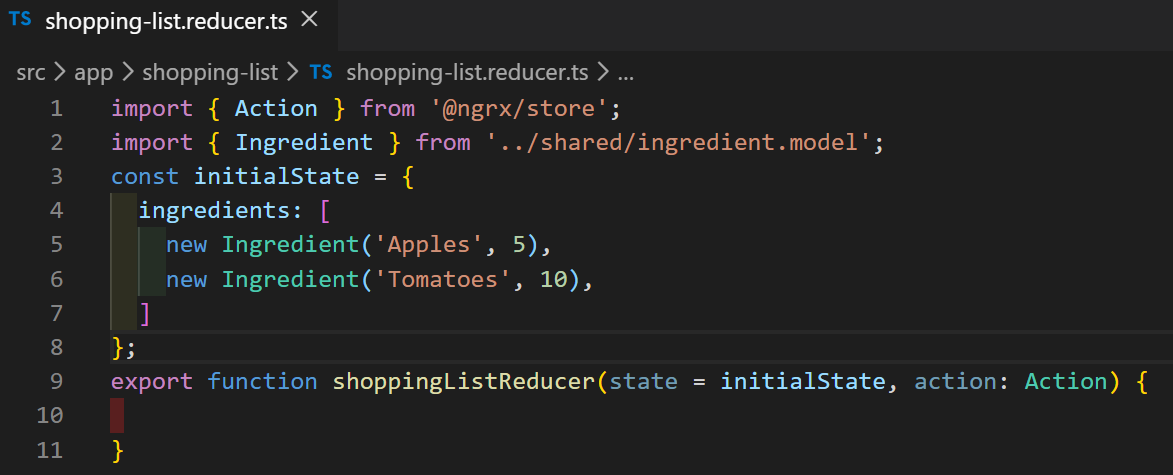
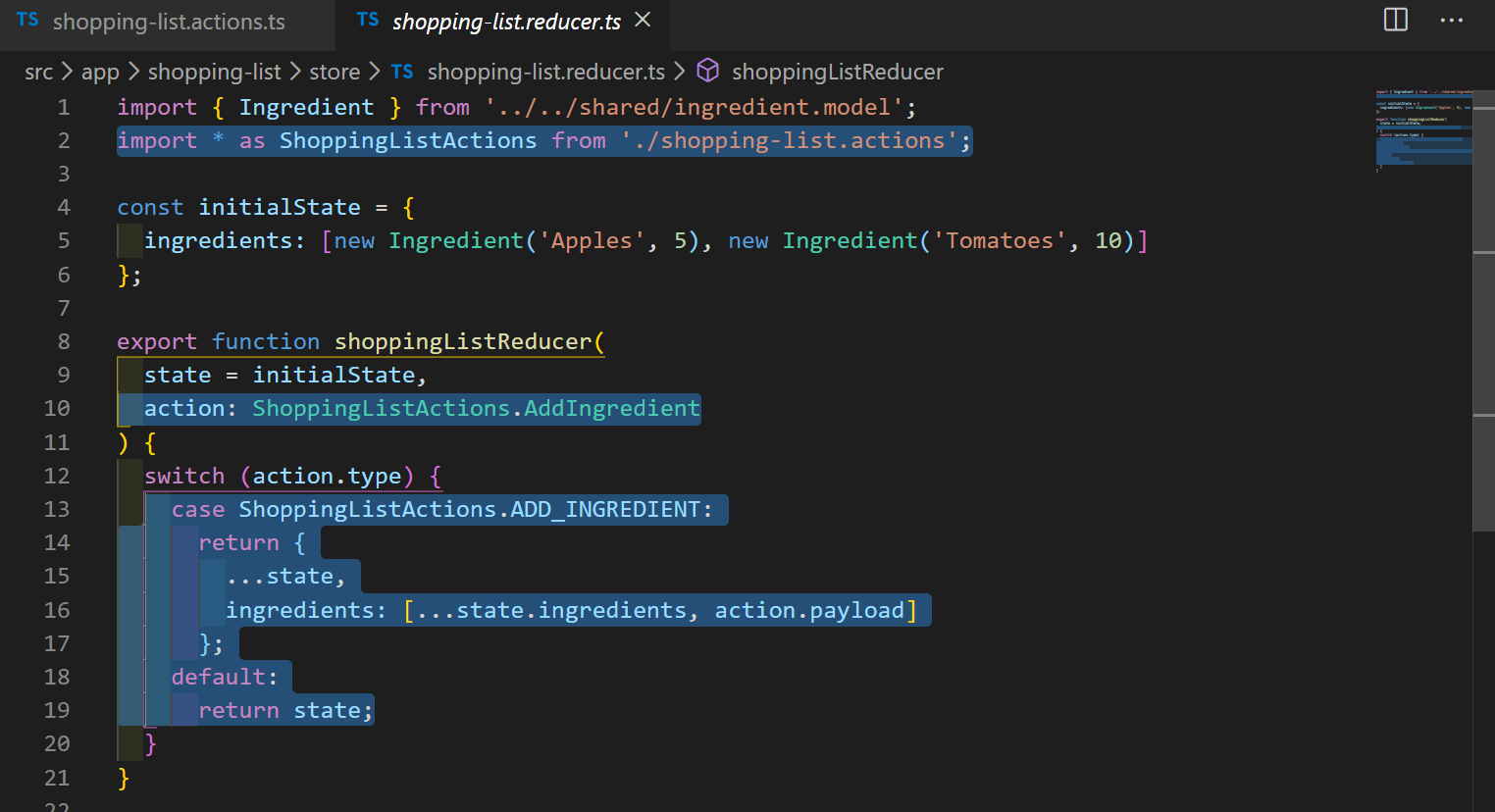
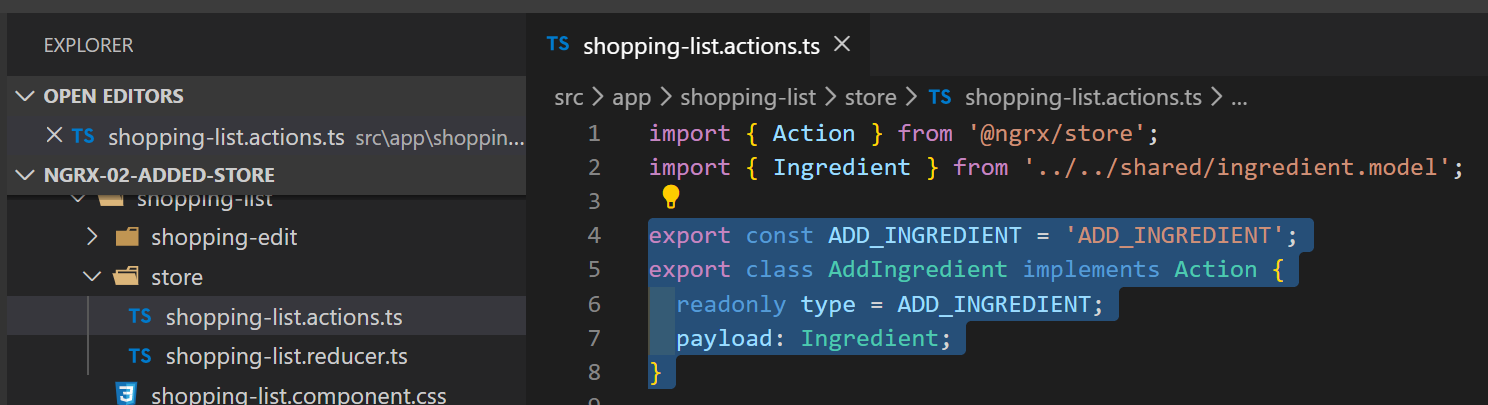
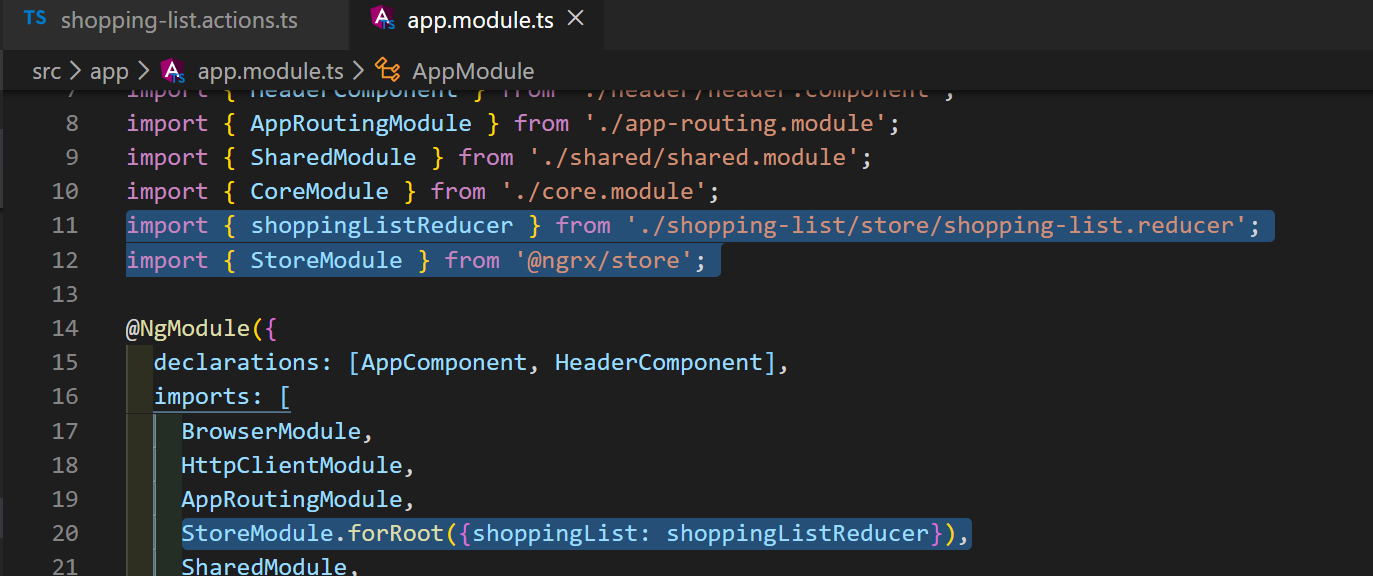
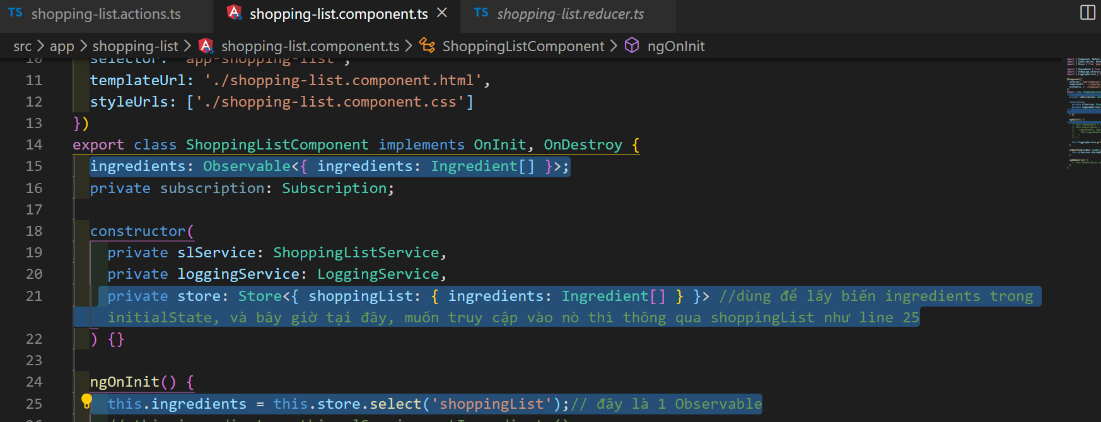
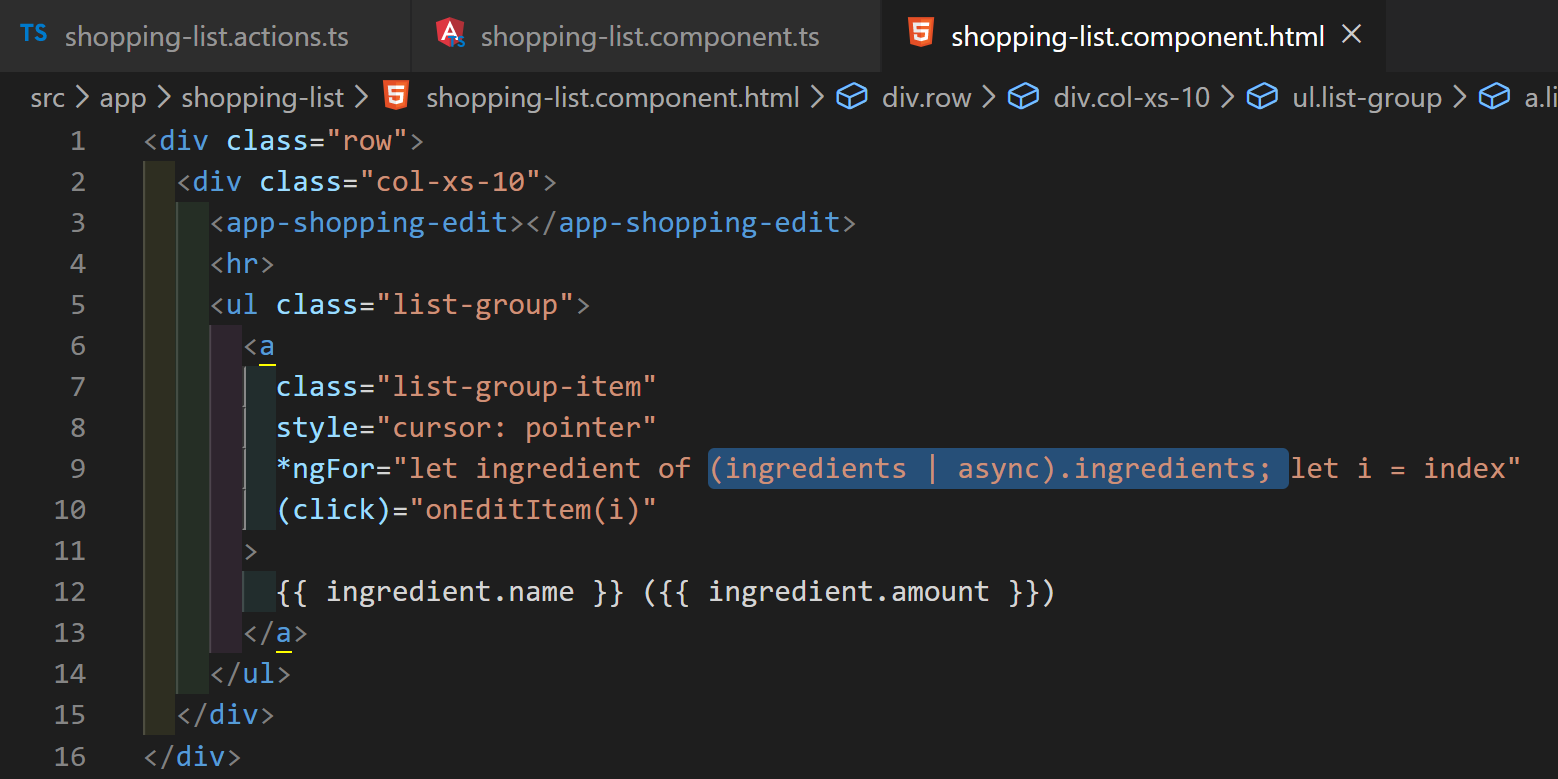
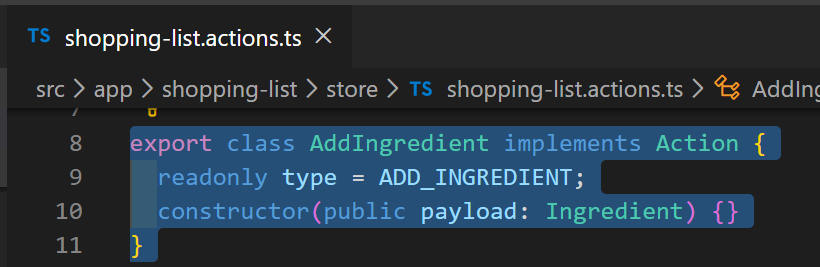
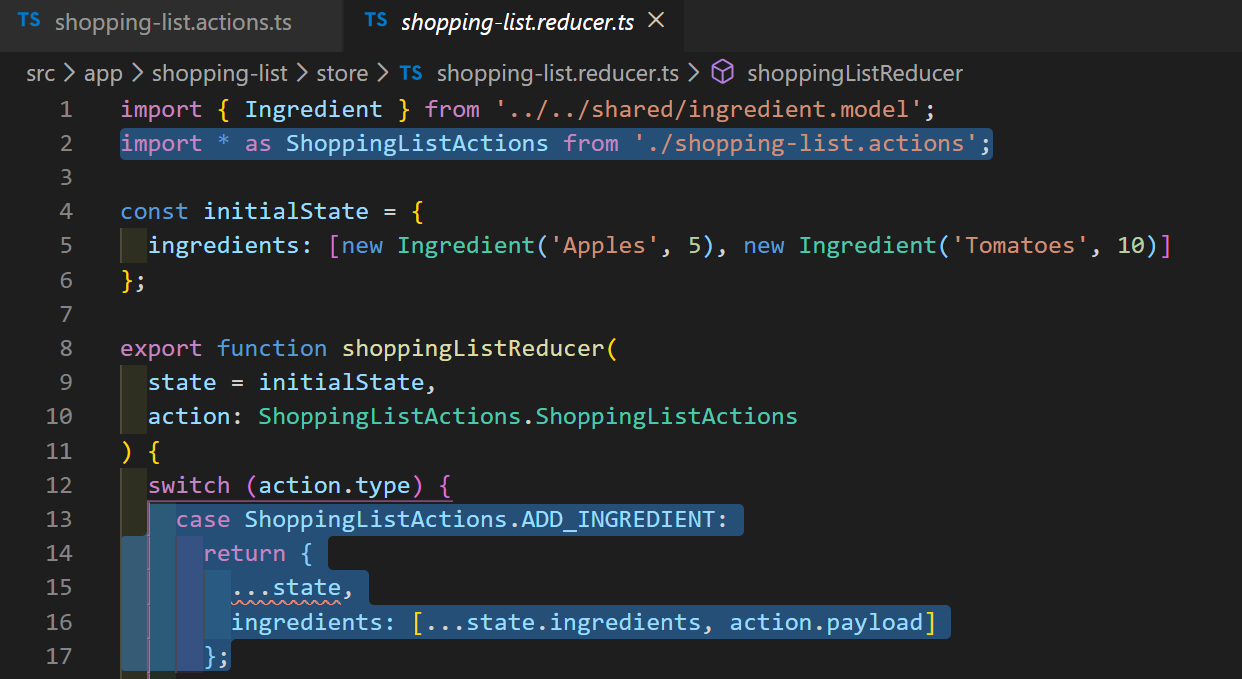
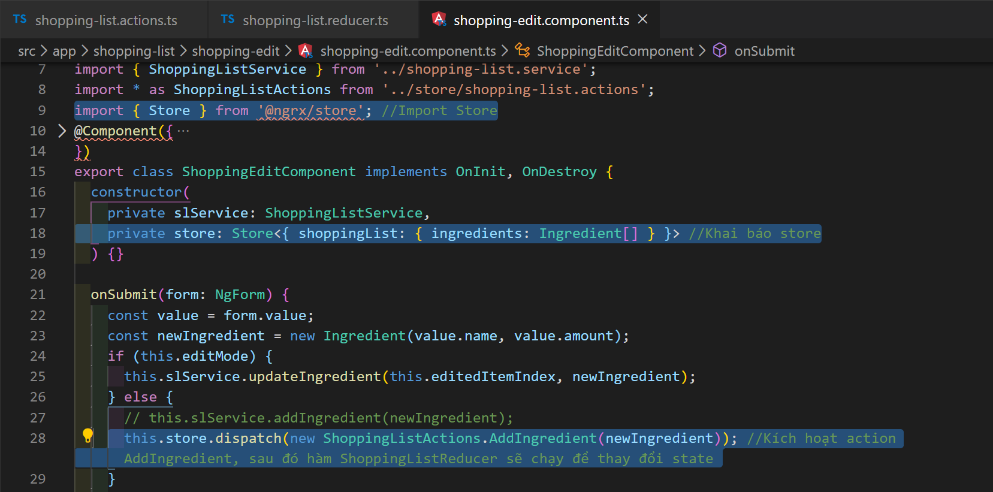
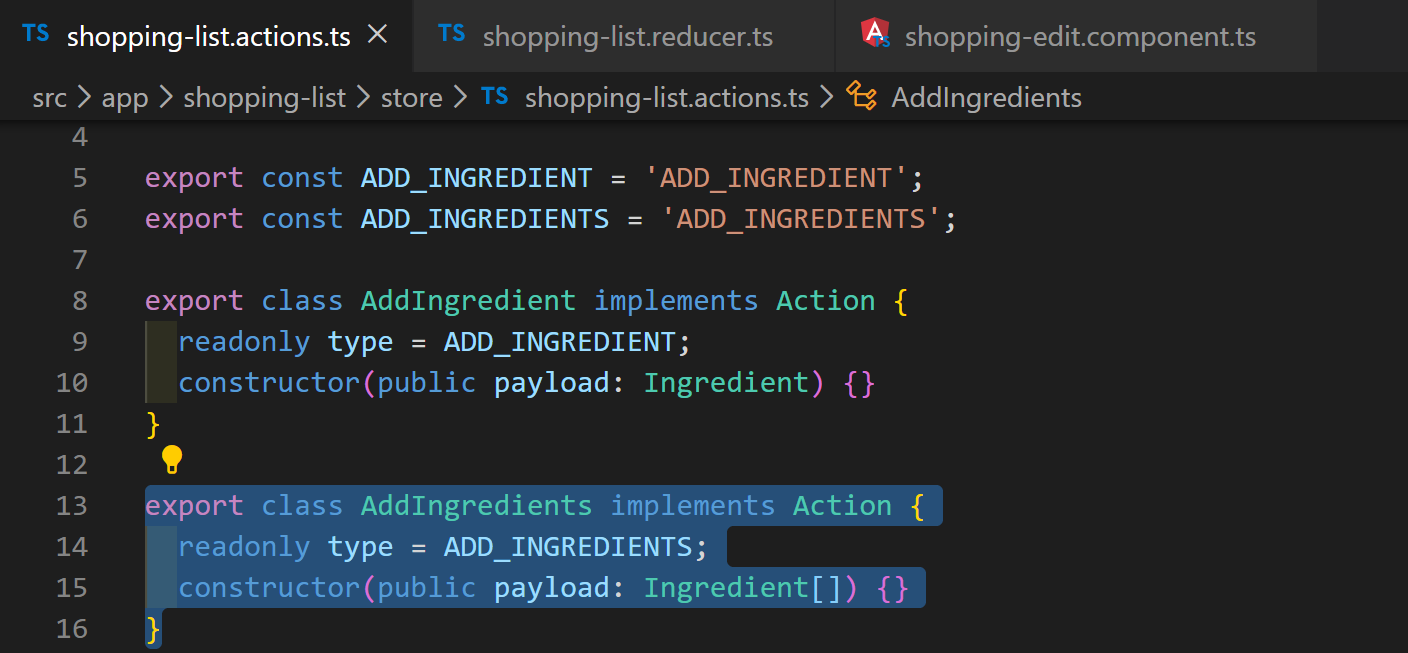
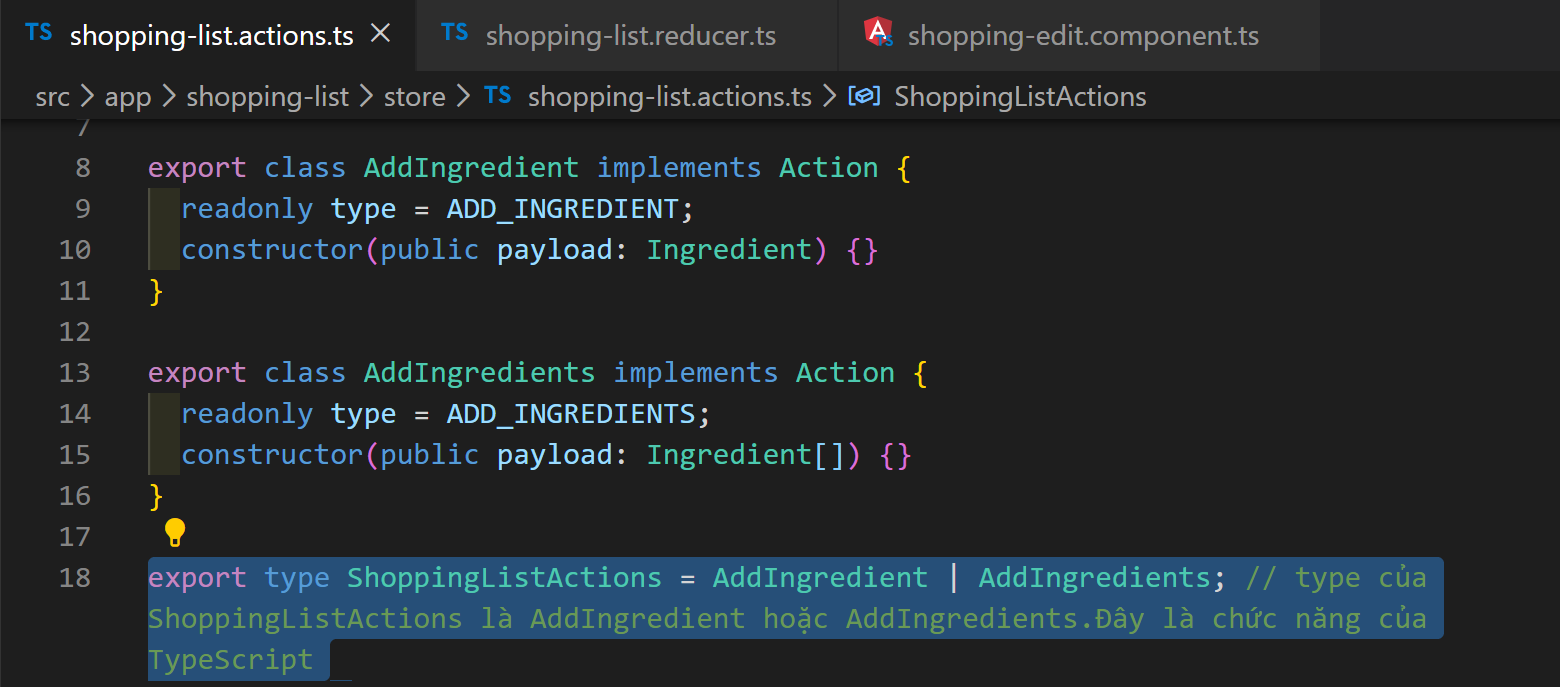
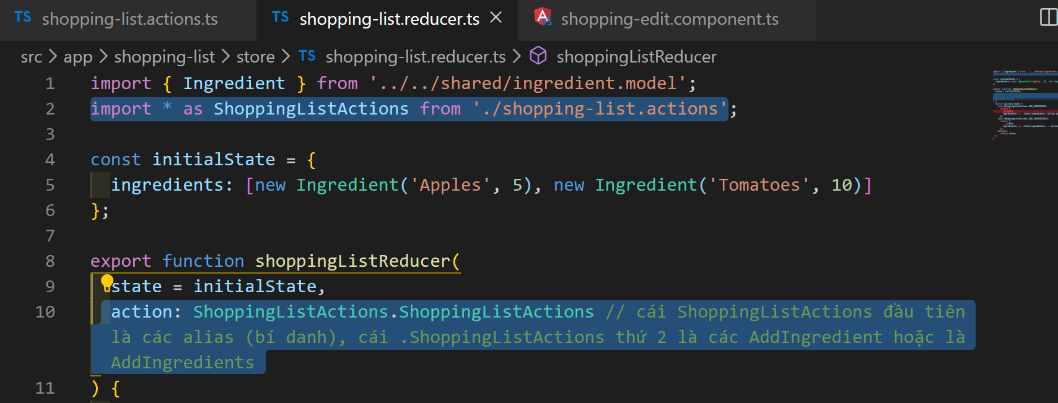
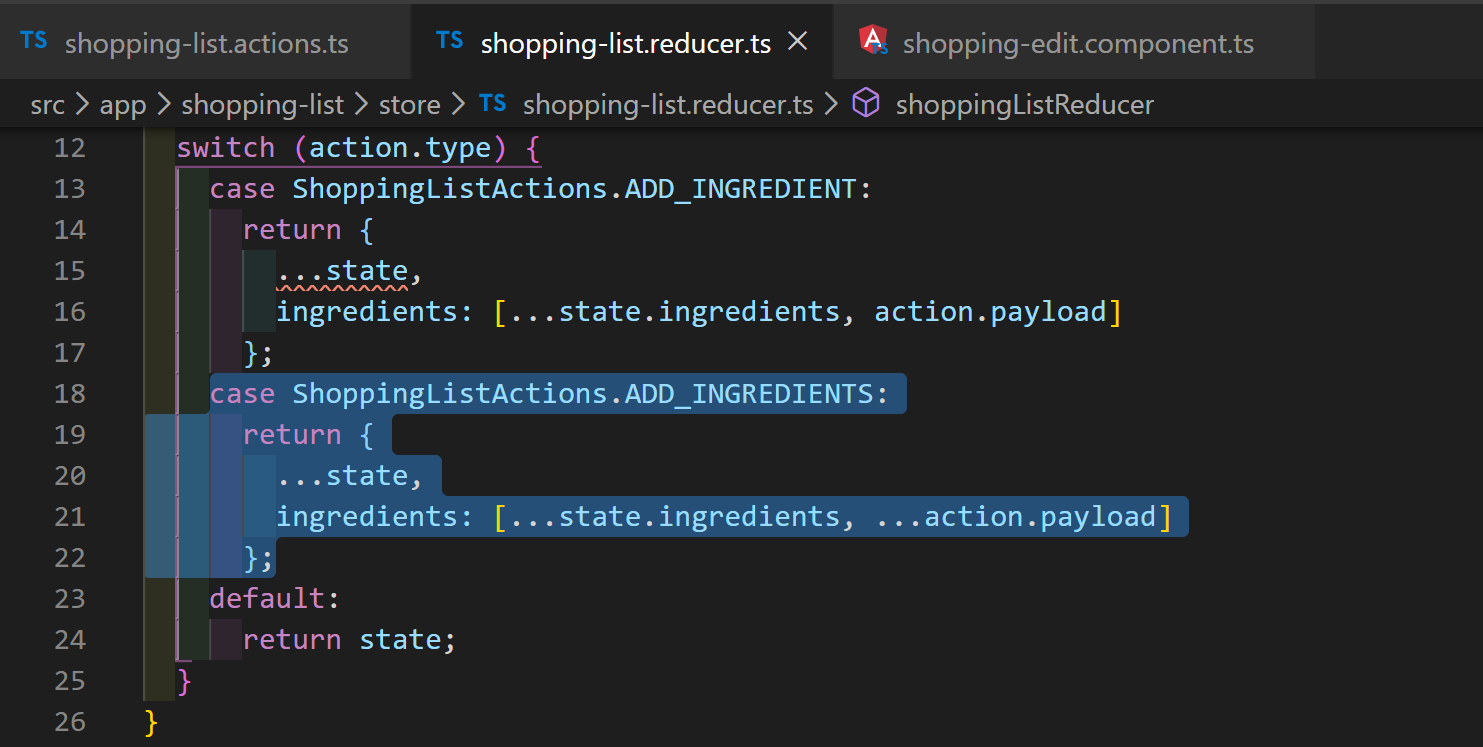
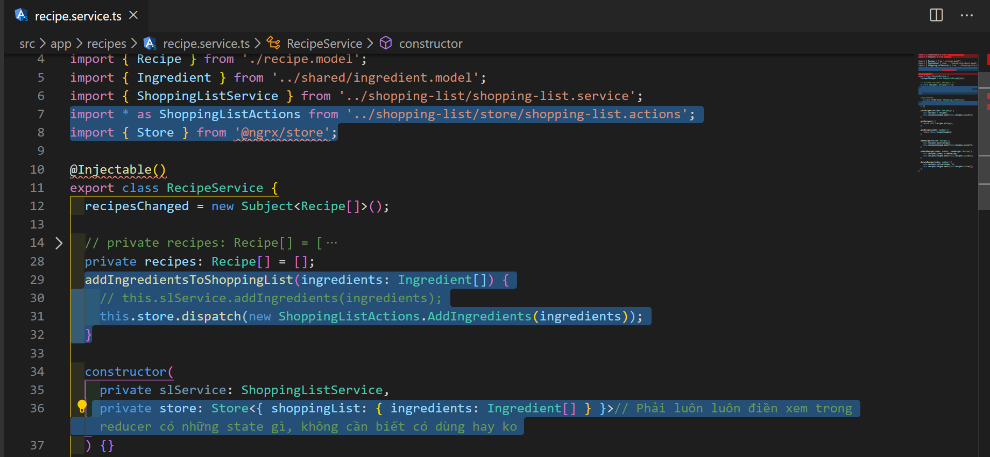
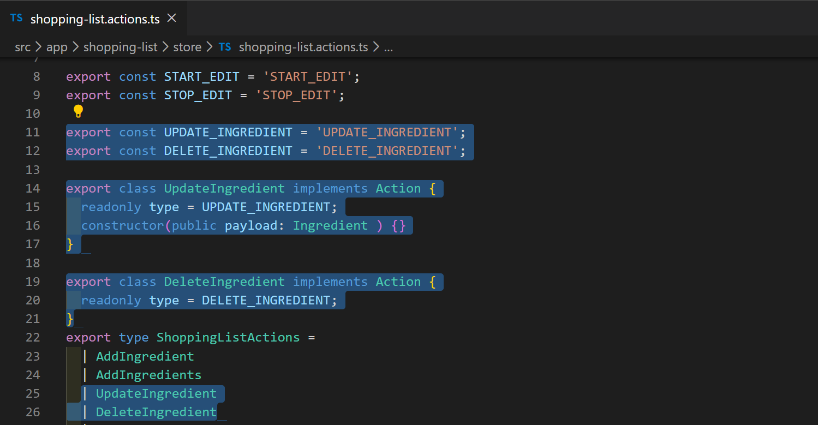
SECTION 24: Working with NgRx in our Project

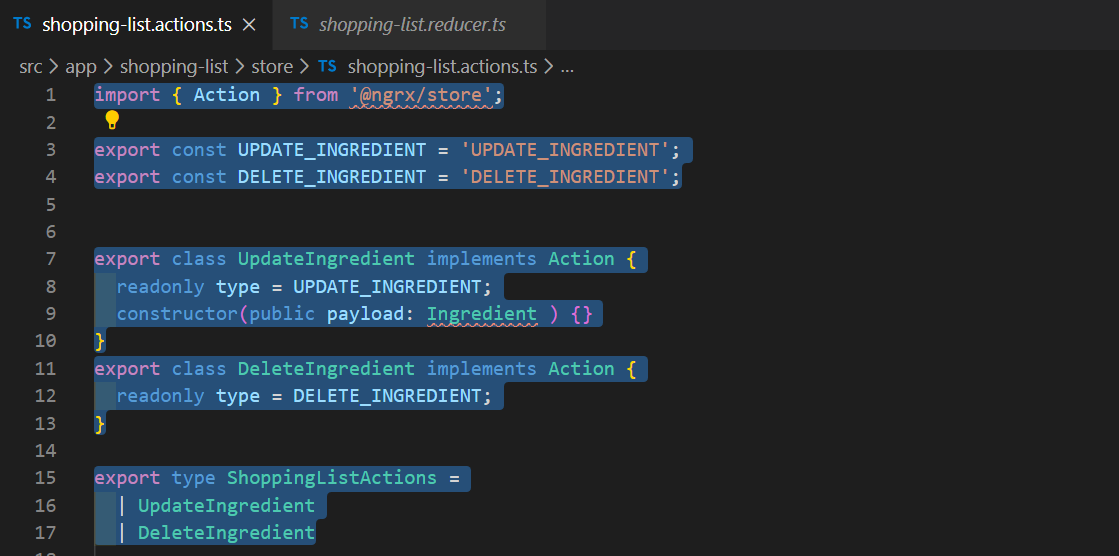
1. **Installing package:** 
2. **Getting Started with Reducers:** cài đặt, khởi tạo reducer
   1. 
3. **Adding logic to the Reducer:** Đây là cấu trúc chuẩn của reducer
   1. - Thay đổi state ingredients, thêm Ingredient vào state.ingredients thông qua action.payload.
4. **Understanding and Adding Action:**
   1. Tạo folder Store để chứa các tập tin liên quan đến store như reducer, action.
   2. - Type thì bắt buộc có của Action, còn payload thì tùy vào mình đặt tên, setup.(Như ví dụ trên là đang truyền thêm vào 1 ingredient)
5. **Setting up NgRx store:** 
   1. 
6. **Sellecting State:** Nghĩa là truy cập vào state của store từ component bất kỳ.
   1. - Giong như bên redux
   2. - vì property ingredients là 1 observable, không thể hiển thị như bth được, mà phải làm như trên (line 9).Hoặc ta vẫn subscribe() như bình thương, nhưng ko nên vì dễ dẫn đến leak memory, bug.
7. **Dispatching Actions:** Giong như truy cập vào State của store, bây giờ ta sẽ truy cập, kích hoạt 1 action của store để thay đổi state từ 1 component bất kỳ:
   1. Gỉa sử giờ ta kích hoạt hàm AddIngredient() của ShoppingListReducer:

 -Đây là action AddIngredient.**Nếu có truyền thêm tham số thì viết như thêm. Chỉ có Type là viết ngoài.**

- Đây là shoppingListReducer(), trong đó có action AddIngredient.

-Kích hoạt, chạy action AddIngredient để thay đổi State.

1. **Multiple Actions:** Cách thêm nhiều Actions vào trong 1 reducer:**Thêm Chức năng AddIngredients() vào store, sau đó cấu hình, chạy, kích hoạt, dispatch():**
   1.  -Thêm 1 action AddIngredients vào trong file ShoppingListActions.ts
   2. 
   3. - cái .ShoppingListActions thứ 2 là các AddIngredient hoặc là AddIngredients của 8.2
   4.  -Giai quyết action AddIngredients() logic
   5. -Import, inject store vào Component, sau đó thực thi hàm AddIngredients().
2. **Preparing Update() and Delete() actions:** Các bước chuẩn để viết khai báo 1 action cho Store để thực thi (viết hàm Delete() và Update()), Muốn viết hàm gì cứ làm tương tự các bước sau:
   1. **Khai báo trong actions:**  -Khởi tạo Constant UPDATE,DELETE. Khai báo actions Update, Delete. Export type Update, Delete (line 25 26)
3. **Các bước để cấu hình chuẩn, sử dụng store trong Angular và ví dụ với chức năng Update(), Delete()**:
   1. ***Tạo các folder Store con trong các Component con để quản lý, gồm 2 file Action và Reducer:*** 
      1. Trong file ShoppingListAction.ts:

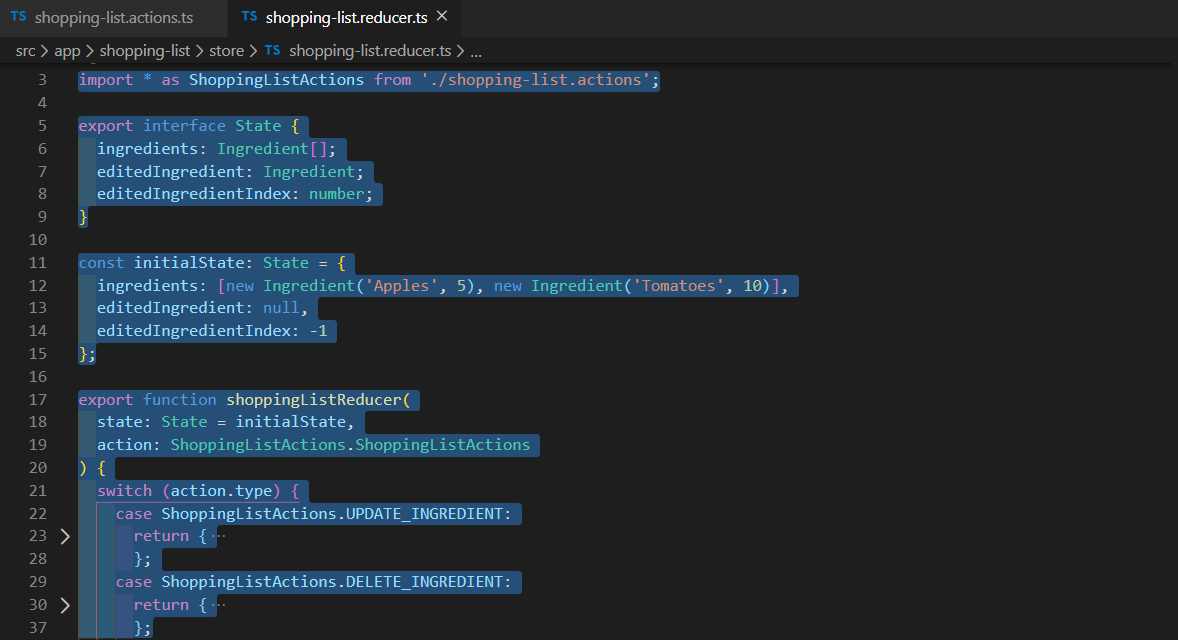
-

***-Bước 1: Import {Action}.***

***-Bước 2: Khởi tạo và Export các Const (line 8, 9), các Action (line 17-> 23)***

***- Bước 3: Và nhớ export type(line 13,14).***

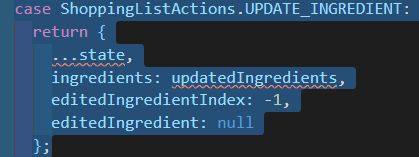
* + 1. Trong file ShoppingListReducer.ts ():

-

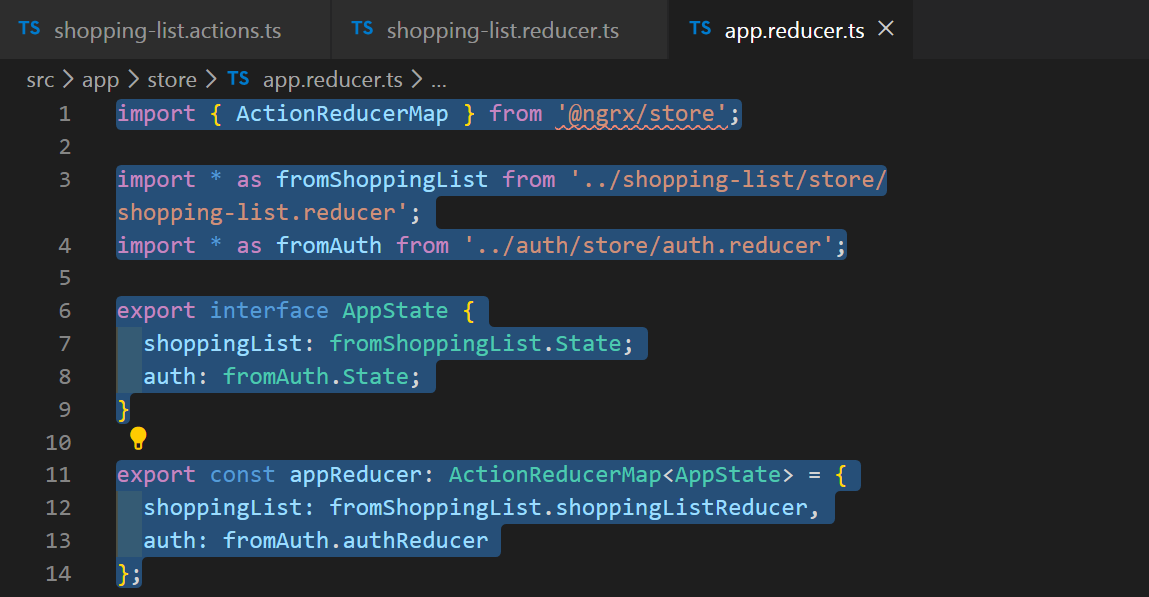
- ***Bước 1: Import file ShoppingListActions của 10.1.3 (Line 3).***

***- Bước 2: Khởi tạo interface State để khởi tạo KIỂU DỮ LIỆU của các State đang quản lý trong reducer này.(line 5)***

***- Bước 3: Khởi tạo InitialState có dạng State, để KHỞI TẠO GIÁ TRỊ BAN ĐẦU cho các State đang quản lý.(line 11)***

***- Bước 4: Khởi tạo Reducer, gồm 2 tham số truyền vào, là initialState và action do mình export bên ShoppingActions(dòng 15,10.1.1).Trong đó, với mỗi case Type khác nhau, ta sẽ xử lý logic khác nhau để thay đổi giá trị các State:***

* 1. **Gộp các reducers con thành 1 reducer AppReducer cho Store dễ quản lý, sử dụng:** Mỗi 1 component con (cũng khá lớn) ta sẽ tạo 1 reducer để quản lý, như vậy toàn APP ta sẽ có nhiều reducer con như thế.Vậy ta sẽ có 1 app Reducer để merge các reducer con ấy thành 1.
     1. **Tạo 1 folder Store bên ngoài Folder <App>, tạo file App.reducer.ts trong đó:**



- ***Bước 1:*** import {ActionReducerMap} (line 1): dùng để merge các Reducers thành 1 Reducer tổng.

- ***Bước 2:*** import TẤT CẢ các file Reducer(line 3,4).

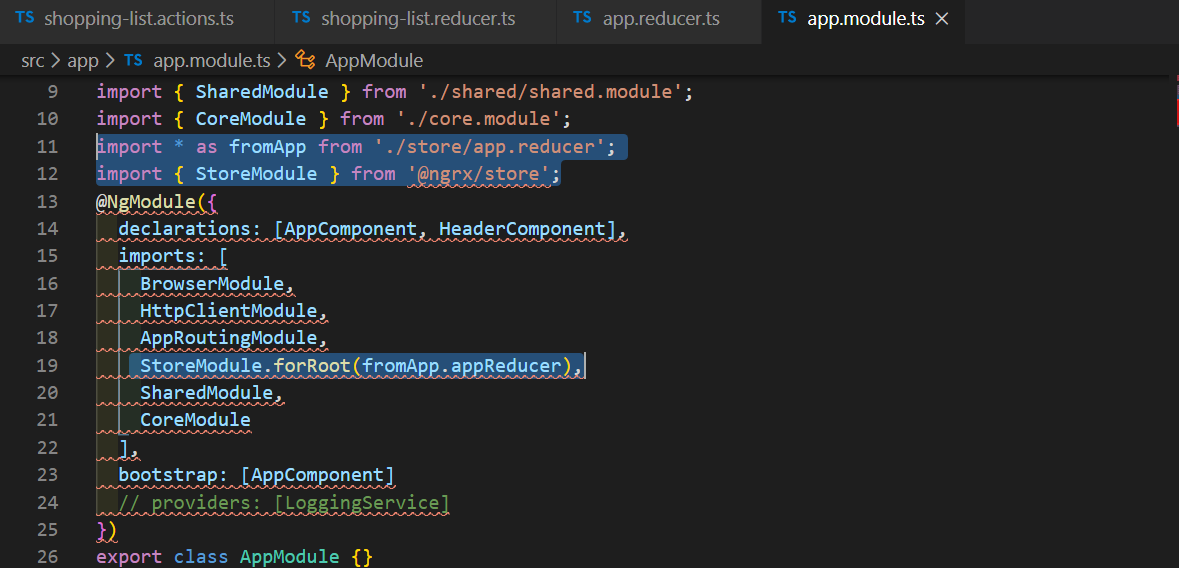
- ***Bước 3:***Khởi tạo interface AppState (line 6): dùng để khai báo kiểu dữ liệu của các State của các Reducer được import vào(như ShoppingListState, AuthState).

- ***Bước 4:*** Gộp TẤT CẢ reducers con thành 1 AppReducer (line 11):

+ActionReducerMap: dùng để gộp, nhớ import (line 1).

+Truyền TẤT CẢ Reducers vào appReducer.

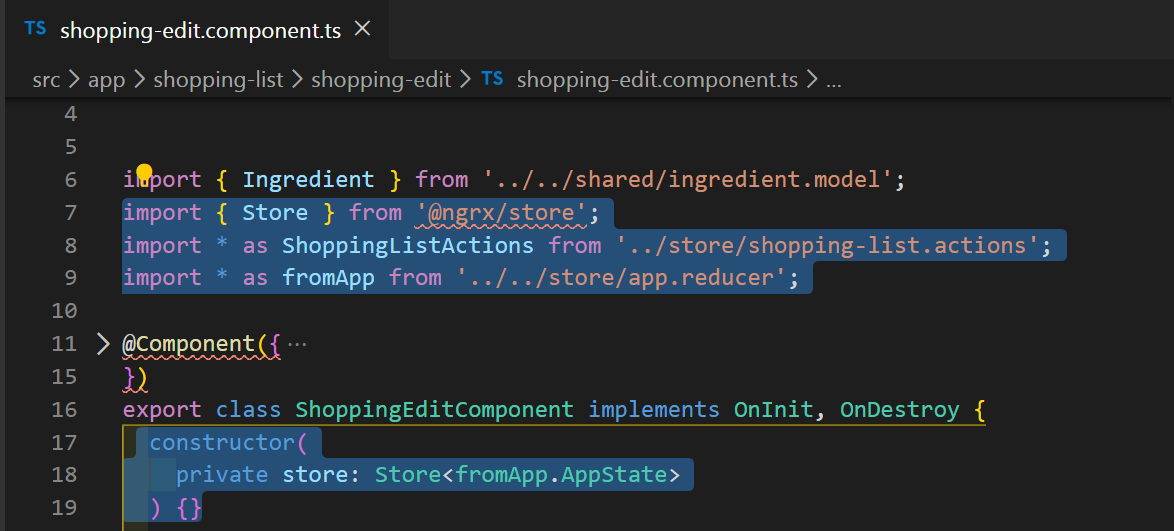
* 1. **Import, Setup, Cấu hình Store vào trong AppModule:**



- ***Bước 1:*** Import {StoreModule} (line 11).

- ***Bước 2:*** Import {appReducer} (line 12) từ AppReducer mà ta đã tạo ở 10.2.1.

- ***Bước 3:*** Truyền AppReducer vào methods .forRoot()(line 19).

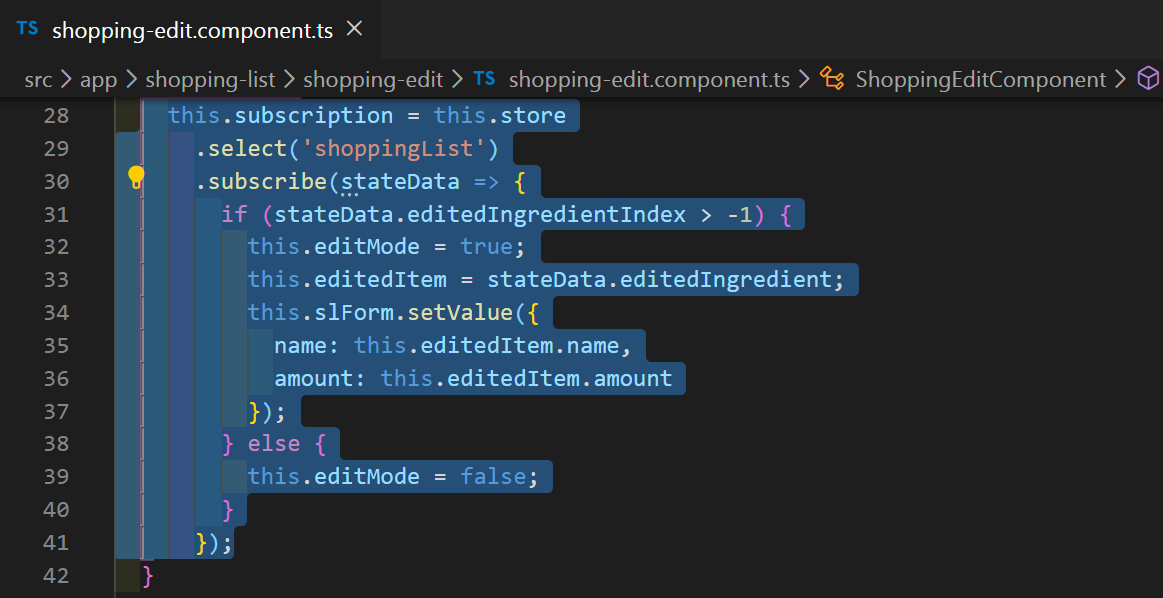
* 1. **Kích hoạt, Sử dụng Store từ 1 component bất kì của App:**
     1. **Các bước Import:**

**- *Bước 1:*** Import {Store}(line 7).

**- *Bước 2:*** Import Actions mình cần dùng(line 8).

**- *Bước 3:*** import AppReducer(line 9).

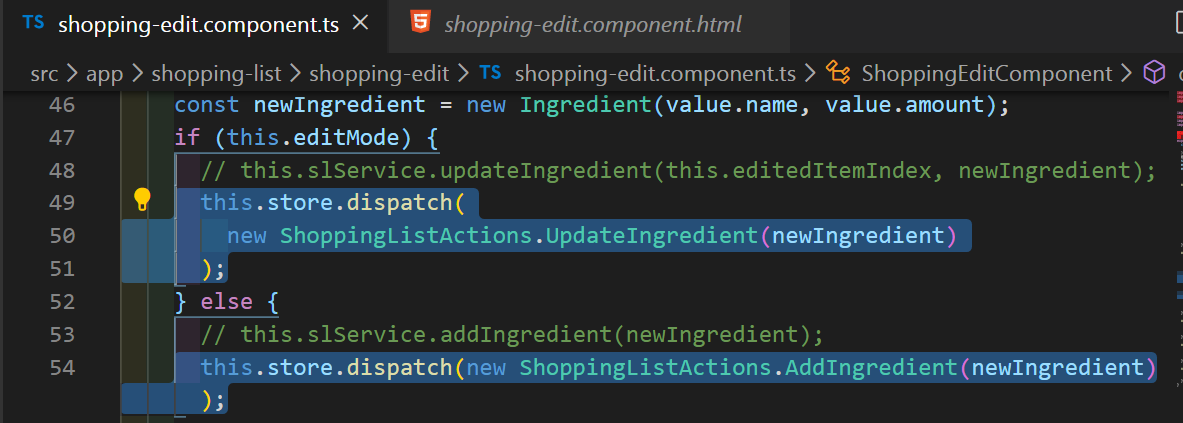
- ***Bước 4:*** khởi tạo Store thông qua constructor() (line 17).

* + 1. **Truy cập,sử dụng vào State của Store bằng method .select():**

**-**Dùng .select() để truy cập vào State TỔNG của reducer mà nó select()(line 12, hình 10.2.1). Đây là 1 State tổng(là 1 object, gồm nhiều state khác bên trong) nên ta phải truy cập thêm nữa để truy cập vào các state bên trong(line 31).

**-**Khi ta Select(), đây chỉ là 1 Observable(), nên ta cần phải có hàm .subscribe() để cập nhật các State vào cái property của Class, từ đó ta output ra màn hinh hợp lý.

**10.4.2. Kích hoạt,sử dụng Action của Store bằng method .Dispatch(new Action):**



-Bước 1: dùng hàm .dispatch() và truyền vào Action để kích hoạt Reducer thực thi các chức năng trong Reducer do mình thiết lập, từ đó thay đổi state, từ đó cập nhật lại giao diện của chương trình.