HOW SOFTWARE WORKS · BUILDER · 12 MIN READ

Sign-in explained: accounts, sessions and permissions

Authentication vs authorization, the ways people sign in (passwords, magic links, Google, passkeys), how sessions work, and why you should never build it yourself.

Pass the check (4 of 5) online to earn the “Gatekeeper” badge:
https://digitalbuilders757.com/university/study/sign-in-explained

Two different questions

Authentication asks 'who are you?' (signing in). Authorization asks 'what are you allowed to do?' (permissions). An app needs both: a signed-in member still shouldn't see the admin page or another member's data. Mixing them up is one of the most common security bugs in AI-built apps.

Ways to sign in

- Email and password, ideally with multifactor authentication (a code or app prompt as a second step).
- Magic links: a one-time sign-in link sent to the user's email; no password to forget or leak.
- Sign in with Google, Apple or GitHub (OAuth): another provider vouches for the user.
- Passkeys: the device unlocks with a fingerprint, face or PIN, and nothing reusable is typed or sent, so they resist phishing.

Sessions: staying signed in

After signing in, the app gives the browser a session, usually a token stored in a cookie. Each request carries it, so the server knows who's asking. Sessions expire for safety, which is why you sometimes see 'session expired' or 'JWT expired' and need to sign in again. Signing out should end the session on the server too, not just hide the menu.

Don't build it yourself

Use a proven auth service such as Supabase Auth, Clerk or Auth0 instead of storing passwords yourself. Then check permissions on the server and in the database (row level security), never only by hiding buttons in the browser. Give people a safe password reset, rate-limit sign-in attempts, and collect only the account details you need.

Key terms

Authentication
Checking who someone is (signing in).

Authorisation
Checking what a signed-in person is allowed to do.

Session
Proof you've signed in, usually a token in a cookie, sent with each request until it expires.

OAuth (sign in with Google)
Signing in through another provider, like Google or GitHub, which vouches for who you are.

Magic link
A one-time sign-in link sent to your email, so there's no password to remember.

Passkey
A sign-in your device unlocks with a fingerprint, face or PIN; nothing reusable is typed, so it resists phishing.

Two-factor authentication (2FA)
A second check at sign-in, like a code from an app, on top of your password.

Row level security (RLS)
Database rules that decide which rows each user may read or change.

Try it: Map your app's gates

1. List every page in your app and write who may see it: anyone, signed-in members, or admins.
2. For each, note where the check happens. If it's only in the browser, ask your AI helper to add a server or database check.
3. Sign out, then try to open a members-only page directly by its URL.
4. Turn on multifactor authentication for your own accounts on the services your app uses.

KNOWLEDGE CHECK

Sign-in explained: accounts, sessions and permissions

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. What's the difference between authentication and authorization?

A. They mean the same thing
B. Authentication is for websites; authorization is the same check for mobile apps
C. Authentication is who you are; authorization is what you can do
D. Authentication is only for admins

2. The admin button is hidden for members, but the admin page has no other check. Is that safe?

A. Yes, members can't see the button
B. Yes, if the page is dark mode
C. Only on phones
D. No: anyone who types the URL or calls the API can get in; check on the server and in the database

3. What makes passkeys resistant to phishing?

A. Your device proves who you are, and nothing reusable is typed
B. They're very long passwords that are too hard for anyone to guess
C. They're one-time codes sent by SMS to your phone each time you sign in
D. They never expire and can be shared with family safely

4. Why do sessions expire?

A. To annoy users
B. So a stolen or forgotten session can't be used forever
C. Because browsers delete all cookies every day to save space
D. They don't, ever

5. Best way to add sign-in to a new app?

A. Store passwords in a spreadsheet
B. Write your own password encryption
C. Use a proven auth service and enforce permissions on the server and database
D. Skip sign-in and trust the browser

Official sites

Supabase docs: Auth: https://supabase.com/docs/guides/auth
OWASP: Authentication Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

Sign-in explained: accounts, sessions and permissions

1. C: Authentication is who you are; authorization is what you can do
Sign-in proves identity; permissions decide access. Apps need both.

2. D: No: anyone who types the URL or calls the API can get in; check on the server and in the database
Hiding buttons is not security. Permissions must be enforced server-side.

3. A: Your device proves who you are, and nothing reusable is typed
There's no password for a fake page to capture.

4. B: So a stolen or forgotten session can't be used forever
Expiry limits the damage from a leaked or abandoned session.

5. C: Use a proven auth service and enforce permissions on the server and database
Auth is hard to get right; proven services handle the risky parts.