Day 4 - Dynamic Frontend Components [General E-Commerce]
Video Link : Screen Recording 2025-02-08 101052.mp4
Code Snippets :
Cart Page

```
'use client';
import { useS'use client';
import { useState, useEffect } from 'react';
import Image from 'next/image';
import Link from 'next/link';
import { urlFor } from '@/sanity/lib/image'; // Import Sanity image
builder
import { useRouter } from 'next/navigation';

const ShoppingCartPage = () => {
  const [cart, setCart] = useState<any[]>([]);
  const [total, setTotal] = useState(0);
  const [deliveryFee, setDeliveryFee] = useState(500); // Default ₹500
if below ₹14,000
  const router = useRouter();

  // Fetch cart data from localStorage
  useEffect(() => {
    try {
      const storedCart = JSON.parse(localStorage.getItem('cart') ||
'[]');
      if (Array.isArray(storedCart)) {
        setCart(storedCart);
        calculateTotal(storedCart);
      } else {
        console.error('Invalid cart data');
        setCart([]);
      }
    } catch (error) {
      console.error('Error parsing cart data', error);
      setCart([]);
    }
  }, []);

  // Calculate total price of the cart
  const calculateTotal = (cartItems: any[]) => {
    const totalPrice = cartItems.reduce(
      (acc, item) => acc + item.price * (item.quantity || 0),
      0
```

```
    );

    setTotal(totalPrice);
    setDeliveryFee(totalPrice >= 14000 ? 0 : 500); // Free delivery if
₹14,000 or more
  };

  // Remove item from cart
  const removeFromCart = (id: string) => {
    const updatedCart = cart.filter((item) => item.id !== id);
    setCart(updatedCart);
    localStorage.setItem('cart', JSON.stringify(updatedCart));
    calculateTotal(updatedCart);
  };

  // Handle item quantity change
  const changeQuantity = (id: string, type: 'increment' | 'decrement')
=> {
    const updatedCart = cart.map((item) => {
      if (item.id === id) {
        if (type === 'increment') {
          item.quantity = (item.quantity || 0) + 1;
        } else if (type === 'decrement' && item.quantity > 1) {
          item.quantity -= 1;
        }
      }
      return item;
    });
    setCart(updatedCart);
    localStorage.setItem('cart', JSON.stringify(updatedCart));
    calculateTotal(updatedCart);
  };

  // Handle checkout
  const handleCheckout = () => {
    if (cart.length === 0) {
      alert('Your cart is empty!');
      return;
    }

    const finalAmount = total + deliveryFee;
    alert(`Proceeding to checkout. Total: ₹${finalAmount}`);
```

```
    // Navigate to a checkout page
    router.push('/checkout');
  };

  if (cart.length === 0)
    return <p className="text-center text-xl mt-12">Your cart is
empty.</p>;

  return (
    <div className="flex flex-col bg-white mt-[150px] px-4 sm:px-6
lg:px-16">
      {/* Free Delivery Section */}
      <div className="bg-gray-100 p-4 text-sm text-gray-600 mb-8
border-t border-b border-gray-200">
        {total >= 14000 ? (
          <p className="text-green-600 font-semibold">
            You qualify for free delivery!
          </p>
        ) : (
          <p>
            Free Delivery applies to orders of ₹14,000.00 or more.{' '}
            <a href="#" className="underline">
              View details
            </a>
          </p>
        )}
      </div>

      {/* Main Container */}
      <div className="flex flex-col lg:flex-row gap-8">
        {/* Left Section - Bag Items */}
        <div className="flex-1">
          <h1 className="text-2xl font-bold mb-6">Bag</h1>

          {/* Cart Items */}
          {cart.map((item) => (
            <div key={item.id} className="flex items-start
justify-between border-b pb-4 mb-4">
              <div className="flex">
                {/* Sanity Image Fix */}
                <Image
                  src={item.image ? urlFor(item.image).url() :
'/placeholder.png'}
```

```
                  alt={item.name || 'Product Image'}
                  width={100}
                  height={100}
                  className="border-2 border-gray-300 rounded-md"
                />
                <div className="ml-4">
                  <h3 className="font-medium
text-gray-800">{item.name}</h3>
                  <p className="text-sm
text-gray-600">{item.description}</p>
                  <p className="text-gray-900 font-medium mt-1">
                    MRP: ₹{item.price}
                  </p>
                  <p className="text-sm text-gray-600">
                    Size: {item.size || 'N/A'} | Quantity:
{item.quantity || 0}
                  </p>
                  <div className="flex mt-2">
                    <button
                      onClick={() => changeQuantity(item.id,
'decrement')}
                      className="text-gray-600 hover:text-black text-xl
mr-2"
                    >
                      -
                    </button>
                    <button
                      onClick={() => changeQuantity(item.id,
'increment')}
                      className="text-gray-600 hover:text-black
text-xl"
                    >
                      +
                    </button>
                  </div>
                </div>
              </div>
              <div className="flex flex-col items-center">
                <button
                  onClick={() => removeFromCart(item.id)}
                  className="text-gray-600 hover:text-black mt-2"
                >
                  🗑️
```

```
                </button>
              </div>
            </div>
          ))}
        </div>

        {/* Right Section - Summary */}
        <div className="w-full lg:w-1/3 bg-gray-50 rounded-lg p-6
border border-gray-200">
          <h2 className="text-lg font-bold text-gray-800
mb-6">Summary</h2>

          {/* Subtotal */}
          <div className="flex justify-between text-gray-800 mb-4">
            <p>Subtotal</p>
            <p>₹{isNaN(total) ? '0.00' :
total.toLocaleString('en-IN')}</p>
          </div>

          {/* Delivery Fee */}
          <div className="flex justify-between text-gray-800 mb-4">
            <p>Delivery Fee</p>
            <p>₹{deliveryFee}</p>
          </div>

          {/* Total */}
          <div className="flex justify-between font-bold text-lg
text-gray-900 mb-6">
            <p>Total</p>
            <p>₹{isNaN(total + deliveryFee) ? '0.00' : (total +
deliveryFee).toLocaleString('en-IN')}</p>
          </div>

          {/* Checkout Button */}
          <Link href = "/orderdetails">          <button
            onClick={handleCheckout}
            className="w-full bg-black text-white py-3 rounded-full
hover:bg-gray-800 transition duration-300"
          >
            Member Checkout
          </button>
          </Link>
```

```
        </div>
      </div>
    </div>
  );
};

export default ShoppingCartPage;

tate, useEffect } from 'react';
import Image from 'next/image';
import Link from 'next/link';
import { urlFor } from '@/sanity/lib/image'; // Import Sanity image
builder
import { useRouter } from 'next/navigation';

const ShoppingCartPage = () => {
  const [cart, setCart] = useState<any[]>([]);
  const [total, setTotal] = useState(0);
  const [deliveryFee, setDeliveryFee] = useState(500); // Default ₹500
if below ₹14,000
  const router = useRouter();

  // Fetch cart data from localStorage
  useEffect(() => {
    try {
      const storedCart = JSON.parse(localStorage.getItem('cart') ||
'[]');
      if (Array.isArray(storedCart)) {
        setCart(storedCart);
        calculateTotal(storedCart);
      } else {
        console.error('Invalid cart data');
        setCart([]);
      }
    } catch (error) {
      console.error('Error parsing cart data', error);
      setCart([]);
    }
  }, []);

  // Calculate total price of the cart
  const calculateTotal = (cartItems: any[]) => {
    const totalPrice = cartItems.reduce(
```

```
      (acc, item) => acc + item.price * (item.quantity || 0),
      0
    );

    setTotal(totalPrice);
    setDeliveryFee(totalPrice >= 14000 ? 0 : 500); // Free delivery if
₹14,000 or more
  };

  // Remove item from cart
  const removeFromCart = (id: string) => {
    const updatedCart = cart.filter((item) => item.id !== id);
    setCart(updatedCart);
    localStorage.setItem('cart', JSON.stringify(updatedCart));
    calculateTotal(updatedCart);
  };

  // Handle item quantity change
  const changeQuantity = (id: string, type: 'increment' | 'decrement')
=> {
    const updatedCart = cart.map((item) => {
      if (item.id === id) {
        if (type === 'increment') {
          item.quantity = (item.quantity || 0) + 1;
        } else if (type === 'decrement' && item.quantity > 1) {
          item.quantity -= 1;
        }
      }
      return item;
    });
    setCart(updatedCart);
    localStorage.setItem('cart', JSON.stringify(updatedCart));
    calculateTotal(updatedCart);
  };

  // Handle checkout
  const handleCheckout = () => {
    if (cart.length === 0) {
      alert('Your cart is empty!');
      return;
    }

    const finalAmount = total + deliveryFee;
```

```
    alert(`Proceeding to checkout. Total: ₹${finalAmount}`);

    // Navigate to a checkout page
    router.push('/checkout');
  };

  if (cart.length === 0)
    return <p className="text-center text-xl mt-12">Your cart is
empty.</p>;

  return (
    <div className="flex flex-col bg-white mt-[150px] px-4 sm:px-6
lg:px-16">
      {/* Free Delivery Section */}
      <div className="bg-gray-100 p-4 text-sm text-gray-600 mb-8
border-t border-b border-gray-200">
        {total >= 14000 ? (
          <p className="text-green-600 font-semibold">
            You qualify for free delivery!
          </p>
        ) : (
          <p>
            Free Delivery applies to orders of ₹14,000.00 or more.{' '}
            <a href="#" className="underline">
              View details
            </a>
          </p>
        )}
      </div>

      {/* Main Container */}
      <div className="flex flex-col lg:flex-row gap-8">
        {/* Left Section - Bag Items */}
        <div className="flex-1">
          <h1 className="text-2xl font-bold mb-6">Bag</h1>

          {/* Cart Items */}
          {cart.map((item) => (
            <div key={item.id} className="flex items-start
justify-between border-b pb-4 mb-4">
              <div className="flex">
                {/* Sanity Image Fix */}
                <Image
```

```
                  src={item.image ? urlFor(item.image).url() :
'/placeholder.png'}
                  alt={item.name || 'Product Image'}
                  width={100}
                  height={100}
                  className="border-2 border-gray-300 rounded-md"
                />
                <div className="ml-4">
                  <h3 className="font-medium
text-gray-800">{item.name}</h3>
                  <p className="text-sm
text-gray-600">{item.description}</p>
                  <p className="text-gray-900 font-medium mt-1">
                    MRP: ₹{item.price}
                  </p>
                  <p className="text-sm text-gray-600">
                    Size: {item.size || 'N/A'} | Quantity:
{item.quantity || 0}
                  </p>
                  <div className="flex mt-2">
                    <button
                      onClick={() => changeQuantity(item.id,
'decrement')}
                      className="text-gray-600 hover:text-black text-xl
mr-2"
                    >
                      -
                    </button>
                    <button
                      onClick={() => changeQuantity(item.id,
'increment')}
                      className="text-gray-600 hover:text-black
text-xl"
                    >
                      +
                    </button>
                  </div>
                </div>
              </div>
              <div className="flex flex-col items-center">
                <button
                  onClick={() => removeFromCart(item.id)}
                  className="text-gray-600 hover:text-black mt-2"
```

```
                >
                  🗑️
                </button>
              </div>
            </div>
          ))}
        </div>

        {/* Right Section - Summary */}
        <div className="w-full lg:w-1/3 bg-gray-50 rounded-lg p-6
border border-gray-200">
          <h2 className="text-lg font-bold text-gray-800
mb-6">Summary</h2>

          {/* Subtotal */}
          <div className="flex justify-between text-gray-800 mb-4">
            <p>Subtotal</p>
            <p>₹{isNaN(total) ? '0.00' :
total.toLocaleString('en-IN')}</p>
          </div>

          {/* Delivery Fee */}
          <div className="flex justify-between text-gray-800 mb-4">
            <p>Delivery Fee</p>
            <p>₹{deliveryFee}</p>
          </div>

          {/* Total */}
          <div className="flex justify-between font-bold text-lg
text-gray-900 mb-6">
            <p>Total</p>
            <p>₹{isNaN(total + deliveryFee) ? '0.00' : (total +
deliveryFee).toLocaleString('en-IN')}</p>
          </div>

          {/* Checkout Button */}
          <Link href = "/orderdetails">          <button
            onClick={handleCheckout}
            className="w-full bg-black text-white py-3 rounded-full
hover:bg-gray-800 transition duration-300"
          >
            Member Checkout
          </button>
```

```
          </Link>

        </div>
      </div>
    </div>
  );
};

export default ShoppingCartPage;
```

Search Bar

```
"use client";

import { useState, useEffect } from "react";
import Image from "next/image";
import { FaSearch, FaHeart, FaShoppingBag } from "react-icons/fa";
import Link from "next/link";
import { client } from "@/sanity/lib/client";
import imageUrlBuilder from "@sanity/image-url";

// ✅ Image URL Builder for Sanity
const builder = imageUrlBuilder(client);
const urlFor = (source: any) =>
  source?.asset ? builder.image(source).auto("format").fit("max").url()
: "/placeholder.png";

// ✅ Product Type Definition
type Product = {
  productName: string;
  category: string;
  slug?: { current?: string };
  image?: { asset?: { url: string } };
};

const Navbar = () => {
  const [searchTerm, setSearchTerm] = useState("");
  const [products, setProducts] = useState<Product[]>([]);
  const [filteredProducts, setFilteredProducts] =
useState<Product[]>([]);
  const [searchActive, setSearchActive] = useState(false);

  // ✅ Fetch Products Once
```

```
  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const query = `*[_type == "product"]{
          productName, category, slug, image { asset -> { url } }
        }`;
        const data = await client.fetch(query);
        setProducts(data);
      } catch (error) {
        console.error("Sanity Fetch Error:", error);
      }
    };
    fetchProducts();
  }, []);

  // ✅ Filter Products for Search
  useEffect(() => {
    if (searchTerm.trim() === "") {
      setFilteredProducts([]);
      setSearchActive(false);
    } else {
      const filtered = products.filter(
        (product) =>

product.productName.toLowerCase().includes(searchTerm.toLowerCase()) ||

product.category.toLowerCase().includes(searchTerm.toLowerCase())
      );
      setFilteredProducts(filtered);
      setSearchActive(true);
    }
  }, [searchTerm, products]);

  return (
    <>
      {/* ✅ Push Content Below Top Bar */}
      <div className="mt-[36px]">
        {/* 🏷️ Category Navigation (Below Top Bar) */}
        <div className="flex items-center justify-center py-2 border-b
shadow-sm bg-gray-50">
          <ul className="flex space-x-6 text-md font-semibold">
            <li className="cursor-pointer hover:underline">
              <Link href="/allproducts">All Products</Link>
```

```
            </li>
            <li className="cursor-pointer hover:underline">
              <Link href="/categories/men">Men</Link>
            </li>
            <li className="cursor-pointer hover:underline">
              <Link href="/categories/women">Women</Link>
            </li>
            <li className="cursor-pointer hover:underline">
              <Link href="/categories/kids">Kids</Link>
            </li>
          </ul>
        </div>

        {/* 🔍 Search Bar, Cart, Wishlist Section */}
        <div className="w-full py-2 px-6 flex items-center
justify-between shadow-md bg-white">
          {/* Nike Logo */}
          <Link href="/">
            <Image src="/nike.png" alt="Nike logo" width={40}
height={40} className="cursor-pointer" />
          </Link>

          {/* 🔍 Search Bar */}
          <div className="relative flex-1 max-w-[300px] mx-4">
            <input
              type="text"
              placeholder="Search..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
              className="p-2 pl-10 w-full border border-gray-300
rounded-full focus:outline-none focus:ring-2 focus:ring-blue-500
text-sm"
            />
            <FaSearch className="absolute left-3 top-1/2 transform
-translate-y-1/2 text-gray-500" />

            {/* 🔍 Search Dropdown */}
            {searchActive && (
              <div className="absolute top-full left-0 w-full bg-white
border border-gray-300 shadow-lg rounded-md z-50">
                <ul className="py-2">
                  {filteredProducts.length > 0 ? (
                    <>
```

```
                        {filteredProducts.slice(0, 3).map((product,
index) => (
                          <li key={index} className="px-4 py-2
hover:bg-gray-100">
                            <Link
href={`/product/${product.slug?.current}`} onClick={() =>
setSearchTerm("")}>
                              <div className="flex items-center">
                                <Image
                                  src={urlFor(product.image)}
                                  alt={product.productName || "Product
Image"}
                                  width={40}
                                  height={40}
                                  className="rounded-md"
                                />
                                <div className="ml-3">
                                  <p className="font-semibold
text-sm">{product.productName}</p>
                                  <p className="text-xs
text-gray-500">{product.category}</p>
                                </div>
                              </div>
                            </Link>
                          </li>
                        ))}
                        {filteredProducts.length > 3 && (
                          <li className="px-4 py-2 text-blue-600
cursor-pointer hover:bg-gray-100 text-center text-sm">
                            <Link href={`/allproducts`} onClick={() =>
setSearchTerm("")}>
                              Show All Items
                            </Link>
                          </li>
                        )}
                      </>
                    ) : (
                      <li className="px-4 py-2 text-gray-500 text-sm">No
products found.</li>
                    )}
                  </ul>
                </div>
              )}
```

```
          </div>

          {/* 🛒 Cart & Wishlist Icons */}
          <div className="flex items-center space-x-4">
            <Link href="/cartpage">
              <button className="p-2 bg-blue-600 rounded-full
hover:bg-blue-700 focus:outline-none">
                <FaShoppingBag className="text-white text-sm" />
              </button>
            </Link>
            <button className="p-2 bg-gray-200 rounded-full
hover:bg-gray-300 focus:outline-none">
              <FaHeart className="text-red-500 text-sm" />
            </button>
          </div>
        </div>

        {/* 🚀 Ensure Content is Visible Below Navbar */}
        <div className="mt-[20px]" />
      </div>
    </>
  );
};

export default Navbar;
```

More code in Github Directory

Technical Report on the Development and Integration of Project Components

1. Steps Taken to Build and Integrate Components

The project involved creating a dynamic and user-friendly e-commerce platform with a shopping cart feature and order processing system. The components were built and integrated as follows:

a. Frontend Development:

- **Technologies Used**: Next.js (for SSR and React), Tailwind CSS (for styling), and Sanity CMS (for content management).
- **Component Setup**:

- **Topbar Component**: Designed for a fixed top navigation bar, handling login/logout functionality with `localStorage` integration to persist user session information.
- **Shopping Cart**: The cart was implemented using `useState` and `useEffect` hooks to manage the cart's state and synchronize with `localStorage`. The total price and item quantities were dynamically updated based on user actions (increment/decrement).
- **Order Page**: Created to capture order details, such as name, address, and postal code. The data entered by the user was displayed upon submission and showed a delivery status with a delivery truck emoji.
- **Sanity CMS Integration**: Sanity was used to manage and store product data such as price, name, and description. The integration allowed for easy content updates and ensured real-time reflection of changes on the front end.

b. Backend Integration:

- **Sanity API**: Integrated Sanity to manage product and order data efficiently. Custom schemas were created for products and orders. Data was fetched using Sanity's API and displayed dynamically on the frontend.
- **Cart and Order Data Handling**: The cart state was managed locally using React state and persisted using `localStorage`. Upon user checkout, the order details were stored and confirmed, and the cart was cleared.

2. Challenges Faced and Solutions Implemented

a. Challenge: Cart Management and Persistence:

- **Problem**: Ensuring that cart data was properly persisted across page reloads and user sessions.
- **Solution**: Used `localStorage` to store cart items. This allowed cart data to persist even if the user refreshed the page or navigated away. The cart state was updated dynamically based on user interactions, and changes were reflected immediately.

b. Challenge: Responsive Design:

- **Problem**: Ensuring the components (especially the Topbar and Cart) were responsive and worked well on mobile devices.
- **Solution**: Tailwind CSS was used extensively for building a responsive layout. Media queries were applied to adjust the design based on screen size, ensuring a seamless experience on both desktop and mobile devices. Additionally, the mobile menu was implemented with a hamburger icon that toggled visibility for smaller screens.

c. Challenge: Data Handling from Sanity CMS:

- **Problem**: Fetching data dynamically from Sanity and integrating it into React components.
- **Solution**: Used Sanity's `groq` query language to fetch product data and render it in components. Proper error handling was implemented to manage issues related to

empty or invalid data. Additionally, `useEffect` was utilized for data fetching and updating the UI.

d. Challenge: Delivery Status Update on Order Page:

- **Problem**: Showing the delivery status in real-time after the user completes the order.
- **Solution**: Implemented a status update that appears on the screen once the order details are submitted. A "Delivery Truck" emoji is displayed to indicate that the order is on the way. This was done by using React's state management to toggle the display of a success message and a visual indicator.

3. Best Practices Followed During Development

a. Component-based Architecture:

- The project was built using React's component-based architecture, which made it easier to break down the UI into reusable and isolated components. This helped in managing code efficiently and ensured reusability across different parts of the project.

b. Responsive Design:

- Tailwind CSS was used for styling, ensuring that the layout adjusted correctly for different screen sizes. Classes like `sm`, `md`, `lg` were utilized to make the website mobile-first and responsive.

c. State Management:

- Local state was used with React's `useState` for managing the shopping cart and order data. This simplified the data flow within each component. The use of `localStorage` allowed for persistence across page reloads and was handled in a way that avoided any unnecessary complexity.

d. Error Handling and Data Validation:

- Proper error handling was implemented, especially in the data-fetching logic. This included fallback mechanisms in case of failed API calls and validation of user inputs on the order page.

e. Security Considerations:

- Sensitive data (such as user session data) was handled using `localStorage`, with precautions to avoid storing sensitive information like passwords. Instead, only necessary session details (like the email) were stored.

f. Code Quality and Maintenance:

- The project followed clean code principles. Meaningful variable names, component names, and functions were used to improve readability. Additionally, the code was modular and structured in a way that made it easy to maintain and scale.

g. Version Control:

- Git was used for version control to track changes and collaborate with other team members. Regular commits were made, ensuring that each feature was added incrementally and was easy to track.

h. Testing:

- While no formal testing framework was introduced, the development environment was designed to allow easy integration of testing tools. During development, components were manually tested in various scenarios to ensure they performed correctly.