**(Nếu project làm từng bước như video mà không chạy như trong video thì nên thử start lại server và frontend)**

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

import React from 'react';

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';

function App() {

  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>

                <a href="signin.html">Sign In</a>

            </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="/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;

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

import React, { useEffect } from "react";

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

import { addToCart, removeFromCart } from "../actions/cartActions";

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

function CartScreen(props) {

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

    const { cartItems } = cart;

    const productId = props.match.params.id;

    const quantity = props.location.search? Number(props.location.search.split("=")[1]):1;

    const dispatch = useDispatch();

    const removeFromCartHandler = (productId) => {

        dispatch(removeFromCart(productId));

    }

    useEffect(() => {

        if(productId){

            dispatch(addToCart(productId, quantity));

        }

    }, [])

    const checkoutHandler = () => {

        props.history.push("/signin?redirect=shipping");

    }

    return <div className="cart">

        <div className="cart-list">

            <ul className="cart-list-container">

                <li>

                    <h3>

                        Shopping Cart

                    </h3>

                    <div>

                        Price

                    </div>

                </li>

                {

                    cartItems.length === 0 ?

                    <div>

                        Cart is empty

                    </div>

                    :

                    cartItems.map(item =>

                        <li>

                            <div className="cart-image">

                                <img src={item.image} alt="product" />

                            </div>

                            <div className="cart-name">

                                <div>

                                    <Link to={"/product/" + item.product }>

                                        {item.name}

                                    </Link>

                                </div>

                                <div>

                                    Quantity:

                                    <select value={item.quantity} onChange={(e) => dispatch(addToCart(item.product, e.target.value))}>

                                        {[...Array(item.countInStock).keys()].map(x =>

                                            <option key={x+1} value={x+1}>{x+1}</option>

                                        )}

                                    </select>

                                    <button type="button" className="button" onClick={() => removeFromCartHandler(item.product)} >

                                        Delete

                                    </button>

                                </div>

                            </div>

                            <div className="cart-price">

                                ${item.price \* item.quantity}

                            </div>

                        </li>

                    )

                }

            </ul>

        </div>

        <div className="cart-action">

            <h3>

                Subtotal ( {cartItems.reduce((a, c) => a + c.quantity, 0)} items )

                :

                $ {cartItems.reduce((a, c) => a + c.price \* c.quantity, 0)}

            </h3>

            <button onClick={checkoutHandler} className="button primary full-width" disabled={cartItems.length === 0}>

                Proceed to Checkout

            </button>

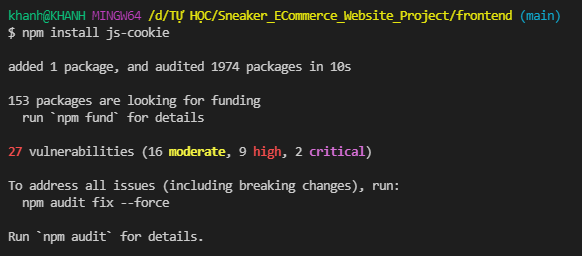
        </div>

    </div>

}

export default CartScreen;

**- Install “js-cookie”**



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

import Axios from "axios";

// import Cookies from "js-cookie";

import { CART\_ADD\_ITEM, CART\_REMOVE\_ITEM } from "../constants/cartConstants";

// Nếu muốn lưu với Cookie (Server Side)

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

const addToCart = (productId, quantity) => async (dispatch, getState) => {

    try {

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

        dispatch({

            type: CART\_ADD\_ITEM, payload: {

                product: data.\_id,

                name: data.name,

                image: data.image,

                price: data.price,

                countInStock: data.countInStock,

                quantity

            }

        });

        // Nếu muốn lưu với Cookie (Server Side)

        const { cart: { cartItems } } = getState();

        Cookies.set("cartItems", JSON.stringify(cartItems));

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

        // localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems));

    } catch (error) {

    }

}

const removeFromCart = (productId) => (dispatch, getState) => {

    dispatch({type: CART\_REMOVE\_ITEM, payload: productId});

    // Nếu muốn lưu với Cookie (Server Side)

    const { cart: { cartItems } } = getState();

    Cookies.set("cartItems", JSON.stringify(cartItems));

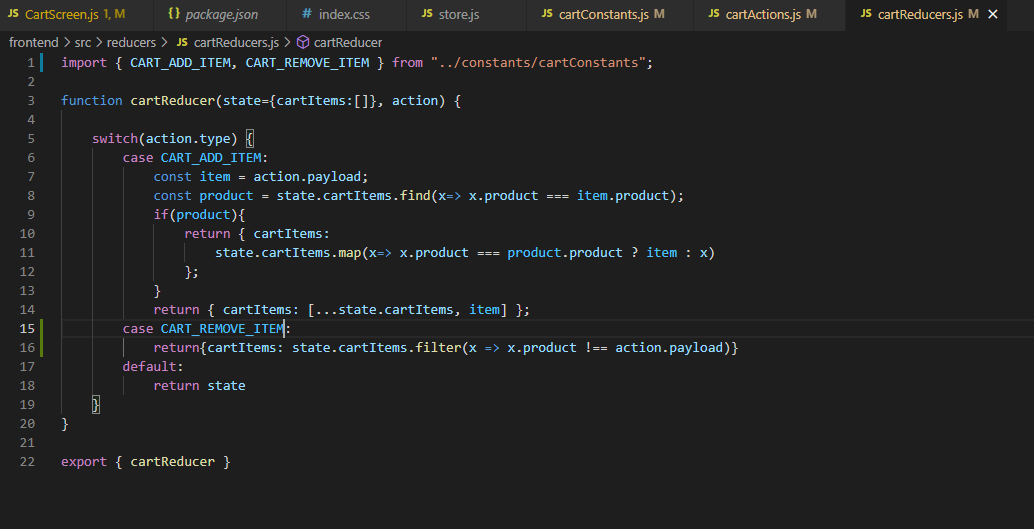
    // Nếu muốn lưu với Local Storage (Client Side)

    // localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems));

}

export { addToCart, removeFromCart }

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



**- 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 } from './reducers/productReducers';

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

// 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')) : [] } };

// 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

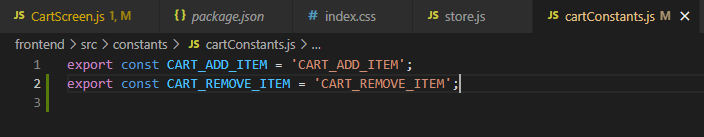
})

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

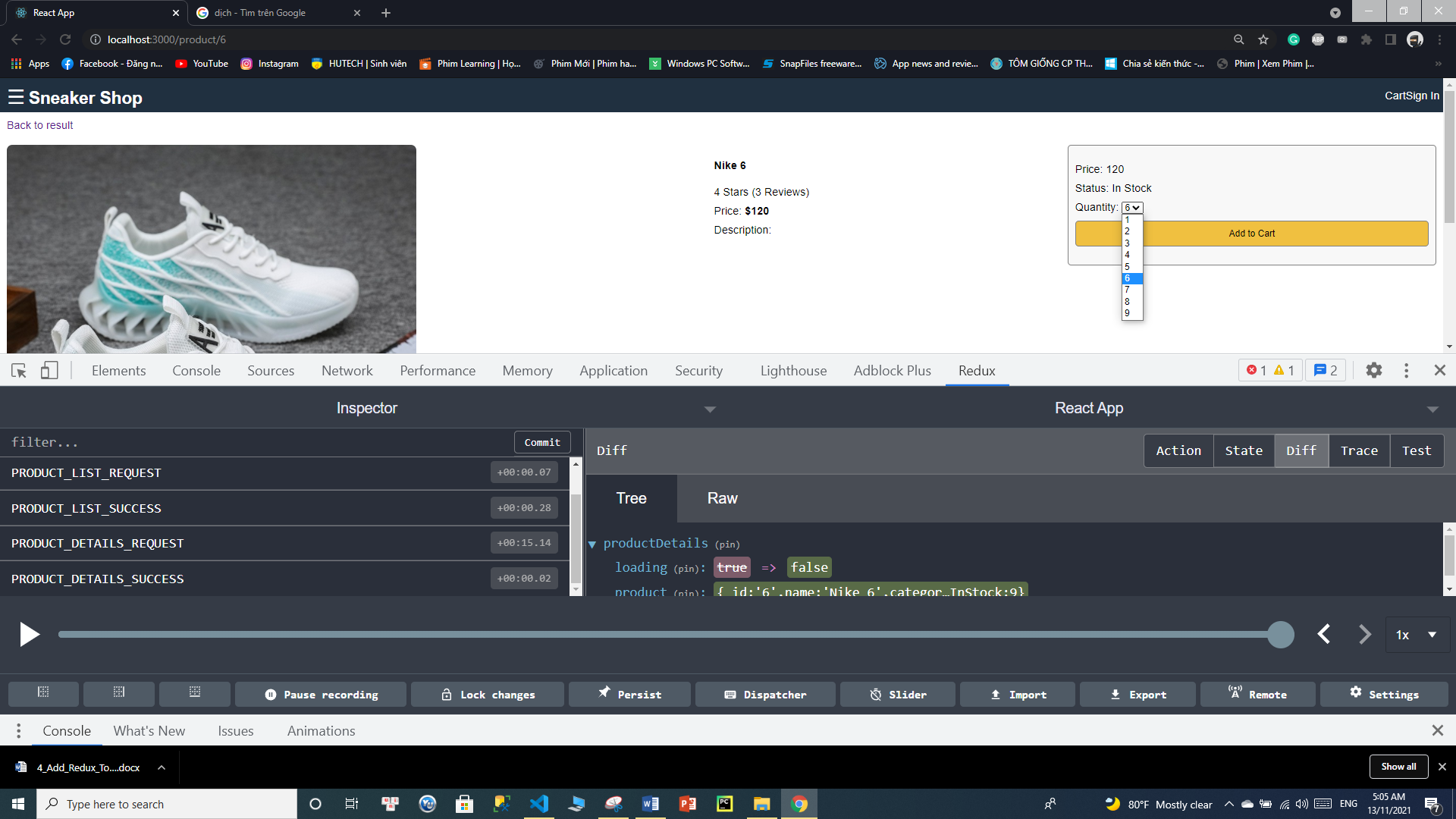
const store = createStore(reducer, initialState, composeEnhancer(applyMiddleware(thunk)));

export default store;

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

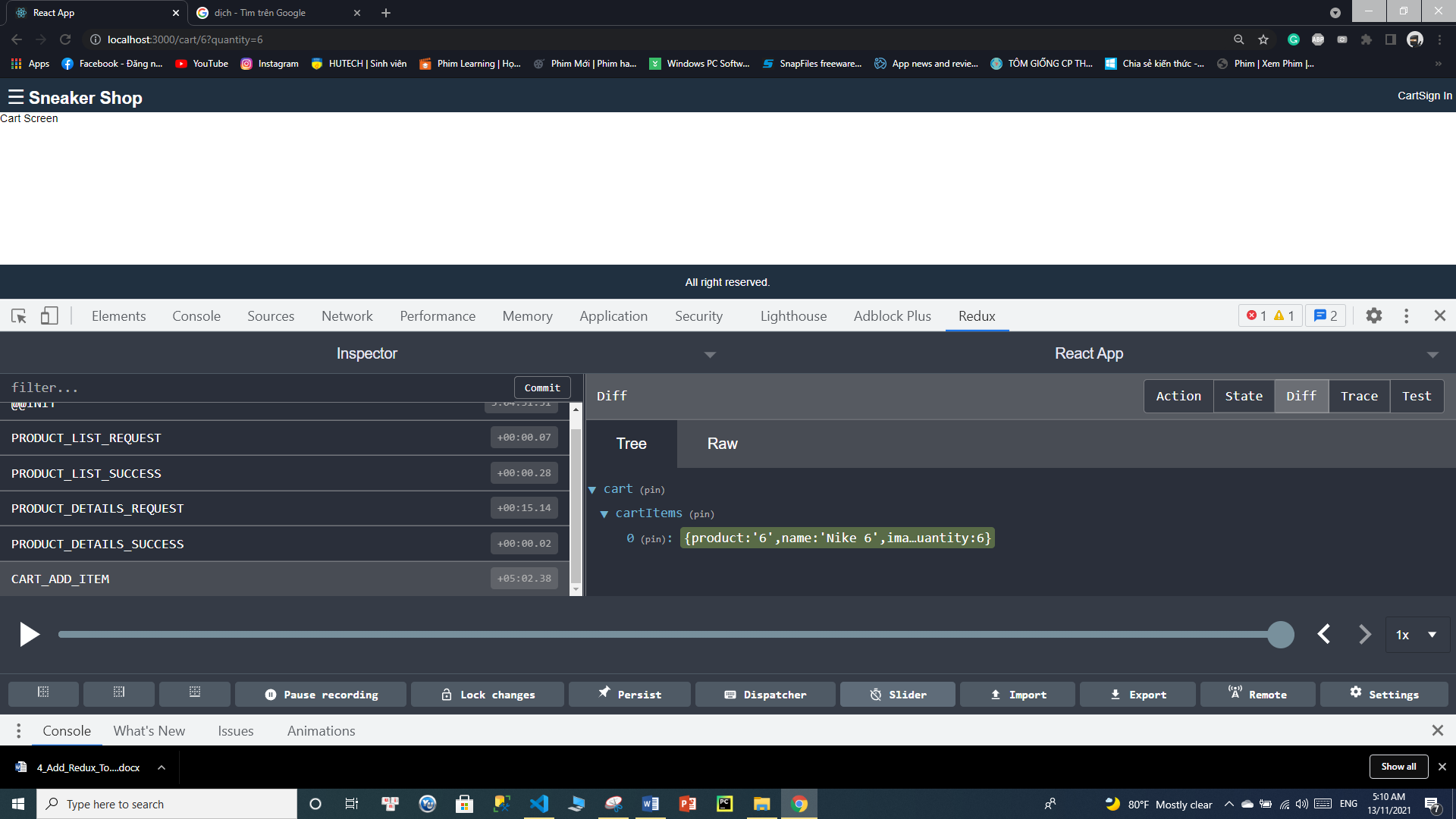


**- Run project: To Checking (First)**



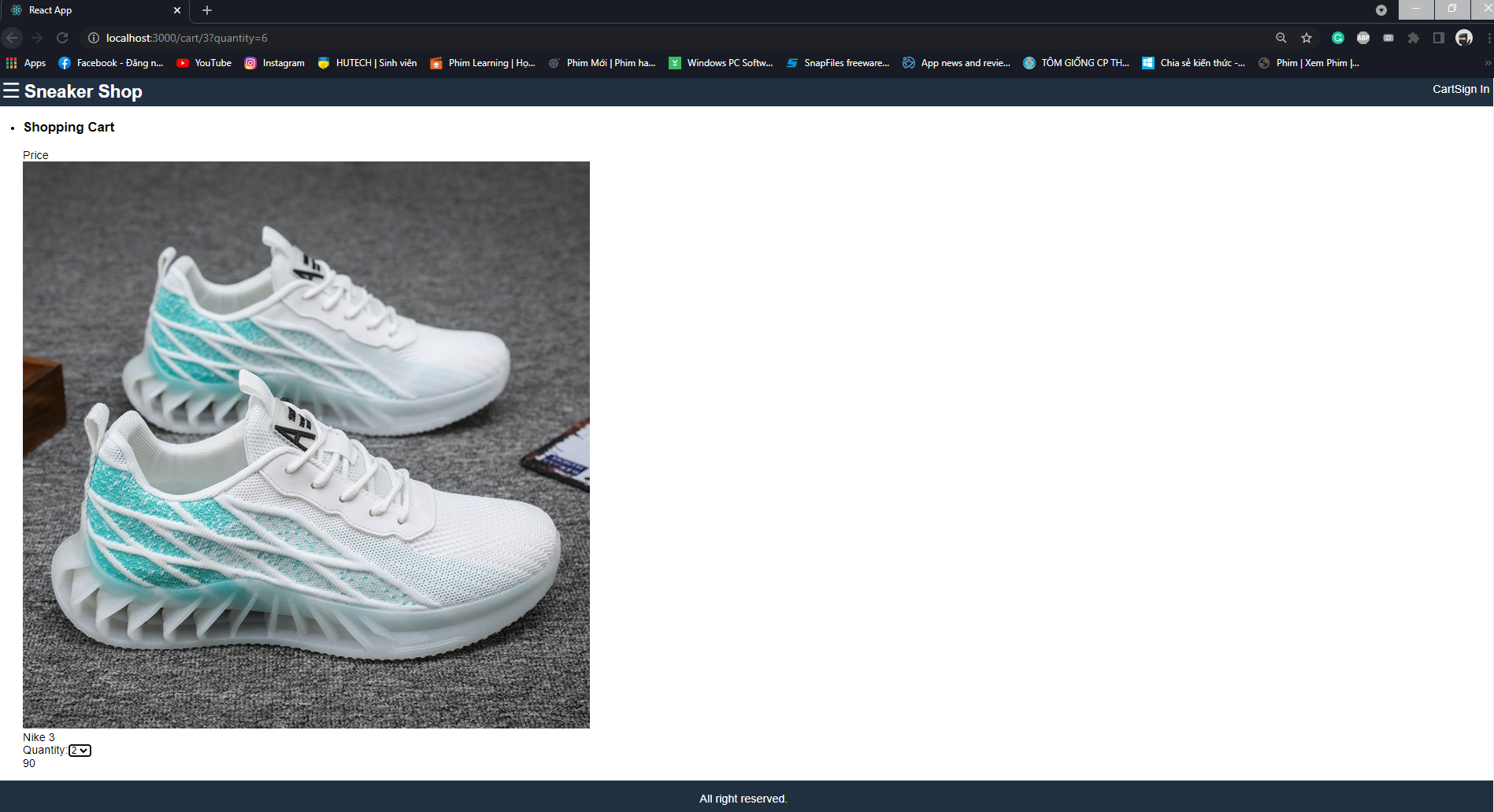
Select quantity and press “Add to Cart” button.

**Image 1: Step 1**

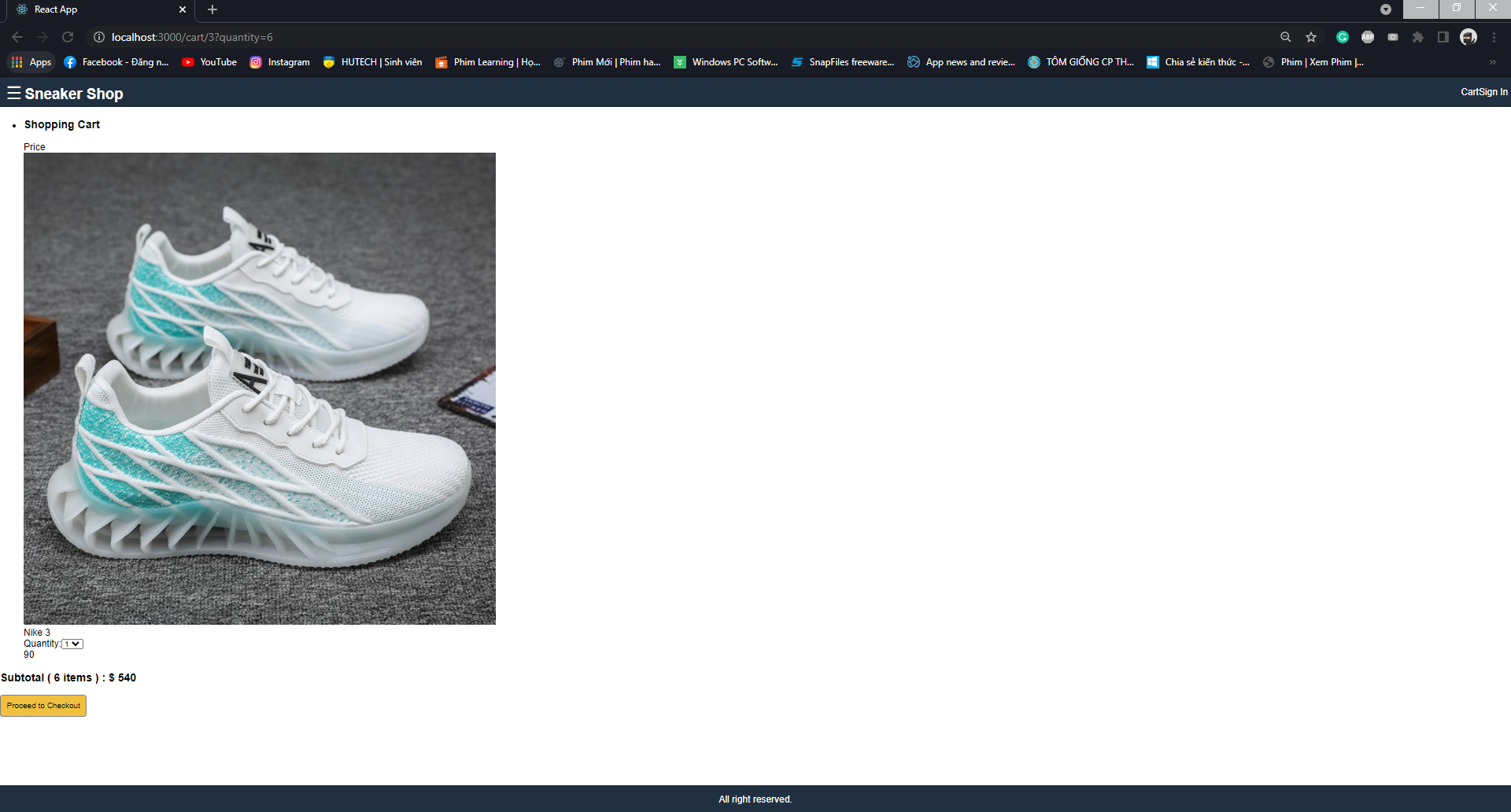


**Image 2: Result**

**- Run project: To Checking (Second) ... 2:48:14**



**- Run project: To Checking (Third) ... 2:50:33**



**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/index.css” file, css for Cart:**

/\* Shopping Cart \*/

.cart {

  display: flex;

  flex-wrap: wrap;

  margin: 1rem;

  align-items: flex-start;

}

.cart-list {

  flex: 3 1 60rem;

}

.cart-action {

  flex: 1 1 20rem;

  background-color: #f8f8f8;

  border-radius: 0.5rem;

  padding: 1rem;

}

.cart-list-container {

  padding: 0;

  list-style-type: none;

  padding: 1rem;

}

.cart-list-container li {

  display: flex;

  justify-content: space-between;

  align-items: flex-end;

  padding-bottom: 1rem;

  margin-bottom: 1rem;

  border-bottom: 0.1rem #808080 solid;

}

.cart-list-container li img {

  max-width: 10rem;

  max-height: 10rem;

}

.cart-list-container li:first-child {

  display: flex;

  justify-content: space-between;

  align-items: flex-end;

}

.cart-image {

  flex: 1 1;

}

.cart-name {

  flex: 8 1;

}

.cart-price {

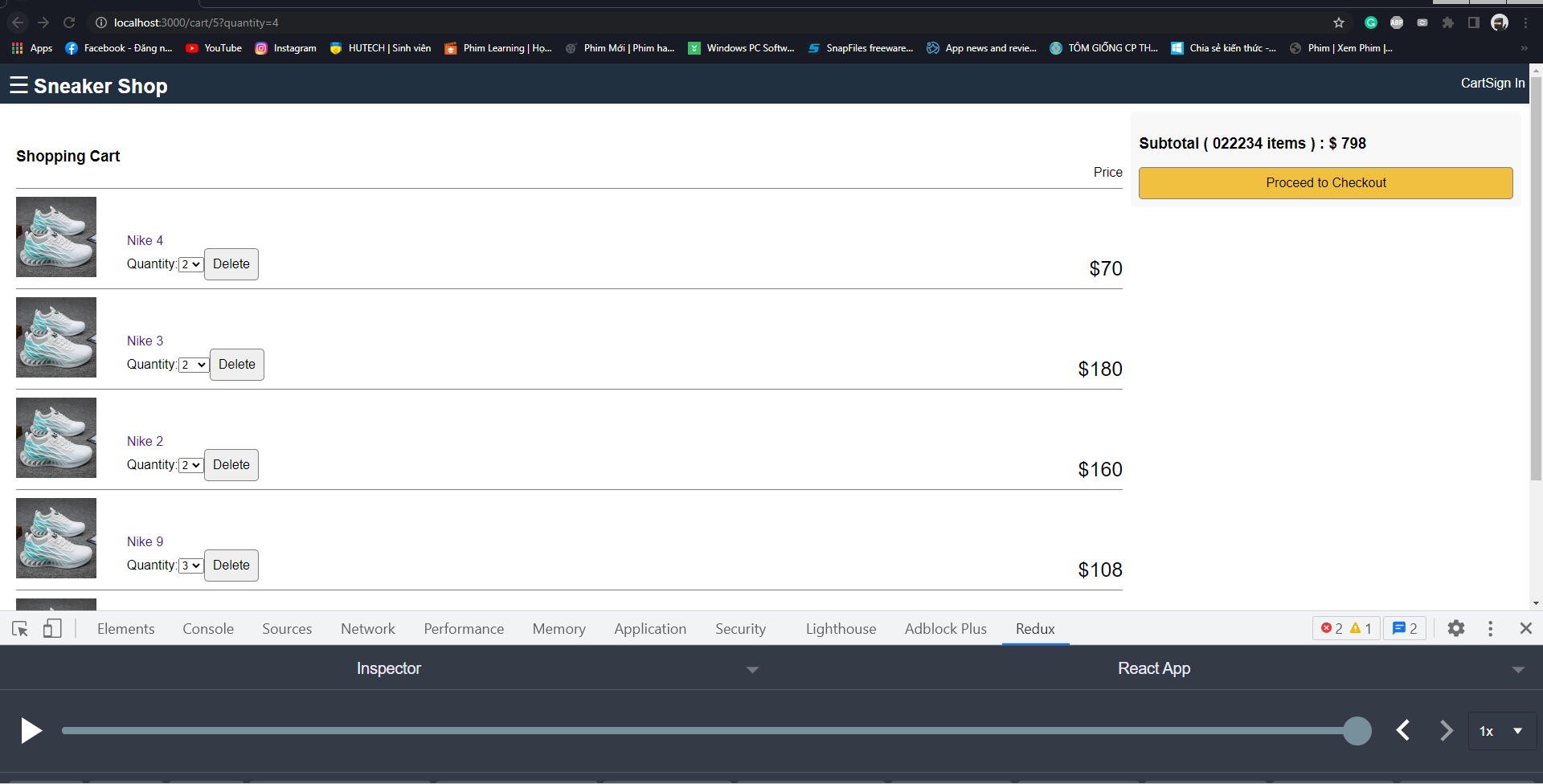
  flex: 1 1;

  font-size: 2.5rem;

  text-align: right;

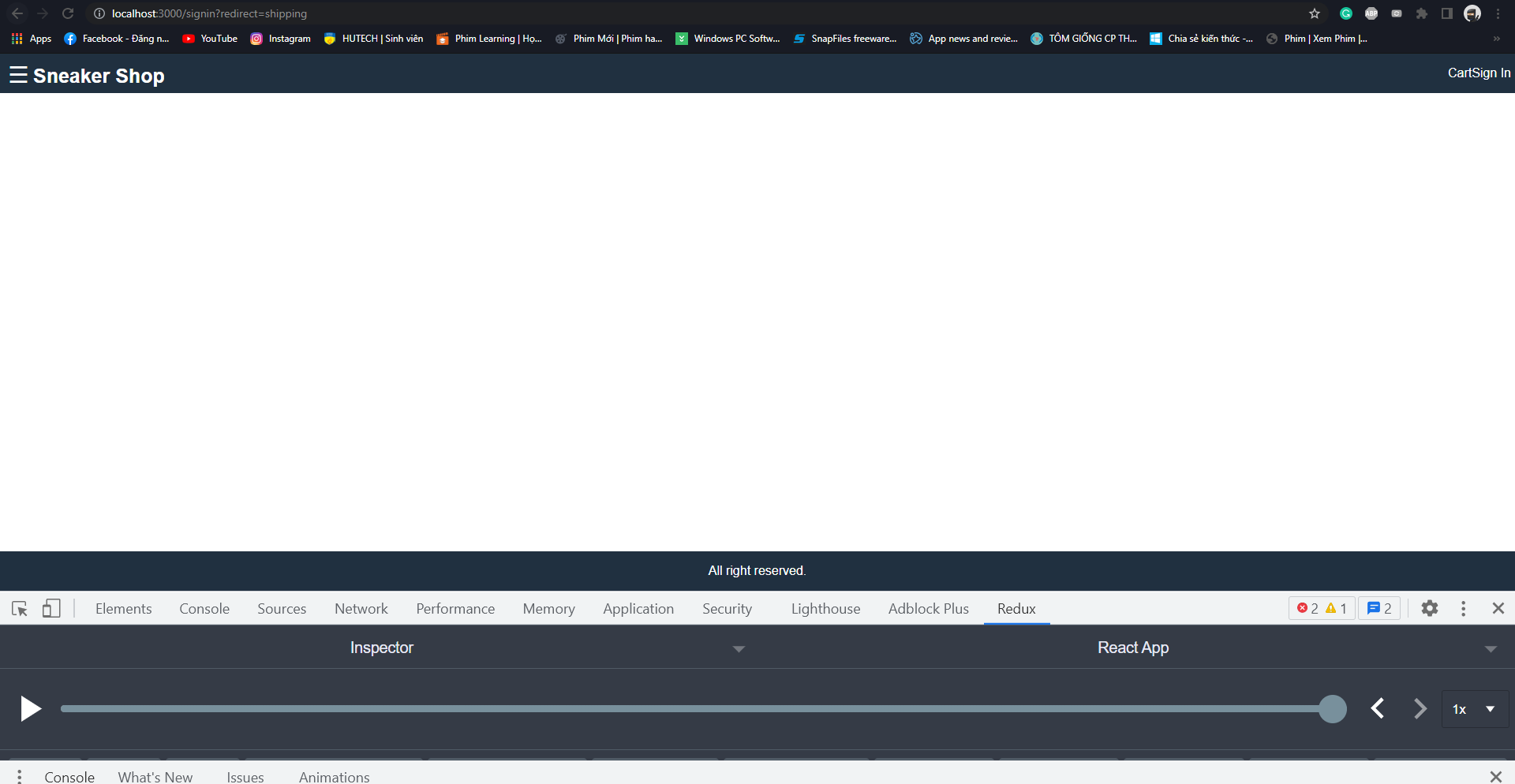
}

**- Run project: To Checking (4th) ... 3:03:00**



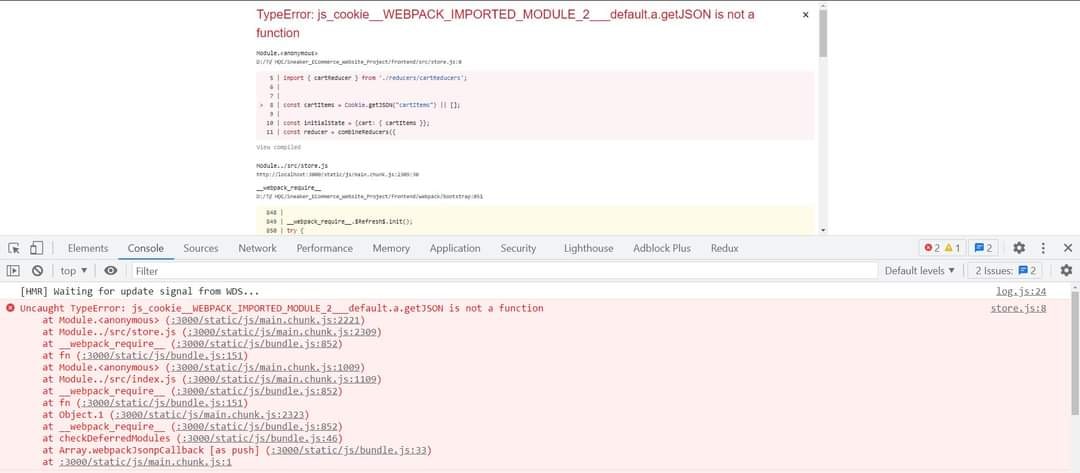
Click here

**Image 1: Step 1**



**Image 2: Result**

**(Error 1) Nếu làm giống như trong clip mà gặp lỗi như sau thì có 2 phương án lựa chọn là “Lưu vào Cookie” (Server Side) hoặc lưu vào “Local Storage” (Client Side):**

****

**(Solution 1 (Error 1)): Save to “Cookie” (Server Side)**

- In **“SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/actions/cartActions.js” file:**

import Axios from "axios";

// import Cookies from "js-cookie";

import { CART\_ADD\_ITEM, CART\_REMOVE\_ITEM } from "../constants/cartConstants";

// Nếu muốn lưu với Cookie (Server Side)

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

const addToCart = (productId, quantity) => async (dispatch, getState) => {

    try {

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

        dispatch({

            type: CART\_ADD\_ITEM, payload: {

                product: data.\_id,

                name: data.name,

                image: data.image,

                price: data.price,

                countInStock: data.countInStock,

                quantity

            }

        });

        // Nếu muốn lưu với Cookie (Server Side)

        const { cart: { cartItems } } = getState();

        Cookies.set("cartItems", JSON.stringify(cartItems));

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

        // localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems));

    } catch (error) {

    }

}

const removeFromCart = (productId) => (dispatch, getState) => {

    dispatch({type: CART\_REMOVE\_ITEM, payload: productId});

    // Nếu muốn lưu với Cookie (Server Side)

    const { cart: { cartItems } } = getState();

    Cookies.set("cartItems", JSON.stringify(cartItems));

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

    // localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems));

}

export { addToCart, removeFromCart }

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/store.js” file:**

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

import thunk from 'redux-thunk';

// import Cookies from 'js-cookie';

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

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

// 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')) : [] } };

// 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

})

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

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

export default store;

**(Solution 2 (Error 1)): Save to “Local Storage” (Client Side)**

- In **“SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/actions/cartActions.js” file:**

import Axios from "axios";

// import Cookies from "js-cookie";

import { CART\_ADD\_ITEM, CART\_REMOVE\_ITEM } from "../constants/cartConstants";

// Nếu muốn lưu với Cookie (Server Side)

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

const addToCart = (productId, quantity) => async (dispatch, getState) => {

    try {

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

        dispatch({

            type: CART\_ADD\_ITEM, payload: {

                product: data.\_id,

                name: data.name,

                image: data.image,

                price: data.price,

                countInStock: data.countInStock,

                quantity

            }

        });

        // Nếu muốn lưu với Cookie (Server Side)

        // const { cart: { cartItems } } = getState();

        // Cookies.set("cartItems", JSON.stringify(cartItems));

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

        localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems));

    } catch (error) {

    }

}

const removeFromCart = (productId) => (dispatch, getState) => {

    dispatch({type: CART\_REMOVE\_ITEM, payload: productId});

    // Nếu muốn lưu với Cookie (Server Side)

    // const { cart: { cartItems } } = getState();

    // Cookies.set("cartItems", JSON.stringify(cartItems));

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

    localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems));

}

export { addToCart, removeFromCart }

**- In “SNEAKER\_ECOMMERCE\_WEBSITE\_PROJECT/frontend/src/store.js” file:**

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

import thunk from 'redux-thunk';

// import Cookies from 'js-cookie';

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

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

// 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')) : [] } };

// 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

})

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

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

export default store;

**- Run project: To Checking (5th) ... 3:06:58**

Mình có thể thay đổi số lượng các sản phẩn trong Cart hoặc xóa một vài sản phẩm khỏi Cart hoặc thêm vào các sản phẩm vào Cart, sau đó refresh website của mình thì các sản phẩm đó được update lại chứ không có mất hết sau khi refresh trang, vì nó đã được lưu vào Cookie (dùng Redux để kiểm tra) hoặc lưu vào Local Storage (bật tab Application trong DevTool để kiểm tra storage)