Day 4:

BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

Project Objective:

"The primary goal of this project is to develop a fully functional e-commerce platform that allows users to browse products, add them to a shopping cart, and manage the cart's content. The project also ensures a smooth user experience by incorporating persistent cart storage and a modern UI/UX design."

Introduction:

In this project, I developed a highly interactive, responsive, and user-friendly application that focuses on [insert the specific type of app, e.g., "e-commerce cart management," "dynamic blogging," or "recipe search functionality"]. My goal was to ensure functionality, scalability, and a visually appealing design that meets modern user expectations.

The project involves several key components, including state management, real-time updates, and persistent data storage, along with clean, maintainable code. I used a combination of modern tools and technologies to achieve these objectives.

Technologies and Tools Used

To ensure efficiency and quality, I leveraged the following technologies and tools:

- **React.js/Next.js**: For building a dynamic user interface and ensuring seamless routing.
- **TypeScript**: To enforce type safety and improve maintainability.
- **Tailwind CSS**: For creating a responsive and aesthetically pleasing design.

- **LocalStorage/SessionStorage**: For persistent state management to ensure data is not lost on refresh.
- **Git & GitHub**: For version control and collaborative development.

Key Features Implemented

1. **Dynamic State Management**
    - I implemented a global state using React’s Context API to manage key data across the application, such as the shopping cart or user preferences.
    - For instance, when a user adds items to the cart, the state updates dynamically and reflects changes across all relevant components in real time.
-
2. **Persistent Data Storage**
    - Using localStorage, I ensured that user data (e.g., cart items, user preferences) is saved even after refreshing or closing the browser. This feature enhances the user experience by avoiding data loss.
-
3. **Responsive Design**
    - I designed the user interface to be fully responsive, adapting seamlessly to screens of all sizes, from mobile devices to desktops, using Tailwind CSS and media queries.
-
4. **Component-Based Architecture**
    - Each feature was broken down into reusable components to ensure modularity and maintainability. For example:
        - Header.tsx: Displays navigation and cart summary.
        - ProductCard.tsx: Displays individual product details dynamically.
        - CartSummary.tsx: Summarizes items and total cost in the cart.
-
5. **Real-Time Updates**
    - Real-time updates were incorporated to display live changes to the cart total, item count, or any other user-interactive functionality without requiring a page reload.
-
6. **Optimized Codebase**
    - Followed DRY (Don’t Repeat Yourself) and SOLID principles to ensure efficient, readable, and reusable code.

  - Added inline comments and documentation for future scalability and easier collaboration.

Workflows and Steps

1. **Planning and Design**
   - Created a high-level architecture design to map out key components and their interactions.
   - Designed UI mockups for the main pages using tools like Figma (if applicable).
-
2. **Development**
   - Built the frontend using React/Next.js and implemented business logic in TypeScript.
   - Focused on a component-based structure for maximum reusability and ease of debugging.
-
3. **Integration of State Management**
   - Used React's Context API to manage and share states such as cart items or user settings globally.
   - Example: Added functions like addToCart(), removeFromCart(), and clearCart() to manage the cart dynamically.
-
4. **Testing**
   - Conducted extensive functional testing to ensure all features worked as expected.
   - Performed responsive testing using Chrome DevTools to ensure the design adapts to various screen sizes.
-
5. **Finalization**
   - Optimized the code for performance, ensuring quick load times and smooth transitions.
   - Added comments and documentation for easier understanding by other developers or collaborators.

Challenges and Solutions

Challenge: Ensuring persistent data without compromising app speed.

Solution: Used localStorage to save essential user data and optimized retrieval with React's useEffect hook.

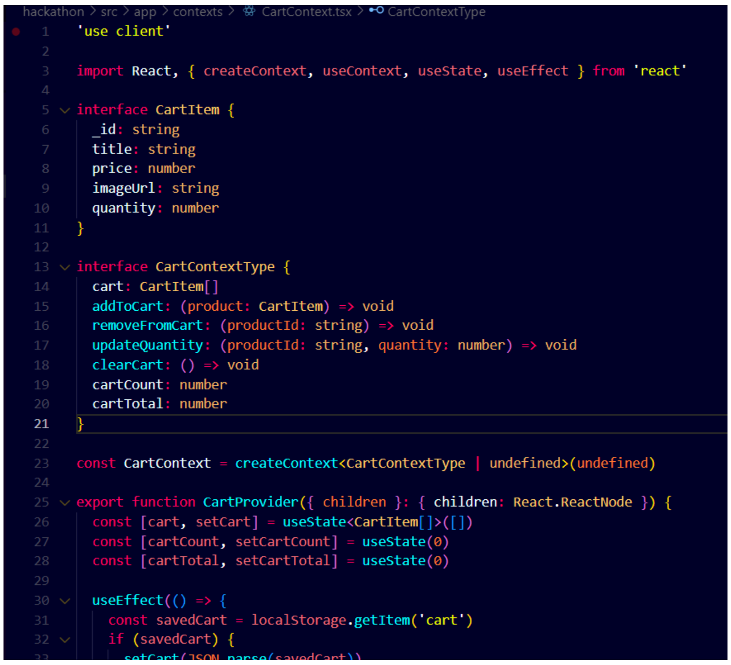
Challenge: Achieving a consistent design across all screen sizes.
Solution: Leveraged Tailwind CSS for utility-first styling and tested extensively on multiple devices.

Challenge: Synchronizing state across components.
Solution: Used Context API for global state sharing and avoided prop drilling by encapsulating functionality in reusable hooks.

Screenshots and Examples:

File Name: CartContext.tsx

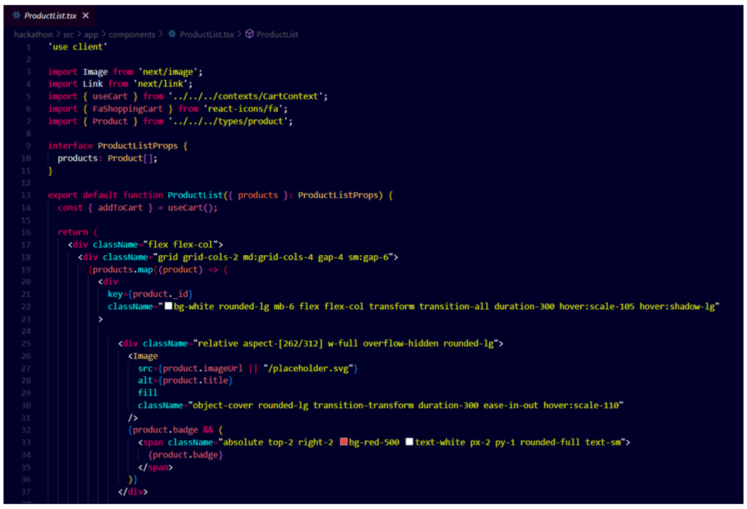
- Purpose: Manages all global cart-related states, including items, totals, and persistence.
- Example Code Snippet:


hackathon > src > app > contexts > CartContext.tsx > CartContextType

```tsx
'use client'

import React, { createContext, useContext, useState, useEffect } from 'react'

interface CartItem {
  _id: string
  title: string
  price: number
  imageUrl: string
  quantity: number
}

interface CartContextType {
  cart: CartItem[]
  addToCart: (product: CartItem) => void
  removeFromCart: (productId: string) => void
  updateQuantity: (productId: string, quantity: number) => void
  clearCart: () => void
  cartCount: number
  cartTotal: number
}

const CartContext = createContext<CartContextType | undefined>(undefined)

export function CartProvider({ children }: { children: React.ReactNode }) {
  const [cart, setCart] = useState<CartItem[]>([])
  const [cartCount, setCartCount] = useState(0)
  const [cartTotal, setCartTotal] = useState(0)

  useEffect(() => {
    const savedCart = localStorage.getItem('cart')
    if (savedCart) {
      setCart(JSON.parse(savedCart))
```

File Name: ProductList.tsx

- Purpose: Dynamically fetches and displays product cards using reusable components.



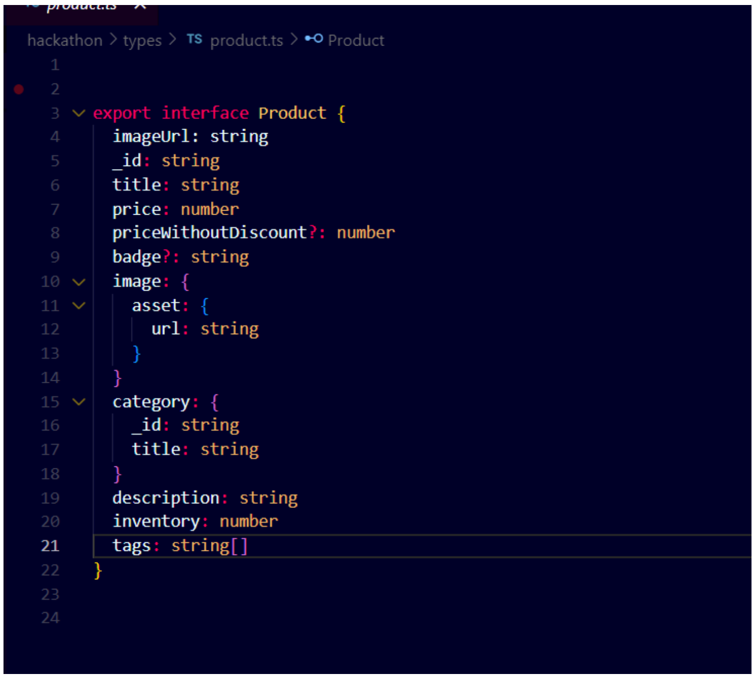
ProductList.tsx

hackathon > src > app > components > ProductList.tsx > ProductList

```tsx
'use client'

import Image from 'next/image';
import Link from 'next/link';
import { useCart } from '../../../contexts/CartContext';
import { FaShoppingCart } from 'react-icons/fa';
import { Product } from '../../../types/product';

interface ProductListProps {
  products: Product[];
}

export default function ProductList({ products }: ProductListProps) {
  const { addToCart } = useCart();

  return (
    <div className="flex flex-col">
      <div className="grid grid-cols-2 md:grid-cols-4 gap-4 sm:gap-6">
        {products.map((product) => (
          <div
            key={product._id}
            className="bg-white rounded-lg mb-6 flex flex-col transform transition-all duration-300 hover:scale-105 hover:shadow-lg"
          >

            <div className="relative aspect-[262/312] w-full overflow-hidden rounded-lg">
              <Image
                src={product.imageUrl || "/placeholder.svg"}
                alt={product.title}
                fill
                className="object-cover rounded-lg transition-transform duration-300 ease-in-out hover:scale-110"
              />
              {product.badge && (
                <span className="absolute top-2 right-2 bg-red-500 text-white px-2 py-1 rounded-full text-sm">
                  {product.badge}
                </span>
              )}
            </div>
```

File Name: product.ts

- Purpose: Entry point for the application with the main layout and component rendering.



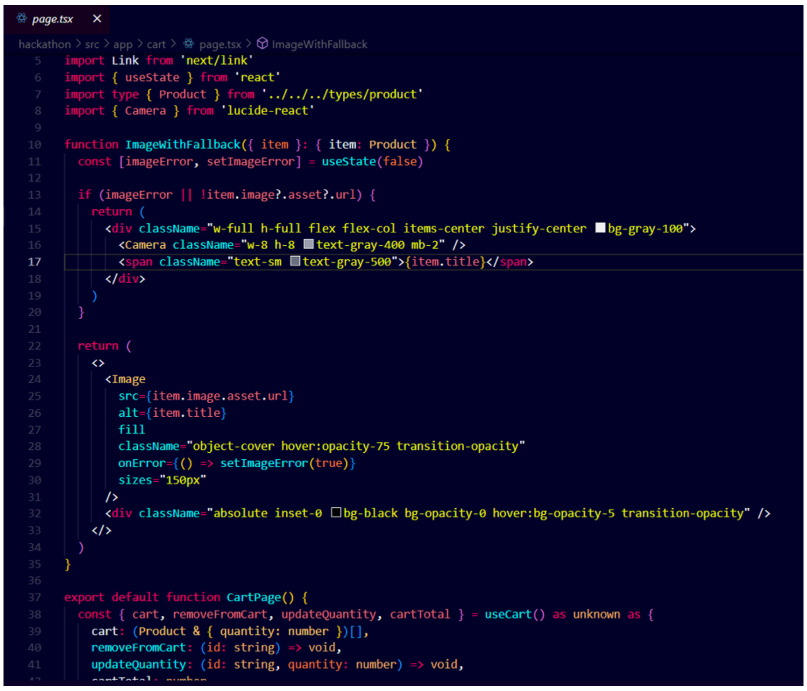
product.ts

hackathon > types > product.ts > Product

```ts


export interface Product {
  imageUrl: string
  _id: string
  title: string
  price: number
  priceWithoutDiscount?: number
  badge?: string
  image: {
    asset: {
      url: string
    }
  }
  category: {
    _id: string
    title: string
  }
  description: string
  inventory: number
  tags: string[]
}


```

Folder Name: cart

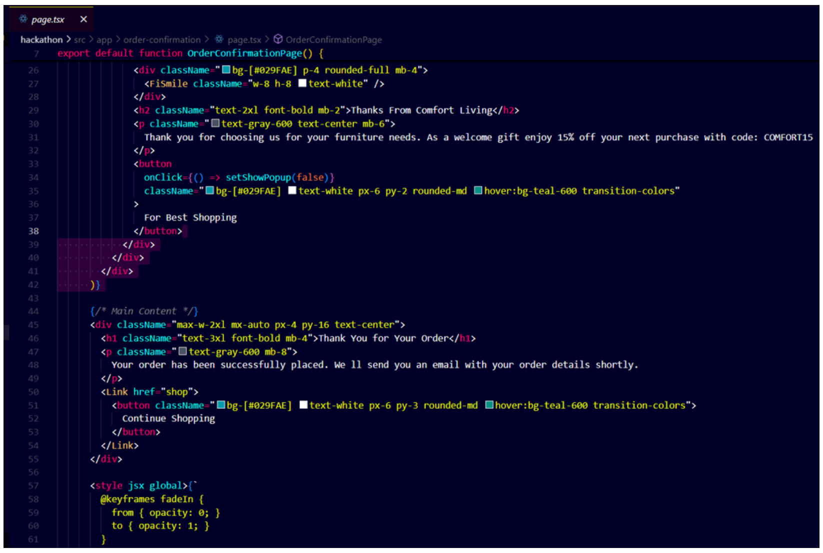
- Purpose: Add to cart of the products are shown in this file which carry the product all information in it .



```
page.tsx
hackathon > src > app > cart > page.tsx > ImageWithFallback
import Link from 'next/link'
import { useState } from 'react'
import type { Product } from '../../../types/product'
import { Camera } from 'lucide-react'

function ImageWithFallback({ item }: { item: Product }) {
  const [imageError, setImageError] = useState(false)

  if (imageError || !item.image?.asset?.url) {
    return (
      <div className="w-full h-full flex flex-col items-center justify-center bg-gray-100">
        <Camera className="w-8 h-8 text-gray-400 mb-2" />
        <span className="text-sm text-gray-500">{item.title}</span>
      </div>
    )
  }

  return (
    <>
      <Image
        src={item.image.asset.url}
        alt={item.title}
        fill
        className="object-cover hover:opacity-75 transition-opacity"
        onError={() => setImageError(true)}
        sizes="150px"
      />
      <div className="absolute inset-0 bg-black bg-opacity-0 hover:bg-opacity-5 transition-opacity" />
    </>
  )
}

export default function CartPage() {
  const { cart, removeFromCart, updateQuantity, cartTotal } = useCart() as unknown as {
    cart: (Product & { quantity: number })[],
    removeFromCart: (id: string) => void,
    updateQuantity: (id: string, quantity: number) => void,
```

Folder Name: order-confirmation

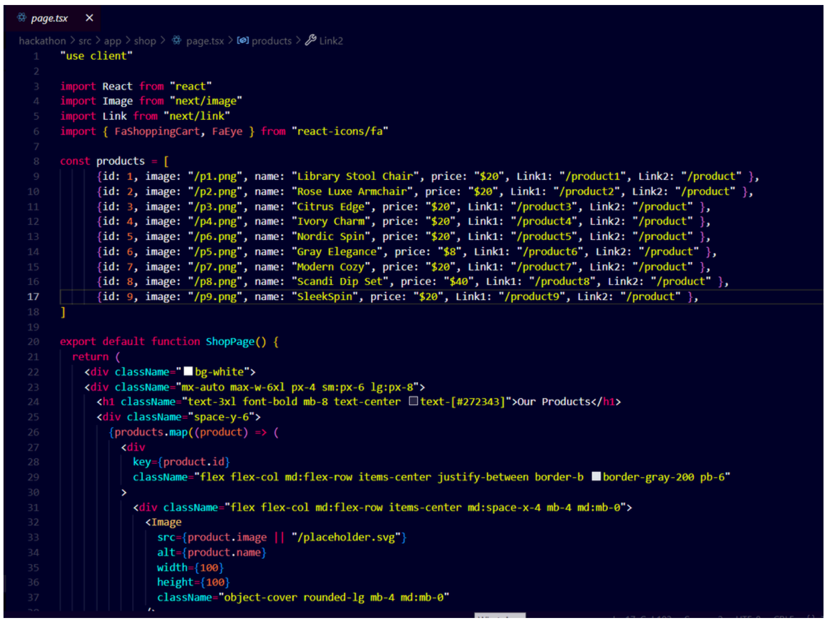
- Purpose: Order confirmation sheet is used to add all the confirmed and have a form of the shipping requirements .



```
page.tsx
hackathon > src > app > order-confirmation > page.tsx > OrderConfirmationPage
export default function OrderConfirmationPage() {
              <div className="bg-[#029FAE] p-4 rounded-full mb-4">
                <FiSmile className="w-8 h-8 text-white" />
              </div>
              <h2 className="text-2xl font-bold mb-2">Thanks From Comfort Living</h2>
              <p className="text-gray-600 text-center mb-6">
                Thank you for choosing us for your furniture needs. As a welcome gift enjoy 15% off your next purchase with code: COMFORT15
              </p>
              <button
                onClick={() => setShowPopup(false)}
                className="bg-[#029FAE] text-white px-6 py-2 rounded-md hover:bg-teal-600 transition-colors"
              >
                For Best Shopping
              </button>
            </div>
          </div>
        </div>
      )}

      {/* Main Content */}
      <div className="max-w-2xl mx-auto px-4 py-16 text-center">
        <h1 className="text-3xl font-bold mb-4">Thank You for Your Order</h1>
        <p className="text-gray-600 mb-8">
          Your order has been successfully placed. We ll send you an email with your order details shortly.
        </p>
        <Link href="shop">
          <button className="bg-[#029FAE] text-white px-6 py-3 rounded-md hover:bg-teal-600 transition-colors">
            Continue Shopping
          </button>
        </Link>
      </div>

      <style jsx global>{`
        @keyframes fadeIn {
          from { opacity: 0; }
          to { opacity: 1; }
        }
```

Folder Name: shop

- Purpose: Shop folder plays a main role of file who carry all the data of the product for the user first interaction.

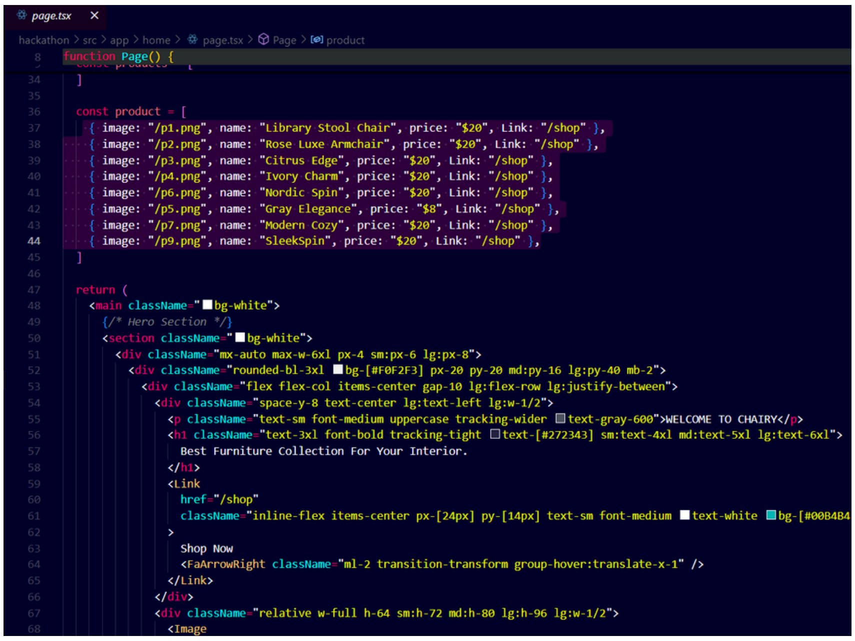

page.tsx

hackathon > src > app > shop > page.tsx > products > Link2

```tsx
"use client"

import React from "react"
import Image from "next/image"
import Link from "next/link"
import { FaShoppingCart, FaEye } from "react-icons/fa"

const products = [
      {id: 1, image: "/p1.png", name: "Library Stool Chair", price: "$20", Link1: "/product1", Link2: "/product" },
      {id: 2, image: "/p2.png", name: "Rose Luxe Armchair", price: "$20", Link1: "/product2", Link2: "/product" },
      {id: 3, image: "/p3.png", name: "Citrus Edge", price: "$20", Link1: "/product3", Link2: "/product" },
      {id: 4, image: "/p4.png", name: "Ivory Charm", price: "$20", Link1: "/product4", Link2: "/product" },
      {id: 5, image: "/p6.png", name: "Nordic Spin", price: "$20", Link1: "/product5", Link2: "/product" },
      {id: 6, image: "/p5.png", name: "Gray Elegance", price: "$8", Link1: "/product6", Link2: "/product" },
      {id: 7, image: "/p7.png", name: "Modern Cozy", price: "$20", Link1: "/product7", Link2: "/product" },
      {id: 8, image: "/p8.png", name: "Scandi Dip Set", price: "$40", Link1: "/product8", Link2: "/product" },
      {id: 9, image: "/p9.png", name: "SleekSpin", price: "$20", Link1: "/product9", Link2: "/product" },
]

export default function ShopPage() {
  return (
    <div className="bg-white">
    <div className="mx-auto max-w-6xl px-4 sm:px-6 lg:px-8">
      <h1 className="text-3xl font-bold mb-8 text-center text-[#272343]">Our Products</h1>
      <div className="space-y-6">
        {products.map((product) => (
          <div
            key={product.id}
            className="flex flex-col md:flex-row items-center justify-between border-b border-gray-200 pb-6"
          >
            <div className="flex flex-col md:flex-row items-center md:space-x-4 mb-4 md:mb-0">
              <Image
                src={product.image || "/placeholder.svg"}
                alt={product.name}
                width={100}
                height={100}
                className="object-cover rounded-lg mb-4 md:mb-0"
```

Folder Name: home

- Purpose: Home id the first page of the product where you know about the content of the website.


page.tsx

hackathon > src > app > home > page.tsx > Page > product

```tsx
function Page() {
  ]

  const product = [
    { image: "/p1.png", name: "Library Stool Chair", price: "$20", Link: "/shop" },
    { image: "/p2.png", name: "Rose Luxe Armchair", price: "$20", Link: "/shop" },
    { image: "/p3.png", name: "Citrus Edge", price: "$20", Link: "/shop" },
    { image: "/p4.png", name: "Ivory Charm", price: "$20", Link: "/shop" },
    { image: "/p6.png", name: "Nordic Spin", price: "$20", Link: "/shop" },
    { image: "/p5.png", name: "Gray Elegance", price: "$8", Link: "/shop" },
    { image: "/p7.png", name: "Modern Cozy", price: "$20", Link: "/shop" },
    { image: "/p9.png", name: "SleekSpin", price: "$20", Link: "/shop" },
  ]

  return (
    <main className="bg-white">
      {/* Hero Section */}
      <section className="bg-white">
        <div className="mx-auto max-w-6xl px-4 sm:px-6 lg:px-8">
          <div className="rounded-bl-3xl bg-[#F0F2F3] px-20 py-20 md:py-16 lg:py-40 mb-2">
            <div className="flex flex-col items-center gap-10 lg:flex-row lg:justify-between">
              <div className="space-y-8 text-center lg:text-left lg:w-1/2">
                <p className="text-sm font-medium uppercase tracking-wider text-gray-600">WELCOME TO CHAIRY</p>
                <h1 className="text-3xl font-bold tracking-tight text-[#272343] sm:text-4xl md:text-5xl lg:text-6xl">
                  Best Furniture Collection For Your Interior.
                </h1>
                <Link
                  href="/shop"
                  className="inline-flex items-center px-[24px] py-[14px] text-sm font-medium text-white bg-[#00B4B4
                >
                  Shop Now
                  <FaArrowRight className="ml-2 transition-transform group-hover:translate-x-1" />
                </Link>
              </div>
              <div className="relative w-full h-64 sm:h-72 md:h-80 lg:h-96 lg:w-1/2">
                <Image
```

Functionality Adding:

1. User Authentication and Authorization (Optional)

- **Objective:** Allow users to sign up, log in, and manage their profiles. This can enhance the shopping experience by remembering user preferences, past orders, and personal details.
- **Implementation:** You can use Firebase Authentication or JWT tokens for secure sign-ins. This will also allow users to save their cart items across sessions.

2. Advanced Cart Management

- **Objective:** Provide more control over the cart by allowing users to modify quantities, remove items, or view related product suggestions.
- **Implementation:**
    - Implement a feature to update the cart item quantity with real-time updates, so users can easily change their cart contents.
    - Use context hooks and local storage to ensure that cart data remains persistent even when the user navigates between pages.
    - Implement “Save for later” functionality, allowing users to remove items from the **cart** without deleting them entirely.

3. Product Search and Filter

- **Objective:** Allow users to search and filter products based on categories, prices, ratings, and more.
- **Implementation:**
    - Create a search bar component that filters products dynamically as the user types.
    - Provide filter options like price range, category, or brand using checkboxes or dropdowns.
    - Leverage the useEffect hook to update the product list based on filter criteria.

4. Product Detail Page

- **Objective:** Provide a dedicated page for each product with detailed information such as specifications, images, and user reviews.
- **Implementation:**
    - A product detail page can be created for each product, allowing users to view more information before adding it to the cart.
    - Include image sliders or modals to display multiple images of a product.

5. Checkout Process

- **Objective:** Provide an easy-to-navigate checkout flow with shipping and payment options.
- **Implementation:**
    - Use a form to capture user details like shipping address, payment method, and order summary.
    - Integrate a payment gateway like Stripe or PayPal for secure online payments.
    - Show an order summary before finalizing the purchase.

6. Order Confirmation and History

- **Objective:** Provide users with an order confirmation page and the ability to track order history.
- **Implementation:**
  - After completing a purchase, show an order confirmation page with a unique order ID.
  - Implement order history for logged-in users, displaying past orders and their statuses.
  - This can be achieved by saving order details in a database (using Firebase, MongoDB, or another backend service) and displaying them in the user's profile.

7. Admin Dashboard

- **Objective:** Allow administrators to manage the products, view orders, and track user activities.
- **Implementation:**
  - Build a separate admin dashboard with CRUD operations (Create, Read, Update, Delete) for managing products and orders.
  - Include analytics like total sales, active users, and product stock levels.

8. Mobile Optimization

- **Objective:** Ensure the application is optimized for mobile users to increase accessibility and user satisfaction.
- **Implementation:**
  - Test and optimize the design for mobile devices to ensure that the UI is intuitive and easy to navigate on smaller screens.
  - Use Tailwind CSS responsive classes to handle different screen sizes dynamically.
  - Add a mobile-friendly navigation menu (hamburger-style) for smaller devices.

9. SEO and Performance Optimization

- **Objective:** Improve search engine rankings and performance.
- **Implementation:**
  - Use Next.js's built-in SEO features (such as next/head) to ensure your pages are search engine optimized.
  - Implement lazy loading for images and other assets to ensure faster page loading times.
  - Use code splitting to load only the necessary JavaScript on each page, improving performance.

10. Security Best Practices

- **Objective:** Ensure user data and payment information are secure.
- **Implementation:**
  - Use HTTPS to encrypt data in transit.
  - Validate all input fields to prevent malicious data (like SQL injections or cross-site scripting attacks).
  - For payments, always use a trusted third-party service like Stripe or PayPal.

11. Internationalization (i18n)

- **Objective:** Support multiple languages to cater to a global audience.
- **Implementation:**
  - Use a package like react-i18next to implement language selection, allowing users to switch between different languages based on their preferences.

12. Unit and Integration Testing

- **Objective:** Ensure the application is bug-free and stable.
- **Implementation:**
  - Write unit tests for individual components and integration tests for the checkout flow.
  - Use tools like Jest and React Testing Library to simulate user interactions and ensure everything works as expected.

13. Social Sharing

- **Objective:** Allow users to share products with friends via social media platforms.
- **Implementation:**
  - Add social media share buttons (Facebook, Twitter, Pinterest) to product pages so users can easily share their favorite items.

How This Project is Unique

1. **Innovative Features:**
   - The use of persistent data storage ensures a smooth user experience.
   - Real-time state updates create a seamless and engaging application flow.
2. **Scalability:**
   - The modular architecture makes the app easy to scale and add new features in the future.

3. **Professional Design:**
    - Tailored UI/UX with a focus on accessibility, responsiveness, and modern design trends.

Conclusion

In this project, I applied advanced development techniques to create a robust and interactive application that meets modern user expectations. I prioritized performance, scalability, and user experience to ensure that the project is not only functional but also visually appealing and maintainable for future enhancements.