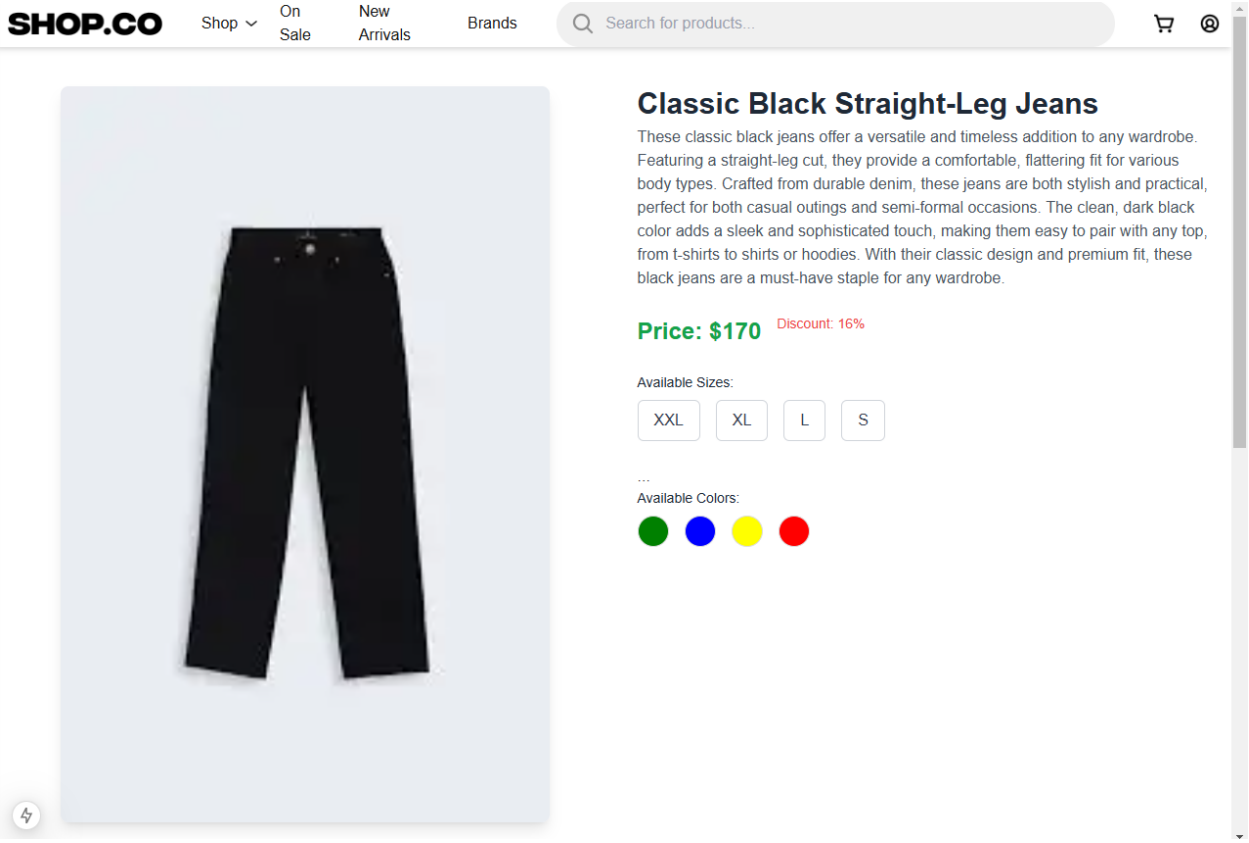
Day 4 - Dynamic Frontend Components - [SHOP.CO]

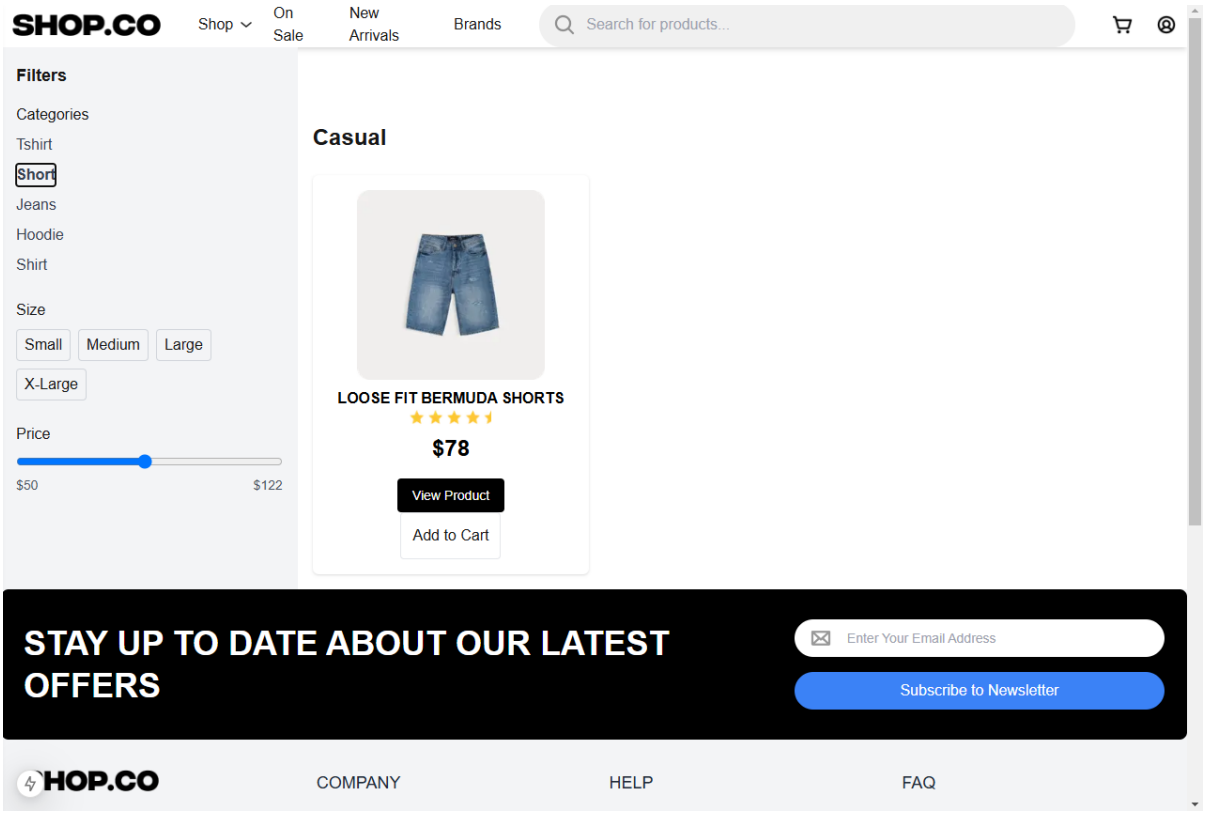
1. Functional Deliverables

Screenshots or Screen Recordings

- **Product Listing Page with Dynamic Data**
- **Individual Product Detail Pages**:



- **Working Category Filters, Search Bar, and Pagination**:

- 

Additional Features:

2. Code Deliverables

Key Components Code Snippets:

- **ProductList Component**:

```jsx
export default function Products() {

  return (
    <div className="lg:w-full w-full pt-[150px]">
      {/* New Arrivals Section */}
      <div className="lg:w-full w-full pt-[150px]">
        <div className="lg:w-[403px] w-[269px] h-[38px] lg:h-[58px] flex absolute lg:top-[991px] top-[1147px]
          NEW ARRIVALS
        </div>
        {/* Product Container */}
        <div className="grid lg:grid-cols-4 md:grid-cols-3 sm:grid-cols-2 grid-cols-1 gap-6 mt-8 justify-cente
          {products.map((product) => (
            <div
              key={product._id}
              className="flex flex-col items-center bg-white p-4 shadow rounded-md"
            >
              {product.image && (
                <Image
                  src={urlFor(product.image).url()}
                  alt={product.name}
                  width={200}
                  height={200}
                  className="rounded-lg"
                />
              )}
              <div className="font-bold text-black mt-2">{product.name}</div>

              <p className="font-satoshi font-bold text-[24px] leading-[32.4px] text-black mt-2">
                ${product.price}
              </p>

              <Link key={product.slugCurrent} href={`/products/${product.slugCurrent}`}>
                <button className="mt-4 bg-black text-white text-sm font-medium py-2 px-4 rounded hover:bg-gra
                  View Product
                </button>
              </Link>
              <button
              onClick={() => handleClick(product)}
              className='p-3 border rounded'
            >
              Add to Cart
            </button>
```

API Integration and Dynamic Routing:

- **API Integration**: We use `fetch` to connect to Sanity CMS for retrieving product data dynamically.

- **Dynamic Routing**: Routes like `/product/[slug]` are dynamically generated using Next.js's file-based routing.

3. Documentation

Steps Taken to Build and Integrate Components:

1. **Product List**:

The `ProductList` component was created to dynamically fetch and render products from the Sanity CMS API. We used `useEffect` to fetch data on component mount and stored it in a `useState` hook.

2. **Product Detail Page**:

Dynamic routing was set up to show detailed information for each product. We used Next.js's `getServerSideProps` to fetch data for individual product pages based on the product ID.

3. **Search and Filters**:

Implemented a `SearchBar` component and category filters that allow users to search and filter the product list by category. This enhances the user experience by making it easier to find products.

4. **Pagination**:

Pagination was implemented to handle large product lists efficiently. We used basic pagination logic with page numbers and adjusted the API call accordingly.

Challenges Faced and Solutions Implemented:

- **Challenge 1**: Struggled with dynamic data rendering from Sanity API due to incorrect API keys.

Solution: Double-checked API configurations in the Sanity dashboard, ensuring correct keys and dataset were used.

- **Challenge 2**: Difficulty with dynamic routing due to missing or incorrect product IDs.

Solution: Used placeholder data for testing routes and confirmed dynamic routing worked before integrating live data.

Best Practices Followed During Development:

1. **Component Reusability**: Focused on building modular, reusable components like `ProductCard` and `ProductList` to reduce duplication and improve maintainability.
2. **Separation of Concerns**: Kept logic for data fetching and UI rendering separate for better clarity and testing.
3. **API Error Handling**: Included error handling in API calls to ensure smooth user experience in case of failures.