**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/backend/util.js” file, update the following code:**

import jwt from 'jsonwebtoken';

import config from './config';

// const jwt = require('jsonwebtoken');

// const config = require('./config');

const getToken = (user) => {

    return jwt.sign({

        \_id: user.\_id,

        name: user.name,

        email: user.email,

        isAdmin: user.isAdmin,

    }, config.JWT\_SECRET, {

        expiresIn: '48h'

    })

}

const isAuth = (req, res, next) => {

    const token = req.headers.authorization;

    if(token) {

        const onlyToken = token.slice(7, token.length);

        jwt.verify(onlyToken, config.JWT\_SECRET, (err, decode) => {

            if(err) {

                return res.status(401).send({ message: 'Invalid Token' });

            }

            req.user = token;

            next();

            return

        });

    }

    return res.status(401).send({ message: "Token is not supplied." });

}

const isAdmin = (req, res, next) => {

    if(req.user && req.user.isAdmin) {

        return next();

    }

    return res.status(401).send({ message: 'Admin Token is not valid.' });

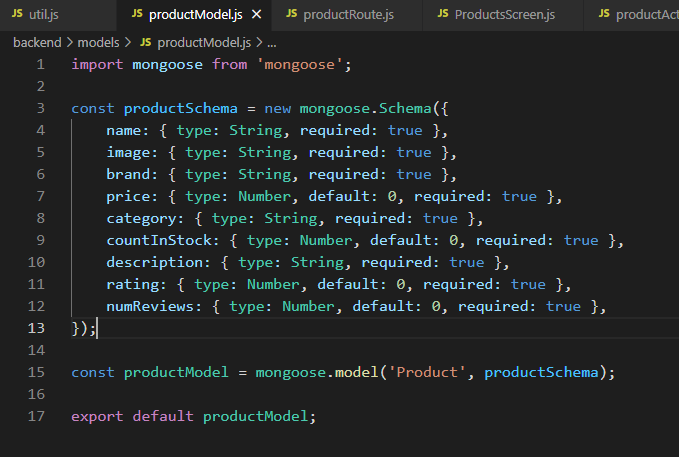
}

export {

    getToken, isAuth, isAdmin

}

**- Create “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/backend/models/productModel.js” file, add the following code:**



**- Create “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/backend/routes/productRoute.js” file, add the following code:**

import express from 'express';

import Product from '../models/productModel';

import { getToken } from '../util';

const router = express.Router();

router.get("/", async (req, res) => {

    const products = await Product.find({});

    res.send(products);

});

router.post("/", async (req, res) => {

    const product = new Product({

        name: req.body.name,

        price: req.body.price,

        image: req.body.image,

        brand: req.body.brand,

        category: req.body.category,

        countInStock: req.body.countInStock,

        description: req.body.description,

        rating: req.body.rating,

        numReviews: req.body.numReviews,

    });

    const newProduct = await product.save();

    if(newProduct) {

        return res.status(201).send({ message: 'New Product Created', data: newProduct });

    }

    return res.status(500).send({ message: 'Error in Creating Product.' });

})

router.put("/:id", async (req, res) => {

    const productId = req.params.id;

    const product = await Product.findById(productId);

    if(product){

        product.name = req.body.name;

        product.price = req.body.price;

        product.image = req.body.image;

        product.brand = req.body.brand;

        product.category = req.body.category;

        product.countInStock = req.body.countInStock;

        product.description = req.body.description;

        const updatedProduct = await product.save();

        if(updatedProduct) {

            return res.status(200).send({ message: 'Product Updated', data: updatedProduct });

        }

    }

    return res.status(500).send({ message: 'Error in Updating Product.' });

})

export default router;

**- Create “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/screens/ProductsScreen.js” file, add the following code:**

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

import { useSelector, useDispatch } from 'react-redux';

import { deleteProduct, listProducts, saveProduct } from '../actions/productActions';

function ProductsScreen(props) {

    const [modalVisible, setModalVisible] = useState(false);

    const [id, setId] = useState('');

    const [name, setName] = useState('');

    const [price, setPrice] = useState('');

    const [image, setImage] = useState('');

    const [brand, setBrand] = useState('');

    const [category, setCategory] = useState('');

    const [countInStock, setCountInStock] = useState('');

    const [description, setDescription] = useState('');

    // ...

    const productList = useSelector(state => state.productList);

    const { loading, products, error } = productList;

    // ...

    const productSave = useSelector(state => state.productSave);

    const { loading: loadingSave, success: successSave, error: errorSave } = productSave;

    const productDelete = useSelector(state => state.productDelete);

    const { loading: loadingDelete, success: successDelete, error: errorDelete } = productDelete;

    const dispatch = useDispatch();

    useEffect(() => {

        if(successSave){

            setModalVisible(false);

        }

        dispatch(listProducts());

        return () => {

            //

        };

    }, [successSave]);

    // Open the modal (Create new product Form)

    const openModal = (product) => {

        setModalVisible(true);

        setId(product.\_id);

        setName(product.name);

        setPrice(product.price);

        setDescription(product.description);

        setImage(product.image);

        setBrand(product.brand);

        setCategory(product.category);

        setCountInStock(product.countInStock);

    }

    // Submit Create the new product

    const submitHandler = (e) => {

        e.preventDefault();

        dispatch(saveProduct({ \_id: id, name, price, image, brand, category, countInStock, description}));

    }

    // Handler Delete the product

    const deleteHandler = (product) => {

        dispatch(deleteProduct(product.\_id));

    }

    return <div className="content content-margined">

        <div className="product-header">

            <h3>Products</h3>

            <button onClick={() => openModal({})}>Create Product</button>

        </div>

        {modalVisible &&

            <div className="form">

                <form onSubmit={submitHandler}>

                    <ul className="form-container">

                        <li>

                            <h2>Create Product</h2>

                        </li>

                        <li>

                            {loadingSave && <div>Loading...</div>}

                            {errorSave && <div>{errorSave}</div>}

                        </li>

                        <li>

                            <label htmlFor="name">Name</label>

                            <input type="text" name="name" value={name} id="name" onChange={(e) => setName(e.target.value)}>

                            </input>

                        </li>

                        <li>

                            <label htmlFor="price">Price</label>

                            <input type="text" name="price" value={price} id="price" onChange={(e) => setPrice(e.target.value)}>

                            </input>

                        </li>

                        <li>

                            <label htmlFor="image">Image</label>

                            <input type="text" name="image" value={image} id="image" onChange={(e) => setImage(e.target.value)}>

                            </input>

                        </li>

                        <li>

                            <label htmlFor="brand">Brand</label>

                            <input type="text" name="brand" value={brand} id="brand" onChange={(e) => setBrand(e.target.value)}>

                            </input>

                        </li>

                        <li>

                            <label htmlFor="countInStock">CountInStock</label>

                            <input type="text" name="countInStock" value={countInStock} id="countInStock" onChange={(e) => setCountInStock(e.target.value)}>

                            </input>

                        </li>

                        <li>

                            <label htmlFor="category">Category</label>

                            <input type="text" name="category" value={category} id="category" onChange={(e) => setCategory(e.target.value)}>

                            </input>

                        </li>

                        <li>

                            <label htmlFor="description">Description</label>

                            <textarea name="description" value={description} id="description" onChange={(e) => setDescription(e.target.value)}>

                            </textarea>

                        </li>

                        <li>

                            <button type="submit" className="button primary">{id ? "Update" : "Create"}</button>

                        </li>

                        <li>

                            <button type="button" onClick={()=>setModalVisible(false)} className="button secondary">Back</button>

                        </li>

                    </ul>

                </form>

            </div>

        }

        <div className="product-list">

            <table className="table">

                <thead>

                    <tr>

                        <th>ID</th>

                        <th>Name</th>

                        <th>Price</th>

                        <th>Category</th>

                        <th>Brand</th>

                        <th>Action</th>

                    </tr>

                </thead>

                <tbody>

                    {products.map((product) => (

                        <tr>

                            <td>{product.\_id}</td>

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

                            <td>{product.price}</td>

                            <td>{product.category}</td>

                            <td>{product.brand}</td>

                            <td>

                                <button className="button" onClick={() => openModal(product)}>Edit</button>{' '}

                                <button className="button" onClick={() => deleteHandler(product)}>Delete</button>

                            </td>

                        </tr>

                    ))}

                </tbody>

            </table>

        </div>

    </div>

}

export default ProductsScreen;

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/actions/productActions.js” file, update the following code:**

import {  PRODUCT\_DELETE\_FAIL, PRODUCT\_DELETE\_REQUEST, PRODUCT\_DELETE\_SUCCESS, PRODUCT\_DETAILS\_FAIL, PRODUCT\_DETAILS\_REQUEST, PRODUCT\_DETAILS\_SUCCESS, PRODUCT\_LIST\_FAIL, PRODUCT\_LIST\_REQUEST, PRODUCT\_LIST\_SUCCESS, PRODUCT\_SAVE\_FAIL, PRODUCT\_SAVE\_REQUEST, PRODUCT\_SAVE\_SUCCESS } from "../constants/productConstants"

import axios from 'axios';

// import Axios from 'axios';

const listProducts = () => async (dispatch) => {

    try {

        dispatch({ type: PRODUCT\_LIST\_REQUEST });

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

        dispatch({ type: PRODUCT\_LIST\_SUCCESS, payload: data });

    }

    catch (error) {

        dispatch({ type: PRODUCT\_LIST\_FAIL, payload: error.message });

    }

}

const saveProduct = (product) => async (dispatch, getState) => {

    try {

        dispatch({ type: PRODUCT\_SAVE\_REQUEST, payload: product });

        const { userSignin:{ userInfo } } = getState();

        if(!product.\_id) {

            const { data } = await axios.post('/api/products', product, {

                headers: {

                    'Authorization': 'Bearer' + userInfo.token,

                }

            });

            dispatch({ type: PRODUCT\_SAVE\_SUCCESS, payload: data });

        } else {

            const { data } = await axios.put('/api/products/' + product.\_id, product, {

                headers: {

                    'Authorization': 'Bearer' + userInfo.token,

                }

            });

            dispatch({ type: PRODUCT\_SAVE\_SUCCESS, payload: data });

        }

    } catch (error) {

        dispatch({ type: PRODUCT\_SAVE\_FAIL, payload: error.message });

    }

}

const detailsProduct = (productId) => async (dispatch) => {

    try {

        dispatch({ type: PRODUCT\_DETAILS\_REQUEST, payload: productId });

        const { data } = await axios.get("/api/products/" + productId);

        dispatch({ type: PRODUCT\_DETAILS\_SUCCESS, payload: data });

    } catch (error) {

        dispatch({ type: PRODUCT\_DETAILS\_FAIL, payload: error.message });

    }

}

const deleteProduct = (productId) => async (dispatch, getState) => {

    try {

        const { userSignin: { userInfo } } = getState();

        dispatch({ type: PRODUCT\_DELETE\_REQUEST, payload: productId });

        const { data } = await axios.delete("/api/products/" + productId, {

            headers: {

                Authorization: 'Bearer' + userInfo.token

            }

        });

        dispatch({ type: PRODUCT\_DELETE\_SUCCESS, payload: data, success: true });

    } catch (error) {

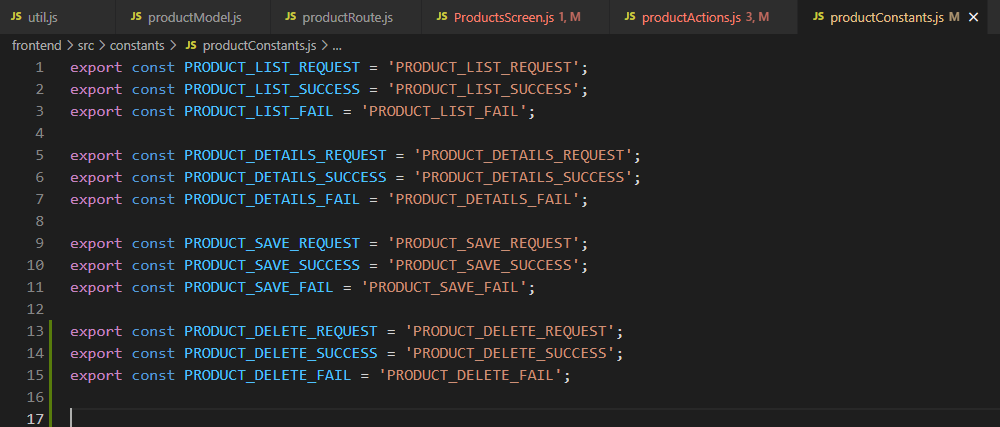
        dispatch({ type: PRODUCT\_DELETE\_FAIL, payload: error.message });

    }

}

export { listProducts, detailsProduct, saveProduct, deleteProduct }

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/constants/productConstants.js” file, update the following code:**



**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/reducers/productReducers.js” file, update the following code:**

import { PRODUCT\_DELETE\_FAIL, PRODUCT\_DELETE\_REQUEST, PRODUCT\_DELETE\_SUCCESS, PRODUCT\_DETAILS\_FAIL, PRODUCT\_DETAILS\_REQUEST, PRODUCT\_DETAILS\_SUCCESS, PRODUCT\_LIST\_FAIL, PRODUCT\_LIST\_REQUEST, PRODUCT\_LIST\_SUCCESS, PRODUCT\_SAVE\_FAIL, PRODUCT\_SAVE\_REQUEST, PRODUCT\_SAVE\_SUCCESS } from "../constants/productConstants";

function productListReducer(state = { products: [] }, action) {

    switch (action.type) {

        case PRODUCT\_LIST\_REQUEST:

            return { loading: true, products: [] };

        case PRODUCT\_LIST\_SUCCESS:

            return { loading: false, products: action.payload };

        case PRODUCT\_LIST\_FAIL:

            return { loading: false, error: action.payload };

        default:

            return state;

    }

}

function productDetailsReducer(state = { product: {} }, action) {

    switch (action.type) {

        case PRODUCT\_DETAILS\_REQUEST:

            return { loading: true };

        case PRODUCT\_DETAILS\_SUCCESS:

            return { loading: false, product: action.payload };

        case PRODUCT\_DETAILS\_FAIL:

            return { loading: false, error: action.payload };

        default:

            return state;

    }

}

function productDeleteReducer(state = { product: {} }, action) {

    switch (action.type) {

        case PRODUCT\_DELETE\_REQUEST:

            return { loading: true };

        case PRODUCT\_DELETE\_SUCCESS:

            return { loading: false, product: action.payload, success: true };

        case PRODUCT\_DELETE\_FAIL:

            return { loading: false, error: action.payload };

        default:

            return state;

    }

}

function productSaveReducer(state = { product: {} }, action) {

    switch (action.type) {

        case PRODUCT\_SAVE\_REQUEST:

            return { loading: true };

        case PRODUCT\_SAVE\_SUCCESS:

            return { loading: false, success: true, product: action.payload };

        case PRODUCT\_SAVE\_FAIL:

            return { loading: false, error: action.payload };

        default:

            return state;

    }

}

export { productListReducer, productDetailsReducer, productSaveReducer, productDeleteReducer }

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/store.js” file, update the following code:**

import { createStore, combineReducers, applyMiddleware, compose } from 'redux';

import thunk from 'redux-thunk';

// import Cookies from 'js-cookie';

import { productListReducer, productDetailsReducer, productSaveReducer, productDeleteReducer } from './reducers/productReducers';

import { cartReducer } from './reducers/cartReducers';

import { userRegisterReducer, userSigninReducer } from './reducers/userReducers';

// Nếu muốn lưu với Cookie (Server Side)

const Cookies = require('js-cookie');

// Nếu muốn lưu với Cookie (Server Side)

const initialState = {

    cart: { cartItems: Cookies.get('cartItems')? JSON.parse(Cookies.get('cartItems')) : [] },

    userSignin: { userInfo: Cookies.get('userInfo')? JSON.parse(Cookies.get('userInfo')) : null },

 };

// Nếu muốn lưu với Local Storage (Client Side)

// const initialState = {

//     cart: {

//         cartItems: localStorage.getItem('cartItems')? JSON.parse(localStorage.getItem('cartItems')): []

//     }

// };

const reducer = combineReducers({

    productList: productListReducer,

    productDetails: productDetailsReducer,

    cart: cartReducer,

    userSignin: userSigninReducer,

    userRegister: userRegisterReducer,

    productSave: productSaveReducer,

    productDelete: productDeleteReducer,

})

const composeEnhancer = window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;

const store = createStore(reducer, initialState, composeEnhancer(applyMiddleware(thunk)));

export default store;

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/App.js” file, update the following code:**

import React from 'react';

import { useSelector } from 'react-redux';

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

import './App.css';

import CartScreen from './screens/CartScreen';

import HomeScreen from './screens/HomeScreen';

import ProductScreen from './screens/ProductScreen';

import ProductsScreen from './screens/ProductsScreen';

import RegisterScreen from './screens/RegisterScreen';

import SigninScreen from './screens/SigninScreen';

function App() {

  const userSignin = useSelector( state => state.userSignin);

  const { userInfo } = userSignin;

  const openMenu = () => {

    document.querySelector(".sidebar").classList.add("open");

  }

  const closeMenu = () => {

    document.querySelector(".sidebar").classList.remove("open");

  }

  return (

    <BrowserRouter>

    <div className="grid-container">

        <header className="header">

            <div className="brand">

                <button onClick={openMenu}>

                    &#9776;

                </button>

                <Link to="/" >Sneaker Shop</Link>

            </div>

            <div className="header-links">

                <a href="cart.html">Cart</a>

                {

                    userInfo ? <Link to="/profile">{userInfo.name}</Link> : <Link to="/signin">Sign In</Link>

                }

            </div>

        </header>

        <aside className="sidebar">

            <h3>Shopping Categories</h3>

            <button className="sidebar-close-button" onClick={closeMenu}>x</button>

            <ul>

                <li>

                    <a href="index.html">Man</a>

                </li>

                <li>

                    <a href="index.html">Woman</a>

                </li>

            </ul>

        </aside>

        <main className="main">

            <div className="content">

                <Route path="/products" component={ProductsScreen} />

                <Route path="/signin" component={SigninScreen} />

                <Route path="/register" component={RegisterScreen} />

                <Route path="/product/:id" component={ProductScreen} />

                <Route path="/cart/:id?" component={CartScreen} />

                <Route path="/" exact={true} component={HomeScreen} />

            </div>

        </main>

        <footer className="footer">

            All right reserved.

        </footer>

    </div>

    </BrowserRouter>

  );

}

export default App;

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/backend/server.js” file, update the following code:**

import express from 'express';

import data from './data';

// import dotenv from 'dotenv';

import config from './config';

import mongoose from 'mongoose';

import bodyParser from 'body-parser';

import userRoute from './routes/userRoute';

import productRoute from './routes/productRoute';

// import cors from 'cors';

// const express = require('express')

// const data = require('./data')

// const dotenv = require('dotenv')

// const config = require('./config')

// const mongoose = require('mongoose')

// const bodyParser = require('body-parser')

// const userRoute = require('./routes/userRoute')

// const cors = require('cors');

// dotenv.config();

const mongodbUrl = config.MONGODB\_URL;

mongoose.connect(mongodbUrl, {

    useNewUrlParser: true

}).catch(error => console.log(error.reason));

const app = express();

// Đại chỉ viết khúc này để test nó có chạy không lỗi khi trình duyệt trả về localhost:5000 không

// app.get('/',(req,res)=>{

//     res.json('abc')

// })

// Set cors in NodeJS (Nếu lỗi Cors)

// app.use(cors());

// Set cors in NodeJS (Nếu lỗi Cors)

// app.use(function(req, res, next) {

    //     res.header("Access-Control-Allow-Origin", "\*");

    //     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    //     next();

    // });

app.use(bodyParser.json());

Don’t need it anymore

app.use("/api/users", userRoute);

app.use("/api/products", productRoute);

// app.get("/api/products/:id", (req, res) => {

//     const productId = req.params.id;

//     const product = data.products.find(x=>x.\_id === productId);

//     if(product)

//         res.send(product);

//     else

//         res.status(404).send({ msg: "Product Not Found." })

// });

// app.get("/api/products", (req, res) => {

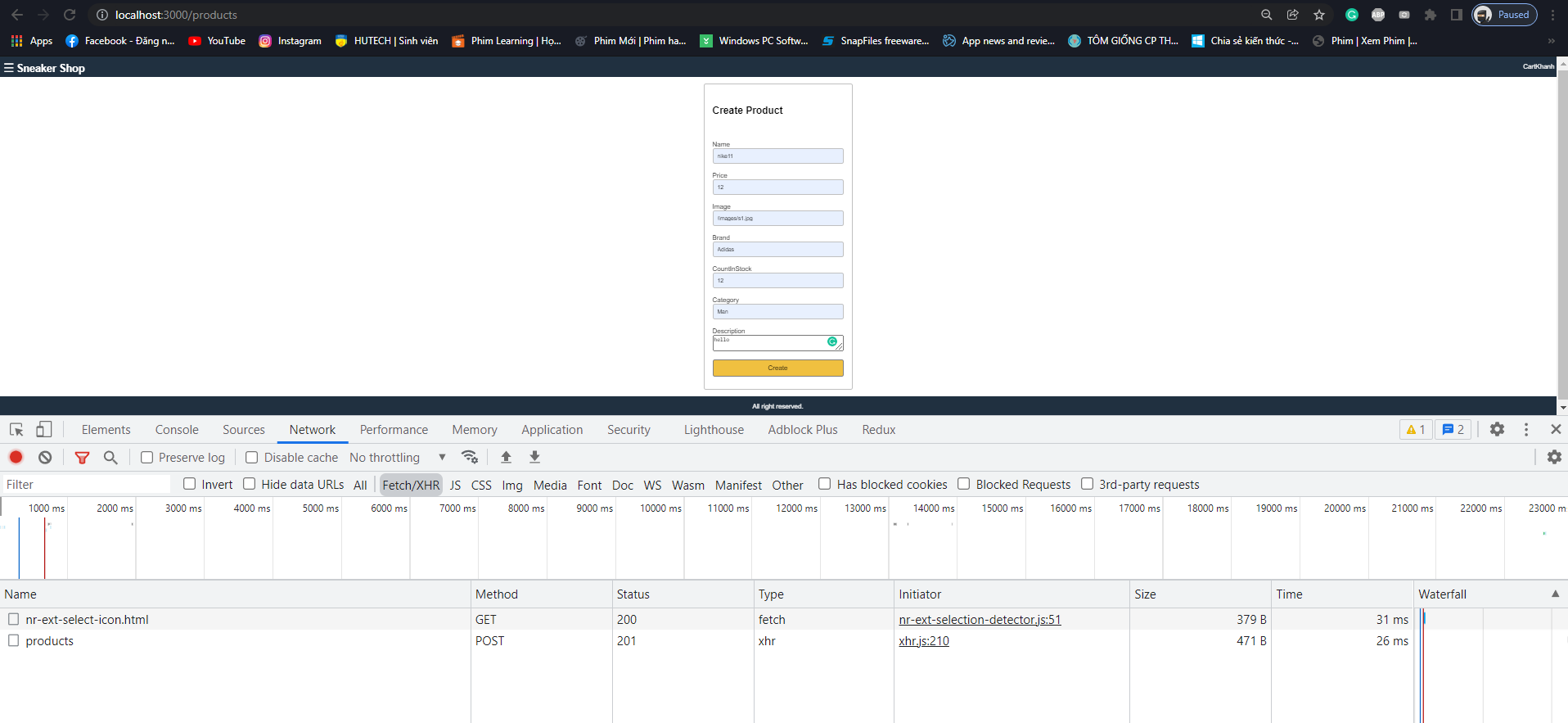
//     res.send(data.products);

// });

app.listen(5000, () => { console.log("Server started at http://localhost:5000") });

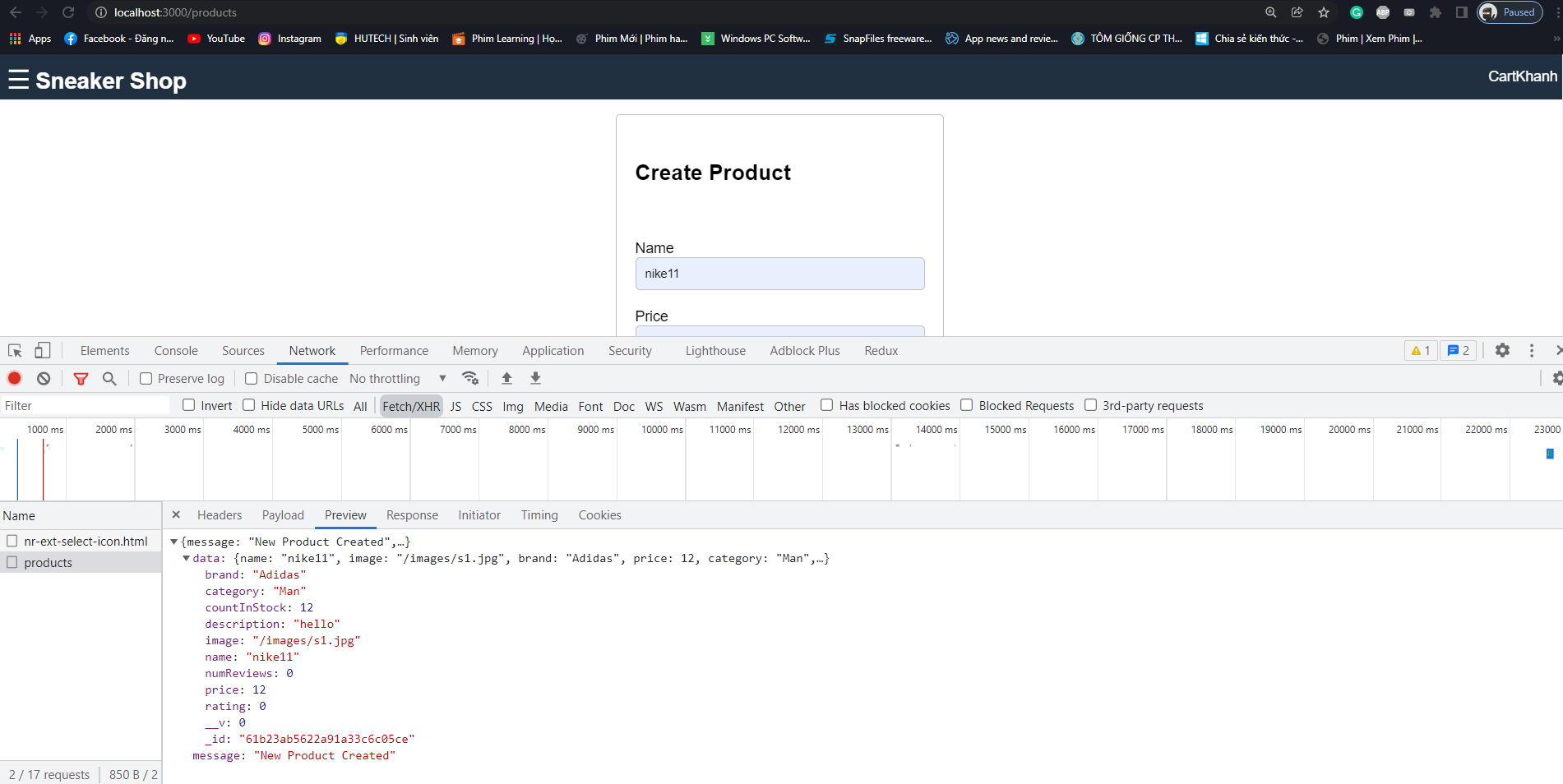
**- Result: to checking ... 4:16:34**

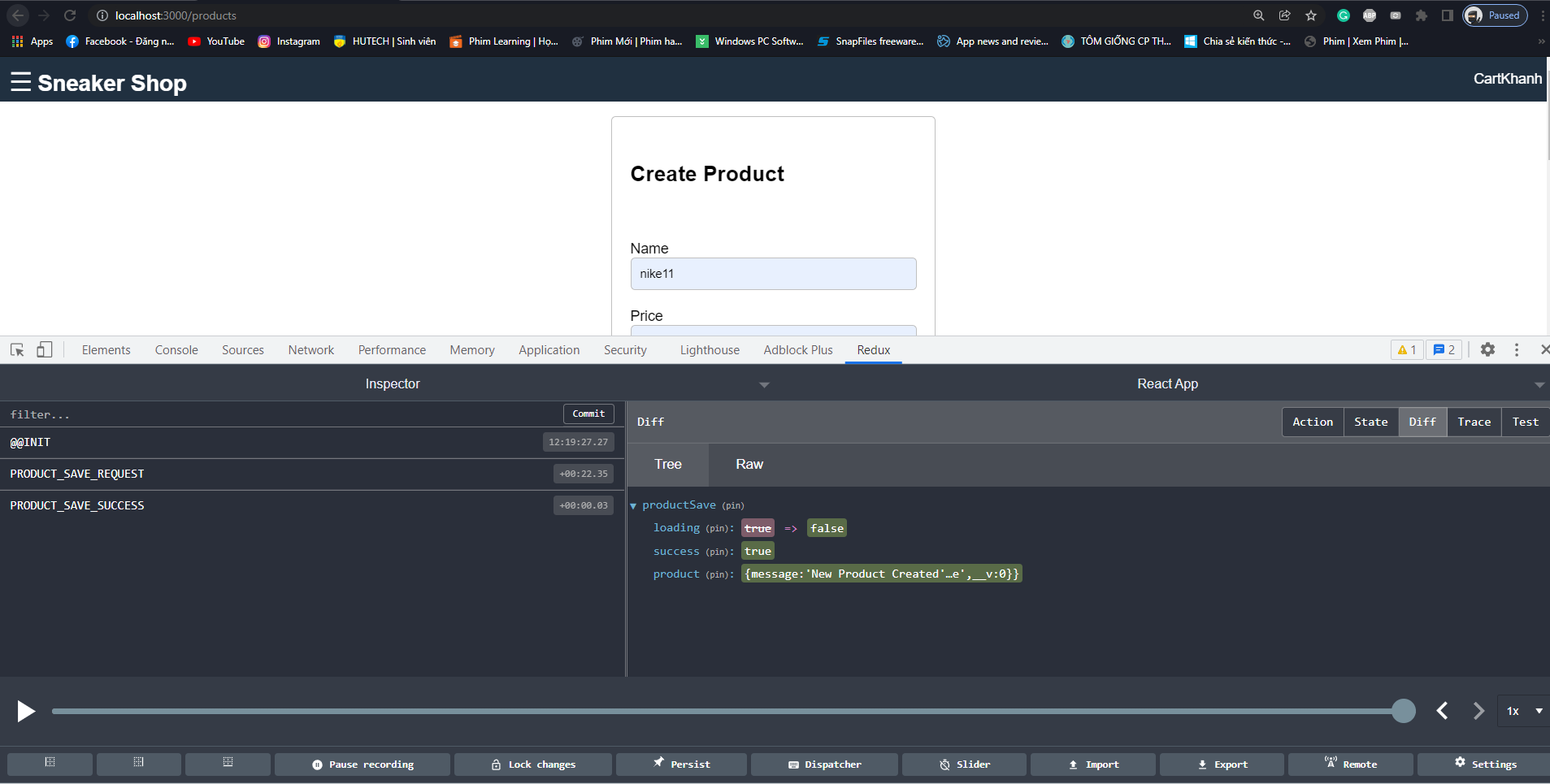
**\* Note: chỉ khi đăng nhập với User Account (là Admin hoặc không phải Admin) mới có quyền thêm sản phẩm mới vào database.**

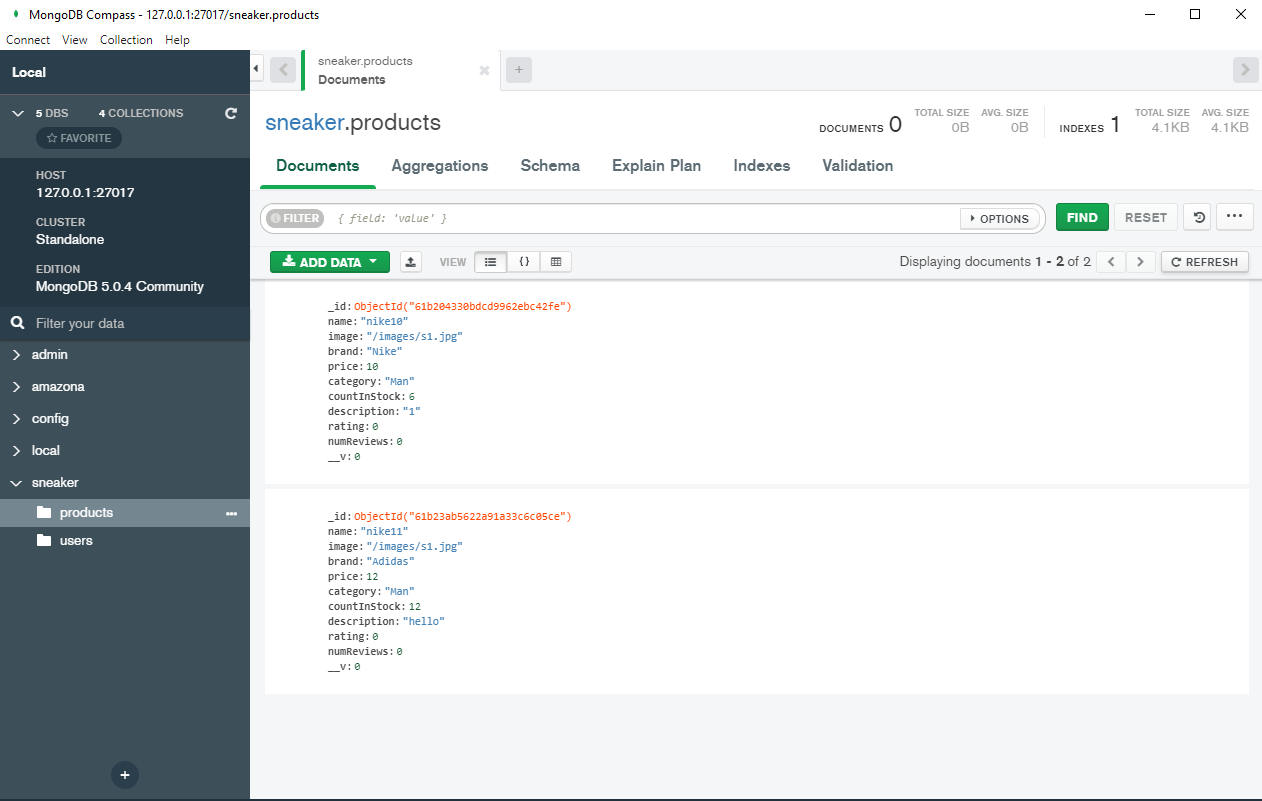


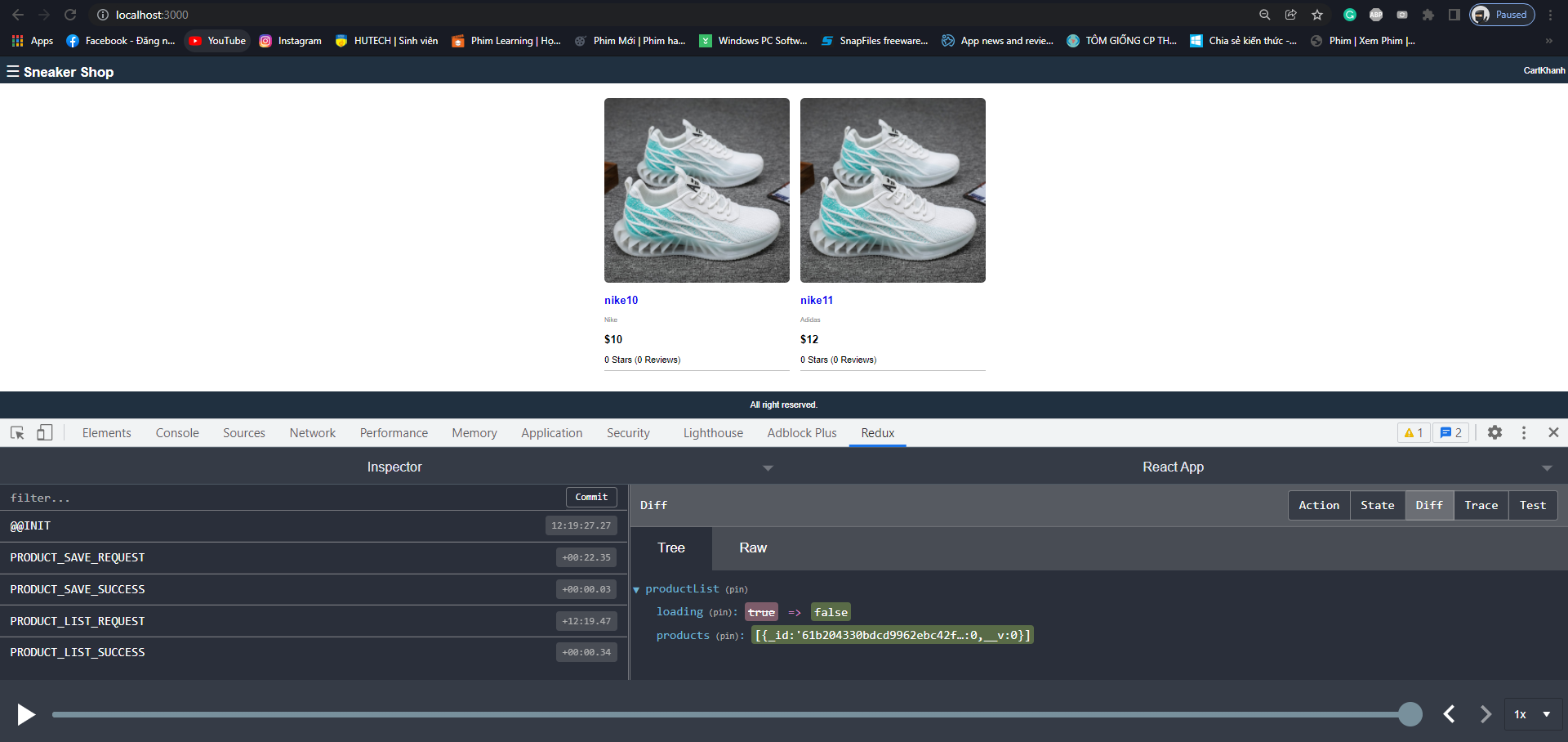
Signed in with User Account (Admin or non-Admin)

Filled in all input field and clicked the “Create” button









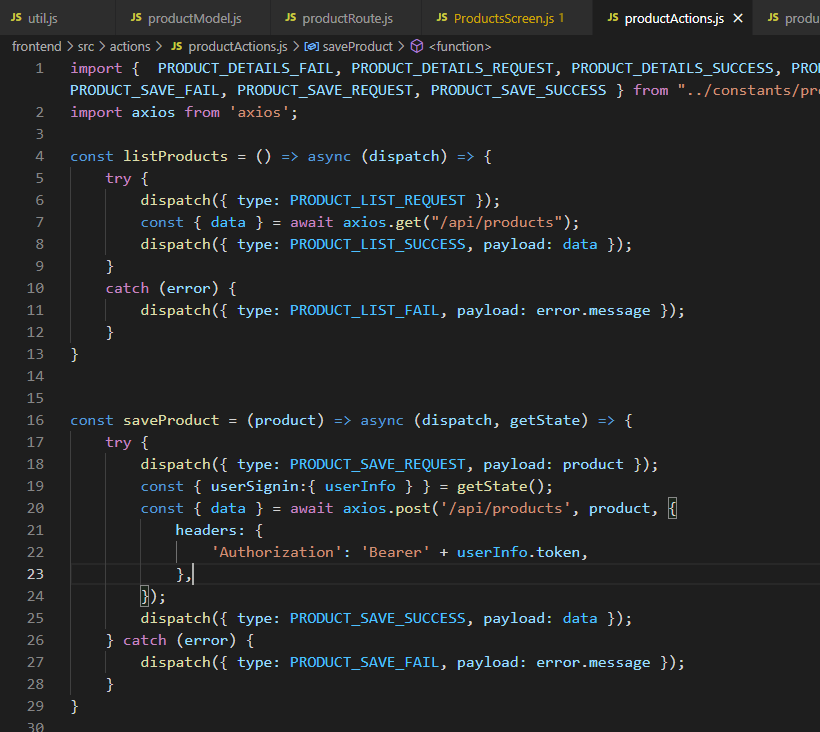
**(Error 1 ... 4:16:45) Sau khi điền vào hết tất cả các ô trong form Create Product và click nút Create mà ở Network nó không trả về gì hết, còn ở Redux thì nó bắn ra Constant “PRODUCT\_SAVE\_FAIL”**

**(Solution (Error 1)): Làm như trong clip mà không ra được kết quả mong muốn thì thực hiện những thao tác sau:**

**+ Xóa tất cả cache và cookie trên trình duyệt**

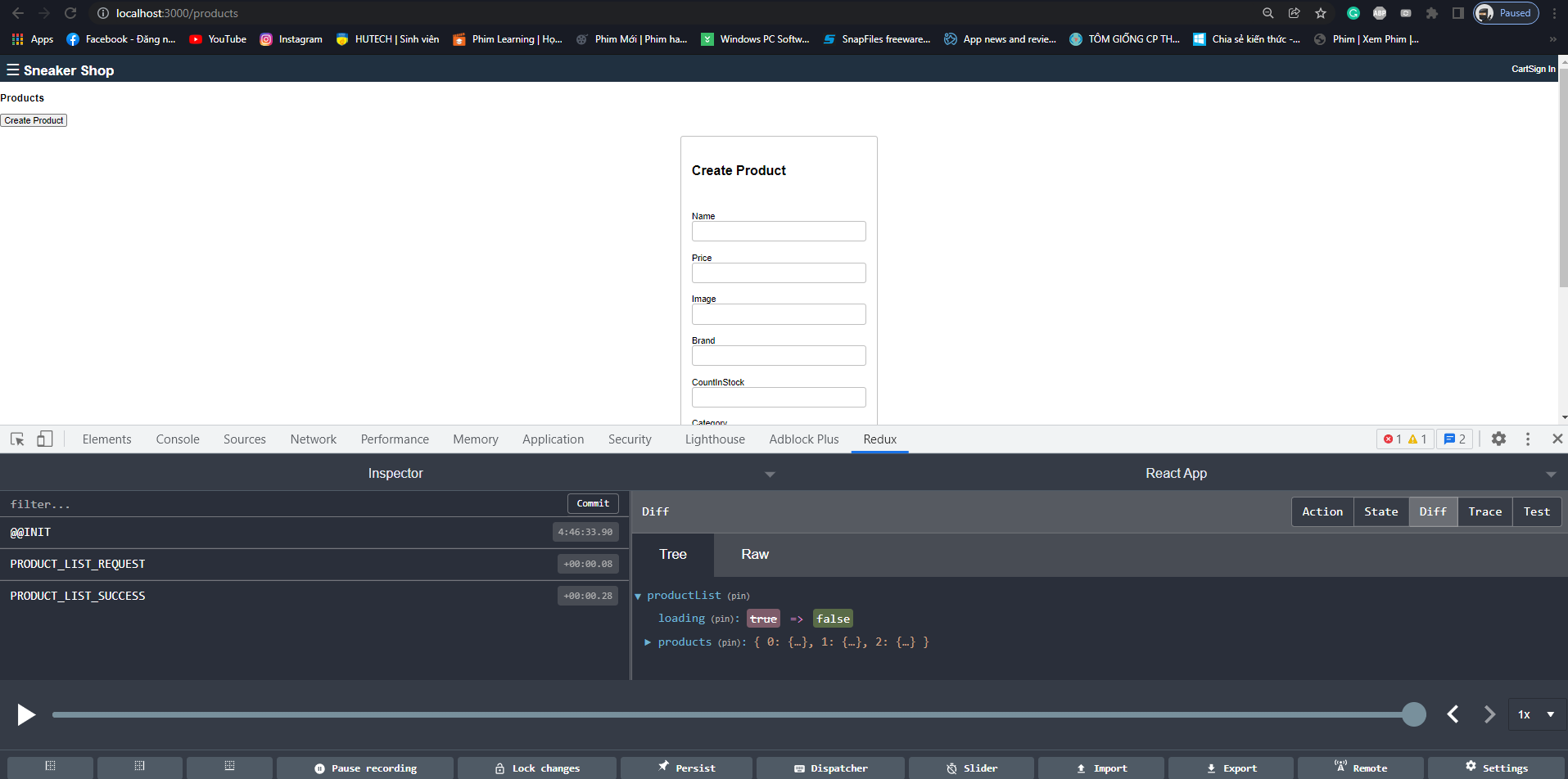
**+ Nhớ đăng nhập với User Account (Admin or non-Admin) trước khi vào form Create Product để thêm product**

**+ Ở “productActions.js” file:**

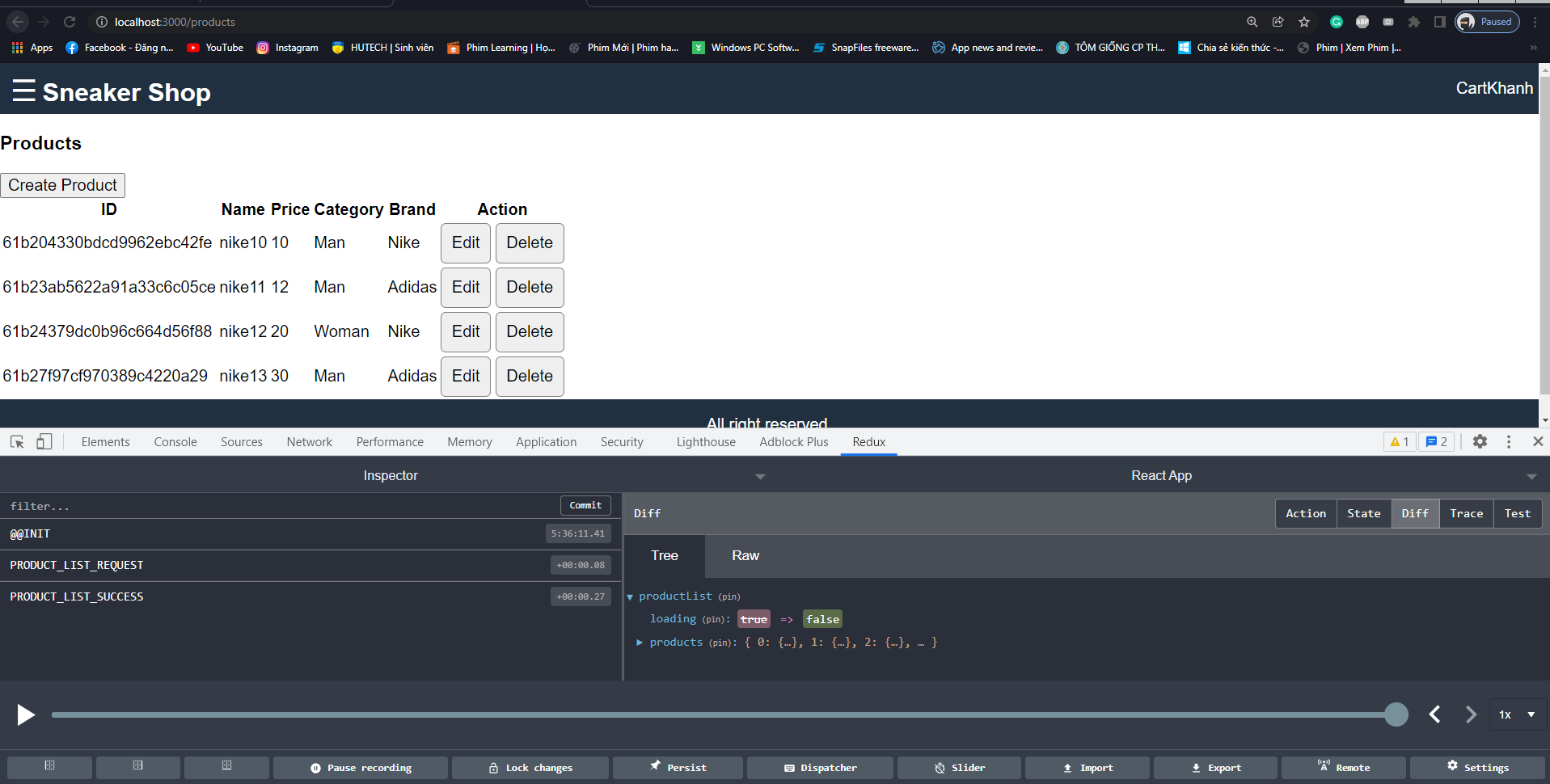


Dùng “axios” thay cho “Axios”

**- Result: to checking ... 4:20:17**



**- Result: to checking ... 4:22:34**



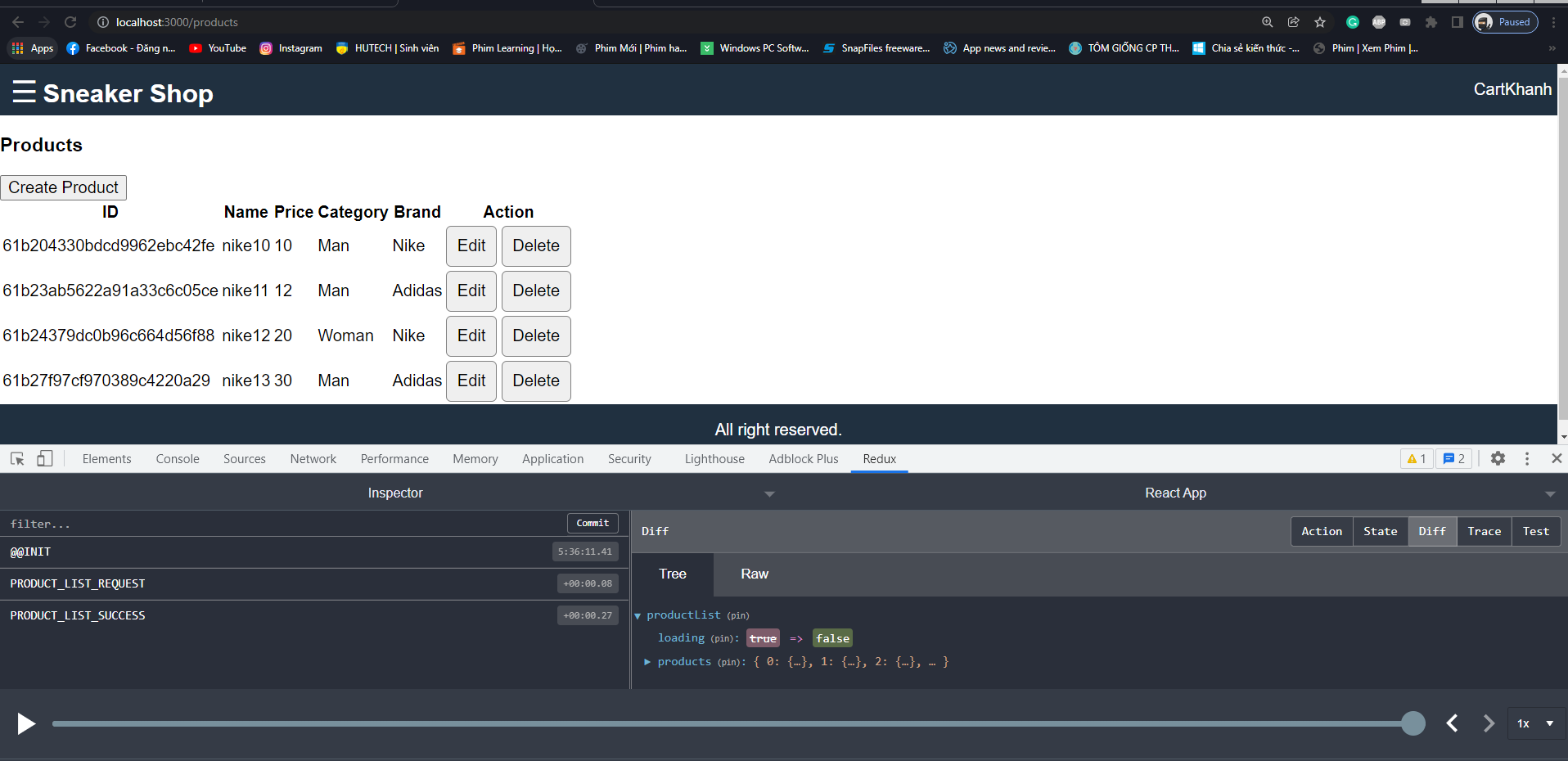
Display the products included in the database (MongoDB)

**(Error 2 ... 4:22:34) Nếu làm như trong video mà nó không hiện ra list product (các product trong database).**

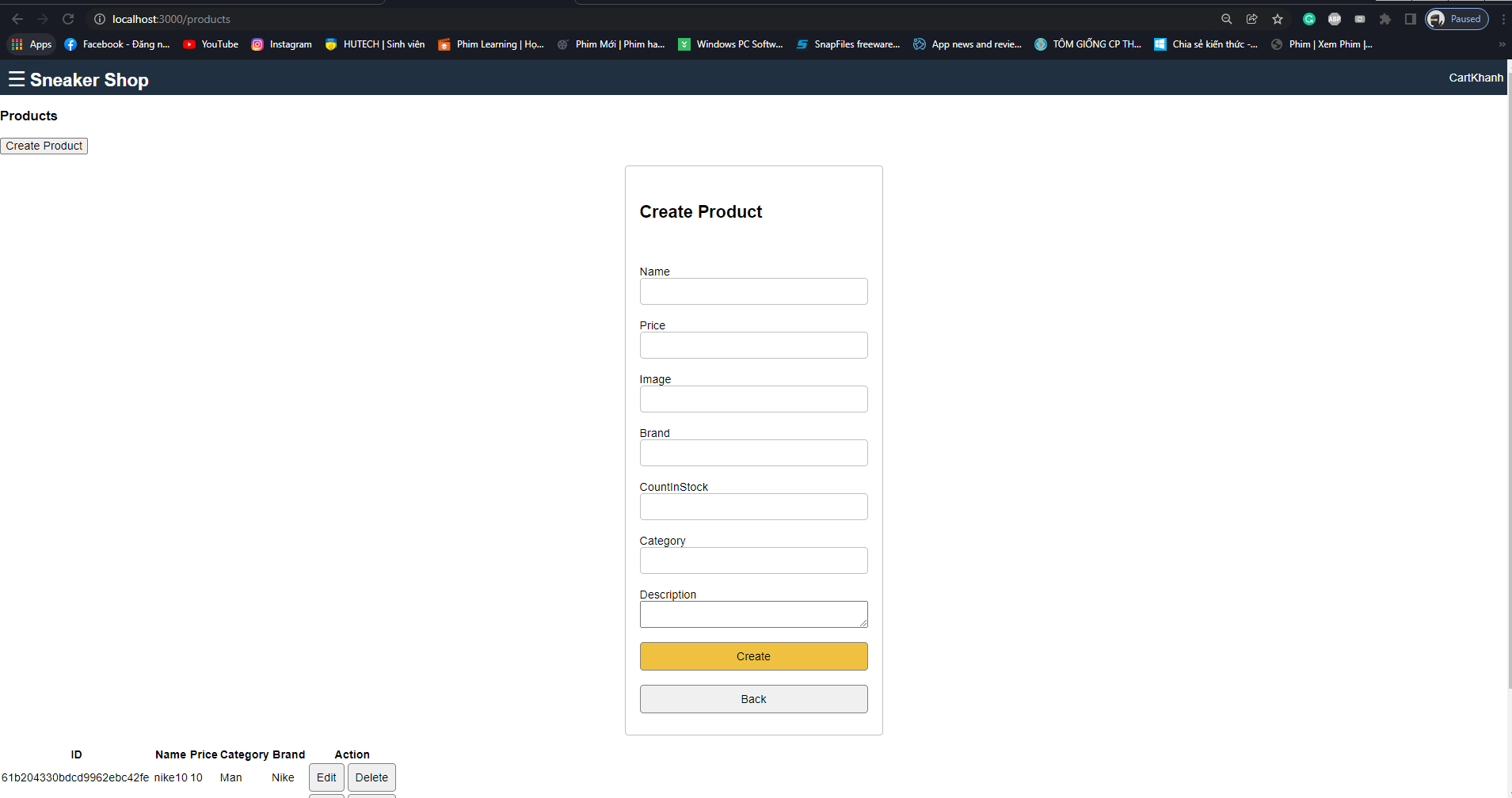
**(Solution (Error 2)):**

**Ở lỗi này là do trong video hướng dẫn tác giả đã cắt bớt khúc quan trọng làm code mình code bị sai, cách fix là download source code của người ta về rồi soi xem khác nhau cái gì ( ở lúc fix được lỗi này là khác nhau ở file ProductsScreen.js )**

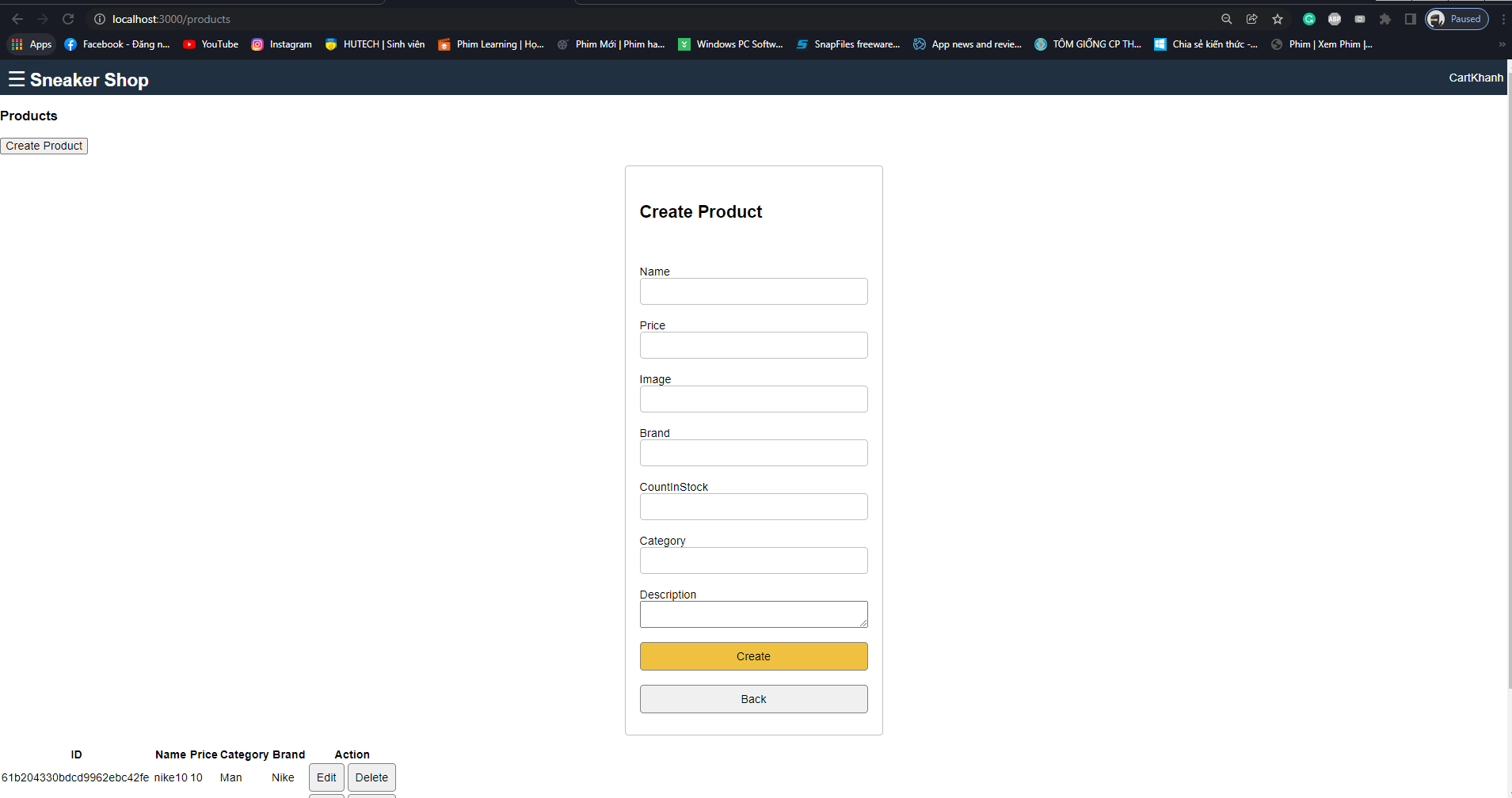
**- Result: to checking ... 4:23:08**



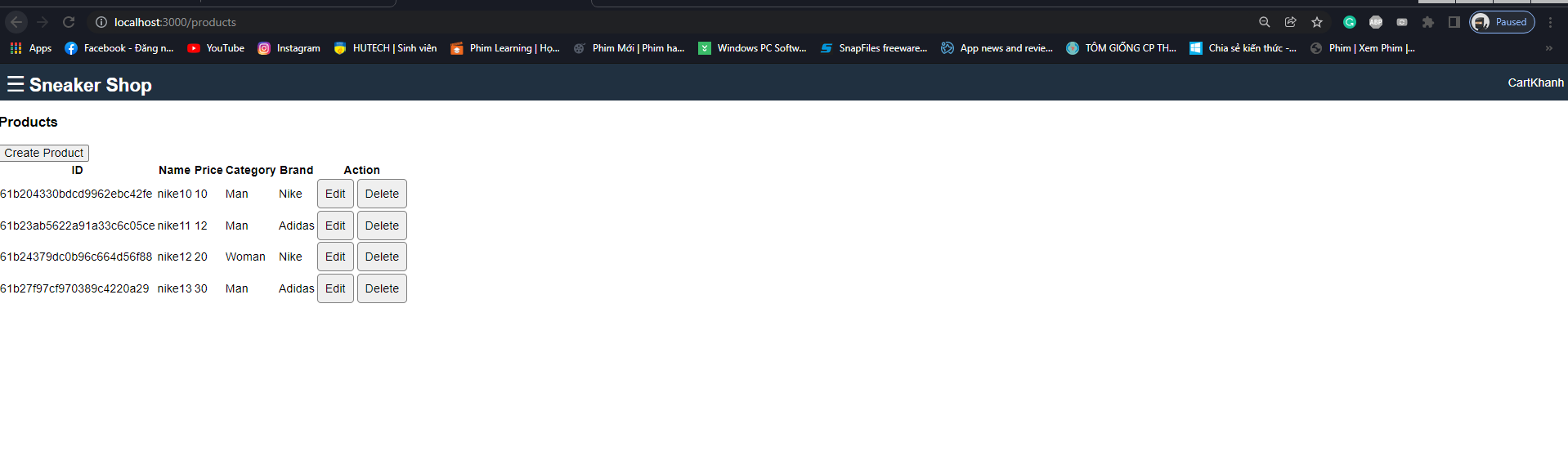
Click “Create Product” button



**Image 1,2 : Step 1 and result at Step 1**

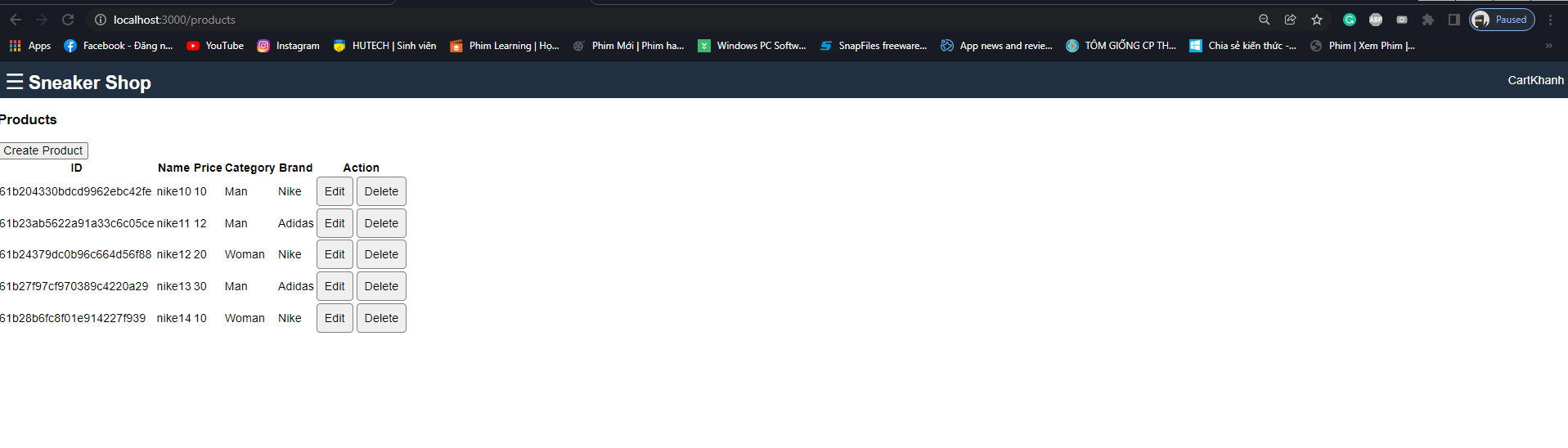


Click “Back” button

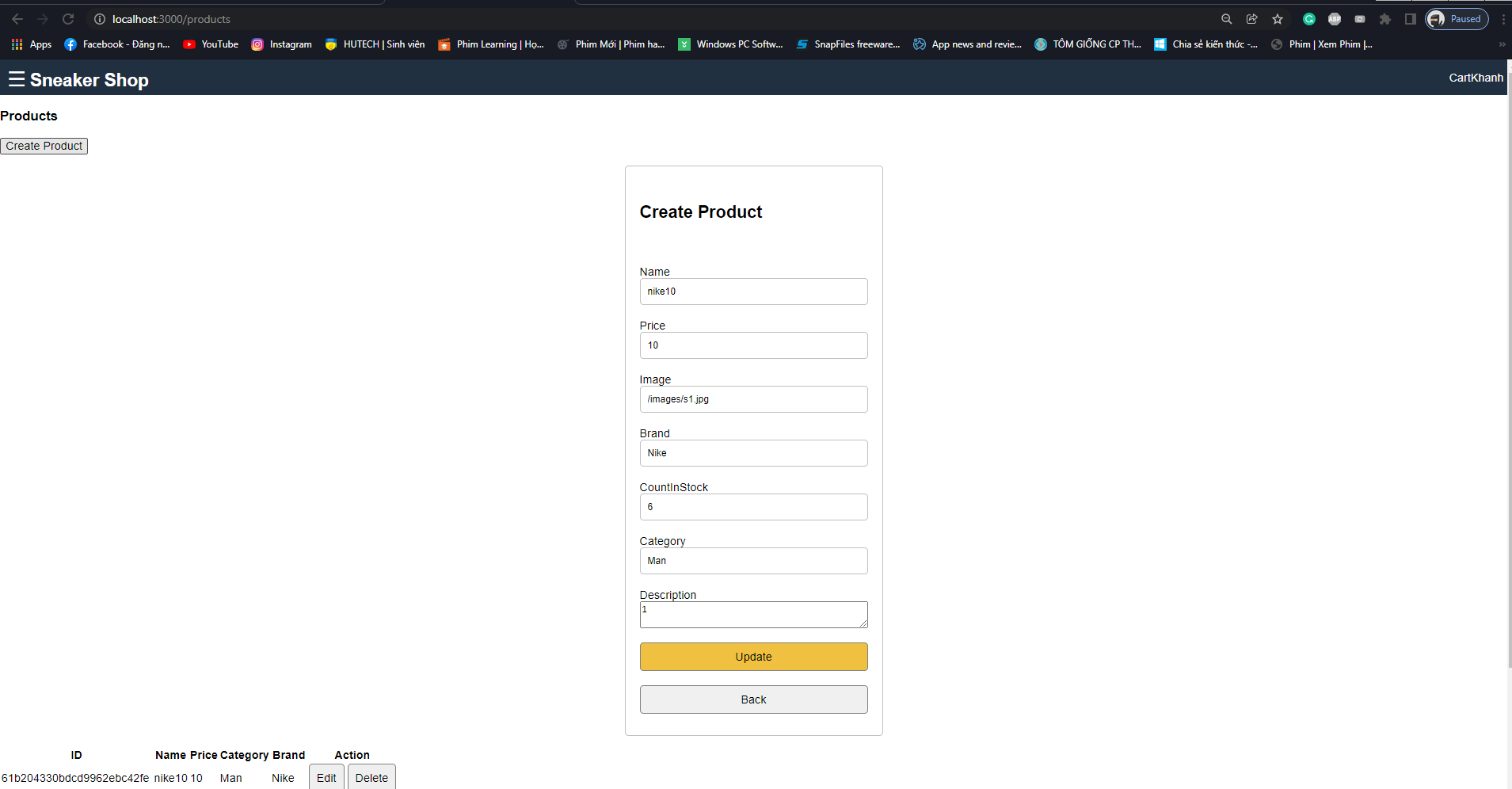


**Image 3,4 : Step 2 and result at Step 2**

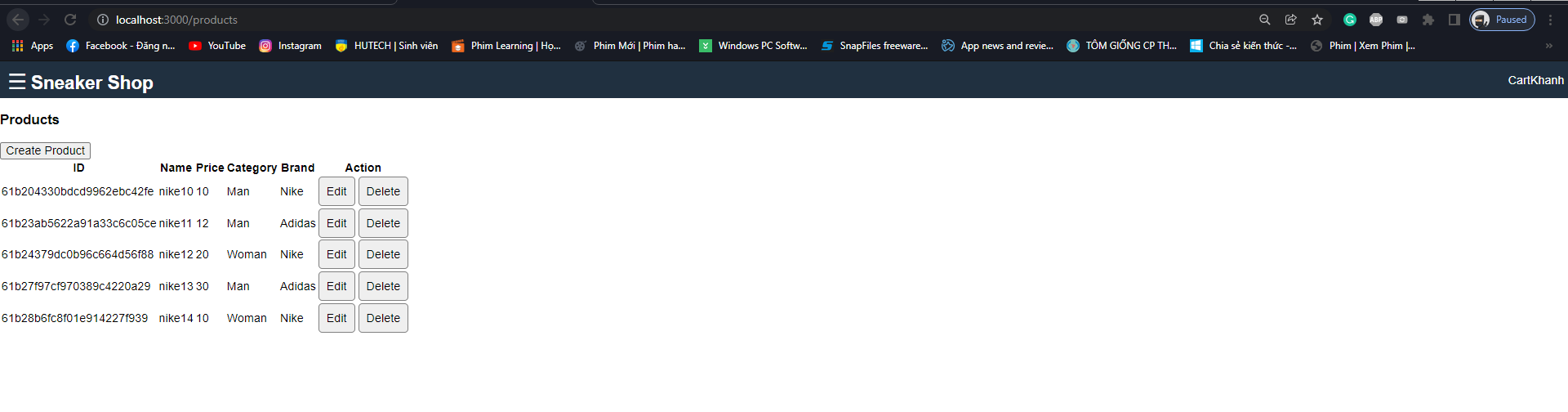
**- Result: to checking ... 4:24:37**



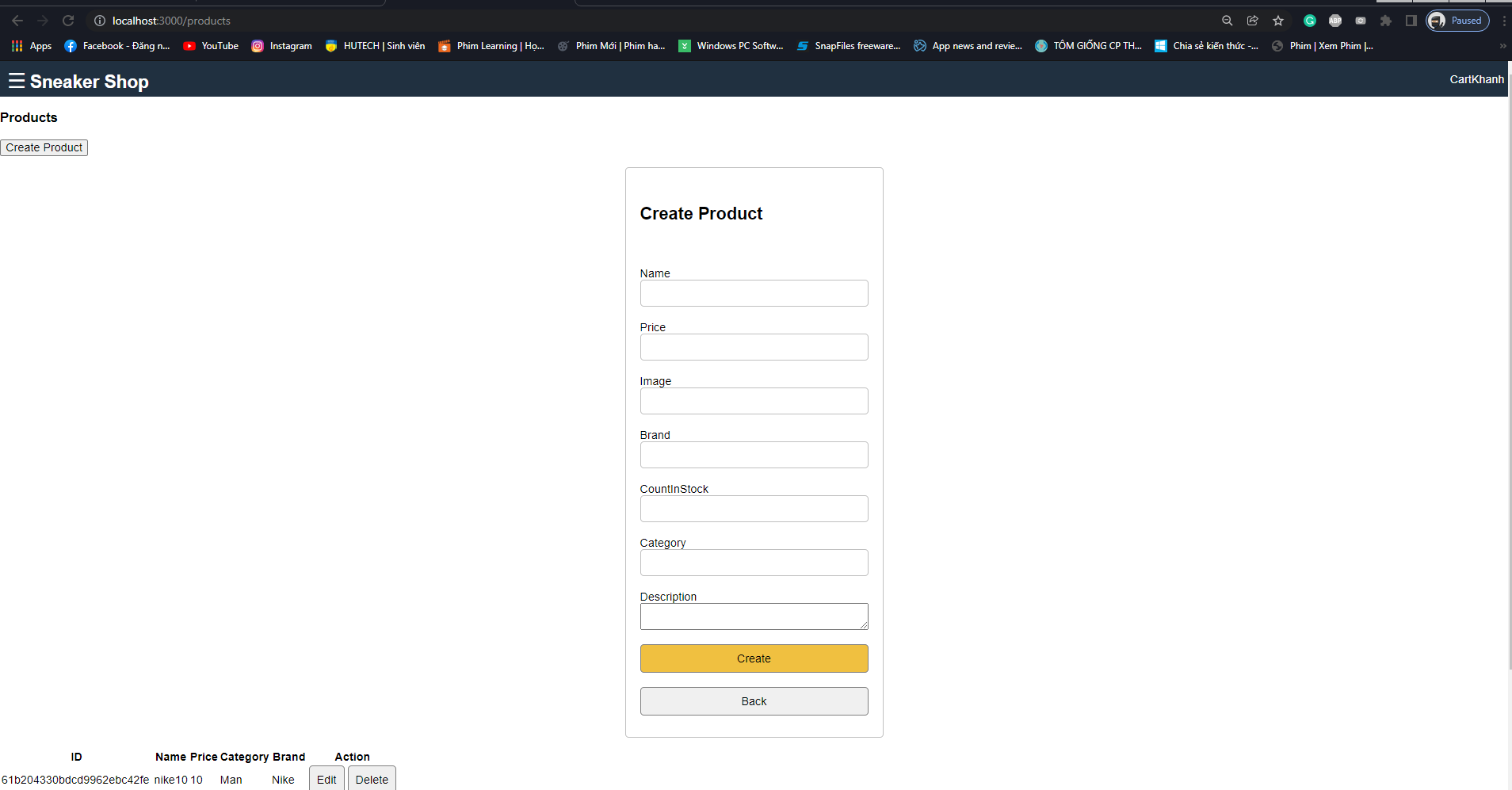
Press the “Edit” button at any product



**Image 1,2 : Step 1 and result at Step 1**

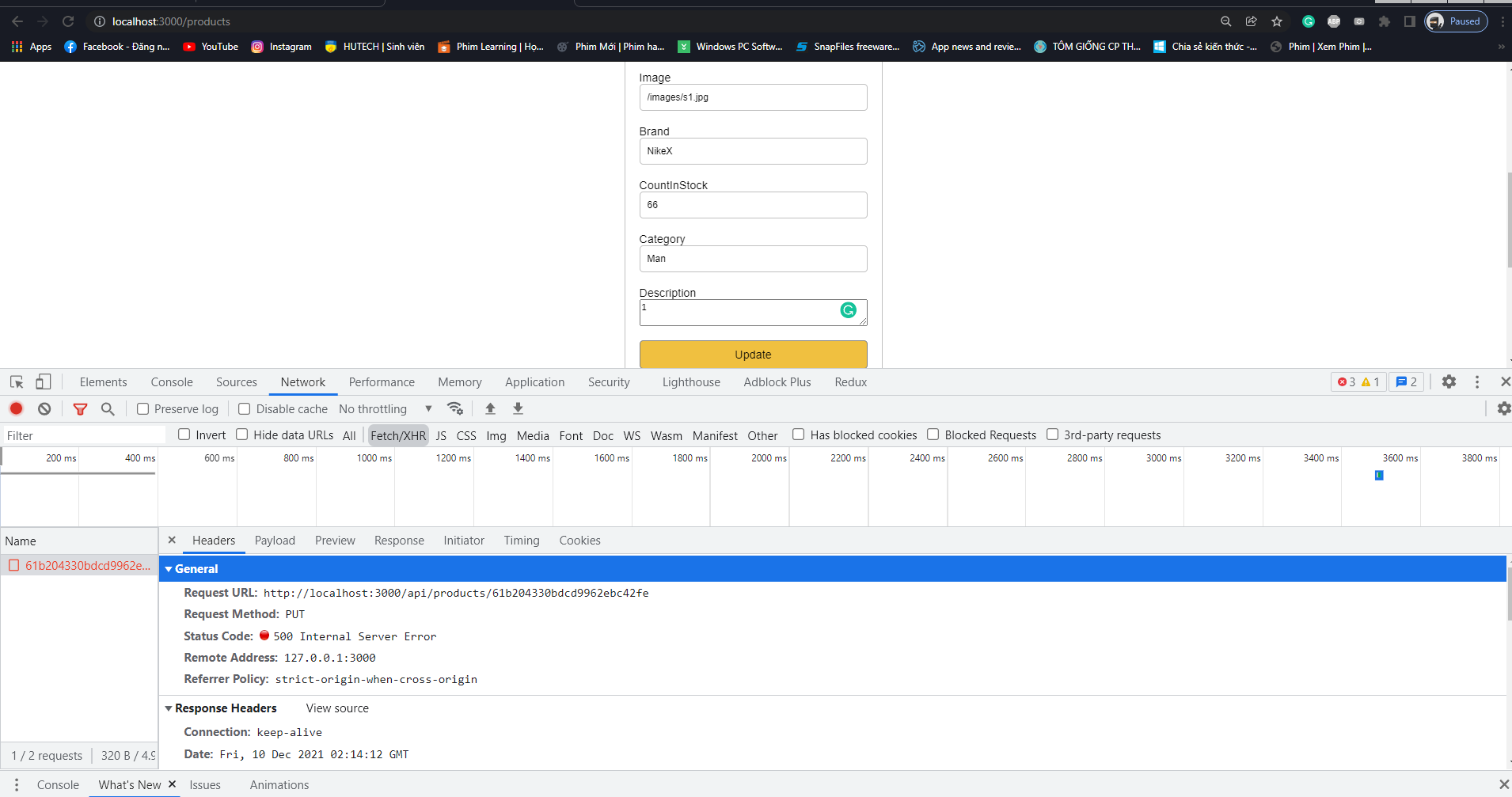


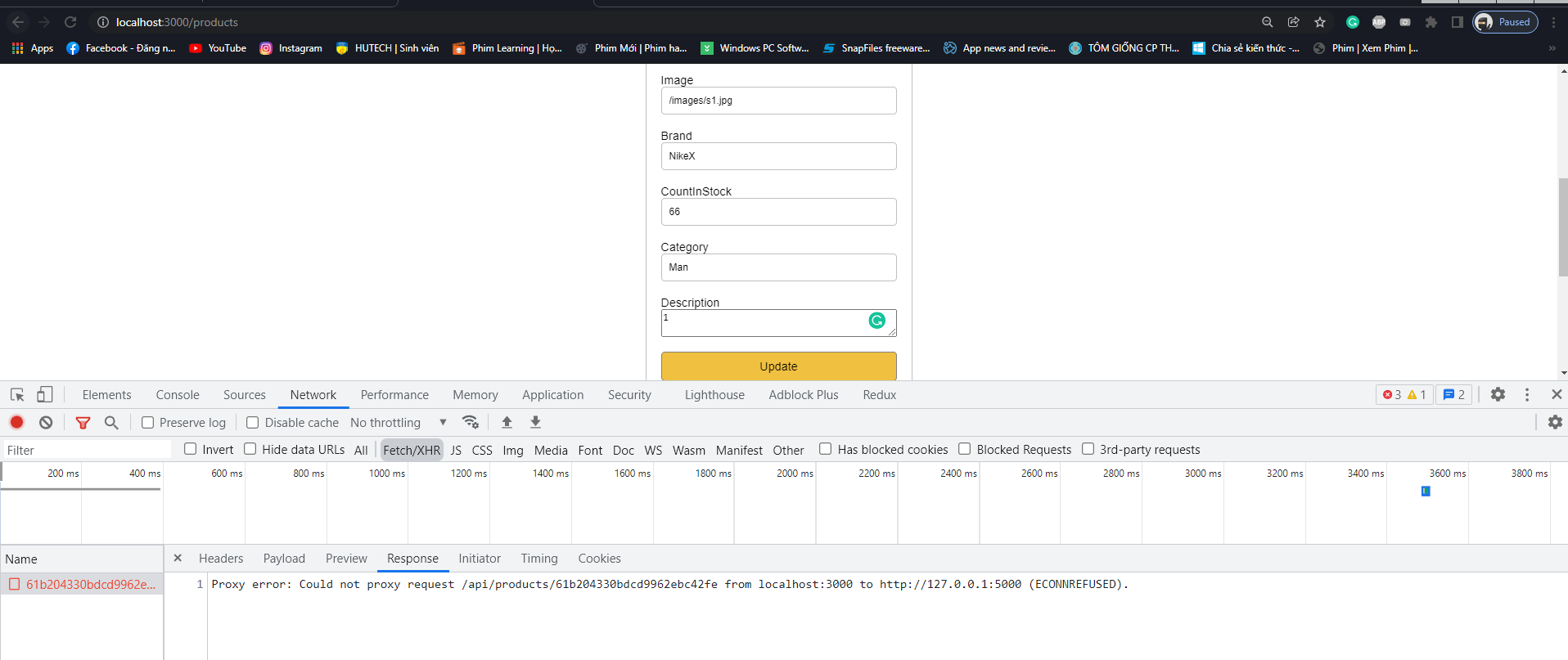
Press the “Create” button

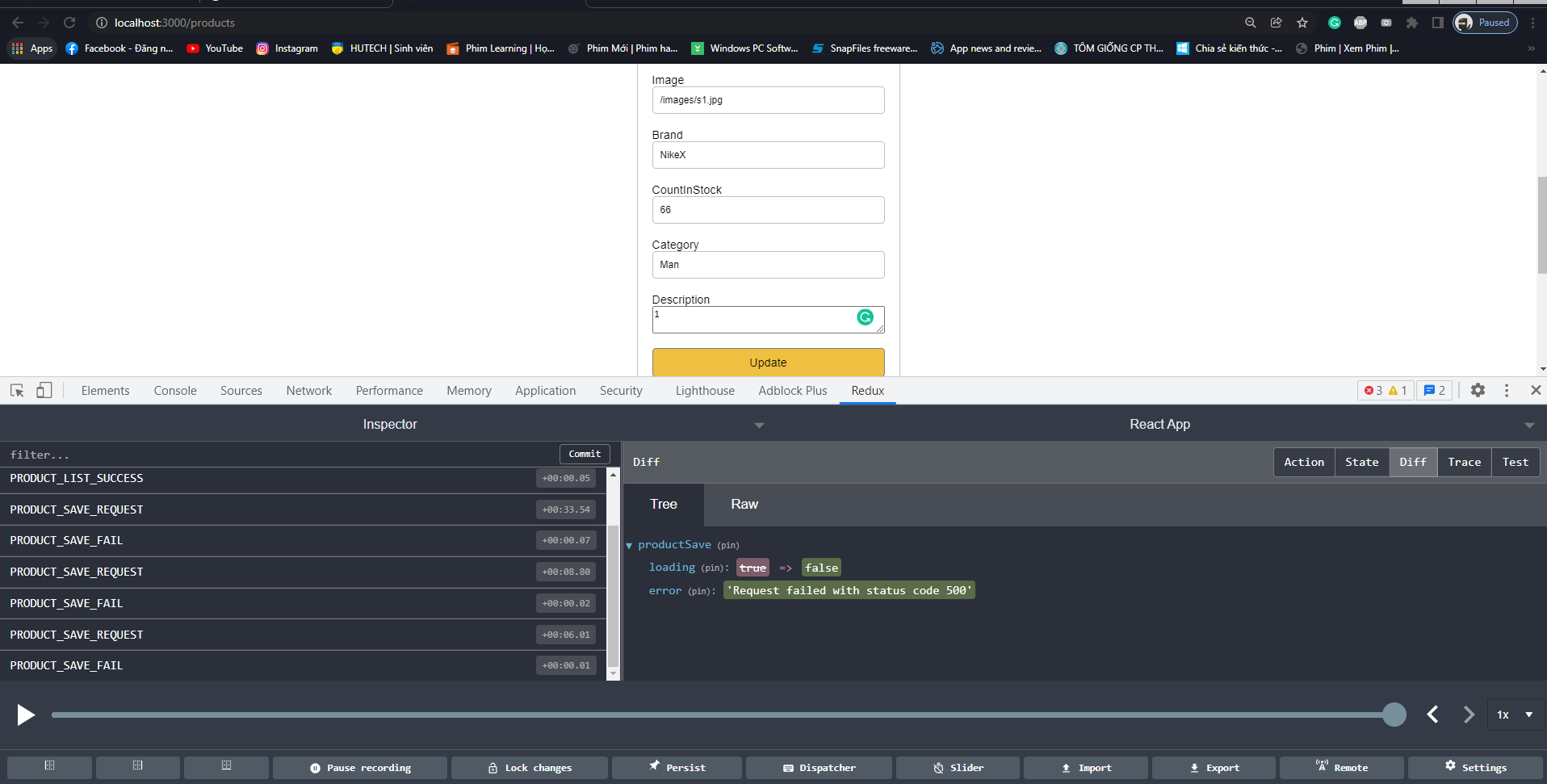


**Image 3,4 : Step 2 and result at Step 2**

**(Error 3 ... 4:27:30) Nếu làm như trong video mà tại lúc chỉnh sửa thông tin product sau khi nhấn vào nút “Edit” của 1 product bất kỳ mà nó bắn ra lỗi 500 chứ nó không update lại thông tin vừa chỉnh sửa.**



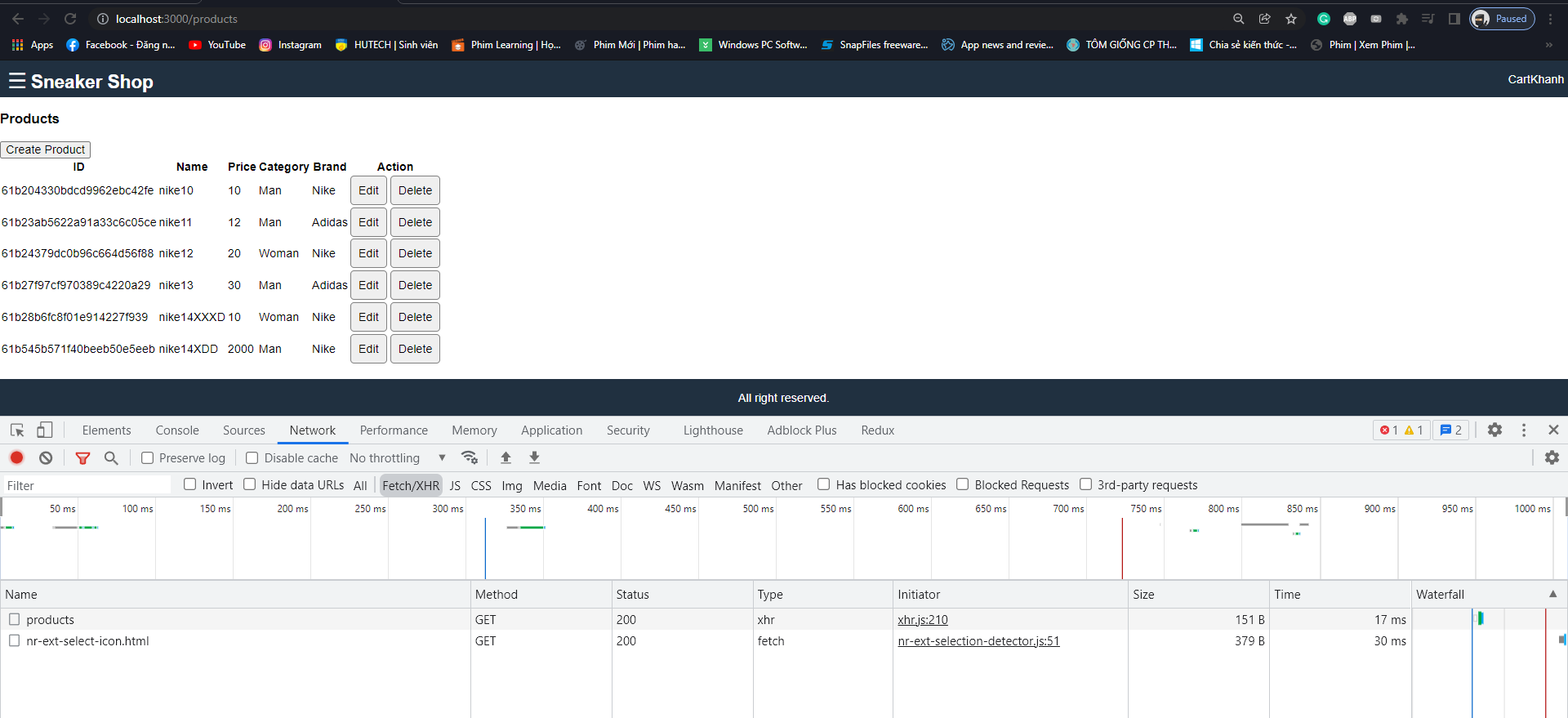




**(Solution (Error 3)):**

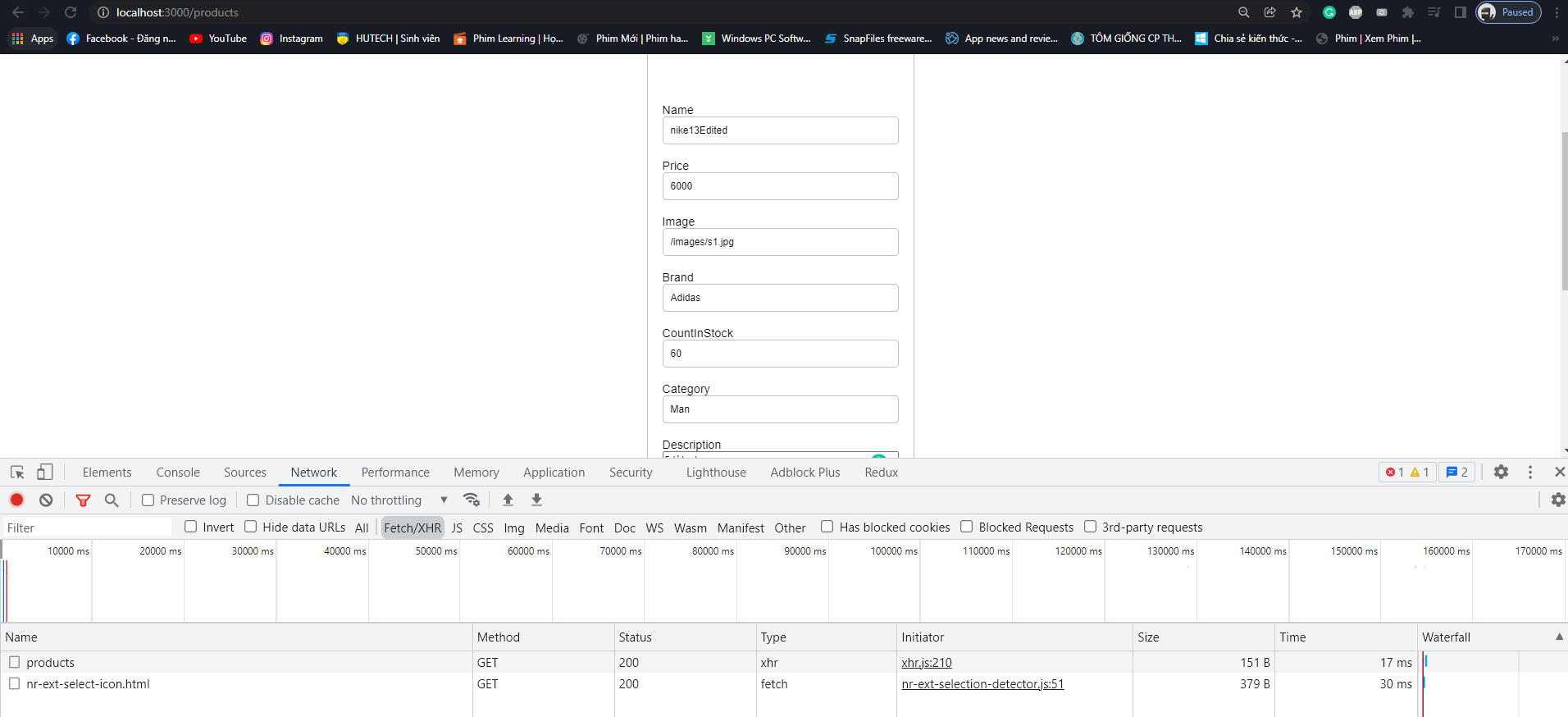
**Làm tiếp đến phút 4:28:45 vì khúc đó nó có update lại code của nó.**

**- Result: to checking ... 4:28:46**



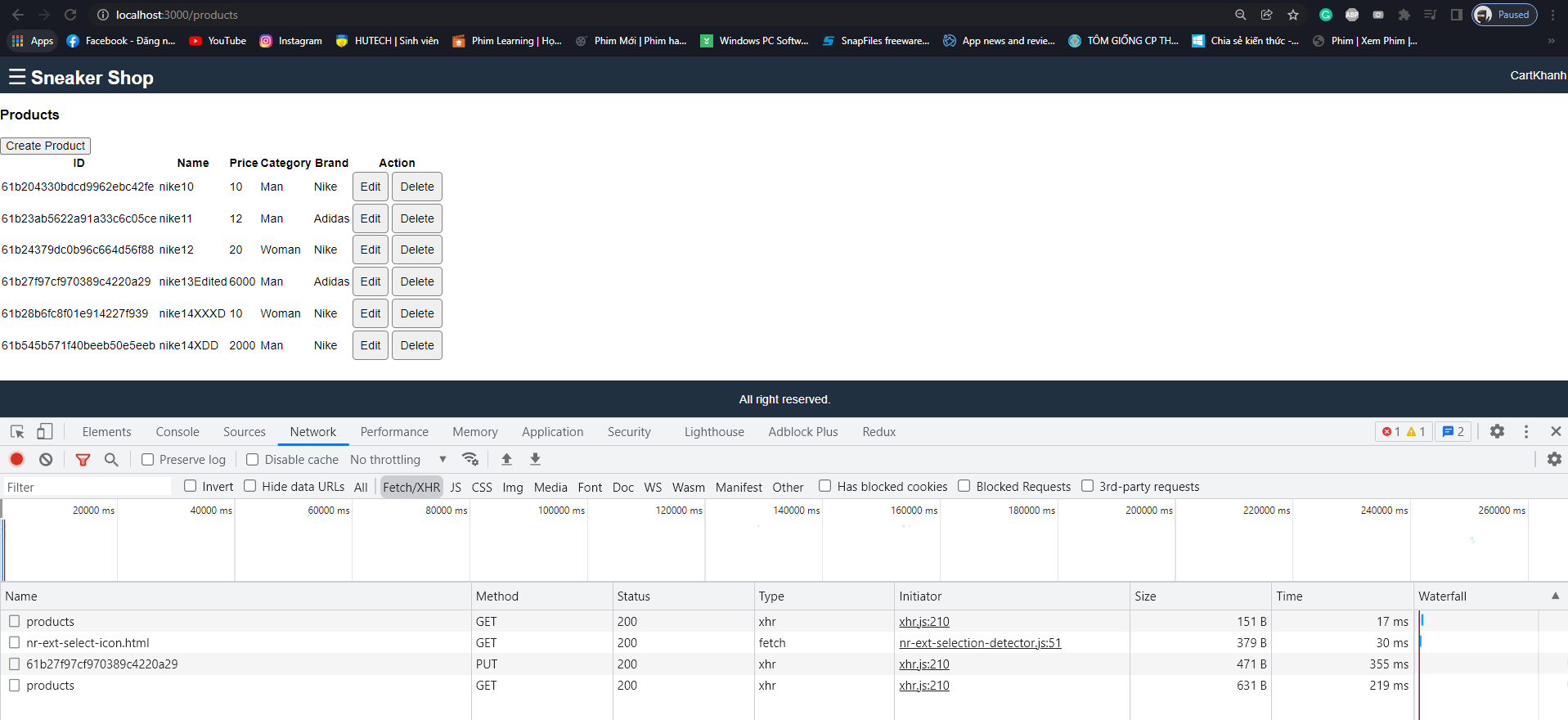
Press the “Edit” button at any product

**Image 1 : Step 1**

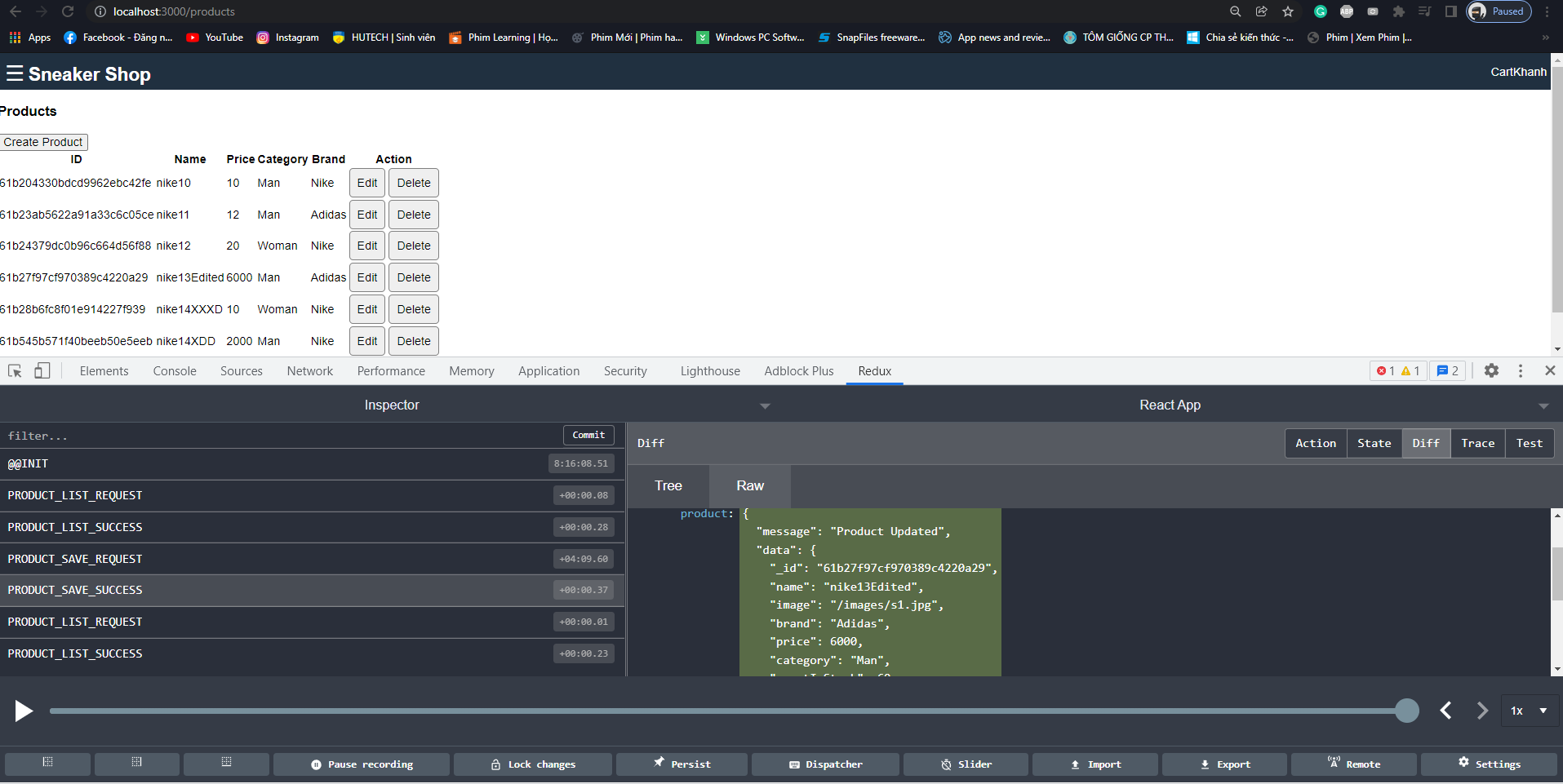


Editing Product’s info and press “Update” button

**Image 2 : Step 2**



I can see product’s info is changed



**Image 3,4 : Result**

**- Create “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/actions/userActions.js” file, add the following code:**

**(Error 3 ... 3:55:26) Nếu làm như trong video mà tại lúc test đăng ký, khi nhập hết các field và nhấn nút đăng ký mà nó bắn ra lỗi 500.**

**(Solution (Error 3)):**

**- Result: to checking ... 3:55:26**

2