Name: Muhammad Hammad
Father Name: Muhammad Siddique
Roll No: 00413850
Days/Time: Saturday -07:00PM – 10:00PM

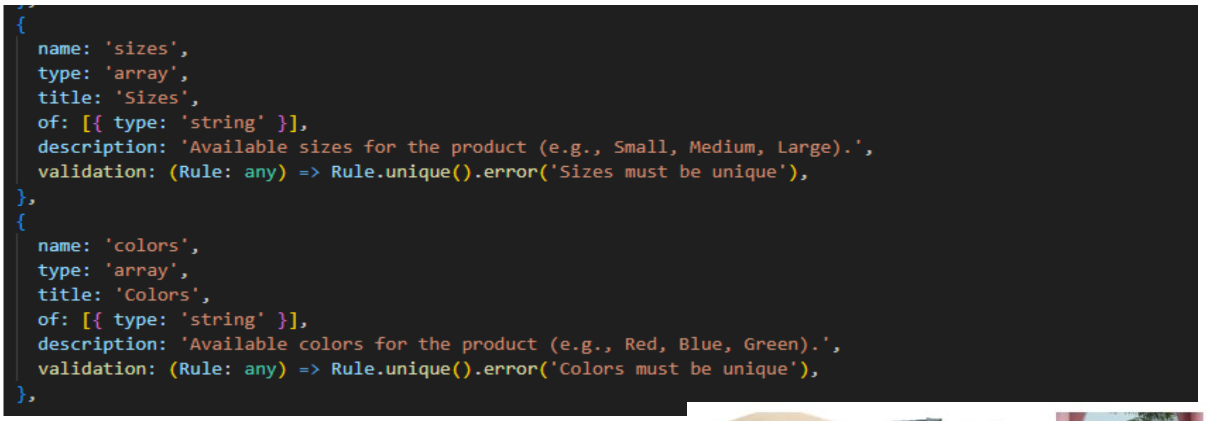
DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

Furniture E-commerce Marketplace

Day 4 Adjustments

On Day 4, additional fields were added to further enhance the schema and enable more customization for product details:

1. **New Field: Sizes**
   - **Type:** Array of strings
   - **Description:** Lists the available sizes for the product (e.g., Small, Medium, Large).
   - **Validation:** Ensures all sizes are unique to avoid duplication.
2. **New Field: Colors**
   - **Type:** Array of strings
   - **Description:** Lists the available colors for the product (e.g., Red, Blue, Green).
   - **Validation:** Ensures all colors are unique to prevent duplication.

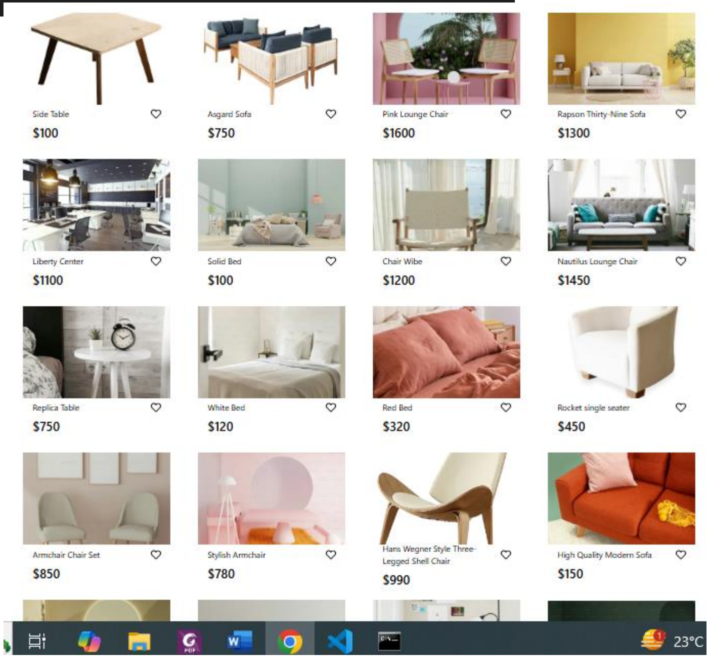


```
{
  name: 'sizes',
  type: 'array',
  title: 'Sizes',
  of: [{ type: 'string' }],
  description: 'Available sizes for the product (e.g., Small, Medium, Large).',
  validation: (Rule: any) => Rule.unique().error('Sizes must be unique'),
},
{
  name: 'colors',
  type: 'array',
  title: 'Colors',
  of: [{ type: 'string' }],
  description: 'Available colors for the product (e.g., Red, Blue, Green).',
  validation: (Rule: any) => Rule.unique().error('Colors must be unique'),
},
```

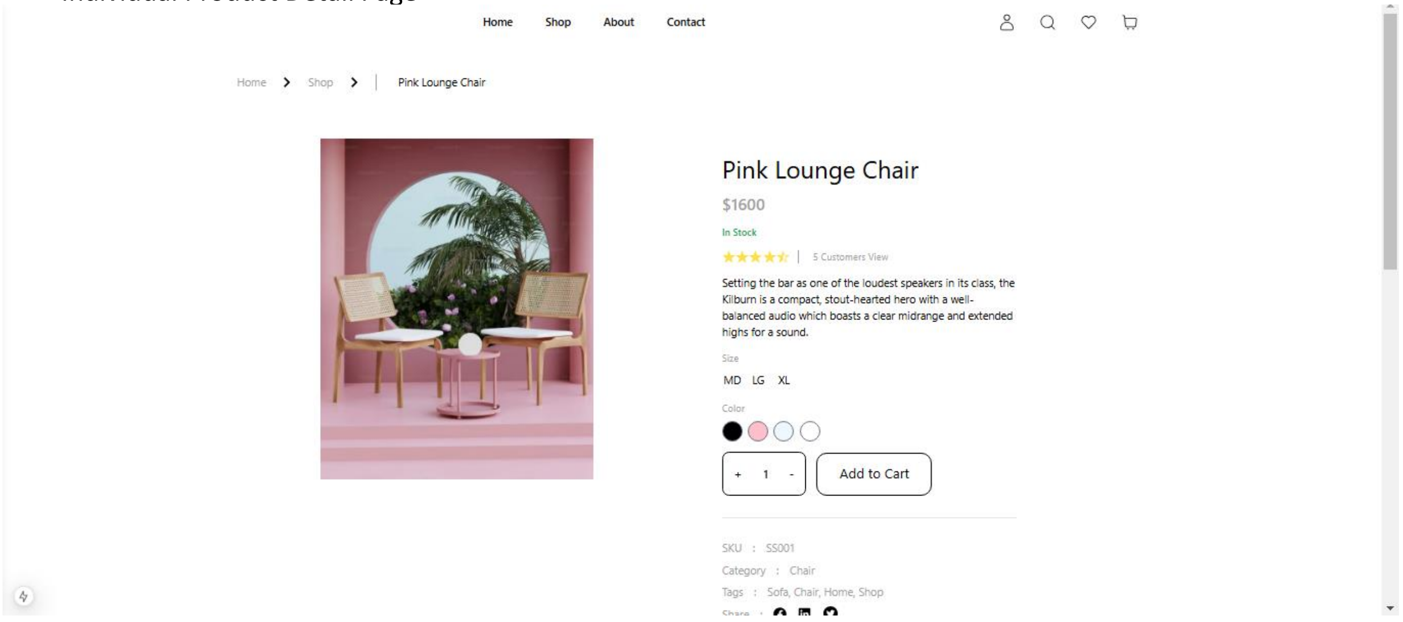
Submission Content Outline

1. **Functional Deliverables**
   Below are the components/screens implemented for the project. Screenshots have been captured for the following deliverables:

- Product Listing Page



- Individual Product Detail Page



Home Shop About Contact

Home > Shop > | Pink Lounge Chair

Pink Lounge Chair

$1600

In Stock

★★★★☆ | 5 Customers View

Setting the bar as one of the loudest speakers in its class, the Kilburn is a compact, stout-hearted hero with a well-balanced audio which boasts a clear midrange and extended highs for a sound.

Size

MD LG XL

Color

● ● ● ●

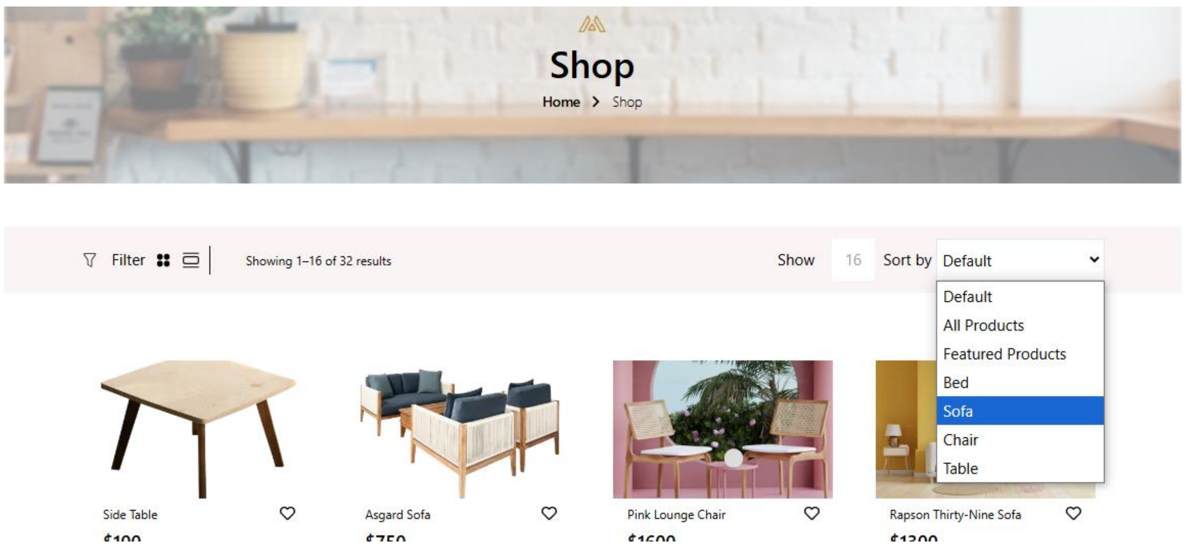
+ 1 - Add to Cart

SKU : SS001

Category : Chair

Tags : Sofa, Chair, Home, Shop

- Category Filters



Shop

Home > Shop

Filter | Showing 1–16 of 32 results

Show 16 Sort by Default

Default
All Products
Featured Products
Bed
Sofa
Chair
Table

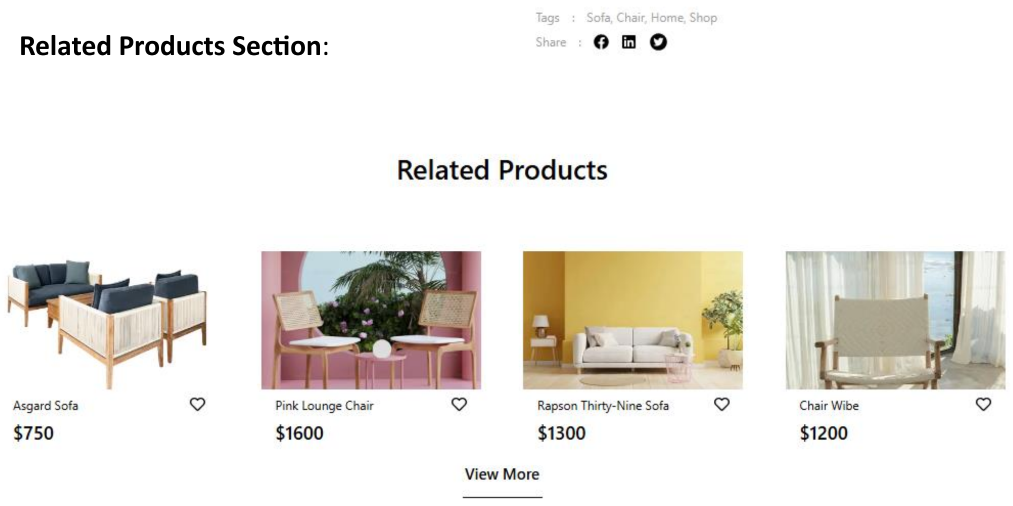
Side Table

Asgard Sofa

Pink Lounge Chair

Rapson Thirty-Nine Sofa

- **Related Products Section**:



Tags : Sofa, Chair, Home, Shop

Share :

Related Products

Asgard Sofa
$750

Pink Lounge Chair
$1600

Rapson Thirty-Nine Sofa
$1300

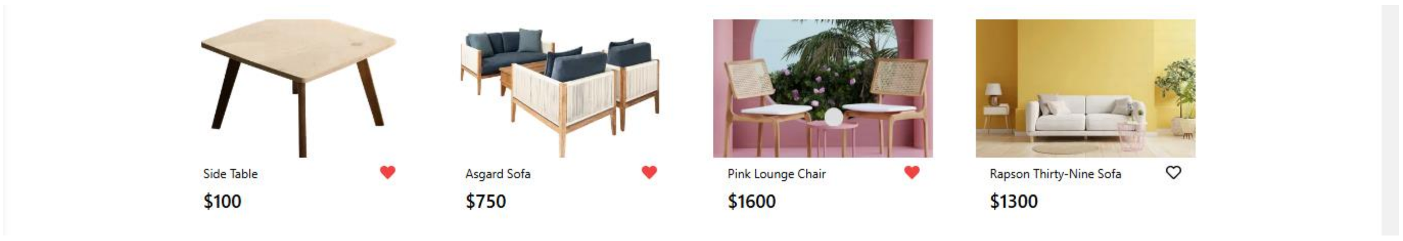
Chair Wibe
$1200

View More

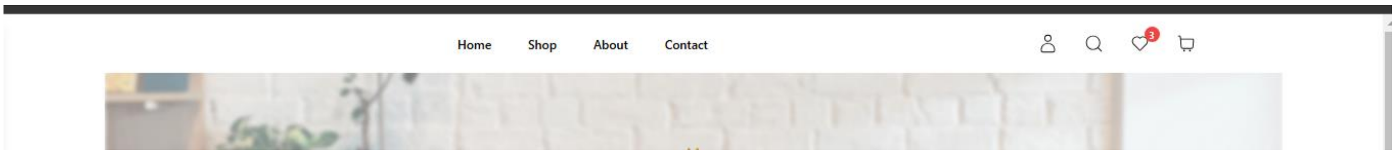
- **Pagination and Search Bar**

  The functionality for **pagination** and **search bar** was not implemented in this iteration. Corresponding components were created, but functionality remains pending.

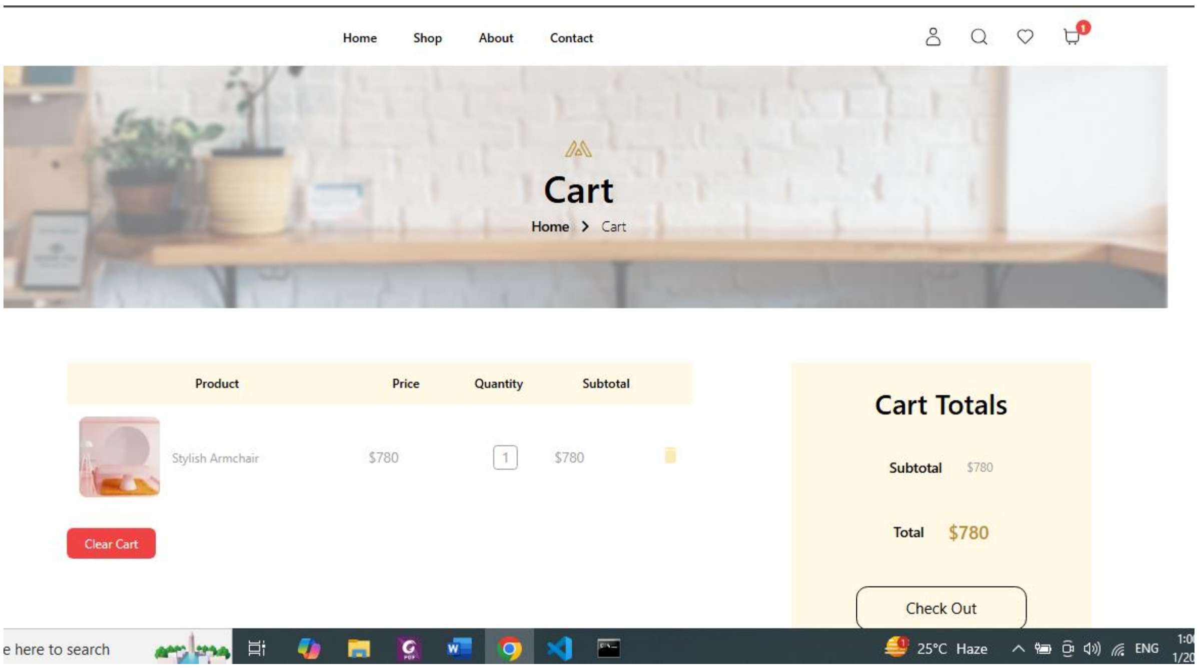
- **Add to Favorites**:
  "Users can now add products to their favorites easily."



- **Header with Numbering**:
  "Enhanced header with dynamic product numbering for better navigation."



- **Add to Cart**
  Content: The "Add to Cart" button is enabled only after selecting the size and color. Once a product is added, the header displays the updated cart number, reflecting the total items in the cart.



Product	Price	Quantity	Subtotal
Stylish Armchair	$780	1	$780

Cart Totals

Subtotal	$780
Total	$780

2. Code Deliverables:

Code snippets for my key components:

- ProductCard:

```tsx
'use client'

import { FavoriteContext } from "@/app/context/favContext";
import Image from "next/image";
import Link from "next/link";
import React, { useContext } from "react";
import { FaRegHeart, FaHeart } from "react-icons/fa";

interface ProductCardProps {
  _id: string;
  name: string;
  image: string;
  price: number;
}

interface FavoriteItem {
  _id: string;
  name: string;
  price: number;
  image: string;
}

const ProductCard: React.FC<ProductCardProps> = ({
  _id,
  name,
  image,
  price,
}) => {
  const { favorites, addToFavorites, removeFromFavorites } = useContext(FavoriteContext);

  // Check if the product is in the favorites list
  const isFavorite = favorites.some((favorite) => favorite._id === _id);

  const handleFavoriteClick = () => {
    const product: FavoriteItem = {
      _id,
      name,
      price,
      image,
    };

    if (isFavorite) {
      removeFromFavorites(_id);
    } else {
      addToFavorites(product);
    }
  };

  return (
    <div key={_id} className="group15 flex flex-col items-start">
      <div className="image-div mx-auto w-[200px] h-[150px] md:h-[180px] md:w-[300px] flex justify-
center items-center aspect-square overflow-hidden rounded-md">
        <Link href={`/shop/${_id}`}>
          <Image
            src={image}
            width={287}
            height={287}
            alt={name}
            className="object-cover"
          />
        </Link>
      </div>
```

```
        <div className="h-[71px] w-[194px] md:w-[250px] flex flex-col justify-center gap-2 mt-1 mx-auto">
          <div className="flex justify-between items-center">
            <p className="text-sm md:text-base lg:text-base xl:text-base font-normal">
              {name}
            </p>
            <button onClick={handleFavoriteClick} className="text-xl">
              {isFavorite ? (
                <FaHeart className="text-red-500" />
              ) : (
                <FaRegHeart />
              )}
            </button>
          </div>
          <p className="text-md md:text-lg lg:text-2xl xl:text-2xl font-medium">
            ${price}
          </p>
        </div>
      </div>
    );
  };

  export default ProductCard;

```

- ProductGrid

```

import { client } from "@/sanity/lib/client";
import PaginationButtons from "./paginationButtons";
import ProductCard from "./productCard";
import Filter from "./filter";

export interface Products {
  _id: string;
  name: string;
  image: string;
  price: number;
  featuredProducts?: boolean;
  category? : string;
}

export default async function ProductGrid({ filter }: { filter: string }) {

  let query = `*[_type == "product"]{
    name,
    _id,
    "image": image.asset->url,
    price,
    featuredProducts,
    category
  }`;

  if (filter === "featured-products") {
    query = `*[_type == "product" && isFeaturedProduct == true]{
      name,
      _id,
      "image": image.asset->url,
      price,
      featuredProducts
    }`;
  } else if (filter === "all-products") {
    query = `*[_type == "product"]{
      name,
      _id,
      "image": image.asset->url,
      price,
      featuredProducts
    }`;
  } else if (filter !== "default") {
    query = `*[_type == "product" && category == '${filter}']{
      name,
      _id,
      "image": image.asset->url,
      price,
      featuredProducts
    }`;
  }
  // console.log(filter)

  const products: Products[] = await client.fetch(query);

  return (
    <div className="font-poppins text-black bg-white">
      <Filter/>
      <div className="w-full max-w-[1536px] mx-auto flex flex-col items-center px-[50px] lg:px-[100px]">
        <div className="container pt-[50px] md:pt-[88px] flex flex-col md:flex-row md:grid md:grid-cols-2 xl:grid-cols-4 gap-4 md:gap-8 ">
          {products.map((product) => (
            <ProductCard
              key={product._id}
              _id={product._id}
              name={product.name}
              image={product.image}
              price={product.price}
            />
          ))}
        </div>
        <PaginationButtons />
      </div>
    </div>
  );
}

```

- Filter

```
"use client";
import React, { useState } from "react";
import { PiCirclesFourFill } from "react-icons/pi";
import { BsViewList } from "react-icons/bs";
import { CiFilter } from "react-icons/ci";
import { useRouter, useSearchParams } from "next/navigation";

export default function Filter() {
    const router = useRouter();
    const searchParams = useSearchParams();
    const currentFilter = searchParams.get("filter") || "default";

    const [selectedFilter, setSelectedFilter] = useState(currentFilter);

    const handleFilterChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
      const selectedValue = e.target.value;
      setSelectedFilter(selectedValue);

      // Update the URL with the selected filter
      router.push(`?filter=${selectedValue}`);
    };

  return (
    <div className="text-black font-poppins ">
      <div className="bg-[#FAF4F4] w-full max-w-[1536px] mx-auto h-auto mt-8 lg:mt-14 flex flex-wrap lg:flex-nowrap gap-4 px-4 py-4 lg:px-[100px] justify-between items-center">
        {/* Container 1 */}
        <div className="container-1 flex justify-center lg:justify-between items-center gap-3 w-full lg:w-auto">
```

```
          <div className="filter h-[25px] w-[60px ] lg:h-[30px] lg:w-[85px] flex justify-between items-center">
            <CiFilter className="text-lg lg:text-2xl" />
            <p className="font-normal text-base lg:text-xl"> Filter</p>
          </div>
          <PiCirclesFourFill className="text-lg lg:text-2xl" />
          <BsViewList className="text-lg lg:text-2xl" />
          <div className="font-normal text-xs sm:text-sm h-[30px] w-[190px] lg:text-base lg:h-[37px] lg:w-[237px] flex justify-end lg:justify-end items-center border-l-2 border-black">
            Showing 1–16 of 32 results
          </div>
        </div>

        {/* Container 2 */}
        <div className="container-2 flex justify-center lg:justify-between items-center gap-3 w-full lg:w-auto">
          <div className="h-[35px] w-[90px] lg:h-[55px] lg:w-[126px] font-normal text-sm lg:text-xl flex justify-between items-center">
            <p>Show</p>
            <div className=" h-[35px] w-[35px] flex justify-center items-center text-[#9F9F9F] lg:h-[55px] lg:w-[55px] bg-white">
              16
            </div>
          </div>

          {/* Sort by Dropdown */}
          <div className="h-[35px] w-[180px] lg:h-[55px] lg:w-[288px] font-normal text-sm lg:text-xl flex justify-between items-center">
            <p>Sort by</p>
            <select
              value={selectedFilter}
              onChange={handleFilterChange}
              className="h-[35px] w-[120px] lg:h-[55px] lg:w-[220px] bg-white border px-1"
            >
              <option value="default">Default</option>
              <option value="all-products">All Products</option>
              <option value="featured-products">Featured Products</option>
              <option value="Bed">Bed</option>
              <option value="Sofa">Sofa</option>
              <option value="Chair">Chair</option>
              <option value="Table">Table</option>
            </select>
          </div>
        </div>
      </div>
    </div>
  );
}

```

- AddtoCart/Detail of Product

```
"use client";
import { useCart } from "@/app/context/cardContext";
import Image from "next/image";
import React, { useState } from "react";
import { AiFillTwitterCircle } from "react-icons/ai";
import { FaFacebook, FaLinkedin } from "react-icons/fa";
import { IoStar, IoStarHalf } from "react-icons/io5";

interface ProductDetail {
  name: string;
  description: string;
  image: string;
  category: string;
  price: number;
  stockLevel: number;
  _id: string;
  sizes: string[];
  colors: string[];
}

interface ProductDetailComponentProps {
  product: ProductDetail;
}

const ProductDetailComponent: React.FC<ProductDetailComponentProps> = ({
  product,
}) => {
  const [quantity, setquantity] = useState<number>(1);
  const { addToCart } = useCart(); // Get the `addToCart` function from CartContext
  const [selectedSize, setSelectedSize] = useState<string>(""); // State for selected size
  const [selectedColor, setSelectedColor] = useState<string>(""); // State for selected size

  const handleAddToCart = () => {
    if (!selectedSize) {
      alert("Please select a size before adding to the cart."); // Prompt user to select size
      return; // Prevent adding to the cart if no size is selected
    }
    if (!selectedColor) {
      alert("Please select a color before adding to the cart."); // Prompt user to select size
      return; // Prevent adding to the cart if no size is selected
    }
    if (product) {
      addToCart({
        name: product.name,
        price: product.price,
        image: product.image,
        quantity: quantity,
        _id: product._id,
        total: product.price,
        size: selectedSize,
        color: selectedColor,
      });
    } else {
      alert("Please select a size before adding to cart.");
    }
  };

  return (
    <div className="product-div w-full max-w-[1536px] h-auto lg:h-[820px] mx-auto xl:py-8 px-4 sm:px-8
lg:px-16">
      <div className="h-auto w-full max-w-[1240px] mx-auto bg-white flex flex-col lg:flex-row items-
center lg:items-start justify-between gap-6 lg:gap-0">
        {/* Product Image */}
        <div className="picture-div h-[300px] sm:h-[400px] lg:h-[500px] w-full sm:w-[600px] lg:w-
[550px]flex items-center justify-center ">
          <Image
            src={product.image}
            width={550}
            height={500}
            alt={product.name}
            className="object-contain xl:w-[550px] xl:h-[500px]"
          />
        </div>

        {/* Product Details */}
        <div className="detail-div w-full lg:w-[600px] font-normal lg:pr-36 flex flex-col justify-
between gap-4 p-4 sm:p-6">
          {/* Product Name and Price */}
          <p className="text-2xl sm:text-3xl lg:text-4xl">{product.name}</p>
          <p className="text-xl sm:text-2xl lg:text-2xl text-[#9F9F9F] font-medium">
            ${product.price}
          </p>
          {/* Stock Level */}
          <p
```

```
            className={`text-sm font-medium ${
              product.stockLevel > 0 ? "text-green-600" : "text-red-600"
            }`}
          >
            {product.stockLevel > 0 ? "In Stock" : "Out of Stock"}
          </p>
          {/* Rating */}
          <div className="rating-div flex items-center gap-3">
            <div className="star-div flex justify-between items-center text-lg sm:text-xl text-yellow-
300">
              <IoStar />
              <IoStar />
              <IoStar />
              <IoStar />
              <IoStarHalf />
            </div>
            <div className="border-l-2 border-[#9F9F9F] px-3 sm:px-5 text-[#9F9F9F] text-sm">
              5 Customers View
            </div>
          </div>

          {/* Product Description */}
          <p className="text-sm sm:text-base">
            Setting the bar as one of the loudest speakers in its class, the
            Kilburn is a compact, stout-hearted hero with a well-balanced audio
            which boasts a clear midrange and extended highs for a sound.
          </p>

          {/* Size Options */}
          <div className="flex flex-col h-auto gap-2">
            <p className="text-sm text-[#9F9F9F]">Size</p>
            <div className="flex gap-2">
              {product.sizes.map((size, index) => (
                <div
                  key={index}
                  onClick={() => setSelectedSize(size)} // Set selected size on click
                  className={`h-[30px] w-[30px] cursor-pointer flex items-center justify-center rounded-
[5px] ${
                    selectedSize === size
                      ? "bg-[#FBEBB5]"
                      : "hover:bg-[#FBEBB5]"
                  }`}
                >
                  {size}
                </div>
              ))}
            </div>
          </div>

          {/* Color Options */}
          <div className="flex flex-col h-auto gap-2">
            <p className="text-sm text-[#9F9F9F]">Color</p>
            <div className="flex gap-2">
              {product.colors.map((color, index) => (
                <div
                  key={index}
                  onClick={() => setSelectedColor(color)} // Set selected color on click
                  style={{ backgroundColor: color }} // Set background color dynamically
                  className={`h-[30px] w-[30px] cursor-pointer flex items-center justify-center rounded-
full border ${
                    selectedColor === color ? "border-black" : "border-gray-500"
                  }`}
                ></div>
              ))}
            </div>
          </div>

          {/* Quantity and Add to Cart */}
```

```jsx
          <div className="flex gap-4 items-center">
            <div className="h-[45px] w-[75px] md:h-[64px] md:w-[123px] flex justify-around items-center border-2 border-black rounded-[10px]">
              <button
                className="text-lg"
                onClick={() => {
                  setquantity(quantity + 1);
                }}
                disabled={product.stockLevel === 0}
              >
                +
              </button>
              <p className="text-base font-medium">{quantity}</p>
              <button
                className="text-lg"
                onClick={() => {
                  if (quantity != 1) {
                    setquantity(quantity - 1);
                  }
                }}
                disabled={product.stockLevel === 0}
              >
                -
              </button>
            </div>
            <button
              className="text-base sm:text-xl py-2 md:py-4 px-4 md:px-8 sm:px-10 rounded-[15px] border-black border-2 whitespace-nowrap"
              disabled={product.stockLevel === 0}
              onClick={handleAddToCart} // Add to cart functionality
            >
              {product.stockLevel > 0 ? "Add to Cart" : "Out of Stock"}
            </button>
          </div>

          {/* Additional Information */}
          <hr className="my-4 text-[#D9D9D9]" />
          <div className="flex flex-col gap-2 text-[#9F9F9F] text-sm sm:text-base">
            <div className="flex gap-4">
              <p>SKU</p>
              <p>:</p>
              <p>SS001</p>
            </div>
            <div className="flex gap-4">
              <p>Category</p>
              <p>:</p>
              <p>{product.category}</p>
            </div>
            <div className="flex gap-4">
              <p>Tags</p>
              <p>:</p>
              <p>Sofa, Chair, Home, Shop</p>
            </div>
            <div className="flex gap-4 items-center">
              <p>Share</p>
              <p>:</p>
              <div className="flex items-center gap-4 text-black">
                <FaFacebook className="h-[20px] w-[20px]" />
                <FaLinkedin className="h-[20px] w-[20px]" />
                <AiFillTwitterCircle className="h-[25px] w-[25px]" />
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};
```

```
export default ProductDetailComponent;
```

- FavouriteContext

```
'use client';

import { createContext, useState, ReactNode } from "react";

export interface FavoriteItem {
  _id: string;
  name: string;
  price: number;
  image: string;
}

interface FavoriteContextType {
  favorites: FavoriteItem[];
  addToFavorites: (item: FavoriteItem) => void;
  removeFromFavorites: (slug: string) => void;
}

export const FavoriteContext = createContext<FavoriteContextType>({
  favorites: [],
  addToFavorites: () => {},
  removeFromFavorites: () => {},
});

export const FavoriteProvider = ({ children }: { children: ReactNode }) => {
  const [favorites, setFavorites] = useState<FavoriteItem[]>([]);
  const [notification, setNotification] = useState<string | null>(null);

  const showNotification = (message: string) => {
    setNotification(message);
    setTimeout(() => setNotification(null), 5000); // Hide after 5 seconds
  };

  const addToFavorites = (item: FavoriteItem) => {
    setFavorites((prevFavorites) => [...prevFavorites, item]);
    showNotification(`${item.name} added to favorites!`);
  };

  const removeFromFavorites = (slug: string) => {
    const newFavorites = favorites.filter((item) => item._id !== slug);
    setFavorites(newFavorites);
    showNotification(`Item removed from favorites!`);
  };

  return (
    <FavoriteContext.Provider
      value={{ favorites, addToFavorites, removeFromFavorites }}
    >
      {children}
      {notification && (
        <div className="text-sm md:text-lg fixed bottom-8 right-8 bg-green-500 text-white p-4 rounded 
shadow-lg animate-fade">
          {notification}
        </div>
      )}
    </FavoriteContext.Provider>
  );
};
```

- CartContext

```
'use client';
```

```
import React, { createContext, useContext, useState, ReactNode } from 'react';

export interface CartItem {
  _id: string;
  name: string;
  price: number;
  quantity: number;
  total: number;
  image: string;
  size:string;
  color:string;
}

interface CartContextType {
  cart: CartItem[];
  addToCart: (item: CartItem) => void;
  clearCart: () => void;
  deleteCartItem: (slug: string) => void;
}

const CartContext = createContext<CartContextType | undefined>(undefined);

export const CartProvider = ({ children }: { children: ReactNode }) => {
  const [cart, setCart] = useState<CartItem[]>([]);
  console.log(cart)
  const [notification, setNotification] = useState<string | null>(null);

  // Function to display notifications
  const showNotification = (message: string) => {
    setNotification(message);
    setTimeout(() => setNotification(null), 5000); // Hide after 5 seconds
  };

  const addToCart = (item: CartItem) => {

    setCart((prev) => {
      const existingItem = prev.find((cartItem) => cartItem._id === item._id);
      if (existingItem) {
        showNotification(`${item.name} quantity updated in the cart!`);
        return prev.map((cartItem) =>
          cartItem._id === item._id
            ? { ...cartItem, quantity: cartItem.quantity + item.quantity, total: cartItem.total + 
item.total }
            : cartItem
        );
      }

      showNotification(`${item.name} added to the cart!`);
      return [...prev, item];
    });
  };

  const deleteCartItem = (slug: string) => {
    const itemToDelete = cart.find((item) => item._id === slug);
    if (itemToDelete) {
      showNotification(`${itemToDelete.name} removed from the cart!`);
    }

    const newCart = cart.filter((val) => val._id !== slug);
    setCart([...newCart]);
  };

  const clearCart = () => {
    showNotification('Cart cleared!');
    setCart([]);
  };

  return (
    <CartContext.Provider value={{ cart, addToCart, clearCart, deleteCartItem }}>
      {children}

      {/* Notification UI */}
      {notification && (
        <div className="fixed text-sm md:text-lg  bottom-8 right-8 bg-green-500 text-white p-4 rounded
shadow-lg animate-fade">
          {notification}
        </div>
      )}
    </CartContext.Provider>
  );
};

export const useCart = () => {
  const context = useContext(CartContext);
  if (!context) throw new Error('useCart must be used within a CartProvider');
  return context;
};

```

- Header

```
"use client";
import { useCart } from "@/app/context/cardContext";
import { FavoriteContext } from "@/app/context/favContext";
import Link from "next/link";
import { useRouter } from "next/navigation"; // For navigation

import React, { useContext, useState } from "react";
import { CiShoppingCart, CiSearch, CiHeart, CiUser } from "react-icons/ci";
import { FiMenu, FiX } from "react-icons/fi";

export default function Header() {
  const [isMenuOpen, setIsMenuOpen] = useState(false);
  const { cart } = useCart();
  const { favorites } = useContext(FavoriteContext);

  // Calculate the total number of items in the cart
  const totalItems = cart.reduce((acc, item) => acc + item.quantity, 0);
  // Total number of favorite items
  const totalFavorites = favorites.length;

  return (
    <div className="font-poppins text-black relative">
      {/* Header Container */}
      <div className="bg-white w-full max-w-[1536px] mx-auto py-6 px-4 sm:px-8 flex justify-between
items-center">
        {/* Hamburger Menu for Small Screens */}
        <div className="sm:hidden">
          <button onClick={() => setIsMenuOpen(!isMenuOpen)}>
            {isMenuOpen ? <FiX size={28} /> : <FiMenu size={28} />}
          </button>
        </div>

        {/* Navigation Menu (Desktop Version) */}
        <div className="hidden sm:flex gap-12 flex-1 justify-center items-center">
          <ul className="flex flex-row items-center text-base font-medium text-center gap-12">
            <li>
              <Link href="/">Home</Link>
            </li>
            <li>
              <Link href="/shop">Shop</Link>
            </li>
            <li>
              <Link href="/blogs">About</Link>
            </li>
            <li>
              <Link href="/contact">Contact</Link>
            </li>
          </ul>
        </div>

        {/* Icons Section */}
        <div className="flex items-center gap-6 sm:gap-8 text-[24px] sm:text-[28px] md:pr-20">
          <Link href="/myAccount">
            <CiUser />
          </Link>
          <Link href="#">
            <CiSearch />
          </Link>
          <Link href="/favourite" className="relative">
            <CiHeart />
            {totalFavorites > 0 && (
              <span className="absolute -top-2 -right-3 bg-red-500 text-white text-xs font-bold rounded-full w-5 h-5 flex items-center justify-center">
                {totalFavorites}
              </span>
            )}
          </Link>
          <Link href="/cart" className="relative">
            <CiShoppingCart />
            {totalItems > 0 && (
              <span className="absolute -top-2 -right-3 bg-red-500 text-white text-xs font-bold rounded-full w-5 h-5 flex items-center justify-center">
                {totalItems}
              </span>
            )}
          </Link>
        </div>
      </div>

      {/* Mobile Slide-In Menu */}
      <div
        className={`fixed top-0 left-0 h-full w-[70%] bg-[#FBEBB5] z-50 shadow-lg transform ${
          isMenuOpen ? "translate-x-0" : "-translate-x-full"
        } transition-transform duration-300 ease-in-out`}
      >
        <button
          onClick={() => setIsMenuOpen(false)}
          className="absolute top-4 left-4 text-black text-2xl"
        >
          <FiX />
        </button>
        <ul className="flex flex-col items-center gap-6 mt-16 text-lg font-medium">
          <li>
            <Link href="/" onClick={() => setIsMenuOpen(false)}>
              Home
            </Link>
          </li>
          <li>
            <Link href="/shop" onClick={() => setIsMenuOpen(false)}>
              Shop
            </Link>
          </li>
          <li>
            <Link href="/blogs" onClick={() => setIsMenuOpen(false)}>
              About
            </Link>
          </li>
          <li>
            <Link href="/contact" onClick={() => setIsMenuOpen(false)}>
              Contact
            </Link>
          </li>
          </ul>
        </div>

        {/* Overlay when Menu is Open */}
        {isMenuOpen && (
          <div
            onClick={() => setIsMenuOpen(false)}
            className="fixed inset-0 bg-black bg-opacity-30 z-40"
          ></div>
        )}
      </div>
  );
}

```

Technical Report

1. Steps Taken to Build and Integrate Components

- **Product Grid**
  Built the ProductGrid component to dynamically fetch and display products from Sanity CMS. Implemented filtering based on categories like "featured-products" and "all-products."
- **Product Card**
  Created the ProductCard component to display individual product information such as image, name, price, and other relevant details.
- **Add to Cart**
  Integrated the "Add to Cart" functionality, where the button is only enabled when the user selects both size and color for the product.
- **Add to Favorites**
  Developed the "Add to Favorites" feature, allowing users to mark products they like. This involves tracking product IDs and saving them to a user's profile.
- **Header Cart Numbering**
  Implemented a dynamic cart system that updates the cart count in the header whenever an item is added.
- **Category Filters and Search Bar**
  Built the category filters to let users view products based on different categories. The search bar is prepared for future functionality to allow for keyword-based filtering.
- **Pagination**
  Integrated pagination to navigate through multiple product pages efficiently.

2. Challenges Faced and Solutions Implemented

- **Handling Dynamic Data Fetching**
  Challenge: Ensuring that data from Sanity CMS is correctly fetched based on user-selected filters.
  Solution: Used query parameters in the URL and fetched the filtered data dynamically using Sanity's client API. Utilized async/await to handle asynchronous data fetching.
- **Managing Product Variants (Sizes and Colors)**
  Challenge: Implementing the functionality where the "Add to Cart" button remains disabled until the user selects size and color.
  Solution: Added a state management system to handle the availability of the "Add to Cart" button and dynamically disable or enable it based on the selection of size and color.
- **Ensuring Smooth Routing for Product Details**
  Challenge: Implementing dynamic routing for individual product detail pages.

Solution: Used Next.js dynamic routing to fetch data for each product based on its ID, ensuring that the correct product data is displayed.

- **Handling Undefined Filters**
  Challenge: Managing undefined or incorrect filter values from the query string.
  Solution: Implemented default filters and checks to ensure that invalid or undefined filter values fall back to a default state.

3. Best Practices Followed During Development

- **Component Reusability**
  Designed components like ProductCard and Filter with reusability in mind, ensuring they can be used across various sections without significant changes.
- **State Management**
  Used React's state management for managing the status of components (like "Add to Cart" button) and ensuring UI reflects changes in real-time.
- **Responsive Design**
  Ensured all components and pages are mobile-responsive, offering an optimized experience across different devices and screen sizes.