**Hướng dẫn làm bài Progress Test 2**

1. **Tạo dự án React + Vite**

* Tạo mới 1 thư mục tên fer202. Vào Visual studio code, vào File, chọn Open Folder, duyệt đến thư mục fer202 vừa tạo.
* Vào Terminal, chọn New Terminal, gõ lệnh sau để tạo dự án React

Npx create-react-app……

* npx create-react-app progress-test2

cd my-pd

npm install

npm start

Lệnh này dùng để khởi động server development.

Mặc định chạy ở http://localhost:3000/

1. **Cài đặt các thư viện cần dùng**

Nhấn Ctrl+C để dùng server, tiếp tục gõ lệnh sau để cài đặt các gói thư viện cần dùng

npm i react-router-dom axios bootstrap react-bootstrap react-icons prop-types

* Tiếp đến, import file bootstrap vào index.js:

import 'bootstrap/dist/css/bootstrap.min.css';

1. **Chuẩn bị API giả lập (Json Server)**

* Đặt file db.json đã cung cấp ở thư mục gốc của dự án (cùng thư mục với package.json)
* Khởi tạo server bằng cách. Tại Visual Studio code, Terminal chọn Split Terminal

Gõ lệnh sau: npx json-server --watch db.json --port 3001

Chạy JSON Server trên cổng 3001:

<http://localhost:3001/phones>

<http://localhost:3001/accounts>

Tạo file .env ở gốc dự án (không phải thư mục src)

Click vào project progress-test2 (thư mục gốc), tạo mới chứa nội dung:

REACT\_APP\_API\_URL=http://localhost:3001

1. **Tạo cấu trúc thư mục**

Click phải vào thư mục src, New Folder

|  |  |
| --- | --- |
|  | * Tạo thư mục components: chứa các file .jsx là các components của 1 trang: Navbar.jsx, ProductCard.jsx, Footer.jsx… * Tạo thư mục Context: chứa các context của ứng dụng: AuthContext.jsx, CartContext.jsx, ThemeContext.jsx, ToastContext.jsx,… * Tạo thư mục hooks: chứa các hooks cần thiết * Tạo thư mục pages: chứa các trang của ứng dụng: ProductsPage.jsx, ProductDetails.jsx, LoginPage.jsx,…. * Tạo thư mục rootes: chứa AppRoutes.jsx chứa các cài đặt để điều hướng đến các trang * Tạo thư mục services: chứa file dịch vụ cung cấp cho các trang ví dụ: api.js * Tạo thư mục utils: chứa các phương thức tiện tích dùng cho chương trình: ví dụ format.js, helper.js,… |

Sau khi tạo xong cấu trúc thư mục yêu cầu

1. **Cấu hình API client**

File src/services/api.js:

import axios from 'axios';

const api = axios.create({

  baseURL: import.meta.env.VITE\_API\_URL || 'http://localhost:3001'});

  export default api;

1. **Khung Router + App shell**

File src/routes/AppRoutes.js:

import { Route, Routes } from 'react-router-dom';

import ProduductsPage from '../pages/ProductsPage';

export default function AppRoutes() {

  return <Routes>

    <Route path="/products" element={<ProduductsPage />} />

  </Routes>;

}

Cấu hình src/App.js

import './App.css'

import AppRoutes from './routes/AppRoutes.js'

function App() {

  return (

    <>

      <div className='container py-4'>

        <AppRoutes />

      </div>

    </>

  )

}

export default App

**Cấu hình file src/index.js:**

import React, { StrictMode } from 'react'

import { createRoot } from 'react-dom/client'

import React from 'react';

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import App from './App';

import 'bootstrap/dist/css/bootstrap.min.css';

import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <BrowserRouter>

    <App />

  </BrowserRouter>

);

1. **Tạo hook filter/sort cho products**

File src/hooks/useProductFilters.js

import { useMemo } from 'react';

export const useProductFilters = (products, searchQuery, sortBy, filterBy) => {

  return useMemo(() => {

    let filtered = products;

    // search

    const q = (searchQuery || '').trim().toLowerCase();

    if (q) {

      filtered = filtered.filter(p =>

        (p.name || p.title || '').toLowerCase().includes(q) ||

        (p.description || '').toLowerCase().includes(q)

      );

    }

    // filter category

    if (filterBy && filterBy !== 'all') {

      filtered = filtered.filter(p => (p.category || '').toLowerCase() === filterBy.toLowerCase());

    }

    // sort

    const sorted = [...filtered];

    switch (sortBy) {

      case 'name-asc': sorted.sort((a, b) => (a.name || a.title || '').localeCompare(b.name || b.title || '')); break;

      case 'name-desc': sorted.sort((a, b) => (b.name || b.title || '').localeCompare(a.name || a.title || '')); break;

      case 'price-asc': sorted.sort((a, b) => parseFloat(a.price) - parseFloat(b.price)); break;

      case 'price-desc': sorted.sort((a, b) => parseFloat(b.price) - parseFloat(a.price)); break;

    }

    return sorted;

  }, [products, searchQuery, sortBy, filterBy]);

};

1. **Viết các phương thức format:**

File src/utils/format.js

export function formatPrice(v) {

  const n = typeof v === 'string' ? parseFloat(v) : v;

  return Number.isNaN(n) ? '$0.00' : `$${n.toFixed(2)}`;

}

export function assetUrl(p) {

  if (!p) return '';

  if (/^https?:\/\//i.test(p)) return p;     // nếu là URL tuyệt đối

  return '/' + p.replace(/^\/+/, '');        // ép về /images/products/....

}

1. **Xây dựng component ProductCard**

Viết code cho component ProductCard:

**src/components/ProductCard.js**

import React from 'react';

import { useNavigate } from 'react-router-dom';

import { Card, Button, Badge, ButtonGroup } from 'react-bootstrap';

import { FaEye, FaCartPlus, FaHeart } from 'react-icons/fa';

import { formatPrice } from '../utils/format';

const ProductCard = ({ product }) => {

  const navigate = useNavigate();

  /\* code for cart, details and favourites event handling

 code for toast notifications \*/

  return (

    <Card className="h-100 shadow-sm product-card">

      <Card.Img

        variant="top"

        src={product.image}

        alt={product.name}

        style={{ height: '200px', objectFit: 'cover' }}

      />

      <Card.Body className="d-flex flex-column">

        <Card.Title className="h6 mb-2">{product.name}</Card.Title>

        <Card.Text className="flex-grow-1 small text-muted mb-2">

          {product.description}

        </Card.Text>

        <div className="mb-3">

          <Badge bg="primary" className="fs-6">

            {formatPrice(product.price)}

          </Badge>

        </div>

        <ButtonGroup className="w-100">

          <Button

            variant="outline-primary"

            size="sm"

            className="flex-fill"

          >

            <FaEye className="me-1" />

            View Details

          </Button>

          <Button

            variant="success"

            size="sm"

            className="flex-fill"

          >

            <FaCartPlus className="me-1" />

            Add to Cart

          </Button>

       {/\*   code for checking if product is in favourites

            \*/}

            <Button

              variant="outline-danger"

              size="sm"

              className="flex-fill"

            >

              <FaHeart className="me-1" />

              Favourite

            </Button>

        </ButtonGroup>

      </Card.Body>

    </Card>

  );

};

export default ProductCard;

Viết code cho trang ProductsPage

**src/components/ProductsPage.js**

import React, { useEffect, useState } from 'react';

import { Container, Row, Col, Form, InputGroup, Button, Card } from 'react-bootstrap';

import { FaSearch, FaTimesCircle, FaFilter, FaSort } from 'react-icons/fa';

import ProductCard from '../components/ProductCard';

import { useProductFilters } from '../hooks/useProductFilters';

import api from '../services/api';

export default function ProductsPage() {

  const [searchQuery, setSearchQuery] = useState('');

  const [sortBy, setSortBy] = useState('newest');

  const [filterBy, setFilterBy] = useState('all');

  const [products, setProducts] = useState([]);

  useEffect(() => {

    (async () => {

      const { data } = await api.get('/products');

      const normalized = (data || []).map(p => ({

        id: p.id,

        name: p.title || p.name,

        image: p.image || `https://picsum.photos/seed/${p.id}/600/400`,

        price: p.price,

        description: p.description,

        category: p.category

      }));

      setProducts(normalized);

    })();

  }, []);

  const filteredProducts = useProductFilters(products, searchQuery, sortBy, filterBy);

  return (

    <Container className="py-4">

      <h1 className="mb-4">Products</h1>

      <Card className="mb-4">

        <Card.Body>

          <Row className="g-3">

            <Col md={6}>

              <InputGroup>

                <InputGroup.Text><FaSearch /></InputGroup.Text>

                <Form.Control

                  placeholder="Search products..."

                  value={searchQuery}

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

                />

                {searchQuery && (

                  <Button variant="outline-secondary" onClick={() => setSearchQuery('')}>

                    <FaTimesCircle />

                  </Button>

                )}

              </InputGroup>

            </Col>

            <Col md={3}>

              <InputGroup>

                <InputGroup.Text><FaFilter /></InputGroup.Text>

                <Form.Select value={filterBy} onChange={(e) => setFilterBy(e.target.value)}>

                  <option value="all">All Categories</option>

                  <option value="iphone">iphone</option>

                   <option value="samsung">Sam sung</option>

                    <option value="google">Google</option>

                    <option value="oppo">oppo</option>

                </Form.Select>

              </InputGroup>

            </Col>

            <Col md={3}>

              <InputGroup>

                <InputGroup.Text><FaSort /></InputGroup.Text>

                <Form.Select value={sortBy} onChange={(e) => setSortBy(e.target.value)}>

                  <option value="name-asc">Name A→Z</option>

                  <option value="name-desc">Name Z→A</option>

                  <option value="price-asc">Price ↑</option>

                  <option value="price-desc">Price ↓</option>

                </Form.Select>

              </InputGroup>

            </Col>

          </Row>

        </Card.Body>

      </Card>

      <Row className="g-4">

        {filteredProducts.map(p => (

          <Col key={p.id} xs={12} sm={6} md={4}>

            <ProductCard product={p} />

          </Col>

        ))}

      </Row>

      {filteredProducts.length === 0 && (

        <Row className="mt-4"><Col><Card className="text-center"><Card.Body>No mobile found.</Card.Body></Card></Col></Row>

      )}

    </Container>

  );

}

1. **File Css tham khảo**

Copy tất cả nội dung bên dưới vào file App.css

.App {

  min-height: 100vh;

  background-color: var(--bs-body-bg);

}

.cursor-pointer {

  cursor: pointer;

}

.product-card {

  transition: transform 0.2s ease-in-out;

}

.product-card:hover {

  transform: translateY(-5px);

}

/\* Custom scrollbar \*/

::-webkit-scrollbar {

  width: 8px;

}

::-webkit-scrollbar-track {

  background: #f1f1f1;

}

::-webkit-scrollbar-thumb {

  background: #888;

  border-radius: 4px;

}

::-webkit-scrollbar-thumb:hover {

  background: #555;

}

/\* Toast positioning \*/

.toast-container {

  z-index: 1055;

}

/\* Navbar brand hover effect \*/

.navbar-brand:hover {

  opacity: 0.8;

}

/\* Button group spacing \*/

.btn-group .btn {

  margin-right: 2px;

}

.btn-group .btn:last-child {

  margin-right: 0;

}

/\* Ensure navbar dropdown overlays sticky content \*/

.navbar .dropdown-menu.show {

  z-index: 2000;

}

.dishes {

  display: flex;

  flex-wrap: wrap;

  gap: 16px;

}

.dish-item {

  border: 1px solid #ccc;

  padding: 16px;

  width: 200px;

  text-align: center;

}

.dish-item img {

  width: 100%;

  height: auto;

}

.dish-item button {

  margin-top: 10px;

  padding: 8px 16px;

  background-color: #4caf50;

  color: white;

  border: none;

  cursor: pointer;

}

.dish-item button:hover {

  background-color: #45a049;

}

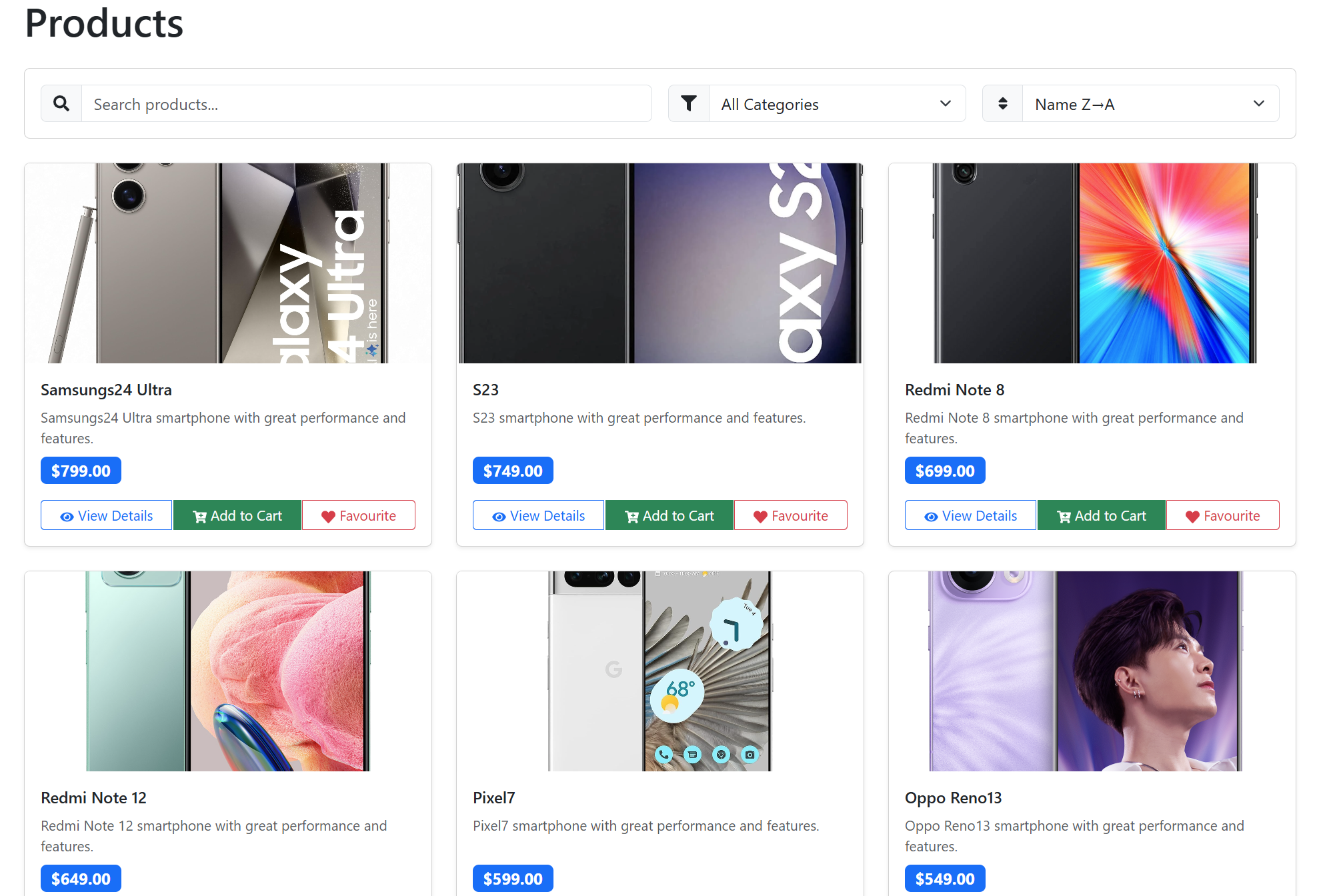
**Thực hiện chạy chương trình:**

Mở 2 cửa sổ Terminal:

Terminal 1: npm run dev

Terminal 2: npx json-server --watch db.json --port 3001

Kết quả chạy chương trình:



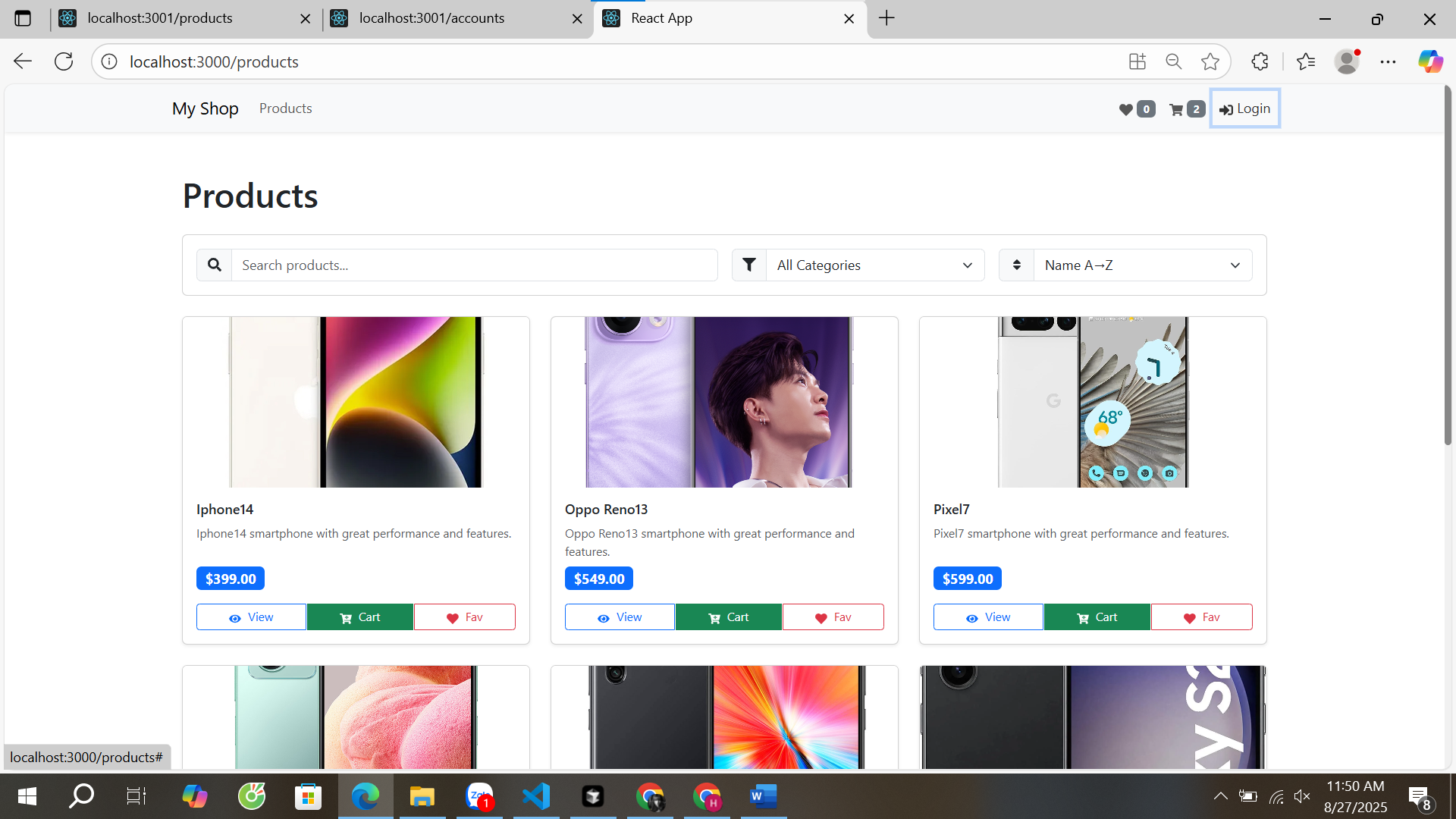
Sau khi thực hiện các bước làm trên: hiển thị, tìm kiếm, lọc và sắp xếp, đã đạt yêu cầu, 50% số điểm.

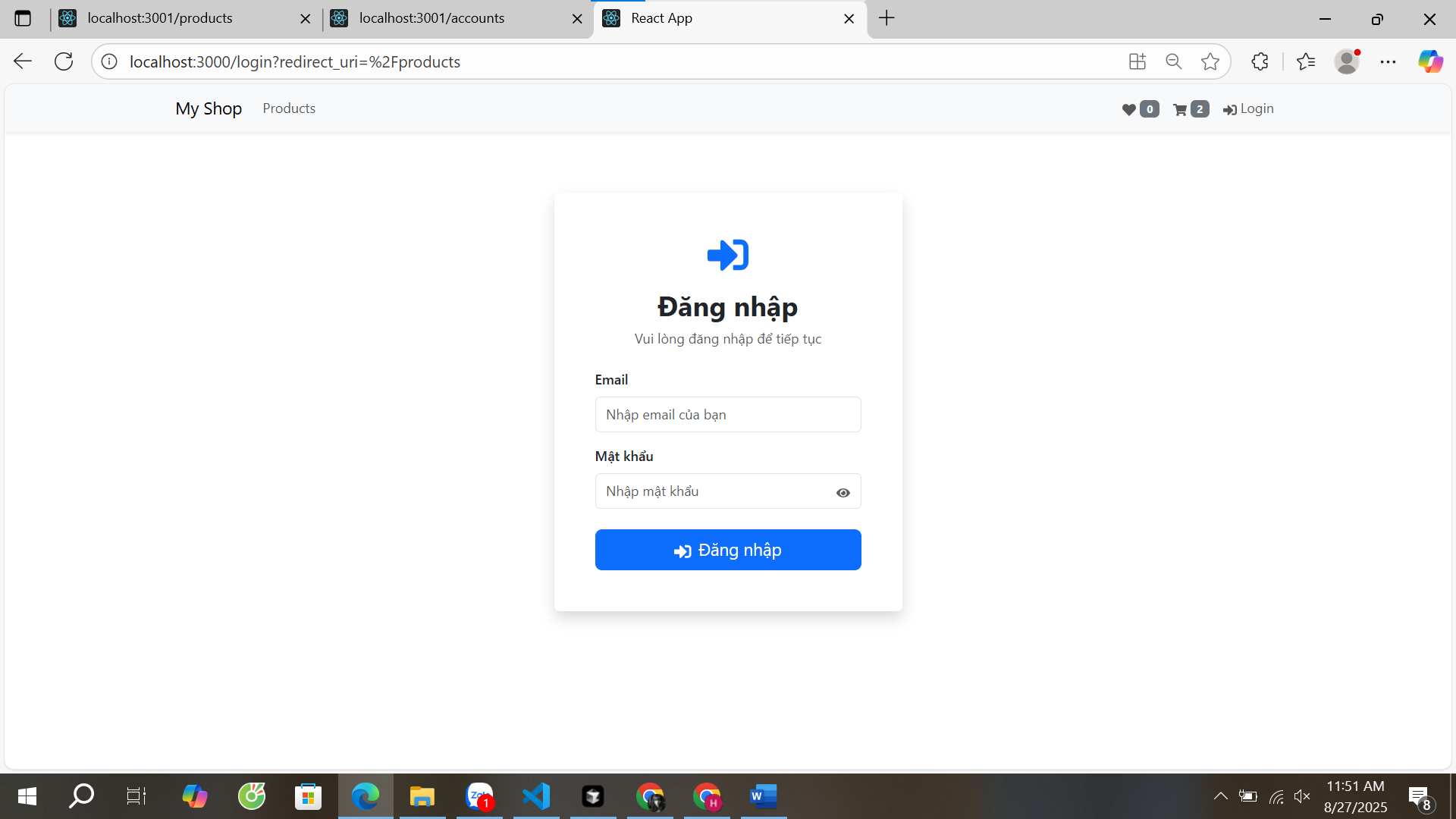
Hãy thực hiện tiếp các yêu cầu sau:

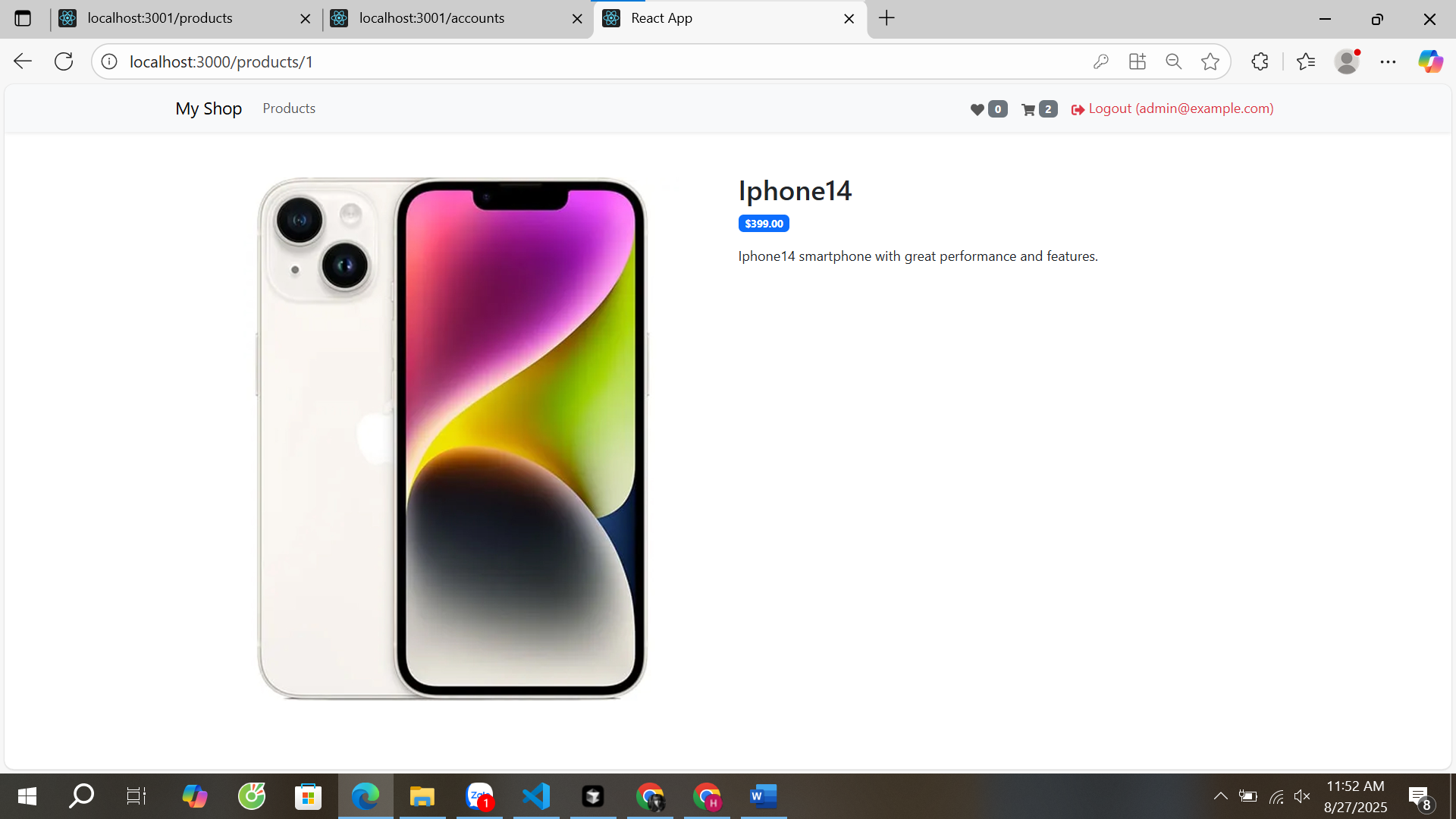
1. Xử lý NavBar

* Viết code cho NavBar component, có chứa 3 icon: Favourites, Cart, Login
* Tiến hành nhúng NavBar này vào trang ProductsPage.

1. Xử lý View Details
2. Xử lý click vào Login, cho phép hiển thị form và check valid thông tin đăng nhập từ file db.json.







Navbar.js

import { Navbar, Container, Nav, Badge } from 'react-bootstrap';

import { Link, useNavigate, useLocation } from 'react-router-dom';

import { FaHeart, FaShoppingCart, FaSignInAlt, FaSignOutAlt } from 'react-icons/fa';

import { useFavorites } from '../contexts/FavoriteContext';

import { useAuth } from '../contexts/AuthContext';

import { useToast } from '../contexts/ToastContext';

import { useCart } from '../contexts/CartContext';

export default function **NavBar**() {

  const **navigate** = useNavigate();

  const loc = useLocation();

  const { getFavoriteCount } = useFavorites();

  const { user, logout } = useAuth();

  const { showSuccess, showInfo } = useToast();

  const { getCartCount } = useCart();

  const **handleLogout** = () => {

    logout();

    showSuccess('Đăng xuất thành công!');

  };

  const **handleShowFavorites** = () => {

    showInfo(`Bạn có ${getFavoriteCount()} sản phẩm yêu thích`);

  };

  const **handleShowCart** = () => {

    showInfo(`Giỏ hàng có ${getCartCount()} sản phẩm`);

  };

  return (

    <Navbar *bg*="light" *expand*="lg" *className*="shadow-sm">

      <Container>

        <Navbar.Brand *as*={Link} *to*="/products">My Shop</Navbar.Brand>

        <Navbar.Toggle />

        <Navbar.Collapse>

          <Nav *className*="me-auto">

            <Nav.Link *as*={Link} *to*="/products">Products</Nav.Link>

          </Nav>

          <Nav>

            <Nav.Link *onClick*={handleShowFavorites}>

              <FaHeart />{' '}

              <Badge *bg*="secondary">{getFavoriteCount()}</Badge>

            </Nav.Link>

            <Nav.Link *onClick*={handleShowCart}>

              <FaShoppingCart />{' '}

              <Badge *bg*="secondary">{getCartCount()}</Badge>

            </Nav.Link>

            {user ? (

              <Nav.Link *onClick*={handleLogout} *className*="text-danger">

                <FaSignOutAlt /> Logout ({user.email})

              </Nav.Link>

            ) : (

              <Nav.Link *onClick*={() => navigate(`/login?redirect\_uri=${encodeURIComponent(loc.pathname + loc.search)}`)}>

                <FaSignInAlt /> Login

              </Nav.Link>

            )}

          </Nav>

        </Navbar.Collapse>

      </Container>

    </Navbar>

  );

}

ProductCart.js

import { Card, Button, Badge, ButtonGroup } from 'react-bootstrap';

import { FaEye, FaCartPlus, FaHeart, FaHeartBroken } from 'react-icons/fa';

import { useNavigate } from 'react-router-dom';

import { formatPrice } from '../utils/format';

import { useFavorites } from '../contexts/FavoriteContext';

import { useAuth } from '../contexts/AuthContext';

import { useToast } from '../contexts/ToastContext';

import { useCart } from '../contexts/CartContext';

export default function ProductCard({ product }) {

  const navigate = useNavigate();

  const { toggleFavorite, isFavorite } = useFavorites();

  const { isAuthenticated } = useAuth();

  const { showSuccess, showWarning, showError } = useToast();

  const { addToCart: addToCartContext } = useCart();

  const handleAddToCart = () => {

    if (!isAuthenticated()) {

      showWarning('Vui lòng đăng nhập để thêm sản phẩm vào giỏ hàng!');

      return;

    }

    addToCartContext(product.id, 1);

    showSuccess(`Đã thêm ${product.name} vào giỏ hàng!`);

  };

  const handleToggleFavorite = () => {

    if (!isAuthenticated()) {

      showWarning('Vui lòng đăng nhập để thêm sản phẩm vào yêu thích!');

      return;

    }

    const wasFavorite = isFavorite(product.id);

    toggleFavorite(product.id);

    if (wasFavorite) {

      showSuccess(`Đã xóa ${product.name} khỏi yêu thích!`);

    } else {

      showSuccess(`Đã thêm ${product.name} vào yêu thích!`);

    }

  };

  return (

    <Card className="h-100 shadow-sm">

      <Card.Img

        variant="top"

        src={product.image}

        alt={product.name}

        style={{ height: 200, objectFit: 'cover' }}

      />

      <Card.Body className="d-flex flex-column">

        <Card.Title className="h6 mb-2">{product.name}</Card.Title>

        <Card.Text className="flex-grow-1 small text-muted mb-2">

          {product.description}

        </Card.Text>

        <div className="mb-3">

          <Badge bg="primary" className="fs-6">{formatPrice(product.price)}</Badge>

        </div>

        <ButtonGroup className="w-100">

          <Button variant="outline-primary" size="sm" className="flex-fill"

            onClick={() => navigate(`/products/${product.id}`)}>

            <FaEye className="me-1" /> View

          </Button>

          <Button variant="success" size="sm" className="flex-fill" onClick={handleAddToCart}>

            <FaCartPlus className="me-1" /> Cart

          </Button>

          <Button

            variant={isFavorite(product.id) ? "danger" : "outline-danger"}

            size="sm"

            className="flex-fill"

            onClick={handleToggleFavorite}

          >

            {isFavorite(product.id) ? (

              <>

                <FaHeartBroken className="me-1" /> Remove Fav

              </>

            ) : (

              <>

                <FaHeart className="me-1" /> Fav

              </>

            )}

          </Button>

        </ButtonGroup>

      </Card.Body>

    </Card>

  );

}

AuthContext.js

import { createContext, useContext, useState, useEffect } from 'react';

const AuthContext = createContext();

export function AuthProvider({ children }) {

  const [user, setUser] = useState(null);

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

  // Load user from localStorage on mount

  useEffect(() => {

    const savedUser = JSON.parse(localStorage.getItem('user') || 'null');

    setUser(savedUser);

    setIsLoading(false);

  }, []);

  // Save to localStorage whenever user changes

  useEffect(() => {

    if (user) {

      localStorage.setItem('user', JSON.stringify(user));

    } else {

      localStorage.removeItem('user');

    }

  }, [user]);

  const login = (userData) => {

    setUser(userData);

  };

  const logout = () => {

    setUser(null);

  };

  const isAuthenticated = () => {

    return user !== null;

  };

  const value = {

    user,

    isLoading,

    login,

    logout,

    isAuthenticated

  };

  return (

    <AuthContext.Provider value={value}>

      {children}

    </AuthContext.Provider>

  );

}

export function useAuth() {

  const context = useContext(AuthContext);

  if (!context) {

    throw new Error('useAuth must be used within an AuthProvider');

  }

  return context;

}

CartContext.js

import { createContext, useContext, useState, useEffect } from 'react';

const CartContext = createContext();

export function CartProvider({ children }) {

  const [cart, setCart] = useState([]);

  // Load cart from localStorage on mount

  useEffect(() => {

    const savedCart = JSON.parse(localStorage.getItem('cart') || '[]');

    setCart(savedCart);

  }, []);

  // Save to localStorage whenever cart changes

  useEffect(() => {

    localStorage.setItem('cart', JSON.stringify(cart));

  }, [cart]);

  const addToCart = (productId, quantity = 1) => {

    setCart(prev => {

      const existingItem = prev.find(item => item.id === productId);

      if (existingItem) {

        return prev.map(item =>

          item.id === productId

            ? { ...item, qty: item.qty + quantity }

            : item

        );

      } else {

        return [...prev, { id: productId, qty: quantity }];

      }

    });

  };

  const removeFromCart = (productId) => {

    setCart(prev => prev.filter(item => item.id !== productId));

  };

  const updateQuantity = (productId, quantity) => {

    if (quantity <= 0) {

      removeFromCart(productId);

    } else {

      setCart(prev => prev.map(item =>

        item.id === productId ? { ...item, qty: quantity } : item

      ));

    }

  };

  const clearCart = () => {

    setCart([]);

  };

  const getCartCount = () => {

    return cart.reduce((total, item) => total + item.qty, 0);

  };

  const getCartTotal = (products) => {

    return cart.reduce((total, item) => {

      const product = products.find(p => p.id === item.id);

      return total + (product ? product.price \* item.qty : 0);

    }, 0);

  };

  const value = {

    cart,

    addToCart,

    removeFromCart,

    updateQuantity,

    clearCart,

    getCartCount,

    getCartTotal

  };

  return (

    <CartContext.Provider value={value}>

      {children}

    </CartContext.Provider>

  );

}

export function useCart() {

  const context = useContext(CartContext);

  if (!context) {

    throw new Error('useCart must be used within a CartProvider');

  }

  return context;

}

FavoriteContext.js

import { createContext, useContext, useState, useEffect } from 'react';

const FavoriteContext = createContext();

export function FavoriteProvider({ children }) {

  const [favorites, setFavorites] = useState([]);

  // Load favorites from localStorage on mount

  useEffect(() => {

    const savedFavorites = JSON.parse(localStorage.getItem('wishlist') || '[]');

    setFavorites(savedFavorites);

  }, []);

  // Save to localStorage whenever favorites change

  useEffect(() => {

    localStorage.setItem('wishlist', JSON.stringify(favorites));

  }, [favorites]);

  const addToFavorites = (productId) => {

    if (!favorites.includes(productId)) {

      setFavorites(prev => [...prev, productId]);

    }

  };

  const removeFromFavorites = (productId) => {

    setFavorites(prev => prev.filter(id => id !== productId));

  };

  const toggleFavorite = (productId) => {

    if (favorites.includes(productId)) {

      removeFromFavorites(productId);

    } else {

      addToFavorites(productId);

    }

  };

  const isFavorite = (productId) => {

    return favorites.includes(productId);

  };

  const getFavoriteCount = () => {

    return favorites.length;

  };

  const value = {

    favorites,

    addToFavorites,

    removeFromFavorites,

    toggleFavorite,

    isFavorite,

    getFavoriteCount

  };

  return (

    <FavoriteContext.Provider value={value}>

      {children}

    </FavoriteContext.Provider>

  );

}

export function useFavorites() {

  const context = useContext(FavoriteContext);

  if (!context) {

    throw new Error('useFavorites must be used within a FavoriteProvider');

  }

  return context;

}

ToastContext.js

import { createContext, useContext, useState } from 'react';

import { Toast, ToastContainer } from 'react-bootstrap';

const ToastContext = createContext();

export function ToastProvider({ children }) {

  const [toasts, setToasts] = useState([]);

  const addToast = (message, type = 'success', duration = 3000) => {

    const id = Date.now();

    const newToast = {

      id,

      message,

      type,

      duration

    };

    setToasts(prev => [...prev, newToast]);

    // Auto remove toast after duration

    setTimeout(() => {

      removeToast(id);

    }, duration);

  };

  const removeToast = (id) => {

    setToasts(prev => prev.filter(toast => toast.id !== id));

  };

  const showSuccess = (message) => addToast(message, 'success');

  const showError = (message) => addToast(message, 'danger');

  const showWarning = (message) => addToast(message, 'warning');

  const showInfo = (message) => addToast(message, 'info');

  const value = {

    addToast,

    removeToast,

    showSuccess,

    showError,

    showWarning,

    showInfo

  };

  return (

    <ToastContext.Provider value={value}>

      {children}

      {/\* Toast Container \*/}

      <ToastContainer

        position="top-end"

        className="p-3"

        style={{ zIndex: 9999 }}

      >

        {toasts.map(toast => (

          <Toast

            key={toast.id}

            onClose={() => removeToast(toast.id)}

            bg={toast.type}

            delay={toast.duration}

            autohide

          >

            <Toast.Header closeButton>

              <strong className="me-auto">

                {toast.type === 'success' && 'Thành công!'}

                {toast.type === 'danger' && 'Lỗi!'}

                {toast.type === 'warning' && 'Cảnh báo!'}

                {toast.type === 'info' && 'Thông tin!'}

              </strong>

            </Toast.Header>

            <Toast.Body className={toast.type === 'success' ? 'text-white' : ''}>

              {toast.message}

            </Toast.Body>

          </Toast>

        ))}

      </ToastContainer>

    </ToastContext.Provider>

  );

}

export function useToast() {

  const context = useContext(ToastContext);

  if (!context) {

    throw new Error('useToast must be used within a ToastProvider');

  }

  return context;

}

useProductFilters.js

import { useMemo } from 'react';

export const useProductFilters = (products, searchQuery, sortBy, filterBy) => {

  return useMemo(() => {

    let filtered = products;

    const q = (searchQuery || '').trim().toLowerCase();

    if (q) {

      filtered = filtered.filter(p =>

        (p.name || p.title || '').toLowerCase().includes(q) ||

        (p.description || '').toLowerCase().includes(q)

      );

    }

    if (filterBy && filterBy !== 'all') {

      filtered = filtered.filter(

        p => (p.category || '').toLowerCase() === filterBy.toLowerCase()

      );

    }

    const sorted = [...filtered];

    switch (sortBy) {

      case 'name-asc':  sorted.sort((a,b)=>(a.name||a.title||'').localeCompare(b.name||b.title||'')); break;

      case 'name-desc': sorted.sort((a,b)=>(b.name||b.title||'').localeCompare(a.name||a.title||'')); break;

      case 'price-asc': sorted.sort((a,b)=>parseFloat(a.price)-parseFloat(b.price)); break;

      case 'price-desc':sorted.sort((a,b)=>parseFloat(b.price)-parseFloat(a.price)); break;

      default: break;

    }

    return sorted;

  }, [products, searchQuery, sortBy, filterBy]);

};

LoginPage.js

import { useEffect, useState } from 'react';

import { useLocation, useNavigate } from 'react-router-dom';

import { Card, Form, Button, Container, Alert, Row, Col } from 'react-bootstrap';

import { FaSignInAlt, FaEye, FaEyeSlash } from 'react-icons/fa';

import api from '../services/api';

import { useAuth } from '../contexts/AuthContext';

import { useToast } from '../contexts/ToastContext';

export default function LoginPage() {

  const nav = useNavigate();

  const loc = useLocation();

  const { login } = useAuth();

  const { showSuccess, showError } = useToast();

  const params = new URLSearchParams(loc.search);

  const redirect = params.get('redirect\_uri') || '/products';

  const [email, setEmail] = useState('');

  const [password, setPassword] = useState('');

  const [error, setError] = useState('');

  const [isLoading, setIsLoading] = useState(false);

  const [showPassword, setShowPassword] = useState(false);

  const submit = async (e) => {

    e.preventDefault();

    setIsLoading(true);

    setError('');

    try {

      // Validate form

      if (!email.trim() || !password.trim()) {

        setError('Vui lòng nhập đầy đủ email và mật khẩu');

        return;

      }

      // Check email format

      const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

      if (!emailRegex.test(email.trim())) {

        setError('Email không đúng định dạng');

        return;

      }

      // Call API to check login

      const { data } = await api.get(`/accounts?email=${encodeURIComponent(email.trim())}&password=${encodeURIComponent(password)}`);

      if (Array.isArray(data) && data.length > 0) {

        const userData = data[0];

        if (userData.isActive) {

          login(userData);

          showSuccess('Đăng nhập thành công! Chào mừng bạn quay trở lại.');

          nav(redirect, { replace: true });

        } else {

          setError('Tài khoản đã bị khóa. Vui lòng liên hệ admin.');

        }

      } else {

        setError('Email hoặc mật khẩu không chính xác');

      }

    } catch (error) {

      console.error('Login error:', error);

      setError('Có lỗi xảy ra khi đăng nhập. Vui lòng thử lại.');

    } finally {

      setIsLoading(false);

    }

  };

  useEffect(() => {

    // nếu đã login thì quay về

    const u = JSON.parse(localStorage.getItem('user') || 'null');

    if (u) nav(redirect, { replace: true });

  }, [nav, redirect]);

  return (

    <Container className="py-5">

      <Row className="justify-content-center">

        <Col md={6} lg={5} xl={4}>

          <Card className="shadow border-0">

            <Card.Body className="p-5">

              <div className="text-center mb-4">

                <FaSignInAlt size={48} className="text-primary mb-3" />

                <h2 className="fw-bold">Đăng nhập</h2>

                <p className="text-muted">Vui lòng đăng nhập để tiếp tục</p>

              </div>

              <Form onSubmit={submit}>

                <Form.Group className="mb-3">

                  <Form.Label className="fw-semibold">Email</Form.Label>

                  <Form.Control

                    type="email"

                    placeholder="Nhập email của bạn"

                    value={email}

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

                    required

                    className="py-2"

                  />

                </Form.Group>

                <Form.Group className="mb-4">

                  <Form.Label className="fw-semibold">Mật khẩu</Form.Label>

                  <div className="position-relative">

                    <Form.Control

                      type={showPassword ? "text" : "password"}

                      placeholder="Nhập mật khẩu"

                      value={password}

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

                      required

                      className="py-2 pe-5"

                    />

                    <Button

                      type="button"

                      variant="link"

                      className="position-absolute end-0 top-0 h-100 text-muted"

                      onClick={() => setShowPassword(!showPassword)}

                      style={{ zIndex: 10 }}

                    >

                      {showPassword ? <FaEyeSlash /> : <FaEye />}

                    </Button>

                  </div>

                </Form.Group>

                {error && (

                  <Alert variant="danger" className="mb-3">

                    {error}

                  </Alert>

                )}

                <div className="d-grid">

                  <Button

                    type="submit"

                    variant="primary"

                    size="lg"

                    disabled={isLoading}

                    className="py-2"

                  >

                    {isLoading ? 'Đang đăng nhập...' : (

                      <>

                        <FaSignInAlt className="me-2" />

                        Đăng nhập

                      </>

                    )}

                  </Button>

                </div>

              </Form>

            </Card.Body>

          </Card>

        </Col>

      </Row>

    </Container>

  );

}

 ProductDetails.js

import { useEffect, useState } from 'react';

import { useParams } from 'react-router-dom';

import api from '../services/api';

import { Container, Row, Col, Image, Badge } from 'react-bootstrap';

import { formatPrice, assetUrl } from '../utils/format';

export default function ProductDetails() {

  const { id } = useParams();

  const [p, setP] = useState(null);

  useEffect(() => {

    (async () => {

      const { data } = await api.get(`/products/${id}`);

      setP({

  id: data.id,

  name: data.title || data.name,

  image: assetUrl(data.image) || `https://picsum.photos/seed/${data.id}/800/600`,

  price: data.price,

  description: data.description

});

    })();

  }, [id]);

  if (!p) return <Container className="py-4">Loading...</Container>;

  return (

    <Container className="py-4">

      <Row className="g-4">

        <Col md={6}>

<Image

  src={p.image}

  alt={p.name}

  fluid

  onError={(e) => {

    e.currentTarget.onerror = null;

    e.currentTarget.src = `https://picsum.photos/seed/${p.id}/800/600`;

  }}

/>        </Col>

        <Col md={6}>

          <h2>{p.name}</h2>

          <Badge bg="primary" className="mb-3">{formatPrice(p.price)}</Badge>

          <p>{p.description}</p>

        </Col>

      </Row>

    </Container>

  );

}

ProductsPage.js

import { useEffect, useState } from 'react';

import { Container, Row, Col, Form, InputGroup, Button, Card } from 'react-bootstrap';

import { FaSearch, FaTimesCircle, FaFilter, FaSort } from 'react-icons/fa';

import ProductCard from '../components/ProductCard';

import { useProductFilters } from '../hooks/useProductFilters';

import api from '../services/api';

import { assetUrl } from '../utils/format';

export default function ProductsPage() {

  const [searchQuery, setSearchQuery] = useState('');

  const [sortBy, setSortBy] = useState('name-asc');

  const [filterBy, setFilterBy] = useState('all');

  const [products, setProducts] = useState([]);

  useEffect(() => {

    (async () => {

      const { data } = await api.get('/products'); // qua proxy hoặc ENV

      const normalized = (data || []).map(p => ({

  id: p.id,

  name: p.title || p.name,

  image: assetUrl(p.image) || `https://picsum.photos/seed/${p.id}/600/400`,

  price: p.price,

  description: p.description,

  category: p.category

}));

      setProducts(normalized);

    })();

  }, []);

  const filteredProducts = useProductFilters(products, searchQuery, sortBy, filterBy);

  return (

    <Container className="py-4">

      <h1 className="mb-4">Products</h1>

      <Card className="mb-4">

        <Card.Body>

          <Row className="g-3">

            <Col md={6}>

              <InputGroup>

                <InputGroup.Text><FaSearch /></InputGroup.Text>

                <Form.Control

                  placeholder="Search products..."

                  value={searchQuery}

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

                />

                {searchQuery && (

                  <Button variant="outline-secondary" onClick={() => setSearchQuery('')}>

                    <FaTimesCircle />

                  </Button>

                )}

              </InputGroup>

            </Col>

            <Col md={3}>

              <InputGroup>

                <InputGroup.Text><FaFilter /></InputGroup.Text>

                <Form.Select value={filterBy} onChange={(e) => setFilterBy(e.target.value)}>

                  <option value="all">All Categories</option>

                  <option value="iphone">iphone</option>

                  <option value="samsung">samsung</option>

                </Form.Select>

              </InputGroup>

            </Col>

            <Col md={3}>

              <InputGroup>

                <InputGroup.Text><FaSort /></InputGroup.Text>

                <Form.Select value={sortBy} onChange={(e) => setSortBy(e.target.value)}>

                  <option value="name-asc">Name A→Z</option>

                  <option value="name-desc">Name Z→A</option>

                  <option value="price-asc">Price ↑</option>

                  <option value="price-desc">Price ↓</option>

                </Form.Select>

              </InputGroup>

            </Col>

          </Row>

        </Card.Body>

      </Card>

      <Row className="g-4">

        {filteredProducts.map(p => (

          <Col key={p.id} xs={12} sm={6} md={4}>

            <ProductCard product={p} />

          </Col>

        ))}

      </Row>

      {filteredProducts.length === 0 && (

        <Row className="mt-4">

          <Col>

            <Card className="text-center"><Card.Body>No results</Card.Body></Card>

          </Col>

        </Row>

      )}

    </Container>

  );

}

AppRoutes.js

import { Routes, Route, Navigate } from 'react-router-dom';

import ProductsPage from '../pages/ProductsPage';

import ProductDetails from '../pages/ProductDetails';

import LoginPage from '../pages/LoginPage';

export default function AppRoutes() {

  return (

    <Routes>

      <Route path="/" element={<Navigate to="/products" replace />} />

      <Route path="/products" element={<ProductsPage />} />

      <Route path="/products/:id" element={<ProductDetails />} />

      <Route path="/login" element={<LoginPage />} />

    </Routes>

  );

}

api.js

import axios from 'axios';

const api = axios.create({ baseURL: '' }); // gọi '/products' sẽ được proxy sang 3001

export default api;

format.js

export function formatPrice(v) {

  const n = typeof v === 'string' ? parseFloat(v) : v;

  return Number.isNaN(n) ? '$0.00' : `$${n.toFixed(2)}`;

}

export function assetUrl(p) {

  if (!p) return '';

  if (/^https?:\/\//i.test(p)) return p;          // URL tuyệt đối

  return '/' + String(p).replace(/^\/+/, '');

}