

FINISH LINE SCAN · TECHNICAL

Your app works. It's not safe to launch yet.

A prioritized list of what's blocking launch, what to fix in the first month, and what can safely wait.

Product	Homeowner finance app (anonymized)
Built with	Cursor + Claude · Next.js 16, React 19, Supabase Postgres, Drizzle
Scan type	Technical Finish Line Scan (read-only repo access)
Reviewed by	Naveen, StartBuilds
Turnaround	Delivered in 41 hours

Launch readiness

BLOCKED

5

Fix before launch

4

Fix in first 30 days

4

Can wait

SUMMARY

The short version

The product looks finished and most screens are polished. Under the hood, the parts that protect real users and their financial data aren't built yet. None of this is unusual for an AI-built app, and all of it is fixable.

If you only read three things

1. **Anyone can open the app without logging in.** The login page is a design, not a lock. Every dashboard route is public.
2. **Your database is reachable from outside the app.** Supabase's public API is exposed without row-level rules, and an open endpoint can wipe and rebuild data.
3. **Some screens look done but aren't connected to anything.** Real users would enter data into screens that don't save it.

#	FINDING	PRIORITY	EFFORT
01	Login is UI-only; all app routes are public	BEFORE LAUNCH	1.5–2 days
02	Seed endpoint can wipe/rebuild data, no auth	BEFORE LAUNCH	1 hour
03	Supabase public Data API exposed, RLS not enforced	BEFORE LAUNCH	0.5–1 day
04	Credentials committed to repo history	BEFORE LAUNCH	2 hours
05	Mock-backed screens look like working features	BEFORE LAUNCH	0.5 day
06	Database changes can't be safely rolled back	FIRST 30 DAYS	1 day
07	No background job queue for provider syncs	FIRST 30 DAYS	1.5–2 days
08	5 TypeScript errors; build doesn't block on them	FIRST 30 DAYS	2–3 hours
09	No error tracking or request IDs	FIRST 30 DAYS	0.5 day
10	Search & pagination won't scale past demo data	CAN WAIT	0.5 day
11	Dashboard queries run one after another	CAN WAIT	2 hours
12	Image cache resets on every cold start	CAN WAIT	0.5 day
13	Duplicated helpers & unused dependencies	CAN WAIT	2–3 hours

Launch blockers

These put real user data at risk or would break trust on day one. Don't onboard users until these are done.

01 Login is UI-only; all app routes are public

BLOCKER`/login · /signup · /app/*`

WHAT WE FOUND

The login and signup pages are static forms. The users table is never populated, and every route under /app loads without a session.

WHY IT MATTERS

For a finance product, this means any visitor with the URL sees dashboards. There's also no way to say "this record belongs to this user."

Fix: Add real auth (Supabase Auth or Auth.js), protect /app/* in middleware, and check ownership on the server for every record, not just in the UI.

02 Seed endpoint can wipe and rebuild data with no auth

BLOCKER`POST /api/v1/seed`

WHAT WE FOUND

The demo-data endpoint is unauthenticated. With write credentials configured, any request to it rebuilds the dataset.

WHY IT MATTERS

One curl command from a stranger could reset production data. These endpoints get found by automated scanners within days.

Fix: Remove it from production builds entirely. Seed through a CLI script that only runs locally or in CI.

03 Supabase public Data API exposed, RLS not enforced

BLOCKER`Supabase · public schema`

WHAT WE FOUND

Tables live in the public schema with the Data API enabled. Row-level security isn't enabled on them, so the anon key in the frontend can read them directly.

WHY IT MATTERS

Your app's own permissions don't matter if the database answers anyone holding the public key. This is the #1 issue we see in AI-built apps.

Fix: Enable RLS on every table with owner-based policies, or disable the Data API since the app queries through Drizzle on the server anyway.

04 Credentials committed to repo history

BLOCKER

git history · .env files

WHAT WE FOUND

.env is now gitignored, but earlier commits contain database credentials and a provider token.

WHY IT MATTERS

Deleting the file doesn't remove it from history. Anyone with repo access, now or later, has working keys.

Fix: Rotate every exposed key first. Then move secrets to your host's secret manager and add a pre-commit secret scanner.

05 Mock-backed screens look like working features

BLOCKER

workspace · settings · public share pages

WHAT WE FOUND

Workspace, settings, privacy and share screens are fully designed but read from static mock files. Nothing entered there is saved.

WHY IT MATTERS

A user who updates privacy settings and finds them unchanged won't trust you with their bank data. Visually complete isn't the same as working.

Fix: Hide unconnected screens behind a feature flag for launch. Ship fewer screens that all work.

Safe to launch, but don't wait long

These won't hurt users on day one. They'll hurt you the first time something breaks in production.

06 Database changes can't be safely rolled back

30 DAYS

Schema changes run as one "create if not exists" batch. It can't rename, drop or change columns safely. **Fix:** switch to versioned Drizzle migrations with a staging run before production.

07 No background job queue for provider syncs

30 DAYS

Banking and property data will sync inside web requests. When a provider is slow or down, users see a spinner or an error with no retry. **Fix:** add a durable queue with retries, backoff and idempotency, and show "last synced" in the UI.

08 5 TypeScript errors; build doesn't block on them

30 DAYS

Includes a wrong prop on the profile page and a return type missing required fields. Each is small, but ignored type errors are where AI-introduced bugs hide. **Fix:** fix all five and make typecheck a CI gate.

09 No error tracking or request IDs

30 DAYS

If a user reports "it broke," there's no way to see what happened without reproducing it locally. **Fix:** add Sentry (or similar), structured logs and a request ID on every API call and job.

CAN WAIT

Fine for now

Real, but not worth your time until you have real traffic.

#	FINDING	NOTE
10	Search & pagination won't scale past demo data	LIMIT/OFFSET and ILIKE are fine now. Revisit around 50k+ rows.
11	Dashboard queries run one after another	Run independent queries in parallel. Easy speed win later.
12	Image cache resets on cold start	Move to object storage/CDN if cold starts get noticeable.
13	Duplicated helpers & unused deps	Housekeeping. Clean up when you next touch those files.

RECOMMENDED PLAN

How we'd get you to launch

Blockers first, in this order, so the riskiest gaps close in the first two days.

- Day 1** **Close the open doors.** Rotate leaked keys, remove the seed endpoint, lock down Supabase (RLS or Data API off).
- Days 2–3** **Real auth.** Sessions, protected routes, server-side ownership checks, and tests that prove user A can't see user B's data.
- Day 4** **Honest product surface.** Flag off mock-backed screens, fix type errors, add CI gate.
- Days 5–7** **Production basics.** Versioned migrations, staging environment, error tracking, request IDs.
- Optional** **Integration spine.** Background queue with retries for the first banking or property provider. Scoped separately once the provider is chosen.

FINISH LINE FIX · FINDINGS 01–09

\$1,000 fixed

About 7 working days · Scope and price agreed before work starts · Read-only access until you approve

[Book a 15-min call to go through this →](#)



Naveen

Founder, StartBuilds · 12+ years full-stack (Next.js, TypeScript, Node, Postgres)

You can also take this list to your own developer, or fix it yourself with Cursor. The findings are yours either way. Sample report: product details anonymized; pricing shown is illustrative.