**Từng bước rõ ràng** để chuyển giỏ hàng từ fix cứng sang dùng dữ liệu thật từ CartContext.

**🧩 Tổng quan các bước**

1. **[✓] Xác định vị trí fix cứng ItemDetails**
2. **[Bước 1] Thêm cartItems vào CartContext**
3. **[Bước 2] Sửa ShoppingCartTemplate.jsx để dùng cartItems.map()**
4. **[Bước 3] Sửa ItemDetails.jsx để nhận và hiển thị item**
5. (Tùy chọn sau này) Kết nối với nút "Add to Cart" để thêm sản phẩm vào cartItems

**✅ Bắt đầu từ bước 1**

**🔧 BƯỚC 1: Thêm cartItems vào CartContext**

**🔹 Mục tiêu:**

Tạo biến cartItems lưu danh sách sản phẩm trong giỏ hàng.

**📁 Mở file: src/Components/context/CartContext.js**

**✏️ Thêm/Chỉnh sửa:**

jsx

Sao chépChỉnh sửa

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

// 1. Tạo context

const CartContext = createContext();

// 2. Provider bao bọc toàn app

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

const [isCartOpen, setIsCartOpen] = useState(false);

const [cartItems, setCartItems] = useState([]); // ✅ thêm dòng này

const openCart = () => setIsCartOpen(true);

const closeCart = () => setIsCartOpen(false);

// ✅ Hàm thêm sản phẩm

const addToCart = (item) => {

setCartItems((prev) => [...prev, item]);

};

return (

<CartContext.Provider

value={{ isCartOpen, openCart, closeCart, cartItems, addToCart }} // ✅ truyền thêm

>

{children}

</CartContext.Provider>

);

};

// 3. Custom hook cho tiện

export const useCart = () => useContext(CartContext);

**✅ Sau khi xong bước này, bạn đã có:**

* cartItems chứa danh sách sản phẩm trong giỏ
* addToCart(item) để thêm sản phẩm
* useCart() trả về cartItems để hiển thị

**🔧 BƯỚC 2: Dùng .map() trong ShoppingCartTemplate.jsx để hiển thị giỏ hàng thật**

**📁 Mở file: ShoppingCartTemplate.jsx**

**✏️ Tìm và XÓA đoạn fix cứng này:**

jsx

Sao chépChỉnh sửa

<ItemDetails />

<ItemDetails />

<ItemDetails />

<ItemDetails />

**✏️ Sau đó thêm đoạn sau vào thay thế:**

jsx

Sao chépChỉnh sửa

{cartItems.length === 0 ? (

<p className="text-muted">Your cart is empty.</p>

) : (

cartItems.map((item, index) => (

<ItemDetails key={index} item={item} />

))

)}

**✏️ Đồng thời đảm bảo bạn có dòng này ở đầu file:**

js

Sao chépChỉnh sửa

const { isCartOpen, closeCart, cartItems } = useCart(); // 🟢 lấy thêm cartItems

**✅ Kết quả sau chỉnh sửa (phần chính):**

jsx

Sao chépChỉnh sửa

<div className="col-lg-7">

<div className="text-start mb-3">

<button

className="btn btn-info btn-lg"

onClick={() => {

closeCart();

navigate("/");

}}

>

<i className="fas fa-long-arrow-alt-left me-2" />

<strong>Continue Shopping</strong>

</button>

</div>

<hr />

<div className="d-flex justify-content-between align-items-center mb-4">

<div>

<h4>Shopping cart</h4>

<p className="mb-0">

You have {cartItems.length} item{cartItems.length !== 1 ? "s" : ""} in your cart </p>

</div>

<div>

<p className="mb-0">

<span className="text-muted">Sort by:</span>{" "}

<a href="#!" className="text-body">

price <i className="fas fa-angle-down" />

</a>

</p>

</div>

</div>

{/\* 🟢 Duyệt danh sách sản phẩm trong giỏ \*/}

{cartItems.length === 0 ? (

<p className="text-muted">Your cart is empty.</p>

) : (

cartItems.map((item, index) => (

<ItemDetails key={index} item={item} />

))

)}

</div>

**📌 Giải thích:**

* cartItems.map(...) sẽ lặp qua từng sản phẩm.
* item là từng object chứa thông tin sản phẩm (title, price, image, v.v.)
* Dữ liệu này sẽ truyền vào component ItemDetails để hiển thị.
* **🔧 BƯỚC 3: Sửa ItemDetails.jsx để nhận và hiển thị dữ liệu từ item**
* **📁 Mở file: ItemDetails.jsx**
* **🎯 Mục tiêu:**
* Sửa component này để nhận prop item từ ShoppingCartTemplate và hiển thị thông tin sản phẩm thật (title, price, image, quantity,...)
* **✏️ Mã ItemDetails.jsx đã chỉnh sửa theo form của bạn:**
* jsx
* Sao chépChỉnh sửa
* import React, { useState } from "react";
* function ItemDetails({ item }) {
* const [quantity, setQuantity] = useState(item.quantity || 1);
* const increaseQuantity = () => {
* setQuantity((prev) => prev + 1);
* };
* const decreaseQuantity = () => {
* setQuantity((prev) => (prev > 1 ? prev - 1 : 1));
* };
* return (
* <div>
* <div className="card mb-3">
* <div className="card-body">
* <div className="d-flex justify-content-between">
* {/\* LEFT: Image + Title \*/}
* <div className="d-flex flex-row align-items-center">
* <div>
* <img
* src={item.image}
* className="img-fluid rounded-3"
* alt={item.title}
* style={{ width: 65 }}
* />
* </div>
* <div className="ms-3">
* <h5>{item.title}</h5>
* <p className="small mb-0">{item.description || "No description"}</p>
* </div>
* </div>
* {/\* RIGHT: Quantity + Price + Delete \*/}
* <div className="d-flex flex-row align-items-center gap-3">
* {/\* Quantity \*/}
* <div className="d-flex align-items-center gap-2">
* <button
* className="btn btn-sm btn-outline-secondary"
* onClick={decreaseQuantity}
* >
* −
* </button>
* <h5 className="fw-normal mb-0">{quantity}</h5>
* <button
* className="btn btn-sm btn-outline-primary"
* onClick={increaseQuantity}
* >
* +
* </button>
* </div>
* {/\* Price \*/}
* <div style={{ width: 80 }}>
* <h5 className="mb-0">${(item.price \* quantity).toFixed(2)}</h5>
* </div>
* {/\* Delete \*/}
* <a href="#!" style={{ color: "#cecece" }}>
* <i className="fas fa-trash-alt" />
* </a>
* </div>
* </div>
* </div>
* </div>
* </div>
* );
* }
* export default ItemDetails;
* **✅ Giải thích thay đổi:**

| **Thay vì fix cứng** | **Đã thay bằng** |
| --- | --- |
| Iphone 11 pro | item.title |
| img src=... | item.image |
| $900 \* quantity | item.price \* quantity |
| quantity = 1 fix | `useState(item.quantity |

**🛒 Tiếp theo: Thêm sản phẩm vào giỏ hàng từ nút "Add to Cart" (trang chính)**

**✅ Mục tiêu:**

* Khi người dùng bấm nút **"Add to Cart"** tại mỗi sản phẩm → dữ liệu được đẩy vào giỏ hàng qua addToCart().
* Giỏ hàng (ShoppingCartTemplate) hiển thị đúng sản phẩm vừa thêm.
* Header hiển thị đúng số lượng.
* **🔧 BƯỚC 1: Đảm bảo context CartContext đã có hàm addToCart()**
* **📁 Mở CartContext.js và kiểm tra:**
* jsx
* Sao chépChỉnh sửa
* const addToCart = (product) => {
* setCartItems((prevItems) => {
* const existingItem = prevItems.find((item) => item.id === product.id);
* if (existingItem) {
* // Nếu đã có sản phẩm → tăng số lượng
* return prevItems.map((item) =>
* item.id === product.id
* ? { ...item, quantity: item.quantity + 1 }
* : item
* );
* } else {
* // Nếu chưa có → thêm mới
* return [...prevItems, { ...product, quantity: 1 }];
* }
* });
* };
* ✅ Và đừng quên export addToCart từ context:
* js
* Sao chépChỉnh sửa
* value={{ cartItems, addToCart, isCartOpen, openCart, closeCart }}

**🔧 BƯỚC 2: Gọi addToCart() từ Product.jsx**

Trong file Product.jsx, bạn cần:

jsx

Sao chépChỉnh sửa

import { useCart } from "../context/CartContext"; // đường dẫn đúng của bạn

const Product = ({ id, image, title, price, description }) => {

const { addToCart } = useCart();

const handleAddToCart = (e) => {

e.stopPropagation(); // ⛔ Ngăn không bị click lan sang phần navigate

addToCart({ id, image, title, price });

};

return (

<div className="card h-100 shadow-sm">

<img

src={image}

alt={title}

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

style={{ cursor: "pointer" }}

className="card-img-top"

/>

<div className="card-body">

<h5 className="card-title">{title}</h5>

<p>{description}</p>

<p className="text-success fw-bold">${price}</p>

<button className="btn btn-primary w-100" onClick={handleAddToCart}>

Add to Cart

</button>

</div>

</div>

);

};

**🔧 BƯỚC 3: Mở giỏ hàng để test**

Bạn có thể:

* Tự động mở giỏ hàng sau khi thêm:

js

Sao chépChỉnh sửa

addToCart(...);

openCart(); // nếu bạn muốn tự bật overlay giỏ hàng luôn

* Hoặc nhấn icon 🛒 trên Header để kiểm tra số lượng.

**✅ Kết quả mong đợi:**

* Click nút "Add to Cart" trên mỗi sản phẩm → sản phẩm xuất hiện trong giỏ hàng.
* Nếu click 2 lần → số lượng tăng lên (quantity).
* Header hiển thị số lượng đúng (cartItems.length).
* ShoppingCartTemplate hiển thị danh sách bằng cartItems.map(...).

🎯 **xoá sản phẩm khỏi giỏ hàng**

**✅ Mục tiêu:**

Khi người dùng nhấn vào biểu tượng 🗑 (thùng rác) → sản phẩm đó sẽ **biến mất khỏi giỏ hàng**.

**🧱 Kiến trúc bạn đang dùng:**

* CartContext.js quản lý giỏ hàng → ta cần thêm hàm removeFromCart(id)
* ShoppingCartTemplate.jsx render danh sách bằng ItemDetails
* ItemDetails.jsx hiện mỗi sản phẩm và chứa icon xoá (<i className="fas fa-trash-alt" />)

**🛠️ Các bước thực hiện:**

**🛠️ Các bước thực hiện:**

**✅ BƯỚC 1: Thêm hàm removeFromCart trong CartContext.js**

js

Sao chépChỉnh sửa

const removeFromCart = (id) => {

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

};

➡️ Và đừng quên export trong value:

js

Sao chépChỉnh sửa

<CartContext.Provider

value={{

cartItems,

addToCart,

removeFromCart, // 👈 thêm dòng này

isCartOpen,

openCart,

closeCart,

}}

>

**✅ BƯỚC 2: Truyền product.id vào ItemDetails**

Mở ShoppingCartTemplate.jsx, thay vì hard-code ItemDetails() nhiều lần, sửa thành map:

jsx

Sao chépChỉnh sửa

{cartItems.map((item) => (

<ItemDetails key={item.id} item={item} />

))}

**✅ BƯỚC 3: Sửa ItemDetails.jsx để gọi removeFromCart**

**1. Nhận item qua props:**

js

Sao chépChỉnh sửa

function ItemDetails({ item }) {

const { removeFromCart } = useCart();

const [quantity, setQuantity] = useState(item.quantity);

**2. Sửa phần nội dung:**

jsx

Sao chépChỉnh sửa

<img

src={item.image}

className="img-fluid rounded-3"

alt={item.title}

style={{ width: 65 }}

/>

<h5>{item.title}</h5>

<p className="small mb-0">{item.description}</p>

**3. Gắn onClick vào biểu tượng xoá:**

jsx

Sao chépChỉnh sửa

<a

href="#!"

style={{ color: "#cecece" }}

onClick={() => removeFromCart(item.id)} // ✅ xoá sản phẩm

>

<i className="fas fa-trash-alt" />

</a>

**🎯 Kết quả sau khi làm:**

* Mỗi sản phẩm trong giỏ có icon thùng rác 🗑
* Bấm vào sẽ gọi removeFromCart(item.id)
* Context tự cập nhật, cartItems render lại → sản phẩm biến mất khỏi giao diện

**Hiển thị tổng số sản phẩm trong giỏ hàng** ngay trong trang ShoppingCartTemplate

**✅ Mục tiêu:**

Hiển thị:

text

Sao chépChỉnh sửa

You have 3 items in your cart

→ Và con số 3 là **tổng số lượng thực tế**, không phải số loại sản phẩm.

 Khi bấm + hoặc – trong giỏ hàng → phải **cập nhật cartItems trong context**

 Nhờ đó, tổng số sản phẩm (totalItems) cũng cập nhật đúng

**🧠 Ví dụ:**

Nếu cartItems = [  
{ id: 1, quantity: 2 },  
{ id: 2, quantity: 1 }  
]  
→ Tổng cộng có **3 sản phẩm** trong giỏ.

**🛠️ Cách sửa từng bước:**

**🔧 BƯỚC 1: Cập nhật lại logic trong CartContext.js**

**➕ Thêm hàm updateQuantity**

Trong CartContext.js:

js

Sao chépChỉnh sửa

const updateQuantity = (id, newQuantity) => {

setCartItems((prevItems) =>

prevItems.map((item) =>

item.id === id ? { ...item, quantity: newQuantity } : item

)

);

};

Và nhớ export trong value:

js

Sao chépChỉnh sửa

value={{

cartItems,

addToCart,

removeFromCart,

updateQuantity, // 👈 THÊM DÒNG NÀY

...

}}

**🔧 BƯỚC 2: Dùng updateQuantity trong ItemDetails.jsx**

Trong ItemDetails.jsx:

**1. Import updateQuantity từ context:**

js

Sao chépChỉnh sửa

const { removeFromCart, updateQuantity } = useCart();

**2. Loại bỏ useState(quantity) vì ta dùng dữ liệu từ context rồi**

js

Sao chépChỉnh sửa

// ❌ XÓA DÒNG NÀY:

// const [quantity, setQuantity] = useState(item.quantity);

**3. Sửa lại increaseQuantity và decreaseQuantity:**

js

Sao chépChỉnh sửa

const increaseQuantity = () => {

updateQuantity(item.id, item.quantity + 1);

};

const decreaseQuantity = () => {

if (item.quantity > 1) {

updateQuantity(item.id, item.quantity - 1);

}

};

**4. Thay quantity bằng item.quantity trong JSX:**

jsx

Sao chépChỉnh sửa

<h5 className="fw-normal mb-0">{item.quantity}</h5>

Và chỗ tính giá:

jsx

Sao chépChỉnh sửa

<h5 className="mb-0">${(item.price \* item.quantity).toFixed(2)}</h5>

**✅ Kết quả:**

| **Hành động** | **Kết quả** |
| --- | --- |
| Bấm + hoặc – | Context cập nhật số lượng thật |
| Tổng số sản phẩm (totalItems) tự tăng/giảm đúng |  |
| Không cần dùng useState(quantity) local nữa |  |

**🔁 Ví dụ: totalItems sẽ được tính đúng như sau:**

js

Sao chépChỉnh sửa

const totalItems = cartItems.reduce((sum, item) => sum + item.quantity, 0);

Hiển thị:

jsx

Sao chépChỉnh sửa

<p className="mb-0">

You have <strong>{totalItems}</strong> item{totalItems !== 1 ? "s" : ""} in your cart

</p>

✅ Khi tăng/giảm sản phẩm từ giỏ hàng (ItemDetails),  
👉 **số lượng hiển thị tại icon giỏ hàng trên Header** (🛒) **cũng cần tự động cập nhật**!

**🧠 Giải thích logic:**

Bạn đang hiển thị **số lượng giỏ hàng** ở ReactHeaderTemplates.jsx kiểu như:

jsx

Sao chépChỉnh sửa

<span className="badge bg-danger ms-1">{cartItems.length}</span>

⚠️ Nhưng cartItems.length chỉ là **số loại sản phẩm**,  
→ Không tính tổng số lượng.

**✅ Cách chuẩn nhất:**

**👉 Dùng reduce() để tính tổng số lượng thực:**

jsx

Sao chépChỉnh sửa

const totalItems = cartItems.reduce((sum, item) => sum + item.quantity, 0);

**🔧 Bước sửa trong ReactHeaderTemplates.jsx**

**1. Thêm cartItems từ context:**

js

Sao chépChỉnh sửa

const { cartItems } = useCart();

**2. Tính totalItems:**

js

Sao chépChỉnh sửa

const totalItems = cartItems.reduce((sum, item) => sum + item.quantity, 0);

**3. Thay cartItems.length bằng totalItems:**

jsx

Sao chépChỉnh sửa

<button className="btn position-relative" onClick={openCart}>

<i className="bi bi-cart3 fs-4"></i>

{totalItems > 0 && (

<span className="position-absolute top-0 start-100 translate-middle badge rounded-pill bg-danger">

{totalItems}

</span>

)}

</button>

**✅ Kết quả:**

| **Hành động** | **Hiển thị icon giỏ hàng** |
| --- | --- |
| Thêm sản phẩm | 🔴 Badge tăng |
| Bấm + trong giỏ | 🔴 Badge tăng |
| Bấm – trong giỏ | 🔵 Badge giảm |
| Bấm 🗑 xoá sản phẩm | 🟡 Badge cập nhật đúng |