Day 4 - Dynamic Frontend Components - [E-Commerce Furniture Website]

Introduction:

Today, I focused on building dynamic frontend components for my e-commerce furniture website using **Sanity** for data fetching. All data was manually entered into Sanity and seamlessly integrated into the project. My goal was to create a visually appealing and fully functional user experience with features like a product listing page, individual product detail pages, a Wishlist, a reviews section, a checkout page, and a search bar.

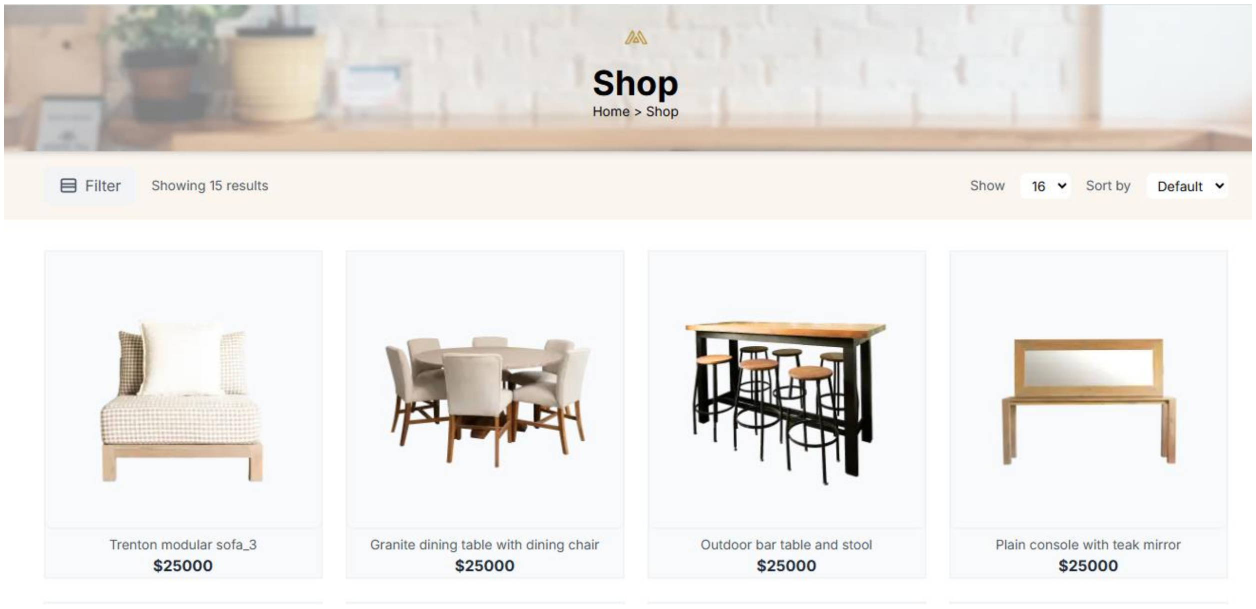
My Experience:

1. Product Listing Page with Dynamic Data

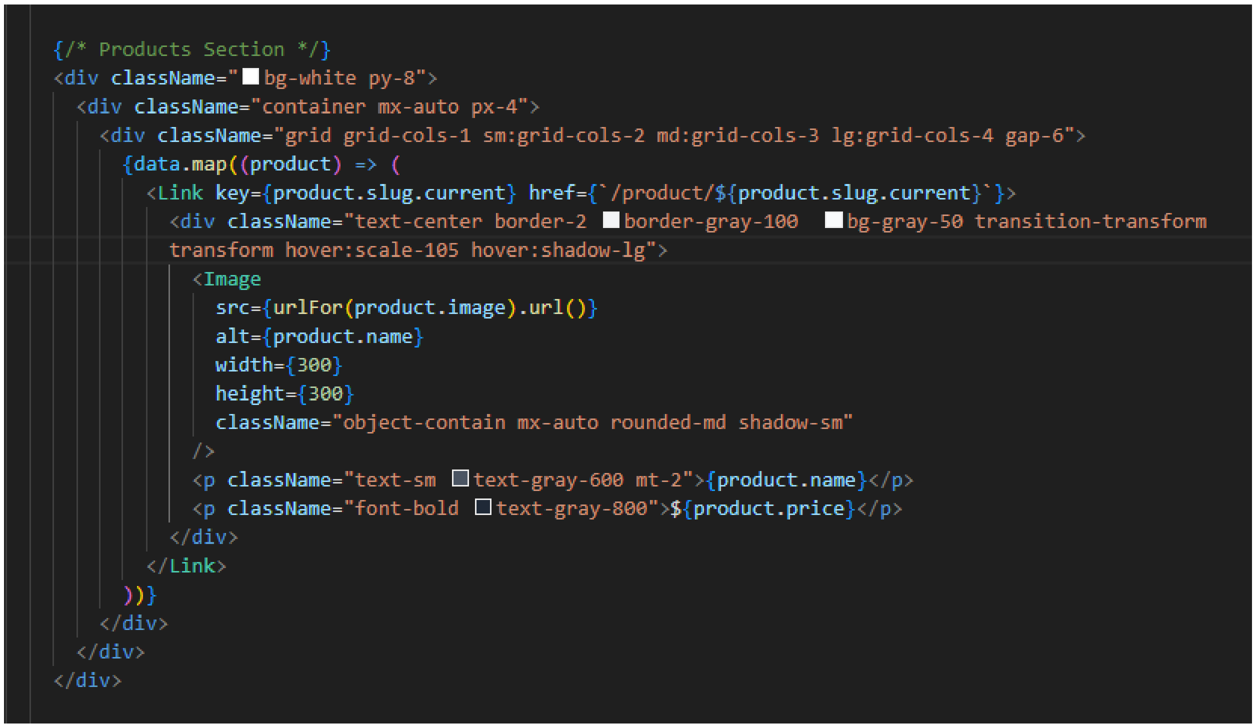
I began by creating the product listing page. Using **Sanity**, I fetched all the furniture products dynamically and displayed them with essential details like name, price, and image.

- The design is responsive and user-friendly.
- Each product is rendered dynamically from the Sanity dataset.

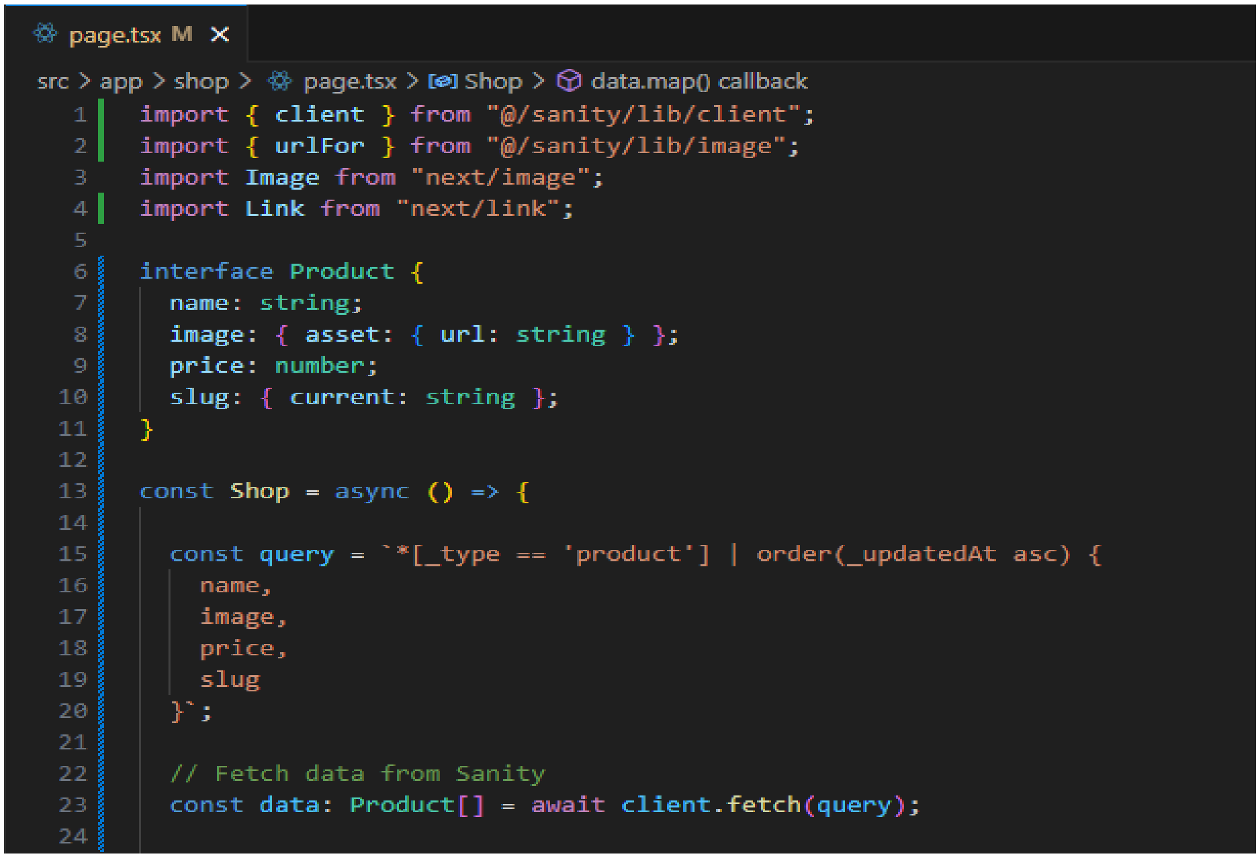
What I learned:
Manually managing and fetching data using Sanity gave me better control over the structure and flow of information.



Here's how I fetched the data for products and use it by using map function.



```
{/* Products Section */}
<div className="bg-white py-8">
  <div className="container mx-auto px-4">
    <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
      {data.map((product) => (
        <Link key={product.slug.current} href={`/product/${product.slug.current}`}>
          <div className="text-center border-2 border-gray-100 bg-gray-50 transition-transform
          transform hover:scale-105 hover:shadow-lg">
            <Image
              src={urlFor(product.image).url()}
              alt={product.name}
              width={300}
              height={300}
              className="object-contain mx-auto rounded-md shadow-sm"
            />
            <p className="text-sm text-gray-600 mt-2">{product.name}</p>
            <p className="font-bold text-gray-800">${product.price}</p>
          </div>
        </Link>
      ))}
    </div>
  </div>
</div>
```



page.tsx M ×

src > app > shop > page.tsx > Shop > data.map() callback

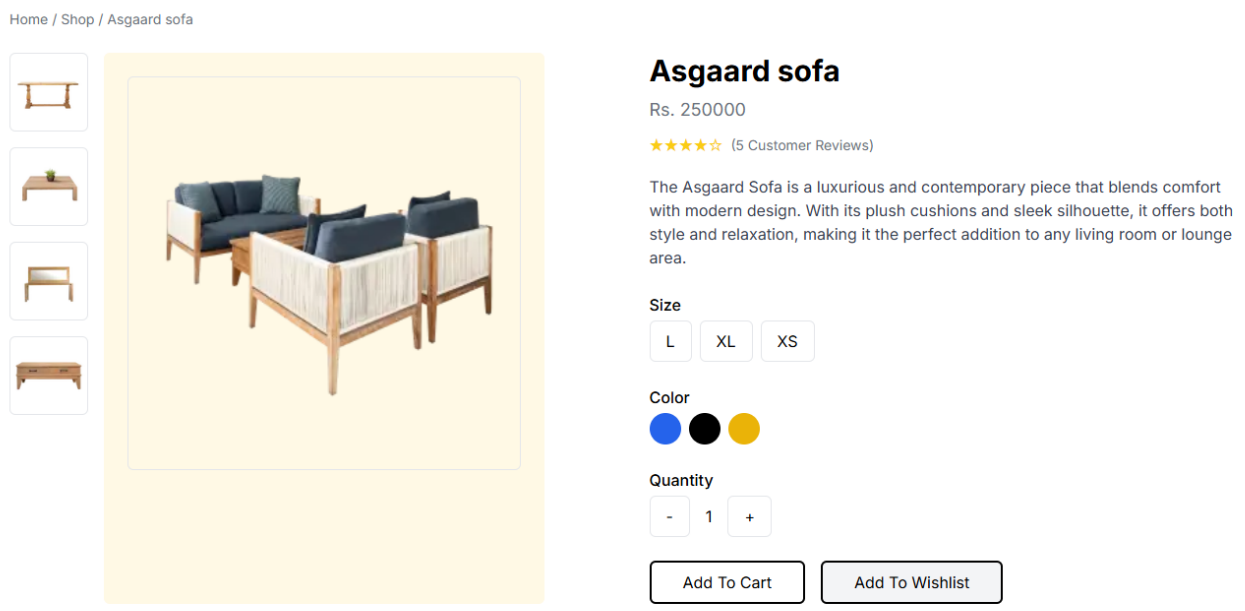
```
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";

interface Product {
  name: string;
  image: { asset: { url: string } };
  price: number;
  slug: { current: string };
}

const Shop = async () => {

  const query = `*[_type == 'product'] | order(_updatedAt asc) {
    name,
    image,
    price,
    slug
  }`;

  // Fetch data from Sanity
  const data: Product[] = await client.fetch(query);

```

2. Individual Product Detail Pages

The next step was developing detailed pages for each product. These pages provide all necessary information, ensuring customers can make informed decisions.

- Each page includes product specifications, pricing, and detailed descriptions.
- I added a **Reviews Section**, where users can view feedback about the product.

What I learned:
Creating a reviews section allowed me to explore the concept of rendering additional dynamic data within the product page, further enriching the user experience.


Home / Shop / Asgaard sofa

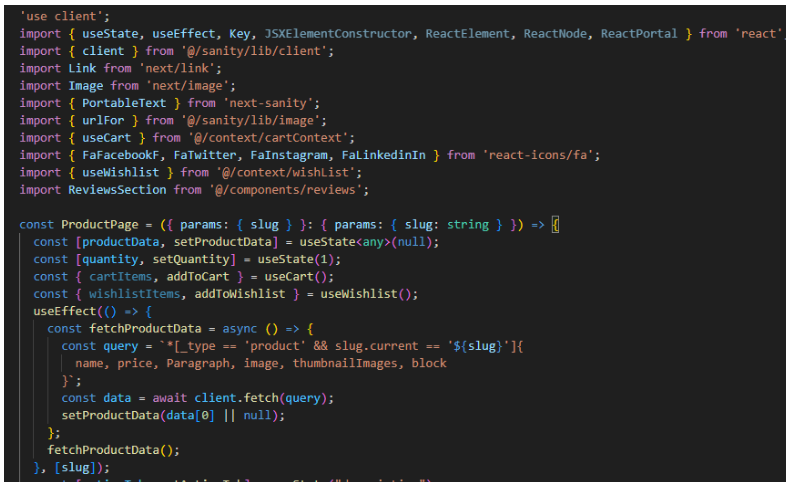
Asgaard sofa

Rs. 250000

★★★★☆ (5 Customer Reviews)

The Asgaard Sofa is a luxurious and contemporary piece that blends comfort with modern design. With its plush cushions and sleek silhouette, it offers both style and relaxation, making it the perfect addition to any living room or lounge area.

Here's how I fetched data for detailed product page.

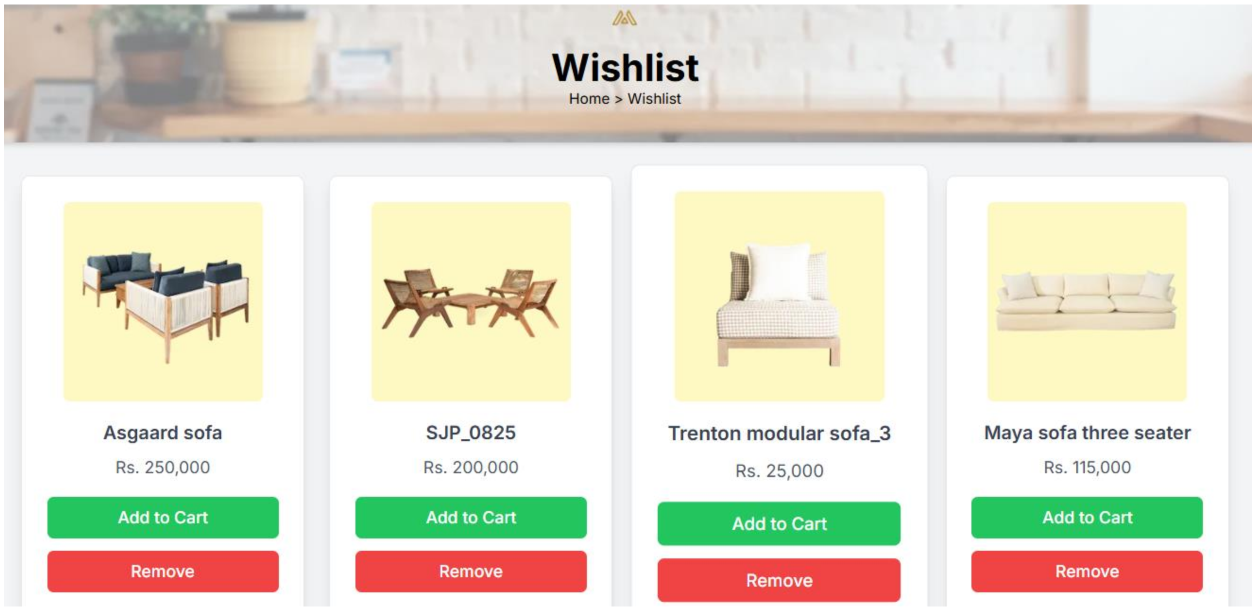

```
'use client';
import { useState, useEffect, Key, JSXElementConstructor, ReactElement, ReactNode, ReactPortal } from 'react'
import { client } from '@/sanity/lib/client';
import Link from 'next/link';
import Image from 'next/image';
import { PortableText } from 'next-sanity';
import { urlFor } from '@/sanity/lib/image';
import { useCart } from '@/context/cartContext';
import { FaFacebookF, FaTwitter, FaInstagram, FaLinkedinIn } from 'react-icons/fa';
import { useWishlist } from '@/context/wishList';
import ReviewsSection from '@/components/reviews';

const ProductPage = ({ params: { slug } }: { params: { slug: string } }) => {
  const [productData, setProductData] = useState<any>(null);
  const [quantity, setQuantity] = useState(1);
  const { cartItems, addToCart } = useCart();
  const { wishlistItems, addToWishlist } = useWishlist();
  useEffect(() => {
    const fetchProductData = async () => {
      const query = `*[_type == 'product' && slug.current == '${slug}'][{
        name, price, Paragraph, image, thumbnailImages, block
      }`;
      const data = await client.fetch(query);
      setProductData(data[0] || null);
    };
    fetchProductData();
  }, [slug]);
```

3. Wishlist Page

I implemented a **Wishlist Page**, where users can save their favorite products for later viewing.

- Users can add or remove items from the Wishlist dynamically.
- The page updates in real-time, reflecting the current state of the Wishlist.

Challenges I faced:
One of the challenges was ensuring that the Wishlist functionality worked across all devices. By testing and debugging, I ensured it performed as expected.



4. Checkout Page

For a seamless shopping experience, I developed a **Checkout Page** where users can view their selected items and proceed to finalize their orders.

- I implemented form validation to ensure accurate user inputs.
- The page is designed to be straightforward and easy to navigate.

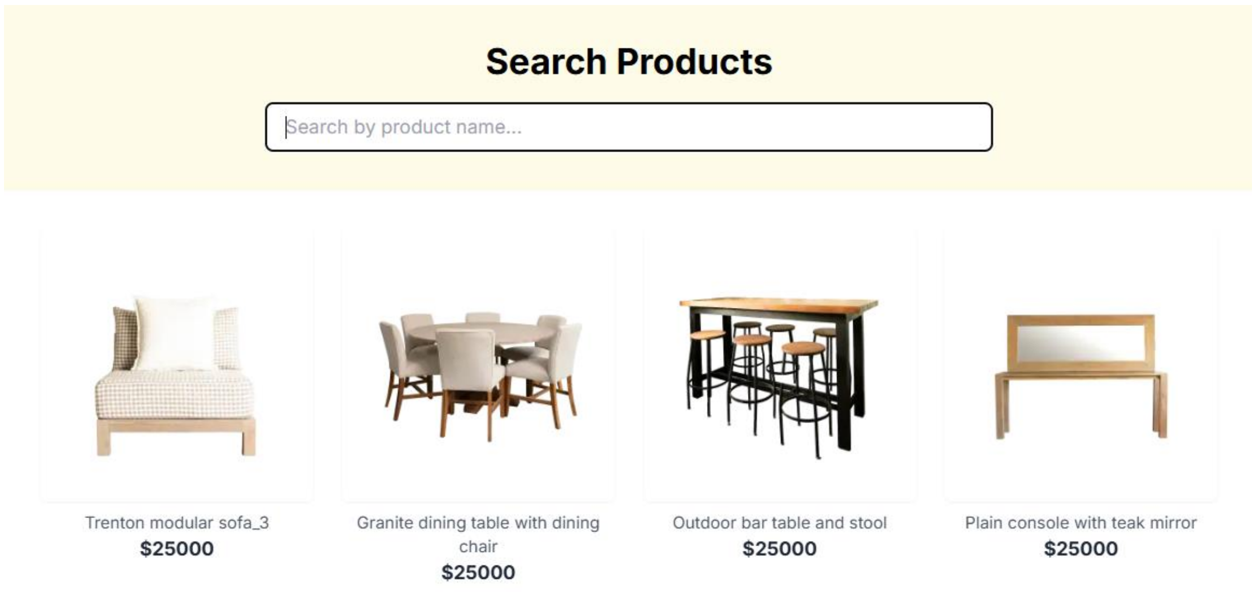
What I learned:
Building the checkout page helped me understand how critical user interaction is during this step. A clean, responsive design and smooth functionality are essential to retain customers.

5. Search Bar

To make it easier for users to find specific products, I implemented a **Search Bar** with real-time search capabilities.

- As users' type, matching results are displayed instantly.
- The feature is optimized for both desktop and mobile devices.

What I enjoyed:
I loved seeing the search bar function in real time. It added a professional touch to the website, and the instant feedback made the experience more interactive.



Search Products

Search by product name...

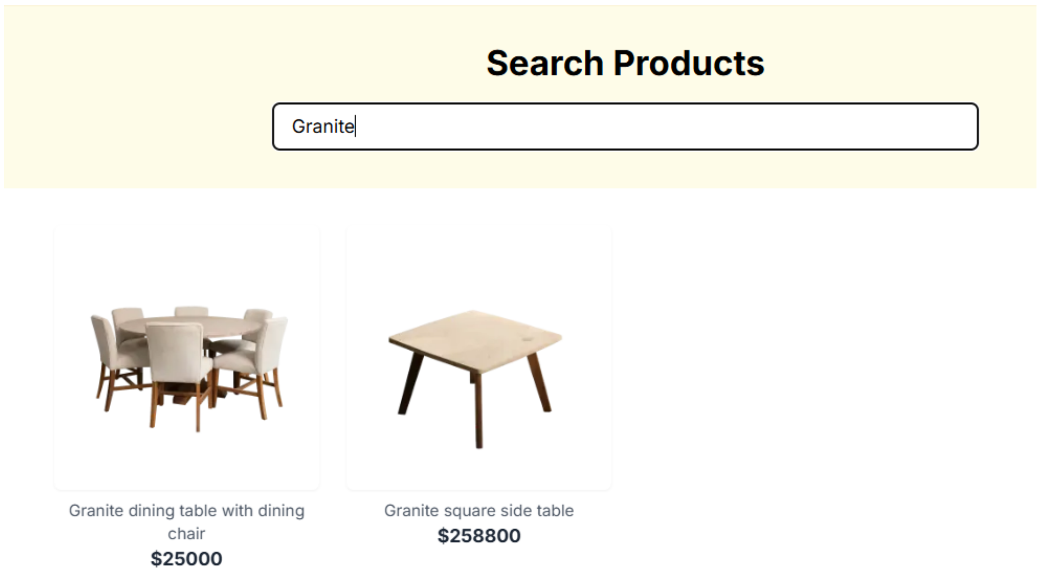
Trenton modular sofa_3
$25000

Granite dining table with dining chair
$25000

Outdoor bar table and stool
$25000

Plain console with teak mirror
$25000

Search functionality implemented perfectly.



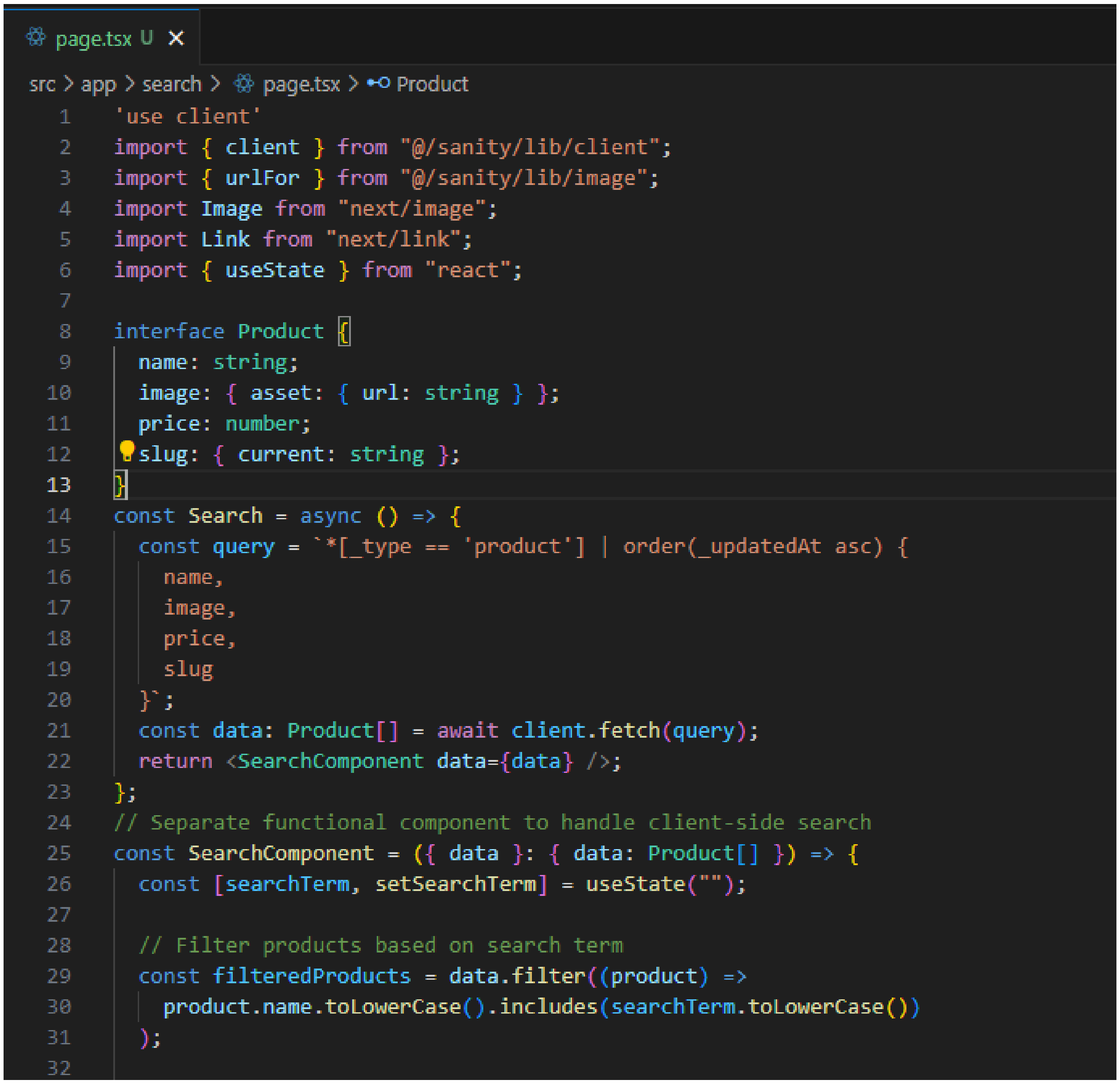
Search Products

Granite

Granite dining table with dining chair
$25000

Granite square side table
$258800

Search Functionality….



page.tsx U ×

src > app > search > page.tsx > Product

```tsx
'use client'
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";
import { useState } from "react";

interface Product {
  name: string;
  image: { asset: { url: string } };
  price: number;
  slug: { current: string };
}
const Search = async () => {
  const query = `*[_type == 'product'] | order(_updatedAt asc) {
    name,
    image,
    price,
    slug
  }`;
  const data: Product[] = await client.fetch(query);
  return <SearchComponent data={data} />;
};
// Separate functional component to handle client-side search
const SearchComponent = ({ data }: { data: Product[] }) => {
  const [searchTerm, setSearchTerm] = useState("");

  // Filter products based on search term
  const filteredProducts = data.filter((product) =>
    product.name.toLowerCase().includes(searchTerm.toLowerCase())
  );

```

6. Reviews Section

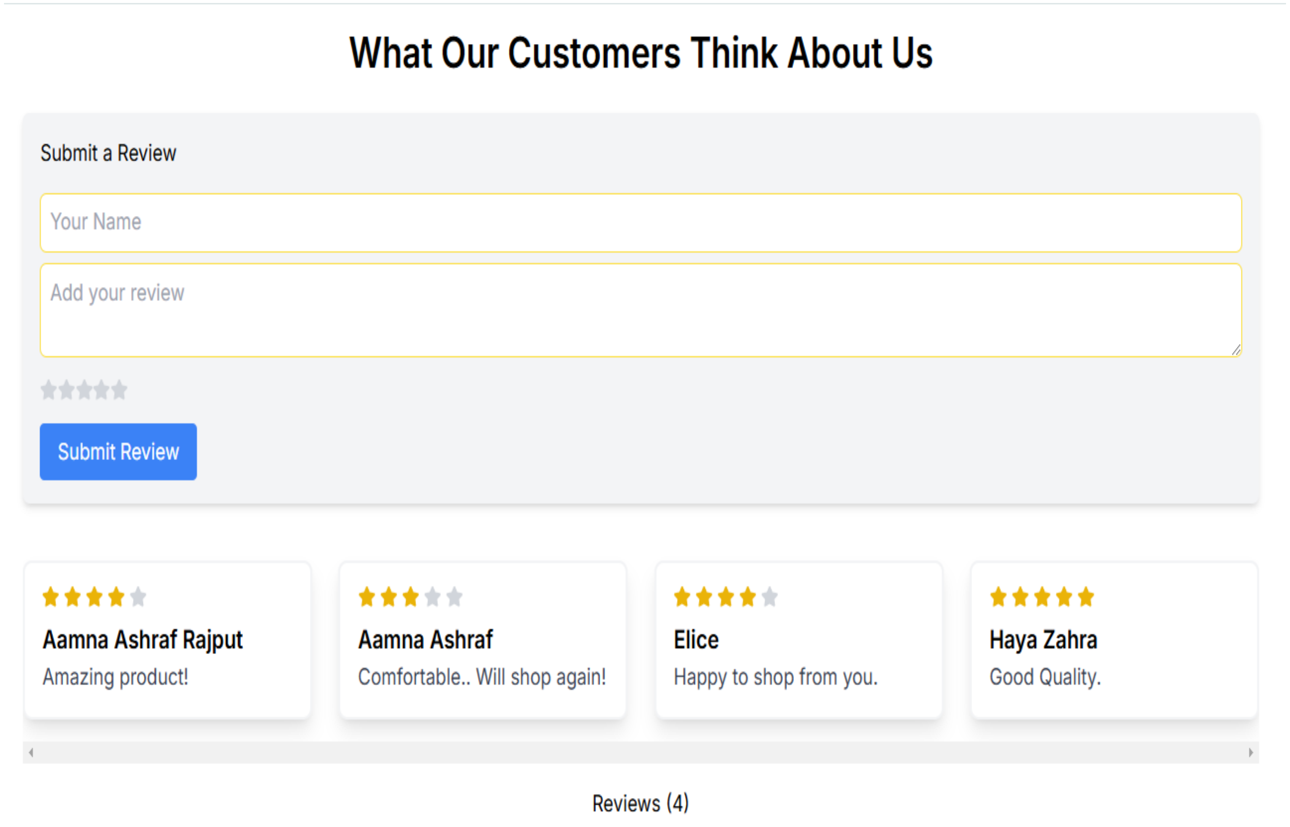
I designed and implemented a **Reviews Section** on the product detail pages to enhance user interaction and engagement. Although the reviews are not fetched dynamically yet, I created the functionality for users to:

- **Add Reviews**: Users can submit their own reviews, including a rating and comment.
- **Read Reviews**: Users can view reviews submitted by others directly on the product page.

- **Features Implemented:**
    - **Rating System**: Allows users to give a star rating (e.g., 1–5 stars) while submitting their reviews.
    - **Review Submission Form**: A form where users can write and submit their feedback.
    - **Review Display Section**: Displays the list of reviews added by users, including their name, rating, and comments.
    - **Real-Time Updates**: Once a review is submitted, it appears immediately on the reviews list without requiring a page reload.

What I Learned:
This task allowed me to explore form handling and managing state for user-submitted data. Adding a rating feature provided a professional touch, making the reviews section more interactive and functional.

Future Plans:
I aim to integrate **dynamic reviews** in the future, fetching and storing the data in a database like **Sanity** for scalability and persistence.


What Our Customers Think About Us

Submit a Review

Your Name

Add your review

Submit Review

★★★★☆
Aamna Ashraf Rajput
Amazing product!

★★★☆☆
Aamna Ashraf
Comfortable.. Will shop again!

★★★★☆
Elice
Happy to shop from you.

★★★★★
Haya Zahra
Good Quality.

Reviews (4)

Technical Implementation

Key Components:

Below are the main components I developed:

1. **Product Card Component**: Displays individual product details dynamically.
2. **Product List Component**: Renders the list of all products fetched from Sanity.
3. **Search Bar Component**: Implements the search functionality with live updates.
4. **Wishlist Component**: Manages user-selected favorite items.
5. **Checkout Component**: Handles the process of order finalization.
6. **Review Section:** Displays the list of reviews added by users, including their name, rating, and comments.

Sanity Integration

Instead of using an external API, I utilized **Sanity** as my CMS to manage and fetch data dynamically.

- All data (product details, reviews, etc.) was manually entered into Sanity.
- I fetched this data using GROQ queries for smooth integration into the frontend.

Challenges and Lessons Learned

- **Challenge 1**: Managing dynamic data in Sanity.
    - **Solution**: I organized the dataset with clear structure and relationships for better data flow.
- **Challenge 2**: Making the Wishlist functionality dynamic.
    - **Solution**: Added proper state management to track and update items dynamically.
- **Challenge 3**: Ensuring search bar responsiveness.
    - **Solution**: Implemented real-time suggestions and optimized the feature for mobile screens.

Through these challenges, I gained confidence in working with Sanity and improved my problem-solving skills in real-time application development.

Conclusion:

Working on these dynamic frontend components was an enriching experience. Using Sanity for data management streamlined the process and allowed for flexible content handling. Each feature—whether it was the product listing page, detailed product views, Wishlist, reviews, or the checkout page—enhanced the website's functionality and user experience.

Moving forward, I plan to focus on further optimizing performance, adding more features like user authentication, and improving the overall design. This project has been a fantastic learning journey, and I look forward to applying these skills in future projects.