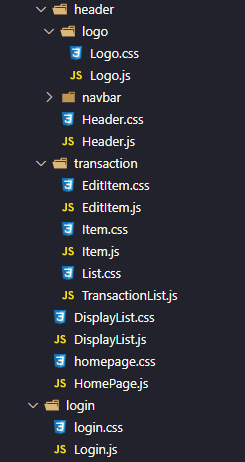
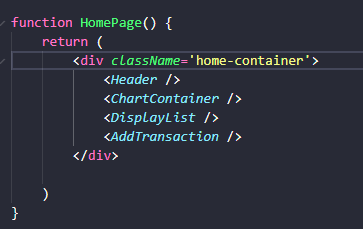
1. Cohesion và Coupling:
2. High cohesion:

Trang web được chia ra làm nhiều component nhỏ, mỗi component đảm nhận một chức năng riêng biệt.



Ví dụ: Component ‘Item’ hiển thị một transaction, component ‘Header’ hiển thị website header.

Các component được ghép lại thành các component lớn hơn và tiếp tục được ghép lại để tạo thành một trang web hoàn chỉnh.



Ví dụ: ‘HomePage’ được tạo thành từ 4 component nhỏ là ‘Header’, ‘ChartContainer’, ‘DisplayList’ và ‘AddTransaction’

1. Low coupling:

Trang web được chia làm nhiều component nhỏ với chức năng riêng biệt, sử dụng dữ liệu được lưu từ Redux và database, ít khi sử dụng tới dữ liệu từ parent component.

* Dễ dàng thay đổi các component trong trang web.

1. Nguyên lý SOLID:

Single Responsibility Principle:

| # | Related modules | Description | Improvement |
| --- | --- | --- | --- |
| 1 | Redux’s actions | Các actions của Redux, bao gồm các actions liên quan đến User, Transactions,Plans… hiện tại đang được để trong cùng một file actions duy nhất. | Chia nhỏ file actions ra làm nhiều modules nhỏ, đảm nhiệm các actions liên quan đến các vấn đề riêng biệt (ví dụ: userActions cho các actions về user, transactionActions cho các actions về transaction,…) |
| 2 | Redux’s reducers | Là các controller cho việc CRUD của Redux, hiện tại đang được để trong 1 file reducers duy nhất | Chia nhỏ file reducers ra làm nhiều modules nhỏ, đảm nhiệm các nhiệm vụ riêng biệt (ví dụ: userReducers cho user, transactionReducers cho transaction,…) |

1. Áp dụng Design Pattern:

* Observer pattern:

Các component trong trang web được bọc trong

<Provider store={store}>

….

…

</Provider>

Với cơ chế này, các component đã đăng kí ( subcribe ) data trong store của redux, và có thể theo dõi, lấy dữ liệu bất cứ khi nào component cần thông qua lời gọi useSelector(). Ở trường hợp này, store của redux đóng vai trò là Provider, các component đóng vai trò là Consumer.