ingré

Hey,

Youssef here,

I took a look at your project, and here is my review… I hope it help ^^

First of all real nice project. It has a good structure and the code quality is overall good. Here are some areas I would personally improve:

- The Authentication is handled in a singleton class. Storing any variable state outside of React is kind of a no go. Consider having a context for the authentication. In the routing, you can check if the user is logged in or not and filter out the pages he shouldn't be able to see.
- A lot of components are called App. Normally, it should be the same as the files names (e.g. `RecipeCard`)
- In `RecipeCard`, multiple components are defined inside of the component. This will lean to a lot of unnecessary re-renders (`saveButton`, `editButton`...). Consider moving them outside the component and passing the needed data through props.
- Consider using Typescript, it will make your developer experience a bit better ;)
- I like the approach you took to integrate `antd` in your code (props delegation+default props) 👍
- This is a common pitfall in React, to have states that need to be kept updated with `useEffect`. Instead, you can have a simpler code by using a normal const

  ```
  // before
  const [pro, setPro] = useState(false);

    useEffect(() => {
      if (!userLoading && !userError && userData?.getUser) {
        // console.log('userData', userData);
        setPro(userData.getUser.pro);
      }
    }, [userLoading, userError, userData]);
  // after
  const pro = !userLoading && !userError &&userData.getUser.pro;
  ```

- The page customize is very complex. Consider adding components and using custom hooks to break it down
- Some code can be simplified. in `Home/index.js`:

  ```
  const handleRefresh = async (query) => {
      try {
        // get credentials from backend
        const { appId, appKey } = await getAppCredentials();

        setLoadingEdamam(true);
        let noQuery = { q: ' ', diet: [], health: [], cuisineType: [], mealType: [], dishType: [] };
        let hits;
        if (query === 'vegetarian') {
  ```

```
    hits = await FetchEdamam({ ...noQuery, health: ['vegetarian'] }, appId, appKey);
   } else if (query === 'vegan') {
    hits = await FetchEdamam({ ...noQuery, health: ['vegan'] }, appId, appKey);
   } else if (query === 'balanced') {
    hits = await FetchEdamam({ ...noQuery, diet: ['balanced'] }, appId, appKey);
   } else if (query === 'breakfast') {
    hits = await FetchEdamam({ ...noQuery, mealType: ['breakfast'] }, appId, appKey);
   } else if (query === 'lunch') {
    hits = await FetchEdamam({ ...noQuery, mealType: ['lunch'] }, appId, appKey);
   } else if (query === 'dinner') {
    hits = await FetchEdamam({ ...noQuery, mealType: ['dinner'] }, appId, appKey);
   } else {
    hits = await FetchEdamam(null, appId, appKey);
   }
   // console.log('hits = ', hits);
   setEdamamRecipes(hits);
   setLoadingEdamam(false);

   setUpdateRecipes(true);
  } catch (e) {
   console.error(e);
  }
 };
// after (better readability)
const healthTags = ['vegetarian', 'vegan']
const dietTags = ['balanced']
const mealTags = ['breakfast', "lunch", "dinner"]

if (healthTags.includes(query)) {
    hits = await FetchEdamam({...noQuery, health: [query]}, appId, appKey);
} else if (dietTags.includes(query)) {
    hits = await FetchEdamam({...noQuery, diet: [query]}, appId, appKey);
} else if (mealTags.includes(query)) {
    hits = await FetchEdamam({...noQuery, mealType: [query]}, appId, appKey);
} else {
    hits = await FetchEdamam(null, appId, appKey);
}
```

- The global state is in general overloaded. I would advise to split the ui and global state more by not having fields like `componentDidMount` in the global state and by moving the fetching function calls outside of the components (either in the reducer or custom hooks)
- I liked that you used apollo for querying since it provides loading and error states. It would be cool to used another hook for making calls, that way, you don't have to take care of loading, success/error states. You can either write one yourself or use react-query's useMutationHook for that.
- I would also consider reducer the amount of console.logs 😊

It's a very cool project, and definitely impressive. Nice job 👍

Kind regards,

Youssef