**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 + Vite

npm create vite@latest progress-test2 --template react

* Chọn React, JavaScript

cd my-pd

npm install

npm run dev

Lệnh này dùng Vite để khởi động server development.

Mặc định chạy ở http://localhost:5173/

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 main.jsx:

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:

VITE\_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.jsx:

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.jsx

import './App.css'

import AppRoutes from './routes/AppRoutes.jsx'

function App() {

  return (

    <>

      <div className='container py-4'>

        <AppRoutes />

      </div>

    </>

  )

}

export default App

**Cấu hình file src/main.jsx:**

import React, { StrictMode } from 'react'

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

import './index.css'

import App from './App.jsx'

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

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

//Sau này sẽ bọc Provider context vào đây

createRoot(document.getElementById('root')).render(

  <StrictMode>

    <BrowserRouter>

    <App />

    </BrowserRouter>

  </StrictMode>,

)

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.jsx**

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.jsx**

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;

}

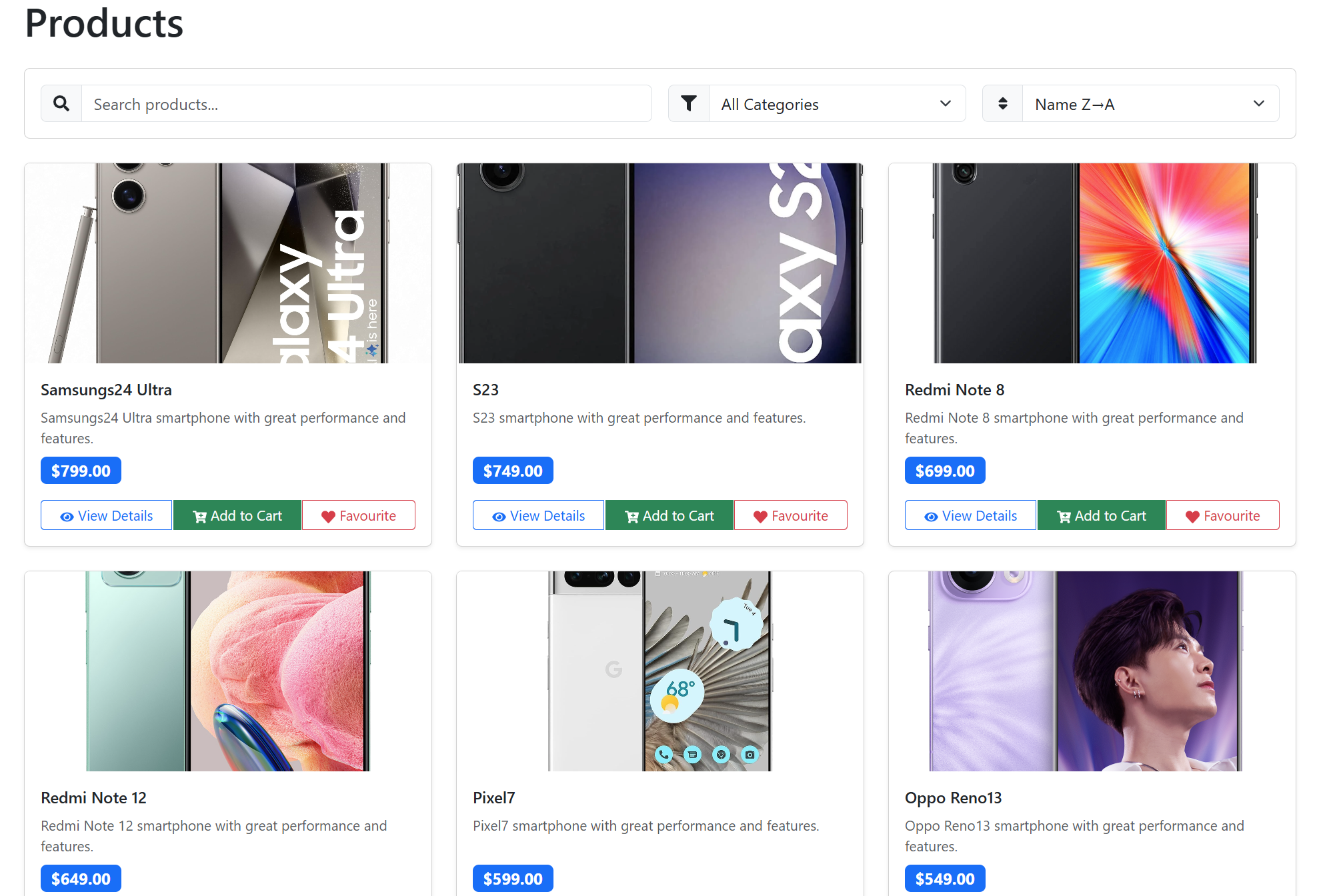
**Thực hiện chạy chương trình:**

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.