

DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR MARKETPLACE---HECTO

1. Functional Deliverables:

Key Features and Functionalities

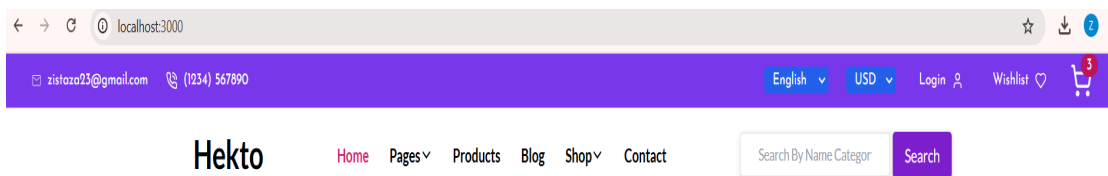
1:Header Functionality:

Cart Icons:

- These counters automatically update in real-time as products are added or removed from the cart.

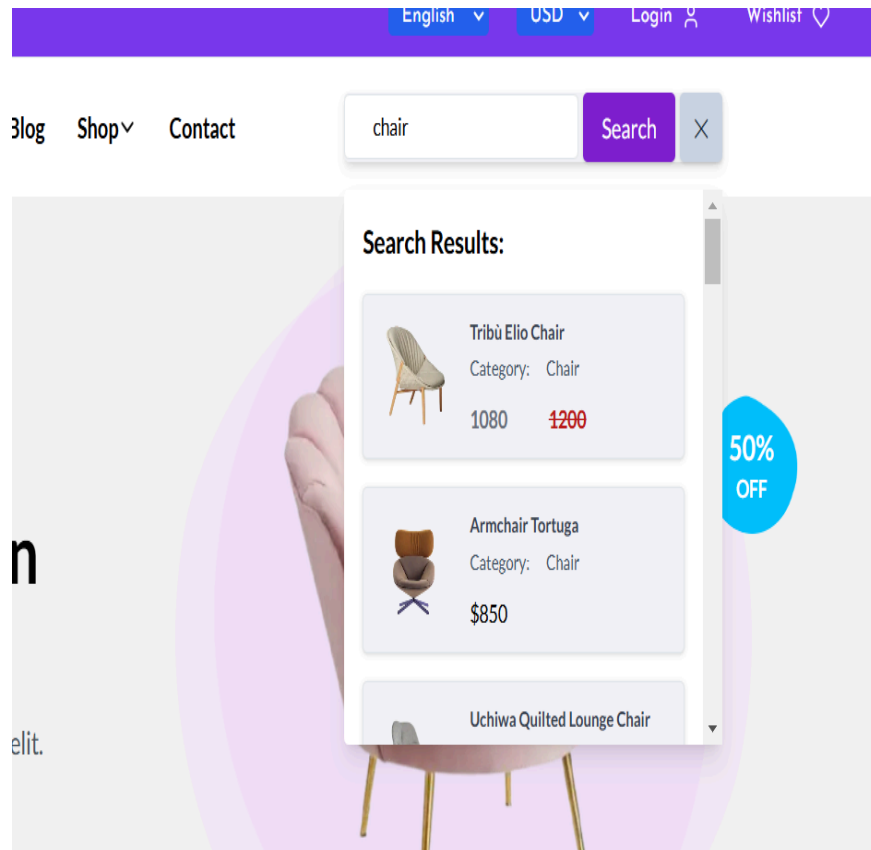
Login:

- Users can log in using Google

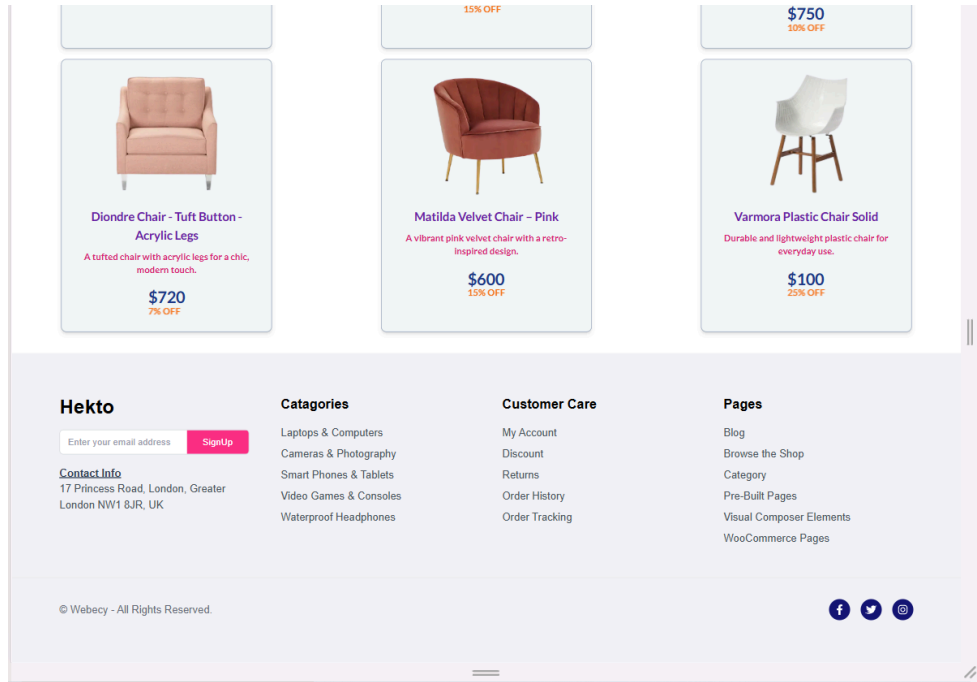


- **2:Search Capability:**
 - An advanced search bar enables users to effortlessly find products by entering names or tags.

- Results are dynamically updated in real-time, providing instant and relevant suggestions as users type.

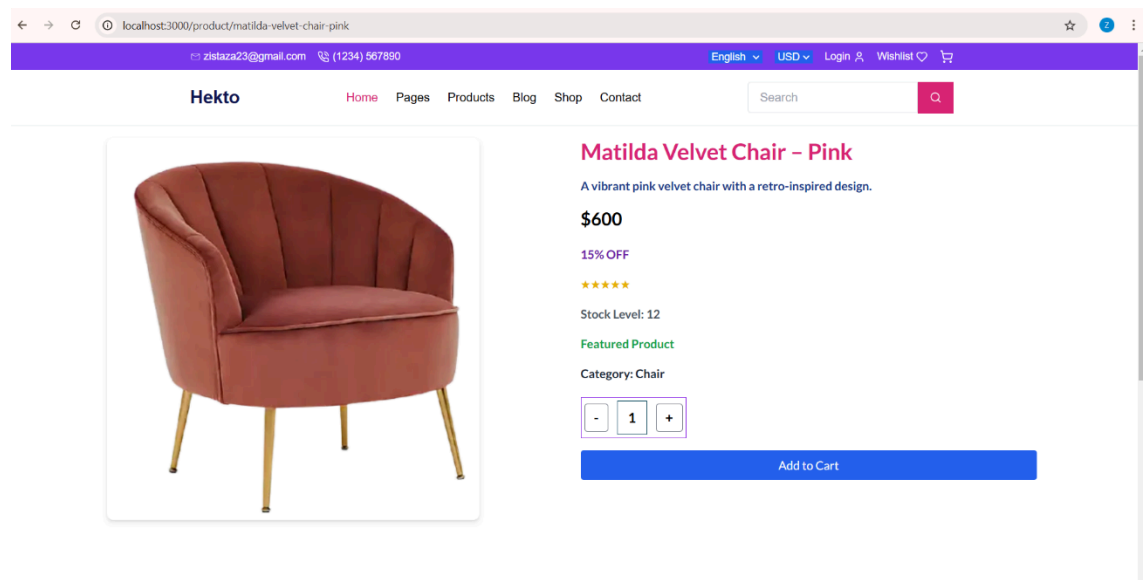
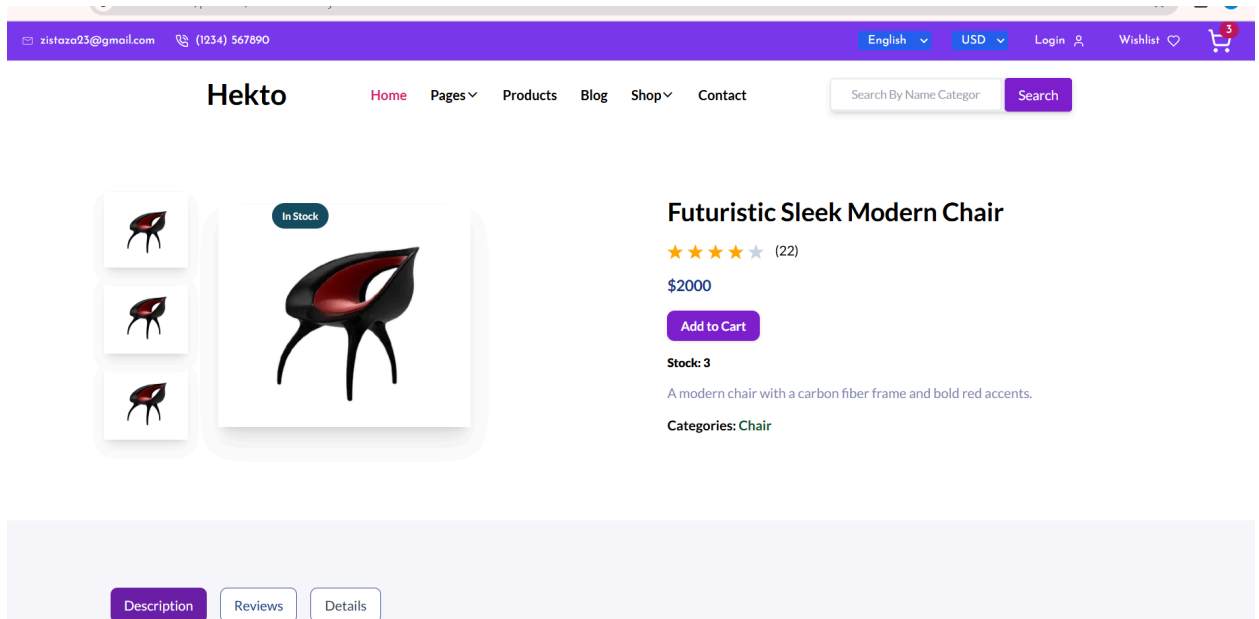


ProductListing Components:



DynamicProduct Detail Component

- Utilized Next.js for dynamic routing to generate unique pages for each product.
- Pages display the product name, description, price, stock status, category, featured status, ratings, discounts, and images.



Reusable Components

- Developed modular and reusable components for improved scalability and easy maintenance.
- Key examples include:
 - ProductCard

- o Navbar
- o Footer

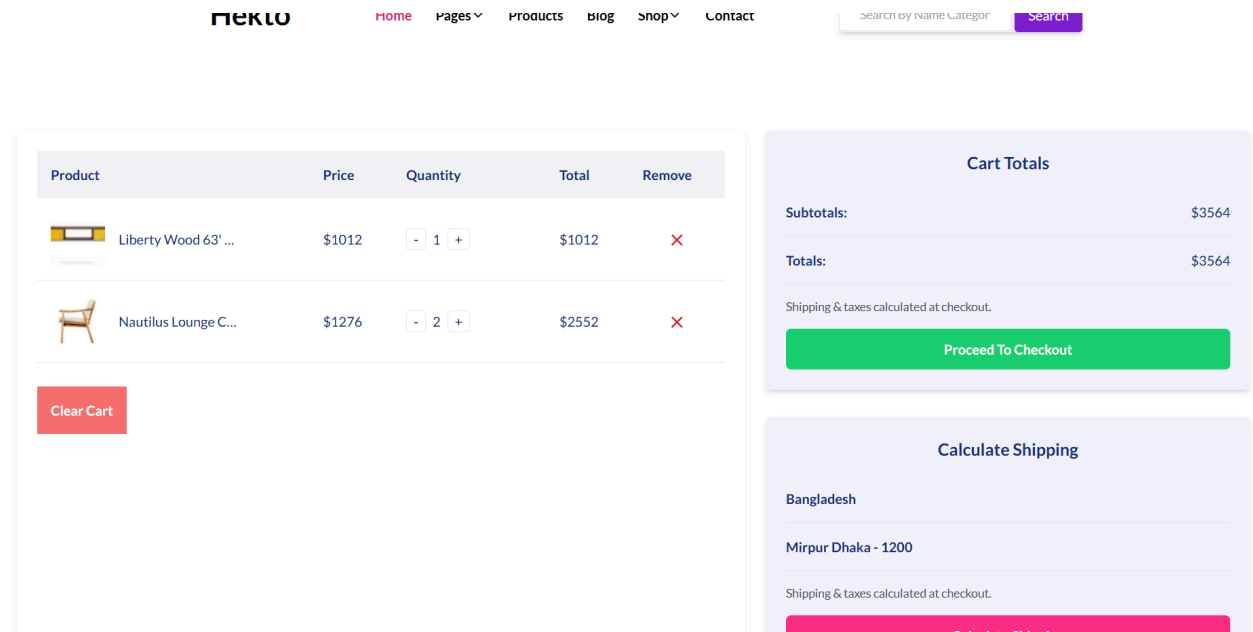
Loader:

- o A loading animation runs while the results are being fetched to improve user experience.



Add-to-Cart Functionality

- Enabled dynamic addition of products to the cart.
 - o Displays the total number of items and the cart's total price.
 - o Allows users to adjust product quantities directly within the cart.



Instant Alerts

- Added toast notifications to provide immediate feedback to users.
- These notifications cover:
 - Confirmation messages when products are added to the cart.
 - Error warnings, such as when users try to add unavailable items.

Product Details

OrderCompleted . Pages . OrderCompleted



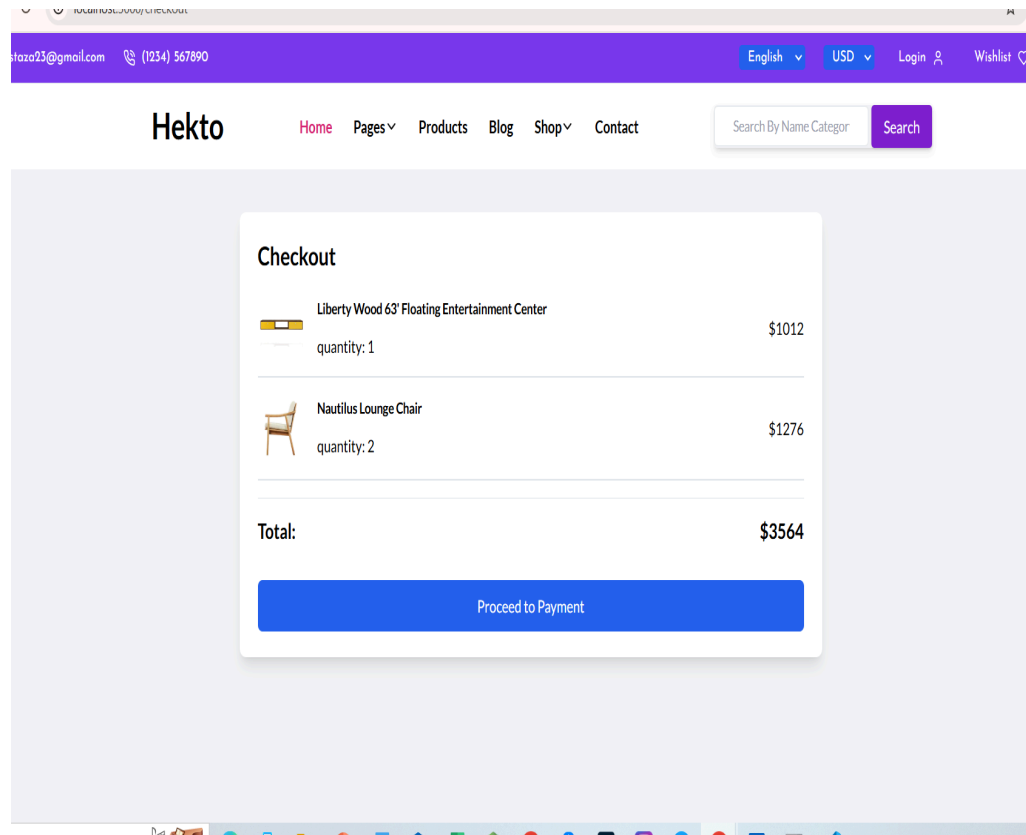
Your order is completed!

Thank you for your order! Your order is being processed and will be completed within 3-6 hours. You will receive an email confirmation when your order is completed.

[Continue Shopping](#)

Checkout Process

- Implemented a multi-step checkout page featuring:
 - Forms to input billing and shipping addresses.
 - Integration of a mock payment gateway.
 - A detailed order summary displayed before final confirmation.



2. Code Deliverables:

Dynamic via [slug]

EXPLORER

src > app > product > [slug] > page.tsx > ...

HECTO-...

src

app

components

Homesection

Trending.tsx M

Uniquefeatures....

layout

ProductListi... U

Pages

About-us

Blog

Contact-us

Error

FAQ

MyAccount

Ordercomplete

ShopGridDefault

ShopList

page.tsx M

ShoppingCart

product\[slug]

AddToCartB... U

page.tsx U

ProductDeta... U

products

page.tsx U

studio\[...to...

page.tsx U

page.tsx ...\[slug] U X

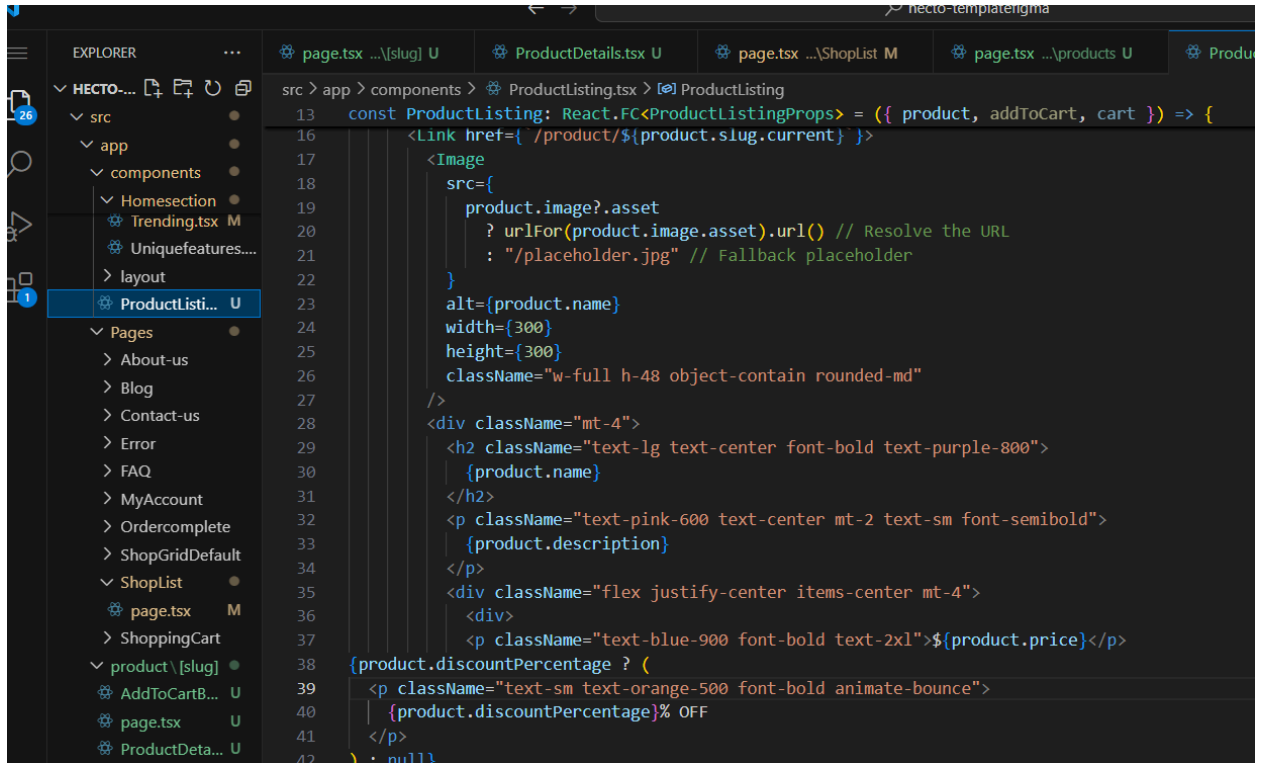
ProductDetails.tsx U

page.tsx ...\ShopList M

page.tsx ...\products U

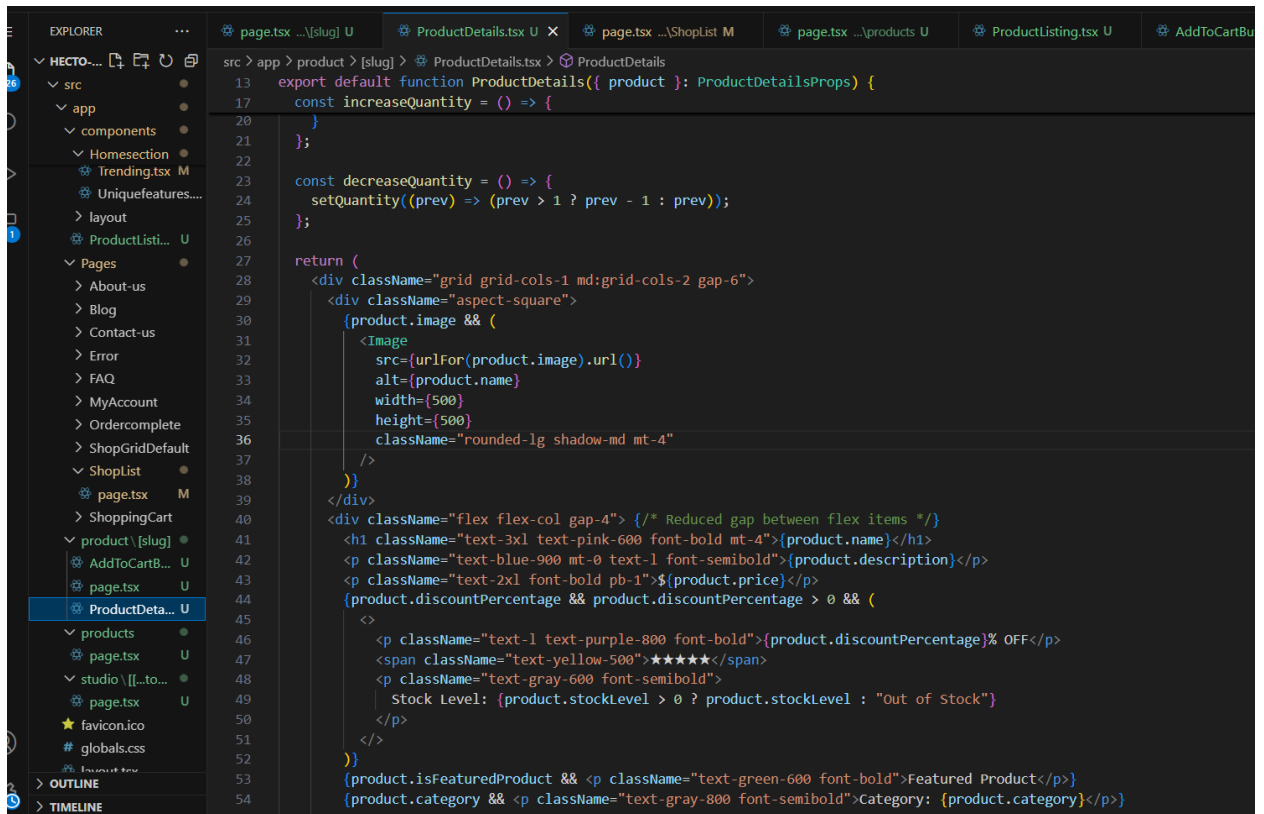
```
8
9  interface ProductPageProps {
10     params: { slug: string };
11  }
12
13  // Fetch the product data in a Server Component
14  async function getProduct(slug: string): Promise<Product> {
15      return client.fetch(
16          groq`*_type == "product" && slug.current == $slug[0]{
17              _id,
18              name,
19              image,
20              price,
21              description,
22              discountPercentage,
23              stockLevel,
24              isFeaturedProduct,
25              category
26          }`,
27          { slug }
28      );
29  }
30
31  export default async function ProductPage({ params }: ProductPageProps) {
32      const product = await getProduct(params.slug); // Fetch data here
33
34      return (
35          <div className="max-w-7xl mx-auto px-4">
36              { /* Pass the product data to the Client Component */ }
37              <ProductDetails product={product} />
38          </div>
39      );
40  }
```

ProductListing



```
src > app > components > ProductListing.tsx > ProductListing
13 const ProductListing: React.FC<ProductListingProps> = ({ product, addToCart, cart }) => {
16   <Link href={ /product/${product.slug.current} }>
17     <Image
18       src={
19         product.image?.asset
20         ? urlFor(product.image.asset).url() // Resolve the URL
21         : "/placeholder.jpg" // Fallback placeholder
22       }
23       alt={product.name}
24       width={300}
25       height={300}
26       className="w-full h-48 object-contain rounded-md"
27     />
28   <div className="mt-4">
29     <h2 className="text-lg text-center font-bold text-purple-800">
30       {product.name}
31     </h2>
32     <p className="text-pink-600 text-center mt-2 text-sm font-semibold">
33       {product.description}
34     </p>
35     <div className="flex justify-center items-center mt-4">
36       <div>
37         <p className="text-blue-900 font-bold text-2xl">${product.price}</p>
38       {product.discountPercentage ? (
39         <p className="text-sm text-orange-500 font-bold animate-bounce">
40           {product.discountPercentage}% OFF
41         </p>
42       ) : null}
```

productdetailcard




```
src > app > product > [slug] > ProductDetails.tsx > ProductDetails
13 export default function ProductDetails({ product }: ProductDetailsProps) {
17   const increaseQuantity = () => {
20   }
21 };
22
23   const decreaseQuantity = () => {
24     setQuantity((prev) => (prev > 1 ? prev - 1 : prev));
25   };
26
27   return (
28     <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
29       <div className="aspect-square">
30         {product.image && (
31           <Image
32             src={urlFor(product.image).url()}
33             alt={product.name}
34             width={500}
35             height={500}
36             className="rounded-lg shadow-md mt-4"
37           />
38         )}
39       </div>
40       <div className="flex flex-col gap-4"> {/* Reduced gap between flex items */}
41         <h1 className="text-3xl text-pink-600 font-bold mt-4">{product.name}</h1>
42         <p className="text-blue-900 mt-0 text-l font-semibold">{product.description}</p>
43         <p className="text-2xl font-bold pb-1">${product.price}</p>
44         {product.discountPercentage && product.discountPercentage > 0 && (
45           <
46             <p className="text-l text-purple-800 font-bold">{product.discountPercentage}% OFF</p>
47             <span className="text-yellow-500">★★★★★</span>
48             <p className="text-gray-600 font-semibold">
49               Stock Level: {product.stockLevel > 0 ? product.stockLevel : "Out of Stock"}
50             </p>
49           </>
50         )}
51       </div>
52     </div>
53   );
54   {product.isFeaturedProduct && <p className="text-green-600 font-bold">Featured Product</p>}
55   {product.category && <p className="text-gray-800 font-semibold">Category: {product.category}</p>}
```

Tools and Technologies Used

- Frontend: Next.js, Tailwind CSS
- Backend CMS: Sanity (for managing dynamic content)
- State Management: React Context API
- Notifications: React Toastify

● Here's the checklist with tick marks to indicate completion

Category	Task	Status
Frontend Components	- Ensure all components are modular and reusable	✓
	- Verify product details page displays correct information	✓
	- Test add-to-cart functionality works as expected	✓
	- Check toast notifications trigger at appropriate actions	✓
	- cart icons dynamically display the number of items.	✓
	- quick actions like adding items to the cart appear on hover.	✓
	- Real-time product search with dynamic filtering	✓
	- Smooth loader animation while fetching results.	✓
	- Add, remove, and update product quantities.	✓
	- Total price calculated dynamically.	✓
	- Payment system integrated to process orders.	
	- Clear cart option available for quick actions	✓
Styling and Responsiveness	- Confirm Tailwind CSS is used for responsive design	✓
	- Ensure layout adjusts properly on different screen sizes	✓
	- Validate visual consistency across devices and browsers	✓
Code Quality	- Confirm clean, modular, and well-commented code	✓
	- Perform linting and fix any issues	✓
	- Run unit tests and ensure all pass	✓
Documentation	- Ensure all components are documented with usage examples	✓

Category	Task	Status
	- Provide clear instructions for running the project	
	- Document the setup for the development environment	