

IDENTITY / SESSIONS / PRIVATE ROWS

SUPABASE AUTH

FROM EMAIL LINK TO AUTHORIZED DATA



One simple rule connects an account to its data:
SIGNED-IN USER ID = ROW OWNER ID

TWO JOBS, TWO ANSWERS**WHO ARE YOU?****AUTHENTICATION**

Supabase verifies a login method and creates a user session.

WHAT CAN YOU USE?**AUTHORIZATION**

Postgres policies decide which rows that session may access.

PASSWORDLESS LOGIN**THE MAGIC-LINK LOOP**

The user enters an email. Supabase sends a one-time link. Clicking it returns the browser to an allowed URL with a signed session.

**ONE-TIME**

The link is designed for a single sign-in.

**ALLOWED REDIRECT**

The destination must be configured by the project owner.

01

ENTER EMAIL

The app requests a magic link.



02

CLICK LINK

Supabase verifies the one-time token.



03

GET SESSION

The browser now carries user identity.



04

QUERY DATA

RLS checks that identity on every request.

THE OWNERSHIP MATCH

AUTHENTICATED USER

8f31...a902

=

PRIVATE_IDEAS.OWNER_ID

8f31...a902

✓ MATCH: ALLOW

The user may read or update the row.

× NO MATCH: BLOCK

Signed-out users and other accounts receive nothing.

ONE TABLE. PRIVATE BY DEFAULT.

The website uses a publishable key. The signed session identifies the user. Grants and Row Level Security decide what happens next.

PRIVATE_IDEAS

id	automatic row ID
owner_id	authenticated user UUID
title	the user's idea
status	idea / building / finished
created_at	automatic timestamp

OWNER_ID is the bridge between AUTH and DATA.

```
create policy "Users can read their own ideas"
on public.private_ideas for select
to authenticated
using ((select auth.uid()) = owner_id);

create policy "Users can update their own ideas"
on public.private_ideas for update
to authenticated
using ((select auth.uid()) = owner_id)
with check ((select auth.uid()) = owner_id);
```

SIGNED OUT

Database refuses the private query.

MAGIC LINK

Email ownership is verified.

ADD ROW

auth.uid() becomes owner_id.

EDIT ROW

Matching owner passes RLS.

SIGN OUT

The same row becomes invisible.

SAFE IN BROWSER CODE

PUBLISHABLE KEY

Identifies the Supabase project. It is intended for clients when database grants and RLS are correctly configured.

NEVER EXPOSE

SERVICE-ROLE KEY

Privileged and able to bypass RLS. Keep it only on trusted servers - never in a public webpage or recording.

RECORDING + PRODUCTION CHECKLIST

- ✓ Add the exact redirect URL before testing.
- ✓ Grant only the operations the app needs.
- ✓ Test while signed out and signed in.
- ✓ Avoid repeated magic-link requests.

- ✓ Enable RLS on exposed tables.
- ✓ Write separate read, insert and update policies.
- ✓ Use a recording-safe email address.
- ✓ Keep privileged keys off the frontend.

OFFICIAL SOURCES: [Auth](#) [Magic links](#) [Redirect URLs](#) [Row Level Security](#)

