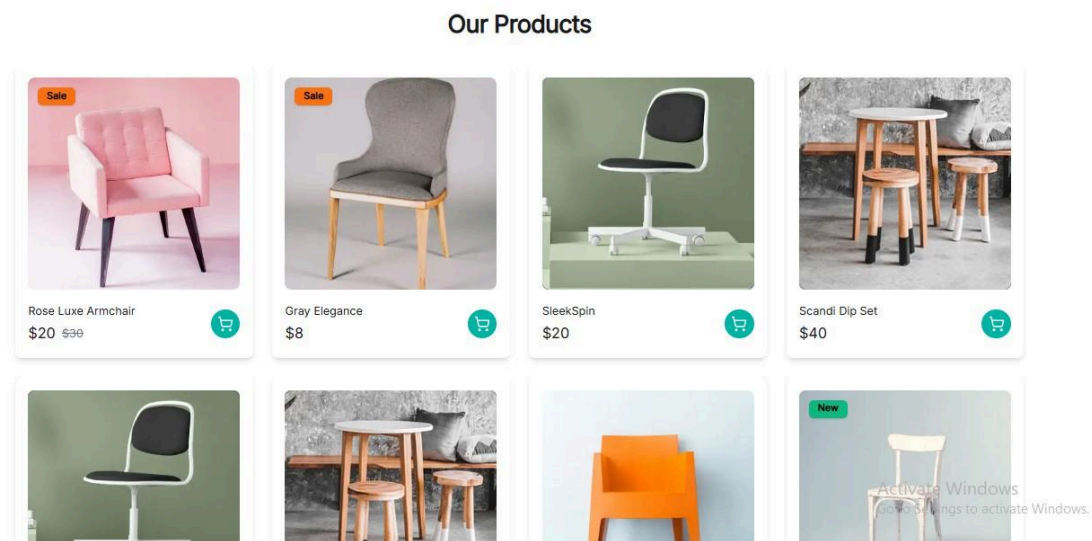


# Day 4: Dynamic Frontend Components - Comforty Chair Marketplace

## Functional Deliverables

### 1. Product Listing Page with Dynamic Data

- The product listing page is fully functional and dynamically fetches product data from Sanity CMS or APIs.
- **Features:**
  - Products are displayed with essential details such as image, title, price, and rating.
  - Data is rendered dynamically to ensure the page updates whenever new products are added to the CMS.



### 2. Individual Product Detail

- Each product detail page is implemented using dynamic routing (`/products/[id]`).
- Displays accurate product details fetched dynamically based on the product ID.

Create by: [Sarwat Samson](#)

- **Features:**

- o High-quality product images.
- o Title, description, price, rating, and inventory status.
- o Add-to-cart functionality.
- o Related products section to suggest similar items.



## Rose Luxe Armchair

\$20.00 USD ~~\$30.00 USD~~

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

 Add To Cart

Activate Windows  
Go to Settings to activate Windows

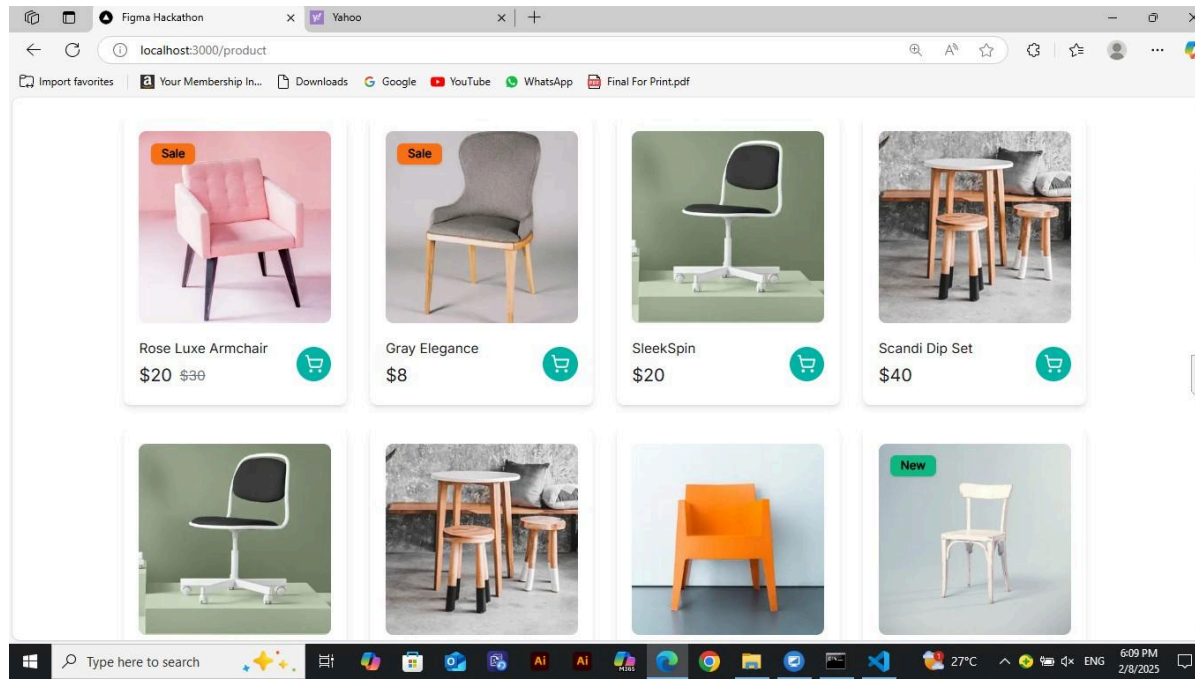
[View All](#)

### FEATURED PRODUCTS

### 3. Advanced Category Filters

- Implemented category filters to refine product views dynamically.
- Users can filter products by:
  - o Categories (e.g., chairs, tables, sofas).
  - o Price range.
  - o Ratings.

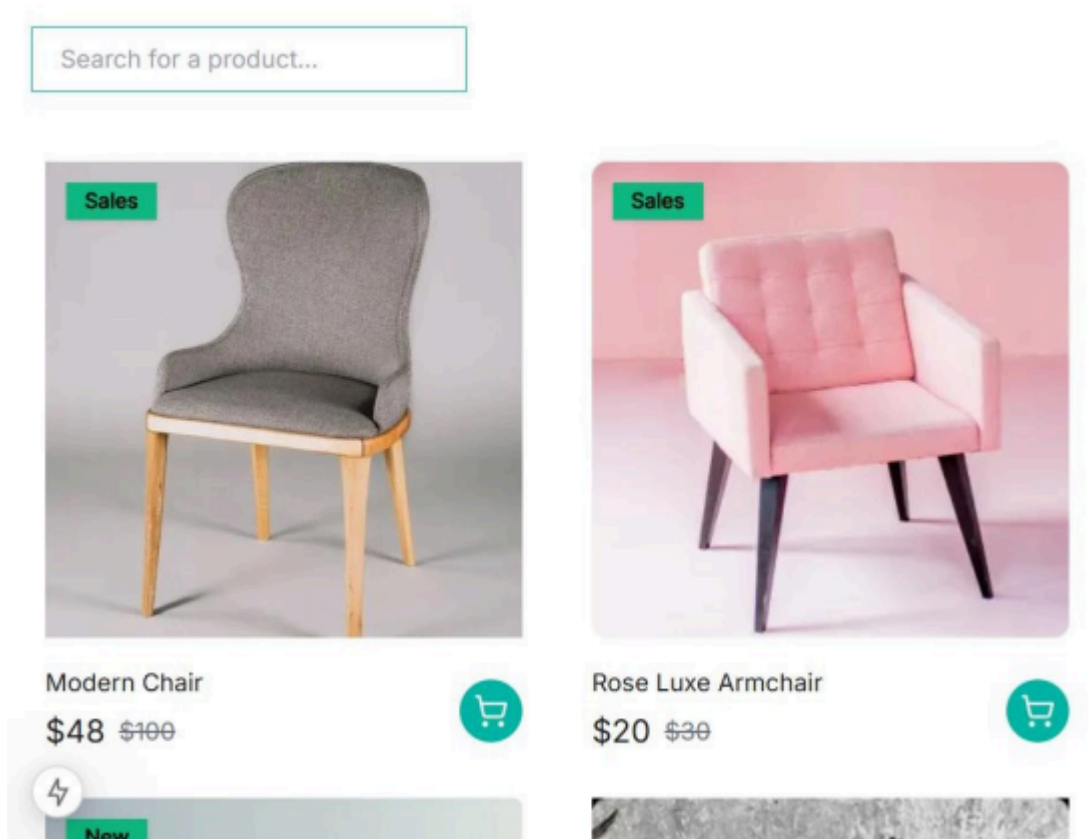
Create by: [Sarwat Samson](#)



#### 4. Search Bar

- A search bar is integrated to filter products by name or tags.
- Features:
  - o Real-time search results.
  - o Case-insensitive search functionality.

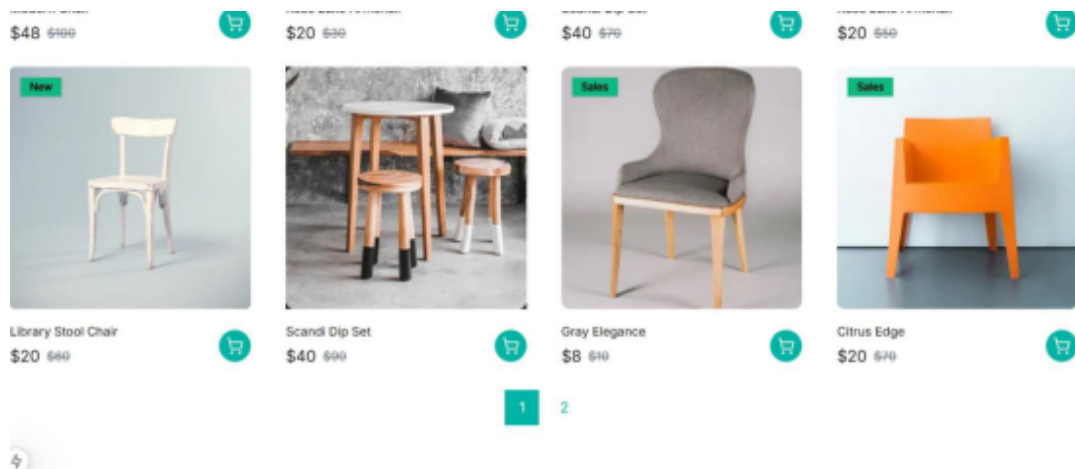
Create by: [Sarwat Samson](#)



### 5. Additional Features

- Pagination:
  - Implemented pagination to load products efficiently, reducing load times for large datasets.
- Related Products:

Displayed on the product detail page to encourage further exploration and purchases.



## 6. Responsive Styling

- All components are styled for responsiveness, ensuring a professional look across devices (mobile, tablet, desktop).
- Used CSS frameworks and custom classes for a polished UI.

## Code Deliverables

### Key Components:

#### Filter & Categories

```

/* Filter Dropdowns */


<select
    value={filter}
    onChange={(e) => setFilter(e.target.value)}
    className="rounded-lg border border-[#0005A5] py-2 px-4 text-sm text-[#0005A5] focus:outline-none focus:ring-2 focus:ring-[#0005A5]"
  >
    <option value="default">Sort By:</option>
    <option value="lowToHigh">Price: Low to High</option>
    <option value="highToLow">Price: High to Low</option>
    <option value="new">New Arrivals</option>
    <option value="sale">On Sale</option>
  </select>

  <select
    value={categoryFilter}
    onChange={(e) => setCategoryFilter(e.target.value)}
    className="rounded-lg border border-[#0005A5] py-2 px-4 text-sm text-[#0005A5] focus:outline-none focus:ring-2 focus:ring-[#0005A5]"
  >
    <option value="default">All Categories</option>
    {categories.map((category) => (
      <option key={category._id} value={category.title}>
        {category.title}
      </option>
    ))}
  </select>
</div>


```

Create by: [Sarwat Samson](#)

## SearchBar

```
/* Search and Filter Section */
<div className="flex flex-col sm:flex-row items-center justify-between gap-4 mb-10">
  { /* Search Bar */ }
  <div className="relative max-w-screen-lg mx-auto w-full">
    <input
      type="text"
      placeholder="Search for a product..."
      value={search}
      onChange={(e) => setSearch(e.target.value)}
      className="w-[250px] rounded-lg border border-[#0005A5] py:2 px:4 text-sm text-gray-700 focus:outline-none focus:ring-2 focus:ring-[#0005A5] shadow-sm"
    />
  </div>
</div>
```

## API Integration Logic

```
useEffect(() => {
  const fetchProduct = async () => {
    try {
      const query = `*_type == "products" && _id == ${id}][0] {
        _id,
        title,
        price,
        priceWithoutDiscount,
        "category": category->{
          _id,
          title
        },
        "imageUrl": image.asset->url,
        description,
        inventory,
        tags,
        badge,
        rating
      }`;

      const response = await client.fetch(query, { id });

      if (!response) {
        setError("Product not found!");
      } else {
        setProduct(response);
      }
    } catch (err) {
      setError("Error fetching product data.");
    } finally {
      setLoading(false);
    }
  }
}, [id]);
```

Create by: [Sarwat Samson](#)

## **Documentation**

### **Steps Taken to Build and Integrate Component**

#### **1. Dynamic Product Listing Page:**

- Created a AllProducts.tsx component to fetch and render products dynamically from Sanity CMS.

#### **2. Dynamic Routing for Product Details:**

- Used Next.js dynamic routes ([id].tsx) to implement individual product pages.

#### **3. Category Filters and Search Bar:**

- Built reusable components (CategoryFilter and SearchBar) with state management.

#### **4. Pagination and Related Products:**

- Implemented logic to load products in chunks and display related products based on categories or tags.

#### **5. Styling and Responsiveness:**

- Ensured UI components are responsive using Tailwind CSS and media queries.

Create by: [Sarwat Samson](#)

## **Challenges Faced and Solutions Implemented**

### **1. Fetching Data Dynamically:**

- Challenge: Ensuring smooth data fetching without delays.
- Solution: Used asynchronous functions and proper error handling.

### **2. Search and Filter Optimization:**

- Challenge: Real-time filtering without performance lag.
- Solution: Used debouncing to optimize search queries.

### **3. Dynamic Routing:**

- Challenge: Handling non-existent product IDs.
- Solution: Added fallback UI for error handling.

## **Best Practices Followed:**

- Modular and reusable component design.
- Clean and maintainable code with proper comments.
- Efficient data fetching with error handling.
- Responsive design principles for cross-device compatibility.

Create by: [Sarwat Samson](#)





