DAY 4: BUILDING DYNAMIC FRONTEND COMPONENTS FOR MARKETPLACE

Creator: Ahsan Paracha Slot: Sunday 2-5 Roll No: 0088100

Table of Contents

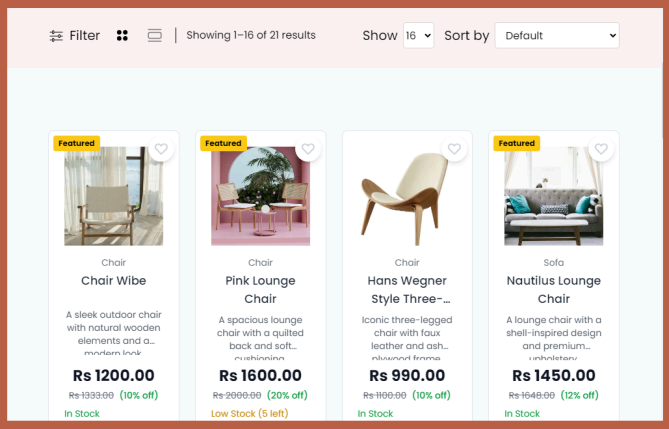
1. Functional Deliverables (Screenshots)

1. Functional Deliverables

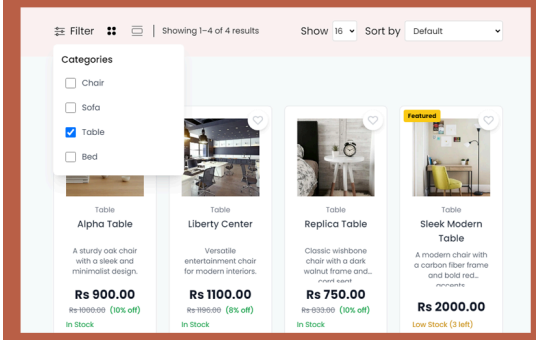
Product Listing Page:

A dynamic page that displays product data (sourced from Sanity.io) with active category filters, a functional search bar, and pagination.

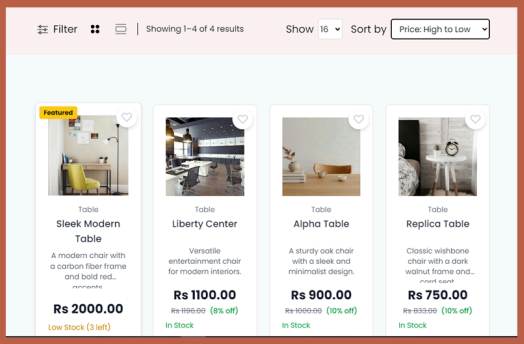
Screenshots



Screenshots



2. Categories Option

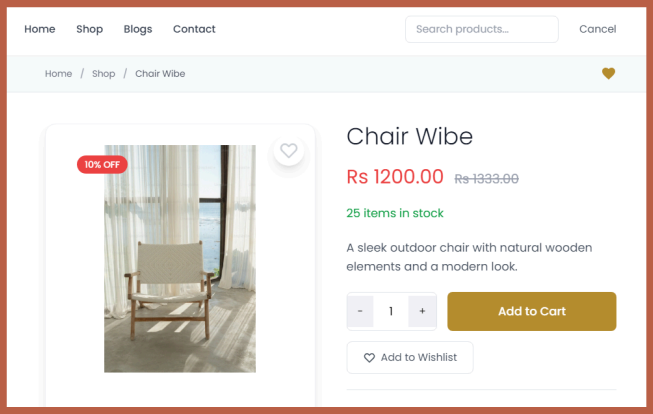


3. Sort by and etc

2. Individual Product Detail Pages:

Screenshots of product in the listing navigates to its dedicated page using dynamic routing with accurate data rendering.

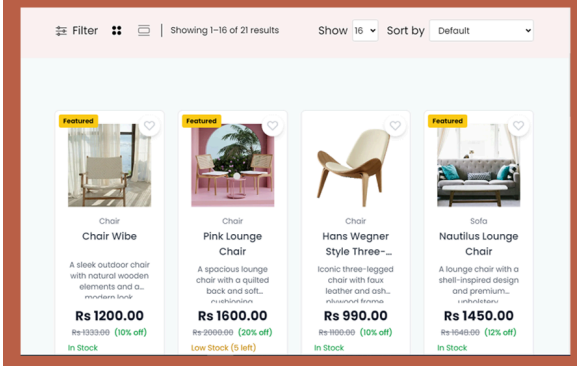
Screenshots

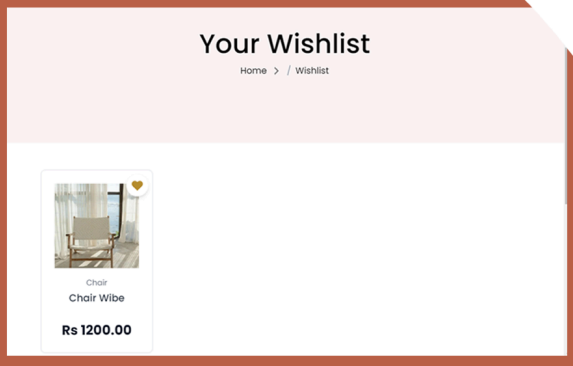


Additional Features:

Wishlist Functionality: The wishlist feature available across all product pages.

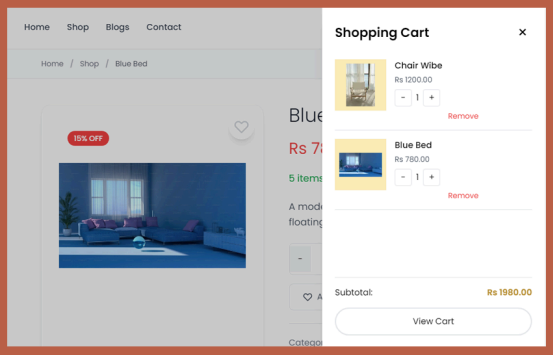
Screenshots

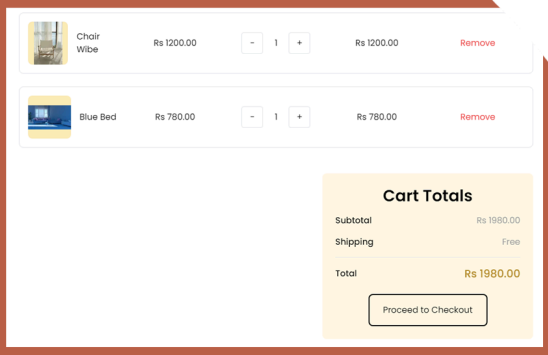




Cart Interactions: Showing the animated cart slider upon adding products, and the seamless redirection to the checkout page.

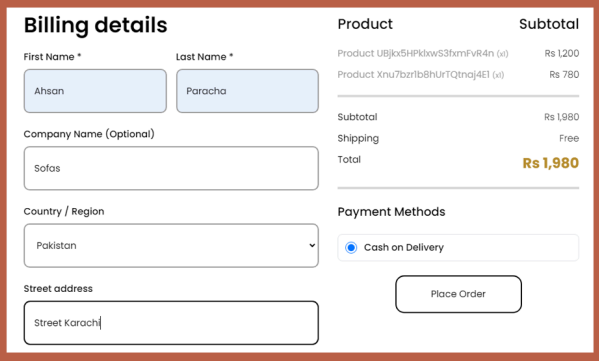
Screenshots

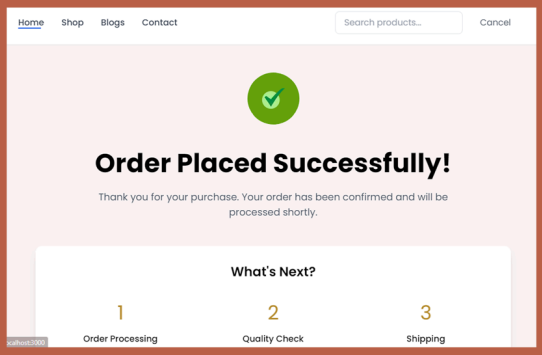




Checkout & Order Confirmation: Displaying the complete flow from proceeding to checkout to viewing the order confirmation page.

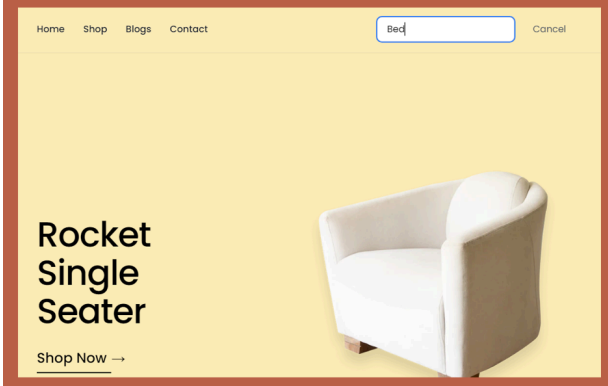
Screenshots

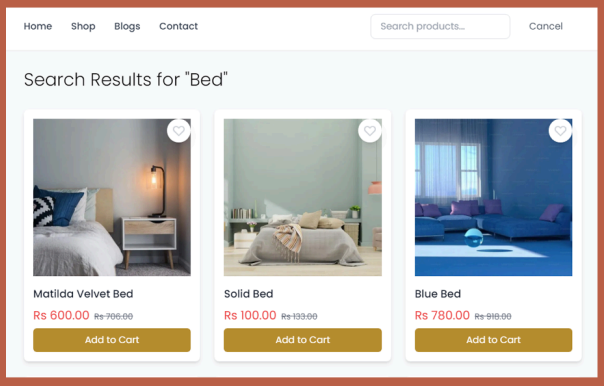




Search Bar:

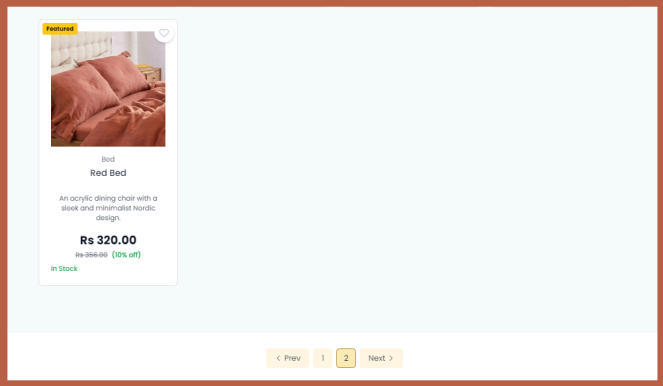
Screenshots





Pagination

Screenshots



2. Code Deliverables

Key Component Code Snippets

- **ProductCard Component:**

```
<div className="max-w-7xl mx-auto px-4 md:px-8 lg:px-16 py-12 grid lg:grid-cols-2 gap-12">
{/* Image Section */}
  <div className="bg-white rounded-2xl shadow-lg p-8 border border-gray-100 relative">
    <button
      onClick={(e) => handleWishlistToggle(e, selectedProduct._id)}
      className="absolute top-4 right-4 z-20 p-2 bg-white rounded-full shadow-lg hover:bg-[#FFF9E5] transition-all"
    >
      <svg
        className={`w-8 h-8 transition-colors ${
          wishlist.includes(selectedProduct._id)
            ? 'text-[#B88E2F] fill-current'
            : 'text-gray-300 hover:text-[#B88E2F]'
        }`}
        stroke="currentColor"
        fill="none"
        viewBox="0 0 24 24"
      >
        <path
          strokeLinecap="round"
          strokeLinejoin="round"
          strokeWidth={2}
          d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"
        />
      </svg>
    </button>
    <div className="relative aspect-square">
      <Image
        src={imageUrl}
        alt={selectedProduct.name}
        fill
        priority
        quality={100}
        className="object-contain"
        sizes="(max-width: 768px) 100vw, 50vw"
      />
      {selectedProduct.discountPercentage && (
        <div className="absolute top-4 left-4 bg-red-500 text-white px-3 py-1 rounded-full text-sm font-medium">
          {selectedProduct.discountPercentage}% OFF
        </div>
      )}
    </div>
  </div>

  {/* Product Details */}
  <div className="space-y-6">
    <h1 className="text-4xl font-light text-gray-900">{selectedProduct.name}</h1>

    <div className="flex items-baseline gap-4">
      <p className={`text-3xl ${selectedProduct.discountPercentage ? 'text-red-500' : 'text-gray-900'}`}>
        Rs {selectedProduct.price.toFixed(2)}
      </p>
      {selectedProduct.discountPercentage && (
        <p className="text-xl text-gray-400 line-through">
          Rs {Math.round(selectedProduct.price / (1 - selectedProduct.discountPercentage/100)).toFixed(2)}
        </p>
      )}
    </div>

    <p className={`text-lg ${selectedProduct.stockLevel > 0 ? 'text-green-600' : 'text-red-600'}`}>
      {selectedProduct.stockLevel > 0 ? `${selectedProduct.stockLevel} items in stock` : 'Currently out of stock'}
    </p>

    <p className="text-gray-600 leading-relaxed text-lg">
      {selectedProduct.description}
    </p>

    <div className="flex flex-wrap gap-4">
      <div className="flex items-center border border-gray-200 rounded-lg overflow-hidden">
```

```
            <button
              onClick={() => setQuantity(Math.max(1, quantity - 1))}
              className="px-4 py-3 bg-gray-100 hover:bg-gray-200 transition-colors"
            >
              -
            </button>
            <span className="px-6 py-3 bg-white text-lg">{quantity}</span>
            <button
              onClick={() => setQuantity(q => q + 1)}
              className="px-4 py-3 bg-gray-100 hover:bg-gray-200 transition-colors"
            >
              +
            </button>
          </div>
          <button
            onClick={handleAddToCart}
            disabled={selectedProduct.stockLevel === 0}
            className={`flex-1 px-8 py-4 text-lg font-medium rounded-lg transition-colors ${
              selectedProduct.stockLevel > 0
                ? 'bg-[#B88E2F] text-white hover:bg-[#A67C00]'
                : 'bg-gray-300 text-gray-500 cursor-not-allowed'
            }`}
          >
            {selectedProduct.stockLevel > 0 ? 'Add to Cart' : 'Out of Stock'}
          </button>
          <button
            onClick={(e) => handleWishlistToggle(e, selectedProduct._id)}
            className={`flex items-center gap-2 px-6 py-3 rounded-lg border transition-colors ${
              wishlist.includes(selectedProduct._id)
                ? 'border-[#B88E2F] bg-[#FFF9E5] text-[#B88E2F]'
                : 'border-gray-200 hover:border-[#B88E2F] text-gray-600'
            }`}
          >
            <svg
              className={`w-5 h-5 ${
                wishlist.includes(selectedProduct._id) ? 'fill-current' : 'fill-none'
              }`}
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"
              />
            </svg>
            {wishlist.includes(selectedProduct._id) ? 'In Wishlist' : 'Add to Wishlist'}
          </button>
        </div>

        <div className="space-y-3 pt-6 border-t border-gray-200">
          <div className="flex justify-between max-w-xs">
            <span className="text-gray-500">Category</span>
            <span className="text-gray-900">{selectedProduct.category}</span>
          </div>
          <div className="flex justify-between max-w-xs">
            <span className="text-gray-500">Stock</span>
            <span className="text-gray-900">{selectedProduct.stockLevel}</span>
          </div>
          {selectedProduct.discountPercentage && (
            <div className="flex justify-between max-w-xs">
              <span className="text-gray-500">Discount</span>
              <span className="text-gray-900">{selectedProduct.discountPercentage}%</span>
            </div>
          )}
        </div>
      </div>
    </div>
```

- **Shop Component (Including Dynamics routes, Filter section and paginaiton):**

```tsx
'use client'
import { useEffect, useState } from 'react'
import Image from "next/image"
import Link from "next/link"
import { createClient } from '@sanity/client'
import imageUrlBuilder from '@sanity/image-url'
import { useWishlist } from '@/context/WishlistContext'

const sanity = createClient({
  projectId: "587tggpl",
  dataset: "production",
  apiVersion: "2025-01-13",
  useCdn: true,
})

const builder = imageUrlBuilder(sanity)

interface SanityProduct {
  _id: string
  name: string
  price: number
  description: string
  discountPercentage?: number
  isFeaturedProduct?: boolean
  stockLevel: number
  category: string
  image: {
    asset: {
      _ref: string
    }
  }
}

type ViewMode = 'grid' | 'list'
type SortOption = 'default' | 'price_asc' | 'price_desc' | 'name_asc' | 'name_desc'

export default function Shop() {
  const { wishlist, addToWishlist, removeFromWishlist } = useWishlist()
  const [products, setProducts] = useState<SanityProduct[]>([])
  const [filteredProducts, setFilteredProducts] = useState<SanityProduct[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [viewMode, setViewMode] = useState<ViewMode>('grid')
  const [itemsPerPage, setItemsPerPage] = useState(16)
  const [currentPage, setCurrentPage] = useState(1)
  const [sortBy, setSortBy] = useState<SortOption>('default')
  const [isFilterOpen, setIsFilterOpen] = useState(false)
  const [categories, setCategories] = useState<string[]>([])
  const [selectedCategories, setSelectedCategories] = useState<string[]>([])

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const query = `*[_type == "product"] {
          _id,
          name,
          price,
          description,
          discountPercentage,
          isFeaturedProduct,
          stockLevel,
          category,
          image {
            asset {
              _ref
            }
          }
        }`

        const data = await sanity.fetch(query)
```

```
        setProducts(data)
        setFilteredProducts(data)

        // Extract unique categories
        const uniqueCategories = Array.from(
          new Set(data.map(p => p.category).filter(c => c))
        ) as string[]
        setCategories(uniqueCategories)

        setError(null)
      } catch (err) {
        setError('Failed to load products')
        console.error("Error fetching products:", err)
      } finally {
        setLoading(false)
      }
    }

    fetchProducts()
  }, [])

  useEffect(() => {
    let filtered = [...products]

    if (selectedCategories.length > 0) {
      filtered = filtered.filter(p => selectedCategories.includes(p.category))
    }

    switch (sortBy) {
      case 'price_asc':
        filtered.sort((a, b) => a.price - b.price)
        break
      case 'price_desc':
        filtered.sort((a, b) => b.price - a.price)
        break
      case 'name_asc':
        filtered.sort((a, b) => a.name.localeCompare(b.name))
        break
      case 'name_desc':
        filtered.sort((a, b) => b.name.localeCompare(a.name))
        break
      default:
        break
    }

    setFilteredProducts(filtered)
    setCurrentPage(1)
  }, [sortBy, selectedCategories, products])

  useEffect(() => {
    setCurrentPage(1)
  }, [itemsPerPage])

  const totalPages = Math.ceil(filteredProducts.length / itemsPerPage)
  const startIndex = (currentPage - 1) * itemsPerPage
  const endIndex = startIndex + itemsPerPage
  const currentProducts = filteredProducts.slice(startIndex, endIndex)

  const handlePageChange = (page: number) => {
    setCurrentPage(Math.max(1, Math.min(page, totalPages)))
  }

  const toggleCategory = (category: string) => {
    setSelectedCategories(prev =>
      prev.includes(category)
        ? prev.filter(c => c !== category)
        : [...prev, category]
    )
  }

  const handleWishlistToggle = (e: React.MouseEvent, productId: string) => {
    e.preventDefault()
    wishlist.includes(productId)
      ? removeFromWishlist(productId)
      : addToWishlist(productId)
  }
```

```
  if (loading) return <div className="text-center py-12">Loading products...</div>
  if (error) return <div className="text-center py-12 text-red-500">{error}</div>

  return (
    <main>
      {/* Hero Section */}
      <div className="relative h-[316px] w-full">
        <Image
          fill
          className="object-cover"
          src="/shop1.png"
          alt="Dining Table"
          priority
        />
        <div className="absolute inset-0 flex flex-col items-center justify-center gap-4 text-center">
          <Image
            width={77}
            height={77}
            src="/logo1.png"
            alt="Logo"
            className="h-auto w-16 md:w-20"
          />
          <h1 className="font-poppins text-3xl font-medium md:text-4xl lg:text-[48px]">Shop</h1>

          <div className="flex">
            <Link href="/" className="flex items-center gap-2 group">
              <p className="font-poppins font-normal text-[16px] text-black transition-colors group-hover:text-black">
                Home
              </p>
              <Image
                width={20}
                height={20}
                src="/arrow.png"
                alt="Arrow"
                className="opacity-50 group-hover:opacity-100 transition-opacity"
              />
            </Link>
            <p className="font-poppins font-normal text-[16px] text-black">
              Shop
            </p>
          </div>
        </div>
      </div>

      {/* Filters Section */}
      <div className="w-full bg-[#FAF4F4] relative">
        <div className="mx-auto flex h-auto flex-col justify-between gap-4 px-4 py-4 md:h-[100px] md:flex-row md:items-center md:px-8 lg:px-16">
          <div className="flex flex-wrap items-center gap-4 relative">
            <div
              className="flex items-center gap-2 cursor-pointer"
              onClick={() => setIsFilterOpen(!isFilterOpen)}
            >
              <Image width={25} height={25} src="/filter.png" alt="Filter" />
              <p className="font-poppins text-base md:text-[20px]">Filter</p>
            </div>

            {/* Filter Dropdown */}
            {isFilterOpen && (
              <div className="absolute left-0 top-full mt-2 z-50 bg-white shadow-xl rounded-lg p-4 md:w-64 w-full">
                <h3 className="font-poppins text-lg font-medium mb-3">Categories</h3>
                <div className="space-y-2">
                  {categories.map(category => (
                    <label
                      key={category}
                      className="flex items-center space-x-3 cursor-pointer hover:bg-gray-50 p-2 rounded"
                    >
                      <input
                        type="checkbox"
                        checked={selectedCategories.includes(category)}
                        onChange={() => toggleCategory(category)}
                        className="form-checkbox h-5 w-5 text-[#B88E2F] rounded border-gray-300 focus:ring-[#B88E2F]"
                      />
                      <span className="font-poppins text-gray-700">{category}</span>
                    </label>
                  ))}
                </div>
              </div>
```

```
            )}

            <div className="flex items-center gap-4">
              <button
                onClick={() => setViewMode('grid')}
                className={`p-1 ${viewMode === 'grid' ? 'opacity-100' : 'opacity-50'}`}
              >
                <Image width={28} height={28} src="/grid.png" alt="Grid view" />
              </button>
              <button
                onClick={() => setViewMode('list')}
                className={`p-1 ${viewMode === 'list' ? 'opacity-100' : 'opacity-50'}`}
              >
                <Image width={24} height={24} src="/view.png" alt="List view" />
              </button>
            </div>

            <div className="hidden h-6 w-px bg-[#000000] md:block" />

            <p className="font-poppins text-sm md:text-base">
              Showing {startIndex + 1}-{Math.min(endIndex, filteredProducts.length)} of {filteredProducts.length} results
            </p>
          </div>

          <div className="flex flex-wrap items-center gap-4">
            <div className="flex items-center gap-2">
              <p className="font-poppins text-base md:text-[20px]">Show</p>
              <select
                value={itemsPerPage}
                onChange={(e) => setItemsPerPage(Number(e.target.value))}
                className="flex h-8 w-12 items-center justify-center rounded border border-[#D9D9D9] bg-white md:h-10"
              >
                {[16, 32, 64].map((num) => (
                  <option key={num} value={num}>{num}</option>
                ))}
              </select>
            </div>

            <div className="flex items-center gap-2">
              <p className="font-poppins text-base md:text-[20px]">Sort by</p>
              <select
                value={sortBy}
                onChange={(e) => setSortBy(e.target.value as SortOption)}
                className="flex h-8 w-32 items-center rounded border border-[#D9D9D9] bg-white px-3 md:h-10 md:w-48"
              >
                <option value="default">Default</option>
                <option value="price_asc">Price: Low to High</option>
                <option value="price_desc">Price: High to Low</option>
                <option value="name_asc">Name: A-Z</option>
                <option value="name_desc">Name: Z-A</option>
              </select>
            </div>
          </div>
        </div>
      </div>

      {/* Products Section */}
      <section className="bg-gray-50 px-4 py-12 md:px-8 md:py-24 lg:px-16">
        <div className={`grid gap-6 ${viewMode === 'grid' ? 'grid-cols-1 sm:grid-cols-2 lg:grid-cols-4' : 'grid-cols-1'}`}>
          {currentProducts.map((product) => {
            const imageUrl = builder.image(product.image).width(600).height(600).url()
            const originalPrice = product.discountPercentage
              ? Math.round(product.price / (1 - product.discountPercentage/100))
              : null
            const isInWishlist = wishlist.includes(product._id)

            return (
              <Link
                key={product._id}
                href={`/products/${product._id}`}
                className={`group block transition-transform hover:scale-[1.02] relative h-full ${
                  viewMode === 'list' ? 'flex gap-6 items-center' : ''
                }`}
              >
                <button
                  onClick={(e) => handleWishlistToggle(e, product._id)}
                  className={`absolute top-2 right-2 z-20 p-2 bg-white rounded-full shadow-md transition-all ${
```

```
                    isInWishlist ? 'text-[#B88E2F]' : 'text-gray-300 hover:text-[#B88E2F]'
                  }`}
                >
                  <svg
                    className="w-6 h-6"
                    fill={isInWishlist ? 'currentColor' : 'none'}
                    stroke="currentColor"
                    viewBox="0 0 24 24"
                  >
                    <path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth={2}
                      d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"
                    />
                  </svg>
                </button>

                {product.isFeaturedProduct && (
                  <div className="absolute top-2 left-2 bg-yellow-400 text-xs font-bold px-2 py-1 rounded z-10">
                    Featured
                  </div>
                )}

                <div className={`h-full flex ${
                  viewMode === 'list' ? 'flex-row w-full' : 'flex-col'
                } rounded-lg border border-gray-200 bg-white p-4 transition-shadow hover:shadow-md md:p-6`}>
                  <div className={`relative mb-4 ${
                    viewMode === 'list' ? 'w-1/3 aspect-square' : 'aspect-square'
                  } overflow-hidden`}>
                    <Image
                      src={imageUrl}
                      alt={product.name}
                      fill
                      className="object-contain transition-transform duration-300 group-hover:scale-105"
                      sizes={viewMode === 'list' ? "33vw" : "(max-width: 640px) 100vw, (max-width: 1024px) 50vw, 25vw"}
                    />
                  </div>

                  <div className={`flex flex-col flex-1 ${
                    viewMode === 'list' ? 'w-2/3 pl-4' : ''
                  }`}>
                    <div className={viewMode === 'list' ? 'text-left' : 'text-center'}>
                      <p className="mb-1 text-sm text-gray-500">{product.category}</p>
                      <p className="mb-2 line-clamp-2 text-base font-medium text-gray-800 md:text-lg min-h-[3rem]">
                        {product.name}
                      </p>
                      <div className={`mb-2 text-sm text-gray-600 ${
                        viewMode === 'list' ? 'line-clamp-5' : 'line-clamp-3'
                      } min-h-[4.5rem]`}>
                        {product.description}
                      </div>
                    </div>

                    <div className="mt-auto">
                      <div className={`flex flex-col ${viewMode === 'list' ? 'items-start' : 'items-center'} gap-1`}>
                        <p className="text-xl font-bold text-gray-900 md:text-2xl">
                          Rs {product.price.toFixed(2)}
                        </p>
                        {originalPrice && (
                          <div className="flex items-center gap-2">
                            <span className="text-sm line-through text-gray-500">
                              Rs {originalPrice.toFixed(2)}
                            </span>
                            <span className="text-sm font-medium text-green-600">
                              ({product.discountPercentage}% off)
                            </span>
                          </div>
                        )}
                      </div>

                      <div className="mt-2 flex items-center gap-2">
                        <span className={`text-sm ${
                          product.stockLevel > 5
                            ? 'text-green-600'
                            : product.stockLevel > 0
```

```
                              ? 'text-yellow-600'
                              : 'text-red-600'
                          }`}>
                            {product.stockLevel > 5
                              ? 'In Stock'
                              : product.stockLevel > 0
                                ? `Low Stock (${product.stockLevel} left)`
                                : 'Out of Stock'}
                          </span>
                        </div>
                      </div>
                    </div>
                  </div>
                </Link>
              )
            })}
          </div>
        </section>

        {/* Pagination */}
        <section className="px-4 py-8 md:px-8 lg:px-16">
          <div className="flex justify-center">
            <nav className="flex flex-wrap items-center justify-center gap-2">
              <button
                onClick={() => handlePageChange(currentPage - 1)}
                disabled={currentPage === 1}
                className="flex h-8 items-center justify-center gap-1 rounded-md border-2 border-[#FFF9E5] bg-[#FFF9E5] px-3 text-sm text-gray-600
transition-colors duration-200 hover:bg-[#FBEBB5] focus:outline-none focus:ring-2 focus:ring-[#B88E2F] md:h-10 md:px-4 md:text-base"
              >
                <svg
                  xmlns="http://www.w3.org/2000/svg"
                  className="h-4 w-4 rotate-180"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M9 5l7 7-7 7"
                  />
                </svg>
                Prev
              </button>

              {Array.from({ length: totalPages }, (_, i) => i + 1).map((page) => (
                <button
                  key={page}
                  onClick={() => handlePageChange(page)}
                  className={`flex h-8 w-8 items-center justify-center rounded-md border-2 text-sm transition-colors duration-200
focus:outline-none focus:ring-2 focus:ring-[#B88E2F] md:h-10 md:w-10 md:text-base ${
                    page === currentPage
                      ? 'border-[#B88E2F] bg-[#FBEBB5] text-gray-800'
                      : 'border-[#FFF9E5] bg-[#FFF9E5] text-gray-600 hover:bg-[#FBEBB5]'
                  }`}
                >
                  {page}
                </button>
              ))}

              <button
                onClick={() => handlePageChange(currentPage + 1)}
                disabled={currentPage === totalPages}
                className="flex h-8 items-center justify-center gap-1 rounded-md border-2 border-[#FFF9E5] bg-[#FFF9E5] px-3 text-sm text-gray-600
transition-colors duration-200 hover:bg-[#FBEBB5] focus:outline-none focus:ring-2 focus:ring-[#B88E2F] md:h-10 md:px-4 md:text-base"
              >
                Next
                <svg
                  xmlns="http://www.w3.org/2000/svg"
                  className="h-4 w-4"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    strokeLinecap="round"
```

```
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M9 5l7 7-7 7"
                  />
                </svg>
              </button>
            </nav>
          </div>
        </section>

        {/* Policy Section */}
        <div className="bg-[#FAF4F4] px-4 py-12 md:py-24">
          <div className="mx-auto grid max-w-screen-xl gap-8 md:grid-cols-3">
            {policyItems.map((item, index) => (
              <div key={index} className="text-center">
                <h2 className="font-poppins text-2xl font-medium md:text-3xl">
                  {item.title}
                </h2>
                <p className="font-poppins mx-auto mt-2 max-w-md text-base text-gray-600 md:text-lg">
                  {item.description}
                </p>
              </div>
            ))}
          </div>
        </div>
      </main>
    )
}

const policyItems = [
  {
    title: "Free Delivery",
    description: "For all orders over $50, consectetur adipim scing elit."
  },
  {
    title: "90 Days Return",
    description: "If goods have problems, consectetur adipim scing elit."
  },
  {
    title: "Secure Payment",
    description: "100% secure payment, consectetur adipim scing elit."
  }
]
```

- **Search Component:**`// app/search/page.tsx`

```
'use client'
import { useEffect, useState } from 'react';
import { useSearchParams } from 'next/navigation';
import { createClient } from '@sanity/client';
import imageUrlBuilder from '@sanity/image-url';
import Link from 'next/link';
import Image from 'next/image';
import { useWishlist } from '@/context/WishlistContext';
import { useCart } from '@/context/CartContext';

const sanity = createClient({
  projectId: "587tggpl",
  dataset: "production",
  apiVersion: "2025-01-13",
  useCdn: true,
});

const builder = imageUrlBuilder(sanity);

interface SanityProduct {
```

```
    _id: string;
    name: string;
    price: number;
    description: string;
    category: string;
    stockLevel: number;
    discountPercentage?: number;
    image: {
      asset: {
        _ref: string;
      };
    };
  }

  export default function SearchPage() {
    const searchParams = useSearchParams();
    const query = searchParams.get('q') || '';
    const [products, setProducts] = useState<SanityProduct[]>([]);
    const [loading, setLoading] = useState(true);
    const { addToCart } = useCart();
    const { wishlist, addToWishlist, removeFromWishlist } = useWishlist();

    useEffect(() => {
      const fetchProducts = async () => {
        try {
          const searchResults = await sanity.fetch(
            `*[_type == "product" && (
              name match $query ||
              description match $query ||
              category match $query
            )]{
              _id, name, price, description, category,
              stockLevel, discountPercentage, image
            }`,
            { query: `*${query}*` }
          );
          setProducts(searchResults);
        } catch (error) {
          console.error("Search error:", error);
        } finally {
          setLoading(false);
        }
      };

      if (query) fetchProducts();
    }, [query]);

    const handleWishlistToggle = (productId: string) => {
      wishlist.includes(productId)
        ? removeFromWishlist(productId)
        : addToWishlist(productId);
    };

    if (loading) return <LoadingSpinner />;

    return (
      <div className="min-h-screen bg-gray-50">
        <div className="max-w-7xl mx-auto px-4 py-8 sm:px-6 lg:px-8">
          <h1 className="text-3xl font-light mb-8">
            Search Results for "{query}"
          </h1>

          {products.length === 0 ? (
            <div className="text-center py-12">
              <p className="text-gray-500 text-lg">No products found</p>
            </div>
          ) : (
            <div className="grid grid-cols-1 gap-6 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4">
              {products.map((product) => {
                const imageUrl = builder.image(product.image).width(600).height(600).url();
                const originalPrice = product.discountPercentage
                  ? Math.round(product.price / (1 - product.discountPercentage/100))
                  : null;
                const isInWishlist = wishlist.includes(product._id);

                return (
                  <div key={product._id} className="group relative bg-white p-4 rounded-lg shadow-md hover:shadow-lg transition-shadow">
```

```
                    <button
                      onClick={() => handleWishlistToggle(product._id)}
                      className="absolute top-4 right-4 z-20 p-2 bg-white rounded-full shadow-lg hover:bg-[#FFF9E5] transition-all"
                    >
                      <svg
                        className={`w-6 h-6 ${
                          isInWishlist ? 'text-[#B88E2F] fill-current' : 'text-gray-300'
                        }`}
                        fill="none"
                        stroke="currentColor"
                        viewBox="0 0 24 24"
                      >
                        <path
                          strokeLinecap="round"
                          strokeLinejoin="round"
                          strokeWidth={2}
                          d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"
                        />
                      </svg>
                    </button>

                    <Link href={`/products/${product._id}`}>
                      <div className="aspect-square relative mb-4 overflow-hidden">
                        <Image
                          src={imageUrl}
                          alt={product.name}
                          fill
                          className="object-contain group-hover:scale-105 transition-transform"
                          sizes="(max-width: 640px) 100vw, (max-width: 1024px) 50vw, 25vw"
                        />
                      </div>
                    </Link>

                    <div className="space-y-2">
                      <h3 className="text-lg font-medium text-gray-900">
                        {product.name}
                      </h3>
                      <div className="flex items-baseline gap-2">
                        <p className={`text-xl ${
                          product.discountPercentage ? 'text-red-500' : 'text-gray-900'
                        }`}>
                          Rs {product.price.toFixed(2)}
                        </p>
                        {product.discountPercentage && (
                          <p className="text-sm text-gray-500 line-through">
                            Rs {originalPrice?.toFixed(2)}
                          </p>
                        )}
                      </div>
                      <button
                        onClick={() => addToCart(product._id, 1, product.price)}
                        className="w-full bg-[#B88E2F] text-white py-2 rounded-md hover:bg-[#A67C00] transition-colors"
                      >
                        Add to Cart
                      </button>
                    </div>
                  </div>
                );
              })}
            </div>
          )}
        </div>
      </div>
    );
}

const LoadingSpinner = () => (
  <div className="min-h-screen flex items-center justify-center">
    <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-[#B88E2F]" />
  </div>
);
```

API Integration

- ## API Scripts:

```
import { createClient } from '@sanity/client';
import fetch from 'node-fetch';

// Initialize Sanity client
const client = createClient({
  projectId: "587tggp1",
  dataset: "production",
  useCdn: false, // Set to true if you want faster reads
  apiVersion: '2025-01-13',
  token:
"skvogBt05WwY4g1QWg8foS3MuBjqzR3lvpaaDaeEk4E4I1S1tMlNAFJsst8WeDuV3GrJA6Rbh2wmw6jbNWnVnNvZQeAdoiNM4Qhhjy09KeljsbX3z8arSgsoF9iYYm01fmbyCg7m
fwWYvA1FTsJ8Mf2Fkf67yEBUFlrYrCoD6AvOX4N7ojo4", // Replace with your Sanity token
});

// Function to upload an image to Sanity
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

// Function to upload a single product to Sanity
async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imagePath);

    if (imageId) {
      const document = {
        _type: 'product',
        id: product.id,
        name: product.name,
        image: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        price: parseFloat(product.price), // Ensure the price is a number
        description: product.description,
        discountPercentage: product.discountPercentage,
        isFeaturedProduct: product.isFeaturedProduct,
        stockLevel: product.stockLevel,
        category: product.category,
      };

      const createdProduct = await client.create(document);
      console.log(`Product "${product.name}" uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product "${product.name}" skipped due to image upload failure.`);
    }
  } catch (error) {
```

```
    console.error('Error uploading product:', error);
  }
}

// Function to fetch products from the provided API and upload them to Sanity
async function migrateProducts() {
  try {
    const response = await fetch('https://template-0-beta.vercel.app/api/product');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

// Start the migration
migrateProducts();
```

All code snippets are well-commented and structured according to best practices for modular and maintainable frontend development.

---

3. Documentation

Feature Implementation Documentation for the E-Commerce Website

Overview

This document outlines the key features and functionalities implemented in the e-commerce website. The project has been developed as part of the GIAIC Program Hackathon, with a focus on enhancing user experience through dynamic content, interactive UI elements, and smooth navigational flows.

1. Search Functionality

- **Implementation:** A search option has been integrated into the website's header. When a user inputs a query, they are redirected to a dedicated search results page.
- **User Flow:**
    - User enters search text in the header.
    - The system processes the query and displays relevant products on a full-page view.
- **Objective:** Enhance the discoverability of products by allowing users to quickly locate desired items.

2. Wishlist Feature

- **Implementation:** A fully functional wishlist has been added across all product pages.
- **User Flow:**
    - Users can click on a “wishlist” icon on any product.
    - All selected items are saved and can be viewed on a dedicated wishlist page.
- **Objective:** Allow users to save products for later review and future purchase, improving user engagement and retention.

3. Sanity.io Integration for Product Data

- **Implementation:** Product data is sourced from Sanity.io and integrated into both the Home and Shop pages.
- **User Flow:**
    - The homepage and shop page dynamically fetch and display product listings from Sanity.io.
- **Objective:** Ensure that product data is consistent, up-to-date, and easily manageable through a headless CMS.

4. Single Product Page with Dynamic Routing

- **Implementation:** Dynamic routes have been created for each product, resulting in individual product pages.
- **User Flow:**
    - Clicking on a product directs the user to its dedicated page, generated dynamically based on the product’s unique identifier.
- **Objective:** Improve SEO and provide detailed product information, resulting in a more personalized shopping experience.

5. Cart Functionality

- **Implementation:** A comprehensive cart feature has been developed. The user flow is described in text within the documentation to outline its full functionality.
- **User Flow:**
    - Users add products to the cart using an “Add to Cart” button.
    - Upon adding, an animated cart slider appears, visually confirming that the item has been added.
    - When the user clicks on “Proceed to Checkout” from the cart, they are redirected to the checkout page.
- **Objective:** Provide a seamless and interactive shopping experience by combining visual cues (animations) with functional navigation.

6. Checkout and Order Confirmation Process

- **Implementation:**
    - A dedicated checkout page has been created to handle the purchasing process.

    - Once the user completes the checkout process, they are redirected to an order confirmation page where they can view their order details.
- **User Flow:**
    - After reviewing the cart and clicking “Proceed to Checkout,” the user is guided through the payment process.
    - Instead of incorporating Toastify notifications as suggested, an order confirmation page with engaging animations has been implemented.
- **Objective:** To deliver clear confirmation of order placement while maintaining a consistent and attractive user interface.

7. Blog Page

- **Current Status:**
    - The blog page is not yet fully functional. Based on the current requirements and project scope, its development has been deprioritized.
- **Future Consideration:**
    - If needed, functionality may be added in a later phase, depending on project requirements and feedback.

Conclusion

The website is fully functional with key e-commerce features including search, wishlist, dynamic product pages, cart functionality, and a complete checkout process. The use of animations enhances user interaction by providing immediate visual feedback, such as the animated cart slider upon adding an item. Although the blog page is not operational at this time, the primary focus has been to ensure a robust shopping experience for users. This documentation captures the core functionalities developed during the hackathon and serves as a comprehensive reference for the project’s implementation details.