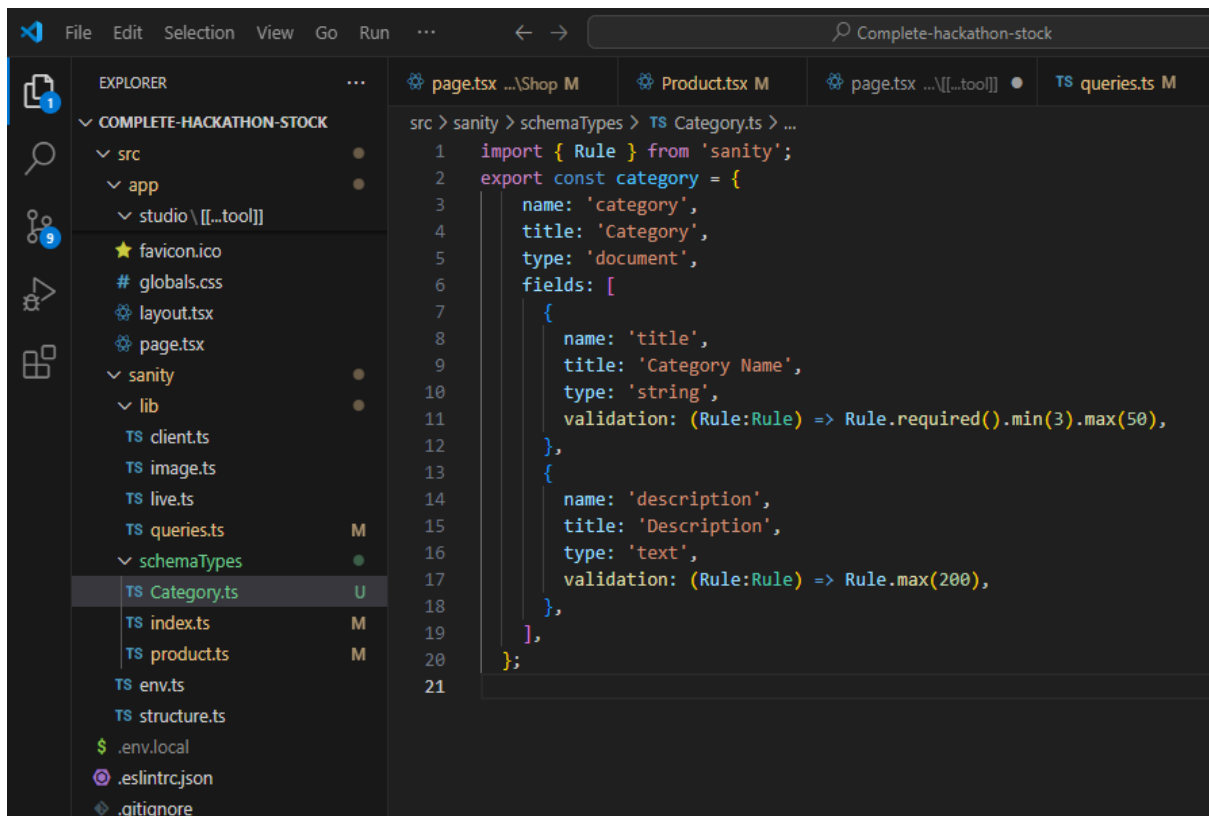


MUHAMMAD RAMZAN

(00367450)

Day 5 Implementation Report

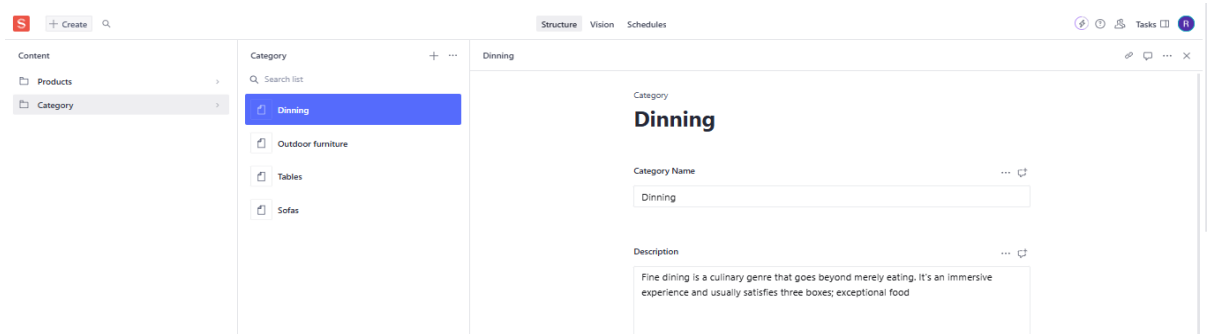
I add product categories in my project first of all i did create a schema of category and give a reference of category in my product schema



The screenshot shows the VS Code editor with a project named 'Complete-hackathon-stock'. The Explorer panel on the left shows the file structure, including a 'schemaTypes' directory. The main editor displays the 'Category.ts' file, which defines a Sanity schema for a category. The schema includes fields for 'title' and 'description', both with validation rules.

```
src > sanity > schemaTypes > TS Category.ts > ...
1  import { Rule } from 'sanity';
2  export const category = {
3    name: 'category',
4    title: 'Category',
5    type: 'document',
6    fields: [
7      {
8        name: 'title',
9        title: 'Category Name',
10       type: 'string',
11       validation: (Rule:Rule) => Rule.required().min(3).max(50),
12     },
13     {
14       name: 'description',
15       title: 'Description',
16       type: 'text',
17       validation: (Rule:Rule) => Rule.max(200),
18     },
19   ],
20 };
21
```

And this is my sanity schema



I add this category functionality on my shop page there my products will show by its category and it will dynamically show on my Mainproduct page

This is my shop page code for the implementation of category

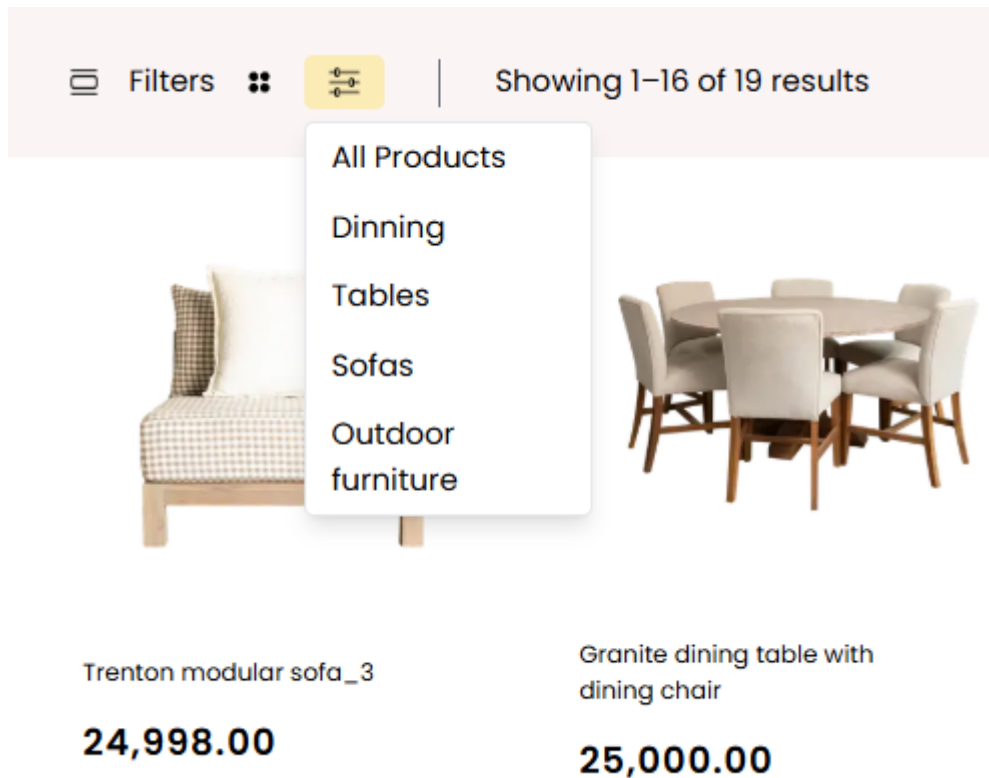
```
// Fetch categories
useEffect(() => {
  const fetchCategories = async () => {
    try {
      const fetchedCategories = await getCategories();
      setCategories(fetchedCategories);
    } catch (error) {
      console.error("Error fetching categories:", error);
    }
  }
  fetchCategories();
}, []);

// Fetch products by category
const handleCategoryClick = async (categoryTitle: string) => {
  if (categoryTitle === "All Products") {
    setProducts(allProducts); // Show all products
    setCategoryProducts({});
    setCurrentPage(1); // Reset to page 1
    return;
  }

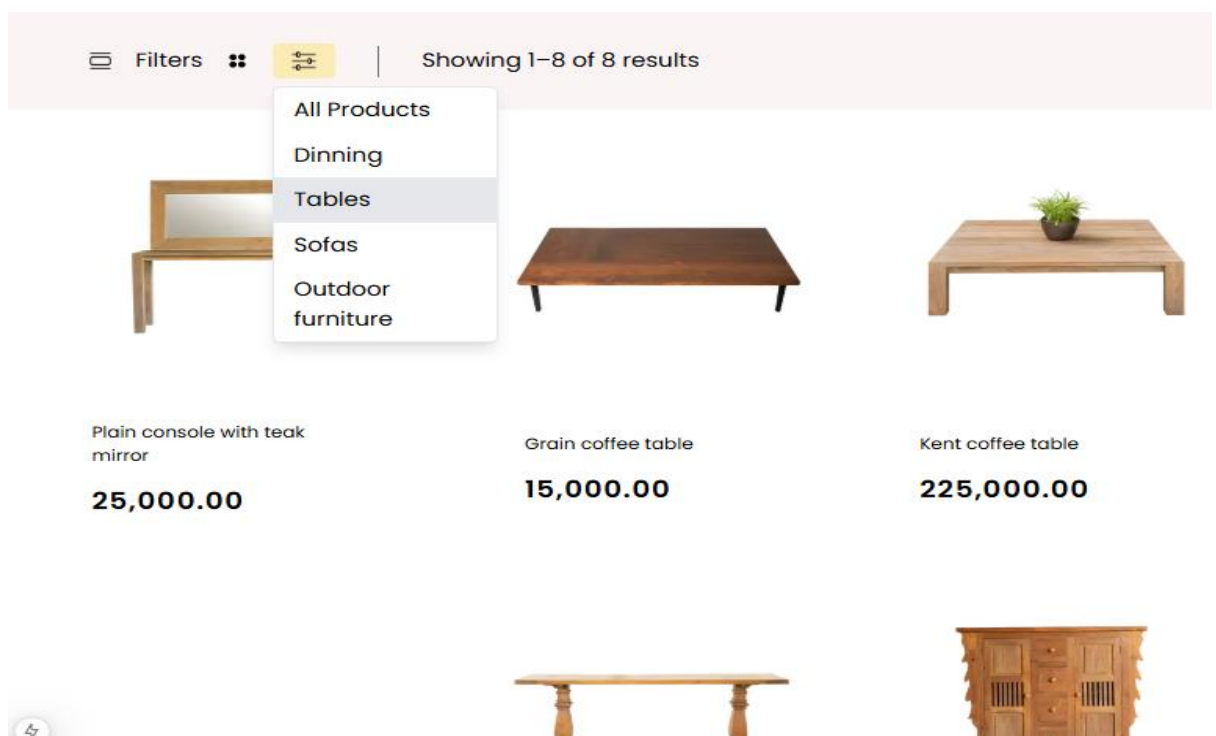
  // Check if products for this category are already fetched
  if (categoryProducts[categoryTitle]) {
    setProducts(categoryProducts[categoryTitle]); // Use already fetched products
    return;
  }
}
```

```
</button>
</button>
{isDropdownOpen && (
  <div className="absolute bg-white border rounded-md shadow-lg mt-2 w-48">
    <button
      onClick={() => {
        setProducts(allProducts); // Show all products when "All Products" is clicked
        setCategoryProducts({});
        setIsDropdownOpen(false); // Close dropdown
        setCurrentPage(1); // Reset to page 1
      }}
      className="w-full text-left px-4 py-2 hover:bg-gray-200"
    >
      All Products
    </button>
    {categories.map((category) => (
      <button
        key={category._id}
        onClick={() => {
          handleCategoryClick(category.title);
          setIsDropdownOpen(false); // Close dropdown after selection
          setCurrentPage(1); // Reset to page 1
        }}
        className="w-full text-left px-4 py-2 hover:bg-gray-200"
      >
        {category.title}
      </button>
    ))}
  </div>
)}
</div>
</div>
```

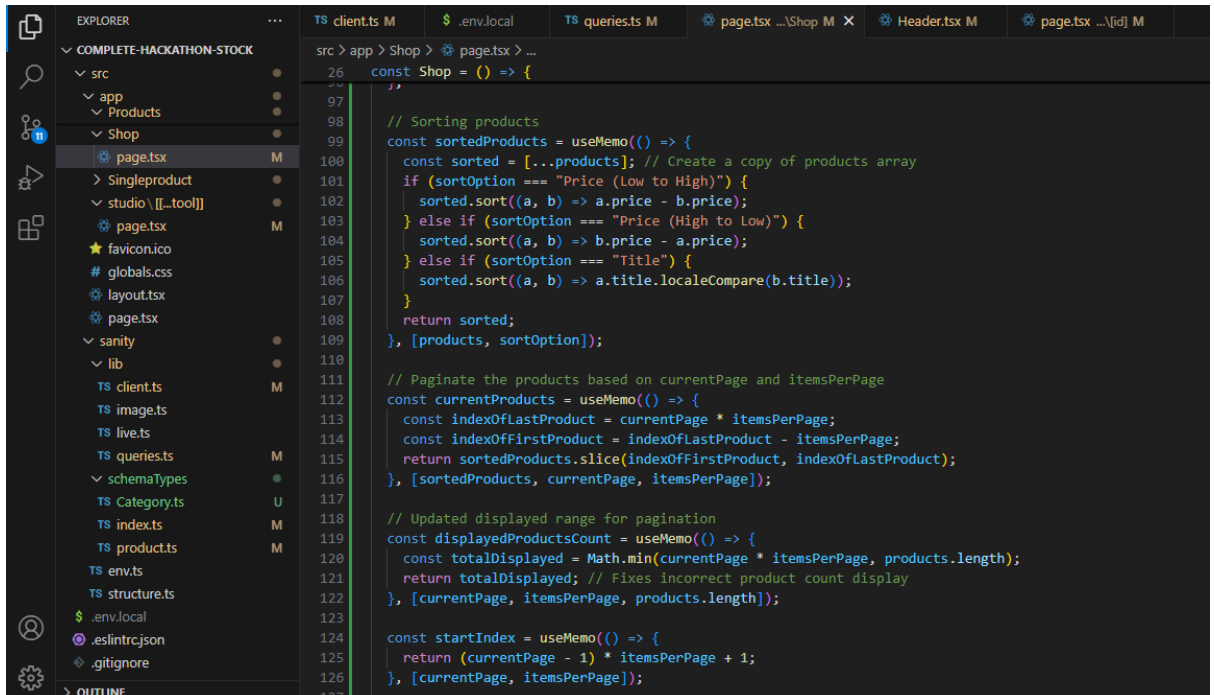
And this is my uiux design there i implement this drop down category functionality



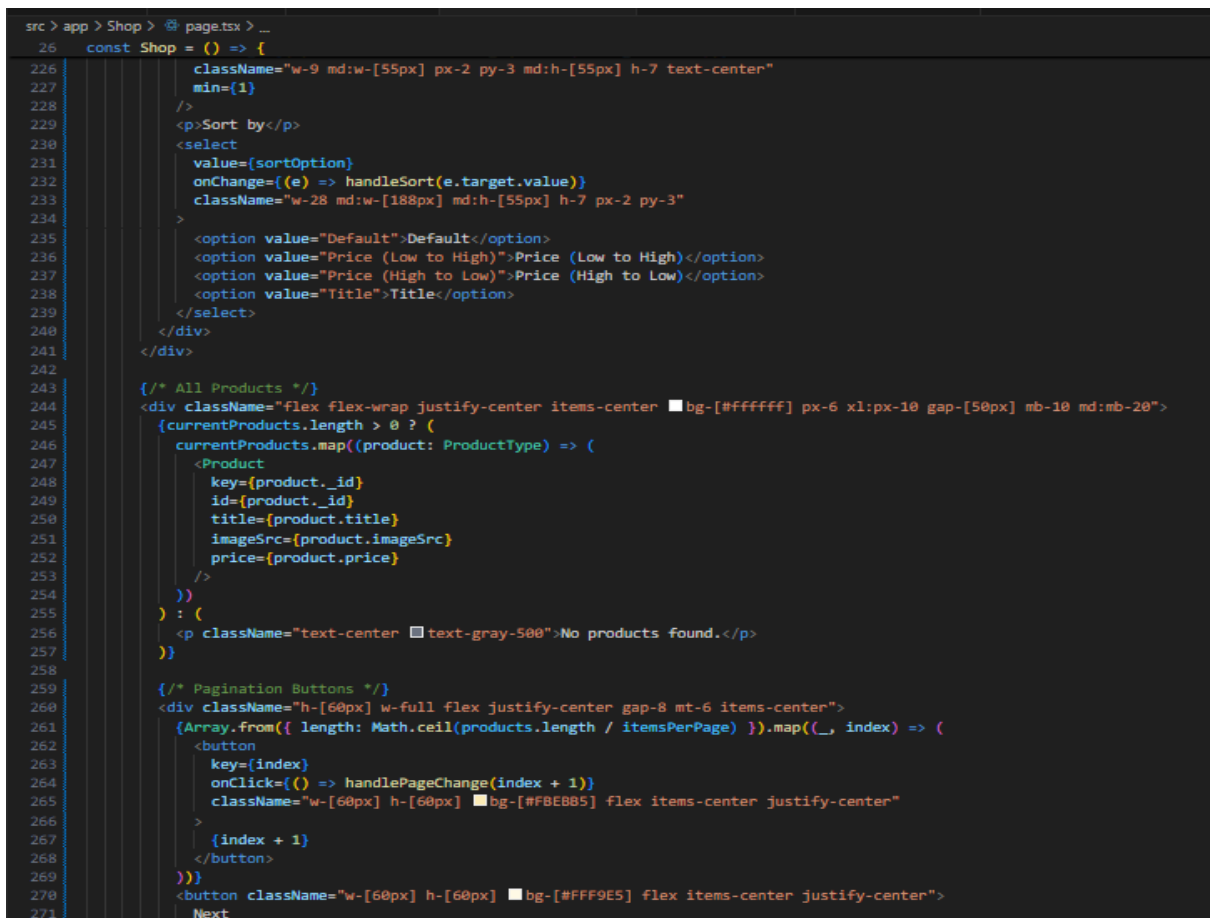
If i will select tables category then only tables will show on my shop page



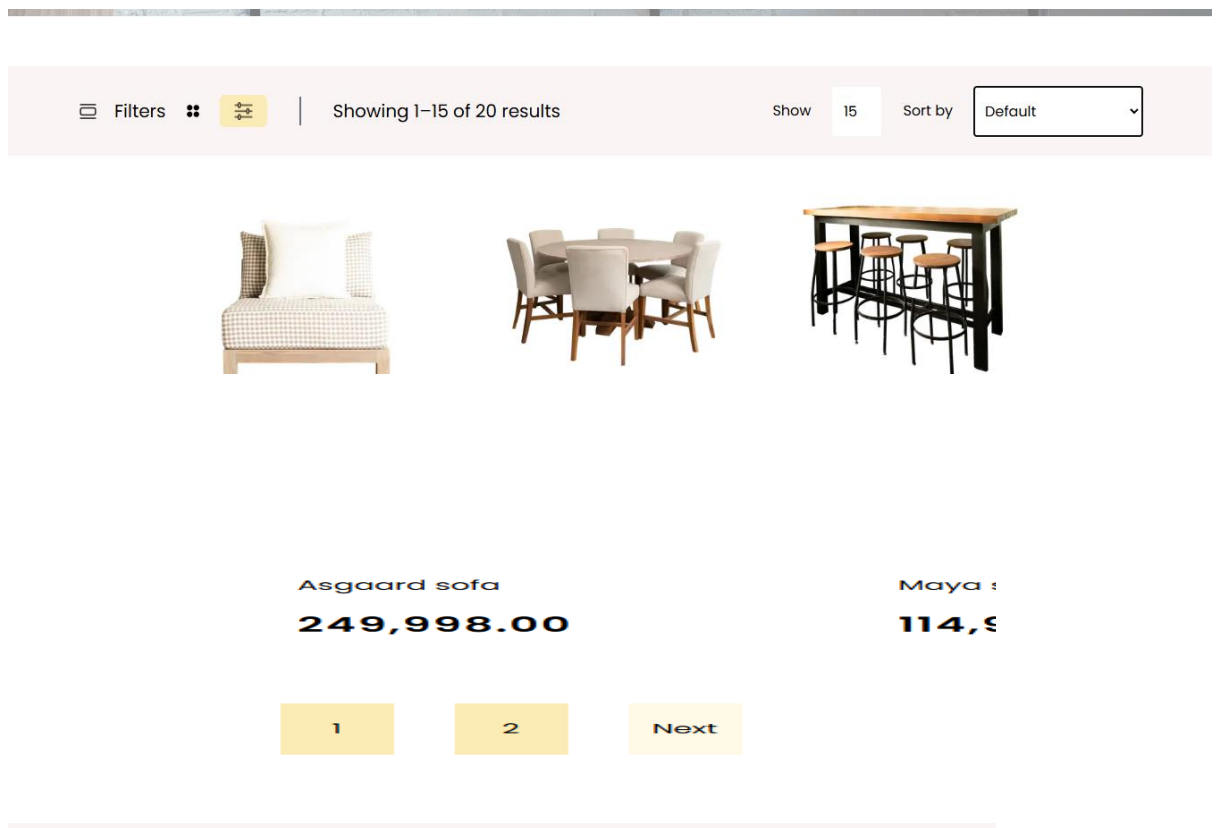
And i add filters functionality, pagination and all functionalities that are required for a shop page



```
src > app > Shop > page.tsx > ...
26 const Shop = () => {
27   // Sorting products
28   const sortedProducts = useMemo(() => {
29     const sorted = [...products]; // Create a copy of products array
30     if (sortOption === "Price (Low to High)") {
31       sorted.sort((a, b) => a.price - b.price);
32     } else if (sortOption === "Price (High to Low)") {
33       sorted.sort((a, b) => b.price - a.price);
34     } else if (sortOption === "Title") {
35       sorted.sort((a, b) => a.title.localeCompare(b.title));
36     }
37     return sorted;
38   }, [products, sortOption]);
39
40   // Paginate the products based on currentPage and itemsPerPage
41   const currentProducts = useMemo(() => {
42     const indexOffLastProduct = currentPage * itemsPerPage;
43     const indexOffFirstProduct = indexOffLastProduct - itemsPerPage;
44     return sortedProducts.slice(indexOffFirstProduct, indexOffLastProduct);
45   }, [sortedProducts, currentPage, itemsPerPage]);
46
47   // Updated displayed range for pagination
48   const displayedProductsCount = useMemo(() => {
49     const totalDisplayed = Math.min(currentPage * itemsPerPage, products.length);
50     return totalDisplayed; // Fixes incorrect product count display
51   }, [currentPage, itemsPerPage, products.length]);
52
53   const startIndex = useMemo(() => {
54     return (currentPage - 1) * itemsPerPage + 1;
55   }, [currentPage, itemsPerPage]);
56 }
```



```
src > app > Shop > page.tsx > ...
26 const Shop = () => {
27   // JSX for the shop page
28   return (
29     <div className="w-9 md:w-[55px] px-2 py-3 md:h-[55px] h-7 text-center" min={1}>
30       <p>Sort by</p>
31       <select
32         value={sortOption}
33         onChange={(e) => handleSort(e.target.value)}
34         className="w-28 md:w-[188px] md:h-[55px] h-7 px-2 py-3">
35         <option value="Default">Default</option>
36         <option value="Price (Low to High)">Price (Low to High)</option>
37         <option value="Price (High to Low)">Price (High to Low)</option>
38         <option value="Title">Title</option>
39       </select>
40     </div>
41
42     <div>
43       <div className="flex flex-wrap justify-center items-center bg-[#ffffff] px-6 xl:px-10 gap-[50px] mb-10 md:mb-20">
44         {currentProducts.length > 0 ? (
45           currentProducts.map((product: ProductType) => (
46             <Product
47               key={product._id}
48               id={product._id}
49               title={product.title}
50               imageSrc={product.imageSrc}
51               price={product.price}
52             />
53           ))
54         ) : (
55           <p className="text-center text-gray-500">No products found.</p>
56         )}
57       </div>
58
59       <div className="h-[60px] w-full flex justify-center gap-8 mt-6 items-center">
60         {Array.from({ length: Math.ceil(products.length / itemsPerPage) }).map((_, index) => (
61           <button
62             key={index}
63             onClick={() => handlePageChange(index + 1)}
64             className="w-[60px] h-[60px] bg-[#FBE885] flex items-center justify-center"
65             >
66             {index + 1}
67           </button>
68         ))}
69         <button className="w-[60px] h-[60px] bg-[#FFF9E5] flex items-center justify-center">
70           Next
71         </button>
72       </div>
73     </div>
74   );
75 }
```



and after complete all these shop functionalities i implement dynamic functionality on my single product page for colours, size, SKU, tags etc

Asgaard sofa

Rs. 249998



5 Customer Review

A comfortable and contemporary Asgaard sofa.

Size

M XL

Colour



- 1 +

Add to Cart

SKU : OD002

Category : Outdoor furniture

Tags : Outdoor Furniture waterproof sofas Outdoor sofas

Share :

This is my schema for colours, size, tags etc

```
src > sanity > schemaTypes > TS products > product > fields
3   export const product = {
7     fields: [
63       // Optional: Set this to true if size is mandatory
64       validation: (Rule : Rule) => Rule.required()
65     ],
66     {
67       name: 'sku',
68       title: 'SKU',
69       type: 'string',
70     },
71   },
72   {
73     name: 'discountPercentage',
74     title: 'Discount Percentage',
75     type: 'number',
76   },
77   {
78     name: 'isFeaturedProduct',
79     title: 'Is Featured Product',
80     type: 'boolean',
81   },
82   {
83     name: 'colors',
84     type: 'array',
85     title: 'Colors',
86     of: [{ type: 'string' }],
87     options: {
88       layout: 'tags', // Allow adding multiple colors
89     },
90   },
91   {
92     name: 'stockLevel',
93     title: 'Stock Level',
94     type: 'number',
95   },
96   {
97     name: 'category',
98     title: 'Category',
99     type: 'reference', // Use reference for better category management
100    to: [{ type: 'category' }],
101  },
102  ],
103  name: 'tags',
104  title: 'Tags',
105  type: 'array',
106  of: [{ type: 'string' }],
107  options: {
108    layout: 'tags', // Allow adding multiple colors
109  },
110  ],
}
```

And this is my single product page code there i implement this

```
<div className="flex gap-5 my-5">
  {size?.map((s, index) => (
    <div
      key={index}
      className="w-[30px] h-[30px] lg:w-[40px] lg:h-[40px] bg-[#FBE8B5] rounded"
    >
      {s}
    </div>
  ))}
</div>
<p className="text-[#9F9F9F] mt-3">Colour</p>

<div className="flex gap-5 my-5">
  {colors?.length > 0 ? (
    colors.map((color, index) => (
      <div
        key={index}
        style={{ backgroundColor: color }}
        className="w-[30px] h-[30px] rounded-full"
      ></div>
    ))
  ) : (
    <p>No colors available</p>
  )}
</div>
```

🔗 **Category Schema and Linking:**

- Created a category schema in Sanity and linked it to the product schema to categorize products (e.g., "Tables" for dining tables, "Sofas" for sofas).

🔗 **Category Filtering:**

- Implemented a category filter on the shop page, allowing users to view products based on the selected category (e.g., only displaying tables when the "Tables" category is selected).
- Added a dropdown UI component to enable users to filter products dynamically by category.

🔗 **Dynamic Attributes for Products:**

- Developed dynamic functionality on the single product page to handle various product attributes like:
 - **Colors:** Users can select from available colors.
 - **Sizes:** Size options are displayed and selectable.
 - **SKU (Stock Keeping Unit):** SKU details for each product.
 - **Tags:** Tags representing different product features or categories (e.g., "Feature products," "Discount").

🔗 **Schema for Product Attributes:**

- Created individual schemas for product attributes such as color, size, tags, etc., in Sanity.
- Linked these attributes to the product schema to dynamically display them on the product detail page.

🔗 **Pagination and Additional Functionalities:**

- Implemented pagination on the shop page to manage large product lists effectively.

- Enabled dynamic functionality for product attributes to reflect the color, size, and tag selections on the single product page.

