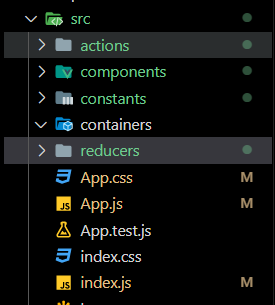
**BÁO CÁO 7: THỰC HIỆN DEMO REDUX**

1. **Ý tưởng**

* Em sẽ xây dựng app cho người dùng đăng ký, đăng nhập. Nếu đăng nhập thành công thì sẽ chuyển đến trang chính để có thể thêm, xóa, sửa dữ liệu.
* Phần đăng ký em sẽ sử dụng thư viện “bcrypt” để mã hóa mật khẩu và lưu vô database. Đồng thời chuyển cặp giá trị username, password thành một token bằng jwt và lưu lại trong database.
* Khi người dùng đăng nhập, mật khẩu sẽ được so sánh với mật khẩu đã mã hóa. Nếu đúng sẽ cho người dùng quyền truy cập vào trang chính đồng thời đưa cho người dùng token và lưu vào trong local storage.
* Khi đăng xuất thì app sẽ xóa token đó trong local storage và chuyển về trang đăng nhập
* Trang chính sẽ hiển thị thông tin người dùng bao gồm họ tên, số điện thoại, giới tính

1. **Cấu trúc thư mục**

****

* Thông thường sẽ có 2 cách chia thư mục: Function-first và Feature-first. Nhưng ở đây em làm theo Function-first.
* Trong *index.js*, em sẽ tạo store đồng thời apply middle *Thunk* để xử lý async await function vào project.

1. **Actions**

* Đây là nơi đặt tất cả các actions, xử lý logic của app. Em chia ra 2 file action chính là *auth.js* để xử lý đăng ký, đăng nhập và *contacts.js* để xử lý logic các chức năng thêm xóa sửa dữ liệu.
* Với action đăng ký em sẽ trả về tên của action này và state của nó sau khi thực hiện action

export const register = (user) => {

  return {

    type: types.REGISTER,

    user,

  };

};

* Đồng thời, em sẽ khai báo một hàm xử lý logic và gọi action này mỗi khi nhấn đăng ký

export const registerRequest = (user) => {

  return (dispatch) => {

    const salt = bcrypt.genSaltSync(10);

    const hassPassword = bcrypt.hashSync(user.password,salt);

    const token = jwt.sign(

      {

        username: user.username,

        password: hassPassword,

      },

      "SECRET\_TOKEN"

    );

    const data = {

      username: user.username,

      password: hassPassword,

      token: token,

    };

    axios

      .post("https://60b0f8b91f26610017fff943.mockapi.io/api/v1/users", data)

      .then((response) => {

        dispatch(register(response.data));

      })

      .catch((err) => {

        console.log(err);

      });

  };

};

* Tương tự như vậy với đăng nhập và các actions khác.

1. **Components**

* Đây sẽ là nơi khai báo tất cả các component và nếu muốn gọi một action thì em sẽ gọi thông qua props. Để làm được điều này em sẽ thêm một hàm là *mapDispatchToProps* và sử hàm *connect()* để nối dispatch props với lại component. Ví dụ cho form đăng nhập:

const mapDispatchToProps = (dispatch) => {

  return {

    loginRequest: (user) => {

      dispatch(loginRequest(user));

    },

  };

};

export default connect(null, mapDispatchToProps)(FormLogin);

1. **Constants**

* Đây là nơi khai báo tên của những actions có trong app, mục đích là để mỗi lần gọi tên thì sẽ tránh trường hợp gọi nhầm tên và dễ kiểm soát lỗi hơn.

export const SET\_CURRENT\_USER = "SET\_CURRENT\_USER";

export const LOGIN = "LOGIN";

export const REGISTER = "REGISTER";

export const FETCH\_CONTACT = "FETCH\_CONTACT";

export const ADD\_CONTACT = "ADD\_CONTACT";

export const EDIT\_CONTACT = "EDIT\_CONTACT";

export const DELETE\_CONTACT = "DELETE\_CONTACT";

export const SEARCH\_CONTACT = "SEARCH\_CONTACT";

1. **Reducers**

* Đây sẽ là nơi quản lý state đầu vào, đầu ra. Bởi vì không nên để một file quản lý tất cả các state nên em sẽ chia thành nhiều file nhỏ quản lý những nhóm state. Cuối cùng em sẽ gộp lại bằng hàm *combineReducer()*

export default combineReducers({

  loginReducer,

  registerReducer,

  contactReducer,

  searchContactReducer,

  auth

});

1. **Source code**
2. **Demo**
3. **Các vấn đề gặp phải**
4. **Vấn đề về chuyển trang**

* **­**Hiện tại em muốn bỏ trang đăng nhập và trang chủ thành 1 đường dẫn. Chính vì thế việc em vẫn chưa biết sử dụng phương pháp gì để chuyển trang thông qua *react router dom.*
* Cách giải quyết hiện tại là em sẽ đưa một đường dẫn khác cho trang đăng nhập, nếu người dùng vô trang chính mà chưa đăng nhập, nó sẽ chuyển địa chỉ qua đăng nhập.

1. **Vấn đề về trễ state**

* Khi vào trang chính thì lập tức data sẽ được lấy về thông qua action *fetchContact* tuy nhiên state của component lại bị trễ state nên không nhận được kết quả sau khi fetch, vì thế không có dữ liệu được hiển thị trên màn hình.

1. **Hướng đi tiếp theo**

* Tìm hiểu Redux-saga để ứng dụng vào project
* Tìm hiểu về webpack và Jest