Ascertify

Code Audit Report

Scan ID: sample
Generated: July 28, 2026 at 03:52 PM GMT+3
View online →

Static analysis · no code executed

Your scan report

Scanned **Jul 21, 2026** Source **GitHub**

We found *3 critical issues*, plus 2 items worth raising with your developer.

3	2	2	5
Critical	Needs attention	Notes	Passed

Static review only — we didn't run the app or test live behavior. See "What we checked, and what we didn't" below.

Critical

These are specific, verifiable, and hard to argue with. Resolve them before you sign off on this build.

Critical — exposed service key

exposed src/lib/supabase.ts:14

```
14  const db = createClient(url, "eyJhbGci…service_role…0xR2")
```

Your secret database admin key is sitting in code that ships to the browser. It bypasses every access rule you have. Anyone who opens your site can read or change all of your data.

Recommended first step: Rotate the key in your Supabase dashboard, then move it to a server-only environment variable. Never reference it from frontend code.

Critical — RLS disabled

exposed supabase/migrations/0007_init.sql:41

```
41  ALTER TABLE public.users DISABLE ROW LEVEL SECURITY;
```

Row-level security is switched off on the users table. Any signed-in visitor can read every row — names, emails, everything — not just their own record. The app works fine in testing because you're the only user.

Recommended first step: Enable RLS on the users table with a policy that limits each user to their own rows, then verify with two separate accounts.

Critical — live credentials

exposed src/lib/openai.ts:6

```
6  const openai = new OpenAI({ apiKey: "sk-proj-8kJd…Wq2A" });
```

A live OpenAI API key is committed to the repository. Anyone with access to the code — or its git history — can copy it and run up your bill.

Recommended first step: Revoke this key now, create a new one, and load it from a server-only environment variable.

Needs attention

Not emergencies, but real. These shape what it will cost you to keep, replace, or hand off this codebase.

Needs attention **hallucinated dependencies** what got built package.json:28

```
28  "react-auth-helper-pro": "^2.1.4",
```

This package doesn't exist in the npm registry. The app can't be installed cleanly from scratch, and a plausible-sounding name like this is exactly what malware squatters target.

Ask your developer: Why is a package that doesn't exist in any registry listed as a dependency, and what code was supposed to use it?

Needs attention **no tests** what got built

The project contains no test files at all. Nothing verifies the app still works after a change, which makes every future edit a gamble.

Ask your developer: What's the plan for catching regressions before users do, and what would it take to cover the checkout flow first?

Notes & signals

Softer signals. We report what we observed, not what it means — read these as questions to ask, not conclusions to draw.

Note **stale dependencies** maintainability package.json:31

Three dependencies haven't been updated in over two years. Old packages accumulate known issues and get harder to upgrade the longer they sit.

Note **code duplication** maintainability

About 18% of the code consists of duplicated blocks. Fixes have to be applied in several places, which is where inconsistencies creep in.

Passed

Checks we ran that came back clean. We show these so you know what the result above does and doesn't cover.

- ✓ No SQL injection patterns found
- ✓ No known CVEs in direct dependencies
- ✓ Database migrations present
- ✓ Lockfile committed
- ✓ No circular dependencies

What we checked, and what we didn't

✓ We checked

- Hardcoded secrets and API keys
- SQL injection patterns
- eval() usage
- Unfiltered Supabase queries
- Test presence and coverage ratio
- README present
- npm scripts complete (dev, build)
- Lockfile committed
- .env.example present
- Circular dependencies
- Code complexity hotspots
- Code duplication
- Unused or dead dependencies
- Undeclared dependencies (imported but missing from package.json)
- Boilerplate ratio
- Firebase rules publicly open
- Row-level security disabled
- Supabase service-role key in frontend code
- Database migrations present
- Unsafe migrations
- Dangerous cascade deletes
- Missing foreign-key indexes
- Boilerplate-only schema
- Known CVEs in direct dependencies
- Deprecated or abandoned dependencies
- Undeclared imports verified against the npm registry

✕ We did not check

- Commit count and delivery pattern
- All commits in 24 hours
- Single-author history
- We did not run your application or execute its tests
- Whether your features actually work as intended
- Runtime login/permission behavior
- Performance, scale, or hosting configuration
- Business-logic correctness
- Anything that requires a live, running system

This is a static review, not a penetration test. A clean result here is not a guarantee that your application is secure — it means the specific things we checked, listed above, came back clean. We report only what we can point to in your code.
