# Day 4 - Dynamic Frontend Components of - E - Commerce Website:

## Functional Deliverables:

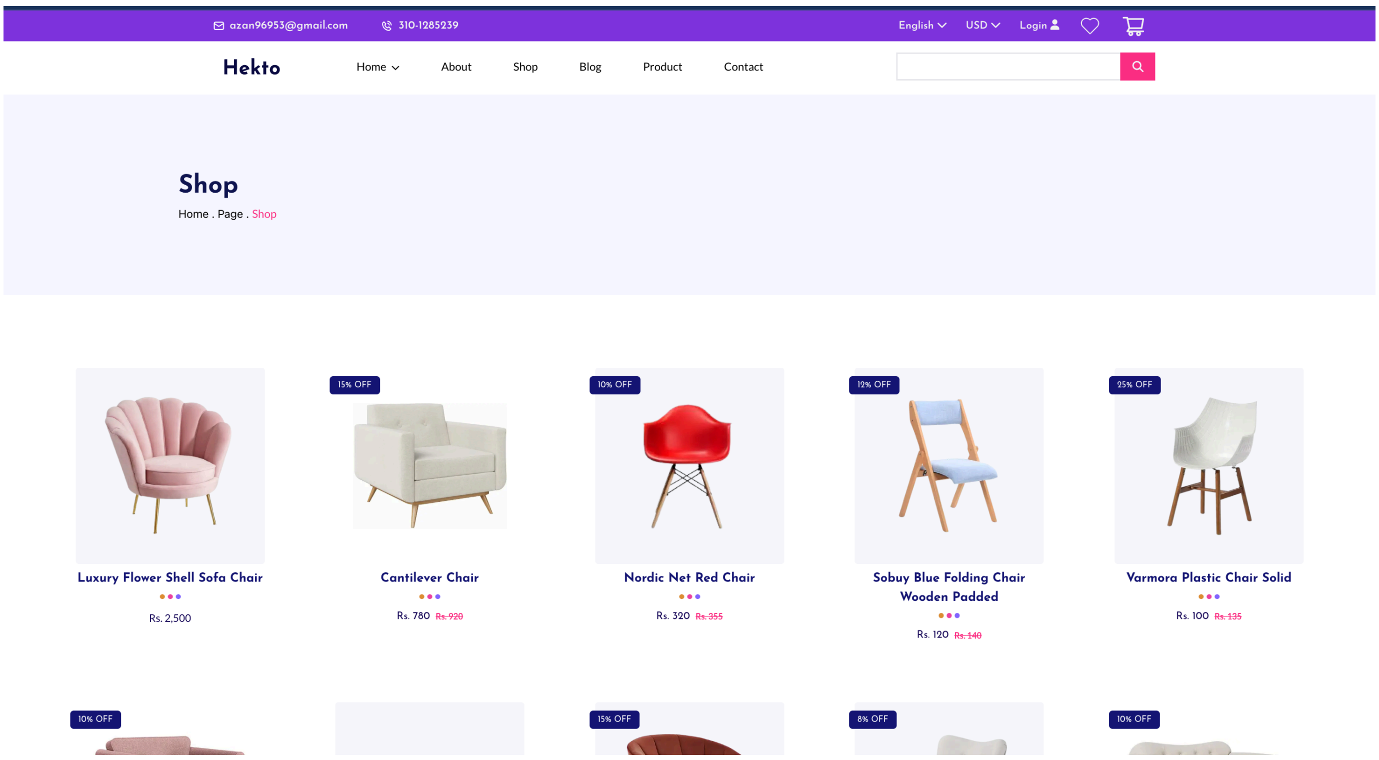
**Product Details**: Implement detailed view pages for each product, allowing users to see more information about the product, such as specifications, reviews, and images.

**Wishlist**: Create a wishlist feature that allows users to save products they are interested in purchasing later.

**Add to Cart**: Enable users to add products to their shopping cart, with options to update quantities or remove items.

**Checkout**: Implement the checkout process, including form fields for shipping, billing, and payment information.

1. **Prodcut Listing Page :**



Luxury Flower Shell Sofa Chair
Rs. 2,500

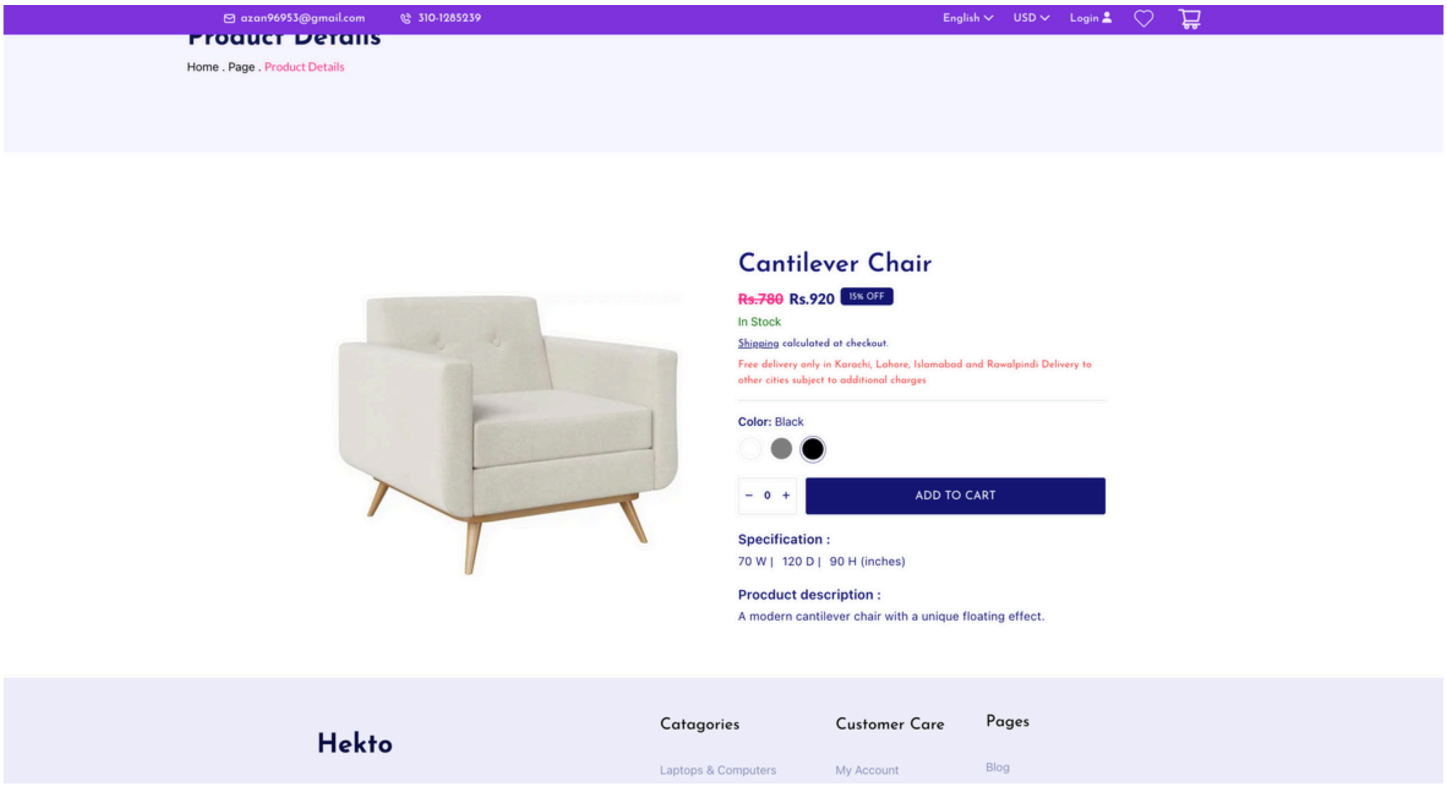
Cantilever Chair
Rs. 780 ~~Rs. 920~~

Nordic Net Red Chair
Rs. 320 ~~Rs. 355~~

Sobuy Blue Folding Chair Wooden Padded
Rs. 120 ~~Rs. 140~~

Varmora Plastic Chair Solid
Rs. 100 ~~Rs. 135~~

**2 Product details Page :**



azan96953@gmail.com 310-1285239 English USD Login

Product Details

Home . Page . Product Details

Cantilever Chair

~~Rs.780~~ Rs.920 15% OFF

In Stock

Shipping calculated at checkout.

Free delivery only in Karachi, Lahore, Islamabad and Rawalpindi Delivery to other cities subject to additional charges

Color: Black

○ ● ●

− 0 + ADD TO CART

Specification :

70 W | 120 D | 90 H (inches)

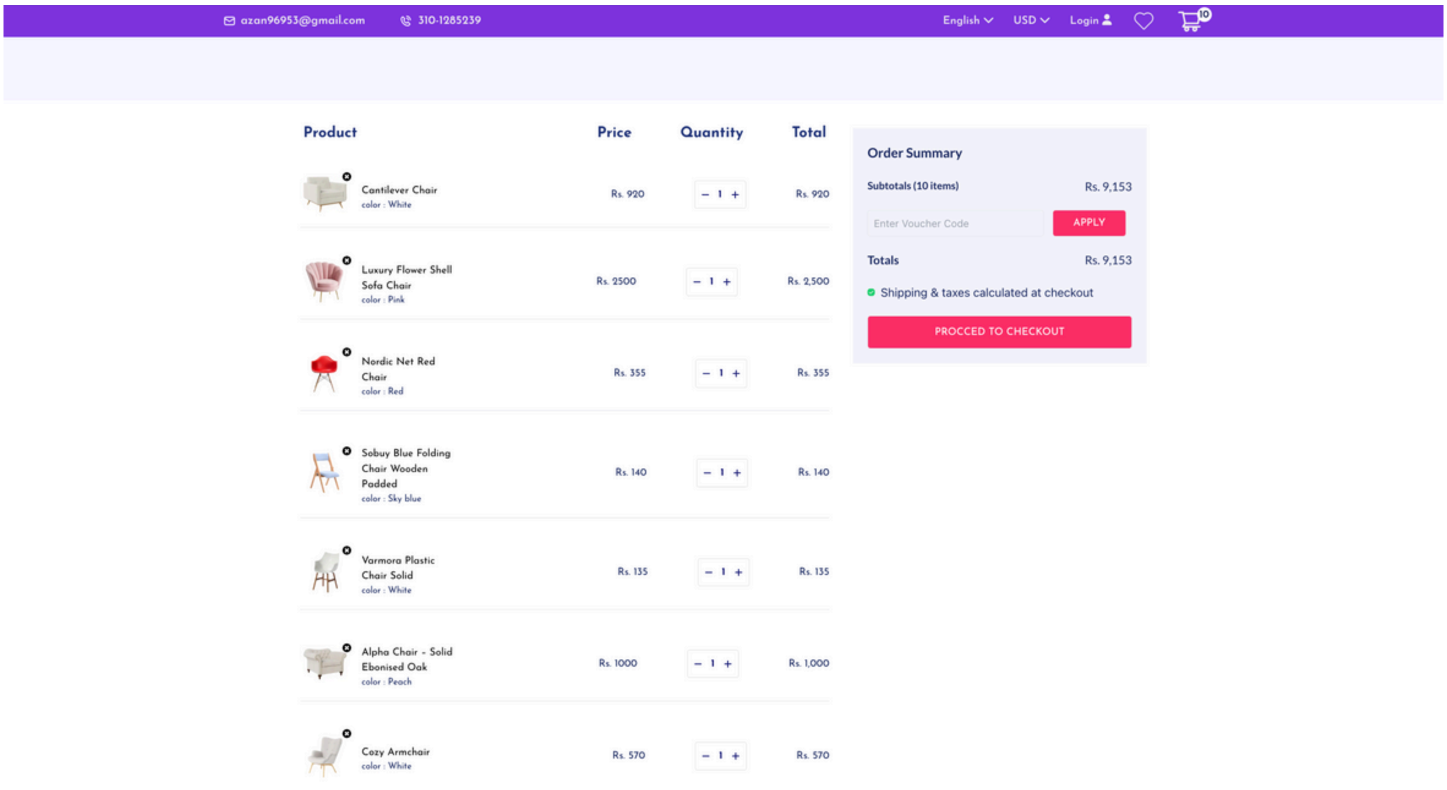
Procduct description :

A modern cantilever chair with a unique floating effect.

Hekto

| Catagories | Customer Care | Pages |
|---|---|---|
| Laptops & Computers | My Account | Blog |

## 3 Shopping Cart :



azan96953@gmail.com 310-1285239 English USD Login 10

| Product | Price | Quantity | Total |
|---|---|---|---|
| Cantilever Chair color : White | Rs. 920 | − 1 + | Rs. 920 |
| Luxury Flower Shell Sofa Chair color : Pink | Rs. 2500 | − 1 + | Rs. 2,500 |
| Nordic Net Red Chair color : Red | Rs. 355 | − 1 + | Rs. 355 |
| Sobuy Blue Folding Chair Wooden Padded color : Sky blue | Rs. 140 | − 1 + | Rs. 140 |
| Varmora Plastic Chair Solid color : White | Rs. 135 | − 1 + | Rs. 135 |
| Alpha Chair - Solid Ebonised Oak color : Peach | Rs. 1000 | − 1 + | Rs. 1,000 |
| Cozy Armchair color : White | Rs. 570 | − 1 + | Rs. 570 |

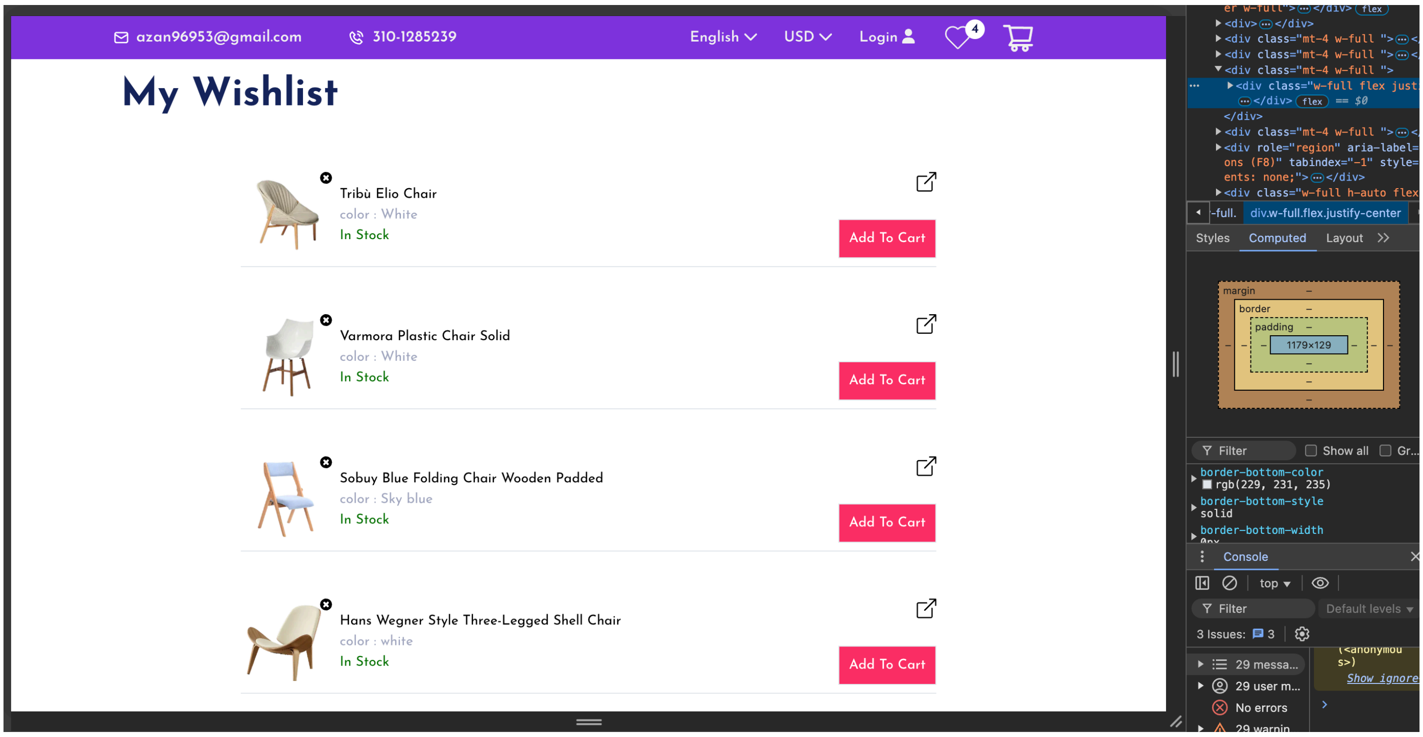
Order Summary

| | |
|---|---|
| Subtotals (10 items) | Rs. 9,153 |
| Enter Voucher Code | APPLY |
| Totals | Rs. 9,153 |

● Shipping & taxes calculated at checkout

PROCCED TO CHECKOUT

## 4 Wishlist :



azan96953@gmail.com 310-1285239 English USD Login 4

## My Wishlist

| | | |
|---|---|---|
| Tribù Elio Chair | color : White | In Stock | Add To Cart |
| Varmora Plastic Chair Solid | color : White | In Stock | Add To Cart |
| Sobuy Blue Folding Chair Wooden Padded | color : Sky blue | In Stock | Add To Cart |
| Hans Wegner Style Three-Legged Shell Chair | color : white | In Stock | Add To Cart |

```
er w-full">⋯</div> flex
▶<div>⋯</div>
▶<div class="mt-4 w-full ">⋯<
▶<div class="mt-4 w-full ">⋯<
▼<div class="mt-4 w-full ">
  ▶<div class="w-full flex just
   ⋯</div> flex == $0
</div>
▶<div class="mt-4 w-full ">⋯<
▶<div role="region" aria-label=
ons (F8)" tabindex="-1" style=
ents: none;">⋯</div>
▶<div class="w-full h-auto flex
```

-full. div.w-full.flex.justify-center

Styles Computed Layout

margin – border – padding – 1179×129

Filter Show all Gr...

```
border-bottom-color
 rgb(229, 231, 235)
border-bottom-style
solid
border-bottom-width
```

Console

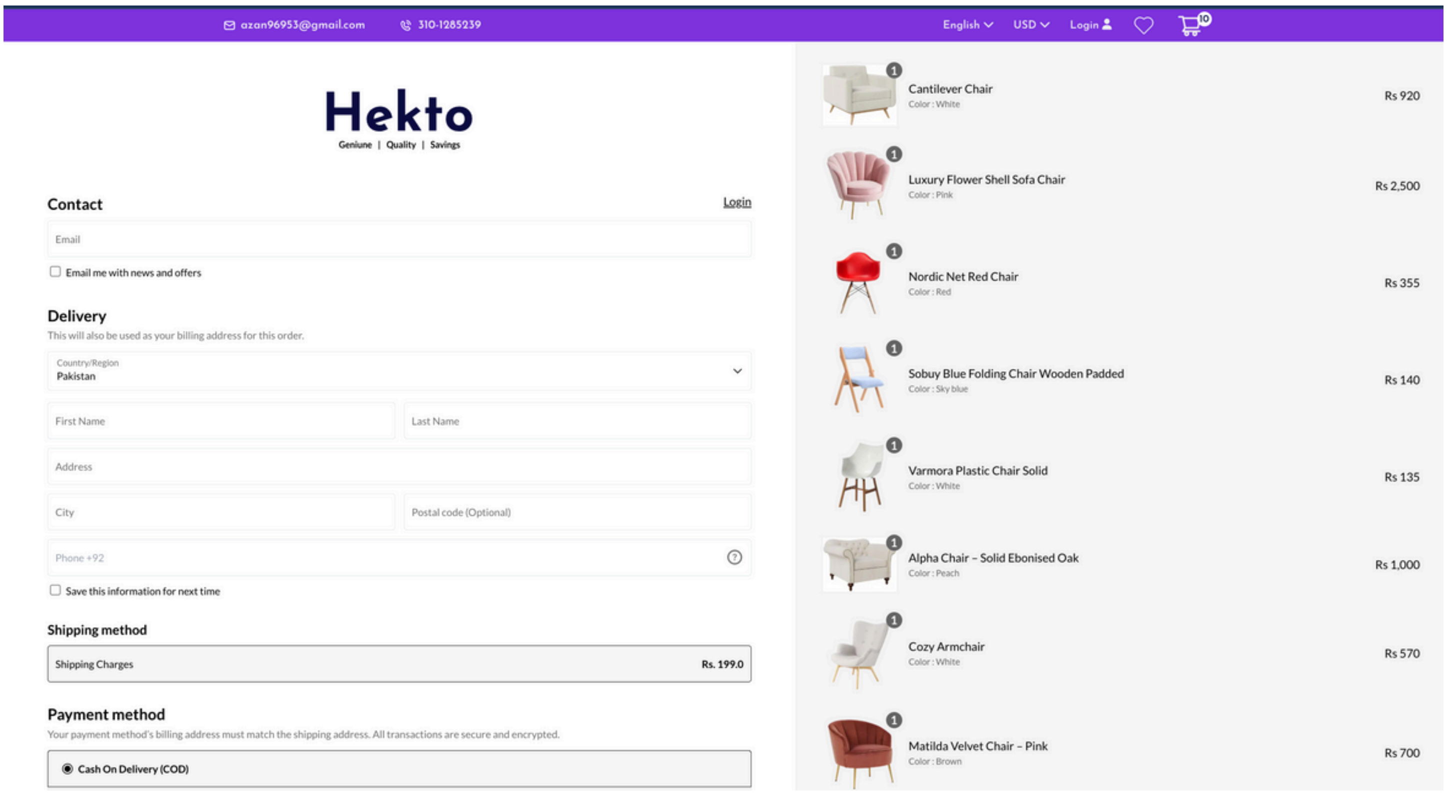
top Filter Default levels

3 Issues: 3

29 messa... 29 user m... No errors 29 warnin

```
(<anonymou
s>)
Show ignore
```

## 5 : check out



azan96953@gmail.com 310-1285239 English USD Login 10

Hekto
Geniune | Quality | Savings

**Contact** Login

Email

☐ Email me with news and offers

**Delivery**

This will also be used as your billing address for this order.

Country/Region: Pakistan

First Name | Last Name

Address

City | Postal code (Optional)

Phone +92

☐ Save this information for next time

**Shipping method**

Shipping Charges Rs. 199.0

**Payment method**

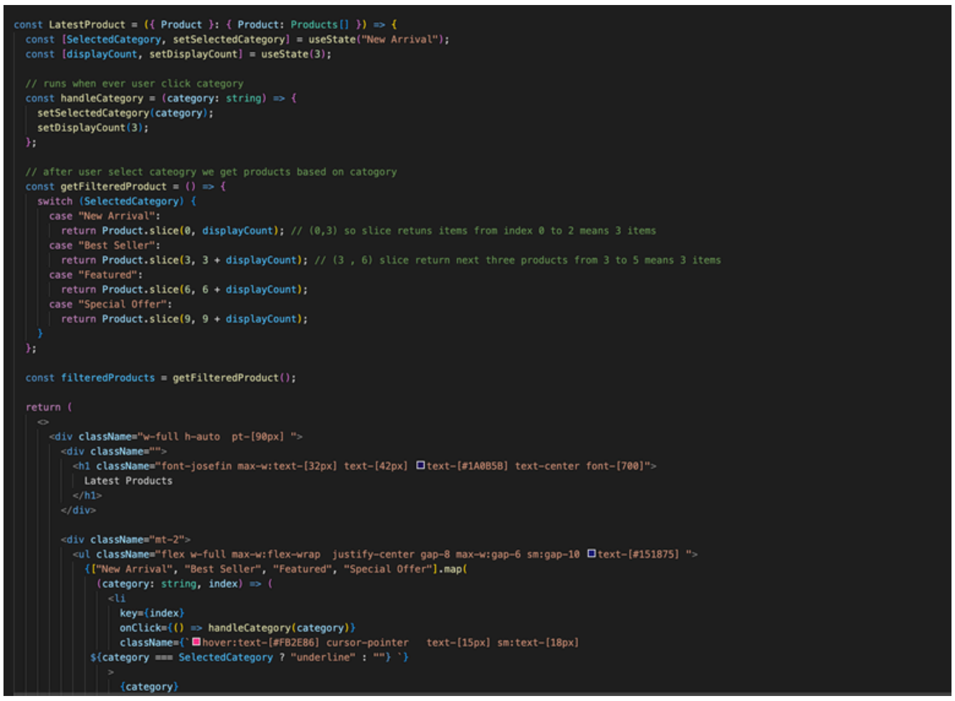
Your payment method's billing address must match the shipping address. All transactions are secure and encrypted.

● Cash On Delivery (COD)

| | | | |
|---|---|---|---|
| 1 | Cantilever Chair | Color : White | Rs 920 |
| 1 | Luxury Flower Shell Sofa Chair | Color : Pink | Rs 2,500 |
| 1 | Nordic Net Red Chair | Color : Red | Rs 355 |
| 1 | Sobuy Blue Folding Chair Wooden Padded | Color : Sky blue | Rs 140 |
| 1 | Varmora Plastic Chair Solid | Color : White | Rs 135 |
| 1 | Alpha Chair – Solid Ebonised Oak | Color : Peach | Rs 1,000 |
| 1 | Cozy Armchair | Color : White | Rs 570 |
| 1 | Matilda Velvet Chair – Pink | Color : Brown | Rs 700 |

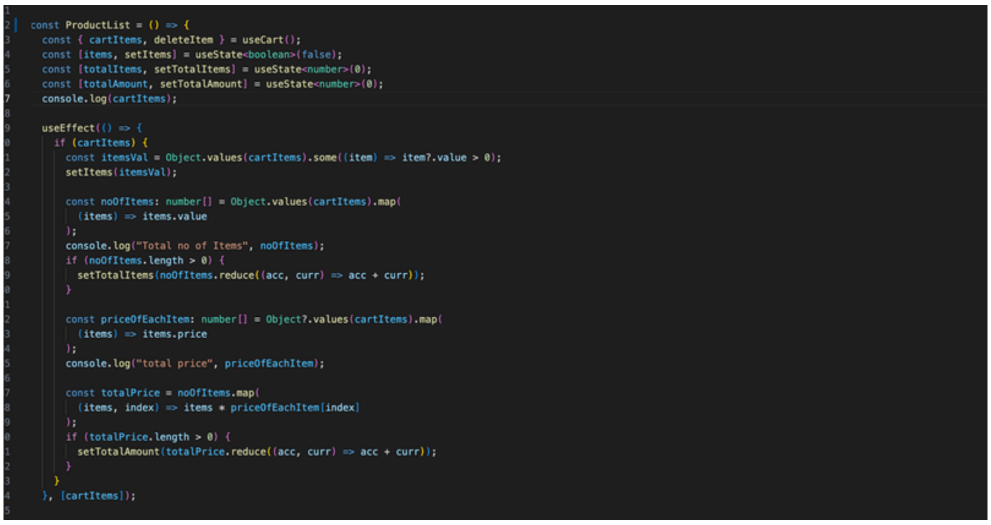
# Code Deliverables:

1. **Prodcut filteration based on category :**



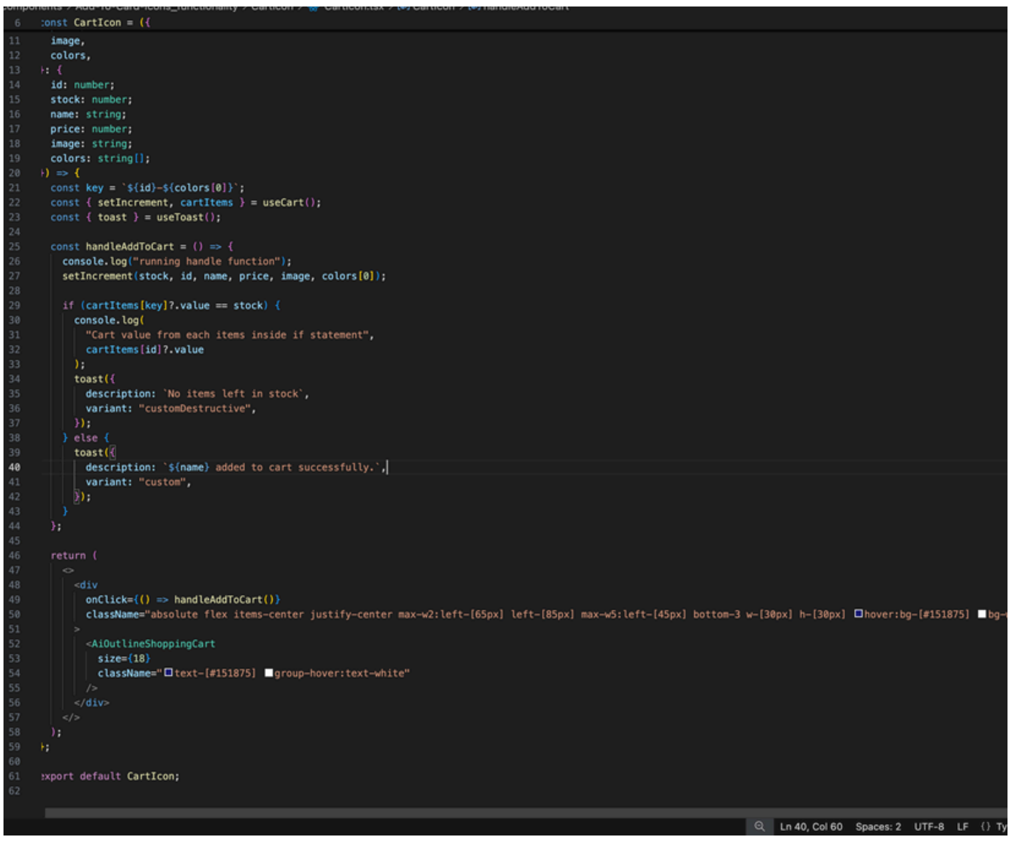
```
const LatestProduct = ({ Product }: { Product: Products[] }) => {
  const [SelectedCategory, setSelectedCategory] = useState("New Arrival");
  const [displayCount, setDisplayCount] = useState(3);

  // runs when ever user click category
  const handleCategory = (category: string) => {
    setSelectedCategory(category);
    setDisplayCount(3);
  };

  // after user select cateogry we get products based on catogory
  const getFilteredProduct = () => {
    switch (SelectedCategory) {
      case "New Arrival":
        return Product.slice(0, displayCount); // (0,3) so slice retuns items from index 0 to 2 means 3 items
      case "Best Seller":
        return Product.slice(3, 3 + displayCount); // (3 , 6) slice return next three products from 3 to 5 means 3 items
      case "Featured":
        return Product.slice(6, 6 + displayCount);
      case "Special Offer":
        return Product.slice(9, 9 + displayCount);
    }
  };

  const filteredProducts = getFilteredProduct();

  return (
    <>
      <div className="w-full h-auto  pt-[90px] ">
        <div className="">
          <h1 className="font-josefin max-w:text-[32px] text-[42px] text-[#1A0B5B] text-center font-[700]">
            Latest Products
          </h1>
        </div>

        <div className="mt-2">
          <ul className="flex w-full max-w:flex-wrap  justify-center gap-8 max-w:gap-6 sm:gap-10 text-[#151875] ">
            {["New Arrival", "Best Seller", "Featured", "Special Offer"].map(
              (category: string, index) => (
                <li
                  key={index}
                  onClick={() => handleCategory(category)}
                  className={`hover:text-[#FB2E86] cursor-pointer   text-[15px] sm:text-[18px]
              ${category === SelectedCategory ? "underline" : ""} `}
                >
                  {category}
```

## 2 Product price calculation :



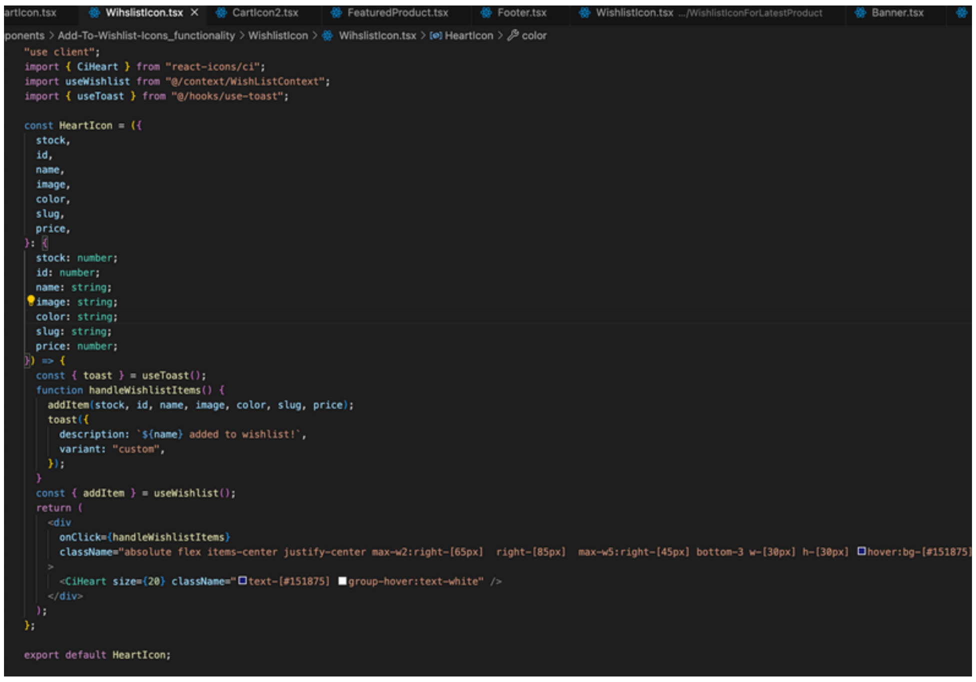
```
const ProductList = () => {
  const { cartItems, deleteItem } = useCart();
  const [items, setItems] = useState<boolean>(false);
  const [totalItems, setTotalItems] = useState<number>(0);
  const [totalAmount, setTotalAmount] = useState<number>(0);
  console.log(cartItems);

  useEffect(() => {
    if (cartItems) {
      const itemsVal = Object.values(cartItems).some((item) => item?.value > 0);
      setItems(itemsVal);

      const noOfItems: number[] = Object.values(cartItems).map(
        (items) => items.value
      );
      console.log("Total no of Items", noOfItems);
      if (noOfItems.length > 0) {
        setTotalItems(noOfItems.reduce((acc, curr) => acc + curr));
      }

      const priceOfEachItem: number[] = Object?.values(cartItems).map(
        (items) => items.price
      );
      console.log("total price", priceOfEachItem);

      const totalPrice = noOfItems.map(
        (items, index) => items * priceOfEachItem[index]
      );
      if (totalPrice.length > 0) {
        setTotalAmount(totalPrice.reduce((acc, curr) => acc + curr));
      }
    }
  }, [cartItems]);
```

## 3 Card Icon functionality



```tsx
onst CartIcon = ({
  image,
  colors,
}: {
  id: number;
  stock: number;
  name: string;
  price: number;
  image: string;
  colors: string[];
}) => {
  const key = `${id}-${colors[0]}`;
  const { setIncrement, cartItems } = useCart();
  const { toast } = useToast();

  const handleAddToCart = () => {
    console.log("running handle function");
    setIncrement(stock, id, name, price, image, colors[0]);

    if (cartItems[key]?.value == stock) {
      console.log(
        "Cart value from each items inside if statement",
        cartItems[id]?.value
      );
      toast({
        description: `No items left in stock`,
        variant: "customDestructive",
      });
    } else {
      toast({
        description: `${name} added to cart successfully.`,
        variant: "custom",
      });
    }
  };

  return (
    <>
      <div
        onClick={() => handleAddToCart()}
        className="absolute flex items-center justify-center max-w2:left-[65px] left-[85px] max-w5:left-[45px] bottom-3 w-[30px] h-[30px] hover:bg-[#151875] bg-
      >
        <AiOutlineShoppingCart
          size={18}
          className="text-[#151875] group-hover:text-white"
        />
      </div>
    </>
  );
};

xport default CartIcon;
```

### 4 Wishlist Icon functionality


artIcon.tsx
WihslistIcon.tsx
CartIcon2.tsx
FeaturedProduct.tsx
Footer.tsx
WishlistIcon.tsx .../WishlistIconForLatestProduct
Banner.tsx
ponents > Add-To-Wishlist-Icons_functionality > WishlistIcon > WihslistIcon.tsx > HeartIcon > color

```tsx
"use client";
import { CiHeart } from "react-icons/ci";
import useWishlist from "@/context/WishListContext";
import { useToast } from "@/hooks/use-toast";

const HeartIcon = ({
  stock,
  id,
  name,
  image,
  color,
  slug,
  price,
}: {
  stock: number;
  id: number;
  name: string;
  image: string;
  color: string;
  slug: string;
  price: number;
}) => {
  const { toast } = useToast();
  function handleWishlistItems() {
    addItem(stock, id, name, image, color, slug, price);
    toast({
      description: `${name} added to wishlist!`,
      variant: "custom",
    });
  }
  const { addItem } = useWishlist();
  return (
    <div
      onClick={handleWishlistItems}
      className="absolute flex items-center justify-center max-w2:right-[65px]  right-[85px]  max-w5:right-[45px] bottom-3 w-[30px] h-[30px] hover:bg-[#151875]
    >
      <CiHeart size={20} className="text-[#151875] group-hover:text-white" />
    </div>
  );
};

export default HeartIcon;
```

# Technical Documentation :

## Project Overview

**The project is a dynamic e-commerce marketplace built using Next.js and Sanity CMS. It includes features like product listing, product details, shopping cart, wishlist, and category-**

based filtering. The frontend is modular, reusable, and responsive, while the backend leverages Sanity for content management.

## 1 Steps Taken to Build and Integrate Components :

### 1. Global State Management with Context API

Purpose: To manage shared state (e.g., cart items, wishlist items) across the application.

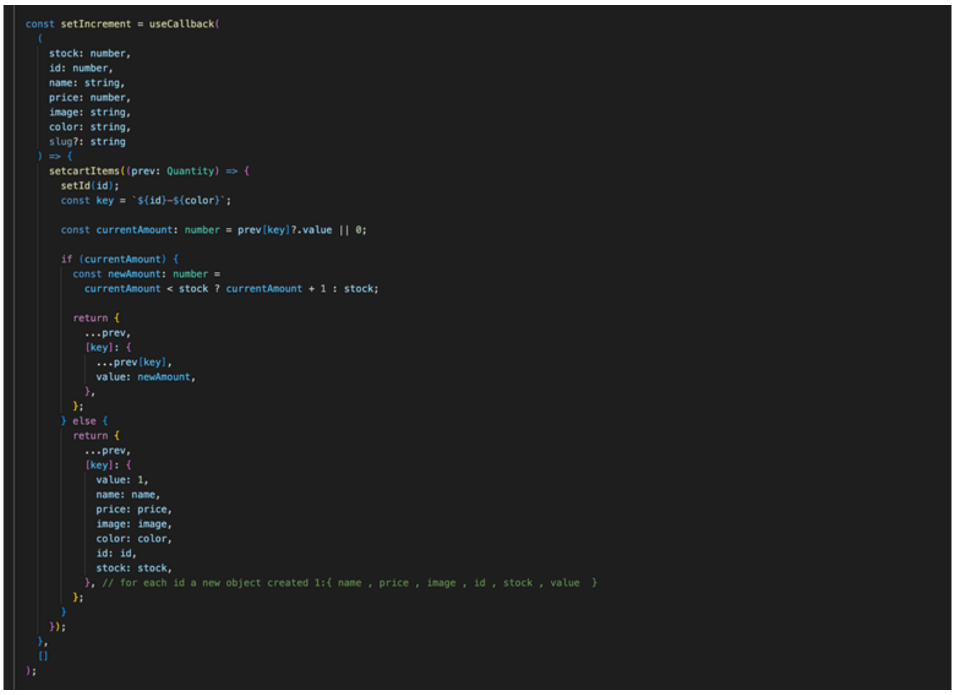
### Implementation:

Created CartContext and WishlistContext to provide global access to cart and wishlist data.

- Ensured that each product with multiple colors is uniquely identified using a combination of productId and color.

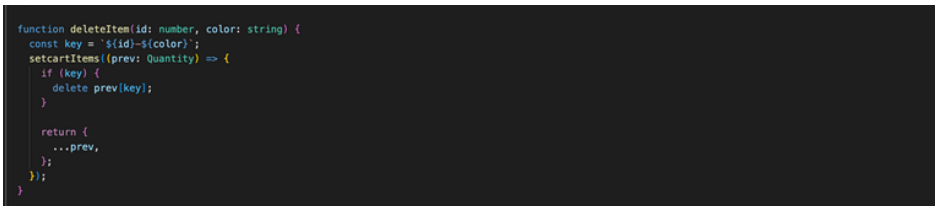
#### In case of Card

#### 1 Function to add Products



```
const setIncrement = useCallback(
  (
    stock: number,
    id: number,
    name: string,
    price: number,
    image: string,
    color: string,
    slug?: string
  ) => {
    setcartItems((prev: Quantity) => {
      setId(id);
      const key = `${id}-${color}`;

      const currentAmount: number = prev[key]?.value || 0;

      if (currentAmount) {
        const newAmount: number =
          currentAmount < stock ? currentAmount + 1 : stock;

        return {
          ...prev,
          [key]: {
            ...prev[key],
            value: newAmount,
          },
        };
      } else {
        return {
          ...prev,
          [key]: {
            value: 1,
            name: name,
            price: price,
            image: image,
            color: color,
            id: id,
            stock: stock,
          }, // for each id a new object created 1:{ name , price , image , id , stock , value  }
        };
      }
    });
  },
  []
);
```

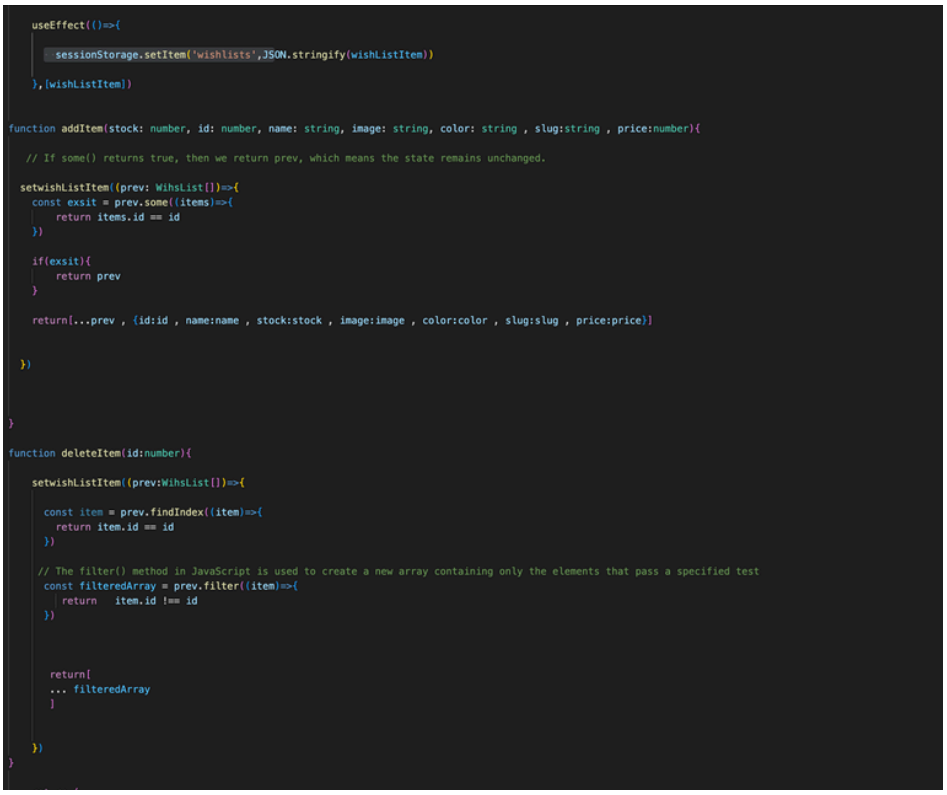
#### 2 Function to delete Products



```
function deleteItem(id: number, color: string) {
  const key = `${id}-${color}`;
  setcartItems((prev: Quantity) => {
    if (key) {
      delete prev[key];
    }

    return {
      ...prev,
    };
  });
}
```

#### In case of wishlist

1. **Function to add and delete the Items**



```
    useEffect(()=>{

        sessionStorage.setItem('wishlists',JSON.stringify(wishListItem))

    },[wishListItem])


function addItem(stock: number, id: number, name: string, image: string, color: string , slug:string , price:number){

   // If some() returns true, then we return prev, which means the state remains unchanged.

  setwishListItem((prev: WihsList[])=>{
    const exsit = prev.some((items)=>{
        return items.id == id
    })

    if(exsit){
        return prev
    }

    return[...prev , {id:id , name:name , stock:stock , image:image , color:color , slug:slug , price:price}]


  })



}

function deleteItem(id:number){

    setwishListItem((prev:WihsList[])=>{

      const item = prev.findIndex((item)=>{
        return item.id == id
      })

     // The filter() method in JavaScript is used to create a new array containing only the elements that pass a specified test
      const filteredArray = prev.filter((item)=>{
        return   item.id !== id
      })



       return[
       ... filteredArray
       ]


    })
}
```

## 2 Modular Component Design :

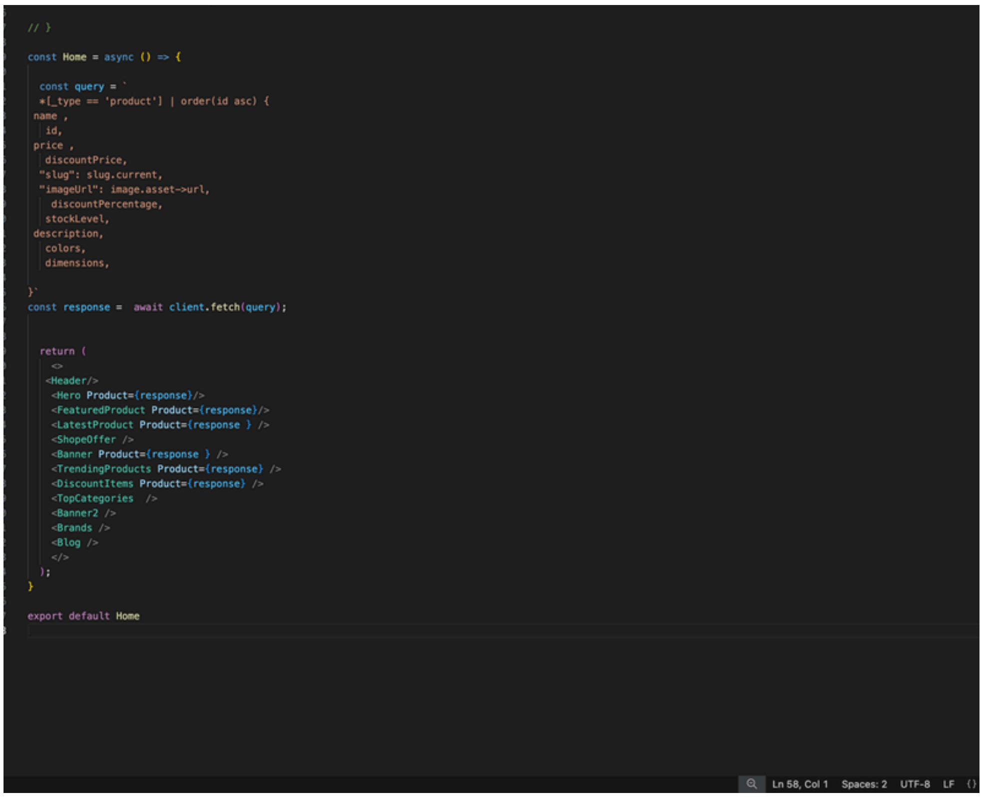
**Purpose: To create reusable and maintainable components.**

**Implement :**

**A Next.js Image for optimizedlike ProductCard, ProductList, CartIcon, and WishlistIcon to be modular and reusable.**

**Used props to pass data and conditional rendering to handle dynamic behavior (e.g., discount percentage, stock level).**

**Integrated Next.js Image for optimized image loading**



```
// }

const Home = async () => {

  const query = `
  *[_type == 'product'] | order(id asc) {
 name ,
   id,
 price ,
   discountPrice,
  "slug": slug.current,
  "imageUrl": image.asset->url,
    discountPercentage,
   stockLevel,
 description,
   colors,
   dimensions,

}`
const response =  await client.fetch(query);


  return (
    <>
   <Header/>
    <Hero Product={response}/>
    <FeaturedProduct Product={response}/>
    <LatestProduct Product={response } />
    <ShopeOffer />
    <Banner Product={response } />
    <TrendingProducts Product={response} />
    <DiscountItems Product={response} />
    <TopCategories  />
    <Banner2 />
    <Brands />
    <Blog />
    </>
  );
}

export default Home
```

Ln 58, Col 1   Spaces: 2   UTF-8   LF

### 3. Dynamic Routing for Product Details

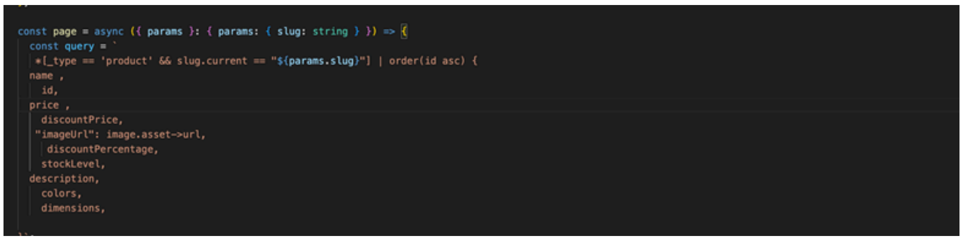
**Purpose: To display individual product details based on the product slug.**

**Implementation:**

**Used Next.js dynamic routing (pages/productsDetails/[slug].tsx) to fetch and render product details.**

**Queried Sanity CMS using GROQ to fetch product data based on the slug.**

**javascript**



```
const page = async ({ params }: { params: { slug: string } }) => {
  const query = `
   *[_type == 'product' && slug.current == "${params.slug}"] | order(id asc) {
  name ,
    id,
  price ,
    discountPrice,
   "imageUrl": image.asset->url,
     discountPercentage,
    stockLevel,
  description,
    colors,
    dimensions,
```

### 4.Responsive Design

**Purpose: To ensure the application works seamlessly across devices.**

**Implementation:**

**Used CSS Flexbox and Grid for layout structuring.**

**Applied media queries to adjust styles for different screen sizes.**

**Ensured images and text are responsive using relative units (e.g., %, rem).**

## 2 Challenges Faced and Solutions Implemented :

### 1. Challenge: Handling Multiple Colors for a Single Product

**Problem: When a user added the same product in different colors, the cart would overwrite the existing item.**

**Solution: Created a unique key for each cart item using productId-color (e.g., 1-pink, 1-white).Updated the cart state to store items based on this unique key.**

```
const uniqueKey = `${productId}-${color}`;
```

### 2. Challenge: Responsiveness Issues

**Problem: Some components did not render correctly on smaller screens.**

**Solution: Used CSS media queries and flexible layouts to ensure responsiveness.Tested the application on multiple devices and screen sizes**

### 3. Challenge: Fetching and Using Data in Context API

**Problem: Difficulty in fetching and managing data in the Context API.**

**Solution: Recalled TypeScript concepts to properly type the context state and actions. Used arrays and objects to structure the data and ensure easy access.**

## Best Practices Followed :

**Modular Component Design:**

**Created small, reusable components to improve code maintainability and reusability.**

**TypeScript for Type Safety:**

**Used TypeScript to define types for props, state, and API responses, reducing runtime errors.**

**Optimized Image Loading:**

**Used Next.js Image for optimized and responsive image loading.**

**Global State Management:**

**Leveraged Context API for global state management, avoiding prop drilling.**

**Responsive Design:**

**Ensured the application is fully responsive using modern CSS techniques.**

**Dynamic Routing:**

**Used Next.js dynamic routing for SEO-friendly and user-friendly URLs.**

## Repository Submission:

Muhammad-Azan-1/**7-Days-Hackhtone**

1 Contributor

0 Issues

0 Stars

0 Forks