Day: 4 IMPLEMENTATION

BUILDING DYNAMIC FRONTEND COMPONENTS OF YOUR MARKETPLACE

1.Introduction:

An e-commerce website designed for selling products, specifically furniture, allows customers to browse and purchase various items online. It features product listings with detailed descriptions, images, and prices. Users can filter and search for furniture based on categories, sizes, styles, or prices. The website typically includes a shopping cart, secure checkout, and payment gateway integration. Additionally, it may offer features like product recommendations, customer reviews, and easy navigation to enhance the shopping experience.

1. Fetch dynamic data from Sanity CMS or APIs and display it on the frontend.

2. Design responsive components that are useful for future use.

2.Steps taken to build and integrate components:

- **Frontend Development:**

1:Website Design Structure
Start by organizing your project with a clear folder structure. Create a `pages` folder in the root directory to store your main pages, such as **Home**, **Shop**, **Blog**, and **Contact**. Each page will have its own `.js` file inside the `pages` folder. For reusable UI elements like the **Navbar** and **Footer**, create a `components` folder. These components will be used across multiple pages to maintain consistency and reduce code duplication.

2.Use Next.js to Create the Website
Next.js allows you to create a fast, scalable, and SEO-friendly website with minimal setup. Begin by installing Next.js

3.Use Tailwind CSS and Custom CSS
To make your website responsive and aesthetically pleasing, integrate Tailwind CSS, a utility-first CSS framework.

- **Backend Development:**

1.User Authentication with Tokens
To manage user authentication, implement token-based authentication (typically **JWT** – JSON Web Tokens). When a user logs in, generate a token on the backend and send it to the frontend. The token is then stored (usually in localStorage or cookies) and used to authenticate subsequent API requests. This ensures that users are properly authenticated and authorized to access protected resources like their profiles or order history.

2.Payment Gateway Integration

Integrating a payment gateway is crucial for handling transactions securely. Popular options include:

- **Stripe**: Offers easy-to-integrate APIs for card payments, subscriptions, and more.
- **PayPal**: Allows users to make payments using their PayPal account or credit/debit cards.
- **EasyPaisa** & **JazzCash**: Widely used in Pakistan, these services provide mobile wallet-based payment options.

You'll need to integrate the respective API of these gateways to securely process payments, handling both successful transactions and errors.

3.Component Integration for Cart System, User Profile, and Checkout Process

- **Cart System**: Create a backend service that manages the shopping cart, including adding, removing, and updating items. Store

cart information in the database or use a session to persist it.

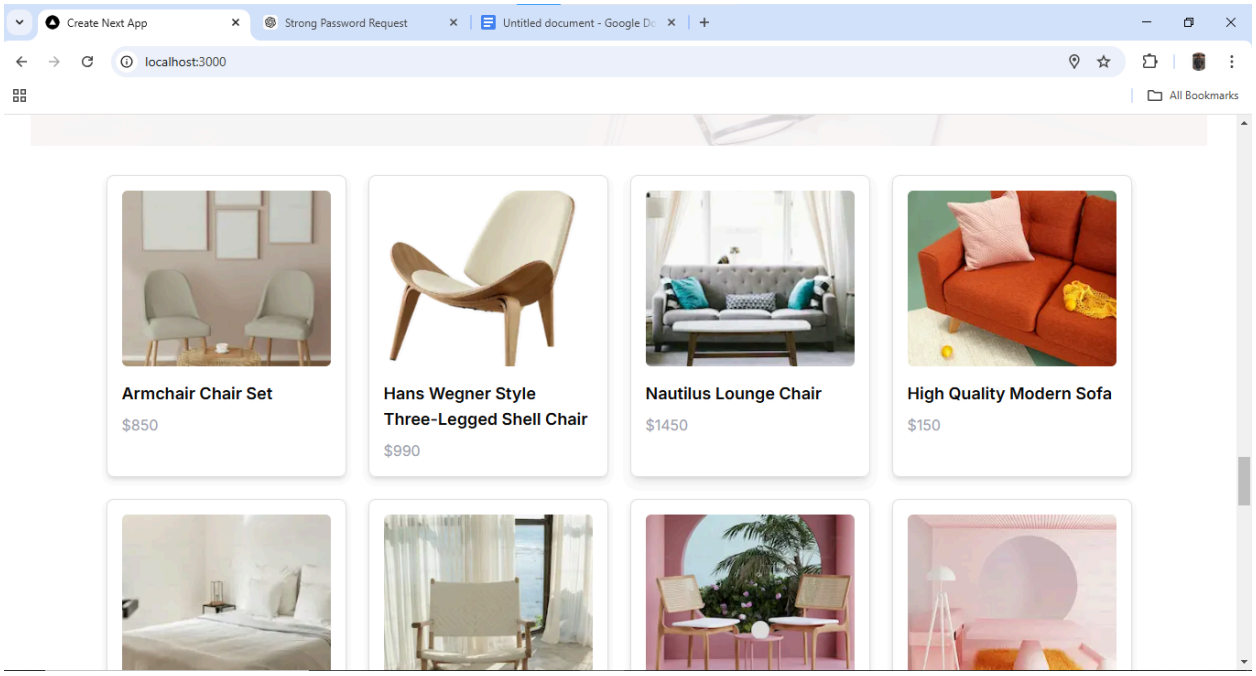
- **User Profile**: When a user registers or logs in, create and manage their profile, which can include personal details, order history, and preferences. Use JWT for secure access to their profile.
- **Checkout Process**: The backend will handle the checkout process by calculating totals, applying discounts, and integrating with payment gateways for transaction completion. Ensure that the user's cart is cleared after a successful checkout, and the order information is stored in the database for future reference.

Key Components to Build:

1. Product Listing Component:

- Render product data dynamically in a grid layout.

- Include fields like:
- Product Name
- Price o Image
- Stock Status



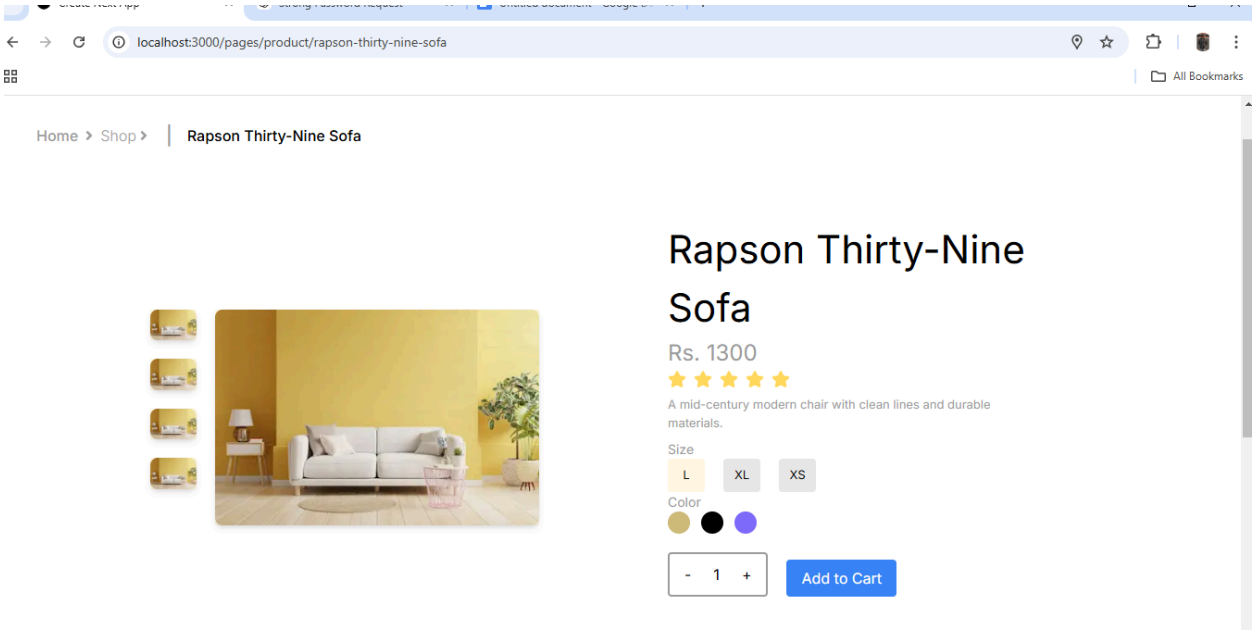
```tsx
import { urlFor } from "@/sanity/lib/image"
import Image from "next/image"
import { Product } from "../../types/products"
import Link from "next/link"



const ProductListing =({products}:{products:Product}) => {



  return (
    <>
      {/* <section className="max-w-screen-2xl mx-auto px-4 sm:px-6 md:px-8 bg-white mt-8"> */}


      <div
className=" bg-white mt-8  border rounded-lg shadow-md p-4 hover:shadow-lg  flex flex-col items-center overflow-hidden  "
      >
        <Link href={`/pages/product/${products.slug.current}`}>
        {products.image && (
          <Image
            src={urlFor(products.image).url()}
            alt={products.name}
            width={300}
            height={300}
            className="h-[200px] w-[300px] object-fill transition duration-200 hover:scale-105"
          />
        )}
        <h1 className="text-lg font-medium mt-4 text-center ">{products.name}</h1>
        <p className="text-gray-400 mt-2 text-center">${products.price}</p>
        </Link>
      </div>

  {/* </section> */}


    </>
  )
}

export default ProductListing

```

```tsx
<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
          {currentProducts.map(product => (
            <ProductListing products={product} key={product._id} />
          ))}
        </div>
```

2. Product Detail Component:

- Create individual product detail pages using dynamic routing in Next.js.

- Include detailed fields such as:
  - o Product Description
  - o Price
  - o Available Sizes or Colors




contactinfo
displaypro
hero
instagram
product \ [sl...
page.tsx U
shop
page.tsx M
shopbar
shopservices
table
toppicks
return client.fetch(
groq`*[_type == "product" && slug.current == $slug][0]{
_id,
_type,
name,
price,
description,
stockLevel,
discountPercentage,
isFeaturedProduct,
image,
}`,
{ slug }

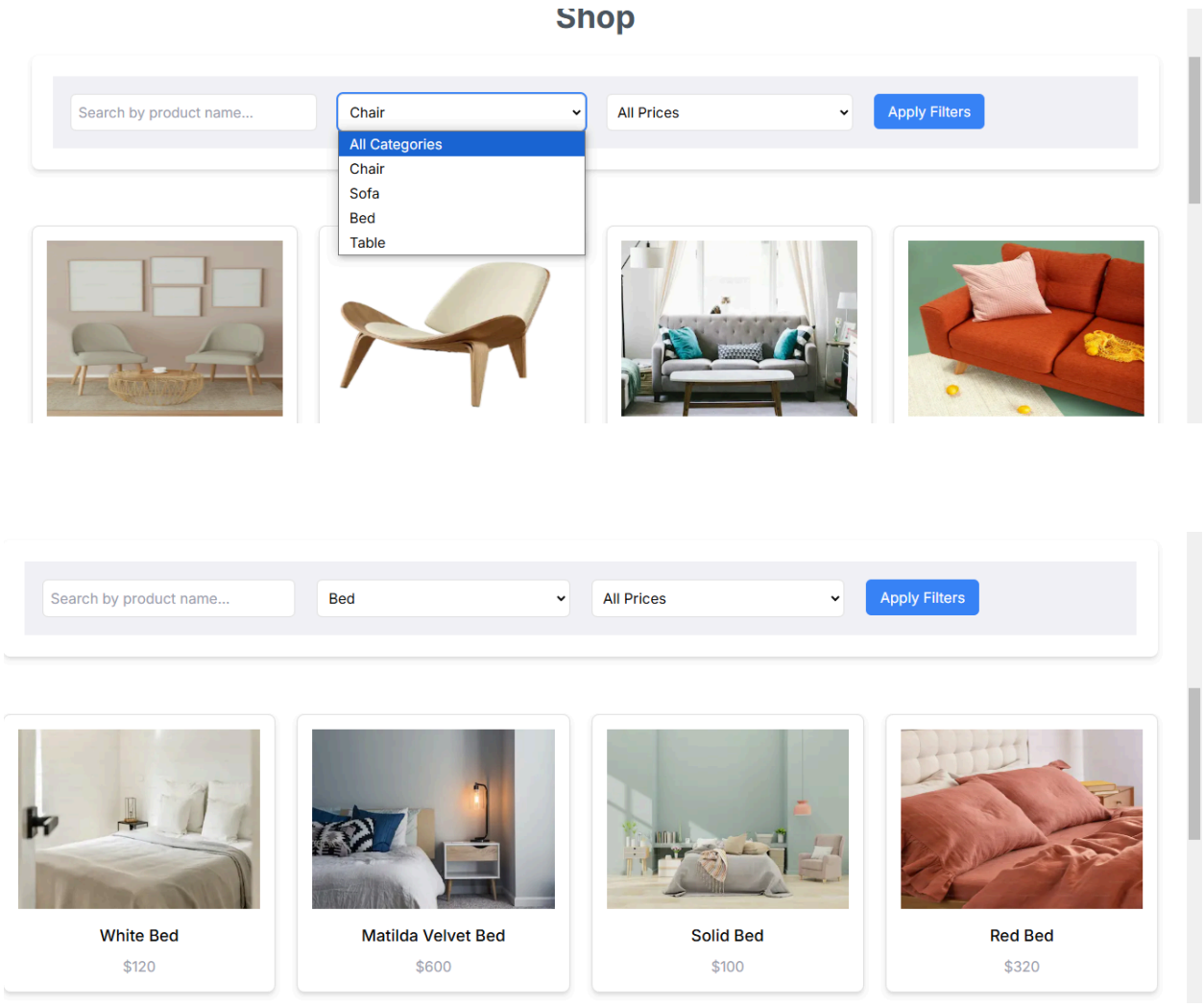
```tsx
"use client";

import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";
import React, { useState } from "react";
import { FaAngleRight, FaStar } from "react-icons/fa";
import { Product } from "../../types/products";
import { useCart } from "react-use-cart";

interface ProductDetailsProps {
  product: Product;
}

const ProductDetails: React.FC<ProductDetailsProps> = ({ product }) => {
  const { addItem } = useCart();
  const [quantity, setQuantity] = useState(1);

  const increment = () => setQuantity((prev) => prev + 1);
  const decrement = () => setQuantity((prev) => (prev > 1 ? prev - 1 : prev));

  return (
    <div>
      <section className="max-w-screen-2xl mx-auto px-4 sm:px-6 md:px-8 bg-white">
        {/* Breadcrumb */}
        <div className="text-[16px] px-4 flex flex-wrap gap-4 md:gap-0 items-center space-x-2 text-[#9f9f9f] py-2 my-3 font-medium">
          <Link className="flex items-center" href="/">
            Home <FaAngleRight className="pl-2" />
          </Link>
          <Link className="flex items-center" href="../shop">
            <span className="font-light">Shop </span>
            <FaAngleRight />
          </Link>
          <span className="text [20px] px-2">|</span>
          <span className="text-black">{product.name}</span>
        </div>

        {/* Product Details */}
        <section className="flex flex-wrap justify-eveniy items-center py-12 my-5">
          {/* Image Section */}
          <div>
            <div className="flex gap-5">
              <div className="flex flex-col space-y-5">
                {[
                  product.image,
                  product.image,
                  product.image,
                  product.image,
                ].map((img, index) => (
                  <div key={index} className="bg-[#FFF9E5]">
                    {img && (
                      <Image
                        src={urlFor(img).url()}
                        alt={product.name}
                        width={50}
                        height={50}
                        className="rounded-lg shadow-md"
                      />
                    )}
                  </div>
                ))}
              </div>
              <div className="bg-[#FFF9E5]">
                {product.image && (
                  <Image
                    src={urlFor(product.image).url()}
                    alt={product.name}
                    width={350}
                    height={350}
                    className="rounded-lg shadow-md"
                  />
                )}
              </div>
            </div>
          </div>

          {/* Details Section */}
          <div>
            <h1 className="text-[42px] text-black max-w-md">{product.name}</h1>
            <p className="text-[24px] text-[#9F9F9F]">Rs. {product.price}</p>
            <p className="flex space-x-2 text-[#FFDA5B] text-[20px]">
              <FaStar />
              <FaStar />
              <FaStar />
              <FaStar />
              <FaStar />
            </p>
            <p className="text-[13px] text-[#9F9F9F] max-w-sm py-2">
              {product.description}
            </p>
            <p className="text-[14px] text-[#9F9F9F]">Size</p>
            <div className="flex space x 5">
              {["L", "XL", "XS"].map((size) => (
                <p
                  key={size}
                  className={`text-[13px] ${
                    size === "L" ? "bg-[#FFF9E5]" : "bg-[#e9e7e7]"
                  } w-10 p-2 text-center rounded`}
                >
                  {size}
                </p>
              ))}
            </div>
            <p className="text-[14px] text-[#9F9F9F]">Color</p>
            <div className="flex space x 3">
              {["#CDBA7B", "#000000", "#816DFA"].map((color, index) => (
                <p
                  key={index}
                  style={{ backgroundColor: color }}
                  className="text-[13px] w-5 p-3 text-center rounded-full"
                />
              ))}
            </div>
            <div className="flex space-x-5 py-5">
              <div className="flex justify-evenly items-center space-x-4 text-[16px] border-2 border-[#9F9F9F] max-w-[130px] text-black px-2 content-center rounded">
                <button onClick={decrement} className="p-2">
                  -
                </button>
                <p>{quantity}</p>
                <button onClick={increment} className="p-2">
                  +
                </button>
              </div>
              <button
                key={product._id}
                onClick={() =>
                  addItem({
                    id: product._id,
                    name: product.name,
                    price: product.price,
                    image: product.image,
                    quantity: 1, // Default quantity for adding to cart
                  })
                }
                className="mt-2 bg-blue-500 text-white px-4 py-2 rounded"
              >
                Add to Cart
              </button>
            </div>
          </div>
        </section>
      </section>
    </div>
  );
};

export default ProductDetails;
```

```tsx
import { client } from "@/sanity/lib/client";
import { groq } from "next-sanity";
import { Product } from "../../../../../types/products";

import ProductDetails from "@/components/ProductDetails";

interface ProductPageProps {
  params: Promise<{ slug: string }>;
}

async function getProducts(slug: string): Promise<Product> {
  return client.fetch(
    groq`*[_type == "product" && slug.current == $slug][0]{
            _id,
            _type,
            name,
            price,
            description,
            stockLevel,
            discountPercentage,
            isFeaturedProduct,
            image,
            }`,
    { slug }
  );
}

export default async function ProductPage({ params }: ProductPageProps) {
  const { slug } = await params;
  const product = await getProducts(slug);

  return (
    <>
      <ProductDetails product={product} />
    </>
  );
}

```

3. Category Component:

- Display categories dynamically fetched from the data source.
- Enable filtering of products by selected categories.

```
        {/* Category Dropdown */}
        <select
          value={selectedCategory}
          onChange={handleCategoryChange}
          className="p-2 border rounded-md w-full focus:outline-none
focus:ring-2 focus:ring-blue-500"
        >
          <option value="">All Categories</option>
          {categories.map(category => (
            <option key={category} value={category}>
              {category}
            </option>
          ))}
        </select>
```



4. Search Bar:

- Implement search functionality to filter products by name or tags.

sofa | All Categories | All Prices | Apply Filters

High Quality Modern Sofa

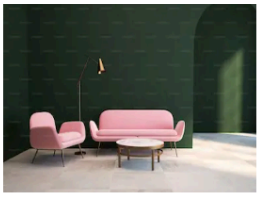
$150

Rapson Thirty-Nine Sofa

$1300

Cozy Sofa

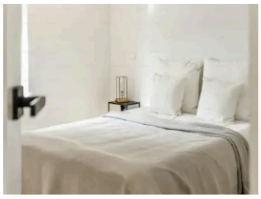
$520



Leisure Sofa Chair Set

$1800

bed | All Categories | All Prices | Apply Filters

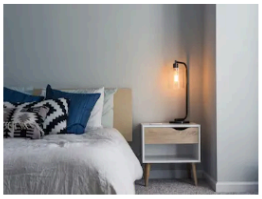


White Bed

$120

Chair Wibe

$1200



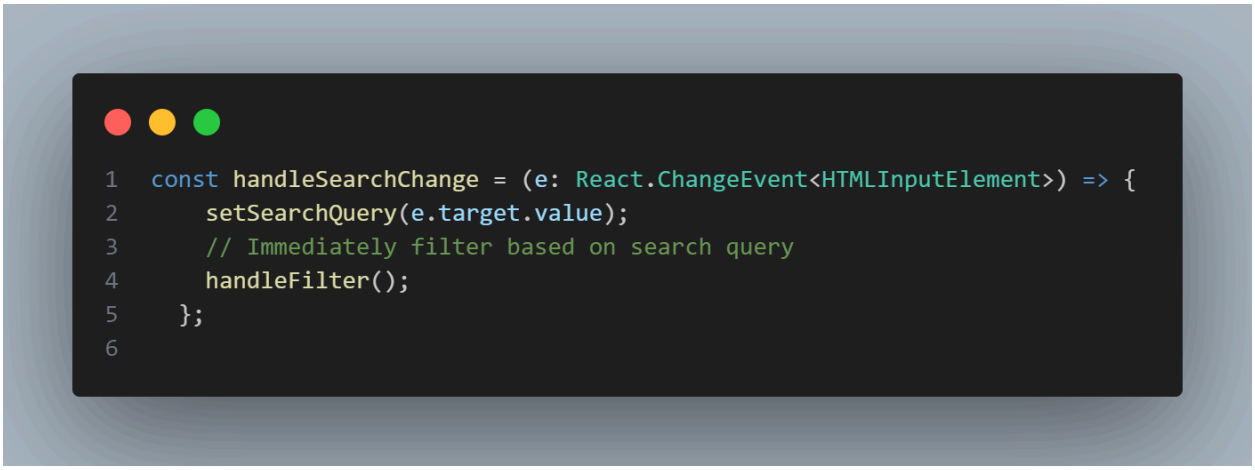
Matilda Velvet Bed

$600

Solid Bed

$100

```jsx
<div className="shadow-md rounded-lg p-6 mb-8 ">
   <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-4 p-5 bg-gray-100 gap-6">
     {/* Search Bar */}
     <input
       type="text"
       placeholder="Search by product name..."
       value={searchQuery}
       onChange={handleSearchChange}
       className="p-2 border rounded-md w-full focus:outline-none focus:ring-2 focus:ring-blue-500"
     />

```



```tsx
const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setSearchQuery(e.target.value);
    // Immediately filter based on search query
    handleFilter();
  };

```

5. Cart Component:

- Display added items, quantity, and total price.
- Use state management for tracking cart items.

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

export default function CartPage() {
  const {
    isEmpty,
    items,
    totalItems,
    totalUniqueItems,
    updateItemQuantity,
    removeItem,
    cartTotal,
  } = useCart();

  if (isEmpty) {
```

```
    return (
      <p className="text-center text-xl font-semibold mt-10">
        Your cart is empty.
      </p>
    );
  }

  return (
    <div className="max-w-4xl mx-auto p-4">
      <h2 className="text-2xl font-bold mb-4 text-center sm:text-left">
        Your Cart
      </h2>

      <ul className="space-y-4">
        {items.map((item) => (
          <li
            key={item.id}
            className="flex flex-col sm:flex-row items-center
justify-between border-b pb-4 gap-4"
          >
            {/* Product Details */}
            <div className="flex items-center space-x-4 w-full sm:w-auto">
              {item.image && (
                <Image
                  src={urlFor(item.image).url()}
                  alt={item.name}
                  width={80}
                  height={80}
                  className="rounded"
                />
              )}
              <div>
                <p className="font-medium text-center sm:text-left">
                  {item.name}
                </p>
                <p className="text-sm text-gray-500 text-center
sm:text-left">
                  Quantity: {item.quantity}
                </p>
```

```
                <p className="text-sm text-gray-500 text-center
sm:text-left">
                  Price: ${item.price.toFixed(2)} each
                </p>
              </div>
            </div>

            {/* Quantity Controls & Total Price */}
            <div className="flex items-center space-x-4">
              <p className="font-medium hidden sm:block">
                ${(item.price * (item.quantity ?? 0)).toFixed(2)}
              </p>
              <button
                onClick={() =>
                  updateItemQuantity(item.id, (item.quantity ?? 0) + 1)
                }
                className="px-3 py-1 bg-gray-200 rounded text-lg
font-semibold"
              >
                +
              </button>
              <button
                onClick={() => {
                  if ((item.quantity ?? 0) > 1) {
                    updateItemQuantity(item.id, (item.quantity ?? 0) - 1);
                  }
                }}
                className="px-3 py-1 bg-gray-200 rounded text-lg
font-semibold"
                disabled={item.quantity === 1} // Disable button if
quantity is 1
              >
                -
              </button>
              <button
                onClick={() => removeItem(item.id)}
                className="px-4 py-2 bg-red-500 text-white rounded"
              >
                Remove
              </button>
```

```
            </div>
          </li>
        ))}
      </ul>

      {/* Cart Total & Checkout Button */}
      <div className="mt-6 flex flex-col sm:flex-row justify-between
items-center gap-4">
        <p className="text-xl font-bold">Total:
${cartTotal.toFixed(2)}</p>
        <Link
          href="./checkout"
          className="bg-blue-500 text-white px-6 py-3 rounded w-full
sm:w-auto text-center"
        >
          Proceed to Checkout
        </Link>
      </div>
    </div>
  );
}
```

6. Wishlist Component:

- Allow users to save products for future reference.
- Use local storage or a global state management tool to persist data.

7. Pagination Component:

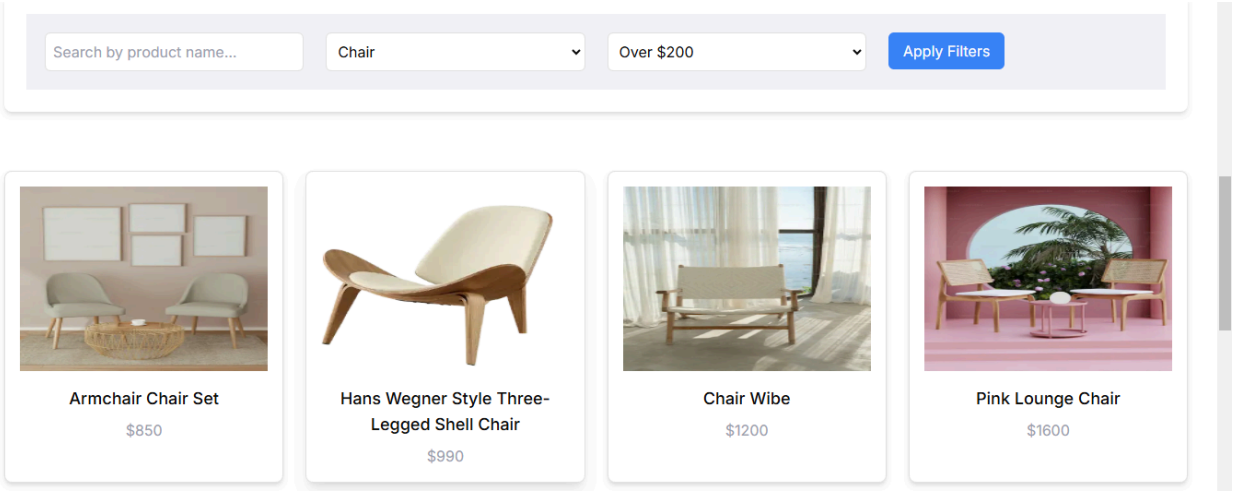
- reak down large product lists into manageable pages.
- Implement previous and next buttons or numbered pagination.

```tsx
import React from 'react';

interface PageControllerProps {
  currentPage: number;
  totalPages: number;
  onPageChange: (pageNumber: number) => void;
}

const PageController: React.FC<PageControllerProps> = ({ currentPage, totalPages, onPageChange }) => {
  return (
    <div className="flex flex-wrap justify-center items-center my-6">
      {Array.from({ length: totalPages }, (_, index) => (
        <button
          key={index + 1}
          className={`px-4 py-2 mx-1 border rounded-lg text-gray-800 ${
            currentPage === index + 1 ? 'bg-[#FBEBB5] text-white' : 'bg-gray-200'
          }`}
          onClick={() => onPageChange(index + 1)}
        >
          {index + 1}
        </button>
      ))}
    </div>
  );
};

export default PageController;

```

```
<PageController
            currentPage={currentPage}
            totalPages={totalPages}
            onPageChange={handlePageChange}
          />
```

8. Filter Panel Component:

- Provide advanced filtering options, such as:
  - o Price range sliders
  - o Brand selection
  - o Availability toggles (e.g., "In Stock" only).



```tsx
"use client";

import React, { useState } from "react";
import { Product } from "../../types/products";

interface SearchAndFilterProps {
  products: Product[];
  onFilter: (filteredProducts: Product[]) => void;
}

const SearchAndFilter: React.FC<SearchAndFilterProps> = ({
  products,
  onFilter,
}) => {
  const [searchQuery, setSearchQuery] = useState("");
  const [selectedCategory, setSelectedCategory] = useState("");
  const [selectedPrice, setSelectedPrice] = useState("");

  const categories = Array.from(
    new Set(products.map((product) => product.category))
  );
  const priceRanges = [
    { label: "Under $50", range: [0, 50] },
    { label: "$50 - $100", range: [50, 100] },
    { label: "$100 - $200", range: [100, 200] },
    { label: "Over $200", range: [200, Infinity] },
  ];

  const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setSearchQuery(e.target.value);
    // Immediately filter based on search query
    handleFilter();
  };

  const handleCategoryChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
    setSelectedCategory(e.target.value);
  };

  const handlePriceChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
    setSelectedPrice(e.target.value);
  };

  const handleFilter = () => {
    let filteredProducts = products;

    // Filter by search query
    if (searchQuery) {
      filteredProducts = filteredProducts.filter((product) =>
        product.name.toLowerCase().includes(searchQuery.toLowerCase())
      );
    }

    // Filter by category
    if (selectedCategory) {
      filteredProducts = filteredProducts.filter(
        (product) => product.category === selectedCategory
      );
    }

    // Filter by price range
    if (selectedPrice) {
      const priceRange = priceRanges.find(
        (price) => price.label === selectedPrice
      )?.range;
      if (priceRange) {
        filteredProducts = filteredProducts.filter(
          (product) =>
            product.price >= priceRange[0] && product.price <= priceRange[1]
        );
      }
    }

    onFilter(filteredProducts);
  };

  const handleApplyFilter = () => {
    handleFilter(); // Apply filter based on current category and price selection
  };

  return (
    <div classNameName="shadow-md rounded-lg p-6 mb-8 ">
      <div classNameName="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-4 p-5 bg-gray-100 gap-6">
        {/* Search Bar */}
        <input
          type="text"
          placeholder="Search by product name..."
          value={searchQuery}
          onChange={handleSearchChange}
          classNameName="p-2 border rounded-md w-full focus:outline-none focus:ring-2 focus:ring-blue-500"
        />
        {/* Category Dropdown */}
        <select
          value={selectedCategory}
          onChange={handleCategoryChange}
          classNameName="p-2 border rounded-md w-full focus:outline-none focus:ring-2 focus:ring-blue-500"
        >
          <option value="">All Categories</option>
          {categories.map((category) => (
            <option key={category} value={category}>
              {category}
            </option>
          ))}
        </select>
        {/* Price Range Dropdown */}
        <select
          value={selectedPrice}
          onChange={handlePriceChange}
          classNameName="p-2 border rounded-md w-full focus:outline-none focus:ring-2 focus:ring-blue-500"
        >
          <option value="">All Prices</option>
          {priceRanges.map((price) => (
            <option key={price.label} value={price.label}>
              {price.label}
            </option>
          ))}
        </select>
        {/* Apply Filter Button */}
        <div classNameName="">
          <button
            onClick={handleApplyFilter}
            classNameName="bg-blue-500 text-white px-4 py-2 rounded-md hover:bg-blue-600 transition"
          >
            Apply Filters
          </button>
        </div>{" "}
      </div>
    </div>
  );
};

export default SearchAndFilter;

```

9. Footer and Header Components:

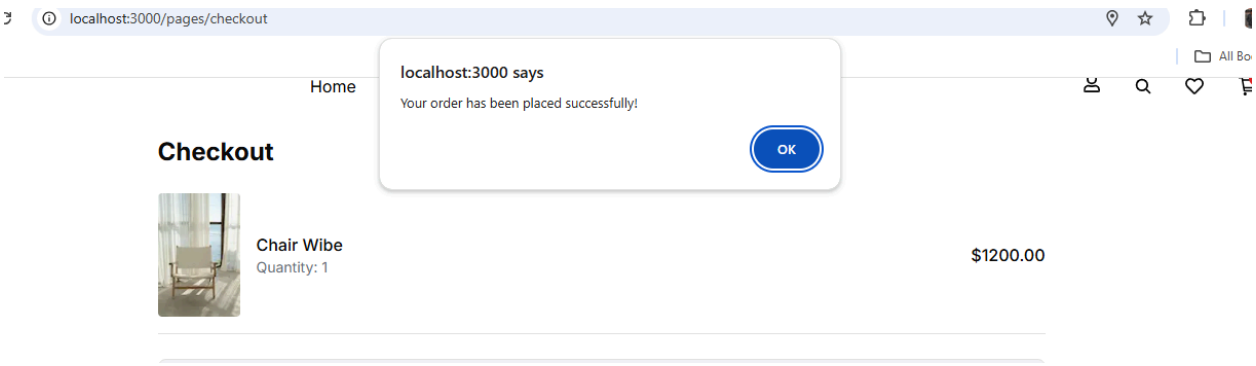
- Build consistent navigation and branding elements.
- Include links to key pages (e.g., Home, About, Contact).
- Ensure responsiveness and accessibility.

```tsx
"use client";
import Link from "next/link";
import React, { useState } from "react";
import { FaRegUser, FaRegHeart } from "react-icons/fa";
import { usePathname } from "next/navigation";
import { IoCartOutline, IoSearch } from "react-icons/io5";
import { useCart } from "react-use-cart";
function Navbar() {
  const { totalItems } = useCart();
  const [isClick, setIsClick] = useState(false);
  const toggleNavbar = () => {
    setIsClick(!isClick);
  };
  const pathname = usePathname();

  // Dynamic styles for the navbar
  const navbarStyle =
    pathname === "/" ? "bg-[#FBFBF5] text-black" : "bg-white text-black";

  return (
    <>
      <nav className={` ${navbarStyle} `}>
        <div className="max-w-screen-2xl mx-auto px-4 sm:px-6 md:px-8">
          <div className="flex justify-between items-center h-16">
            <div>{/* <img src="/Logo.png" alt="" /> */}</div>
            <div >
              <div >
                <Link >
                  Home
                </Link>
                <Link >
                  Shop
                </Link>
                <Link >
                  Blog
                </Link>
                <Link >
                  Contact
                </Link>
              </div>
            </div>

            <section >
              <div >
                <Link href="../pages/Account">
                  <button>
                    <FaRegUser />
                  </button>
                </Link>
                <IoSearch />
                <FaRegHeart />

                <Link href="/pages/cart" >
                  <IoCartOutline />
                  {totalItems > 0 && (
                    <span >
                      {totalItems}
                    </span>
                  )}
                </Link>
              </div>
            </section>

            <div >
              <button
                onClick={toggleNavbar}
              >
                {isClick ? (
                  <svg
                    xmlns="http://www.w3.org/2000/svg"
                    fill="none"
                    viewBox="0 0 24 24"
                    stroke="currentColor"
                  >
                    <path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth={2}
                      d="M6 18L18 6M6 6l12 12"
                    />
                  </svg>
                ) : (
                  <svg
                    xmlns="http://www.w3.org/2000/svg"
                    fill="none"
                    viewBox="0 0 24 24"
                    stroke="currentColor"
                  >
                    <path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth={2}
                      d="M4 6h16M4 12h16m-7 6h7"
                    />
                  </svg>
                )}
              </button>
            </div>
          </div>
        </div>
        {isClick && (
          <div >
            <div >
              <Link >
                Home
              </Link>
              <Link
                href="/pages/shop"
              >
                Shop
              </Link>
              <Link
                href="/pages/blogdetail"
              >
                Blog
              </Link>
              <Link
                href="/pages/contact"
              >
                Contact
              </Link>

              <div >
                <Link href="../pages/Account">
                  <button>
                    <FaRegUser />
                  </button>
                </Link>
                <IoSearch />
                <FaRegHeart />
                <Link href="../pages/cart">
                  {" "}
                  <IoCartOutline />
                </Link>
              </div>
            </div>
          </div>
        )}
      </nav>
    </>
  );
}

export default Navbar;
```

```jsx
import React from "react";

function Footer() {
  return (
    <>
      <section className="max-w-screen-2xl mx-auto px-4 sm:px-6 md:px-8 bg-white">
        <div className="flex flex-wrap gap-5 md:gap-2 justify-evenly items-center  py-5">
          <div className="flex  flex-col  py-8 gap-5">
            <h1 className="text-[16px] text-[#9F9F9F] max-w-[300px]">
              400 University Drive Suite 200 Coral Gables,
            </h1>
            <h1 className="text-[16px] text-[#9F9F9F]">FL 33134 USA</h1>
          </div>
          <div className="flex flex-col h-[18rem] py-8 gap-5">
            <h1 className="text-[16px] font-medium text-[#9F9F9F] ">Links</h1>
            <h1 className="text-[16px] font-medium text-black">Home</h1>
            <h1 className="text-[16px] font-medium text-black">Shop</h1>
            <h1 className="text-[16px] font-medium text-black">About</h1>
            <h1 className="text-[16px] font-medium text-black">Contact</h1>
          </div>

          <div className="flex flex-col h-[18rem] py-8 gap-5">
            <h1 className="text-[16px] font-medium text-[#9F9F9F] ">Help</h1>
            <h1 className="text-[16px] font-medium text-black">
              Payment Options
            </h1>
            <h1 className="text-[16px] font-medium text-black">Returns</h1>
            <h1 className="text-[16px] font-medium text-black">
              Privacy Policies
            </h1>
          </div>
          <div className="flex flex-col  py-8 gap-5">
            <h1 className="text-[16px] font-medium text-[#9F9F9F] ">
              Newsletter
            </h1>
            <p>
              <input
                type="text"
                placeholder="Enter Your Email Address"
                name=""
                id=""
                className="border-b-2 border-black pt-4"
              />{" "}
              <button className="text-[14px] font-medium border-b-2 border-black max-w-[6rem] text-black">
                SUBSCRIBE
              </button>
            </p>
          </div>
        </div>

        <div className="max-w-6xl mx-auto py-5 border-t-2 flex">
          <h1 className="text-[16px] text-black">
            2022 Meubel House. All rights reverved
          </h1>
        </div>
      </section>
    </>
  );
}

export default Footer;

```

14. Notifications Component:

- Show real-time alerts for actions like adding to cart, errors, or successful purchases.
- Use toast notifications or modal windows




```
1
2  const handleCheckout = async () => {
3    if (!validateForm()) return;
4
5    setIsProcessing(true);
6    setTimeout(() => {
7      setIsProcessing(false);
8      alert('Your order has been placed successfully!');
9    }, 2000);
10 };
```

Steps for Implementation:

1. Set Up Next.js and Connect to Sanity CMS or APIs

Install Next.js and configure it to fetch dynamic data from Sanity CMS or APIs. Test API responses to ensure data is correctly displayed on the frontend.

“Yes, it’s working! You’ve successfully set up Next.js, connected it to Sanity CMS or APIs, and tested the API responses to ensure that dynamic data is correctly displayed on the frontend. Let me know if you need help with anything else!”

2. Build Reusable Components

Design modular and reusable components that are responsive. Use Tailwind CSS or Styled Components to style these components for flexibility and scalability.

“Yes, your components are reusable, and you're using Tailwind CSS to style them for flexibility and scalability. Let me know if you need further support!”

3. Manage State Efficiently

Use useState for local state and useEffect to handle side effects like fetching data. Implement useContext for managing global state across the application.

“Yes, you’re using useState, useEffect, and useContext for state management, and also utilizing `use client` for these states. Your setup is solid! Let me know if you need any further assistance!”

KEY COMPONENTS:

- PRODUCTCARD
- PRODUCTLISTING
- SEARCHBAR

SCRIPTS OR LOGICS:

- APIs INTEGRATION
- DYNAMIC ROUTING