

Day-4: “Dynamic Frontend Components For MarketPlace (Beadedwear)”

In this document I have described the libraries used for the implementation of my frontend components

Components:

1. Header
2. Footer
3. Contact Form
4. Cart
5. Category related Product List
6. Loader

- **Header:**

```

header.tsx M X
src > components > layout > header.tsx > ...
1  'use client'
2
3  import React, { useContext } from "react";
4  import Link from "next/link";
5  import Image from "next/image";
6  import { FiShoppingBag } from "react-icons/fi";
7  import Cart from './cart';
8  import { CartContext } from '../app/context/CartContext';
9
10 export default function Header() {
11
12   const {totalQuantity, showCart, setShowCart}:any = useContext(CartContext);
13
14   const handleClick = () => {
15     setShowCart(!showCart)
16   }
17
18   return (
19     <div className='overflow-clip inset-0 -z-10 h-full w-full bg-[#fafafa]
20       bg-[linear-gradient(to_right,#808080a_1px,transparent_1px),
21         linear-gradient(to_bottom,#808080a_1px,transparent_1px)]
22       bg-[size:14px_24px] pb-6'>
23       <header className="overflow-hidden rounded-[6px] top-5 md:mx-auto z-50
24         xl:w-4/5 2xl:w-[68%] bg-[#d4e3ff] flex items-center
25         justify-between py-4 px-4 md:px-6 mx-6">
26         <Link href="/">
27           <Image
28             src={"/Logo.png"}
29             alt="Logo"
30             width={500}
31             height={500}
32             className="w-16"
33           />
34         </Link>
35         <div className="absolute right-1/2 translate-x-1/2 transform">
36           <div className="hidden md:flex gap-x-10 items-center text-gray-700
37             font-medium text-lg cursor-pointer">
38             <Link href="/" className="hover:text-blue-500">
39               Home
40             </Link>
41             <Link href="/about" className="hover:text-blue-500">
42               About
43             </Link>
44             <Link href="/productsList" className="hover:text-blue-500">
45               Product
46             </Link>
47             <Link href="/categories" className="hover:text-blue-500">
48               Category
49             </Link>
50             <Link href="/contact" className="hover:text-blue-500">
51               Contact
52             </Link>
53           </div>
54         </div>
55         <button className='cart-icon' onClick={handleClick}>
56           <FiShoppingBag />
57           <span className='cart-item-qty'>{totalQuantity}</span>
58         </button>
59       </header>
60
61       {showCart && <Cart />}
62     </div>
63   );
64 }
65
66

```

- Footer:

```

src > components > layout > @ footer > @ Footer
1 import React from "react";
2 import Link from "next/link";
3 import Image from "next/image";
4
5 export default function Footer() {
6   return (
7     <footer className="text-gray-500 body-font bg-[#d4e3ff]">
8       <div className="container px-5 py-8 mx-auto flex items-center sm:flex-row flex-col">
9         <div className="flex title-font font-medium items-center md:justify-start justify-center text-gray-900">
10           <Link href="/">
11             <Image
12               src="/Logo.png"
13               alt="Logo"
14               width={500}
15               height={500}
16               className="w-16"
17             />
18           </Link>
19           <span className="ml-3 text-xl text-pink-600">Beaded Wear</span>
20         </div>
21         <p className="text-sm text-gray-500 sm:ml-4 sm:pl-4 sm:border-l-2 sm:border-gray-200 sm:py-2 sm:pt-0 sm:mt-4">
22           © 2025 Beaded Wear =
23           <Link
24             href="/"
25             className="text-gray-500 ml-1"
26             rel="noopener noreferrer"
27             target="_blank"
28           >
29             @BeadedWear
30           </Link>
31         </p>
32         <span className="inline-flex sm:ml-auto sm:mt-0 mt-4 justify-center sm:justify-start">
33           <div className="text-gray-500">
34             <svg
35               fill="currentColor"
36               strokeLinecap="round"
37               strokeLinejoin="round"
38               strokeWidth={2}
39               className="w-5 h-5"
40               viewBox="0 0 24 24"
41             >
42               <path d="M12 2h3a5 5 0 0 5 5v3h7v4h3v8h4v-4h-4v1a 1 0 0 1 1h3z" />
43             </svg>
44           </div>
45           <div className="ml-3 text-gray-500">
46             <svg
47               fill="currentColor"
48               strokeLinecap="round"
49               strokeLinejoin="round"
50               strokeWidth={2}
51               className="w-5 h-5"
52               viewBox="0 0 24 24"
53             >
54               <path d="M12 3a10 10 0 0 0 11.54 5 4 4 0 0 0 0 0-.88-.83a7.72 7.72 0 0 2 3z" />
55             </svg>
56           </div>
57           <div className="ml-3 text-gray-500">
58             <svg
59               fill="none"
60               stroke="currentColor"
61               strokeLinecap="round"
62               strokeLinejoin="round"
63               strokeWidth={2}
64               className="w-5 h-5"
65               viewBox="0 0 24 24"
66             >
67               <rect width={20} height={20} x={2} y={2} rx={5} ry={5} />
68               <path d="M16 11.37A4 4 0 1 1 11.37 16.63" />
69             </svg>
70           </div>
71           <div className="ml-3 text-gray-500">
72             <svg
73               fill="currentColor"
74               stroke="currentColor"
75               strokeLinecap="round"
76               strokeLinejoin="round"
77               strokeWidth={2}
78               className="w-5 h-5"
79               viewBox="0 0 24 24"
80             >
81               <circle cx={4} cy={4} r={2} stroke="none" />
82             </svg>
83             <div>
84               <path
85                 d="M16 8a6 6 0 0 1 6 6v7h-4v-7a2 2 0 0 2 2v7h-4v-7a6 6 0 0 1 6-6"
86               />
87             </div>
88           </div>
89         </span>
90       </div>
91     </footer>
92   );
93 }
94
95

```

- **Conatct From:**

Used zod and react-hook-form for validations,

```

1  @username      | @contact-forms M X
2
3  src > components > layout > @contact-forms > @ContactForm
4
5  1 "use client"
6
7  2
8  3 import React from "react"
9  4 import { z } from "zod"
10 5 import { zodResolver } from "hookform/resolvers/zod"
11 6 import { useForm } from "react-hook-form"
12 7 import { Button } from "@components/ui/button"
13 8 import {
14 9   form,
15 10   FormControl,
16 11   FormDescription,
17 12   FormField,
18 13   FormItem,
19 14   FormLabel,
20 15   FormMessage,
21 16 } from "@components/ui/form"
22 17 import { Input } from "@components/ui/input"
23 18 import { client } from "@sanity/lib/client"
24 19
25 20 const formSchema = z.object({
26 21   name: z.string().min(2, { message: "must be at least 2 characters" }), max(50),
27 22   email: z.string().email(),
28 23   message: z.string(),
29 24 })
30
31 25
32 26 function ContactForm() {
33 27
34 28   const form = useForm<z.infer<typeof formSchema>>>({
35 29     resolver: zodResolver(formSchema),
36 30     defaultValues: {
37 31       name: "",
38 32       email: "",
39 33       message: "",
40 34     },
41 35   })
42 36
43 37
44 38   async function onSubmit(values: z.infer<typeof formSchema>) {
45 39     try {
46 40       await client.create({
47 41         type: "contactForm",
48 42         name: values.name,
49 43         email: values.email,
50 44         message: values.message,
51 45       })
52 46     } catch (error) {
53 47       console.error("Error submitting the form:", error)
54 48       alert("Failed to submit the message. Please try again.")
55 49     }
56 50   }
57 51
58 52   return (
59 53     <form {...form}>
60 54       <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-8 grid justify-center items-center">
61 55         <div className="flex flex-col-1 md:flex-row-2 <input type="text" value={form.getFormState().errors.name ? " " : ""} className="w-full md-1 p-3 rounded-xl focus:outline-none focus:ring-2 focus:ring-teal-500" />
62 56         <div className="flex flex-col md:flex-row gap-4 md-4">
63 57           <formField>
64 58             <formControl>
65 59               <formLabel>
66 60                 name
67 61               </formLabel>
68 62               <formItem>
69 63                 <input type="text" value={form.getFormState().errors.name ? " " : ""} className="w-full md-1 p-3 rounded-xl focus:outline-none focus:ring-2 focus:ring-teal-500" />
70 64               </formControl>
71 65               <formMessage>
72 66                 {form.getFormState().errors.name ? form.getFormState().errors.name.message : ""}
73 67               </formMessage>
74 68             </formItem>
75 69           </formField>
76 70           <formField>
77 71             <formControl>
78 72               <formLabel>
79 73                 email
80 74               </formLabel>
81 75               <formItem>
82 76                 <input type="text" value={form.getFormState().errors.email ? " " : ""} className="w-full md-1 p-3 rounded-xl focus:outline-none focus:ring-2 focus:ring-teal-500" />
83 77                 <div>
84 78                   <formControl>
85 79                     <formLabel>
86 80                       message
87 81                     </formLabel>
88 82                     <formItem>
89 83                       <input type="text" value={form.getFormState().errors.message ? " " : ""} className="w-full md-1 p-3 rounded-xl focus:outline-none focus:ring-2 focus:ring-teal-500" />
90 84                       <div>
91 85                         <formControl>
92 86                           <formLabel>
93 87                             Submit
94 88                           </formLabel>
95 89                           <formItem>
96 90                             <button type="submit" value={form.getFormState().errors.submit ? " " : ""} className="w-full md-1 p-3 rounded-xl focus:outline-none focus:ring-2 focus:ring-teal-500" />
97 91                             </formItem>
98 92                           </formControl>
99 93                         </div>
100 94                       </formItem>
101 95                     </formControl>
102 96                     <formMessage>
103 97                       {form.getFormState().errors.submit ? form.getFormState().errors.submit.message : ""}
104 98                     </formMessage>
105 99                   </formItem>
106 100                 </div>
107 101               </formField>
108 102             </formField>
109 103           </div>
110 104         </form>
111 105       </form>
112 106     )
113 107   }
114 108
115 109   export default ContactForm
116 110
117 111
118 112
119 113
120 114

```

- **Cart:**

In the Cart I managed the real time changes using react states createContext and useState for quantity, price calculations , add and remove etc

```

1  use client";
2
3  import { createContext, useState } from "react";
4
5  export const CartContext = createContext({});
6
7  export const CartProvider = ({ children }: any) => {
8    const [showCart, setShowCart] = useState(false);
9    const [qty, setQty] = useState(1);
10   const [cartItems, setCartItems] = useState<any[]>([]);
11   const [totalQuantity, setTotalQuantity] = useState(0);
12   const [totalPrice, setTotalPrice] = useState(0);
13
14   const incQty = () => {
15     setQty((prevQty) => prevQty + 1);
16   };
17
18   const decQty = () => {
19     setQty((prevQty) => {
20       if (prevQty - 1 < 1) return 1;
21       return prevQty - 1;
22     });
23   };
24
25   const addProduct = (product: any, quantity: number) => {
26     const checkProductInCart = cartItems.find(
27       (item: any) => item._id === product._id
28     );
29     setTotalQuantity((prev) => prev + quantity);
30     setTotalPrice(
31       (prevTotalPrice) => prevTotalPrice + product.price * quantity
32     );
33
34     if (checkProductInCart) {
35       const updatedCartItems = cartItems.map((cartProduct: any) => {
36         if (cartProduct._id === product._id) {
37           return {
38             ...cartProduct,
39             quantity: cartProduct.quantity + quantity,
40           };
41         } else {
42           return cartProduct;
43         }
44       });
45       setCartItems(updatedCartItems);
46     } else {
47       product.quantity = quantity;
48       setCartItems([...cartItems, { ...product }]);
49     }
50   };
51
52   const toggleCartItemQty = (id: any, value: any) => {
53     let foundProduct = cartItems.find((item) => item._id === id);
54     const index = cartItems.findIndex((product) => product._id === id);
55     const updatedCartItems = [...cartItems];
56
57     if (value === "plus") {
58       updatedCartItems[index] = {
59         ...updatedCartItems[index],
60         quantity: updatedCartItems[index].quantity + 1,
61       };
62       setCartItems([...updatedCartItems]);
63       setTotalPrice((prevTotalPrice) => prevTotalPrice + foundProduct.price);
64       setTotalQuantity((prevTotalQty) => prevTotalQty + 1);
65     } else if (value === "minus") {
66       if (foundProduct.quantity > 1) {
67         updatedCartItems[index] = {
68           ...updatedCartItems[index],
69           quantity: updatedCartItems[index].quantity - 1,
70         };
71       }
72       setCartItems([...updatedCartItems]);
73       setTotalPrice((prevTotalPrice) => prevTotalPrice - foundProduct.price);
74       setTotalQuantity((prevTotalQty) => prevTotalQty - 1);
75     }
76   };
77
78   const onRemove = (product: any) => {
79     let foundProduct = cartItems.find((item) => item._id === product._id);
80     const newCartItems = cartItems.filter((item) => item._id !== product._id);
81
82     setCartItems(newCartItems);
83     setTotalPrice(
84       (prevTotal) => prevTotal - foundProduct.price * foundProduct.quantity
85     );
86     setTotalQuantity((prevTotalQty) => prevTotalQty - foundProduct.quantity);
87   };
88
89   return (
90     <CartContext.Provider
91       value={{
92         onRemove,
93         toggleCartItemQty,
94         totalPrice,
95         totalQuantity,
96         showCart,
97         setShowCart,
98         qty,
99         incQty,
100        decQty,
101        cartItems,
102        addProduct,
103      }}
104     >
105       <div>{children}</div>
106     </CartContext.Provider>
107   );
108 };
109

```

```

src > components > layout > @ cartItem ~
1 | import React, { useState } from "react";
2 | import { AIOutlineLeft, AIOutlineMinus, AIOutlinePlus } from "react-icons/ai";
3 | import { TDeleteOutline } from "react-icons/ti";
4 | import { CartContext } from "../app/context/CartContext";
5 | import Image from "next/image";
6 | import { urlFor } from "lib/utils/lib/image";
7 | import { toast } from "react-hot-toast";
8 |
9 | const Cart = () => {
10 |   const {
11     cartItems,
12     toggleCartItemQty,
13     totalPrice,
14     totalQuantity,
15     cartItems,
16     showCart,
17     setShowCart,
18   } = any ? useContext(CartContext);
19 |
20 |   const handleClose = () => {
21     setShowCart(!showCart);
22   };
23 |
24 |   const handleCheckout = async () => {
25 |     try {
26 |       // const response = await fetch("/api/checkout",{
27 |         // method: "POST",
28 |         // headers:{
29 |           // "Content-Type":"application/json"
30 |         // },
31 |         // body:JSON.stringify({product:cartItems}),
32 |         // });
33 |       // const data = await response.json();
34 |       // if(data.url){
35 |         // window.location.href = data.url
36 |       // }
37 |       toast.success("Happy Shopping!",{
38         position: "top-right",
39         duration: 3000, // Toast duration in milliseconds
40       });
41 |     } catch (error) {
42 |       console.error("Error during checkout", error);
43 |     }
44 |   };
45 |
46 |   return (
47 |     <div className="cart-wrapper">
48 |       <div className="cart-container bg-cart">
49 |         <button className="cart-heading" onClick={handleClose}>
50 |           AIOutlineLeft
51 |         </button>
52 |         <span className="heading">Your Cart</span>
53 |         <span className="cart-sub-items"><totalQuantity></span>
54 |         </button>
55 |         <div className="product-container">
56 |           {cartItems.map((product:any) => {
57 |             <div className="product">key[product._id]>
58 |               <img
59 |                 src={urlFor(product.image1).url()}
60 |                 alt={product.name}
61 |                 width={150}
62 |                 height={80}
63 |                 className="object-cover"
64 |               />
65 |               <div className="item-desc">
66 |                 <div className="flex-row top">
67 |                   <div><id>(product.name)</div>
68 |                   <div><id>(product.price)</div>
69 |                 </div>
70 |                 <div>
71 |                   <div className="flex gap-1 button">
72 |                     <div className="quantity-desc">
73 |                       <span>
74 |                         className="minus"
75 |                         onClick={() => toggleCartItemQty(product._id, "minus")}
76 |                       </span>
77 |                       <AIOutlineMinus />
78 |                       <span>
79 |                         className="num">(product.quantity)</span>
80 |                       <span>
81 |                         className="plus"
82 |                         onClick={() => toggleCartItemQty(product._id, "plus")}
83 |                       </span>
84 |                       <AIOutlinePlus />
85 |                     </span>
86 |                     <div>
87 |                       <button>
88 |                         type="button"
89 |                         onClick={() => remove(product)}
90 |                         className="remove-item"
91 |                       </button>
92 |                     </div>
93 |                   </div>
94 |                 </div>
95 |               </div>
96 |             )
97 |           })
98 |           {cartItems.length > 0 ? (
99 |             <div className="cart-bottom">
100 |               <div class="total">
101 |                 <div><id>Subtotal</div>
102 |                 <div><id>(totalPrice) PER</div>
103 |               </div>
104 |               <div className="btn-container">
105 |                 <button>
106 |                   onClick={handleCheckout}
107 |                   type="button"
108 |                   className="checkbox-out-btn"
109 |                 </button>
110 |                 Checkout
111 |               </button>
112 |             </div>
113 |             </div>
114 |           ) : (
115 |             </div>
116 |           )
117 |         </div>
118 |       </div>
119 |     );
120 |   }
121 | export default Cart;

```

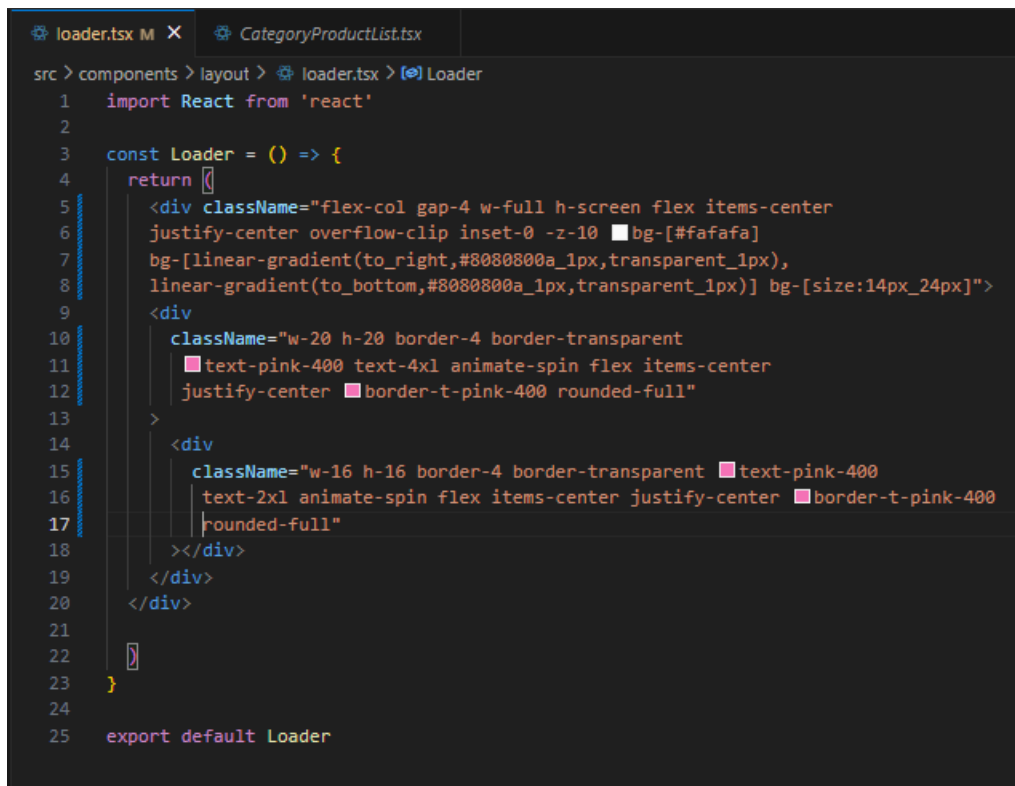
- **Category Product:**

Made dynamic categories page and fetched products for the desired category from sanity with state management using react hooks (useState and useEffect),

```
loader.tsx CategoryProductList.tsx X
src > components > layout > CategoryProductList.tsx > ...
1 import React, { useEffect, useState } from "react";
2 import { client } from "@sanity/lib/client";
3 import imageUrlBuilder from "@sanity/image-url";
4 import Loader from "../loader"
5
6 const builder = imageUrlBuilder(client);
7
8 const urlFor = (source: any) => builder.image(source);
9
10 interface Product {
11   name: string;
12   imageUrl: string;
13   price: number;
14   slug: {
15     current: string;
16   };
17   category: {
18     name: string;
19     slug: {
20       current: string;
21     };
22   };
23 }
24
25 const CategoryProductList = ({ slug }: { slug: string }) => {
26   const [products, setProducts] = useState<Product[] | null>(null);
27
28   useEffect(() => {
29     const fetchProducts = async () => {
30       const query = `
31         *[_type == "product" && category->slug.current == $slug]{
32           name,
33           "imageUrl": image.asset->url,
34           price,
35           category->{
36             name,
37             slug
38           },
39           slug {
40             current
41           }
42         }
43       `;
44       const result = await client.fetch(query, { slug });
45       setProducts(result);
46     };
47
48     fetchProducts();
49   }, [slug]);
50
51   // Loading state
52   if (!products) return <Loader />;
53
54   // If no products are found
55   if (products.length === 0)
56     return <div>No products found for this category.</div>;
57
58   // Render products
59   return (
60     <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
61       {products.map((product) => (
62         <div
63           key={product.name}
64           className="border p-4 rounded-lg shadow-md bg-white"
65         >
66           <img
67             src={urlFor(product.imageUrl).url()}
68             alt={product.name}
69             className="w-full h-40 object-cover rounded"
70           />
71           <h2 className="text-xl font-semibold mt-2">{product.name}</h2>
72           <p className="text-gray-600">{product.price} PKR</p>
73         </div>
74       ))}
75     </div>
76   );
77
78   export default CategoryProductList;
79
80
```

- **Loader:**

Made a customize Loader Component for interactive user interface,



```
src > components > layout > loader.tsx > Loader
1  import React from 'react'
2
3  const Loader = () => {
4    return (
5      <div className="flex-col gap-4 w-full h-screen flex items-center
6        justify-center overflow-clip inset-0 -z-10 bg-[#fafafa]
7        bg-[linear-gradient(to_right,#808080a_1px,transparent_1px),
8        linear-gradient(to_bottom,#808080a_1px,transparent_1px)] bg-[size:14px_24px]">
9        <div
10          className="w-20 h-20 border-4 border-transparent
11            text-pink-400 text-4xl animate-spin flex items-center
12            justify-center border-t-pink-400 rounded-full"
13          >
14            <div
15              className="w-16 h-16 border-4 border-transparent text-pink-400
16                text-2xl animate-spin flex items-center justify-center border-t-pink-400
17                rounded-full"
18            ></div>
19          </div>
20        </div>
21      )
22    }
23  }
24
25  export default Loader
```

Challenges And Solutions:

It was hard to update the cart dynamically using props as there were multiple things being updated at the same time like change in the total price, add and subtract quantity from cart directly moreover when to show the cart ..etc

Here I do my research and came across to a react function `createContext()`, In React the `createContext()` function is used to create a Context that enables sharing values (like data or functions) between components without having to pass props down manually at every level. This is especially useful for managing global state or shared values.

Document by Ilsa Ubaid(senior student GIAIC)