✅ 1. Authentication – Xác thực người dùng

🌱 Khái niệm:

Xác thực là quá trình **kiểm tra danh tính** người dùng (ai đang đăng nhập).

Trong ứng dụng SPA (Single Page App), phổ biến nhất là dùng **JWT (JSON Web Token)**:

- Khi user đăng nhập, server trả về 1 JWT.
- Token này được **gửi kèm mỗi request** (qua Authorization header hoặc cookie).
- Server sẽ giải mã token và xác nhận quyền truy cập.

🔐 Lưu trữ Token:

Có 3 lựa chọn phổ biến:

Cách lưu	Ưu điểm	Nhược điểm
localStorage	Dễ dùng, giữ được sau reload	**Dễ bị XSS đánh cắp**
sessionStorage	Dễ dùng, reset khi đóng tab	Cũng bị XSS
Cookie (HttpOnly) ✅	**Bảo mật cao, tránh được XSS**	Không truy cập được bằng JS, cần cấu hình tốt từ server

👉 **Khuyên dùng cookie HttpOnly + Secure** nếu có quyền cấu hình server. Trên frontend có thể dùng thư viện js-cookie để quản lý cookie (phiên bản demo, không phải HttpOnly).

🛡️ Chống XSS:

- Luôn **sanitize** đầu vào của người dùng trước khi render.
- Dùng thư viện như DOMPurify để làm sạch HTML.

Ví dụ:

```
js
import DOMPurify from 'dompurify';
```

```
const SafeHTML = ({ html }) => (
  <div dangerouslySetInnerHTML={{ __html: DOMPurify.sanitize(html) }} />
);
```

🔓 2. Authorization – Phân quyền

🌱 Khái niệm:

Authorization là quá trình kiểm tra **người dùng được phép làm gì**.

✅ RBAC – Role Based Access Control:

- Mỗi user có 1 role: ADMIN, USER, MODERATOR, v.v.
- Dựa vào role để bật/tắt UI hoặc bảo vệ route.

ts
```
const canAccess = user.role === 'ADMIN';
```

React Component ví dụ:

tsx
```
const RBAC = ({ allowedRoles, user, children }) => {
  return allowedRoles.includes(user.role) ? children : null;
};
```

🧩 PBAC – Permission Based Access Control:

- Granular hơn RBAC.
- Dựa vào **quyền cụ thể hoặc thuộc tính**, ví dụ:
    - User chỉ được xóa **comment của mình**.
    - User chỉ được sửa **project mà mình tạo**.

Ví dụ:

ts
```
const canDeleteComment = comment.userId === currentUser.id;
```

🧬 3. User State – Quản lý thông tin người dùng

Dữ liệu user (tên, email, role, token) là một **global state** vì mọi nơi trong app đều cần dùng.

🛠️ Cách quản lý user:

a) React Context:

ts

```
const UserContext = createContext(null);
const useUser = () => useContext(UserContext);
```

b) React Query + react-query-auth:

- Có thể fetch và cache user qua React Query.
- react-query-auth cung cấp useUser, login, logout, register...

c) State lib khác: Redux, Jotai, Zustand,...

🕵️ 4. Gợi ý cấu hình toàn diện

Khi user login:

1. Gửi request -> nhận token.
2. Lưu token vào cookie.
3. Gọi /me API để lấy user info.
4. Lưu user info vào global state.

Khi load app:

1. Kiểm tra xem có token không (cookie).
2. Gọi /me để xác thực user.

3. Nếu lỗi (401) → logout.

✅ Tổng kết

Mục tiêu	Cách làm
Xác thực	Dùng JWT, lưu bằng cookie hoặc localStorage (ưu tiên cookie + HttpOnly)
Phân quyền	Dùng RBAC (vai trò) và PBAC (quyền theo logic)
Trạng thái người dùng	Lưu trong React Context hoặc React Query
Bảo mật dữ liệu	Sanitize đầu vào, chống XSS, dùng cookie bảo vệ
Báo lỗi	Dùng Sentry hoặc console.error ở ErrorBoundary