPU4PGivP  
btjbdmMYx6

[Continue Shopping](#)

1 error



## 2. Code Deliverables:

### Key Components:

- **ProductCard Component:**

```
"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);
    }
  };
};
```

```
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,
        };
      })
    );
  } catch (error) {
    console.error("Checkout error:", error);
  }

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

  if (!response.ok) {
    throw new Error("Checkout failed");
  }

  const session = await response.json();
  router.push(`/checkout/${session.id}`);
};
```

```

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>

```

```

                    </h3>
                    <p className="text-sm dark:text-gray-600">Classic</p>
                  </div>
                </div>
              </div>
            </li>
          </ul>
        )}
      <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={loading}
          disabled={loading}
          onClick={handleCheckout}
        >
          {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: {formatTotal().toFixed(2)}

```

### ProductList Component:

```

1 import React from "react";
2 import Heading from "../Heading";
3 import Image from "next/image";
4
5 ;
6 import { client } from "@sanity/lib/client";
7 import Link from "next/link";
8
9 interface type {
10   _id: number;
11   name: string;
12   subtext: string;
13   price: number;
14   image: string;
15 }
16
17
18
19 async function getData(){
20   const res = client.fetch(
21     `*[_type == "product"]{
22       _id,
23       name,
24       subtext,
25       price,
26       "image": image.asset->url
27     }`
28   );
29   return res;
30 }
31
32
33 async function Features() {
34   const datatype[] = await getData()
35
36
37   return (
38     <div className="container mx-auto p-4 max-w-[1050px]">
39       {<!-- Heading -->}
40       <div>
41         <Heading
42           title="BESTSELLER PRODUCTS"
43           subtitle="Featured Products"
44           paragraph="Problems trying to resolve the conflict between"
45           className="para"
46         />
47       </div>
48
49       {<!-- Product Grid -->}
50       <div className="grid grid-cols-1 sm:grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4 mt-6">
51         {data.map((product, index) => (
52           <div
53             key={index}
54             className="rounded-lg p-4 flex flex-col items-center justify-center w-full"
55           >
56             {<!-- Product Image -->}
57             <Link href={`/${add}/${product._id}`}>
58               <div className="flex justify-center mb-4 text-center group cursor-pointer">
59
60                 <Image
61                   src={product.image}
62                   alt={product.name}
63                   width={200}
64                   height={200}
65                   unoptimized
66                   className="rounded-lg h-96 object-fit cursor-pointer"
67                 />
68               </div>
69             </Link>

```

## API Integration & Dynamic Routing:

- API Integration Logic:

Stripe:(api/checkout\_sessions/route.ts)

```
1  import { stripe } from "../../utils/getStripe";
2  import { NextResponse } from "next/server";
3
4  export async function POST(request: Request) {
5    try {
6      const { products } = await request.json();
7
8      if (!products || !Array.isArray(products) || products.length === 0) {
9        return NextResponse.json(
10         { message: "Invalid product data" },
11         { status: 400 }
12       );
13     }
14
15     const lineItems = products.map((product) => ({
16       price_data: {
17         currency: product.currency,
18         product_data: {
19           name: product.name,
20         },
21         unit_amount: product.amount,
22       },
23       quantity: product.quantity,
24     }));
25
26     const session = await stripe.checkout.sessions.create({
27       payment_method_types: ["card"],
28       mode: "payment",
29       line_items: lineItems,
30       success_url: `${request.headers.get("origin")}/payment-confirmation?session_`,
31       cancel_url: `${request.headers.get("origin")}/payment-cancelled`,
32     });
33
34     return NextResponse.json({ url: session.url });
35   } catch (error: any) {
36     console.error("Stripe Checkout Error:", error.message);
37     return NextResponse.json({ message: error.message }, { status: 500 });
38   }
39 }
```



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

```
1  'use client'
2
3  import { client } from "../../../sanity/lib/client";
4  import { useParams } from "next/navigation";
5  import { useEffect, useState } from "react";
6  import { FaStar, FaHeart, FaShoppingCart, FaEye } from 'react-icons/fa';
7  import Image from "next/image";
8  import { useCart } from "../../../cartContext";
9  import Navbar from "../../../components/Navbar";
10 import CartTop from "../../../components/CartTop";
11 import Quick from "../../../components/Quick";
12 import Best from "../../../components/BestSeller";
13 import Sponser from "../../../components/Sponser";
14
15
16
17 interface Product {
18   _id: number;
19   name: string;
20   description: string;
21   image: string;
22   price: number;
23   currency: string;
24 }
25
26 async function getProductById(productId: string) {
27   const res = await client.fetch(
28     `*[_type == "product" && _id == "${productId}"]{
29     _id,
30     name,
31     price,
32     "image": image.asset->url,
33     description,
34     currency
35   }`
36   );
37   return res[0];
38 }
39
40 const ProductDetails = () => {
41   const { productId } = useParams();
42   const [product, setProduct] = useState<Product | null>(null);
43   const { addToCart } = useCart();
44
45   useEffect(() => {
46     if (productId) {
47       getProductById(productId)
48         .then((data) => {
49           setProduct(data);
50         })
51         .catch((error) => {
52           console.error("Error fetching product:", error);
53         });
54     }
55   }, [productId]);
56
57   if (!product) {
58     return (
59       <div className="flex justify-center items-center min-h-[80vh] text-5xl font-bold">
60         Loading...
61       </div>
62     );
63   }
64
65   const handleAddToCart = () => {
66     addToCart({
67       id: product._id,
68       name: product.name,
69       price: product.price,
```

```

60 const ProductDetails = () => {
61   const handleAddToCart = () => {
62     alert(`${product.name} added to cart`);
63   };
64   return (
65     <>
66       <Navbar />
67       <section className="text-gray-600 body-font overflow-hidden">
68         <div className="container px-5 py-24 mx-auto">
69           <div className="lg:w-4/5 mx-auto flex flex-wrap">
70             <div className="lg:w-1/2 w-full flex flex-col items-center relative">
71               <Image
72                 alt="ecommerce"
73                 className="rounded-lg object-fit border p-5 cursor-pointer shadow-lg"
74                 src={product.image} //
75                 width={500}
76                 height={500}
77               />
78             <div className="lg:w-1/2 w-full lg:pl-10 lg:py-6 mt-6 lg:mt-0">
79               <h2 className="text-sm title-font text-gray-500 tracking-widest">BRAND NAME</h2>
80               <h1 className="text-gray-900 text-3xl title-font font-medium mb-1">{product.name}</h1>
81               <div className="flex mb-4">
82                 <span className="flex items-center">
83                   <FaStar className="w-4 h-4 text-yellow-500" />
84                   <FaStar className="w-4 h-4 text-yellow-500" />
85                   <FaStar className="w-4 h-4 text-yellow-500" />
86                   <FaStar className="w-4 h-4 text-yellow-500" />
87                   <FaStar className="w-4 h-4 text-yellow-500" />
88                 </span>
89                 <span className="text-gray-600 ml-3">4 Reviews</span>
90               </div>
91               <div className="py-4">
92                 <span className="font-bold text-2xl">${product.price}</span>
93                 <div className="font-bold mt-5 text-xl">
94                   Availability : <span className="text-green-500">In Stock</span>
95                 </div>
96               </div>
97               <p className="leading-relaxed">
98                 {product.description}
99               </p>
100               <div className="flex mt-6 items-center pb-5 border-b-2 border-gray-100 mb-5">
101                 <div className="flex items-center">
102                   <span className="mr-3">Color</span>
103                   <button className="border-2 border-gray-300 rounded-full w-10 h-10 focus:outline-none" />
104                   <button className="border-2 border-gray-300 ml-1 bg-blue-700 rounded-full w-10 h-10 focus:outline-none" />
105                   <button className="border-2 border-gray-300 ml-1 bg-green-700 rounded-full w-10 h-10 focus:outline-none" />
106                   <button className="border-2 border-gray-300 ml-1 bg-orange-500 rounded-full w-10 h-10 focus:outline-none" />
107                 </div>
108               </div>
109               <div className="flex justify-center items-center">
110                 <button
111                   onClick={handleAddToCart}
112                   className="bg-blue-500 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition"
113                   Add to Cart
114                 </button>
115                 <div className="flex items-center space-x-4 ml-6">
116                   <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
117                     <FaHeart className="w-5 h-5" />
118                   </button>
119                   <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
120                     <FaShoppingCart className="w-5 h-5" />
121                   </button>
122                   <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
123                     <FaEye className="w-5 h-5" />
124                   </button>
125                 </div>
126               </div>
127             </div>
128           </div>
129         </div>
130       </section>
131       <CartTop />
132       <Quick />
133       <Best />
134       <Sponser />
135     </>
136   );
137 }
138 export default ProductDetails;

```

```

60 const ProductDetails = () => {
61   onClick={handleAddToCart}
62   className="bg-blue-500 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition"
63   Add to Cart
64 </button>
65 <div className="flex items-center space-x-4 ml-6">
66   <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
67     <FaHeart className="w-5 h-5" />
68   </button>
69   <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
70     <FaShoppingCart className="w-5 h-5" />
71   </button>
72   <button className="rounded-full w-10 h-10 bg-gray-200 p-0 border-0 inline-flex items-center justify-center text-gray-500">
73     <FaEye className="w-5 h-5" />
74   </button>
75 </div>
76 </div>
77 </div>
78 </div>
79 </section>
80 <CartTop />
81 <Quick />
82 <Best />
83 <Sponser />
84 </>
85 }
86 export default ProductDetails;

```

# Challenges Faced and Solutions Implemented

When integrating **Stripe** into my project, everything initially worked fine. However, I encountered an error when attempting to process payments. The error message read:

Error: Missing required param: `payment_method`.

I spent several hours troubleshooting this issue, going through the Stripe documentation, and debugging my code. Despite ensuring that all parameters were being passed, the error persisted.

After thorough investigation and research, I discovered the root cause: I was not correctly attaching the payment method ID to the payment intent. The issue was resolved by updating my API call to include the `payment_method` parameter explicitly. This solution not only resolved the issue but also brought additional benefits, such as:

1. **Improved Understanding of Stripe's API Workflow:** Diving deep into the Stripe documentation helped me better understand how payment intents and payment methods work together.
2. **Enhanced Error Handling:** I implemented more robust error handling to ensure similar issues could be caught early.
3. **Efficient Debugging Skills:** This experience reinforced the importance of testing API calls with proper parameters and leveraging logging tools to identify issues.

This challenge highlighted the significance of thoroughly reading API documentation and methodically debugging integration issues to achieve a seamless payment experience.

## Best Practices Followed During Development

While working with **Next.js** and **Tailwind CSS**, I focused on delivering a clean, user-friendly experience. To maintain consistency and efficiency, I built reusable components such as buttons and form elements. Leveraging Tailwind's utility classes, I designed responsive layouts with ease, ensuring a polished look across all devices. I prioritized accessibility by implementing a clear structure, semantic HTML, and keyboard-friendly navigation. Customizing Tailwind's configuration for colors and fonts added a unique touch, while performance optimization through Next.js features like lazy loading and dynamic imports ensured a fast and seamless experience.

With that, I've successfully wrapped up **Day 4!** I'm thrilled for the next hackathon challenge and eager to dive in. My goal remains to push boundaries and create something that aligns with marketplace standards. Let's make it amazing!

