

# Day 4 - Dynamic Frontend Components - Avion

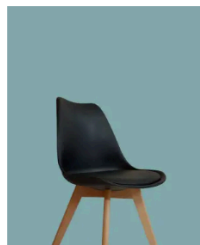
## 1. Functional Deliverables

### 1.1 Product Listing Page:

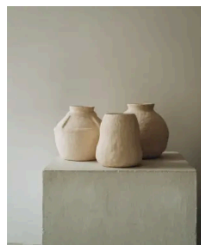
- Implemented a dynamic product listing page that fetches data from Sanity CMS.
- Features included:
  - ◆ Add to Cart functionality.
  - ◆ Wishlist functionality.
  - ◆ Integrated a search bar to dynamically search for products.

#### Screenshots:

- Screenshot showcasing the product listing page.



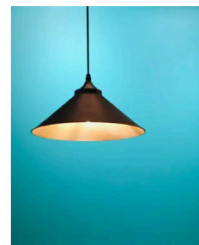
The Dandy chair  
£250



Rustic Vase Set  
£155



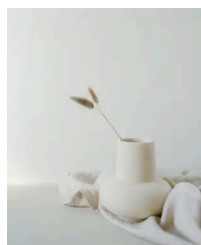
The Silky Vase  
£125



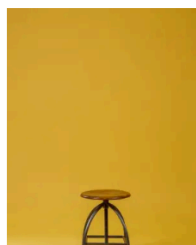
The Lucy Lamp  
£399



The Dandy chair  
£250



Rustic Vase Set  
£155



The Silky Vase  
£125



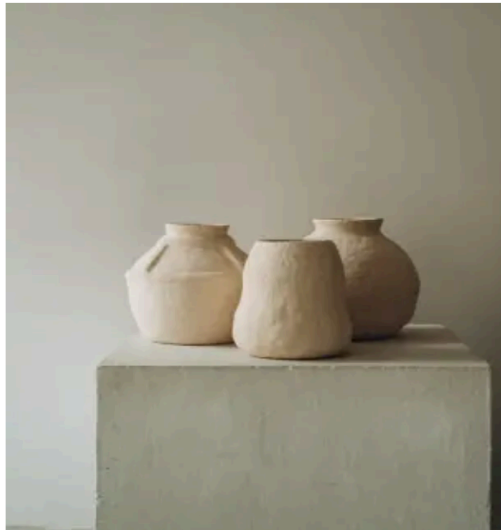
The Lucy Lamp  
£399

### 1.2 Individual Product Detail Pages:

- Developed dynamic routing for individual product pages using Next.js dynamic routes (`/product/[id]`).
- Displayed product-specific details like name, price, description, dimensions, and features.

#### Screenshot:

- Screenshot of a sample product detail page displaying product-specific data with accurate routing.



## Rustic Vase Set

£155

### Description

A beautifully crafted rustic vase set that brings a touch of nature into your home. Perfect for both modern and traditional interiors.

### Features

- Rustic charm
- Handcrafted design
- Durable materials

### Dimensions

| Height | Width | Depth |
|--------|-------|-------|
| 30cm   | 20cm  | 20cm  |

Amount:

Add to Cart

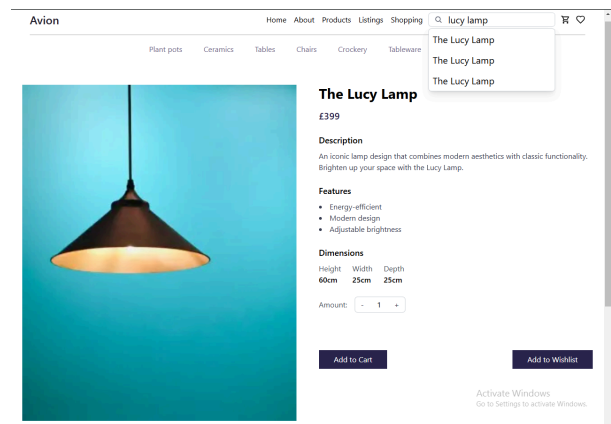
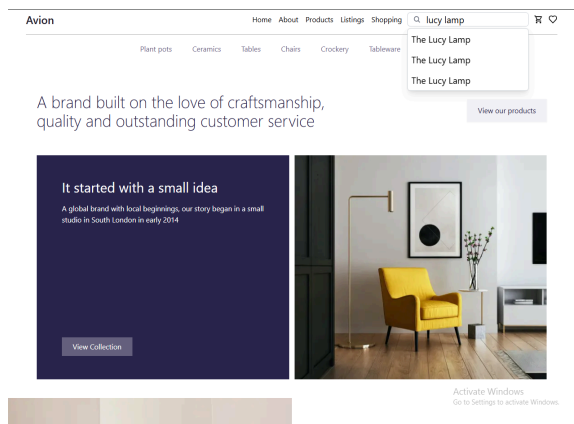
Add to Wishlist

## 1.3 Search Bar:

- ➔ Added a functional search bar in the header that dynamically fetches and filters products based on user input.
- ➔ Implemented a debounce mechanism to improve performance.

## Screenshot:

- ➔ Screenshot showing the working search bar with dynamic results.



## 1.4 Add to Cart and Wishlist Functionality:

- ➔ Integrated an Add to Cart button to allow users to add products to their cart.
- ➔ Implemented a Wishlist button to save favorite products for later.

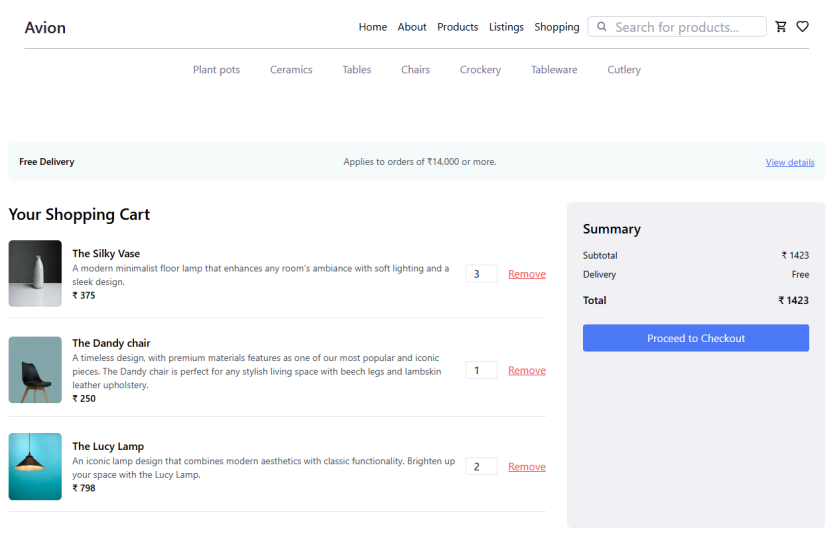
- ➔ Added a Checkout functionality to enable users to review their cart and complete their purchase.

## Screenshots:

- ➔ Screenshot showcasing the Add to Cart and Wishlist functionality in action.

### Add to cart:

```
File Edit Selection View ... my-web
src > app > cart > pages > ShoppingCart > ShoppingCart.jsx
40 "use client";
41
42 import Image from "next/image";
43 import Link from "next/link";
44 import { useRouter, useSearchParams } from "next/navigation";
45 import { useEffect, useState } from "react";
46
47 interface Product {
48   name: string;
49   price: string;
50   description: string;
51   image: string;
52   quantity: number;
53 }
54
55 export default function ShoppingCart() {
56   const router = useRouter();
57   const searchParam = useSearchParams();
58   const [cartItems, setCartItems] = useState<Product[]>([]);
59
60   // Fetch cart items from Local Storage and Add New Items from Query Params
61   useEffect(() => {
62     const cart = localStorage.getItem("cart");
63     const updatedCart = cart ? JSON.parse(cart) : [];
64
65     const name = searchParam.get("name");
66     const price = searchParam.get("price");
67     const description = searchParam.get("description");
68     const image = searchParam.get("image");
69
70     if (name && price && description && image) {
71       const isDuplicate = updatedCart.some(
72         (item: Product) => item.name === name
73       );
74
75       if (!isDuplicate) {
76         const newItem: Product = {
77           name,
78           price,
79           description,
80           image,
81           quantity: 1,
82         };
83         setCartItems(updatedCart.concat(newItem));
84         localStorage.setItem("cart", JSON.stringify(updatedCart.concat(newItem)));
85         router.replace("/cart"); // Remove query params
86       } else {
87         setCartItems(updatedCart.map((item) => {
88           if (item.name === name) {
89             return { ...item, quantity: item.quantity + 1 };
90           }
91           return item;
92         }));
93       }
94     }
95
96     // Remove item from cart
97     const handleRemoveItem = (index: number) => {
98       const updatedCart = [...cartItems];
99       updatedCart.splice(index, 1);
100       localStorage.setItem("cart", JSON.stringify(updatedCart));
101       setCartItems(updatedCart);
102     };
103
104     // Update item quantity
105     const handleQuantityChange = (index: number, quantity: number) => {
106       if (quantity < 1) return; // Ensure quantity is at least 1
107       const updatedCart = [...cartItems];
108       updatedCart[index].quantity = quantity;
109       localStorage.setItem("cart", JSON.stringify(updatedCart));
110       setCartItems(updatedCart);
111     };
112
113     // Calculate total price
114     const calculateTotalPrice = () => {
115       return cartItems.reduce(
116         (total, item) => total + item.price * item.quantity,
117         0
118       );
119     };
120
121     const total = calculateTotalPrice();
122     router.push(`/checkout?total=${total}`);
123   }, [searchParam, router, cartItems]);
124
125   return (
126     <div>
127       <h2>Shopping Cart</h2>
128       <table>
129         <thead>
130           <tr>
131             <th>Product</th>
132             <th>Quantity</th>
133             <th>Price</th>
134             <th>Total</th>
135             <th>Actions</th>
136           </tr>
137         </thead>
138         <tbody>
139           {cartItems.map((item, index) => (
140             <tr>
141               <td>
142                 <div>
143                   <img alt={item.image} />
144                   <div>
145                     <strong>{item.name}</strong>
146                     <p>{item.description}</p>
147                     <p>₹ {item.price}</p>
148                   </div>
149                 </div>
150               </td>
151               <td>
152                 <div>
153                   <input type="text" value={item.quantity} />
154                   <button>Remove</button>
155                 </div>
156               </td>
157               <td>₹ {item.price}</td>
158               <td>₹ {item.price * item.quantity}</td>
159               <td></td>
160             </tr>
161           ))}
162         </tbody>
163       </table>
164       <div>
165         <strong>Subtotal</strong> ₹ {total}
166         <strong>Delivery</strong> Free
167         <strong>Total</strong> ₹ {total}
168         <button>Proceed to Checkout</button>
169       </div>
170     </div>
171   );
172 }
```



## Checkout:

### How would you like to get your order?

Customs regulation for India require a copy of the recipient's KYC. The address on the KYC needs to match the shipping address.

☐ Deliver It

### Enter your name and address:

First Name

Last Name

Address Line 1

Address Line 2

Postal Code

Locality

State/Territory

### What's your contact information?

Email

Phone Number


### What's your PAN?

PAN


### Order Summary

|                   |               |
|-------------------|---------------|
| Subtotal          | ₹ 1423        |
| Delivery/Shipping | Free          |
| <b>Total</b>      | <b>₹ 1423</b> |


(Includes all duties and taxes)



**The Silky Vase**  
Qty: 3  
₹ 375



**The Dandy chair**  
Qty: 1  
₹ 250



**The Lucy Lamp**  
Qty: 2  
₹ 798

Let's Pay

Activate Windows

Go to Settings to activate Windows.

## Wishlist:

The screenshot displays a web browser window with a shopping cart page. The browser's address bar shows the URL `http://localhost:3000/`. The page title is "Shopping Cart". The main content area shows a list of items in the cart, each with a description, price, and quantity. The items are:

- Item 1: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 2: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 3: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 4: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 5: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 6: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 7: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 8: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 9: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 10: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 11: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 12: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 13: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 14: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 15: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 16: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 17: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 18: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 19: "A pair of shoes" with a price of \$120.00 and a quantity of 1.
- Item 20: "A pair of shoes" with a price of \$120.00 and a quantity of 1.

The total price of the items is \$2400.00. The page also includes a "Wishlist" section with a list of items and a "Total" section showing the total price and a "Checkout" button.

The browser's developer tools are open, showing the "Elements" panel. The selected element is a `div` with the class `flex-items-center` and the text `Shopping Cart`. The "Console" panel is also open, showing a message: `src > app > wishlist > page.txt > ...`

### 1.5 Steps Taken to Build and Integrate Components:

- Designed the `ProductCard` component with Add to Cart and Wishlist buttons.
- Implemented the `SearchBar` component for dynamic product searching.
- Developed dynamic routing for individual product pages using Next.js.
- Tested Add to Cart and Wishlist functionality using local storage for persistence.

### 1.6 Challenges Faced and Solutions:

- **Challenge:** Ensuring persistence for cart and wishlist data.
  - ◆ **Solution:** Used `localStorage` to store and retrieve data.
- **Challenge:** Optimizing search functionality for large datasets.
  - ◆ **Solution:** Added a debounce mechanism to minimize API calls.
- **Challenge:** Ensuring accurate routing for product detail pages.
  - ◆ **Solution:** Verified data fetching for each product ID using dynamic routes.

### 1.7 Best Practices Followed:

- Modularized components for better reusability.
- Used local storage for lightweight and persistent data handling.
- Tested all features for responsiveness and functionality across devices.

## Conclusion

The dynamic frontend components, including Add to Cart, Wishlist, Search Bar, and Individual Product Pages, significantly enhance the user experience for the Avion marketplace. These features are scalable and set a strong foundation for further enhancements.

### Self validation Checklist:

- ☑ **Frontend Component Development**
- ☑ **Styling and Responsiveness**
- ☑ **Code Quality**
- ☑ **Documentation and Submission**
- ☑ **Final Review**