Dùng **json-server** để tạo một **API giả lập thanh toán**

**✅ BƯỚC 1: Cài đặt json-server**

Chạy lệnh sau trong thư mục gốc của project:

bash

Sao chépChỉnh sửa

npm install -g json-server

(Nếu bạn dùng môi trường local riêng cho project, thì cài local: npm install --save-dev json-server)

**✅ BƯỚC 2: Tạo file db.json**

Tại thư mục gốc project, tạo file db.json:

json

Sao chépChỉnh sửa

{

"orders": []

}

Tức là API sẽ có 1 endpoint POST/GET tại: http://localhost:3001/orders

**✅ BƯỚC 3: Chạy server**

Chạy lệnh:

bash

Sao chépChỉnh sửa

json-server --watch db.json --port 3001

Bạn sẽ thấy log:

bash

Sao chépChỉnh sửa

Resources

http://localhost:3001/orders

**✅ BƯỚC 4: Tạo file ordersApi.js**

Tạo file: src/api/ordersApi.js

js

Sao chépChỉnh sửa

// src/api/ordersApi.js

export const submitOrder = async (orderData) => {

const response = await fetch("http://localhost:3001/orders", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify(orderData),

});

if (!response.ok) {

throw new Error("Đặt hàng thất bại");

}

return await response.json();

};

**✅ BƯỚC 5: Gọi trong CardDetails.jsx**

Ở đầu file:

js

Sao chépChỉnh sửa

import { submitOrder } from "../../api/ordersApi";

Rồi sửa onSubmit:

js

Sao chépChỉnh sửa

onSubmit={async (values, actions) => {

const { total } = calculateOrderFromCart(cartItems, values.shippingMethod);

const orderData = {

...values,

cartItems,

total,

createdAt: new Date().toISOString(),

};

try {

await submitOrder(orderData);

toast.success("Đặt hàng thành công!");

actions.resetForm();

} catch (error) {

toast.error("Đặt hàng thất bại!");

console.error(error);

}

}}

**✅ BƯỚC 6: Kiểm tra kết quả**

* Mở trình duyệt: <http://localhost:3001/orders>
* Bạn sẽ thấy đơn hàng đã được thêm vào danh sách orders.

**✅ TỔNG KẾT**

| **Việc cần làm** | **Đã hoàn thành?** |
| --- | --- |
| Cài json-server | ✅ |
| Tạo db.json | ✅ |
| Chạy server tại port 3001 | ✅ |
| Tạo ordersApi.js | ✅ |
| Gửi đơn hàng từ CardDetails | ✅ |
| Hiển thị Toast kết quả | ✅ |

**Thêm toast thông báo khi checkout thành công hoặc thất bại** bằng react-toastify

**🎯 Mục tiêu:**

Hiện thông báo ✅ "Đặt hàng thành công!" hoặc ❌ "Đặt hàng thất bại!" khi nhấn Checkout.

**🪜 BƯỚC 1: Cài đặt react-toastify**

Chạy lệnh:

bash

Sao chépChỉnh sửa

npm install react-toastify

**🪜 BƯỚC 2: Cấu hình ToastContainer trong App.jsx**

Ở App.jsx, bạn thêm 2 dòng sau:

**✏️ Import:**

js

Sao chépChỉnh sửa

import { ToastContainer } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

**✅ Thêm vào JSX của App (thường cuối cùng):**

jsx

Sao chépChỉnh sửa

<>

{/\* ...các component khác \*/}

<ToastContainer position="top-right" autoClose={3000} />

</>

**🪜 BƯỚC 3: Dùng toast trong CardDetails.jsx**

**✏️ Import ở đầu file:**

js

Sao chépChỉnh sửa

import { toast } from "react-toastify";

**🧠 Trong onSubmit của FormikWrapper, bạn viết như sau:**

jsx

Sao chépChỉnh sửa

onSubmit={async (values, actions) => {

const { total } = calculateOrderFromCart(cartItems, values.shippingMethod);

const orderData = {

...values,

cartItems,

total,

createdAt: new Date().toISOString(),

};

try {

await submitOrder(orderData);

toast.success("✅ Đặt hàng thành công!");

actions.resetForm();

} catch (error) {

toast.error("❌ Đặt hàng thất bại. Vui lòng thử lại.");

console.error(error);

}

}}

**✅ Kết quả sau khi làm:**

| **Tình huống** | **Thông báo** |
| --- | --- |
| Gửi đơn hàng thành công | ✅ "Đặt hàng thành công!" |
| Gửi đơn hàng thất bại (server lỗi, tắt json-server...) | ❌ "Đặt hàng thất bại" |

**Reset giỏ hàng (cartItems) sau khi đặt hàng thành công**

**✅ MỤC TIÊU:**

Sau khi submit thành công, cartItems sẽ bị xóa sạch, giống như Shopee khi thanh toán xong.

**🪜 BƯỚC 1: Thêm action "CLEAR\_CART" trong CartContext.jsx**

👉 Mở CartContext.jsx, trong cartReducer, bạn thêm:

js

Sao chépChỉnh sửa

case "CLEAR\_CART":

return [];

📌 Full ví dụ cartReducer sau khi thêm:

js

Sao chépChỉnh sửa

const cartReducer = (state, action) => {

switch (action.type) {

case "ADD\_TO\_CART": {

...

}

case "REMOVE\_FROM\_CART": {

...

}

case "UPDATE\_QUANTITY": {

...

}

case "CLEAR\_CART": {

return [];

}

default:

return state;

}

};

**🪜 BƯỚC 2: Thêm hàm clearCart trong CartProvider**

Ngay dưới các hàm addToCart, removeFromCart, bạn thêm:

js

Sao chépChỉnh sửa

const clearCart = () => {

dispatch({ type: "CLEAR\_CART" });

};

Rồi thêm vào value={...} trong CartContext.Provider:

js

Sao chépChỉnh sửa

<CartContext.Provider

value={{

isCartOpen,

openCart,

closeCart,

cartItems,

addToCart,

removeFromCart,

updateQuantity,

clearCart, // ✅ thêm dòng này

}}

>

**🪜 BƯỚC 3: Gọi clearCart() trong CardDetails.jsx**

👉 Trong CardDetails.jsx, bạn thêm clearCart từ hook:

js

Sao chépChỉnh sửa

const { cartItems, clearCart } = useCart();

👉 Sau khi submitOrder(orderData) thành công, bạn gọi:

js

Sao chépChỉnh sửa

clearCart(); // ✅ reset giỏ hàng

**📌 Toàn bộ onSubmit sau khi cập nhật:**

jsx

Sao chépChỉnh sửa

onSubmit={async (values, actions) => {

const { total } = calculateOrderFromCart(cartItems, values.shippingMethod);

const orderData = {

...values,

cartItems,

total,

createdAt: new Date().toISOString(),

};

try {

await submitOrder(orderData);

toast.success("✅ Đặt hàng thành công!");

clearCart(); // ✅ xoá giỏ hàng sau khi đặt xong

actions.resetForm(); // reset form

} catch (error) {

toast.error("❌ Đặt hàng thất bại!");

console.error(error);

}

}}

**✅ KẾT QUẢ:**

| **Sau khi đặt hàng** | **Kết quả** |
| --- | --- |
| Đơn hàng gửi lên API | ✅ |
| Hiện toast thành công | ✅ |
| Giỏ hàng trống lại (cartItems = []) | ✅ |
| Form reset sạch sẽ | ✅ |