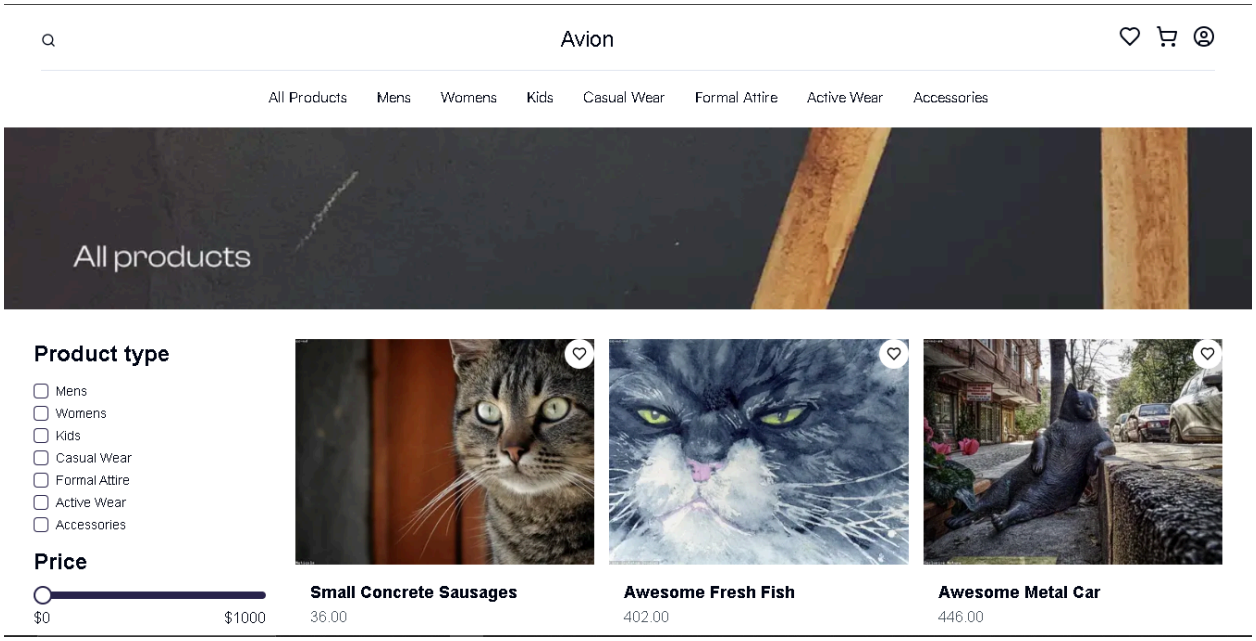
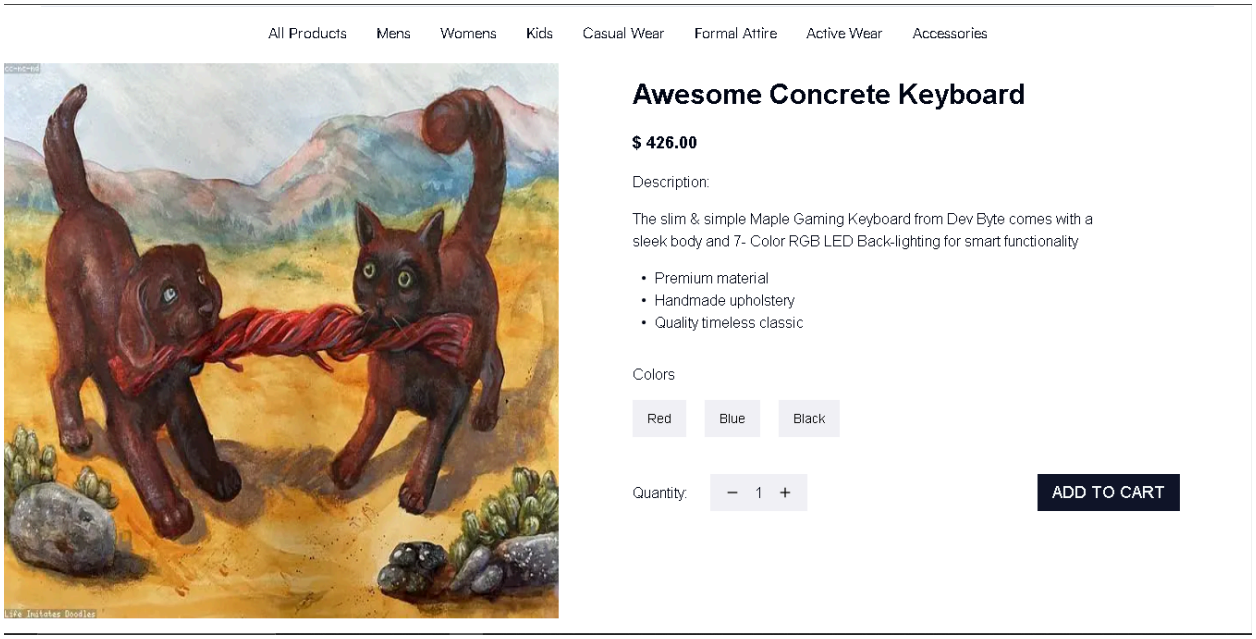
Day 4 - Dynamic Frontend Components - Avion

1. Functional Deliverables:

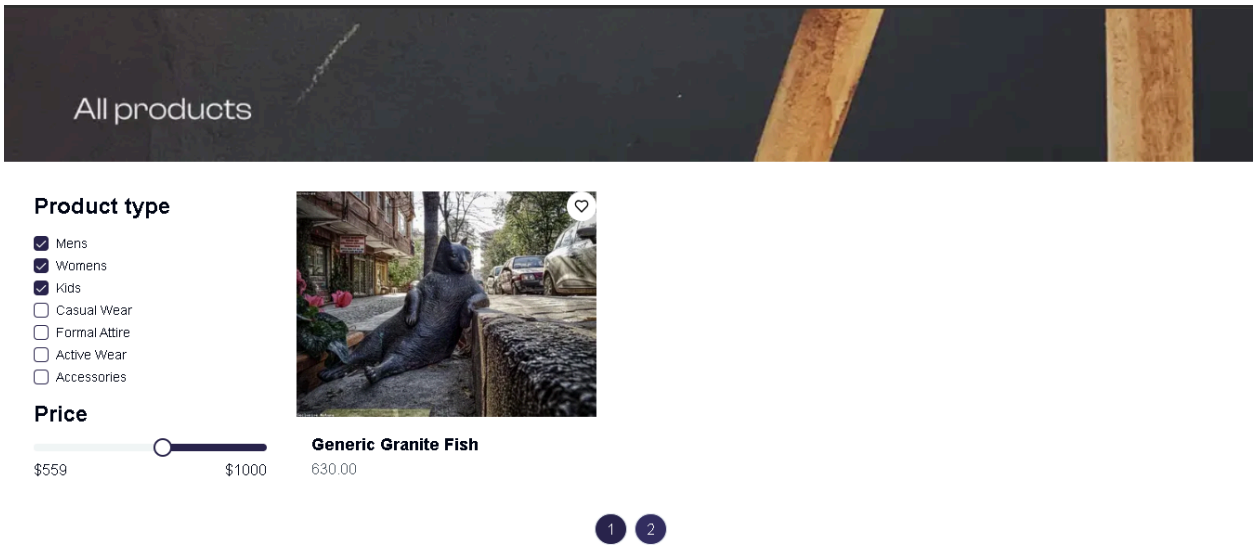
ScreenShot Of Products Listing:



ScreenShot Of Product Details:



ScreenShot Of Products Filtering And Pagination:



Mobile Categories Filtering:

Categories

Filter

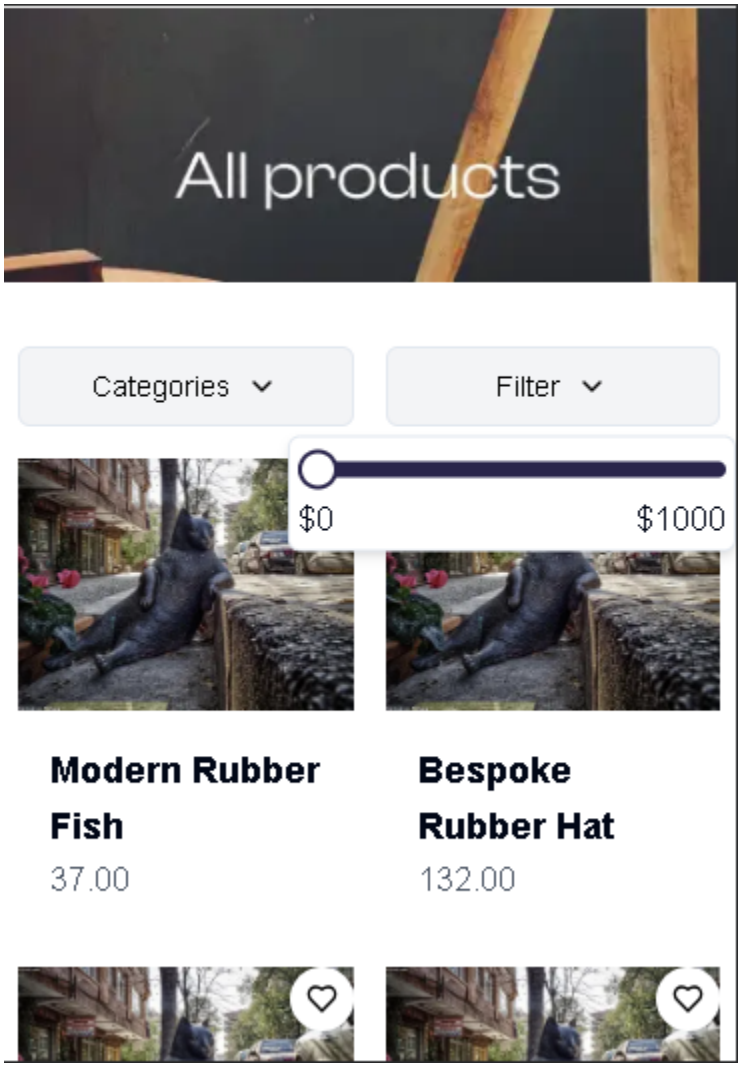
- Mens
- Womens
- Kids
- Casual Wear
- Formal Attire
- Active Wear
- Accessories

espoke

ubber Hat

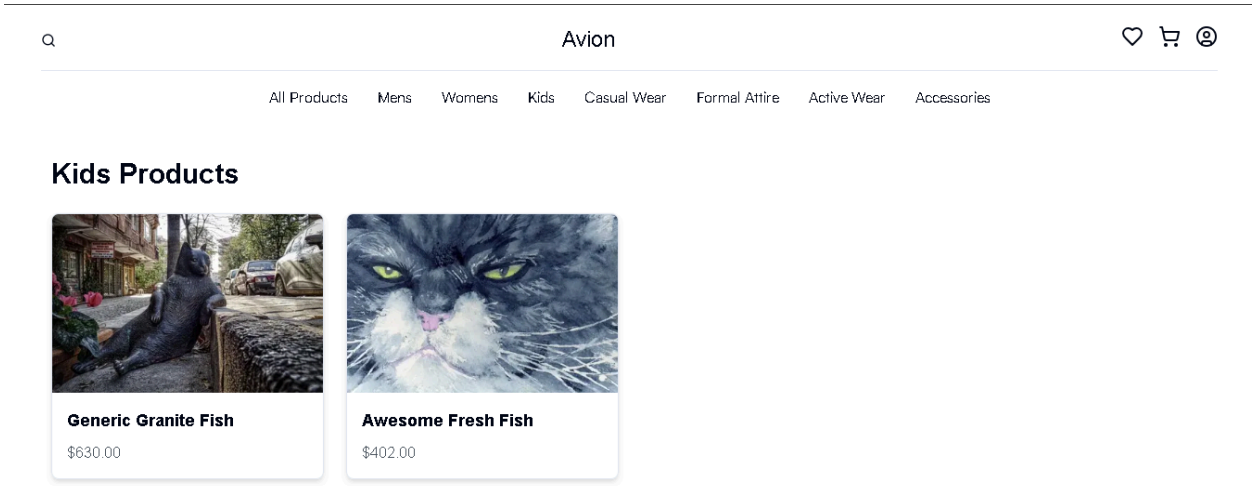
2.00

Mobile Price Filtering:

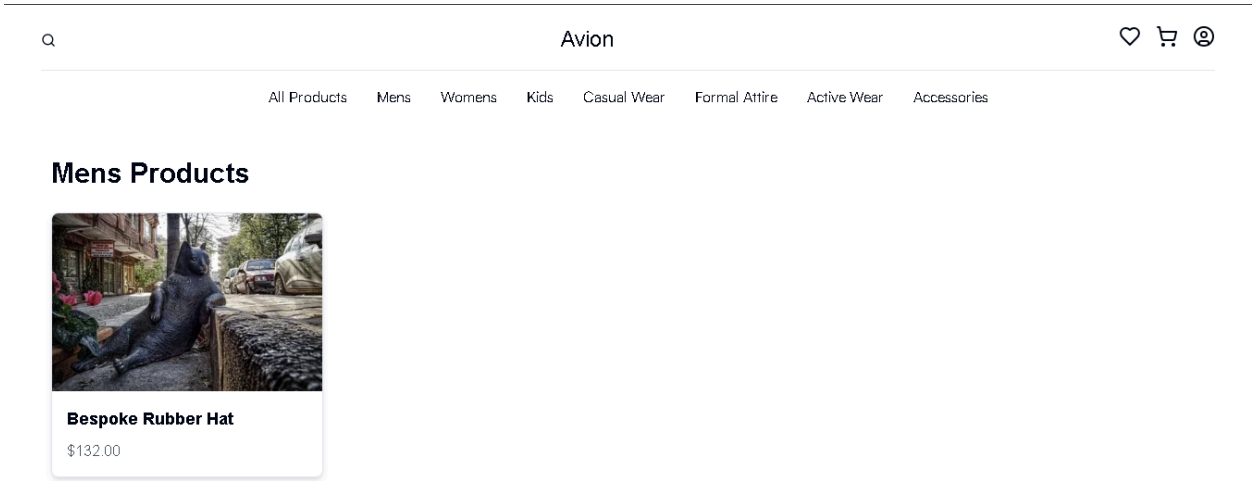


ScreenShots Of Product Categories:

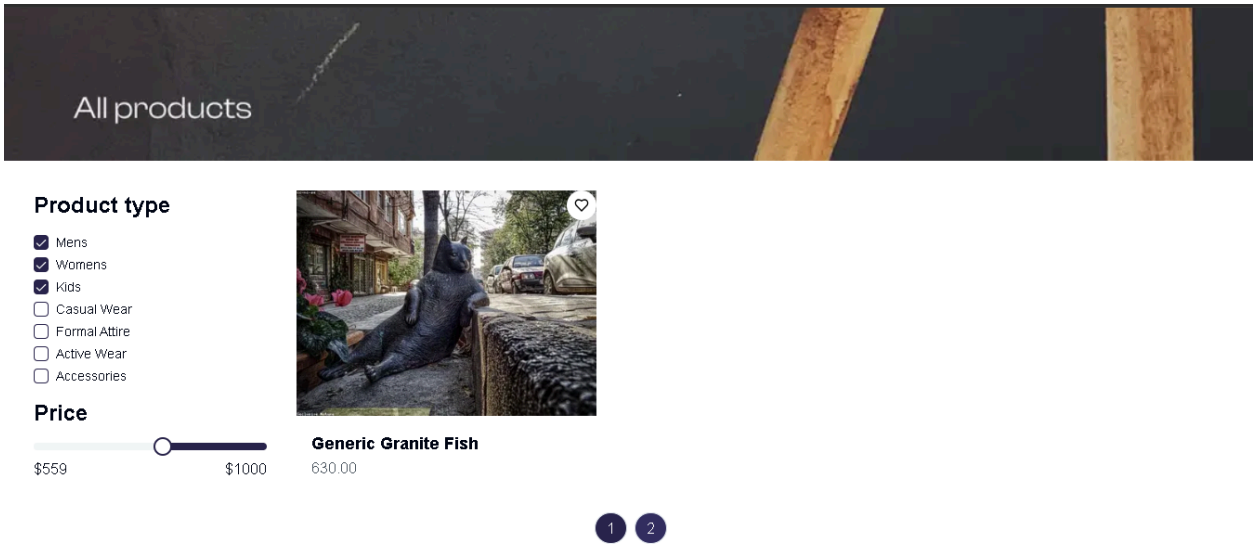
1. Kid's Category:



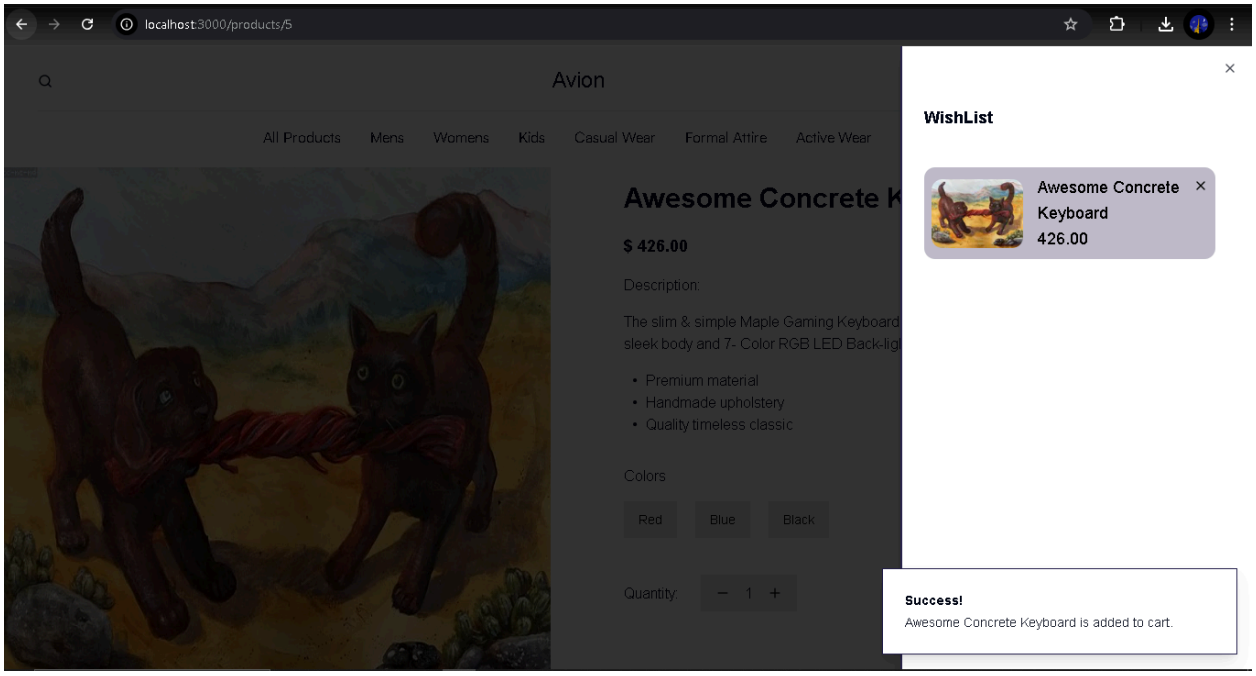
2. Men's Category:



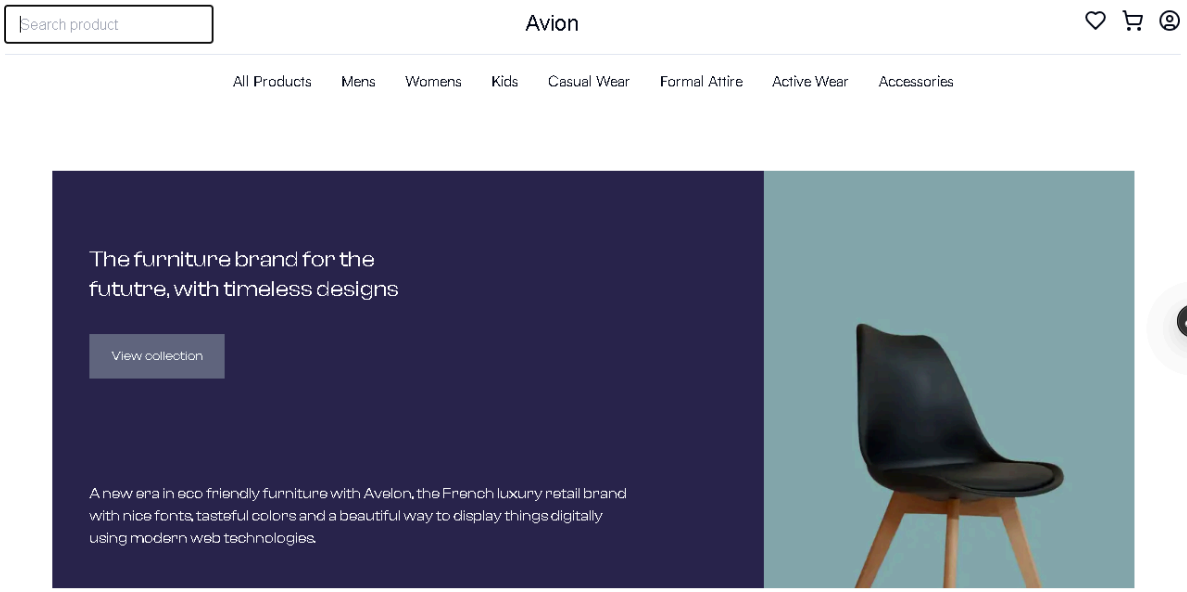
ScreenShot Of Working category filters and pagination:



ScreenShot Of working Wishlist with Toast Notification:



ScreenShot of Working search bar:



2. Code Deliverables:

Code Snippet of Products Listing:

1. 
components > ProductListing.tsx > Product > sizes

```tsx
"use client";

import Image from "next/image";
import Link from "next/link";
import React, { useEffect, useState } from "react";
import { Checkbox } from "./ui/checkbox";
import { Label } from "./ui/label";
import { Button } from "./ui/button";
import { ChevronDown, Heart } from "lucide-react";
import { urlFor } from "@/sanity/lib/image";
import { Image as SanityImage } from "@sanity/types";
import { Slider } from "./ui/slider";
import {
  DropdownMenu,
  DropdownMenuCheckboxItem,
  DropdownMenuContent,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import { useCart } from "@/lib/CartContext";
import { toast } from "@/hooks/use-toast";

interface Product {
  id: string;
  name: string;
  quantity: number;
  price: number;
  images: SanityImage;
  ratings: string;
  sizes: string[];
  colors: string[];
  tags: string[];
  categories: string[];
  description: string;
}
```

2.

```
35
36  const productTypes = [
37    "Mens",
38    "Womens",
39    "Kids",
40    "Casual Wear",
41    "Formal Attire",
42    "Active Wear",
43    "Accessories",
44  ];
45
46  const ProductListing = () => {
47    const [products, setProducts] = useState<Product[]>([]);
48    const [currentPage, setCurrentPage] = useState<number>(1);
49    const [selectedCategories, setSelectedCategories] = useState<string[]>([]);
50    const [sliderValue, setSliderValue] = useState([0, 1000]);
51    const productsPerPage = 6;
52    const { addToWishList } = useCart();
53
54    useEffect(() => {
55      fetch("/api/products")
56        .then((res) => res.json())
57        .then((data) => setProducts(data.data))
58        .catch((error) => {
59          console.error("Error fetching featured products:", error);
60        });
61    }, []);
62
```

3.

```
63
64  const handleAddItemToWishList = (product: Product) => {
65    if (!product?.id) return;
66    addToWishList({
67      id: product.id.toString(),
68      image: product.images?.[0] as SanityImage,
69      name: product.name,
70      price: product.price,
71      description: product.description,
72      quantity: 1,
73      color: product.colors?.[0] || "",
74      size: product.sizes?.[0] || "",
75    });
76    toast({
77      className: "rounded-none border border-[#27224b]",
78      title: "Success!",
79      description: `${product.name}  is added to cart.`,
80      duration: 5000,
81    });
82  };
83
84  const handleCategoryChange = (category: string, checked: boolean) => {
85    if (checked) {
86      setSelectedCategories((prev) => [...prev, category]);
87    } else {
88      setSelectedCategories((prev) => prev.filter((cat) => cat !== category));
89    }
90  };
91
```

4.

components > ProductListing.tsx > Product > sizes

```
46  const ProductListing = () => {
92    const filteredProducts = selectedCategories.length
93      ? products.filter((product) =>
94          product.categories.some((category) =>
95            selectedCategories.includes(category)
96          )
97        )
98      : products;
99
100   const priceFilteredProducts = filteredProducts.filter(
101     (product) =>
102       product.price >= sliderValue[0] && product.price <= sliderValue[1]
103   );
104
105   // Calculate indices for slicing products
106   const indexOfLastProduct = currentPage * productsPerPage;
107   const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
108
109   const currentFilteredProducts = priceFilteredProducts.slice(
110     indexOfFirstProduct,
111     indexOfLastProduct
112   );
113
114   const paginate = (pageNumber: number) => setCurrentPage(pageNumber);
115
116   // Trigger POSTING.
117 >  // useEffect(() => {···
137   // })
138
```

5.

```
return (
  <div className="w-full flex flex-col items-center justify-center pb-10">
    <Image
      src="/Images/Page Headers.png"
      alt="Page Header Image"
      width={1000}
      height={100}
      className="w-full hidden md:flex"
    />
    <Image
      src="/Images/Page Headers (1).png"
      alt="Page Header Image"
      width={1000}
      height={100}
      className="w-full md:hidden"
    />
    <div className="flex w-full px-2 md:px-6 lg:px-8 mt-8 gap-8">
      <div className="md:flex flex-col w-1/4 hidden">
        <h2 className="text-2xl font-semibold mb-4">Product type</h2>
        <div className="space-y-2">
          {productTypes.map((item) => (
            <div key={item} className="flex items-center space-x-2">
              <Checkbox
                id={item}
                onCheckedChange={(checked) =>
                  handleCategoryChange(item, checked as boolean)
                }
              />
              <Label
                htmlFor={item}
                className="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70"
              >
                {item}
```

6.

```
                </Label>
              </div>
            ))}
          </div>
          <h2 className="text-2xl font-semibold my-4">Price</h2>
          <div className="space-y-2">
            <Slider
              value={sliderValue}
              onValueChange={(value) => setSliderValue(value)} // Update slider value
              min={0} // Min price
              max={1000} // Max price
              step={1} // Step for slider movement
            />
          </div>
          <div className="flex justify-between mt-2">
            <span>${sliderValue[0]}</span>
            <span>${sliderValue[1]}</span>
          </div>
        </div>
        <div className="grid grid-cols-2 md:grid-cols-2 lg:grid-cols-3 gap-4 w-full">
          <DropdownMenu>
            <DropdownMenuTrigger asChild>
              <Button
                variant="outline"
                className="md:hidden ■bg-gray-100 ■hover:bg-gray-200 □text-black"
              >
                Categories <ChevronDown />
              </Button>
            </DropdownMenuTrigger>
            <DropdownMenuContent className="w-56">
              {productTypes.map((category) => (
```

7.

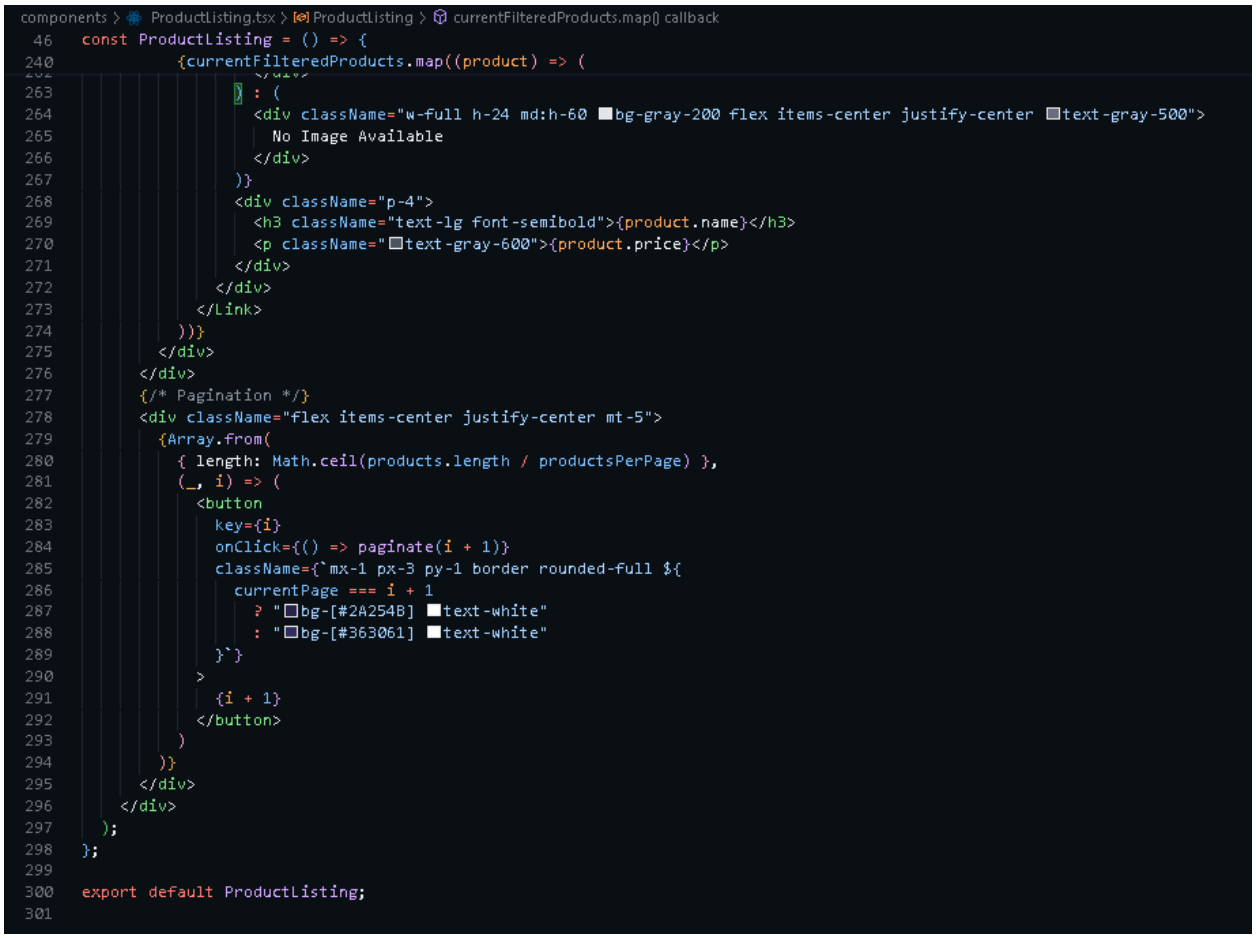
components > ProductListing.tsx > Product > sizes

```
46   const ProductListing = () => {
201                 <DropdownMenuContent className="w-56">
202                   {productTypes.map((category) => (
203                     <DropdownMenuCheckboxItem
204                       key={category}
205                       checked={selectedCategories.includes(category)}
206                       onCheckedChange={(checked) => {
207                         handleCategoryChange(category, checked);
208                       }}
209                     >
210                       {category}
211                     </DropdownMenuCheckboxItem>
212                   ))}
213                 </DropdownMenuContent>
214               </DropdownMenu>
215               <DropdownMenu>
216                 <DropdownMenuTrigger asChild>
217                   <Button
218                     variant="outline"
219                     className="md:hidden ■bg-gray-100 ■hover:bg-gray-200 □text-black"
220                   >
221                     Filter <ChevronDown />
222                   </Button>
223                 </DropdownMenuTrigger>
224                 <DropdownMenuContent className="w-56">
225                   <div className="space-y-2 mt-2">
226                     <Slider
227                       value={sliderValue}
228                       onValueChange={(value) => setSliderValue(value)} // Update slider value
229                       min={0} // Min price
230                       max={1000} // Max price
231                       step={1} // Step for slider movement
232                     />
233                   </div>
```

8.

```
components > ProductListing.tsx > ProductListing > currentFilteredProducts.map() callback
46  const ProductListing = () => {
233                 </div>
234                 <div className="flex justify-between mt-2">
235                   <span>${sliderValue[0]}</span>
236                   <span>${sliderValue[1]}</span>
237                 </div>
238               </DropdownMenuContent>
239             </DropdownMenu>
240             {currentFilteredProducts.map((product) => (
241               <Link
242                 href={`/products/${product.id}`}
243                 key={product.id}
244                 className="block"
245               >
246                 <div className="overflow-hidden">
247                   {/* Check if image exists before rendering */}
248                   {product.images?.[0] ? (
249                     <div className="relative">
250                       <Image
251                         src={urlFor(product.images[0]).url()}
252                         alt={product.name}
253                         width={300}
254                         height={300}
255                         className="w-full h-auto object-cover"
256                       />
257                       <Button
258                       className="absolute top-0 right-0 bg-white hover:bg-white/90 active:scale-95 transition-transform transform
                        duration-300 p-2 rounded-full w-fit h-fit"
259                       onClick={() => handleAddItemToWishList(product)}>
260                         <Heart className="text-black" />
261                       </Button>
262                     </div>
263                   ) : (
264                     <div className="w-full h-24 md:h-60 bg-gray-200 flex items-center justify-center text-gray-500">
```

9.



```
components > ProductListing.tsx > ProductListing > currentFilteredProducts.map() callback
46  const ProductListing = () => {
240             {currentFilteredProducts.map((product) => (
263                   ) : (
264                     <div className="w-full h-24 md:h-60 bg-gray-200 flex items-center justify-center text-gray-500">
265                       No Image Available
266                     </div>
267                   )}
268                   <div className="p-4">
269                     <h3 className="text-lg font-semibold">{product.name}</h3>
270                     <p className="text-gray-600">{product.price}</p>
271                   </div>
272                 </div>
273               </Link>
274             ))}
275           </div>
276         </div>
277         {/* Pagination */}
278         <div className="flex items-center justify-center mt-5">
279           {Array.from(
280             { length: Math.ceil(products.length / productsPerPage) },
281             (_, i) => (
282               <button
283                 key={i}
284                 onClick={() => paginate(i + 1)}
285                 className={`mx-1 px-3 py-1 border rounded-full ${
286                   currentPage === i + 1
287                     ? "bg-[#2A254B] text-white"
288                     : "bg-[#363061] text-white"
289                 }`}
290               >
291                 {i + 1}
292               </button>
293             )
294           )}
295         </div>
296       </div>
297     );
298   };
299
300   export default ProductListing;
301
```

Code Snippet of Product Details:

1.

app > products > [id] > page.tsx > ...

```tsx
"use client";

import { useParams } from "next/navigation";
import Image from "next/image";
import { Button } from "@/components/ui/button";
import { Dot } from "lucide-react";
import WhatMakesUsDiff from "@/components/WhatMakesUsDiff";
import JoinClub from "@/components/JoinClub";
import Link from "next/link";
import { useState, useEffect } from "react";
import { urlFor } from "@/sanity/lib/image";
import { Image as SanityImage } from "@sanity/types";
import { client } from "@/sanity/lib/client";
import { useCart } from "@/lib/CartContext";
import { toast } from "@/hooks/use-toast";
import { Plus, Minus } from "lucide-react";

interface Variation {
  color: string; // Color of the product (e.g., "Red", "Blue")
  size: string; // Size of the product (e.g., "S", "M", "L")
  quantity: number; // Available quantity for the specific color and size
}

interface Product {
  id: number; // Unique identifier for the product
  name: string; // Name of the product
  price: number; // Price of the product
  images: SanityImage[]; // Array of images for the product
  ratings: string; // Rating of the product (e.g., "4.5")
  tags: string[]; // Tags associated with the product (e.g., ["Best Seller"])
  description: string; // Detailed description of the product
  variations: Variation[]; // Variations of the product with different colors, sizes, and quantities
}

const ProductDetails = () => {
  const { addToCart } = useCart();
  const [products, setProducts] = useState<Product[]>([]);
  const [featuredProducts, setFeaturedProducts] = useState<Product[]>([])
  const [selectedColor, setSelectedColor] = useState<string | null>(null);
  const [selectedSize, setSelectedSize] = useState<string | null>(null);
  const [quantity, setQuantity] = useState<number>(1);

```

2.


app > products > [id] > page.tsx > ...
const ProductDetails = () => {
const params = useParams();
const productId = params.id as string;
const product = products.find((p) => String(p.id) === productId);
useEffect(() => {
const fetchProducts = async () => {
const query = `*[_type == "product" && id == $productId]{
id,
name,
price,
"images": images[].asset->{
_id,
_key,
_ref,
url,
metadata
},
ratings,
discountPercentage,
priceWithoutDiscount,
ratingCount,
description,
variations[] {
color,
size,
quantity
},
tags
}`;
try {
const fetchedProducts = await client.fetch(query, {
productId: productId,
});
if (fetchedProducts.length > 0) {
console.log("Fetched product images:", fetchedProducts[0].images);
}

3.

app > products > [id] > page.tsx > ...

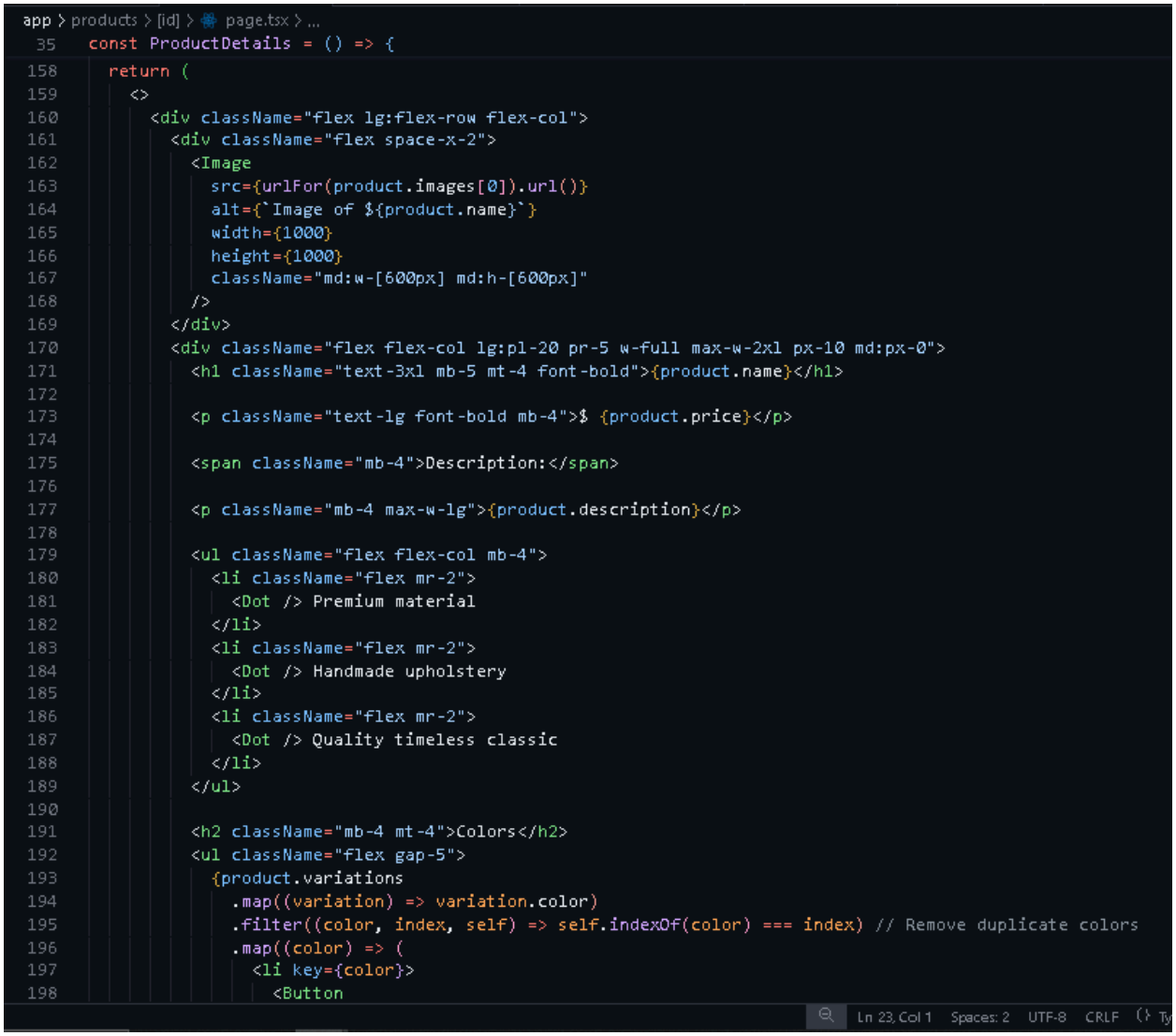
```
35  const ProductDetails = () => {
47    useEffect(() => {
48      const fetchProducts = async () => {
82          setProducts(fetchedProducts);
83        } catch (error) {
84          console.error("Error fetching products:", error);
85        }
86      };
87
88      fetchProducts();
89    }, [productId]);
90
91      useEffect(() => {
92        fetch('/api/products')
93        .then((res) => res.json())
94        .then((data) => setFeaturedProducts(data.data))
95        .catch((error) => {
96          console.error("Error fetching featured products:", error);
97        });
98      }, []);
99
100
101   const handleAddToCart = () => {
102     if (selectedColor && selectedSize && product && product.images.length > 0) {
103       addToCart({
104         id: product?.id.toString(),
105         image: product?.images?.[0],
106         name: product?.name as string,
107         price: product?.price as number,
108         quantity: quantity,
109         description: product.description,
110         color: selectedColor,
111         size: selectedSize,
112       });
113       toast({
114         className: "rounded-none border border-[#27224b]",
115         title: "Success!",
116         description: `${product.name}  is added to cart.`,
117         duration: 5000,
118       });
119     } else {
120       toast({
```

4.

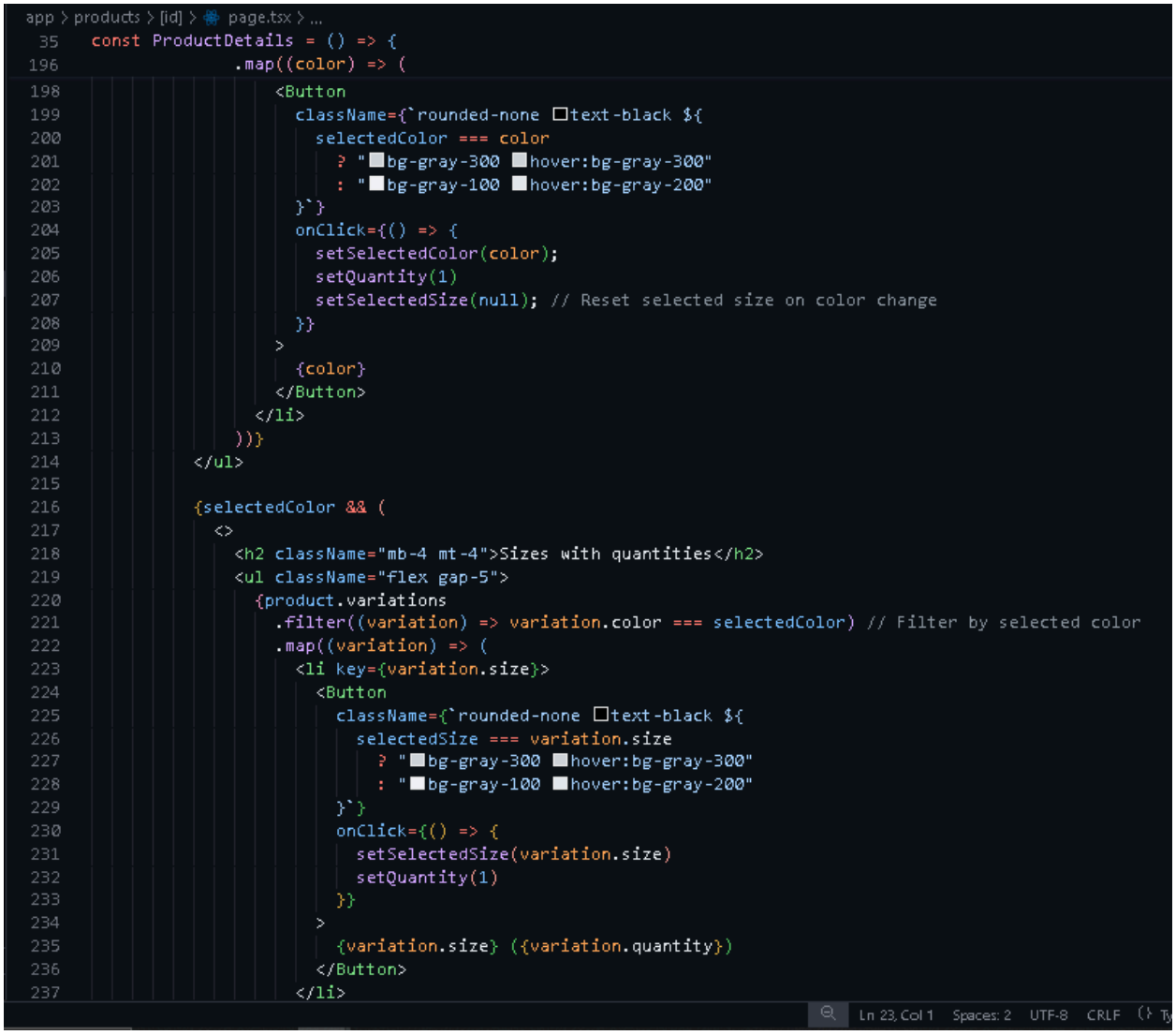
app > products > [id] > page.tsx > ...

```
35  const ProductDetails = () => {
101   const handleAddToCart = () => {
120       toast({
121         className: "rounded-none text-white",
122         variant: "destructive",
123         title: "⚠ Error!",
124         description: "Please select color and size.",
125         duration: 5000,
126       });
127     }
128   };
129
130   const handleIncrement = () => {
131     if (selectedColor && selectedSize) {
132       const selectedVariation = product?.variations.find(
133         (variation) =>
134           variation.color === selectedColor && variation.size === selectedSize
135       );
136       if (selectedVariation && quantity < selectedVariation.quantity) {
137         setQuantity((prevQuantity) => prevQuantity + 1);
138       }
139     }
140   };
141
142   const handleDecrement = () => {
143     if (quantity > 1) {
144       setQuantity((prevQuantity) => {
145         return prevQuantity - 1;
146       });
147     }
148   };
149
150   if (!product) {
151     return (
152       <div className="min-h-screen flex items-center justify-center text-xl font-bold text-red-500">
153         Product not found
154       </div>
155     );
156   }
157
```

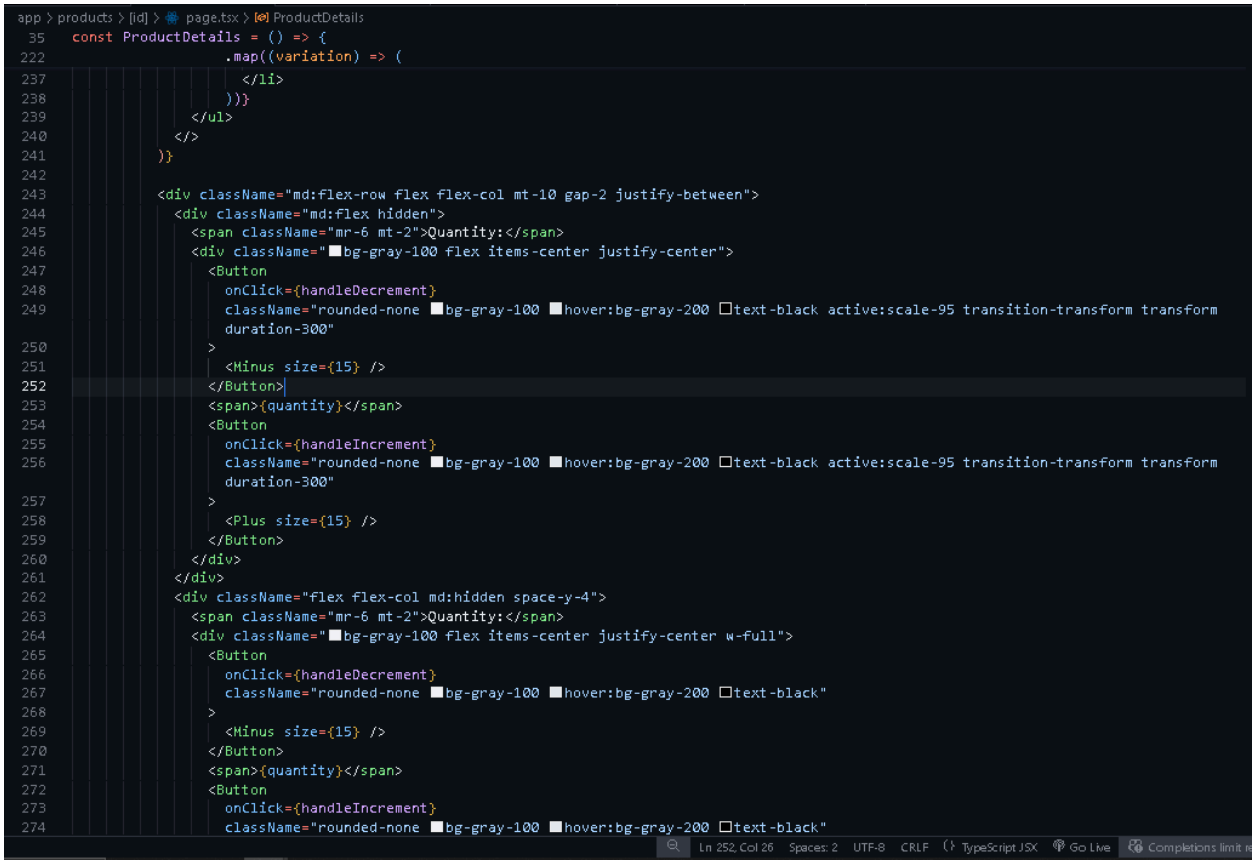
5.


app > products > [id] > page.tsx > ...
35 const ProductDetails = () => {
158 return (
159 <>
160
161
162 <Image
163 src={urlFor(product.images[0]).url()}
164 alt={`Image of ${product.name}`}
165 width={1000}
166 height={1000}
167 className="md:w-[600px] md:h-[600px]"
168 />
169
170
171 <h1 className="text-3xl mb-5 mt-4 font-bold">{product.name}</h1>
172
173 $ {product.price}
174
175 Description:
176
177 {product.description}
178
179 <ul className="flex flex-col mb-4">
180 <li className="flex mr-2">
181 <Dot /> Premium material
182
183 <li className="flex mr-2">
184 <Dot /> Handmade upholstery
185
186 <li className="flex mr-2">
187 <Dot /> Quality timeless classic
188
189
190
191 <h2 className="mb-4 mt-4">Colors</h2>
192 <ul className="flex gap-5">
193 {product.variations
194 .map((variation) => variation.color)
195 .filter((color, index, self) => self.indexOf(color) === index) // Remove duplicate colors
196 .map((color) => (
197 <li key={color}>
198 <Button
Ln 23, Col 1 Spaces: 2 UTF-8 CRLF

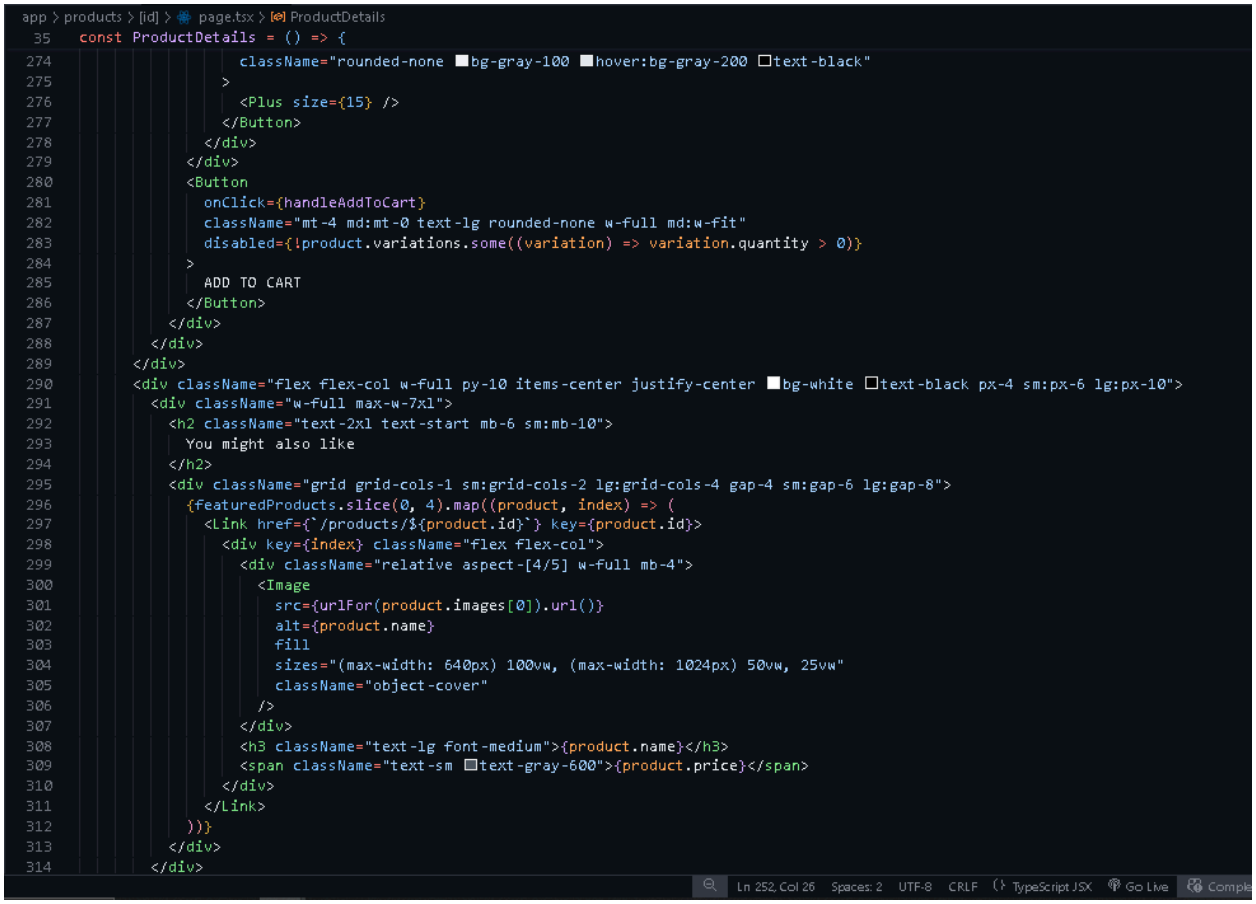
6.


app > products > [id] > page.tsx > ...
35 const ProductDetails = () => {
196 .map((color) => (
198 <Button
199 className={`rounded-none text-black ${
200 selectedColor === color
201 ? "bg-gray-300 hover:bg-gray-300"
202 : "bg-gray-100 hover:bg-gray-200"
203 }`}
204 onClick={() => {
205 setSelectedColor(color);
206 setQuantity(1)
207 setSelectedSize(null); // Reset selected size on color change
208 }}
209 >
210 {color}
211 </Button>
212
213))}
214
215
216 {selectedColor && (
217 <>
218 <h2 className="mb-4 mt-4">Sizes with quantities</h2>
219 <ul className="flex gap-5">
220 {product.variations
221 .filter((variation) => variation.color === selectedColor) // Filter by selected color
222 .map((variation) => (
223 <li key={variation.size}>
224 <Button
225 className={`rounded-none text-black ${
226 selectedSize === variation.size
227 ? "bg-gray-300 hover:bg-gray-300"
228 : "bg-gray-100 hover:bg-gray-200"
229 }`}
230 onClick={() => {
231 setSelectedSize(variation.size)
232 setQuantity(1)
233 }}
234 >
235 {variation.size} ({variation.quantity})
236 </Button>
237
Ln 23, Col 1 Spaces: 2 UTF-8 CRLF

7.


app > products > [id] > page.tsx > ProductDetails
35 const ProductDetails = () => {
222 .map((variation) => (
237
238))}
239
240 </>
241)}
242
243
244
245 Quantity:
246
247 <Button
248 onClick={handleDecrement}
249 className="rounded-none bg-gray-100 hover:bg-gray-200 text-black active:scale-95 transition-transform transform duration-300"
250 >
251 <Minus size={15} />
252 </Button>
253 {quantity}
254 <Button
255 onClick={handleIncrement}
256 className="rounded-none bg-gray-100 hover:bg-gray-200 text-black active:scale-95 transition-transform transform duration-300"
257 >
258 <Plus size={15} />
259 </Button>
260
261
262
263 Quantity:
264
265 <Button
266 onClick={handleDecrement}
267 className="rounded-none bg-gray-100 hover:bg-gray-200 text-black"
268 >
269 <Minus size={15} />
270 </Button>
271 {quantity}
272 <Button
273 onClick={handleIncrement}
274 className="rounded-none bg-gray-100 hover:bg-gray-200 text-black"
Ln 252, Col 26 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Completions limit re

8.



```
app > products > [id] > page.tsx > ProductDetails
35  const ProductDetails = () => {
274                     className="rounded-none bg-gray-100 hover:bg-gray-200 text-black"
275                   >
276                     <Plus size={15} />
277                   </Button>
278                 </div>
279               </div>
280               <Button
281                 onClick={handleAddToCart}
282                 className="mt-4 md:mt-0 text-lg rounded-none w-full md:w-fit"
283                 disabled={!product.variations.some((variation) => variation.quantity > 0)}
284               >
285                 ADD TO CART
286               </Button>
287             </div>
288           </div>
289         </div>
290         <div className="flex flex-col w-full py-10 items-center justify-center bg-white text-black px-4 sm:px-6 lg:px-10">
291           <div className="w-full max-w-7xl">
292             <h2 className="text-2xl text-start mb-6 sm:mb-10">
293               You might also like
294             </h2>
295             <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4 sm:gap-6 lg:gap-8">
296               {featuredProducts.slice(0, 4).map((product, index) => (
297                 <Link href={`/products/${product.id}`} key={product.id}>
298                   <div key={index} className="flex flex-col">
299                     <div className="relative aspect-[4/5] w-full mb-4">
300                       <Image
301                         src={urlFor(product.images[0]).url()}
302                         alt={product.name}
303                         fill
304                         sizes="(max-width: 640px) 100vw, (max-width: 1024px) 50vw, 25vw"
305                         className="object-cover"
306                       />
307                     </div>
308                     <h3 className="text-lg font-medium">{product.name}</h3>
309                     <span className="text-sm text-gray-600">{product.price}</span>
310                   </div>
311                 </Link>
312               ))}
313             </div>
314           </div>
```

Ln 252, Col 26 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Compile

9.

```
app > products > [id] > page.tsx > ProductDetails
35  const ProductDetails = () => {
296               {featuredProducts.slice(0, 4).map((product, index) => (
304                       sizes="(max-width: 640px) 100vw, (max-width: 1024px) 50vw, 25vw"
305                       className="object-cover"
306                     />
307                   </div>
308                   <h3 className="text-lg font-medium">{product.name}</h3>
309                   <span className="text-sm text-gray-600">{product.price}</span>
310                 </div>
311               </Link>
312             ))}
313           </div>
314         </div>
315         <div className="flex justify-center items-center mt-8 sm:mt-12">
316           <Link href="/products">
317             <Button className="bg-gray-200 rounded-none px-6 py-3 sm:py-4 hover:bg-gray-300 text-black w-fit text-sm sm:text-base">
318               View collection
319             </Button>
320           </Link>
321         </div>
322       </div>
323       <WhatMakesUsDiff />
324       <JoinClub />
325     </>
326   );
327 };
328
329 export default ProductDetails;
330
```

Code Snippets of SearchBar:

1. 

components > SearchProduct.tsx > SearchProduct

```tsx
import React, { useState, useRef, useEffect } from "react";
import { useRouter } from "next/navigation";
import { Image as SanityImage } from "@sanity/types";
import { Button } from "./ui/button";
import { Search } from "lucide-react";

interface Product {
  id: string;
  name: string;
  quantity: number;
  price: number;
  images: SanityImage;
  ratings: string;
  sizes: string[];
  colors: string[];
  tags: string[];
  description: string;
}

const SearchProduct = () => {
  const [showSearch, setShowSearch] = useState<boolean>(false);
  const [query, setQuery] = useState<string>("");
  const searchInputRef = useRef<HTMLInputElement>(null);
  const [products, setProducts] = useState<Product[]>([]);
  const router = useRouter();

  useEffect(() => {
    fetch("/api/products")
      .then((res) => res.json())
      .then((data) => setProducts(data.data))
      .catch((error) => {
        console.error("Error fetching featured products:", error);
      });
  }, []);

  const getProductsName = (name: string) => {
    const searchedProduct = products.find(
      (p) => p.name.toLowerCase() === name.toLowerCase()
    );
    if (searchedProduct) {
      router.push(`/products/${searchedProduct.id}`);
      setShowSearch(false);
```

2.

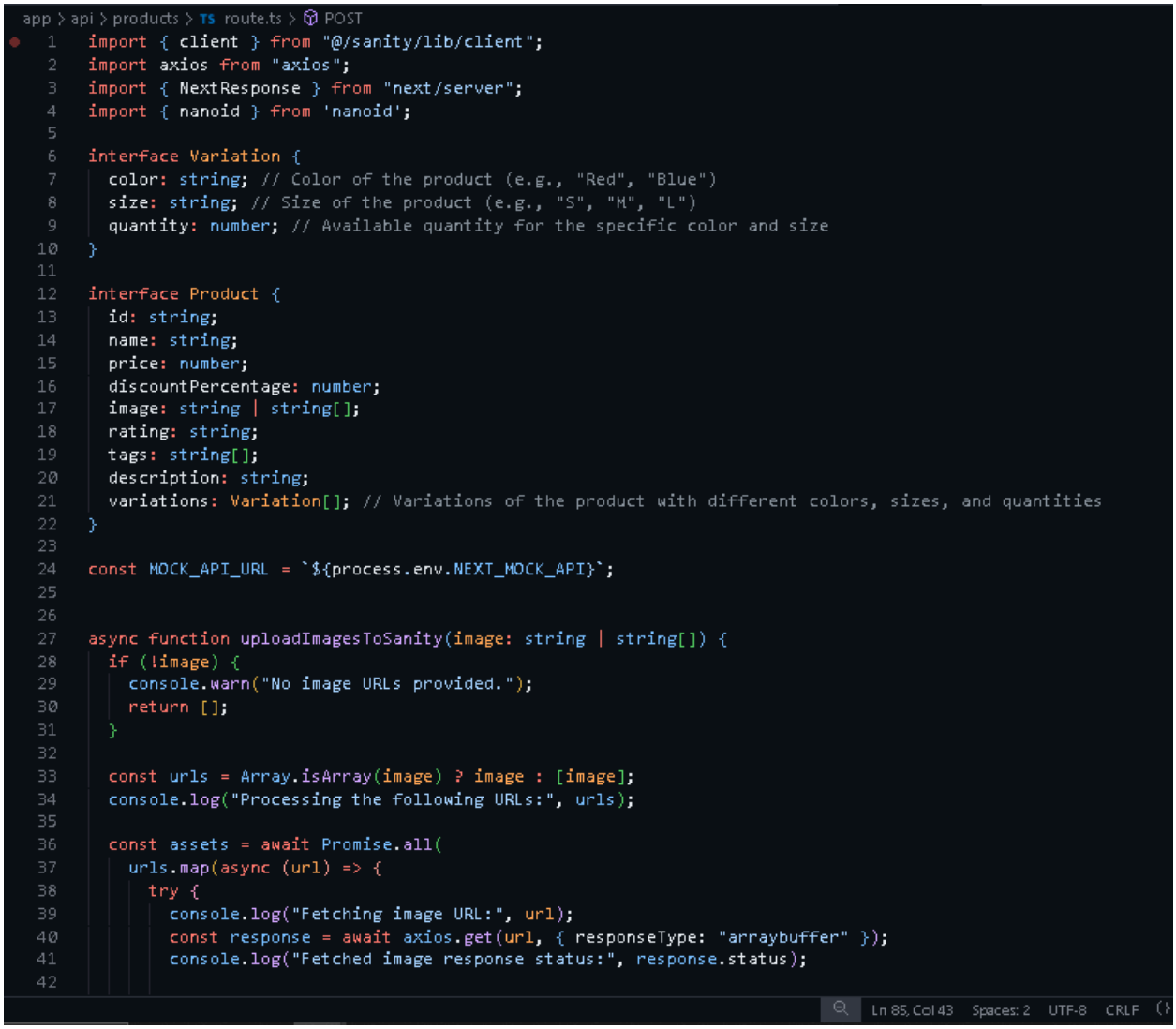

components > SearchProduct.tsx > SearchProduct
20 const SearchProduct = () => {
36 const getProductsName = (name: string) => {
41 router.push(`/products/${searchedProduct.id}`);
42 setShowSearch(false);
43 setQuery("");
44 }
45 };
46
47 const handleSearchClick = () => {
48 if (showSearch && query) {
49 getProductsName(query);
50 } else {
51 setShowSearch((prev) => !prev);
52 }
53 };
54
55 useEffect(() => {
56 if (showSearch && searchInputRef.current) {
57 searchInputRef.current.focus();
58 }
59 }, [showSearch]);
60
61 useEffect(() => {
62 const handleClickOutside = (event: MouseEvent) => {
63 if (
64 searchInputRef.current &&
65 !searchInputRef.current.contains(event.target as Node)
66) {
67 setShowSearch(false);
68 }
69 };
70
71 document.addEventListener("mousedown", handleClickOutside);
72 return () => {
73 document.removeEventListener("mousedown", handleClickOutside);
74 };
75 }, []);
76

3.

```
76
77    return (
78      <div className="relative flex items-center">
79        {showSearch && (
80          <form
81            onSubmit={(e) => {
82              e.preventDefault();
83              getProductsName(query);
84            }}
85            className="absolute animate-in slide-in-from-bottom-full duration-300"
86          >
87            <input
88              ref={searchInputRef}
89              type="text"
90              value={query}
91              onChange={(e) => setQuery(e.target.value)}
92              placeholder="Search product"
93              className="px-4 py-2 rounded-none □text-black -translate-x-32 md:-translate-x-0 w-36 md:w-56"
94            />
95          </form>
96        )}
97        <Button
98          type="button"
99          className={`bg-transparent hover:bg-transparent w-auto h-auto p-0 m-0 ${showSearch ? "opacity-0" : "opacity-100"}`}
100         onClick={handleSearchClick}
101         variant="ghost"
102       >
103         <Search />
104       </Button>
105     </div>
106   );
107 };
108
109 export default SearchProduct;
110
```

Scripts or logic for API integration:

1.


app > api > products > TS route.ts > POST
1 import { client } from "@/sanity/lib/client";
2 import axios from "axios";
3 import { NextResponse } from "next/server";
4 import { nanoid } from 'nanoid';
5
6 interface Variation {
7 color: string; // Color of the product (e.g., "Red", "Blue")
8 size: string; // Size of the product (e.g., "S", "M", "L")
9 quantity: number; // Available quantity for the specific color and size
10 }
11
12 interface Product {
13 id: string;
14 name: string;
15 price: number;
16 discountPercentage: number;
17 image: string | string[];
18 rating: string;
19 tags: string[];
20 description: string;
21 variations: Variation[]; // Variations of the product with different colors, sizes, and quantities
22 }
23
24 const MOCK_API_URL = `${process.env.NEXT_MOCK_API}`;
25
26
27 async function uploadImagesToSanity(image: string | string[]) {
28 if (!image) {
29 console.warn("No image URLs provided.");
30 return [];
31 }
32
33 const urls = Array.isArray(image) ? image : [image];
34 console.log("Processing the following URLs:", urls);
35
36 const assets = await Promise.all(
37 urls.map(async (url) => {
38 try {
39 console.log("Fetching image URL:", url);
40 const response = await axios.get(url, { responseType: "arraybuffer" });
41 console.log("Fetched image response status:", response.status);
42
Ln 85, Col 43 Spaces: 2 UTF-8 CRLF

2.


app > api > products > route.ts > POST
async function uploadImagesToSanity(image: string | string[]) {
const assets = await Promise.all(
urls.map(async (url) => {
try {
console.log("Fetching image URL:", url);
const response = await axios.get(url, { responseType: "arraybuffer" });
console.log("Fetched image response status:", response.status);
const buffer = Buffer.from(response.data, "binary");
const asset = await client.assets.upload("image", buffer, {
filename: `product_image_${Date.now()}.jpg`,
});
console.log("Successfully uploaded asset:", asset);
return {
_type: "image",
_key: nanoid(),
asset: { _type: "reference", _ref: asset._id },
};
} catch (error) {
console.error(`Error uploading image from ${url}:`, error);
return null;
}
})
);
const filteredAssets = assets.filter(Boolean);
console.log("Final filtered assets:", filteredAssets);
return filteredAssets;
}
export async function POST() {
try {
const { data: products } = await axios.get<Product[]>(MOCK_API_URL);
console.log("Fetched products:", products);
if (!Array.isArray(products) || products.length === 0) {
return NextResponse.json(
{ success: false, error: "Invalid or empty product data" },
{ status: 400 }
);
}
Ln 85, Col 43

3.


app > api > products > route.ts > POST
export async function POST() {
async function delay(ms: number) {
return new Promise((resolve) => setTimeout(resolve, ms));
}
const sanityOperations = [];
// First, delete all existing products
// await client.delete({query: '*[_type == "product"]'});
for (const product of products) {
await delay(1000);
console.log("Processing product:", product);
console.log("Image URL:", product.image);
const operation = (async () => {
const images = await uploadImagesToSanity(product.image);
const variations = product.variations.map((variation) => ({
color: variation.color,
size: variation.size,
quantity: variation.quantity,
_key: nanoid(),
}));
const sanityProduct = {
_type: "product",
id: `${product.id}`,
name: product.name,
price: product.price,
priceWithoutDiscount: product.price,
discountPercentage: product.discountPercentage,
description: product.description,
images,
ratings: product.rating,
tags: product.tags,
variations,
};

4.


app > api > products > route.ts > POST
export async function POST() {
const operation = (async () => {
const sanityProduct = {
description: product.description,
images,
ratings: product.rating,
tags: product.tags,
variations,
};
return client.createOrReplace({
_id: `product-${product.id}`,
...sanityProduct,
});
})();
sanityOperations.push(operation);
}
const results = await Promise.all(sanityOperations);
console.log("Products synced successfully!");
return NextResponse.json(
{
success: true,
message: "Products synced successfully!",
data: results,
},
{ status: 200 }
);
} catch (error) {
console.error("Error syncing products:", error);
return NextResponse.json(
{
success: false,
message: "Error syncing products",
error: error instanceof Error ? error.message : String(error),
},
{ status: 500 }
);
}
}

5.

```
148
149  export async function GET(req: Request) {
150    try {
151      const url = new URL(req.url)
152      const category = url.searchParams.get("category")
153
154      const query = category ? `*[_type == "product" && $category in categories]` : `*[_type == "product"]`
155
156      const params = category ? { category } : {}
157
158      const products = await client.fetch(query, params)
159
160      return NextResponse.json({ success: true, data: products }, { status: 200 })
161    } catch (error) {
162      return NextResponse.json(
163        {
164          success: false,
165          message: "Error fetching products",
166          error: error instanceof Error ? error.message : String(error),
167        },
168        { status: 500 },
169      )
170    }
171  }
172
```

3. Documentation:

1. **Products Listing:**

1. Filtering Products by Categories and Price:

- **Categories: Users can filter products by selecting specific categories (e.g., "Mens," "Womens," "Casual Wear," etc.). This is done using checkboxes for each category.**
- **Price Range: A price slider allows users to filter products based on their price range. The slider has a range from $0 to $1000, with the current price range displayed dynamically as the user adjusts it.**

Challenges with solution:

I had a rough idea that for filtering products, I've to use filter methods on products array, but didn't know how how implement how to filter products by categorizes together, so I took help and done first filter for categories and stored it in a variable then put another filter for price ranges on that first category filtered array and rendered that on page.

2. Pagination:

- **The product list is paginated to display a limited number of products per page (6 products) will change later when there are more products. Pagination buttons**

allow users to navigate between pages, ensuring they can view all products without overwhelming the page with too many items.

Challenges with solution:

So for pagination, I did not face any challenge because I had done this before for another website so I just copied the code from there but, I really understood how to implement pagination from this project because I used chat gpt before, but now I fully understood how it works.

3. Wishlist Functionality:

- **Each product has a heart icon that users can click to add the product to their wishlist. Once added, a success toast notification confirms the product was successfully added to the wishlist.**

Challenges with solution:

While implementing wishlist functionality, functionality wasn't that challenging because I made a similar function like Add to cart, named AddToWishlist in my CartContext.tsx and saved wishlist items in local storage, but the challenging part was how to render wishlist items, first I made a separate page for wishlist items, but then I settled using Shadcn sheet components to render wishlist items.

2. **Product Details:**

   1. Dynamic Routing: The page dynamically routes based on the product ID. When users click on a product from the listing, they are taken to its detailed page using Next.js dynamic routing, ensuring the correct product information is displayed.

   Challenges with Solution:

   For dynamic routing I used, "import { useParams} from "next/navigation "", then I used the find function with useParams to find the product with "id" and rendered that product's details.

   2. Dynamic Size Availability: When users select a color from the available options, the available sizes for that color are dynamically displayed based on the current stock, allowing customers to choose the right size for the selected color.

   Challenges with Solution:

   Dynamic size availability was really challenging for me to implement, because I have had to make a function that first filters the colors of products from the variation array from API response, because If a product has three colors and each color has 3 sizes that with every size with the same three color will render that

would create duplication of colors, moreover user would have to select the right color for right size and then select quantity from the same three colors, but I found a way to just render every color once avoiding duplication by using filter and indexOf method by storing the every unique colors index that if the color appears second time the filter function will check it's index and if the index of duplicate color doesn't match the index of that unique color it will remove it and then render just the unique colors, and based on those unique colors I mapped over sizes to render sizes for those unique colors with their real-time stock display.

3. Add to Cart Functionality: Users can easily add products to their cart by selecting the desired color, size, and quantity, then clicking the "Add to Cart" button. The product details, including color, size, and quantity, are added to the cart state.

Challenges with solution:

I implemented Add to Cart functionality by making a function called "addToCart", this function would first check if an item already exists in cart if it does, it would just increment its quantity by the quantity selected by the user, else it would just add that item to cart.

4. Toast Notification: Upon successfully adding an item to the cart, a toast notification appears, confirming the action with a message like "Product Name is added to cart," providing instant feedback to the user.

Challenges with solution:

I used shadcn Toast component for toast notification, I added toast in add to cart function, so that when an item is added to cart a toast notification would appear with dynamically rendering the products name in notification, if the add to cart function fails for any reason like when the user hasn't selected any color or size it would show error notification in toast.

3. **Products Search:**

The `SearchProduct` component is a dynamic search interface for finding products. It integrates with the Sanity CMS, allowing users to search for products by name and navigate to their product details pages directly.

Challenges with solution:

The search product function, takes the product in an input field and when user press enter, it converts the provided name and all the products names to lowercase, and if a product name matches the provided name it would redirect user to that product's details page where can see that product's details and add product to cart.