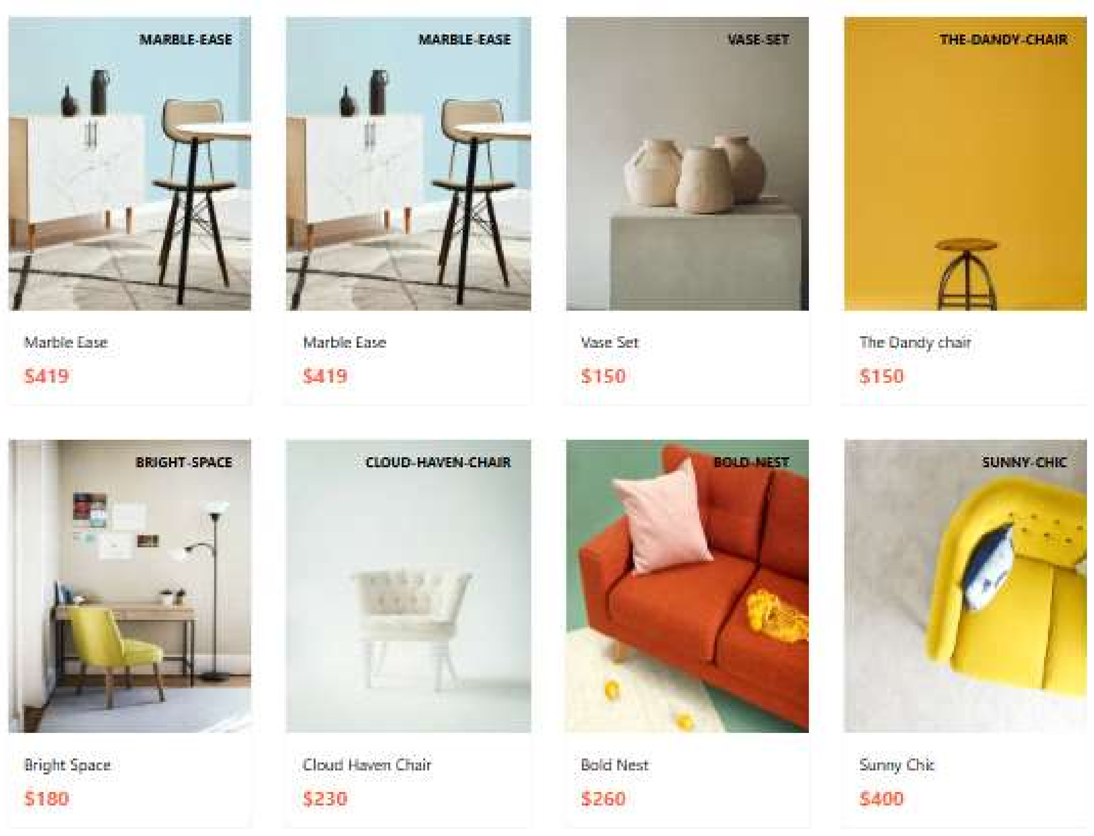
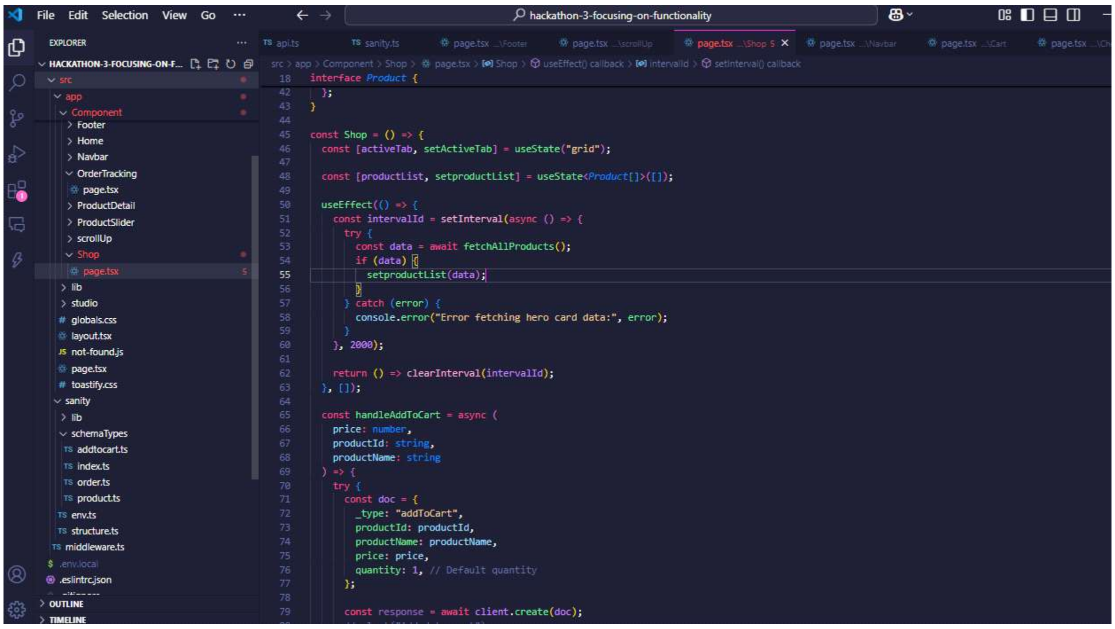
Day 4 - Dynamic Frontend Components - Sinp Ecommerce

Functional Deliverables:

1) Product listing page with dynamic data:

The product listing page seamlessly retrieves and showcases product data from Sanity CMS or APIs in real time. Each product is beautifully presented in an eye-catching card format, featuring its vibrant image, compelling name, and competitive price.

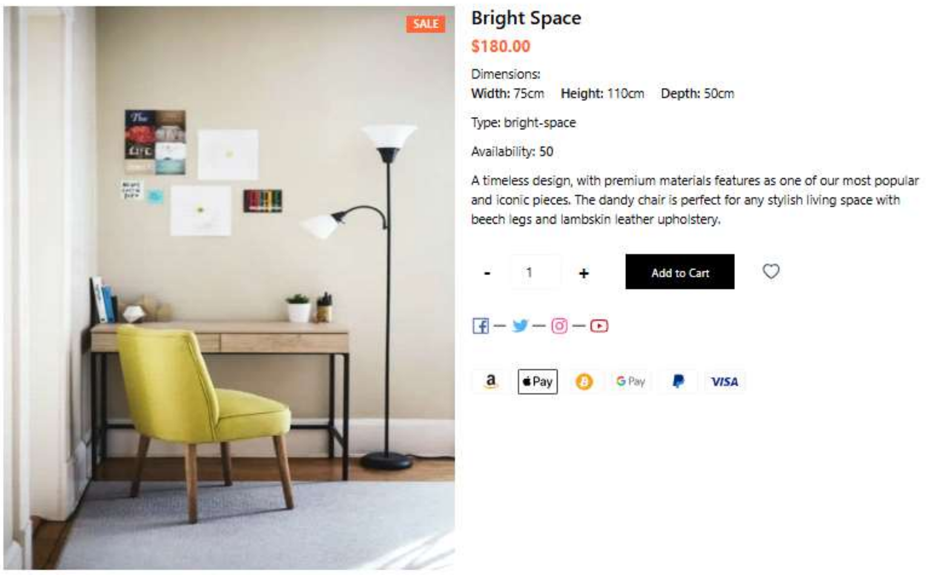



hackathon-3-focusing-on-functionality
src > app > Component > Shop > page.tsx > Shop > useEffect() callback > intervalId > setInterval() callback

```
interface Product {
  };
}

const Shop = () => {
  const [activeTab, setActiveTab] = useState("grid");

  const [productList, setproductList] = useState<Product[]>([]);

  useEffect(() => {
    const intervalId = setInterval(async () => {
      try {
        const data = await fetchAllProducts();
        if (data) {
          setproductList(data);
        }
      } catch (error) {
        console.error("Error fetching hero card data:", error);
      }
    }, 2000);

    return () => clearInterval(intervalId);
  }, []);

  const handleAddToCart = async (
    price: number,
    productId: string,
    productName: string
  ) => {
    try {
      const doc = {
        _type: "addToCart",
        productId: productId,
        productName: productName,
        price: price,
        quantity: 1, // Default quantity
      };

      const response = await client.create(doc);
```

2) Individual product detail pages:

Each product detail page dynamically fetches and renders data using dynamic routing ([id].ts). The page is uniquely crafted to showcase product-specific details, including the product name, image, description, dimensions, and more. This ensures a personalized and engaging shopping experience for the users.



Bright Space

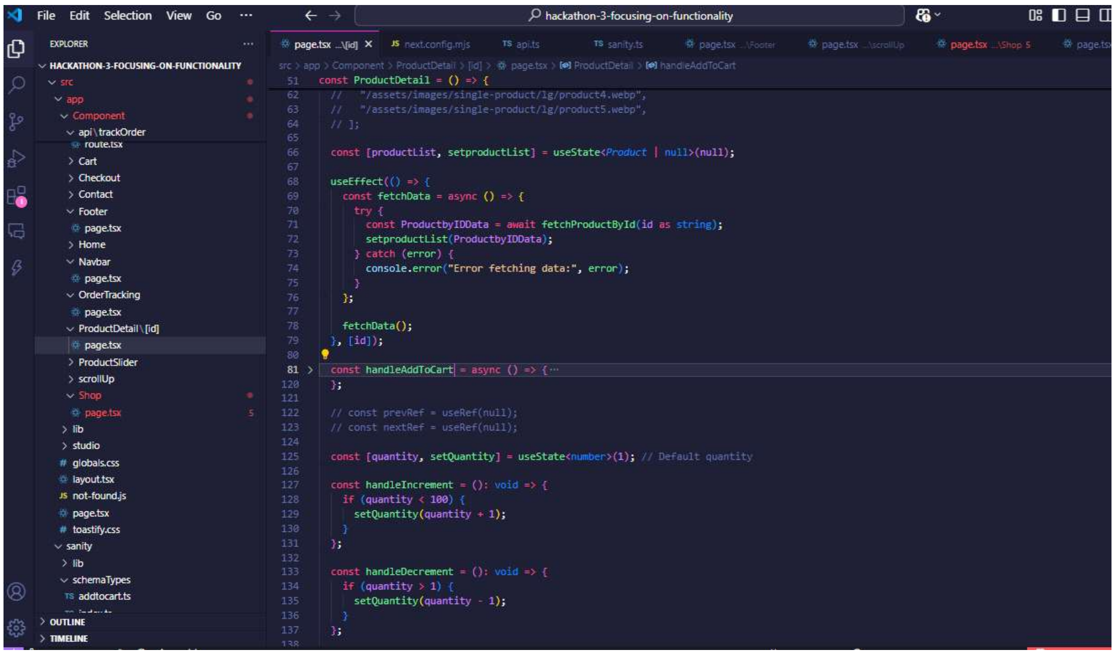
$180.00

Dimensions:
Width: 75cm Height: 110cm Depth: 50cm

Type: bright-space

Availability: 50

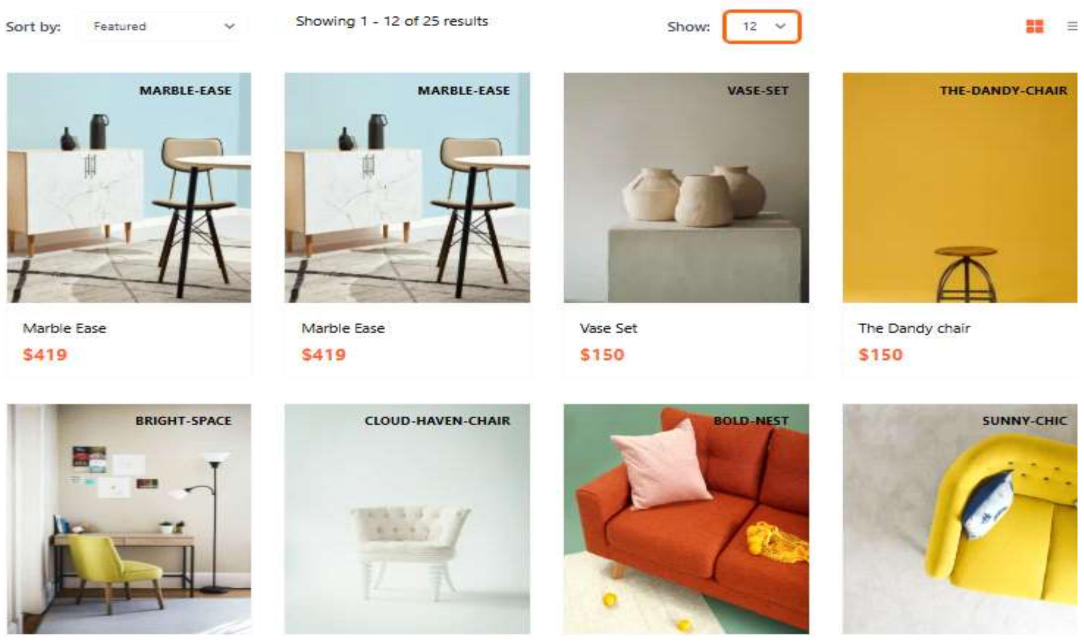
A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

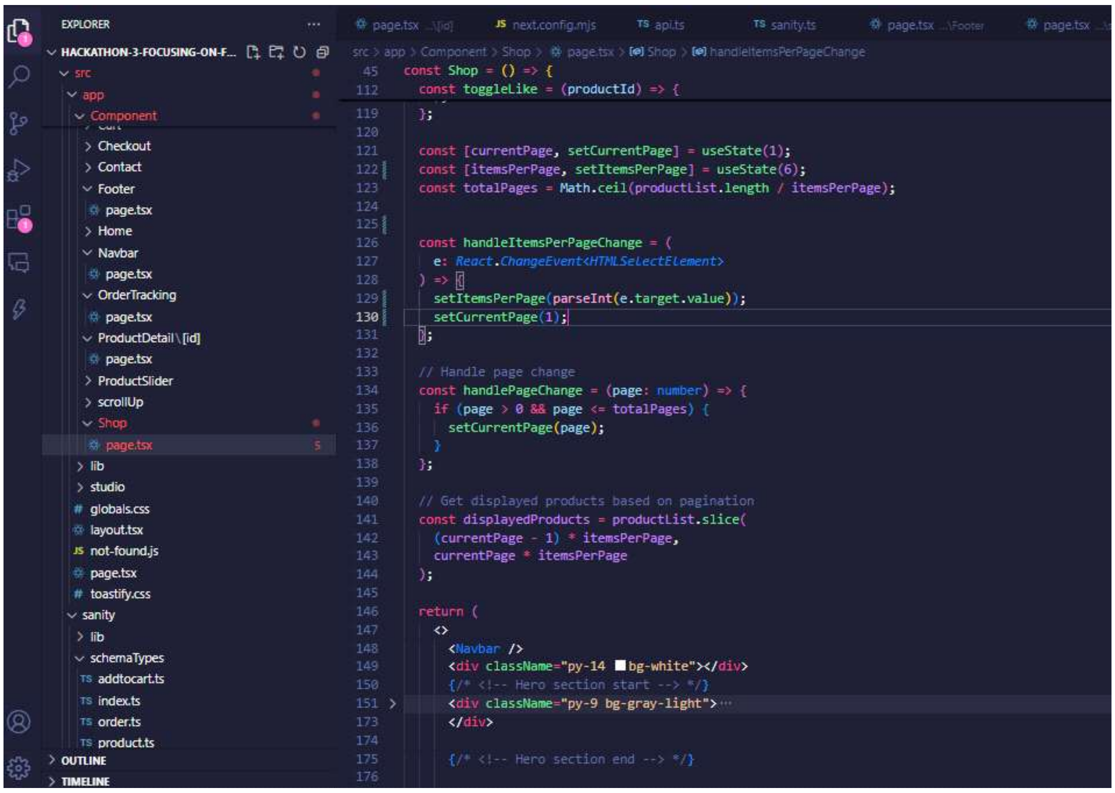


```tsx
const ProductDetail = () => {
  //   "/assets/images/single-product/lg/product4.webp",
  //   "/assets/images/single-product/lg/product5.webp",
  // ];

  const [productList, setproductList] = useState<Product | null>(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const ProductbyIDData = await fetchProductById(id as string);
        setproductList(ProductbyIDData);
      } catch (error) {
        console.error("Error fetching data:", error);
      }
    };

    fetchData();
  }, [id]);

  const handleAddToCart = async () => {···
  };

  // const prevRef = useRef(null);
  // const nextRef = useRef(null);

  const [quantity, setQuantity] = useState<number>(1); // Default quantity

  const handleIncrement = (): void => {
    if (quantity < 100) {
      setQuantity(quantity + 1);
    }
  };

  const handleDecrement = (): void => {
    if (quantity > 1) {
      setQuantity(quantity - 1);
    }
  };
```

3) Working Pagination:

The product listing page is equipped with a fully functional pagination system, allowing users to seamlessly navigate through multiple pages of products. The pagination dynamically loads product data for the selected page, ensuring an efficient and user-friendly browsing experience.

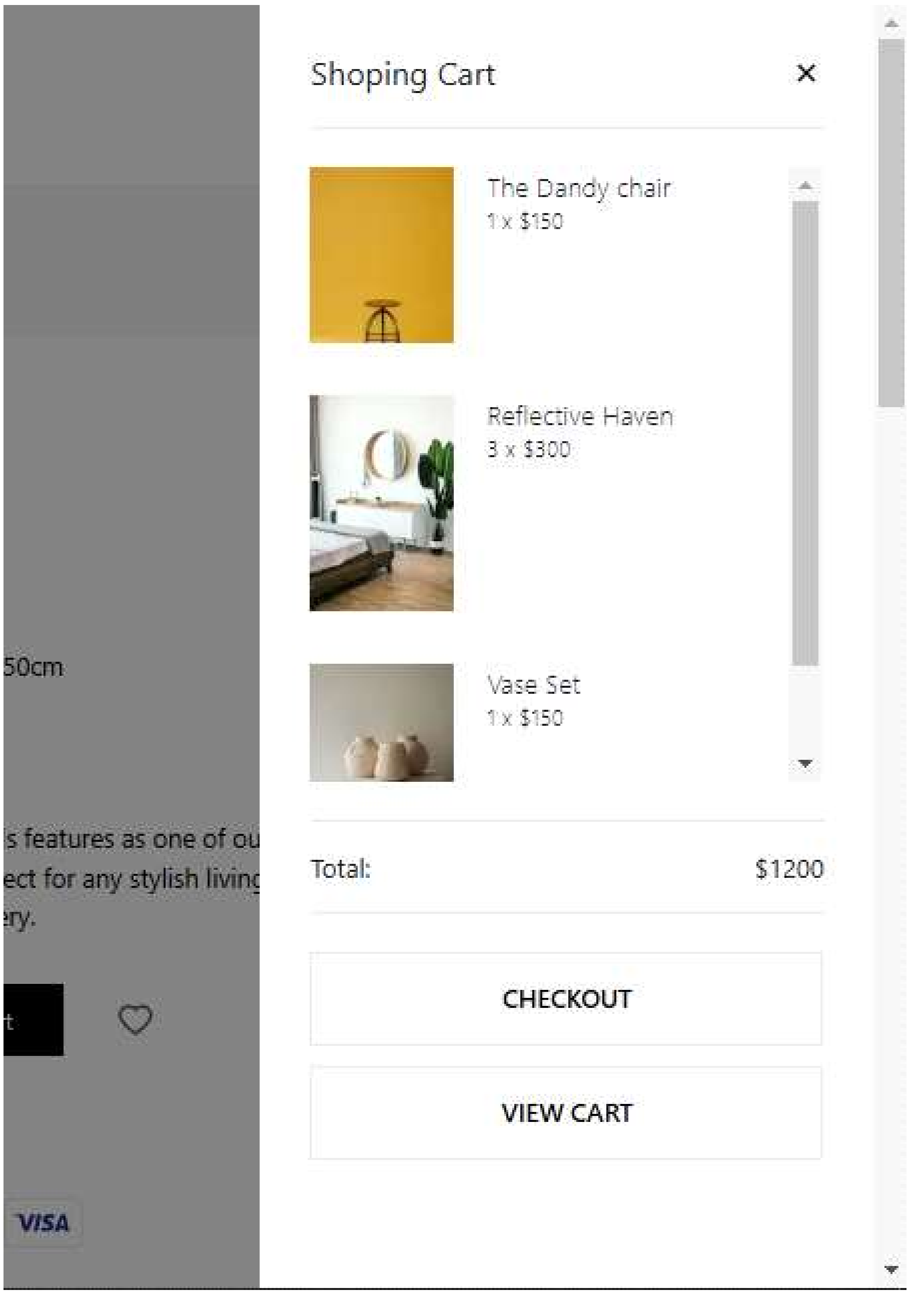




```
const Shop = () => {
  const toggleLike = (productId) => {
  };

  const [currentPage, setCurrentPage] = useState(1);
  const [itemsPerPage, setItemsPerPage] = useState(6);
  const totalPages = Math.ceil(productList.length / itemsPerPage);


  const handleItemsPerPageChange = (
    e: React.ChangeEvent<HTMLSelectElement>
  ) => {
    setItemsPerPage(parseInt(e.target.value));
    setCurrentPage(1);
  };

  // Handle page change
  const handlePageChange = (page: number) => {
    if (page > 0 && page <= totalPages) {
      setCurrentPage(page);
    }
  };

  // Get displayed products based on pagination
  const displayedProducts = productList.slice(
    (currentPage - 1) * itemsPerPage,
    currentPage * itemsPerPage
  );

  return (
    <>
      <Navbar />
      <div className="py-14 bg-white"></div>
      {/* <!-- Hero section start --> */}
      <div className="py-9 bg-gray-light">···
      </div>

      {/* <!-- Hero section end --> */}
```

4) Add to Cart Functionality:

Implementing a seamless Add to Cart functionality, this feature allows users to easily add products to their cart from the product listing page. It dynamically updates the cart state, enabling users to view their selected items and proceed to checkout. This enhances the shopping experience by making product selection and cart management smooth and efficient.



50cm

s features as one of ou
ect for any stylish living
ery.

t

VISA

Shoping Cart ×

The Dandy chair
1 x $150

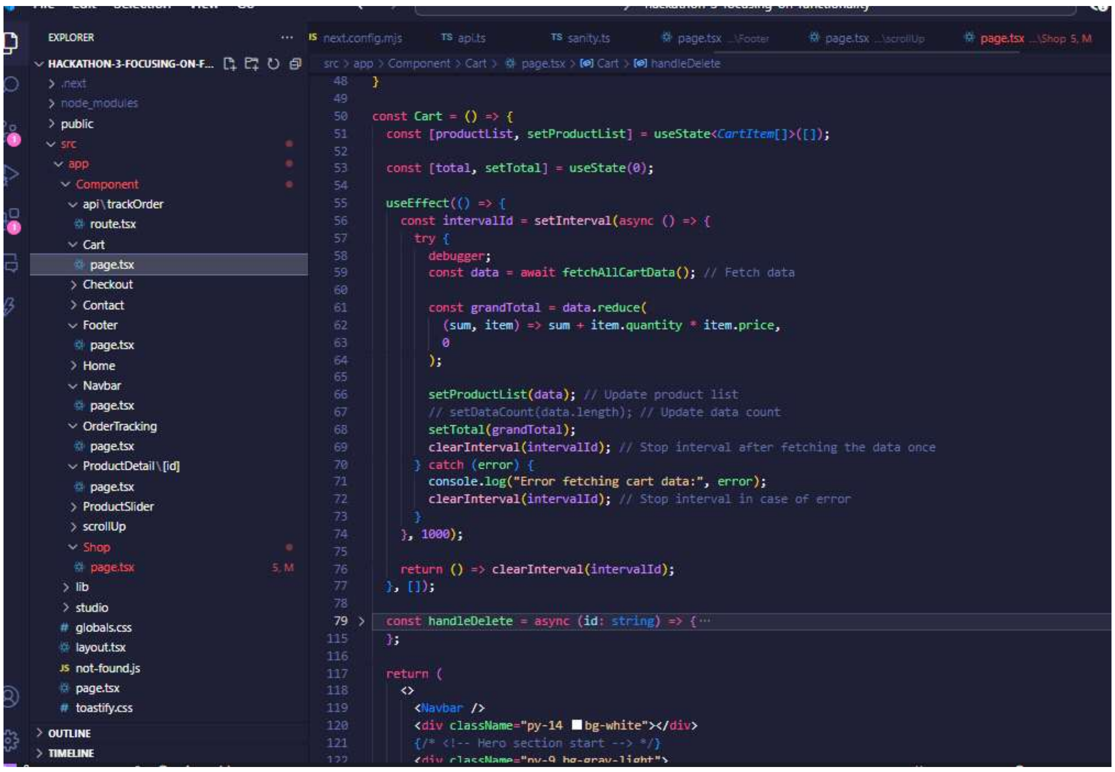
Reflective Haven
3 x $300

Vase Set
1 x $150

Total: $1200

CHECKOUT

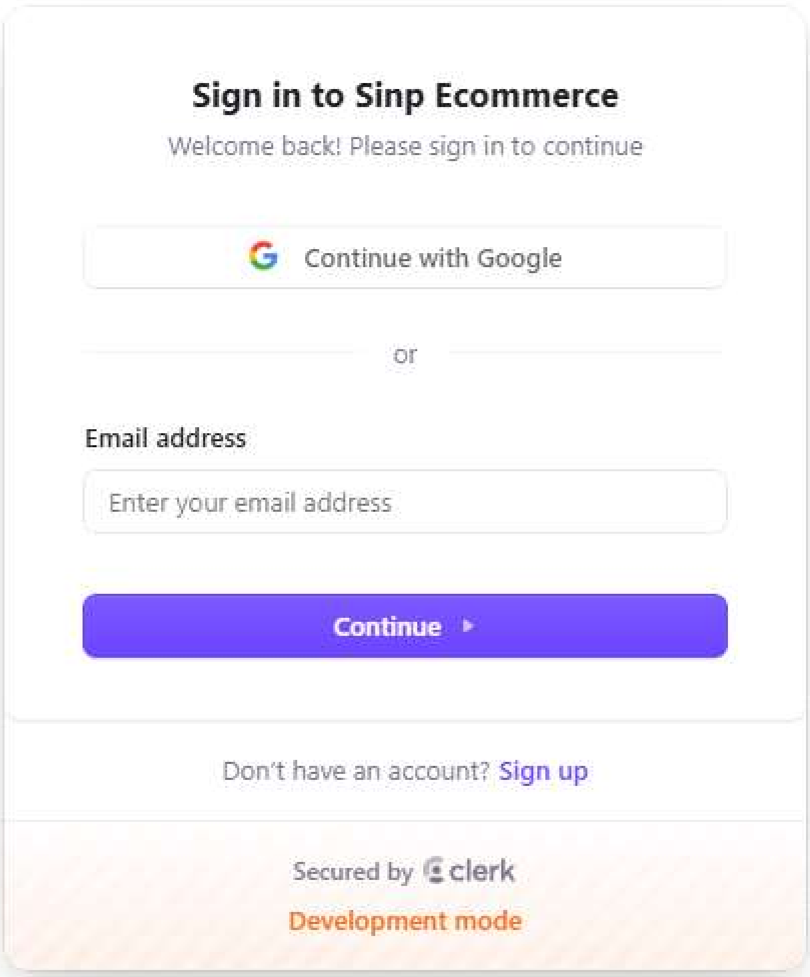
VIEW CART



```tsx
}

const Cart = () => {
  const [productList, setProductList] = useState<CartItem[]>([]);

  const [total, setTotal] = useState(0);

  useEffect(() => {
    const intervalId = setInterval(async () => {
      try {
        debugger;
        const data = await fetchAllCartData(); // Fetch data

        const grandTotal = data.reduce(
          (sum, item) => sum + item.quantity * item.price,
          0
        );

        setProductList(data); // Update product list
        // setDataCount(data.length); // Update data count
        setTotal(grandTotal);
        clearInterval(intervalId); // Stop interval after fetching the data once
      } catch (error) {
        console.log("Error fetching cart data:", error);
        clearInterval(intervalId); // Stop interval in case of error
      }
    }, 1000);

    return () => clearInterval(intervalId);
  }, []);

  const handleDelete = async (id: string) => {…
  };

  return (
    <>
      <Navbar />
      <div className="py-14 bg-white"></div>
      {/* <!-- Hero section start --> */}
      <div className="py-9 bg-gray-light">
```

5) Authentication using Clerk:

Clerk authentication offers a seamless solution for user management in applications. By integrating Clerk into your project, you can easily implement sign-in, sign-up, and user profile features using pre-built UI components. It allows for efficient user authentication, real-time user data retrieval, and secure sign-out functionality, making it a great choice for enhancing your app's security and user experience.


Sign in to Sinp Ecommerce

Welcome back! Please sign in to continue

Continue with Google

or

Email address

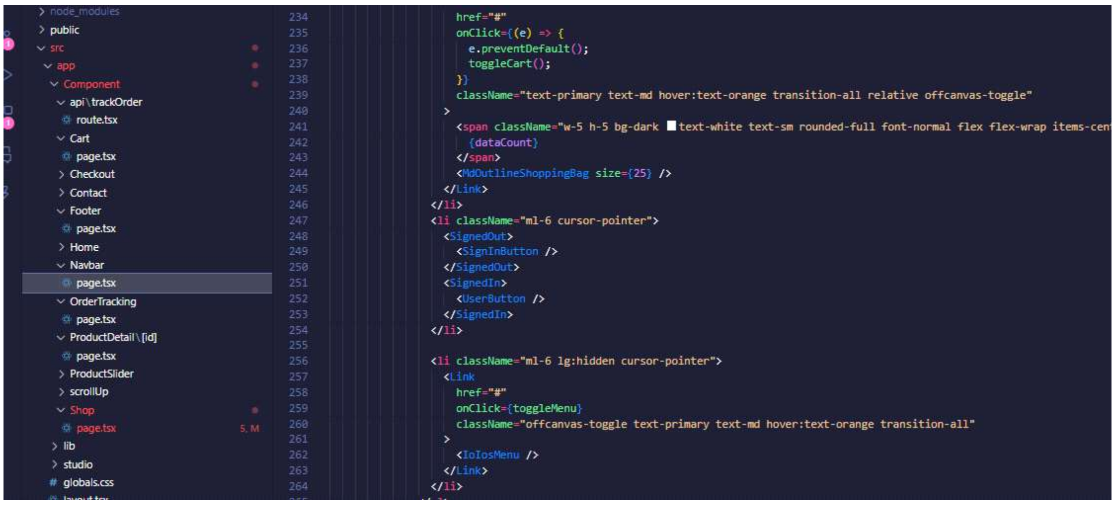
Enter your email address

Continue

Don't have an account? Sign up

Secured by clerk

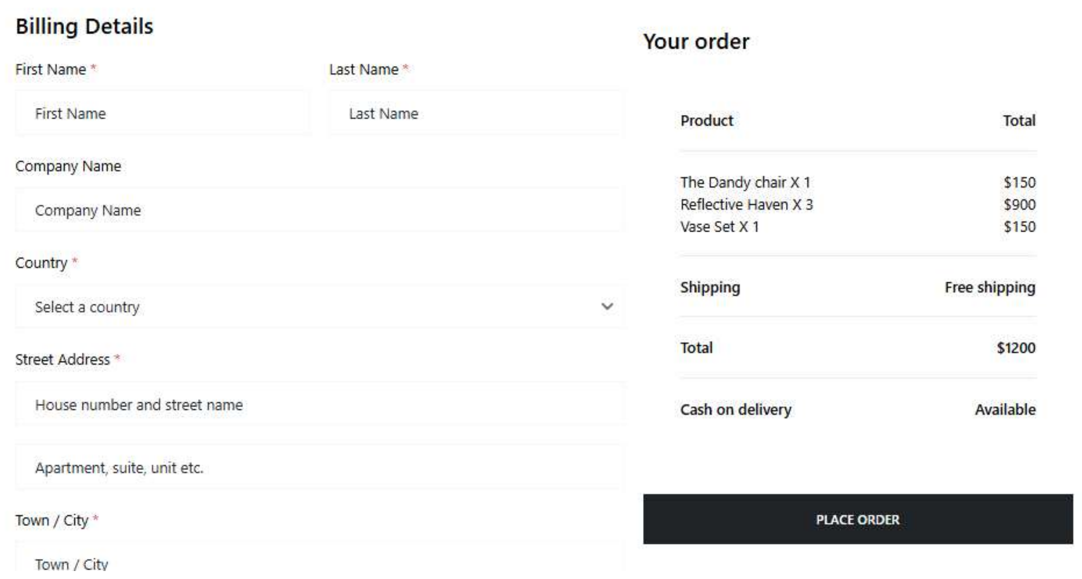
Development mode



```
        href="#"
        onClick={(e) => {
          e.preventDefault();
          toggleCart();
        }}
        className="text-primary text-md hover:text-orange transition-all relative offcanvas-toggle"
      >
        <span className="w-5 h-5 bg-dark text-white text-sm rounded-full font-normal flex flex-wrap items-cen
          {dataCount}
        </span>
        <MdOutlineShoppingBag size={25} />
      </Link>
    </li>
    <li className="ml-6 cursor-pointer">
      <SignedOut>
        <SignInButton />
      </SignedOut>
      <SignedIn>
        <UserButton />
      </SignedIn>
    </li>

    <li className="ml-6 lg:hidden cursor-pointer">
      <Link
        href="#"
        onClick={toggleMenu}
        className="offcanvas-toggle text-primary text-md hover:text-orange transition-all"
      >
        <IoIosMenu />
      </Link>
    </li>
```

6) Checkout Page Dynamic:

The Dynamic Checkout Page is designed with a clean, user-friendly layout. On the left side, users can easily fill out their Billing Details, while on the right side, they can view their Order Summary, including product details and total cost. This layout provides a seamless, intuitive experience for completing purchases efficiently.



Billing Details

First Name * — First Name

Last Name * — Last Name

Company Name — Company Name

Country * — Select a country

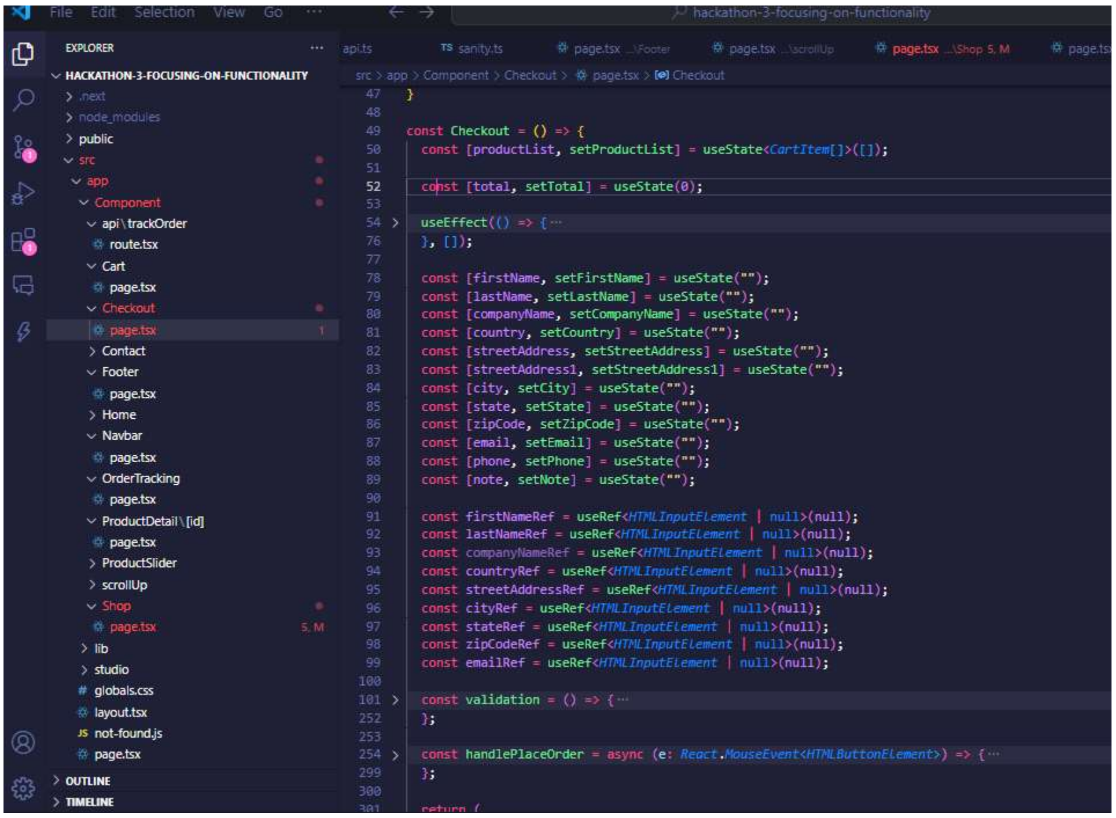
Street Address * — House number and street name / Apartment, suite, unit etc.

Town / City * — Town / City

Your order

Product	Total
The Dandy chair X 1	$150
Reflective Haven X 3	$900
Vase Set X 1	$150
Shipping	Free shipping
Total	$1200
Cash on delivery	Available

PLACE ORDER



hackathon-3-focusing-on-functionality
EXPLORER
HACKATHON-3-FOCUSING-ON-FUNCTIONALITY
src > app > Component > Checkout > page.tsx > Checkout

```
}

const Checkout = () => {
  const [productList, setProductList] = useState<CartItem[]>([]);

  const [total, setTotal] = useState(0);

  useEffect(() => {···
  }, []);

  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [companyName, setCompanyName] = useState("");
  const [country, setCountry] = useState("");
  const [streetAddress, setStreetAddress] = useState("");
  const [streetAddress1, setStreetAddress1] = useState("");
  const [city, setCity] = useState("");
  const [state, setState] = useState("");
  const [zipCode, setZipCode] = useState("");
  const [email, setEmail] = useState("");
  const [phone, setPhone] = useState("");
  const [note, setNote] = useState("");

  const firstNameRef = useRef<HTMLInputElement | null>(null);
  const lastNameRef = useRef<HTMLInputElement | null>(null);
  const companyNameRef = useRef<HTMLInputElement | null>(null);
  const countryRef = useRef<HTMLInputElement | null>(null);
  const streetAddressRef = useRef<HTMLInputElement | null>(null);
  const cityRef = useRef<HTMLInputElement | null>(null);
  const stateRef = useRef<HTMLInputElement | null>(null);
  const zipCodeRef = useRef<HTMLInputElement | null>(null);
  const emailRef = useRef<HTMLInputElement | null>(null);

  const validation = () => {···
  };

  const handlePlaceOrder = async (e: React.MouseEvent<HTMLButtonElement>) => {···
  };

  return (
```

Challenge Faced:

Dynamic Routing

1) Issue: *Implementing dynamic routes for individual product pages and user-specific data without hardcoding paths, leading to scalability issues.*

2) Solution: *Harnessed the power of Next.js dynamic routing with [id].ts, enabling scalable and flexible URLs for each product. This approach ensures seamless product detailing, offering a personalized and dynamic user experience across the website.*

Created BY: Muhammad Fahad Memon