

Day 4 - Dynamic Frontend Components - Oak&Teak

1. Functional Deliverables:

○ Product Listing Component:

Dynamically render product data in a grid layout, including name, price, image, and stock status.



Arc Econo Executive Chair

~~45000~~ 41400.00



Majestic Sofa Set

45000



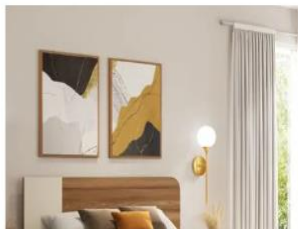
Quest Dresser

35000



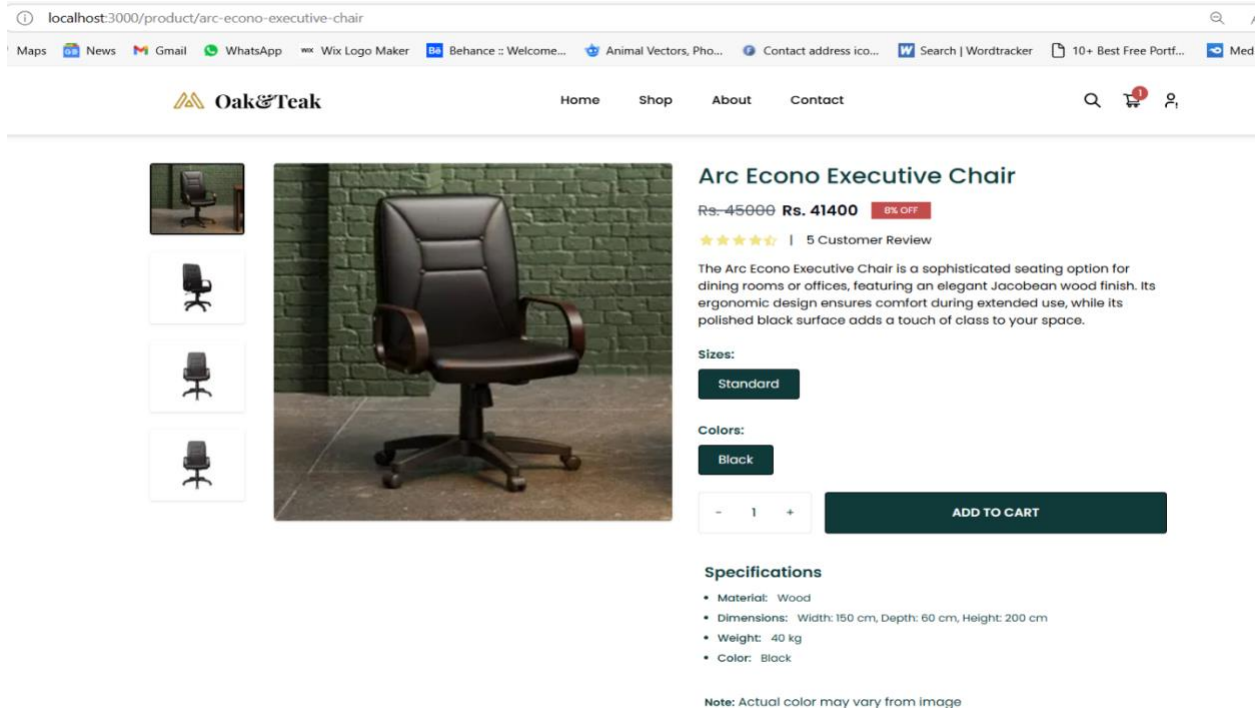
Royal Cocoa Dining Table

55000



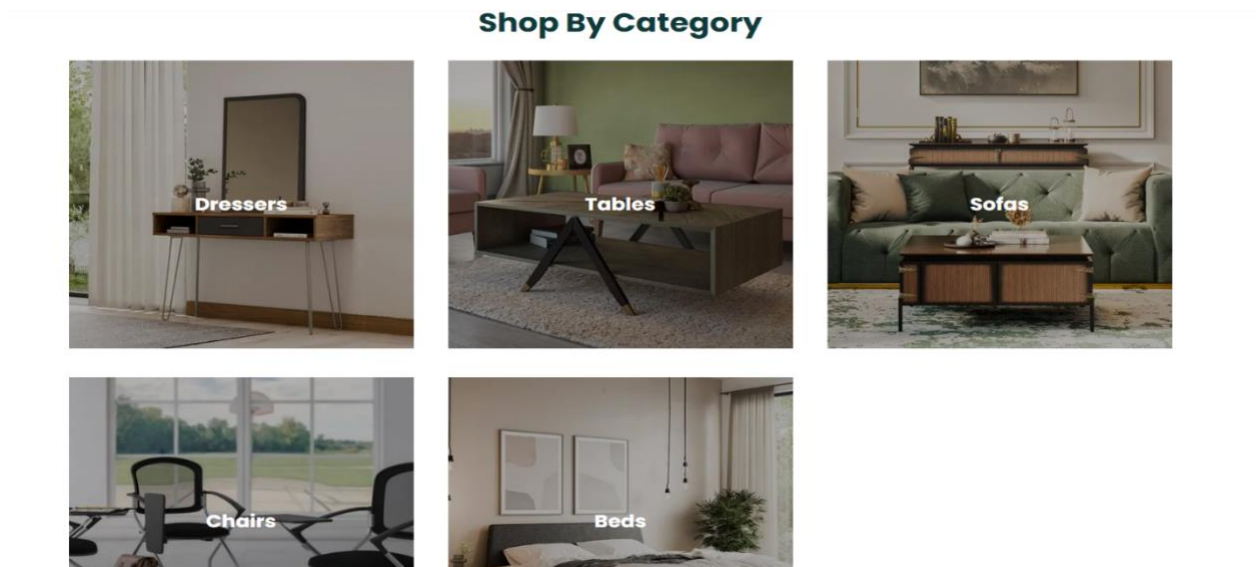
○ Product Detail Component:

Create individual product pages with detailed fields like description, price, sizes, and colors.



○ Category Component:



Display categories dynamically, enabling filtering by selected categories.



○ **Cart Component:**

Display added items, quantity, and total price, using state management for cart tracking.

Your Cart

Product	Subtotal	Quantity
 <div>Arc Econo Executive Chair Black Standard</div>	Rs. 41400	<div>- 1 +</div> 

Cart Totals

SubtotalRs. 41400

Total Items1

Proceed to Checkout

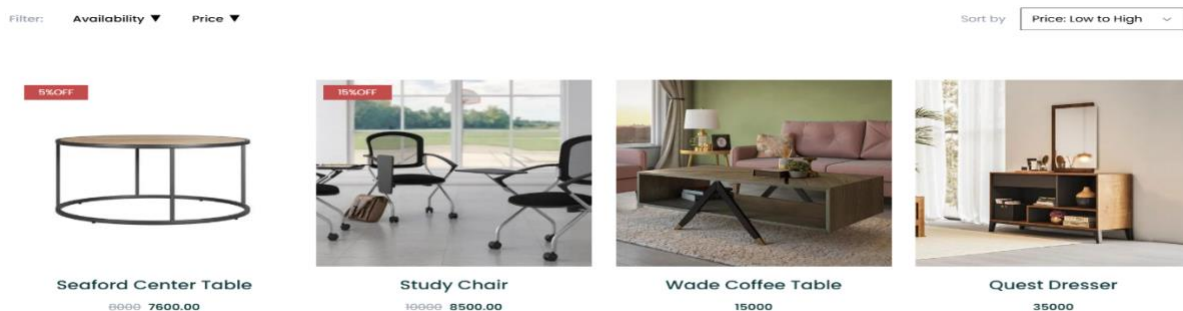
○ **Pagination Component:**

Break product lists into manageable pages with previous and next buttons.



○ **Filter Panel Component:**

Provide filtering options like price range, brand selection, and availability toggles.



- **Suggested Products Component:**

Suggest similar products based on tags, categories, or customer behaviors.

You may also like



MALM Dresser
~~40000~~ 38000.00



Wade Coffee Table
15000



Majestic Sofa Set
45000



Study Chair
~~10000~~ 8500.00

- **Order Tracking Component:**

Provide real-time updates on order status, including estimated delivery and location.

Track Your Order

Enter your phone number to track your order status.

Phone Number

TRACK ORDER

2. Code Deliverables:

- **ProductCard**

```
import { CardProps } from "../../types/cardProps";

const Card: React.FC<CardProps> = ({ data }) => {
  const salePrice = data.discountPercentage
    ? (data.price - (data.price * data.discountPercentage) / 100).toFixed(2)
    : undefined;

  const isSoldOut = data.inventory <= 0;

  return (
    <div className="group text-custom-green w-[300px] space-y-4 relative">
      { /* Tags */ }
      <div className="absolute left-2 z-40 top-2 space-y-2">
        {isSoldOut ? (
          <div className="bg-gray-400 text-white text-xs px-4 py-1">
            SOLD OUT
          </div>
        ) : (
          data.discountPercentage !== undefined &&
          data.discountPercentage > 0 && (
            <div className="bg-[#c34d4d] text-white text-xs px-4 py-1">
              {data.discountPercentage}%OFF
            </div>
          )
        )}
      </div>

      { /* Image Container */ }
      <div className="relative w-full h-[280px] md:h-[300px] flex items-center justify-center bg-transparent">
        { /* Default Image */ }
        <Image
          src={data.img}
          alt={data.heading}
          layout="fill"
          objectFit="cover"
          className="transition-opacity duration-300 opacity-100 group-hover:opacity-0"
        />

        { /* Hover Image */ }
        <Image
          src={data.hoverImg}
          alt={` ${data.heading}-hover`}
          layout="fill"
          objectFit="cover"
        />
      </div>
    </div>
  );
};
```

○ ProductDetail

```
import ProductImages from "@components/ProductImages";
import { notFound } from "next/navigation";
import {
  fetchProductBySlug,
  fetchRelatedProducts,
} from "@sanity/schemaTypes/dataFetchUtils";

interface ProductDetailsProps {
  params: Promise<{ slug: string }>;
}

const Product = async ({ params }: ProductDetailsProps) => {
  const { slug } = await params;
  const product = await fetchProductBySlug(slug);

  if (!product) {
    notFound();
  }

  // console.log(product)
  const relatedProducts = await fetchRelatedProducts(product.category, slug);

  return (
    <section className="space-y-12 py-10">
      <div className="max-w-screen-xl mx-auto grid grid-cols-1 lg:grid-cols-2 gap-8 md:gap-0">
        <ProductImages images={product.imageUrls} />
        <ProductData product={product} />
      </div>
      <hr />
      <RelatedProducts relatedProducts={relatedProducts} />
    </section>
  );
};

export default Product;
```

```

"use client";

import React, { useState } from "react";
import { AiFillStar } from "react-icons/ai";
import { FaStarHalfAlt } from "react-icons/fa";
import CartItemSheet from "../cart";
import { useCart } from "@app/context/cartContext";
import { toast } from "react-hot-toast";
import { Productdetail } from "../../types/product";

export interface ProductDataProps {
  product: Productdetail;
}

const ProductData: React.FC<ProductDataProps> = ({ product }) => {
  const [showCart, setShowCart] = useState(false);
  const [quantity, setQuantity] = useState(1);
  const [activeSize, setActiveSize] = useState<string>(product.sizes[0]);
  const [activeColor, setActiveColor] = useState<string>(product.colors[0]);

  const { addItem } = useCart();

  const salePrice = product.discountPercentage
    ? product.price - (product.price * product.discountPercentage) / 100
    : undefined;

  const incrementQuantity = () => setQuantity((prev) => prev + 1);
  const decrementQuantity = () => {
    if (quantity > 1) {
      setQuantity((prev) => prev - 1);
    }
  };

  console.log(product);

  const finalPrice = salePrice || product.price;

  const isOutOfStock = product.inventory === 0;

  const handleAddToCart = () => {
    if (!isOutOfStock) {
      addItem({
        id: product._id as string,
        name: product.productName,
        price: finalPrice,
        discount: product.discountPercentage
      });
    }
  };
};

```

```

const handleAddToCart = () => {
  if (!isOutOfStock) {
    addItem({
      id: product._id as string,
      name: product.productName,
      price: finalPrice,
      discount: product.discountPercentage,
      quantity,
      images: product.imageUrls[0],
      size: activeSize,
      color: activeColor,
    });

    toast.success(`${product.productName} added to cart`);
  } else {
    toast.error(`Sorry, this product is out of stock.`);
  }
};

return (
  <>
    <div className="px-2 lg:px-4 space-y-10 md:space-y-12">
      <div className="flex flex-col lg:w-[500px] xl:w-[606px] sm:px-4 space-y-3 md:space-y-4">
        <h1 className="text-3xl mb-2 md:mb-0 text-custom-green font-medium">
          {product.productName}
        </h1>

        <div className="flex items-center gap-4">
          {salePrice ? (
            <div className="text-xl">
              <span className="line-through text-gray-600">
                Rs. {product.price}
              </span>
              <span className="text-gray-950 font-semibold ml-2">
                Rs. {salePrice}
              </span>
            </div>
          ) : (
            <p className="text-xl text-gray-950 font-semibold">
              Rs. {product.price}
            </p>
          )}
        </div>
      </div>
    </div>
  </>
)

```