Assignment - Week 10/32 Day 5

The following assignment is purposefully designed to come out of the todo app mindset. Build a minimalistic shoes selling ecommerce application with the following pages and functionalities. I will guide you through the actions and reducers that need to be taken care of. This highly allows you to understand the purpose of multiple reducers and binding them. No need for complex/detailed designs. The routes you need to create are as follows.

/ login - Login Page
/ register - Register Page
/ - Home page where all the products are shown
/ cart - Cart page containing all the products.
/ finish - Success page of notifying purchase made.

The explanation of state and functionality for each page are as follows.

1) Login, Register pages: Allow you to Login, Register. You don't have to connect to any API. Just collect the data from the form and put into the state, the user data is as follows. If possible have form validation like confirm Password, password required in Register page. The data for a user is as follows.
{ email: string, password: string }. authReducer should do this job.
(P.T.O)

Make sure you have relevant redirecting buttons to login and register pages. Like “ALREADY A NEW USER?” button inside the Register page and “VISITING FOR THE FIRST TIME?” button inside the Login page.

2) Home page: It doesn’t need any auth check as we are just viewing the products. Each product should contain the following data.

```
{
  productId: string,
  productImageURL: string (Fetch it from internet),
  productPrice: number/float
}
```

Each product should have a button to add to cart, which allows us to add to cart. **productsReducer should fetch some n products. (Dump an array of objects and call that action inside componentDidMount)**

3) Cart page: This is where you will display all the items that have been added to cart as well as have a button naming **Checkout** at the end of the page. Clicking that button should generate an order and redirect to the Finish page. The cart state should contain the following structure

```
{
  products: [
        {
          productId: string,
          productImageURL: string (Fetch it from internet),
          productPrice: number/float,
```

quantity: number

},

],

totalQuantity: number,

total: number/float

}.

Each product inside the cart should have the ability to increase/decrease the number of items as well as the total price. For example, 2 iPhone 11 Pro Max, 5 iPhone 11, 3 iPhone XS max are added to cart, and the price list for each mobile are 1100 USD, 700 USD, 900 USD respectively. So the total cart value is (2 * 1100) + (5 * 700) + (3 * 900) => 8400 USD. So, you have to not only concentrate on updating the quantity value of the particular product, but also the totalQuantity as well as the total value. If you decrease the quantity below zero, i.e. if you don't want an iPhone 11 Pro max at all, then that product should be removed from the cart. **cartReducer should take care of all these actions.**

Hint: USE REDUCE METHOD TO CALCULATE THIS.

1. **totalQuantity => sum of all product's quantity value**
2. **Total => sum of product of productPrice and quantity for each product.**

4) Finish Page: You can only place an order if you are authenticated. So make sure that this route is protected **(Just check the user state whether**

it's empty or not. If it's empty then redirect to login). This should generate a success message, saying that order is successful. No need to verify or visit the backend whatsoever. **No Reducer is needed here.**

At the end, this is the flow for a minimalistic eCommerce application. I know the above data flows are tough to wrap up, but this is how we are supposed to grow and get ready for our monthly test. As a conclusion you are building a store consisting of reducers. Auth, Products and Cart. Feel free to come up with any component architecture, as that's irrelevant here. Also don't worry about styling at all. The Redux Actions and state management is all that I need at the end.

Try to watch the extra lecture presented on July 10th 2020, where I would have discussed how to redirect to a certain page, how to handle state changes, how to have props drilled and tips and tricks regarding building an application less redundant in terms of mapDispatchToProps and mapStateToProps.

Lastly, please dont feel bad if you were not able to build this mammoth. It’s literally a huge work to wire this thing up, and congratulations for even taking part in this assignment already. Good luck :)