HACKATHON 3 (DAY 04)

Building Dynamic Frontend Components For Market Place

I have built above 20 components for Avion; some are given below with code snippets The full website recording is attached to din folder day 4.  
  
**2. Code Deliverables:**

**Product Listing  
"use client";**

**import \* as React from "react";**

**import { useState, useEffect } from "react";**

**import { useSearchParams } from "next/navigation";**

**import Image from "next/image";**

**import { Navbar } from "@/components/Navbar";**

**import Footer2 from "@/components/footer2";**

**import { Button } from "@/components/ui/button";**

**import {**

**Accordion,**

**AccordionContent,**

**AccordionItem,**

**AccordionTrigger,**

**} from "@/components/ui/accordion";**

**import { Checkbox } from "@/components/ui/checkbox";**

**import sanityClient from "@sanity/client";**

**import Link from "next/link";**

**import toast, { Toaster } from "react-hot-toast";**

**import { auth } from "../../../firebase/firebase";**

**import { User } from "firebase/auth";**

**const sanity = sanityClient({**

**projectId: "ah48gcwm",**

**dataset: "production",**

**apiVersion: "2024-01-04",**

**useCdn: true,**

**});**

**interface Category {**

**name: string;**

**slug: string;**

**}**

**interface Dimensions {**

**width: string;**

**height: string;**

**depth: string;**

**}**

**interface Products {**

**name: string;**

**description: string;**

**image: string;**

**\_id: string;**

**features: string[];**

**dimensions: Dimensions;**

**category: Category;**

**price: number;**

**tags: string[];**

**quantity: number;**

**}**

**interface FilterOptions {**

**categories: string[];**

**priceRange: string;**

**page: number;**

**productsPerPage: number;**

**}**

**async function fetchCategories() {**

**const categories = await sanity.fetch(**

**`\*[\_type == "category"]{**

**name,**

**slug**

**}`**

**);**

**return categories;**

**}**

**async function fetchProducts(**

**filterOptions: FilterOptions,**

**searchTerm?: string**

**) {**

**const { categories, priceRange, page, productsPerPage } = filterOptions;**

**let priceFilter = "";**

**if (priceRange) {**

**if (priceRange === "250+") {**

**priceFilter = "price >= 250";**

**} else {**

**const [min, max] = priceRange.split(" - ").map(Number);**

**priceFilter = `price >= ${min} && price <= ${max}`;**

**}**

**}**

**const categoryFilter =**

**categories.length > 0**

**? `category->name in [${categories.map((c) => `"${c}"`).join(", ")}]`**

**: "";**

**const queryFilters = [**

**`\_type == "product"`,**

**categoryFilter,**

**priceFilter,**

**searchTerm ? `name match "${searchTerm}\*"` : "",**

**]**

**.filter(Boolean)**

**.join(" && ");**

**const query = `\*[${queryFilters}]{**

**\_id,**

**name,**

**"image": image.asset->url,**

**price,**

**features,**

**dimensions,**

**category->,**

**tags,**

**quantity**

**}[${(page - 1) \* productsPerPage}...${page \* productsPerPage}]`;**

**const products = await sanity.fetch(query);**

**return products;**

**}**

**export default function ProductListing() {**

**const searchParams = useSearchParams();**

**const searchTerm = searchParams.get("search") || "";**

**const [selectedCategories, setSelectedCategories] = useState<string[]>([]);**

**const [selectedPriceRange, setSelectedPriceRange] = useState<string>("");**

**const [products, setProducts] = useState<Products[]>([]);**

**const [categories, setCategories] = useState<Category[]>([]);**

**const [isLoading, setIsLoading] = useState(true);**

**const [currentPage, setCurrentPage] = useState(1);**

**const [productsPerPage] = useState(12);**

**const [totalProducts, setTotalProducts] = useState(0);**

**const [isModalOpen, setModalOpen] = useState(false);**

**const [user, setUser] = useState<User | null>(null);**

**const [userWishlist, setUserWishlist] = useState<Products[]>([]);**

**useEffect(() => {**

**const unsubscribe = auth.onAuthStateChanged((currentUser) => {**

**setUser(currentUser); // Set the user state based on auth status**

**});**

**return () => unsubscribe(); // Cleanup listener**

**}, []);**

**useEffect(() => {**

**async function loadInitialData() {**

**try {**

**setIsLoading(true);**

**const categoriesData = await fetchCategories();**

**const uniqueCategories = Array.from(**

**new Set(categoriesData.map((cat: Category) => cat.name))**

**).map((name) =>**

**categoriesData.find((cat: Category) => cat.name === name)**

**);**

**setCategories(uniqueCategories);**

**const initialProducts = await fetchProducts(**

**{**

**categories: [],**

**priceRange: "",**

**page: currentPage,**

**productsPerPage,**

**},**

**searchTerm ? decodeURIComponent(searchTerm) : undefined**

**);**

**setProducts(initialProducts);**

**} catch (error) {**

**console.error("Error loading initial data:", error);**

**} finally {**

**setIsLoading(false);**

**}**

**}**

**loadInitialData();**

**}, [searchTerm, currentPage]);**

**useEffect(() => {**

**async function loadFilteredProducts() {**

**try {**

**setIsLoading(true);**

**const filteredProducts = await fetchProducts(**

**{**

**categories: selectedCategories,**

**priceRange: selectedPriceRange,**

**page: currentPage,**

**productsPerPage,**

**},**

**searchTerm ? decodeURIComponent(searchTerm) : undefined**

**);**

**setProducts(filteredProducts);**

**} catch (error) {**

**console.error("Error loading filtered products:", error);**

**} finally {**

**setIsLoading(false);**

**}**

**}**

**loadFilteredProducts();**

**}, [selectedCategories, selectedPriceRange, searchTerm, currentPage]);**

**const handleCategoryChange = (categoryName: string) => {**

**setSelectedCategories((prev) => {**

**if (prev.includes(categoryName)) {**

**return prev.filter((cat) => cat !== categoryName);**

**} else {**

**return [...prev, categoryName];**

**}**

**});**

**};**

**const handlePriceChange = (priceRange: string) => {**

**setSelectedPriceRange((prev) => (prev === priceRange ? "" : priceRange));**

**};**

**const filterConfig = {**

**"Product type": categories.map((category) => category.name),**

**Price: ["0 - 100", "101 - 250", "250+"],**

**};**

**useEffect(() => {**

**if (user) {**

**// Fetch the user's wishlist from localStorage if available**

**const wishlist = JSON.parse(**

**localStorage.getItem(user.uid + "\_wishlist") || "[]"**

**);**

**console.log("Fetched wishlist from localStorage:", wishlist); // Debugging log**

**setUserWishlist(wishlist);**

**}**

**}, [user]);**

**const addToWishlist = (product: Products) => {**

**if (!user) {**

**setModalOpen(true); // Show modal for unauthenticated users**

**return;**

**}**

**const wishlist = JSON.parse(**

**localStorage.getItem(user.uid + "\_wishlist") || "[]"**

**);**

**const isInWishlist = wishlist.some(**

**(item: Products) => item.\_id === product.\_id**

**);**

**if (isInWishlist) {**

**toast.error(`${product.name} is already in your wishlist!`, {**

**position: "bottom-right",**

**});**

**} else {**

**wishlist.push(product);**

**localStorage.setItem(user.uid + "\_wishlist", JSON.stringify(wishlist)); // Update localStorage**

**setUserWishlist(wishlist); // Update the local state to reflect changes**

**console.log("Updated wishlist in localStorage:", wishlist); // Debugging log**

**toast.success(`${product.name} added to wishlist!`, {**

**position: "bottom-right",**

**});**

**}**

**};**

**const totalPages = Math.ceil(3);**

**return (**

**<>**

**<Toaster />**

**<Navbar />**

**<div className="min-h-screen bg-white">**

**<div className="relative h-[300px] w-full">**

**<Image**

**src="/Frame 143.png"**

**alt="All Products"**

**fill**

**className="object-cover brightness-75"**

**/>**

**<h1 className="font-clash absolute bottom-8 left-4 text-3xl font-normal text-white sm:left-8 lg:left-12">**

**{searchTerm**

**? `Search Results for "${decodeURIComponent(searchTerm)}" (${products.length})`**

**: `All products (${products.length})`}**

**</h1>**

**</div>**

**<div className="mx-auto max-w-[1440px] px-4 py-8 sm:px-6 lg:px-8">**

**<div className="lg:grid lg:grid-cols-[240px,1fr] lg:gap-x-8">**

**<div className="hidden lg:block">**

**<Accordion type="single" collapsible className="w-full">**

**{Object.entries(filterConfig).map(([category, options]) => (**

**<AccordionItem key={category} value={category}>**

**<AccordionTrigger className="text-base font-normal">**

**{category}**

**</AccordionTrigger>**

**<AccordionContent>**

**<div className="space-y-4">**

**{options.map((option) => (**

**<div**

**key={`${category}-${option}`}**

**className="flex items-center space-x-2"**

**>**

**<Checkbox**

**id={`${category}-${option}`}**

**checked={**

**category === "Product type"**

**? selectedCategories.includes(option)**

**: selectedPriceRange === option**

**}**

**onCheckedChange={() =>**

**category === "Product type"**

**? handleCategoryChange(option)**

**: handlePriceChange(option)**

**}**

**/>**

**<label**

**htmlFor={`${category}-${option}`}**

**className="text-sm font-normal leading-none"**

**>**

**{option}**

**</label>**

**</div>**

**))}**

**</div>**

**</AccordionContent>**

**</AccordionItem>**

**))}**

**</Accordion>**

**</div>**

**<div className="mt-6 lg:mt-0">**

**{isLoading ? (**

**<div className="flex justify-center items-center">**

**<div className="spinner-border animate-spin border-4 border-[#2A254B] border-t-transparent rounded-full w-12 h-12"></div>**

**</div>**

**) : (**

**<div>**

**{products.length > 0 ? (**

**<div className="grid grid-cols-2 gap-x-4 gap-y-8 sm:gap-x-6 md:grid-cols-3 lg:gap-x-8">**

**{products.map((product) => (**

**<div key={product.\_id} className="group">**

**<div className="w-[300px] h-[300px] mx-auto aspect-h-1 aspect-w-1 overflow-hidden bg-gray-100">**

**<Image**

**src={product.image}**

**alt={product.name}**

**width={500}**

**height={500}**

**className="w-full h-full object-cover object-center"**

**/>**

**</div>**

**<div className="mt-4 flex justify-between">**

**<div>**

**<h3 className="font-clash text-xl font-extrabold text-gray-900">**

**{product.name}**

**</h3>**

**<p className="text-sm font-clash text-gray-500">**

**€{product.price}**

**</p>**

**<p className="text-sm font-clash text-gray-500">**

**{product.quantity} Pieces**

**</p>**

**</div>**

**</div>**

**{product.\_id ? (**

**<>**

**<Link href={`/products/${product.\_id}`}>**

**<Button**

**className="w-full transition-transform duration-200 hover:scale-105 active:scale-95 mt-1"**

**variant="outline"**

**style={{**

**fontFamily: "var(--font-clash-reg)",**

**}}**

**>**

**View Details**

**</Button>**

**</Link>**

**<Button**

**className="w-full transition-transform duration-200 hover:scale-105 active:scale-95 mt-1 bg-[#2A254B] hover:bg-[#2A254B]/90 text-white font-clash"**

**onClick={() => addToWishlist(product)}**

**>**

**Add to Wishlist**

**</Button>**

**</>**

**) : (**

**<Button**

**disabled**

**className="w-full mt-1"**

**variant="outline"**

**>**

**No Details**

**</Button>**

**)}**

**</div>**

**))}**

**</div>**

**) : (**

**<div className="text-center py-12">**

**<h2 className="text-2xl font-semibold text-gray-700 mb-4">**

**No Products Found**

**</h2>**

**{searchTerm && (**

**<p className="text-gray-500 mb-6">**

**We couldn't find any products matching "**

**{decodeURIComponent(searchTerm)}".**

**</p>**

**)}**

**<p>Try adjusting your search or filters.</p>**

**</div>**

**)}**

**<div className="flex justify-center mt-8 font-clash">**

**<Button**

**disabled={currentPage === 1}**

**onClick={() => setCurrentPage(currentPage - 1)}**

**>**

**Previous**

**</Button>**

**<span className="mx-4">**

**Page {currentPage} of {totalPages}**

**</span>**

**<Button**

**disabled={currentPage === 2} // Disable the "Next" button after the second page**

**onClick={() => setCurrentPage(currentPage + 1)}**

**>**

**Next**

**</Button>**

**</div>**

**</div>**

**)}**

**</div>**

**</div>**

**</div>**

**</div>**

**<Footer2 />**

**{!user && isModalOpen && (**

**<div**

**className="fixed inset-0 bg-black bg-opacity-50 flex justify-center items-center z-50"**

**onClick={() => setModalOpen(false)}**

**>**

**<div**

**className="bg-white p-6 w-4/5 max-w-xl relative z-60"**

**onClick={(e) => e.stopPropagation()}**

**>**

**<button**

**className="absolute top-4 right-4 text-gray-500 hover:text-gray-700"**

**onClick={() => setModalOpen(false)}**

**>**

**&times;**

**</button>**

**<div className="text-center">**

**<p className="text-gray-500 mb-4 font-clash">**

**Please <strong>Sign In</strong> or <strong>Sign Up</strong> to**

**add this item to your wishlist.**

**</p>**

**<Link href="/acc-creation">**

**<Button className="rounded-none bg-[#2A254B] px-8 hover:bg-[#2A254B]/90 font-clash">**

**Sign In / Sign Up**

**</Button>**

**</Link>**

**</div>**

**</div>**

**</div>**

**)}**

**</>**

**);**

**}**

**Product Detail   
"use client"**

**import { useEffect, useState } from "react"**

**import { Toaster, toast } from "react-hot-toast"**

**import sanityClient from "@sanity/client"**

**import { Navbar } from "@/components/Navbar"**

**import Image from "next/image"**

**import { Button } from "@/components/ui/button"**

**import Footer from "@/components/footer"**

**import News from "@/components/newsletter-section"**

**import Link from "next/link"**

**import { auth } from "../../../../firebase/firebase"**

**import type { User } from "firebase/auth"**

**import { RatingsAndReviews } from "@/components/RatingsAndReviews"**

**// Sanity client initialization**

**const sanity = sanityClient({**

**projectId: "ah48gcwm",**

**dataset: "production",**

**apiVersion: "2024-01-04",**

**useCdn: true,**

**})**

**// Type definitions for the product and dimensions**

**interface Dimensions {**

**width: string**

**height: string**

**depth: string**

**}**

**interface CategoryReference {**

**\_ref: string**

**\_type: string**

**}**

**interface Products {**

**name: string**

**description: string**

**image: string**

**\_id: string**

**features: string[]**

**dimensions: Dimensions**

**category: CategoryReference**

**price: number**

**tags: string[]**

**quantity: number**

**}**

**// Fetch product by ID function**

**async function fetchProductById(id: string) {**

**const product = await sanity.fetch(**

**`**

**\*[\_type == "product" && \_id == $id][0] {**

**\_id,**

**name,**

**"image": image.asset->url,**

**price,**

**description,**

**features,**

**dimensions,**

**category,**

**tags,**

**quantity**

**}**

**`,**

**{ id },**

**)**

**return product**

**}**

**// Fetch related products based on category**

**async function fetchRelatedProducts(categoryId: string, currentProductId: string) {**

**const relatedProducts = await sanity.fetch(**

**`**

**\*[\_type == "product" && category.\_ref == $categoryId && \_id != $currentProductId][0...4] {**

**\_id,**

**name,**

**"image": image.asset->url,**

**price**

**}**

**`,**

**{ categoryId, currentProductId },**

**)**

**return relatedProducts**

**}**

**export default function ProductDetailsPage({**

**params,**

**}: {**

**params: Promise<{ id: string }>**

**}) {**

**const [product, setProduct] = useState<Products | null>(null)**

**const [relatedProducts, setRelatedProducts] = useState<Products[]>([])**

**const [id, setId] = useState<string | null>(null)**

**const [isModalOpen, setModalOpen] = useState(false)**

**const [user, setUser] = useState<User | null>(null)**

**// Fetch product ID from URL params**

**useEffect(() => {**

**const getId = async () => {**

**const resolvedParams = await params**

**setId(resolvedParams.id)**

**}**

**getId()**

**}, [params])**

**// Fetch product and related products**

**useEffect(() => {**

**if (id) {**

**const fetchProduct = async () => {**

**const fetchedProduct = await fetchProductById(id)**

**setProduct(fetchedProduct)**

**if (fetchedProduct) {**

**const relatedProducts = await fetchRelatedProducts(fetchedProduct.category.\_ref, fetchedProduct.\_id)**

**setRelatedProducts(relatedProducts)**

**}**

**}**

**fetchProduct()**

**}**

**}, [id])**

**// Listen to user authentication state change**

**useEffect(() => {**

**const unsubscribe = auth.onAuthStateChanged((currentUser) => {**

**setUser(currentUser) // Set the user state based on auth status**

**})**

**return () => unsubscribe() // Cleanup listener**

**}, [])**

**// Handle adding to cart**

**const addToCart = (product: Products) => {**

**if (!user) {**

**setModalOpen(true) // Show modal for unauthenticated users**

**return**

**}**

**// Retrieve the cart for the current user from localStorage**

**const cart = JSON.parse(localStorage.getItem(user.uid + "\_cart") || "[]")**

**// Check if the product is already in the cart**

**const existingProductIndex = cart.findIndex((item: Products) => item.\_id === product.\_id)**

**if (existingProductIndex > -1) {**

**// Update the quantity of the existing product**

**cart[existingProductIndex].quantity += 1**

**toast.success(`Increased quantity of ${product.name} to ${cart[existingProductIndex].quantity}`, {**

**position: "bottom-right",**

**})**

**} else {**

**// Add the product to the cart with quantity set to 1**

**cart.push({ ...product, quantity: 1 })**

**toast.success(`${product.name} added to cart!`, {**

**position: "bottom-right",**

**})**

**}**

**// Save the updated cart to localStorage**

**localStorage.setItem(user.uid + "\_cart", JSON.stringify(cart))**

**}**

**if (!product) return <div>Loading...</div>**

**return (**

**<>**

**<Toaster />**

**<Navbar />**

**<div className="container mx-auto px-4 py-8">**

**<div className="grid md:grid-cols-2 gap-8 mb-16">**

**<div className="bg-gray-100">**

**<Image**

**src={product.image || "/placeholder.svg"}**

**alt={product.name}**

**width={600}**

**height={600}**

**className="w-[500px] h-[500px] mx-auto"**

**/>**

**</div>**

**<div className="self-start mt-6">**

**<h1 className="font-clash text-4xl font-medium mb-2">{product.name}</h1>**

**<p className="text-2xl font-medium mb-6 font-clash">£{product.price}</p>**

**<div className="mb-6">**

**<h2 className="font-medium mb-2 font-clash">Description</h2>**

**<p className="text-gray-600 font-clash">{product.description}</p>**

**</div>**

**<ul className="list-disc list-inside mb-6 text-gray-600 font-clash">**

**{product.features.map((feature, index) => (**

**<li key={index}>{feature}</li>**

**))}**

**</ul>**

**<div className="mb-6">**

**<h2 className="font-medium mb-2 font-clash">Dimensions</h2>**

**<div className="grid grid-cols-3 gap-4 text-gray-600 font-clash">**

**<div>**

**<p className="font-medium font-clash">Height</p>**

**<p>{product.dimensions.height}</p>**

**</div>**

**<div>**

**<p className="font-medium font-clash">Width</p>**

**<p>{product.dimensions.width}</p>**

**</div>**

**<div>**

**<p className="font-medium font-clash">Depth</p>**

**<p>{product.dimensions.depth}</p>**

**</div>**

**</div>**

**</div>**

**<Button**

**onClick={() => addToCart(product)}**

**className="w-auto bg-[#2A254B] hover:bg-[#2A254B]/90 text-white font-clash"**

**>**

**Add to cart**

**</Button>**

**</div>**

**</div>**

**{product && <RatingsAndReviews productId={product.\_id} />}**

**{relatedProducts.length > 0 && (**

**<div className="mt-16">**

**<h2 className="font-clash text-3xl font-medium mb-8 text-center">Related Products</h2>**

**<div className="flex justify-center">**

**<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6 justify-self-center">**

**{relatedProducts.map((relatedProduct) => (**

**<div key={relatedProduct.\_id}>**

**<div className="border-0 shadow-none group">**

**<div className="aspect-square relative overflow-hidden mb-3">**

**<Image**

**src={relatedProduct.image || "/placeholder.svg"}**

**alt={relatedProduct.name}**

**width={300}**

**height={300}**

**className="object-cover w-full h-full transition-transform duration-300 group-hover:scale-105"**

**/>**

**</div>**

**<h3 className="font-clash text-xl font-extrabold text-gray-900">{relatedProduct.name}</h3>**

**<p className="text-sm font-clash text-gray-500">£{relatedProduct.price}</p>**

**<p className="text-sm font-clash text-gray-500">{relatedProduct.quantity} Pieces</p>**

**<Link href={`/products/${relatedProduct.\_id}`}>**

**<Button className="w-full transition-transform duration-200 hover:scale-105 active:scale-95">**

**View Details**

**</Button>**

**</Link>**

**</div>**

**</div>**

**))}**

**</div>**

**</div>**

**</div>**

**)}**

**</div>**

**<News />**

**<Footer />**

**{!user && isModalOpen && (**

**<div**

**className="fixed inset-0 bg-black bg-opacity-50 flex justify-center items-center z-50"**

**onClick={() => setModalOpen(false)}**

**>**

**<div className="bg-white p-6 w-4/5 max-w-xl relative z-60" onClick={(e) => e.stopPropagation()}>**

**<button**

**className="absolute top-4 right-4 text-gray-500 hover:text-gray-700"**

**onClick={() => setModalOpen(false)}**

**>**

**&times;**

**</button>**

**<div className="text-center">**

**<p className="text-gray-500 mb-4 font-clash">**

**Please <strong>Sign In</strong> or <strong>Sign Up</strong> to add this item to your cart.**

**</p>**

**<Link href="/acc-creation">**

**<Button className="rounded-none bg-[#2A254B] px-8 hover:bg-[#2A254B]/90 font-clash">**

**Sign In / Sign Up**

**</Button>**

**</Link>**

**</div>**

**</div>**

**</div>**

**)}**

**</>**

**)**

**}**

**Search Bar   
"use client";**

**import React, { useEffect, useState } from "react";**

**import { useRouter } from "next/navigation";**

**import Link from "next/link";**

**import { Search, Menu, Heart, UserRound, Store, BarChart } from "lucide-react"; // Updated icons**

**import { Button } from "@/components/ui/button";**

**import { Input } from "@/components/ui/input";**

**import Image from "next/image";**

**import { auth } from "../../firebase/firebase";**

**import { signOut } from "firebase/auth";**

**import { User as fbUser } from "firebase/auth";**

**import { getUserType } from "../../firebase/auth"; // Import getUserType**

**export function Navbar() {**

**const [isModalOpen, setModalOpen] = useState(false);**

**const [isMobileMenuOpen, setMobileMenuOpen] = useState(false);**

**const [searchTerm, setSearchTerm] = useState("");**

**const [user, setUser] = useState<fbUser | null>(null);**

**const [userType, setUserType] = useState<string | null>(null);**

**const [loading, setLoading] = useState(true); // Loading state**

**const router = useRouter();**

**useEffect(() => {**

**const unsubscribe = auth.onAuthStateChanged((currentUser) => {**

**setUser(currentUser);**

**setUserType(currentUser ? getUserType(currentUser) : null);**

**setLoading(false); // Set loading to false after fetching user data**

**});**

**return () => unsubscribe();**

**}, []);**

**const handleSearch = () => {**

**if (searchTerm.trim()) {**

**const encodedSearchTerm = encodeURIComponent(searchTerm.trim());**

**router.push(`/product-listing?search=${encodedSearchTerm}`);**

**setModalOpen(false);**

**}**

**};**

**const handleKeyPress = (e: React.KeyboardEvent<HTMLInputElement>) => {**

**if (e.key === "Enter") {**

**handleSearch();**

**}**

**};**

**const handleSignOut = async () => {**

**try {**

**await signOut(auth);**

**if (user) {**

**localStorage.removeItem(`wishlist\_${user.uid}`);**

**}**

**router.push("/");**

**} catch (error) {**

**console.error("Error signing out:", error);**

**}**

**};**

**// Profile section component**

**const ProfileSection = () => (**

**<div className="flex items-center ml-4">**

**<Link href={userType === "admin" ? "/analytics-dashboard" : "/profile"}>**

**<Button variant="ghost" size="icon" className="text-gray-400 hover:text-gray-500">**

**{userType === "admin" ? (**

**<BarChart className="h-6 w-6" /> // Updated icon**

**) : (**

**<UserRound className="h-6 w-6" /> // Updated icon**

**)}**

**<span className="sr-only">**

**{userType === "admin" ? "Analytics Dashboard" : "Profile"}**

**</span>**

**</Button>**

**</Link>**

**<Button variant="ghost" size="sm" className="ml-2 text-gray-700 hover:bg-gray-100" onClick={handleSignOut}>**

**Sign Out**

**</Button>**

**</div>**

**);**

**if (loading) {**

**return <div>Loading...</div>; // Or show a loader**

**}**

**return (**

**<nav className="border-b border-gray-200">**

**<div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">**

**<div className="flex justify-between items-center h-16">**

**{/\* Left side: Search Icon \*/}**

**<div>**

**<Button variant="ghost" size="icon" className="text-gray-400 hover:text-gray-500" onClick={() => setModalOpen(true)}>**

**<Search className="h-6 w-6" />**

**<span className="sr-only">Search</span>**

**</Button>**

**</div>**

**{/\* Center: Logo \*/}**

**<div className="flex-shrink-0">**

**<Link href="/" className="flex items-center">**

**<span className="text-2xl" style={{ fontFamily: "var(--font-clash-reg)" }}>Avion</span>**

**</Link>**

**</div>**

**{/\* Right side: Cart, Wishlist, Profile/Sign-In \*/}**

**<div className="lg:hidden">**

**<Button variant="ghost" size="icon" className="text-gray-400 hover:text-gray-500" onClick={() => setMobileMenuOpen((prev) => !prev)}>**

**<Menu className="h-6 w-6" />**

**<span className="sr-only">Open menu</span>**

**</Button>**

**</div>**

**<div className="hidden lg:flex items-center">**

**{/\* Only show wishlist and cart for regular users \*/}**

**{(!user || userType !== "admin") && (**

**<>**

**<a href="/wishlist">**

**<Button variant="ghost" size="icon" className="ml-4 text-gray-400 hover:text-gray-500">**

**<Heart className="h-6 w-6" />**

**<span className="sr-only">Wishlist</span>**

**</Button>**

**</a>**

**<a href="/cart">**

**<Button variant="ghost" size="icon" className="ml-4 text-gray-400 hover:text-gray-500">**

**<Image src="/icons/Shopping--cart.svg" alt="Shopping cart" width={16} height={16} />**

**<span className="sr-only">Shopping cart</span>**

**</Button>**

**</a>**

**</>**

**)}**

**{/\* User Profile or Sign-In Button \*/}**

**{user ? (**

**<ProfileSection />**

**) : (**

**<Link href="/acc-creation">**

**<Button variant="ghost" size="sm" className="ml-4 text-gray-700 hover:bg-gray-100">Sign In</Button>**

**</Link>**

**)}**

**</div>**

**</div>**

**</div>**

**{isModalOpen && (**

**<div**

**className="fixed inset-0 bg-black bg-opacity-50 flex justify-center items-center z-50"**

**onClick={() => setModalOpen(false)}**

**>**

**<div**

**className="bg-white p-6 w-4/5 max-w-xl relative z-60"**

**onClick={(e) => e.stopPropagation()}**

**>**

**<button**

**className="absolute top-4 right-4 text-gray-500 hover:text-gray-700"**

**onClick={() => setModalOpen(false)}**

**>**

**&times;**

**</button>**

**<h2 className="text-xl font-semibold text-gray-700 mb-4 font-clash">**

**Looking for something?**

**</h2>**

**<p className="text-gray-600 mb-4 font-clash">**

**Input the name of the product and find out if we have it.**

**</p>**

**<div className="items-center space-y-2 font-clash">**

**<Input**

**type="search"**

**placeholder="Search..."**

**onChange={(e) => setSearchTerm(e.target.value)}**

**onKeyPress={handleKeyPress}**

**className="flex-grow border-gray-300"**

**/>**

**<Button**

**onClick={handleSearch}**

**style={{**

**backgroundColor: "#2A254B",**

**color: "white",**

**}}**

**className="px-4 py-2"**

**>**

**Search**

**</Button>**

**</div>**

**</div>**

**</div>**

**)}**

**</nav>**

**);**

**}**

**Cart   
"use client";**

**import { useEffect, useState } from "react";**

**import Image from "next/image";**

**import { Button } from "@/components/ui/button";**

**import { Input } from "@/components/ui/input";**

**import Link from "next/link";**

**import { auth } from "../../firebase/firebase"; // Import Firebase Auth instance**

**import { User } from "firebase/auth"; // Import Firebase User type**

**interface Category {**

**name: string;**

**slug: string;**

**}**

**interface Dimensions {**

**width: string;**

**height: string;**

**depth: string;**

**}**

**interface Products {**

**name: string;**

**description: string;**

**image: string;**

**\_id: string;**

**features: string[];**

**dimensions: Dimensions;**

**category: Category;**

**price: number;**

**tags: string[];**

**quantity: number;**

**}**

**export default function ShoppingCart() {**

**const [cartItems, setCartItems] = useState<Products[]>([]);**

**const [loading, setLoading] = useState<boolean>(true); // To show loading spinner**

**const [user, setUser] = useState<User | null>(null); // For storing the user state**

**// Check the authentication status once component is mounted**

**useEffect(() => {**

**const unsubscribe = auth.onAuthStateChanged((currentUser) => {**

**setUser(currentUser); // Set user state based on authentication status**

**setLoading(false); // Stop loading once the auth state is known**

**});**

**return () => unsubscribe(); // Cleanup listener**

**}, []);**

**// Fetch the cart items from localStorage when user is logged in**

**useEffect(() => {**

**if (user) {**

**const savedCart = localStorage.getItem(user.uid + "\_cart");**

**if (savedCart) {**

**setCartItems(JSON.parse(savedCart));**

**}**

**}**

**}, [user]); // Run this effect only when user is available**

**const updateQuantity = (id: string, newQuantity: number) => {**

**const updatedCart = cartItems.map((item) =>**

**item.\_id === id ? { ...item, quantity: newQuantity } : item**

**);**

**setCartItems(updatedCart);**

**if (user) {**

**localStorage.setItem(user.uid + "\_cart", JSON.stringify(updatedCart));**

**}**

**};**

**const removeItem = (id: string) => {**

**const updatedCart = cartItems.filter((item) => item.\_id !== id);**

**setCartItems(updatedCart);**

**if (user) {**

**localStorage.setItem(user.uid + "\_cart", JSON.stringify(updatedCart));**

**}**

**};**

**const calculateTotal = () => {**

**return cartItems.reduce(**

**(total, item) => total + item.price \* (item.quantity || 1),**

**0**

**);**

**};**

**if (loading) {**

**return (**

**<div className="flex justify-center items-center min-h-screen">**

**<div className="spinner-border animate-spin border-4 border-[#2A254B] border-t-transparent rounded-full w-12 h-12"></div>**

**</div>**

**);**

**}**

**if (!user) {**

**return (**

**<div className="mx-auto max-w-[1440px] px-4 py-16 sm:px-6 lg:px-8">**

**<h1 className="mb-12 text-3xl font-normal font-clash">Your shopping cart</h1>**

**<div className="text-center">**

**<p className="text-gray-500 mb-4 font-clash">**

**Please <strong>Sign In</strong> or <strong>Sign Up</strong> to add items to your cart.**

**</p>**

**<Link href="/acc-creation">**

**<Button className="rounded-none bg-[#2A254B] px-8 hover:bg-[#2A254B]/90 font-clash">**

**Sign In / Sign Up**

**</Button>**

**</Link>**

**</div>**

**</div>**

**);**

**}**

**return (**

**<div className="mx-auto max-w-[1440px] px-4 py-16 sm:px-6 lg:px-8">**

**<h1 className="mb-12 text-3xl font-normal font-clash">Your shopping cart</h1>**

**{cartItems.length === 0 ? (**

**<p className="text-center text-gray-500 font-clash">Your cart is empty</p>**

**) : (**

**<>**

**<div className="divide-y">**

**{cartItems.map((item) => (**

**<div key={item.\_id} className="grid grid-cols-1 gap-4 py-8 md:grid-cols-[2fr,1fr,1fr]">**

**<div className="flex gap-4">**

**<div className="relative h-24 w-24 flex-shrink-0 overflow-hidden bg-gray-100">**

**<Image**

**src={item.image}**

**alt={item.name}**

**fill**

**className="object-cover"**

**/>**

**</div>**

**<div>**

**<h2 className="font-clash text-lg">{item.name}</h2>**

**<p className="text-sm font-clash">Price: £{item.price}</p>**

**<Button onClick={() => removeItem(item.\_id)} className="text-red-500 font-clash bg-transparent hover:underline hover:bg-transparent">**

**Remove**

**</Button>**

**</div>**

**</div>**

**<div className="flex flex-col justify-center items-center">**

**<Input**

**type="number"**

**value={item.quantity || 1}**

**min="1"**

**onChange={(e) => updateQuantity(item.\_id, parseInt(e.target.value))}**

**className="w-20 text-center"**

**/>**

**</div>**

**<div className="flex flex-col justify-center items-center">**

**<p className="font-clash">£{item.price \* (item.quantity || 1)}</p>**

**</div>**

**</div>**

**))}**

**</div>**

**<div className="mt-8 space-y-4 border-t pt-8">**

**<div className="flex justify-end gap-4 text-lg font-clash">**

**<span>Subtotal</span>**

**<span>£{calculateTotal().toFixed(2)}</span>**

**</div>**

**<div className="flex justify-end">**

**<Link href={'/checkout'}>**

**<Button className="rounded-none bg-[#2A254B] px-8 hover:bg-[#2A254B]/90 font-clash">**

**Go to checkout**

**</Button>**

**</Link>**

**</div>**

**</div>**

**</>**

**)}**

**</div>**

**);**

**}**

**Reviews and Ratings   
'use client'**

**import { useState, useEffect } from "react"**

**import { Star } from "lucide-react"**

**import { Button } from "@/components/ui/button"**

**import { auth } from "../../firebase/firebase"**

**import { useRouter } from 'next/navigation'**

**interface Review {**

**rating: number**

**comment: string**

**date: string**

**userId: string**

**userName: string**

**userEmail: string**

**}**

**interface RatingsAndReviewsProps {**

**productId: string**

**}**

**export function RatingsAndReviews({ productId }: RatingsAndReviewsProps) {**

**const [rating, setRating] = useState(0)**

**const [comment, setComment] = useState("")**

**const [reviews, setReviews] = useState<Review[]>([])**

**const user = auth.currentUser**

**const router = useRouter() // Hook to navigate to another page**

**useEffect(() => {**

**const storedReviews = localStorage.getItem(`reviews\_${productId}`)**

**if (storedReviews) {**

**setReviews(JSON.parse(storedReviews))**

**}**

**}, [productId])**

**const handleRatingChange = (newRating: number) => {**

**setRating(newRating)**

**}**

**const handleSubmit = (e: React.FormEvent) => {**

**e.preventDefault()**

**if (!user) {**

**alert("You must be logged in to submit a review!")**

**return**

**}**

**const newReview: Review = {**

**rating,**

**comment,**

**date: new Date().toISOString(),**

**userId: user.uid,**

**userName: user.displayName || "Anonymous",**

**userEmail: user.email || "No email",**

**}**

**const updatedReviews = [...reviews, newReview]**

**setReviews(updatedReviews)**

**localStorage.setItem(`reviews\_${productId}`, JSON.stringify(updatedReviews))**

**setRating(0)**

**setComment("")**

**}**

**const handleDeleteReview = (index: number) => {**

**const reviewToDelete = reviews[index]**

**if (reviewToDelete.userId !== user?.uid) {**

**alert("You can only delete your own reviews!")**

**return**

**}**

**const updatedReviews = reviews.filter((\_, i) => i !== index)**

**setReviews(updatedReviews)**

**localStorage.setItem(`reviews\_${productId}`, JSON.stringify(updatedReviews))**

**}**

**const handleSignUpRedirect = () => {**

**router.push('/acc-creation') // Redirect to the sign-up page**

**}**

**return (**

**<div className="mt-8 mb-16">**

**{user ? (**

**<>**

**<h2 className="font-clash text-3xl font-medium mb-4">Ratings & Reviews</h2>**

**<div className="mb-8">**

**<form onSubmit={handleSubmit} className="space-y-4">**

**<div>**

**<label className="block mb-2 font-clash">Your Rating</label>**

**<div className="flex">**

**{[1, 2, 3, 4, 5].map((star) => (**

**<Star**

**key={star}**

**className={`w-6 h-6 cursor-pointer ${star <= rating ? "text-yellow-400 fill-current" : "text-gray-300"}`}**

**onClick={() => handleRatingChange(star)}**

**/>**

**))}**

**</div>**

**</div>**

**<div>**

**<label htmlFor="comment" className="block mb-2 font-clash">Your Review</label>**

**<textarea**

**id="comment"**

**value={comment}**

**onChange={(e) => setComment(e.target.value)}**

**className="w-full p-2 border rounded font-clash"**

**rows={4}**

**required**

**/>**

**</div>**

**<Button type="submit" className="bg-[#2A254B] hover:bg-[#2A254B]/90 text-white font-clash">**

**Submit Review**

**</Button>**

**</form>**

**</div>**

**</>**

**) : (**

**<div className="text-center">**

**<p className="font-clash text-lg text-gray-500 mb-4">You must be logged in to write a review.</p>**

**<Button onClick={handleSignUpRedirect} className="bg-[#2A254B] hover:bg-[#2A254B]/90 text-white font-clash">**

**Sign In / Sign Up Now**

**</Button>**

**</div>**

**)}**

**<div>**

**<h3 className="font-clash text-2xl font-medium mb-4">Customer Reviews</h3>**

**{reviews.length === 0 ? (**

**<p className="font-clash">No reviews yet. Be the first to review this product!</p>**

**) : (**

**reviews.map((review, index) => (**

**<div key={index} className="mb-4 pb-4 border-b last:border-b-0">**

**<div className="flex items-center mb-2">**

**{[1, 2, 3, 4, 5].map((star) => (**

**<Star**

**key={star}**

**className={`w-4 h-4 ${star <= review.rating ? "text-yellow-400 fill-current" : "text-gray-300"}`}**

**/>**

**))}**

**<span className="ml-2 text-sm text-gray-600 font-clash">**

**{new Date(review.date).toLocaleDateString()}**

**</span>**

**</div>**

**<p className="font-clash">{review.comment}</p>**

**<p className="font-clash text-sm text-gray-500">**

**By: {review.userName} ({review.userEmail})**

**</p>**

**{review.userId === user?.uid && (**

**<Button**

**onClick={() => handleDeleteReview(index)}**

**className="mt-2 text-red-600 hover:text-red-800 font-clash bg-transparent hover:bg-transparent"**

**>**

**Delete Review**

**</Button>**

**)}**

**</div>**

**))**

**)}**

**</div>**

**</div>**

**)**

**}**

### **1. API Integration for GET, POST, and DELETE Requests**

#### **GET Request: Fetching Products**

export async function getProducts() {

const res = await fetch('https://api.avion.com/products');

const products = await res.json();

return products;

}

export async function getProductById(id) {

const res = await fetch(`https://api.avion.com/products/${id}`);

const product = await res.json();

return product;

}

#### **POST Request: Adding a New Product**

To add a new product, I send a POST request to the backend API, which also triggers a webhook to update Sanity:

async function addProduct(productData) {

const response = await fetch('https://api.avion.com/products', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(productData),

});

const data = await response.json();

triggerSanityWebhook(data);

return data;

}

async function triggerSanityWebhook(product) {

await fetch('https://api.sanity.io/webhook/your-webhook-url', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

action: 'create',

product: product,

}),

});

}

#### **DELETE Request: Deleting a Product**

To delete a product, I use a DELETE request to the backend API, which also calls a webhook to reflect the changes in Sanity.

async function deleteProduct(id) {

const response = await fetch(`https://api.avion.com/products/${id}`, {

method: 'DELETE',

});

const result = await response.json();

triggerSanityWebhookDeletion(id);

return result;

}

async function triggerSanityWebhookDeletion(id) {

await fetch('https://api.sanity.io/webhook/your-webhook-url', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

action: 'delete',

productId: id,

}),

});

}

### **2. Sanity Webhook Setup**

I set up webhooks in Sanity to listen for changes. Here's how I do it:

#### **Steps to Set Up Webhooks in Sanity:**

1. **Go to the Sanity project settings.**
2. **Under API, navigate to "Webhooks".**
3. **Add a new webhook** with the endpoint of the backend or server that handles the API requests.
4. **Configure the webhook** to trigger when products are created, updated, or deleted.
5. **Test the webhook** to ensure that product data is correctly sent to the backend.

When a new product is created in Sanity via the webhook, it automatically triggers the API to add the product to the database or make any necessary adjustments.

### **3. Admin Dashboard Logic for Handling Product Operations**

In the admin dashboard, I provide interfaces for adding, updating, and deleting products. Here’s an example of how I set up a basic form and actions to handle these tasks:

import { useState } from 'react';

import { addProduct, deleteProduct } from './api'; // Import the API functions

function AdminDashboard() {

const [productName, setProductName] = useState('');

const [productPrice, setProductPrice] = useState('');

const handleAddProduct = async () => {

const productData = {

name: productName,

price: productPrice,

};

const response = await addProduct(productData);

if (response.success) {

alert('Product added successfully');

} else {

alert('Error adding product');

}

};

const handleDeleteProduct = async (productId) => {

const response = await deleteProduct(productId);

if (response.success) {

alert('Product deleted successfully');

} else {

alert('Error deleting product');

}

};

return (

<div>

<h1>Admin Dashboard</h1>

{/\* Form to Add Product \*/}

<input

type="text"

placeholder="Product Name"

value={productName}

onChange={(e) => setProductName(e.target.value)}

/>

<input

type="number"

placeholder="Product Price"

value={productPrice}

onChange={(e) => setProductPrice(e.target.value)}

/>

<button onClick={handleAddProduct}>Add Product</button>

{/\* List of Products with Delete Option \*/}

{/\* Imagine `products` is an array of product data \*/}

{products.map((product) => (

<div key={product.id}>

<h3>{product.name}</h3>

<p>{product.price}</p>

<button onClick={() => handleDeleteProduct(product.id)}>Delete</button>

</div>

))}

</div>

);

}

export default AdminDashboard;

### **4. Webhook Example in Sanity**

When a product is created or updated in Sanity, a webhook is triggered, and the backend receives this request. Here’s an example of what the payload might look like:

{

"action": "create",

"product": {

"id": "abc123",

"name": "New Product",

"price": 99.99,

"description": "A brand new product for sale."

}

}

When a product is deleted, the payload might look like this:

{

"action": "delete",

"productId": "abc123"

}  
  
  
**A technical report summarizing:**s**Building and Integrating Components in Avion E-Commerce**

#### **Steps Taken to Build and Integrate Components**

I developed Avion E-Commerce, a luxury homeware platform, with a structured approach to ensure seamless functionality and a premium user experience. Each component was carefully built and integrated to enhance usability and efficiency.

1. **Product Listing & Detail Pages:**
   * I displayed products in a grid format with filtering and sorting options.
   * Implemented lazy loading for images to improve page speed.
   * Clicking a product redirects to a detailed page with high-resolution images, descriptions, and pricing.
2. **Category & Search Bar:**
   * I categorized products (e.g., furniture, decor, kitchenware) for easy navigation.
   * Implemented a real-time search bar with autocomplete suggestions powered by API queries.
3. **Cart & Wishlist:**
   * Users can add products to the cart or wishlist, with data stored in localStorage for persistence.
   * Integrated a mini-cart dropdown for quick access.
4. **Checkout Flow & User Profile:**
   * I streamlined the checkout process with multiple payment options and address management.
   * The user profile allows order tracking, address updates, and saved payment methods.
5. **Filter Panel & Related Products:**
   * Implemented a filter panel with options for price range, color, material, and availability.
   * Displayed similar products on the product detail page to increase engagement.
6. **Reviews & Ratings & Customer Feedback:**
   * I enabled users to leave reviews and ratings with moderation from the admin panel.
   * Integrated a feedback section where users can share experiences.
7. **Pagination & Analytics Dashboard:**
   * Implemented pagination for product listings to optimize loading speed.
   * The admin dashboard includes an analytics section displaying sales data, top-performing products, and user behavior insights.
8. **Product Comparison & FAQ & Help Center:**
   * I allowed users to compare multiple products side by side based on key specifications.
   * Added a help center with FAQs for common queries related to shipping, returns, and product care.
9. **Admin Dashboard & Sanity Integration:**
   * Developed an intuitive admin panel for managing products, orders, and user queries.
   * Integrated Sanity CMS with a webhook system to reflect product updates instantly on both the website and admin panel.

#### **Challenges Faced and Solutions Implemented**

1. **Webhook Integration in Sanity**
   * **Challenge:** The biggest hurdle was identifying the correct webhook setup to sync product data between Sanity and the website. I needed to ensure that whenever a product was added, updated, or deleted in the admin dashboard, it would reflect in Sanity and vice versa.
   * **Solution:** I researched and configured Sanity’s webhooks correctly by defining the exact triggers (create, update, delete). Implemented API routes to listen for these webhook calls and update the website accordingly.
2. **Data Synchronization Between Sanity and the Website**
   * **Challenge:** Ensuring that any change in Sanity would instantly reflect on the website without delays.
   * **Solution:** I used real-time data fetching techniques and optimized caching strategies to minimize delays in reflecting changes.
3. **Optimizing Performance for Large Data Sets**
   * **Challenge:** Handling large product listings without impacting performance.
   * **Solution:** I implemented pagination, lazy loading, and optimized API calls to enhance performance.

#### **Best Practices Followed During Development**

1. **Component Reusability:**
   * I designed reusable components like product cards, buttons, and forms to maintain consistency and speed up development.
2. **Scalability and Maintainability:**
   * Used a modular approach, ensuring that future enhancements and updates can be integrated easily.
3. **Performance Optimization:**
   * I implemented lazy loading, caching, and optimized API requests to improve speed and efficiency.
4. **SEO & accessibility considerations:**
   * Ensured SEO-friendly URLs and meta tags.
   * Followed accessibility guidelines to enhance usability.
5. **Security Measures:**
   * Secured API endpoints and used authentication for admin access.
   * Implemented proper data validation to prevent malicious inputs.

By overcoming challenges and following best practices, I ensured that Avion E-Commerce provides a seamless experience for users who love timeless design and quality.