React.js Auth Interview Prep

If you're preparing to switch jobs with **3 years of experience in React.js**, and targeting companies like **Infosys, TCS, Wipro**, you can expect **practical and conceptual questions** around **authentication and authorization**, especially since these are key for building secure enterprise applications.

Below are **common interview questions**, what the interviewer is *really* looking for, and **small code snippets or explanations** for each:

🔐 1. What is the difference between Authentication and Authorization?

What they want to hear:

- Authentication = Who you are (login).
- Authorization = What you are allowed to do (access control).

Example:

```plaintext
Login (Authentication): Are you really the user you claim to be?
Role check (Authorization): Can you access the admin dashboard?
```

🧾 2. How would you implement authentication in a React app?

Expected Answer:

- Use a backend API for login (e.g., Node/Express or Java/Spring).
- Send credentials → Get token (JWT or session ID) → Store token (localStorage/cookies).
- Use token for protected routes.

Code Snippet:

```js
// Login function
async function login(email, password) {
  const res = await fetch('/api/login', {
    method: 'POST',
    headers: {'Content-Type': 'application/json'},
    body: JSON.stringify({ email, password })
  });
  const data = await res.json();
  localStorage.setItem('token', data.token); // Store token
}
```

🚪 3. How do you protect routes in React (Route Guards)?

What they're testing:

- Use of `React Router`, conditional rendering, and token checks.

Code Snippet:

```jsx
// ProtectedRoute.jsx
import { Navigate } from 'react-router-dom';

const ProtectedRoute = ({ children }) => {
  const token = localStorage.getItem('token');
  return token ? children : <Navigate to="/login" />;
};
```

Usage:

```jsx
<Route path="/dashboard" element={<ProtectedRoute><Dashboard /></ProtectedRoute>} />
```

🔑 4. How do you implement Role-based Authorization?

Expected:

- You fetch the user's role after login.
- Render UI or redirect based on role (admin, user, etc.).

Code Snippet:

```js
// Assume you fetched user data
const user = { role: 'admin' };

// Conditional rendering
{user.role === 'admin' && <AdminPanel />}
```

Or for routes:

```jsx
<Route path="/admin" element={user.role === 'admin' ? <AdminPage /> : <Navigate to="/not-authorized" />} />
```

🍪 5. Where do you store JWT tokens and why?

What they expect:

- localStorage: Easy, but vulnerable to XSS.
- HttpOnly cookies: More secure (blocks JS access), but need backend setup.

Explain pros and cons:

```plaintext
localStorage: Simple, but exposed to XSS.
Cookies (HttpOnly): Safer, mitigates XSS, but CSRF must be handled.
```

🔄 6. How do you persist user login (session management)?

Expected:

- Use token in localStorage or cookie.
- On app load, validate token via API.

Code Snippet:

```js
// On App mount
useEffect(() => {
  const token = localStorage.getItem('token');
  if (token) {
    // Optional: validate token with backend
    fetch('/api/validate', {
      headers: { Authorization: `Bearer ${token}` }
    });
  }
}, []);
```

🧪 7. How do you handle token expiration / auto logout?

What they expect:

- Use token expiry claim (`exp` in JWT).
- Auto logout or refresh token.

Example with JWT decode:

```js
import jwtDecode from 'jwt-decode';

const isTokenExpired = (token) => {
  const { exp } = jwtDecode(token);
  return Date.now() >= exp * 1000;
};
```

```
if (isTokenExpired(token)) {
  logout(); // clear storage, redirect
}
```

🔁 8. What is the difference between access token and refresh token?

Expected Answer:

- Access token: Short-lived, used to access APIs.
- Refresh token: Long-lived, used to get new access tokens.

Bonus: Explain refresh flow (backend issues new access token).

🔥 Bonus: Common Mistakes They Might Ask About

- Storing sensitive data in localStorage.
- Not validating JWT on the backend.
- Forgetting to remove tokens on logout.

Would you like me to prepare a mock interview around this for practice? Or maybe help with a sample mini-auth React app you can show in interviews?

Absolutely! Here are **some more advanced and practical questions** related to **authentication/authorization** that companies like **Infosys, TCS, Wipro** may ask you during a React interview — especially since you have **3 years of experience**. These questions test not only your implementation skills but also your **real-world understanding**.

🧲 9. How do you handle authentication in Single Page Applications (SPAs)?

Expected Answer:

- SPAs use client-side routing. Backend doesn't reload the page.
- So, you store token (JWT) and send it in headers for each API call.
- Use `React Context` or `Redux` for storing user state.

Code Snippet – Context-based Auth:

```js
// AuthContext.js
const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const login = (userData) => setUser(userData);
  const logout = () => setUser(null);

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
```

🛡️ 10. How do you secure API calls from React to the backend?

Expected Answer:

- Use HTTPS for all calls.
- Always send `Authorization: Bearer <token>` headers.
- Backend should verify the token (JWT or session).
- Use HttpOnly cookies if you're worried about XSS.

Code Snippet:

```js
fetch('/api/user-profile', {
  headers: {
    'Authorization': `Bearer ${localStorage.getItem('token')}`
  }
});
```

👮 11. How would you handle multi-role UI in React?

Expected:

- Based on the user's role, render different menus, pages, or components.
- Use centralized role check logic (e.g., helpers or hooks).

Code Snippet – Role-based Access:

```js
const checkPermission = (user, requiredRole) => user?.role === requiredRole;

// Example usage
{checkPermission(user, 'admin') && <AdminTools />}
```

🧩 12. How do you prevent unauthorized access to APIs?

Expected Backend Awareness:

- Even if frontend hides buttons or pages, user could call API manually.
- So, backend must **always** verify user token and role.

You should say:

> I ensure that my React app only shows authorized UI, but critical access control is always enforced on the backend using token validation and role checks.

🚫 13. How do you implement Logout functionality securely?

Expected:

- Remove token from storage.
- Optionally notify backend to invalidate refresh token.
- Redirect to login page.

Code Snippet:

```js
const logout = () => {
  localStorage.removeItem('token');
  navigate('/login');
};
```

🔐 14. How do you implement social login (Google, Facebook) in React?

What they expect:

- Use OAuth2.
- Use libraries like `react-oauth/google` or Firebase Auth.
- Get token from Google/Facebook → send it to backend for verification.

Code Snippet (Google OAuth):

```js
import { GoogleLogin } from '@react-oauth/google';

<GoogleLogin
  onSuccess={credentialResponse => {
    // send credentialResponse.credential to backend
  }}
```

```
  onError={() => console.log('Login Failed')}
/>
```

🌍 15. How do you handle authentication across tabs and refresh?

Expected:

- Use `localStorage` or `cookies` to persist login.
- For logout sync, use the `storage` event to sync between tabs.

Code Snippet – Multi-tab logout:

```js
useEffect(() => {
  const handleStorageChange = () => {
    if (!localStorage.getItem('token')) {
      window.location.href = '/login';
    }
  };
  window.addEventListener('storage', handleStorageChange);
  return () => window.removeEventListener('storage', handleStorageChange);
}, []);
```

🧬 16. What is CSRF and how is it different from XSS? How do you prevent them?

Expected Summary:

- **XSS**: Malicious scripts on frontend (protect with CSP, sanitize inputs).
- **CSRF**: Forged requests from another site (use SameSite cookies or CSRF tokens).

Bonus: If you're using cookies for authentication, set:

```plaintext
```

```
Set-Cookie: token=abc123; HttpOnly; SameSite=Strict; Secure
```

💬 17. What libraries or tools have you used for Auth?

Be honest, but popular answers:

- `jsonwebtoken`, `bcrypt`, `express-session`
- `Firebase Auth`
- `Auth0`
- `react-oauth/google`
- `react-router-dom` for route protection
- `Redux Toolkit` or Context for storing user state

If you want, I can also give:

- A **React mini-project repo structure** for auth
- A **custom hook for useAuth**
- A **set of MCQs for self-test**
- **Mock interview questions (Q&A style)**

Let me know what helps you best next 🔥