

Day 4 - Dynamic Frontend Components - Avion

1. Functional Deliverables:

ScreenShot Of Products Listing:

The screenshot shows a product listing interface with the following elements:

- Header:** A search bar, the brand name "Avion", and three icons (heart, cart, user).
- Breadcrumbs:** All Products, Mens, Womens, Kids, Casual Wear, Formal Attire, Active Wear, Accessories.
- Section Header:** "All products" (with a large image of a cat's face in the background).
- Product Type Filter:** A checkbox list for Mens, Womens, Kids, Casual Wear, Formal Attire, Active Wear, and Accessories.
- Price Filter:** A slider from \$0 to \$1000.
- Product Cards:** Three cards are visible:
 - Small Concrete Sausages:** \$36.00
 - Awesome Fresh Fish:** \$402.00
 - Awesome Metal Car:** \$446.00

ScreenShot Of Product Details:

The screenshot shows a product details page for the "Awesome Concrete Keyboard".

Product Image: An illustration of two black cats pulling on a red ribbon in a desert landscape.

Product Title: **Awesome Concrete Keyboard**

Price: **\$ 426.00**

Description: The slim & simple Maple Gaming Keyboard from Dev Byte comes with a sleek body and 7- Color RGB LED Back-lighting for smart functionality

Features:

- Premium material
- Handmade upholstery
- Quality timeless classic

Colors: Red, Blue, Black

Quantity: - 1 +

Add To Cart: A button labeled "ADD TO CART".

ScreenShot Of Products Filtering And Pagination:

The screenshot shows a mobile application interface for filtering products. At the top left, a dark navigation bar displays the text "All products". Below this, there are two sections: "Product type" and "Price".

Product type: A list of categories with checkboxes. The checked categories are Mens, Womens, and Kids. The unchecked categories are Casual Wear, Formal Attire, Active Wear, and Accessories.

Price: A horizontal slider with a circular track and a blue slider bar. The minimum value is \$559 and the maximum value is \$1000. The current price range is indicated by a blue segment between the two values.

A product card for a "Generic Granite Fish" is displayed, showing a black statue of a fish on a concrete base. The price is listed as \$630.00. To the right of the product card is a small heart icon.

Pagination controls at the bottom show page 1 of 2.

Mobile Categories Filtering:

All products

Categories ▾

Filter ▾

Mens

Womens

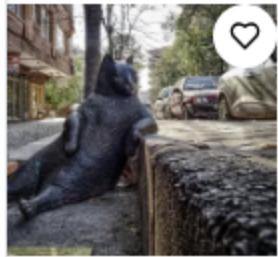
Kids

Casual Wear

Formal Attire

Active Wear

Accessories

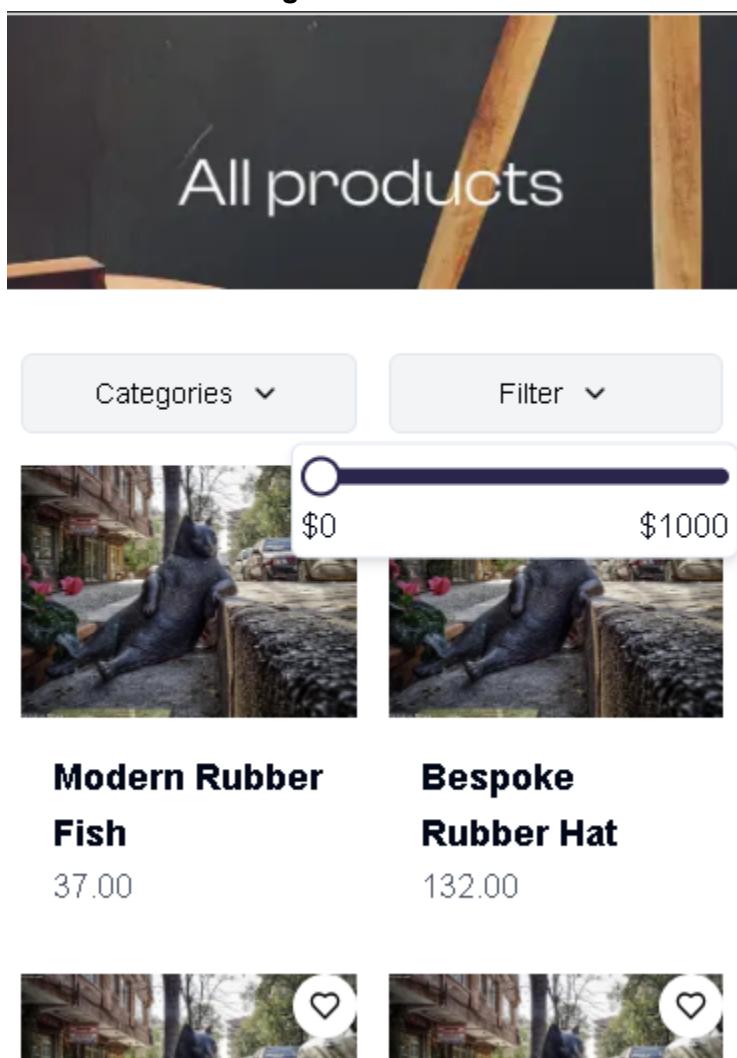


**espoke
ubber Hat**

\$2.00



Mobile Price Filtering:



ScreenShots Of Product Categories:

1. Kid's Category:

Q Avion

All Products Mens Womens Kids Casual Wear Formal Attire Active Wear Accessories

Kids Products



Generic Granite Fish
\$630.00



Awesome Fresh Fish
\$402.00

2. Men's Category:

Q Avion

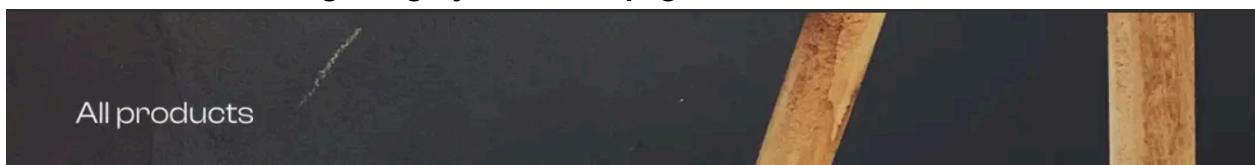
All Products Mens Womens Kids Casual Wear Formal Attire Active Wear Accessories

Mens Products



Bespoke Rubber Hat
\$132.00

ScreenShot Of Working category filters and pagination:



Product type

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



Price

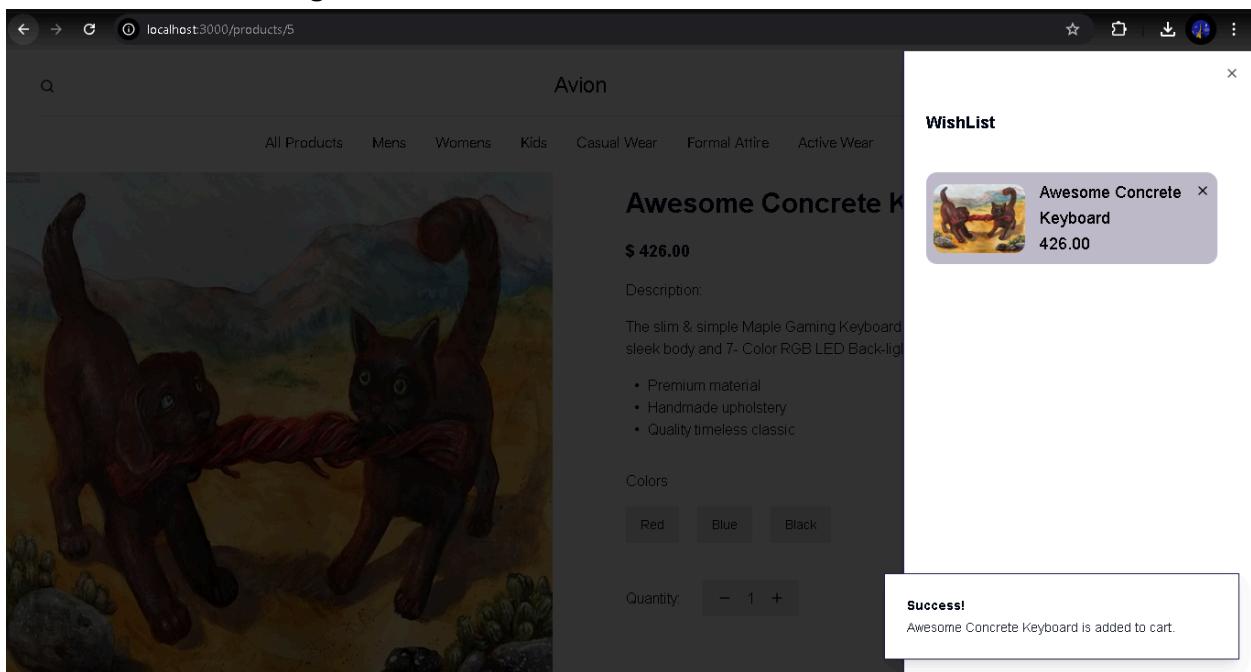
\$559 \$1000

Generic Granite Fish

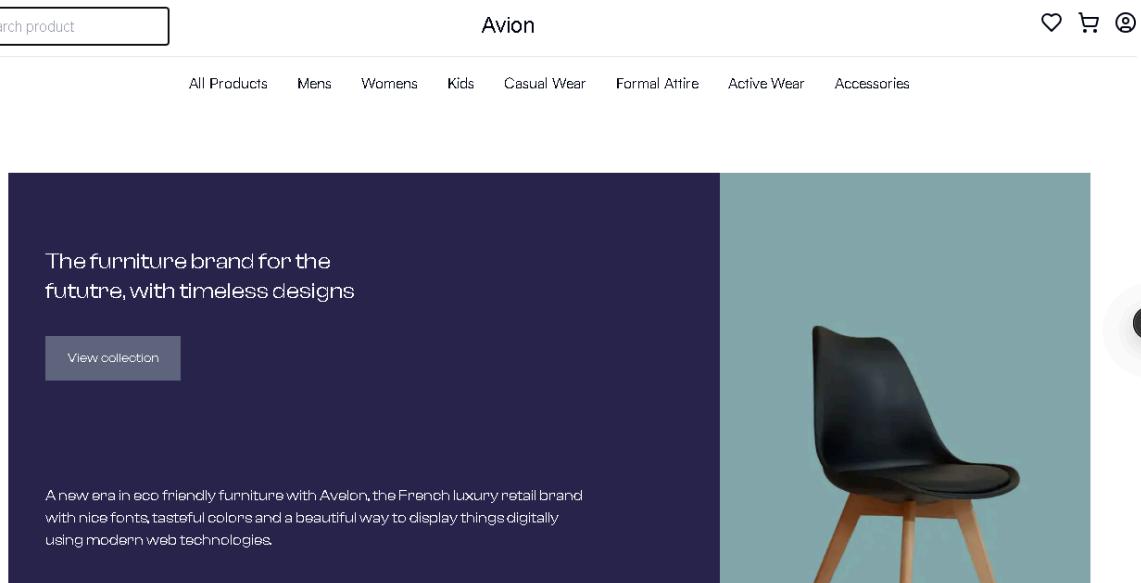
630.00

1 2

ScreenShot Of working Wishlist with Toast Notification:



ScreenShot of Working search bar:



2. Code Deliverables:

Code Snippet of Products Listing:

```
components > 📁 ProductListing.tsx > • Product > ⚙ sizes
1  "use client";
2
3  import Image from "next/image";
4  import Link from "next/link";
5  import React, { useEffect, useState } from "react";
6  import { Checkbox } from "./ui/checkbox";
7  import { Label } from "./ui/label";
8  import { Button } from "./ui/button";
9  import { ChevronDown, Heart } from "lucide-react";
10 import { urlFor } from "@/sanity/lib/image";
11 import { Image as SanityImage } from "@sanity/types";
12 import { Slider } from "./ui/slider";
13 import {
14   DropdownMenu,
15   DropdownMenuCheckboxItem,
16   DropdownMenuContent,
17   DropdownMenuTrigger,
18 } from "@/components/ui/dropdown-menu";
19 import { useCart } from "@/lib/CartContext";
20 import { toast } from "@/hooks/use-toast";
21
22 interface Product {
23   id: string;
24   name: string;
25   quantity: number;
26   price: number;
27   images: SanityImage;
28   ratings: string;
29   sizes: string[];
30   colors: string[];
31   tags: string[];
32   categories: string[];
33   description: string;
34 }
```

```
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
```

2.

```
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
```

3.

```
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
```

4.

```
139     return (
140       <div className="w-full flex flex-col items-center justify-center pb-10">
141         <Image
142           src="/Images/Page Headers.png"
143           alt="Page Header Image"
144           width={1000}
145           height={100}
146           className="w-full hidden md:flex"
147         />
148         <Image
149           src="/Images/Page Headers (1).png"
150           alt="Page Header Image"
151           width={1000}
152           height={100}
153           className="w-full md:hidden"
154         />
155       <div className="flex w-full px-2 md:px-6 lg:px-8 mt-8 gap-8">
156         <div className="md:flex flex-col w-1/4 hidden">
157           <h2 className="text-2xl font-semibold mb-4">Product type</h2>
158           <div className="space-y-2">
159             {productTypes.map((item) => (
160               <div key={item} className="flex items-center space-x-2">
161                 <checkbox
162                   id={item}
163                   onChange={(checked) =>
164                     handleCategoryChange(item, checked as boolean)
165                   }
166                 />
167                 <Label
168                   htmlFor={item}
169                   className="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70"
170                 >
171                   {item}
172                 </Label>
173               </div>
174             ))
175           </div>
176         </div>
177       </div>
178     )
179   )
180 
```

5.

```
172           </Label>
173       </div>
174     )})
175   </div>
176   <h2 className="text-2xl font-semibold my-4">Price</h2>
177   <div className="space-y-2">
178     <Slider
179       value={sliderValue}
180       onValueChange={(value) => setSliderValue(value)} // Update slider value
181       min={0} // Min price
182       max={1000} // Max price
183       step={1} // Step for slider movement
184     />
185   </div>
186   <div className="flex justify-between mt-2">
187     <span>${sliderValue[0]}</span>
188     <span>${sliderValue[1]}</span>
189   </div>
190 </div>
191 <div className="grid grid-cols-2 md:grid-cols-2 lg:grid-cols-3 gap-4 w-full">
192   <DropdownMenu>
193     <DropdownMenuTrigger asChild>
194       <Button
195         variant="outline"
196         className="md:hidden bg-gray-100 hover:bg-gray-200 text-black"
197       >
198         Categories <ChevronDown />
199       </Button>
200     </DropdownMenuTrigger>
201     <DropdownMenuContent className="w-56">
202       {productTypes.map((category) => (
```

6.

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>
```

7.

```
components > ProductListing.tsx > [x] ProductListing > currentFilteredProducts.map() callback
  46  const ProductListing = () => {
  47    </div>
  48    <div className="flex justify-between mt-2">
  49      <span>${sliderValue[0]}</span>
  50      <span>${sliderValue[1]}</span>
  51    </div>
  52  </DropdownMenuContent>
  53 </DropdownMenu>
  54 {currentFilteredProducts.map((product) => (
  55   <Link
  56     href={`/products/${product.id}`}
  57     key={product.id}
  58     className="block"
  59   >
  60     <div className="overflow-hidden">
  61       /* Check if image exists before rendering */
  62       {product.images?.[0] ? (
  63         <div className="relative">
  64           <Image
  65             src={urlFor(product.images[0]).url()}
  66             alt={product.name}
  67             width={300}
  68             height={300}
  69             className="w-full h-auto object-cover"
  70           />
  71           <Button
  72             className="absolute top-0 right-0 bg-white hover:bg-white/90 active:scale-95 transition-transform duration-300 p-2 rounded-full w-fit h-fit"
  73             onClick={() => handleAddItemToWishlist(product)}
  74             | <Heart className="text-black" />
  75           </Button>
  76         </div>
  77       ) : (
  78         <div className="w-full h-24 md:h-60 bg-gray-200 flex items-center justify-center text-gray-500">
```

```
components > ProductListing.tsx > ProductListing > currentFilteredProducts.map() callback
  46  const ProductListing = () => {
  47    <div>
  48      <currentFilteredProducts.map((product) => (
  49        <a href="#">
  50          <div>
  51            <img alt="No Image Available" data-bbox="100px 100px" />
  52          </div>
  53        <div className="p-4">
  54          <h3>{product.name}</h3>
  55          <p>{product.price}</p>
  56        </div>
  57      </a>
  58    )}>
  59  </div>
  60  <div data-bbox="100px 100px" style="text-align: center; margin-top: 5px">
  61    <div>
  62      <button>
  63        <span>1</span>
  64      </button>
  65      <span>2</span>
  66      <span>3</span>
  67      <span>4</span>
  68    </div>
  69  </div>
  70  <div data-bbox="100px 100px" style="text-align: center; margin-top: 10px">
  71    <div>
  72      <button>
  73        <span>View All</span>
  74      </button>
  75    </div>
  76  </div>
  77  <div data-bbox="100px 100px" style="text-align: center; margin-top: 10px">
  78    <div>
  79      <button>
  80        <span>View All</span>
  81      </button>
  82    </div>
  83  </div>
  84  <div data-bbox="100px 100px" style="text-align: center; margin-top: 10px">
  85    <div>
  86      <button>
  87        <span>View All</span>
  88      </button>
  89    </div>
  90  </div>
  91  <div data-bbox="100px 100px" style="text-align: center; margin-top: 10px">
  92    <div>
  93      <button>
  94        <span>View All</span>
  95      </button>
  96    </div>
  97  </div>
  98  </div>
  99  </div>
 100 </div>
 101 </div>
 102 </div>
 103 </div>
 104 </div>
 105 </div>
 106 </div>
 107 </div>
 108 </div>
 109 </div>
 110 </div>
 111 </div>
 112 </div>
 113 </div>
 114 </div>
 115 </div>
 116 </div>
 117 </div>
 118 </div>
 119 </div>
 120 </div>
 121 </div>
 122 </div>
 123 </div>
 124 </div>
 125 </div>
 126 </div>
 127 </div>
 128 </div>
 129 </div>
 130 </div>
 131 </div>
 132 </div>
 133 </div>
 134 </div>
 135 </div>
 136 </div>
 137 </div>
 138 </div>
 139 </div>
 140 </div>
 141 </div>
 142 </div>
 143 </div>
 144 </div>
 145 </div>
 146 </div>
 147 </div>
 148 </div>
 149 </div>
 150 </div>
 151 </div>
 152 </div>
 153 </div>
 154 </div>
 155 </div>
 156 </div>
 157 </div>
 158 </div>
 159 </div>
 160 </div>
 161 </div>
 162 </div>
 163 </div>
 164 </div>
 165 </div>
 166 </div>
 167 </div>
 168 </div>
 169 </div>
 170 </div>
 171 </div>
 172 </div>
 173 </div>
 174 </div>
 175 </div>
 176 </div>
 177 </div>
 178 </div>
 179 </div>
 180 </div>
 181 </div>
 182 </div>
 183 </div>
 184 </div>
 185 </div>
 186 </div>
 187 </div>
 188 </div>
 189 </div>
 190 </div>
 191 </div>
 192 </div>
 193 </div>
 194 </div>
 195 </div>
 196 </div>
 197 </div>
 198 </div>
 199 </div>
 200 </div>
 201 </div>
```

Code Snippet of Product Details:

```
app > products > [id] > page.tsx > ...
1  "use client";
2
3  import { useParams } from "next/navigation";
4  import Image from "next/image";
5  import { Button } from "@/components/ui/button";
6  import { Dot } from "lucide-react";
7  import WhatMakesUsDiff from "@/components/WhatMakesUsDiff";
8  import JoinClub from "@/components/JoinClub";
9  import Link from "next/link";
10 import { useState, useEffect } from "react";
11 import { urlFor } from "@/sanity/lib/image";
12 import { Image as SanityImage } from "@sanity/types";
13 import { client } from "@/sanity/lib/client";
14 import { useCart } from "@/lib/CartContext";
15 import { toast } from "@/hooks/use-toast";
16 import { Plus, Minus } from "lucide-react";
17
18 interface Variation {
19   color: string; // Color of the product (e.g., "Red", "Blue")
20   size: string; // Size of the product (e.g., "S", "M", "L")
21   quantity: number; // Available quantity for the specific color and size
22 }
23
24 interface Product {
25   id: number; // Unique identifier for the product
26   name: string; // Name of the product
27   price: number; // Price of the product
28   images: SanityImage[]; // Array of images for the product
29   ratings: string; // Rating of the product (e.g., "4.5")
30   tags: string[]; // Tags associated with the product (e.g., ["Best Seller"])
31   description: string; // Detailed description of the product
32   variations: Variation[]; // Variations of the product with different colors, sizes, and quantities
33 }
34
35 const ProductDetails = () => {
36   const { addToCart } = useCart();
37   const [products, setProducts] = useState<Product>([]);
38   const [featuredProducts, setFeaturedProducts] = useState<Product>([]);
39   const [selectedColor, setSelectedColor] = useState<string | null>(null);
40   const [selectedSize, setSelectedSize] = useState<string | null>(null);
41   const [quantity, setQuantity] = useState<number>(1);
42 }
```

1.

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

```
35  const ProductDetails = () => {
42
43    const params = useParams();
44    const productId = params.id as string;
45    const product = products.find((p) => String(p.id) === productId);
46
47    useEffect(() => {
48      const fetchProducts = async () => {
49        const query = `*[_type == "product" && id == $productId]{
50          id,
51          name,
52          price,
53          "images": images[].asset->{
54            _id,
55            _key,
56            _ref,
57            url,
58            metadata
59          },
60          ratings,
61          discountPercentage,
62          priceWithoutDiscount,
63          ratingCount,
64          description,
65          variations[] {
66            color,
67            size,
68            quantity
69          },
70          tags
71        }`;
72
73        try {
74          const fetchedProducts = await client.fetch(query, {
75            productId: productId,
76          });
77
78          if (fetchedProducts.length > 0) {
79            console.log("Fetched product images:", fetchedProducts[0].images);
80          }
81        }
82      }
83    }
84  
```

2.

```
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({
```

3.

Θ | a.23.col

```
app > products > [id] > page.tsx ...
  35  const ProductDetails = () => {
  51    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
```

4.

```
app > products > [id] > page.tsx > ...
35  const ProductDetails = () => {
158    return (
159      <>
160        <div className="flex lg:flex-row flex-col">
161          <div className="flex space-x-2">
162            <Image
163              src={urlFor(product.images[0]).url()}
164              alt={`Image of ${product.name}`}
165              width={1000}
166              height={1000}
167              className="md:w-[600px] md:h-[600px]"
168            />
169          </div>
170        <div className="flex flex-col lg:pl-20 pr-5 w-full max-w-2xl px-10 md:px-0">
171          <h1 className="text-3xl mb-5 mt-4 font-bold">{product.name}</h1>
172
173          <p className="text-lg font-bold mb-4">${product.price}</p>
174
175          <span className="mb-4">Description:</span>
176
177          <p className="mb-4 max-w-lg">{product.description}</p>
178
179          <ul className="flex flex-col mb-4">
180            <li className="flex mr-2">
181              <Dot /> Premium material
182            </li>
183            <li className="flex mr-2">
184              <Dot /> Handmade upholstery
185            </li>
186            <li className="flex mr-2">
187              <Dot /> Quality timeless classic
188            </li>
189          </ul>
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
```

5.



Ln 23, Col 1 Spaces: 2 UTF-8 CRLF ⌂ Ty

```
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      </li>
213  )));
214 </ul>
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                      </li>
238                  ))
239          </ul>
240      </>
241  )}
242
```

6.

Ln 23, Col 1 Spaces: 2 UTF-8 CRLF ⌂ Ty

```
app > products > [id] > page.tsx > [e] ProductDetails
 35  const ProductDetails = () => {
222    .map((variation) => (
237      </li>
238    )));
239  </ul>
240  </>
241  )}
242
243  <div className="md:flex-row flex flex-col mt-10 gap-2 justify-between">
244    <div className="md:flex hidden">
245      <span className="mr-6 mt-2">Quantity:</span>
246      <div className="bg-gray-100 flex items-center justify-center">
247        <Button
248          onClick={handleDecrement}
249          className="rounded-none bg-gray-100 hover:bg-gray-200 text-black active:scale-95 transition-transform duration-300"
250        >
251          <Minus size={15} />
252        </Button>
253        <span>{quantity}</span>
254        <Button
255          onClick={handleIncrement}
256          className="rounded-none bg-gray-100 hover:bg-gray-200 text-black active:scale-95 transition-transform duration-300"
257        >
258          <Plus size={15} />
259        </Button>
260      </div>
261    </div>
262    <div className="flex flex-col md:hidden space-y-4">
263      <span className="mr-6 mt-2">Quantity:</span>
264      <div className="bg-gray-100 flex items-center justify-center w-full">
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        <span>{quantity}</span>
272        <Button
273          onClick={handleIncrement}
274          className="rounded-none bg-gray-100 hover:bg-gray-200 text-black"
275        >
```

7.

Ln 252 Col 26 Spaces: 2 UTF-8 CRLF ⓘ TypeScript JSK ⓘ Go Live ⓘ Completions limits

```

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

```

8.

```

app > products > [id] > page.tsx > ProductDetails
35  const ProductDetails = () => {
296      <div>
297          <div>
298              <h2>You might also like</h2>
299              <div>
300                  <div>
301                      <div>
302                          <div>
303                              <Image
304                                  src={urlFor(product.images[0]).url()}
305                                  alt={product.name}
306                                  fill
307                                  sizes="(max-width: 640px) 100vw, (max-width: 1024px) 50vw, 25vw"
308                                  className="object-cover"
309                              </Image>
310                          <h3>{product.name}</h3>
311                          <span>{product.price}</span>
312                      </div>
313                  </div>
314              </div>
315          <div>
316              <div>
317                  <Link href="/products">
318                      <button>View collection</button>
319                  </Link>
320              </div>
321          </div>
322      </div>
323      <WhatMakesUsDiff />
324      <JoinClub />
325  </div>
326  </div>;
327  </div>;
328
329  export default ProductDetails;
330

```

9.

Code Snippets of SearchBar:

```
components > SearchProduct.tsx > [e] SearchProduct
  1 import React, { useState, useRef, useEffect } from "react";
  2 import { useRouter } from "next/navigation";
  3 import { Image as SanityImage } from "@sanity/types";
  4 import { Button } from "./ui/button";
  5 import { Search } from "lucide-react";
  6
  7 interface Product {
  8   id: string;
  9   name: string;
 10   quantity: number;
 11   price: number;
 12   images: SanityImage;
 13   ratings: string;
 14   sizes: string[];
 15   colors: string[];
 16   tags: string[];
 17   description: string;
 18 }
 19
 20 const SearchProduct = () => {
 21   const [showSearch, setShowSearch] = useState<boolean>(false);
 22   const [query, setQuery] = useState<string>("");
 23   const searchInputRef = useRef<HTMLInputElement>(null);
 24   const [products, setProducts] = useState<Product[]>([]);
 25   const router = useRouter();
 26
 27   useEffect(() => {
 28     fetch("/api/products")
 29       .then((res) => res.json())
 30       .then((data) => setProducts(data.data))
 31       .catch((error) => {
 32         console.error("Error fetching featured products:", error);
 33       });
 34   }, []);
 35
 36   const getProductsName = (name: string) => {
 37     const searchedProduct = products.find(
 38       (p) => p.name.toLowerCase() === name.toLowerCase()
 39     );
 40     if (searchedProduct) {
 41       router.push(`/products/${searchedProduct.id}`);
 42       setShowSearch(false);
 43     }
 44   };
 45 }
```

1.

```
components > SearchProduct.tsx > [o] SearchProduct
20  const SearchProduct = () => {
36    const getProductsName = (name: string) => {
37      router.push(`products/${searchedProduct.id}`);
38      setShowSearch(false);
39      setQuery("");
40    }
41  };
42
43  const handleSearchClick = () => {
44    if (showSearch && query) {
45      getProductsName(query);
46    } else {
47      setShowSearch((prev) => !prev);
48    }
49  };
50
51  useEffect(() => {
52    if (showSearch && searchInputRef.current) {
53      searchInputRef.current.focus();
54    }
55  }, [showSearch]);
56
57  useEffect(() => {
58    const handleClickOutside = (event: MouseEvent) => {
59      if (
60        searchInputRef.current &&
61        !searchInputRef.current.contains(event.target as Node)
62      ) {
63        setShowSearch(false);
64      }
65    };
66
67    document.addEventListener("mousedown", handleClickOutside);
68    return () => {
69      document.removeEventListener("mousedown", handleClickOutside);
70    };
71  }, []);
72
```

2.

```
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 outline-transparent -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 ~> export default SearchProduct;
109 ~>
110 ~>
```

3.

Scripts or logic for API integration:

```
app > api > products > 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       } catch (error) {
43         console.error(`Error fetching image URL ${url}: ${error.message}`);
44       }
45     })
46   );
47
48   return assets;
49 }
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
313
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435

```

```
app > api > products > TS route.ts > POST
27   async function uploadImagesToSanity(image: string | string[]) {
28     const assets = await Promise.all(
29       urls.map(async (url) => {
30         try {
31           console.log("Fetching image URL:", url);
32           const response = await axios.get(url, { responseType: "arraybuffer" });
33           console.log("Fetched image response status:", response.status);
34
35           const buffer = Buffer.from(response.data, "binary");
36           const asset = await client.assets.upload("image", buffer, {
37             filename: `product_image_${Date.now()}.jpg`,
38           });
39           console.log("Successfully uploaded asset:", asset);
40
41         return {
42           _type: "image",
43           _key: nanoid(),
44           asset: { _type: "reference", _ref: asset._id },
45         };
46       } catch (error) {
47         console.error(`Error uploading image from ${url}:`, error);
48         return null;
49       }
50     })
51   );
52
53   const filteredAssets = assets.filter(Boolean);
54   console.log("Final Filtered assets:", filteredAssets);
55   return filteredAssets;
56 }
57
58
59
60
61
62
63
64
65
66 export async function POST() {
67   try {
68     const { data: products } = await axios.get<Product[]>(MOCK_API_URL);
69     console.log("Fetched products:", products);
70
71     if (!Array.isArray(products) || products.length === 0) {
72       return NextResponse.json(
73         { success: false, error: "Invalid or empty product data" },
74         { status: 400 }
75       );
76     }
77   }
78 }
```

2.

Ln 85, Col 43

```
app > api > products > ts routes > POST
66  export async function POST() {
67    // ...
68    const delay = (ms: number) => {
69      return new Promise((resolve) => setTimeout(resolve, ms));
70    }
71
72    const sanityOperations = [];
73
74    // First, delete all existing products
75    // await client.delete({query: '*[_type == "product"]'});
76
77    for (const product of products) {
78      await delay(1000);
79      console.log("Processing product:", product);
80      console.log("Image URL:", product.image);
81
82      const operation = (async () => {
83        const images = await uploadImagesToSanity(product.image);
84
85        const variations = product.variations.map((variation) => ({
86          color: variation.color,
87          size: variation.size,
88          quantity: variation.quantity,
89          _key: nanoid(),
90        }));
91
92        const sanityProduct = {
93          _type: "product",
94          id: `${product.id}`,
95          name: product.name,
96          price: product.price,
97          priceWithoutDiscount: product.price,
98          discountPercentage: product.discountPercentage,
99          description: product.description,
100         images,
101         ratings: product.rating,
102         tags: product.tags,
103         variations,
104       };
105     });
106   }
107 }
```

3.

```
app > api > products > route.ts > POST
66  export async function POST() {
92      const operation = (async () => {
102          const sanityProduct = {
109              description: product.description,
110              images,
111              ratings: product.rating,
112              tags: product.tags,
113              variations,
114          };
115
116          return client.createOrReplace({
117              _id: `product-${product.id}`,
118              ...sanityProduct,
119          });
120      })();
121      sanityOperations.push(operation);
122  }
123
124  const results = await Promise.all(sanityOperations);
125
126  console.log("Products synced successfully!");
127  return NextResponse.json(
128      {
129          success: true,
130          message: "Products synced successfully!",
131          data: results,
132      },
133      { status: 200 }
134  );
135 } catch (error) {
136     console.error("Error syncing products:", error);
137     return NextResponse.json(
138         {
139             success: false,
140             message: "Error syncing products",
141             error: error instanceof Error ? error.message : String(error),
142         },
143         { status: 500 }
144     );
145 }
146 }
```

4.

```

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

```

5.

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 to implement how to filter products by categories 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.