# Slot 17 - Using useContext

## Header.js





import React, { useContext } from "react";

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

import { CartContext } from "./CartContext";

import { Navbar, Container, Badge, Button } from "react-bootstrap";

export default function Header({ dark, onToggleTheme }) {

  const { totalCount } = useContext(CartContext);

  const navigate = useNavigate();

  const count = totalCount;

  return (

    <Navbar bg={dark ? "dark" : "light"} variant={dark ? "dark" : "light"} className="py-2">

      <Container fluid className="d-flex justify-content-between align-items-center">

        <Navbar.Brand as={Link} to="/" className="fs-3 fw-bold m-0">

          Food Management

        </Navbar.Brand>

        <div className="d-flex align-items-center gap-2">

          {/\* Nút chuyển đổi chế độ sáng/tối \*/}

          <Button

            variant={dark ? "outline-light" : "outline-dark"}

            onClick={onToggleTheme}

            size="sm"

          >

            {dark ? "☀️ Light" : "🌙 Dark"}

          </Button>

          {/\* Nút giỏ hàng \*/}

          <Button

            variant={dark ? "outline-warning" : "outline-primary"}

            onClick={() => navigate("/cart")}

            className="position-relative"

          >

            🛒

            {count > 0 && (

              <Badge

                bg="danger"

                className="position-absolute top-0 start-100 translate-middle"

              >

                {count}

              </Badge>

            )}

          </Button>

        </div>

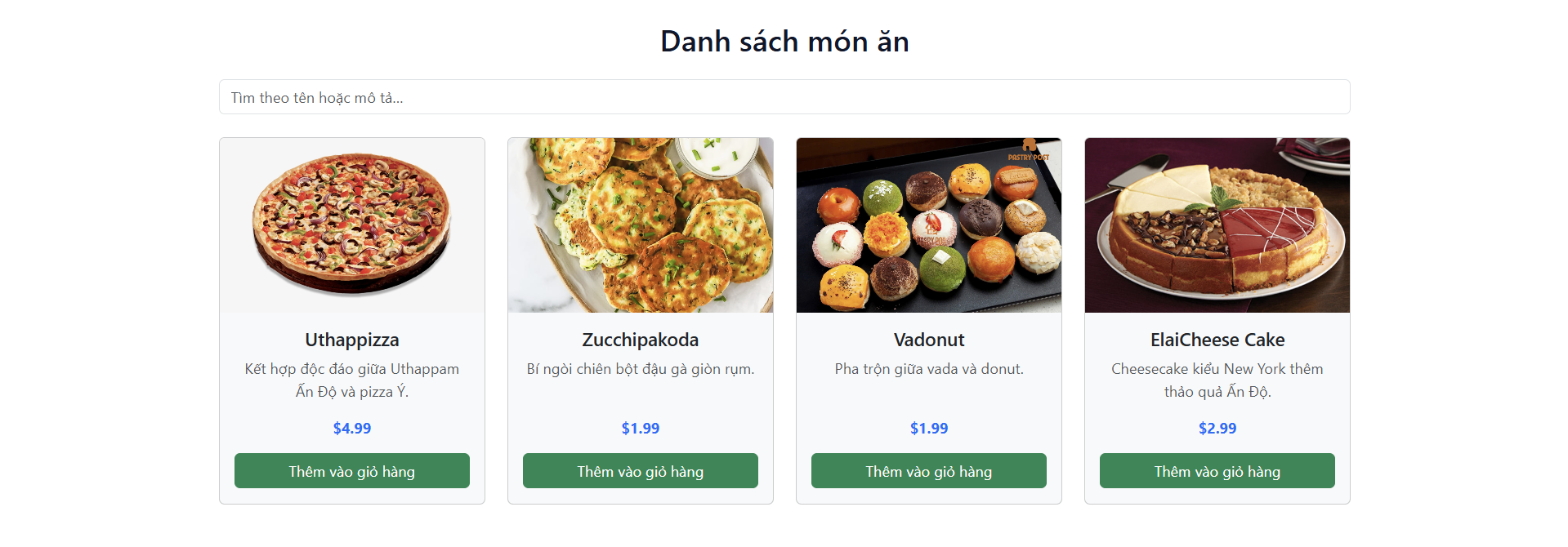
      </Container>

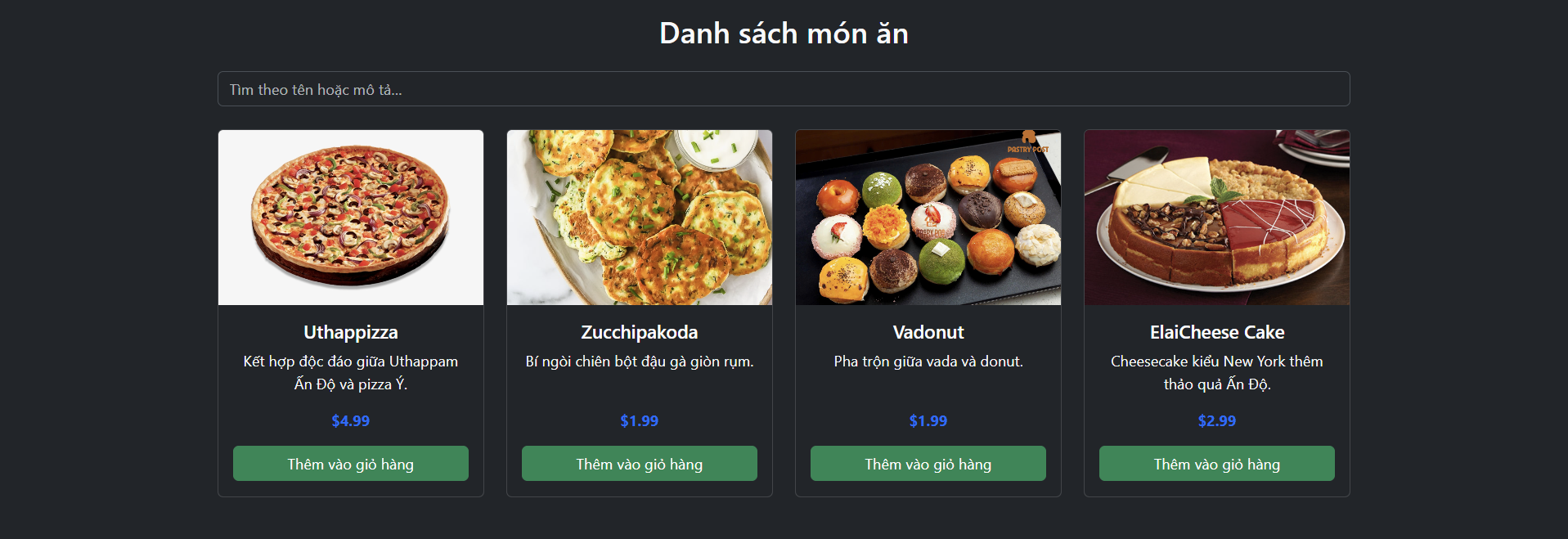
    </Navbar>

  );

}

## DishesList.js





import React, { useContext, useMemo, useState } from "react";

import { CartContext } from "./CartContext";

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

const DishesList = ({ dishes, dark }) => {

  const { addToCart } = useContext(CartContext);

  const [query, setQuery] = useState("");

  const filtered = useMemo(() => {

    const q = query.trim().toLowerCase();

    if (!q) return dishes;

    return dishes.filter(

      (d) =>

        d.name.toLowerCase().includes(q) ||

        d.description.toLowerCase().includes(q)

    );

  }, [query, dishes]);

  return (

    <Container className="my-4">

      <h2 className="mb-4 text-center">Danh sách món ăn</h2>

      <Form.Group className="mb-4">

        <Form.Control

          type="text"

          size="md"

          placeholder="Tìm theo tên hoặc mô tả…"

          value={query}

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

        />

      </Form.Group>

      {filtered.length === 0 ? (

        <Alert variant="info" className="text-center">

          Không tìm thấy món phù hợp.

        </Alert>

      ) : (

        <Row>

          {filtered.map((dish) => (

            <Col key={dish.id} md={6} lg={3} className="mb-4">

              {/\* Thêm bg và text props để hỗ trợ dark mode \*/}

              <Card

                className="h-100 dish-card text-center"

                bg={dark ? "dark" : "light"}

                text={dark ? "white" : "dark"}

              >

                <Card.Img

                  variant="top"

                  src={dish.image}

                  alt={dish.name}

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

                />

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

                  <Card.Title>{dish.name}</Card.Title>

                  <Card.Text className={dark ? "text-light" : "text-muted"} style={{ flexGrow: 1 }}>

                    {dish.description}

                  </Card.Text>

                  <Card.Text className="fw-bold text-primary">

                    ${parseFloat(dish.price).toFixed(2)}

                  </Card.Text>

                  <Button

                    variant="success"

                    onClick={() => addToCart(dish)}

                    className="mt-auto"

                  >

                    Thêm vào giỏ hàng

                  </Button>

                </Card.Body>

              </Card>

            </Col>

          ))}

        </Row>

      )}

    </Container>

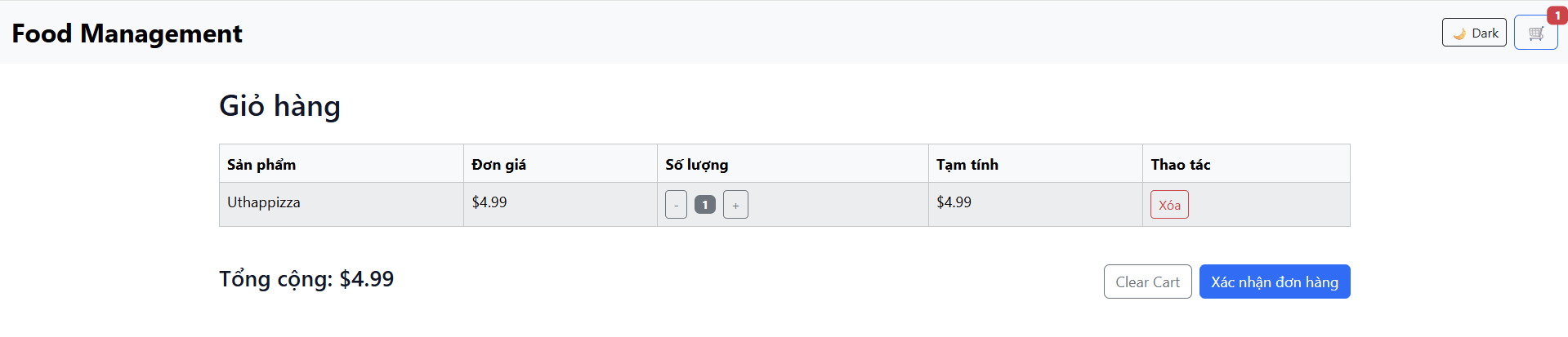
  );

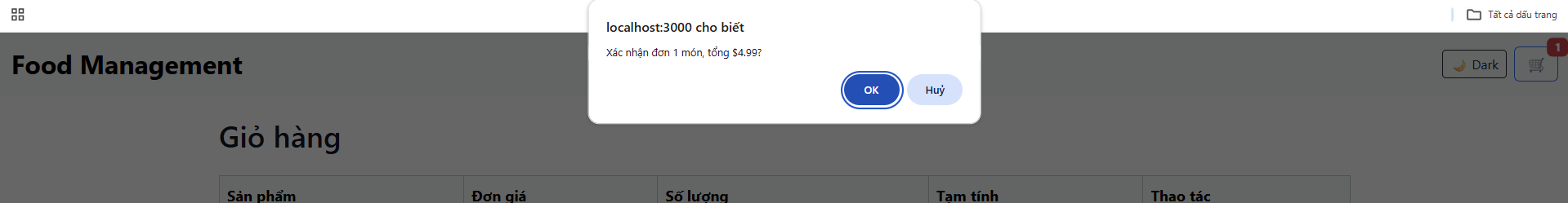
};

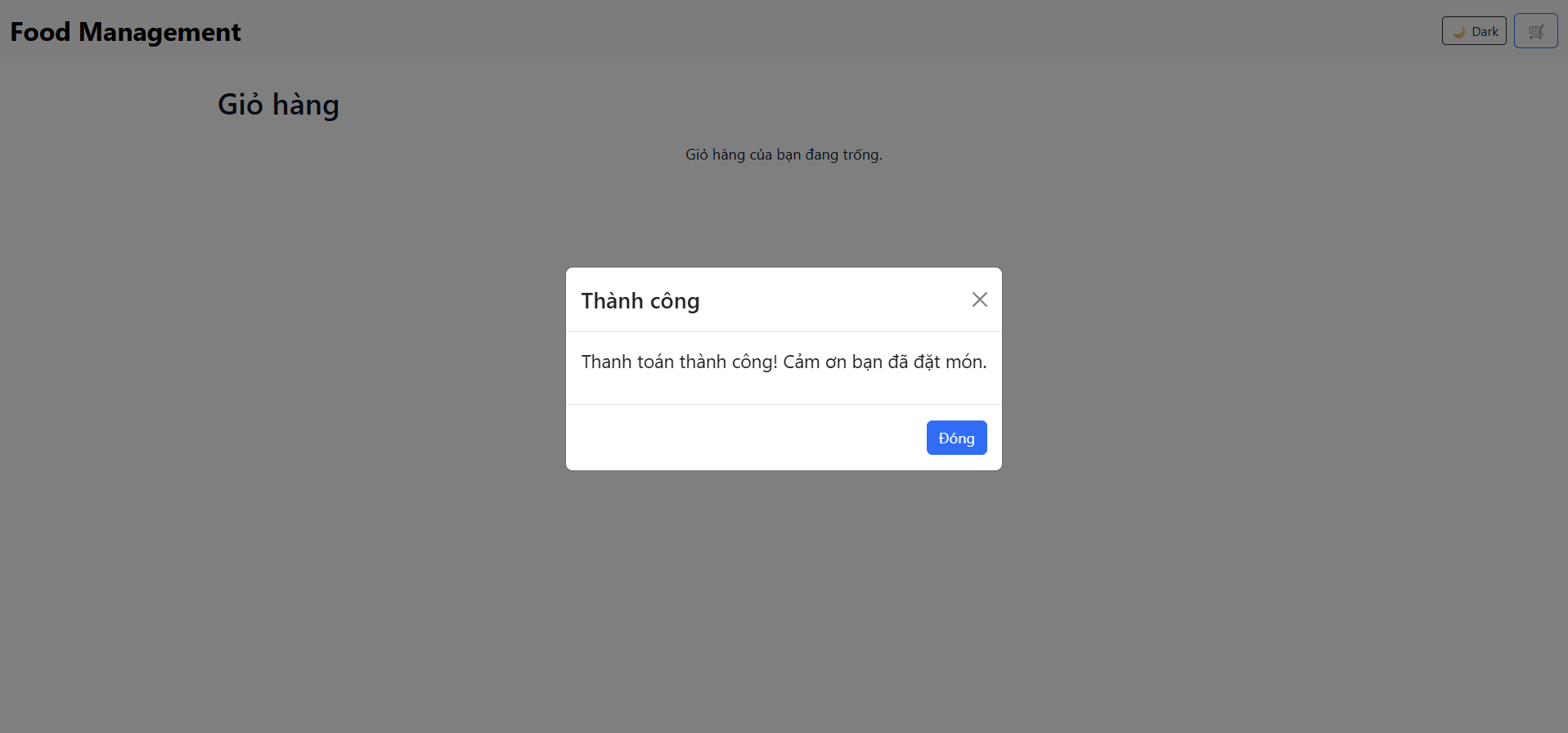
export default DishesList;

## Cart.js









import React, { useContext, useState } from "react";

import { CartContext } from "./CartContext";

import { Container, Table, Button, Badge, Row, Col, Modal } from "react-bootstrap";

const Cart = ({ dark }) => {

  const { cartItems, addToCart, removeFromCart, removeItem, clearCart, totalValue, totalCount } =

    useContext(CartContext);

  const [showSuccessModal, setShowSuccessModal] = useState(false);

  const handleCheckout = () => {

    if (cartItems.length === 0) {

      window.alert("Giỏ hàng đang trống. Vui lòng chọn món trước khi thanh toán.");

      return;

    }

    const ok = window.confirm(

      `Xác nhận đơn ${totalCount} món, tổng $${totalValue}?`

    );

    if (ok) {

      clearCart();

      setShowSuccessModal(true);

    }

  };

  const handleCloseModal = () => {

    setShowSuccessModal(false);

  };

  return (

    <Container>

      <h2 className="my-4">Giỏ hàng</h2>

      {cartItems.length === 0 ? (

        <p className="text-center my-4">Giỏ hàng của bạn đang trống.</p>

      ) : (

        <div>

          {/\* Thêm variant cho Table để hỗ trợ dark mode \*/}

          <Table responsive striped bordered hover variant={dark ? "dark" : "light"}>

            <thead>

              <tr>

                <th>Sản phẩm</th>

                <th>Đơn giá</th>

                <th>Số lượng</th>

                <th>Tạm tính</th>

                <th>Thao tác</th>

              </tr>

            </thead>

            <tbody>

              {cartItems.map((item) => {

                const price = parseFloat(item.price);

                const qty = item.quantity || 1;

                return (

                  <tr key={item.id}>

                    <td>{item.name}</td>

                    <td>${price.toFixed(2)}</td>

                    <td>

                      <div className="d-flex align-items-center">

                        <Button

                          variant="outline-secondary"

                          size="sm"

                          onClick={() => removeFromCart(item.id)}

                        >

                          -

                        </Button>

                        <Badge bg="secondary" className="mx-2">{qty}</Badge>

                        <Button

                          variant="outline-secondary"

                          size="sm"

                          onClick={() => addToCart(item)}

                        >

                          +

                        </Button>

                      </div>

                    </td>

                    <td>${(price \* qty).toFixed(2)}</td>

                    <td>

                      <Button

                        variant="outline-danger"

                        size="sm"

                        onClick={() => removeItem(item.id)}

                      >

                        Xóa

                      </Button>

                    </td>

                  </tr>

                );

              })}

            </tbody>

          </Table>

          <Row className="mt-4">

            <Col className="d-flex justify-content-between align-items-center">

              <h4>Tổng cộng: ${totalValue}</h4>

              <div>

                <Button variant="outline-secondary" onClick={clearCart} className="me-2">

                  Clear Cart

                </Button>

                <Button variant="primary" onClick={handleCheckout}>

                  Xác nhận đơn hàng

                </Button>

              </div>

            </Col>

          </Row>

        </div>

      )}

      {/\* Modal thông báo thành công \*/}

      <Modal

        show={showSuccessModal}

        onHide={handleCloseModal}

        centered

        data-bs-theme={dark ? "dark" : "light"}

      >

        <Modal.Header closeButton>

          <Modal.Title>Thành công</Modal.Title>

        </Modal.Header>

        <Modal.Body>

          <p className="text-center fs-5">Thanh toán thành công! Cảm ơn bạn đã đặt món.</p>

        </Modal.Body>

        <Modal.Footer>

          <Button variant="primary" onClick={handleCloseModal}>

            Đóng

          </Button>

        </Modal.Footer>

      </Modal>

    </Container>

  );

};

export default Cart;

## CartContext.js

import React, { createContext, useState, useEffect, useMemo } from "react";

export const CartContext = createContext();

export const CartProvider = ({ children }) => {

  const [cartItems, setCartItems] = useState([]); // [{ id, name, price, image, description, quantity }]

  // Thêm 1 đơn vị vào giỏ. Nếu đã có thì tăng quantity

  const addToCart = (dish) =>

    setCartItems((previousItems) => {

      const index = previousItems.findIndex((item) => item.id === dish.id);

      if (index !== -1) {

        const updated = [...previousItems];

        updated[index] = {

          ...updated[index],

          quantity: updated[index].quantity + 1,

        };

        return updated;

      }

      return [...previousItems, { ...dish, quantity: 1 }];

    });

  // Giảm 1 đơn vị. Nếu về 0 thì xóa dòng đó

  const removeFromCart = (id) => {

    setCartItems((previousItems) => {

      const index = previousItems.findIndex((item) => item.id === id);

      if (index === -1) return previousItems;

      const updated = [...previousItems];

      const current = updated[index];

      if (current.quantity <= 1) {

        updated.splice(index, 1);

      } else {

        updated[index] = { ...current, quantity: current.quantity - 1 };

      }

      return updated;

    });

  };

  // Xóa hẳn 1 dòng item khỏi giỏ

  const removeItem = (id) => {

    setCartItems((previousItems) => previousItems.filter((item) => item.id !== id));

  };

  const clearCart = () => setCartItems([]);

  const totalValue = useMemo(() => {

    const sum = cartItems.reduce(

      (accumulator, item) =>

        accumulator + parseFloat(item.price) \* (item.quantity || 1),

      0

    );

    return sum.toFixed(2);

  }, [cartItems]);

  const totalCount = useMemo(() => {

    return cartItems.reduce((accumulator, item) => accumulator + (item.quantity || 1), 0);

  }, [cartItems]);

  useEffect(() => {

    const saved = JSON.parse(localStorage.getItem("cartItems"));

    if (Array.isArray(saved)) {

      // Chuẩn hóa dữ liệu cũ (danh sách trùng lặp) thành dạng có quantity

      if (saved.length > 0 && saved[0] && saved[0].quantity == null) {

        const map = new Map();

        for (const it of saved) {

          const prev = map.get(it.id);

          if (prev) {

            map.set(it.id, { ...prev, quantity: (prev.quantity || 1) + 1 });

          } else {

            map.set(it.id, { ...it, quantity: 1 });

          }

        }

        setCartItems(Array.from(map.values()));

      } else {

        setCartItems(saved);

      }

    }

  }, []);

  useEffect(() => {

    localStorage.setItem("cartItems", JSON.stringify(cartItems));

  }, [cartItems]);

  return (

    <CartContext.Provider

      value={{ cartItems, addToCart, removeFromCart, removeItem, clearCart, totalValue, totalCount }}

    >

      {children}

    </CartContext.Provider>

  );

};

## App.js

import React, { useMemo, useState } from "react";

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

import { CartProvider } from "./components/CartContext";

import DishesList from "./components/DishesList";

import Cart from "./components/Cart";

import Header from "./components/Header";

import "./styles.css";

const dishes = [

  {

    id: 0, name: "Uthappizza", image: "images/uthappizza.png", price: "4.99",

    description: "Kết hợp độc đáo giữa Uthappam Ấn Độ và pizza Ý."

  },

  {

    id: 1, name: "Zucchipakoda", image: "images/zucchipakoda.jpg", price: "1.99",

    description: "Bí ngòi chiên bột đậu gà giòn rụm."

  },

  {

    id: 2, name: "Vadonut", image: "images/vadonut.png", price: "1.99",

    description: "Pha trộn giữa vada và donut."

  },

  {

    id: 3, name: "ElaiCheese Cake", image: "images/elaicheesecake.jpg", price: "2.99",

    description: "Cheesecake kiểu New York thêm thảo quả Ấn Độ."

  },

];

export default function App() {

  // 1. Tạo state để theo dõi chế độ dark mode

  const [dark, setDark] = useState(false);

  // 2. Tạo hàm để chuyển đổi giữa light và dark mode

  const toggleTheme = () => {

    setDark(prevDark => !prevDark);

  };

  // 3. Sử dụng useMemo để tạo class conditionally

  const rootClass = useMemo(() => (dark ? "app dark" : "app"), [dark]);

  return (

    <CartProvider>

      <BrowserRouter>

        {/\* 4. Thêm data-bs-theme để Bootstrap nhận biết theme \*/}

        <div className={rootClass} data-bs-theme={dark ? "dark" : "light"}>

          {/\* 5. Truyền state và hàm xuống các component cần thiết \*/}

          <Header dark={dark} onToggleTheme={toggleTheme} />

          <main className="content">

            <Routes>

              <Route path="/" element={<DishesList dishes={dishes} dark={dark} />} />

              <Route path="/cart" element={<Cart dark={dark} />} />

            </Routes>

          </main>

        </div>

      </BrowserRouter>

    </CartProvider>

  );

}

