```
import React, { useState, useEffect,
useContext, createContext, useMemo }
from 'react';
import {
  Search, ShoppingCart, User, Menu, X, Star,
Heart,
  ChevronRight, ChevronDown, Plus, Minus,
Trash2,
  CheckCircle, Truck, ShieldCheck, Zap,
LogOut, LayoutDashboard, Package
} from 'lucide-react';

// --- MOCK DATA ---
const INITIAL_PRODUCTS = [
  {
    id: 1,
    title: "Apple iPhone 15 (Black, 128 GB)",
    price: 66999,
    originalPrice: 79900,
    rating: 4.6,
    reviews: 2300,
```

```
    category: "Mobiles",
    image: "https://images.unsplash.com/
photo-1696446701796-da61225697cc?
auto=format&fit=crop&q=80&w=600",
    description: "Experience the dynamic
island, 48MP Main Camera, and USB-C.
The future of smartphones is here.",
    brand: "Apple"
  },
  {
    id: 2,
    title: "Sony WH-1000XM5 Wireless
Headphones",
    price: 24990,
    originalPrice: 29990,
    rating: 4.8,
    reviews: 1200,
    category: "Electronics",
    image: "https://images.unsplash.com/
photo-1618366712010-f4ae9c647dcb?
auto=format&fit=crop&q=80&w=600",
```

```
    description: "Industry-leading noise
cancellation, crystal clear hands-free
calling, and up to 30-hour battery life.",
    brand: "Sony"
  },
  {
    id: 3,
    title: "Nike Air Jordan 1 Low",
    price: 8995,
    originalPrice: 12995,
    rating: 4.3,
    reviews: 850,
    category: "Fashion",
    image: "https://images.unsplash.com/
photo-1552346154-21d32810aba3?
auto=format&fit=crop&q=80&w=600",
    description: "Premium leather upper, air-
sole unit for cushioning, and the iconic
swoosh design.",
    brand: "Nike"
  },
```

```
  {
    id: 4,
    title: "Samsung 55\" 4K Smart LED TV",
    price: 45999,
    originalPrice: 65999,
    rating: 4.5,
    reviews: 3000,
    category: "Electronics",
    image: "https://images.unsplash.com/
photo-1593784991095-a205069470b6?
auto=format&fit=crop&q=80&w=600",
    description: "Crystal Processor 4K,
PurColor, and PC on TV features for a
seamless entertainment experience.",
    brand: "Samsung"
  },
  {
    id: 5,
    title: "Realme 11 Pro 5G (Sunrise Beige,
256 GB)",
    price: 23999,
```

```
    originalPrice: 27999,
    rating: 4.4,
    reviews: 1500,
    category: "Mobiles",
    image: "https://images.unsplash.com/
photo-1598327773202-3c1f043f47fe?
auto=format&fit=crop&q=80&w=600",
    description: "100MP OIS ProLight
Camera, Curved Vision Display, and
Dimensity 7050 5G Chipset.",
    brand: "Realme"
  },
  {
    id: 6,
    title: "Men's Slim Fit T-Shirt",
    price: 499,
    originalPrice: 999,
    rating: 4.0,
    reviews: 500,
    category: "Fashion",
    image: "https://images.unsplash.com/
```

```
photo-1521572163474-6864f9cf17ab?
auto=format&fit=crop&q=80&w=600",
    description: "100% Cotton, bio-washed,
comfortable fit for everyday wear.",
    brand: "Roadster"
  }
];

const CATEGORIES = [
  { name: "Top Offers", img: "https://cdn-
icons-png.flaticon.com/
128/6632/6632834.png" },
  { name: "Mobiles", img: "https://cdn-icons-
png.flaticon.com/128/644/644458.png" },
  { name: "Electronics", img: "https://cdn-
icons-png.flaticon.com/
128/3659/3659898.png" },
  { name: "Fashion", img: "https://cdn-icons-
png.flaticon.com/128/3050/3050239.png"
},
  { name: "Home", img: "https://cdn-icons-
```

```
png.flaticon.com/128/1946/1946488.png"
},
  { name: "Grocery", img: "https://cdn-icons-
png.flaticon.com/128/1261/1261163.png"
},
];

// --- CONTEXTS ---
const DataContext = createContext();

const DataProvider = ({ children }) => {
  const [user, setUser] = useState(null); // {
name, email, role: 'user' | 'admin' }
  const [products, setProducts] =
useState(INITIAL_PRODUCTS);
  const [cart, setCart] = useState([]);
  const [orders, setOrders] = useState([]);
  const [searchQuery, setSearchQuery] =
useState("");
  const [view, setView] = useState("home");
// home, product-list, product-detail, cart,
```

```
login, admin, checkout, success
  const [selectedProduct,
setSelectedProduct] = useState(null);
  const [categoryFilter, setCategoryFilter] =
useState(null);

  // Load cart from local storage on init
(simulated)
  useEffect(() => {
    const savedCart =
localStorage.getItem('fk_cart');
    if (savedCart)
setCart(JSON.parse(savedCart));
  }, []);

  // Save cart
  useEffect(() => {
    localStorage.setItem('fk_cart',
JSON.stringify(cart));
  }, [cart]);
```

```
  const login = (email, password) => {
    // Mock Auth
    if (email === "admin@flipkart.com" &&
password === "admin") {
      setUser({ name: "Flipkart Admin", email,
role: "admin" });
      return true;
    }
    if (email && password.length >= 6) {
      setUser({ name: "Flipkart User", email,
role: "user" });
      return true;
    }
    return false;
  };

  const logout = () => {
    setUser(null);
    setView("home");
  };
```

```
  const addToCart = (product) => {
    setCart(prev => {
      const existing = prev.find(p => p.id === product.id);
      if (existing) {
        return prev.map(p => p.id === product.id ? { ...p, qty: p.qty + 1 } : p);
      }
      return [...prev, { ...product, qty: 1 }];
    });
    alert("Product added to cart!");
  };

  const removeFromCart = (id) => {
    setCart(prev => prev.filter(p => p.id !== id));
  };

  const updateQty = (id, delta) => {
    setCart(prev => prev.map(p => {
      if (p.id === id) {
```

```
        const newQty = Math.max(1, p.qty +
delta);
        return { ...p, qty: newQty };
      }
      return p;
    }));
  };

  const placeOrder = (address) => {
    const newOrder = {
      id: "ORD" + Math.floor(Math.random() *
100000),
      items: cart,
      total: cart.reduce((acc, item) => acc +
(item.price * item.qty), 0),
      date: new Date().toLocaleDateString(),
      address,
      status: "Processing"
    };
    setOrders(prev => [newOrder, ...prev]);
    setCart([]);
```

```
    setView("success");
  };

  const addProduct = (product) => {
    const newProduct = { ...product, id:
products.length + 1 };
    setProducts([...products, newProduct]);
  };

  const deleteProduct = (id) => {
    setProducts(products.filter(p => p.id !==
id));
  };

  return (
    <DataContext.Provider value={{
      user, login, logout,
      products, setProducts, addProduct,
deleteProduct,
      cart, addToCart, removeFromCart,
updateQty,
```

```
      orders, placeOrder,
      view, setView,
      selectedProduct, setSelectedProduct,
      searchQuery, setSearchQuery,
      categoryFilter, setCategoryFilter
    }}>
      {children}
    </DataContext.Provider>
  );
};

// --- COMPONENTS ---

const Header = () => {
  const { cart, user, setView,
setSearchQuery, logout, searchQuery } =
useContext(DataContext);
  const [showUserMenu,
setShowUserMenu] = useState(false);

  return (
```

```
    <header className="sticky top-0 z-50 bg-[#2874f0] text-white shadow-md">
      <div className="container mx-auto px-4 py-3 flex items-center justify-between gap-4 max-w-7xl">
        {/* Logo */}
        <div className="cursor-pointer flex flex-col items-center" onClick={() => setView('home')}>
          <h1 className="text-xl font-bold italic tracking-wider">Flipkart</h1>
          <span className="text-[10px] italic flex items-center hover:underline text-gray-200">
            Explore <span className="text-[#ffe500] font-bold mx-1">Plus</span> <Star size={10} fill="#ffe500" stroke="none"/>
          </span>
        </div>
```

```
        {/* Search Bar */}
        <div className="flex-1 max-w-2xl
relative hidden md:block">
          <input
            type="text"
            placeholder="Search for products,
brands and more"
            className="w-full py-2 px-4
rounded-sm text-gray-800 focus:outline-
none shadow-sm"
            value={searchQuery}
            onChange={(e) =>
setSearchQuery(e.target.value)}
          />
          <Search className="absolute
right-3 top-2.5 text-[#2874f0]" size={20} />
        </div>

        {/* Action Buttons */}
        <div className="flex items-center
gap-6 font-medium">
```

```
          {user ? (
            <div className="relative group">
              <button
                className="bg-white text-
[#2874f0] px-6 py-1 rounded-sm font-
semibold hover:bg-gray-100 transition flex
items-center gap-1"
                onClick={() =>
setShowUserMenu(!showUserMenu)}
              >
                {user.name.split(' ')[0]}
<ChevronDown size={14}/>
              </button>
              {/* Dropdown */}
              <div className="absolute right-0
top-full mt-2 w-48 bg-white text-gray-800
shadow-xl rounded-sm overflow-hidden
hidden group-hover:block border border-
gray-100">
                {user.role === 'admin' && (
                  <div className="px-4 py-3
```

```
hover:bg-gray-50 cursor-pointer flex items-
center gap-2" onClick={() =>
setView('admin')}>
                  <LayoutDashboard size={16}/>
Admin Panel
                </div>
              )}
              <div className="px-4 py-3
hover:bg-gray-50 cursor-pointer flex items-
center gap-2" onClick={() =>
setView('orders')}>
                <Package size={16}/> My Orders
              </div>
              <div className="px-4 py-3
hover:bg-gray-50 cursor-pointer border-t
flex items-center gap-2" onClick={logout}>
                <LogOut size={16}/> Logout
              </div>
            </div>
          </div>
        ) : (
```

```
            <button
              className="bg-white text-
[#2874f0] px-8 py-1 rounded-sm font-
semibold hover:bg-gray-100 transition"
              onClick={() => setView('login')}
            >
              Login
            </button>
          )}

          <div className="cursor-pointer
hidden md:flex items-center gap-1">
            <span>More</span>
<ChevronDown size={14}/>
          </div>

          <div className="cursor-pointer flex
items-center gap-2" onClick={() =>
setView('cart')}>
            <div className="relative">
              <ShoppingCart size={20} />
```

```
            {cart.length > 0 && (
              <span className="absolute
-top-2 -right-2 bg-[#ff6161] text-white text-
[10px] w-4 h-4 rounded-full flex items-
center justify-center font-bold border
border-[#2874f0]">
                {cart.length}
              </span>
            )}
          </div>
          <span className="hidden
md:block">Cart</span>
        </div>
      </div>
    </div>

    {/* Mobile Search */}
    <div className="md:hidden px-4 pb-3">
      <div className="relative">
        <input
         type="text"
```

```
            placeholder="Search for products..."
            className="w-full py-2 px-4
rounded-sm text-gray-800 focus:outline-
none shadow-sm"
            value={searchQuery}
            onChange={(e) =>
setSearchQuery(e.target.value)}
          />
          <Search className="absolute
right-3 top-2.5 text-[#2874f0]" size={20} />
        </div>
      </div>
    </header>
  );
};

const Categories = () => {
  const { setCategoryFilter, setView } =
useContext(DataContext);

  return (
```

```
    <div className="bg-white shadow-sm
border-b overflow-x-auto">
      <div className="container mx-auto
px-4 py-3 flex justify-between min-w-
[600px] max-w-7xl">
        {CATEGORIES.map((cat, index) => (
          <div
            key={index}
            className="flex flex-col items-
center gap-1 cursor-pointer hover:text-
[#2874f0] transition group"
            onClick={() => {
              setCategoryFilter(cat.name ===
"Top Offers" ? null : cat.name);
              setView('product-list');
            }}
          >
            <div className="w-16 h-16
overflow-hidden flex items-center justify-
center mb-1">
              <img src={cat.img} alt={cat.name}
```

```
className="h-full object-contain group-
hover:scale-110 transition duration-300"/>
            </div>
            <span className="text-sm font-
medium text-gray-700 group-hover:text-
[#2874f0]">{cat.name}</span>
          </div>
        ))}
      </div>
    </div>
  );
};

const BannerSlider = () => (
  <div className="bg-gray-100 p-2">
    <div className="container mx-auto
max-w-7xl">
      <div className="w-full h-48 md:h-72
bg-gradient-to-r from-indigo-500 via-
purple-500 to-pink-500 rounded shadow-
md flex items-center justify-center text-
```

```
white relative overflow-hidden">
        <div className="absolute inset-0 bg-
black/10"></div>
        <div className="z-10 text-center">
          <h2 className="text-3xl md:text-5xl
font-bold mb-2">Big Billion Days</h2>
          <p className="text-lg md:text-xl
mb-4">Sale is Live! Up to 80% Off</p>
          <button className="bg-white text-
indigo-600 px-6 py-2 rounded font-bold
hover:bg-gray-100 shadow-lg">Shop Now</
button>
        </div>
      </div>
    </div>
  </div>
);

const ProductCard = ({ product }) => {
  const { setSelectedProduct, setView } =
useContext(DataContext);
```

```
  return (
    <div
      className="bg-white p-4 hover:shadow-xl transition duration-300 cursor-pointer flex flex-col border border-transparent hover:border-gray-200 rounded-sm relative group"
      onClick={() => {
        setSelectedProduct(product);
        setView('product-detail');
      }}
    >
      <div className="absolute top-2 right-2 text-gray-300 hover:text-red-500 z-10">
        <Heart size={20} fill="currentColor" className="opacity-0 group-hover:opacity-100 transition"/>
      </div>
      <div className="h-48 flex items-center justify-center mb-4 p-2">
```

```
        <img src={product.image}
alt={product.title} className="max-h-full
max-w-full object-contain hover:scale-105
transition duration-500" />
      </div>
      <div className="flex-1 flex flex-col">
        <h3 className="text-gray-800 font-
medium truncate mb-1 hover:text-
[#2874f0]">{product.title}</h3>
        <div className="flex items-center
gap-2 mb-2">
          <span className="bg-[#388e3c]
text-white text-xs px-1.5 py-0.5 rounded
flex items-center gap-0.5">
            {product.rating} <Star size={10}
fill="white" />
          </span>
          <span className="text-gray-500
text-sm">({product.reviews})</span>
        </div>
        <div className="flex items-center
```

```
gap-3 mt-auto">
          <span className="text-lg font-bold
text-gray-900">₹
{product.price.toLocaleString()}</span>
          <span className="text-gray-500
text-sm line-through">₹
{product.originalPrice.toLocaleString()}</
span>
          <span className="text-[#388e3c]
text-sm font-medium">
            {Math.round(((product.originalPrice
- product.price) / product.originalPrice) *
100)}% off
          </span>
        </div>
      </div>
    </div>
  );
};

// --- PAGES ---
```

```
const HomePage = () => {
  const { products, setView } =
useContext(DataContext);
  const featuredProducts =
products.slice(0, 4);

  return (
    <div className="bg-[#f1f3f6] min-h-
screen pb-10">
      <Categories />
      <BannerSlider />

      {/* Deals of the day */}
      <div className="container mx-auto
max-w-7xl mt-4 px-2">
        <div className="bg-white p-4
shadow-sm">
          <div className="flex justify-between
items-center mb-4 border-b pb-3">
            <div>
```

```
            <h2 className="text-xl font-medium">Best of Electronics</h2>
            <p className="text-gray-500 text-sm">Best Devices for You</p>
          </div>
          <button
            className="bg-[#2874f0] text-white px-4 py-2 rounded-sm text-sm font-medium shadow-md hover:bg-blue-700 transition"
            onClick={() => setView('product-list')}
          >
            VIEW ALL
          </button>
        </div>
        <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-4 gap-4">
          {products.filter(p => p.category === 'Electronics').slice(0,4).map(product => (
            <ProductCard key={product.id}
```

```
product={product} />
          ))}
        </div>
      </div>
    </div>

    {/* Suggested for You */}
    <div className="container mx-auto
max-w-7xl mt-4 px-2">
      <div className="bg-white p-4
shadow-sm">
        <h2 className="text-xl font-medium
mb-4">Suggested for You</h2>
        <div className="grid grid-cols-1
sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-6
gap-4">

{products.slice().reverse().map(product =>
(
            <ProductCard key={product.id}
product={product} />
```

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

const ProductListPage = () => {
  const { products, searchQuery,
categoryFilter, setCategoryFilter } =
useContext(DataContext);
  const [sortOrder, setSortOrder] =
useState('relevant'); // low-high, high-low
  const [priceRange, setPriceRange] =
useState(100000);

  const filteredProducts = useMemo(() => {
    let result = products;

    if (categoryFilter) {
```

```
    result = result.filter(p => p.category ===
categoryFilter);
  }

  if (searchQuery) {
    const q = searchQuery.toLowerCase();
    result = result.filter(p =>
      p.title.toLowerCase().includes(q) ||
      p.category.toLowerCase().includes(q)
||
      p.brand.toLowerCase().includes(q)
    );
  }

  result = result.filter(p => p.price <=
priceRange);

  if (sortOrder === 'low-high') {
    result = result.sort((a, b) => a.price -
b.price);
  } else if (sortOrder === 'high-low') {
```

```
      result = result.sort((a, b) => b.price -
a.price);
    }

    return result;
  }, [products, searchQuery, categoryFilter,
priceRange, sortOrder]);

  return (
    <div className="bg-[#f1f3f6] min-h-
screen py-4">
      <div className="container mx-auto
max-w-7xl px-2 flex flex-col md:flex-row
gap-4">
        {/* Sidebar Filters */}
        <div className="w-full md:w-1/5 bg-
white shadow-sm p-4 h-fit">
          <h3 className="text-lg font-medium
mb-4 border-b pb-2">Filters</h3>

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

```
            <h4 className="text-sm font-
medium mb-2 uppercase text-
gray-500">Price</h4>
            <input
             type="range"
             min="0"
             max="100000"
             value={priceRange}
             onChange={(e) =>
setPriceRange(Number(e.target.value))}
             className="w-full accent-
[#2874f0]"
            />
            <div className="flex justify-
between text-sm mt-1">
              <span>Min</span>
              <span>₹{priceRange}</span>
            </div>
          </div>

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

```
          <h4 className="text-sm font-medium mb-2 uppercase text-gray-500">Categories</h4>
          <div className="flex flex-col gap-2">
            {['Mobiles', 'Electronics', 'Fashion', 'Home', 'Grocery'].map(cat => (
              <label key={cat} className="flex items-center gap-2 text-sm cursor-pointer">
                <input
                 type="checkbox"
                 checked={categoryFilter === cat}
                 onChange={() => setCategoryFilter(categoryFilter === cat ? null : cat)}
                />
                {cat}
              </label>
            ))}
          </div>
```

```
          </div>
        </div>

        {/* Product Grid */}
        <div className="flex-1 bg-white
shadow-sm p-4">
          <div className="flex flex-col
md:flex-row justify-between items-center
mb-4 border-b pb-2">
            <h2 className="font-medium text-
lg">
              {categoryFilter || "All Products"}
              <span className="text-gray-500
text-sm ml-2">(Showing
{filteredProducts.length} items)</span>
            </h2>
            <div className="flex items-center
gap-4 text-sm">
              <span className="font-
medium">Sort By</span>
              <span className={`cursor-pointer
```

```
${sortOrder === 'relevant' ? 'text-[#2874f0]
font-bold border-b-2 border-[#2874f0]' : ''}`}
onClick={() => setSortOrder('relevant')}
>Relevance</span>
            <span className={`cursor-pointer
${sortOrder === 'low-high' ? 'text-[#2874f0]
font-bold border-b-2 border-[#2874f0]' : ''}`}
onClick={() => setSortOrder('low-high')}
>Price -- Low to High</span>
            <span className={`cursor-pointer
${sortOrder === 'high-low' ? 'text-[#2874f0]
font-bold border-b-2 border-[#2874f0]' : ''}`}
onClick={() => setSortOrder('high-low')}
>Price -- High to Low</span>
          </div>
        </div>

        {filteredProducts.length > 0 ? (
```