GITREX TECHNOLOGIES

REPORT ID: 3651C122

July 30, 2026

LAUNCH READINESS AUDIT

HealthOS

SECURITY AUDIT

25/100

LAUNCH READINESS SCORE

One honest score that tells you exactly how far your app is from production — no guessing, no wishful thinking.

EXECUTIVE SUMMARY

This security audit of HealthOS identified 10 findings (5 high, 2 medium, 3 low) — 7 tied to Database/Auth/Deployment security and 3 to code architecture (Section I). The overall security posture is at significant risk with a readiness score of 25/100. The findings and prioritized remediation steps are detailed below.

⚠ PARTIAL AUDIT COVERAGE

RLS Configurations was not performed for this audit (No DB connection string) — related findings above should not be read as a confirmed clean bill of health.

Live Endpoint Verification was not performed for this audit (No deployed app URL provided) — related findings above should not be read as a confirmed clean bill of health.

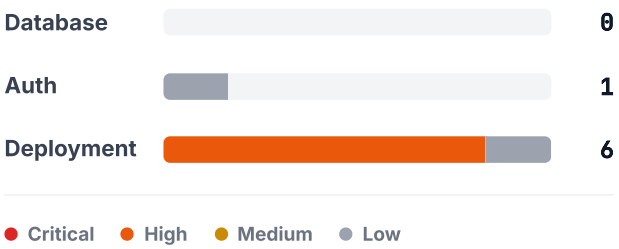
FINDINGS OVERVIEW

Security findings only (Database/Auth/Deployment) — 3 additional Code Architecture findings are covered in Section I below.

BY SEVERITY

High	5	71%
Low	2	29%

BY CATEGORY



I

Codebase & Architecture Review

A senior engineer reads your actual code — structure, maintainability, and the shortcuts your AI builder took that will bite you later.

Our review of HealthOS's codebase covered structure, maintainability, and the kinds of shortcuts AI builders take that don't show up as "vulnerabilities" but still slow teams down or cause outages later — inconsistent data-access patterns, missing error boundaries, and tech debt that compounds as the app grows.

The automated pass did not surface obvious AI-scaffolding red flags, though a manual read of the codebase remains the best way to catch structural and maintainability issues that pattern-based scanning can't see.

10 checklist item(s) in this audit call for judgment-based manual review rather than an automated scan — architecture quality is exactly that kind of call, and it's reflected in the findings and roadmap below.

ARCHITECTURE FINDINGS (3)

1

SEVERITY: MEDIUM ×5 OCCURRENCES

ID: DD18277C-01

Oversized module

The file src/routeTree.gen.ts is a generated route tree file that is inherently large due to listing all routes. Splitting it manually is impractical because it's auto-generated; instead, consider configuring the code generator to produce smaller chunks or suppress this warning for generated files.

Found in **5 locations**:

`src/routeTree.gen.ts` `src/routes/onboarding.tsx` `src/routes/chat.tsx` `src/routes/matches.tsx` `src/components/games/FreeYourMindHeartGame.tsx`

REMEDIATION

```
Split the file into smaller, single-responsibility modules with clear boundaries.
```

2

SEVERITY: MEDIUM

ID: 6AD72BC9-02

No automated tests found in the repository

No test files were found in the repository, which increases risk of regressions and makes refactoring unsafe.

`Target: Code Architecture`

REMEDIATION

```
Add a test runner and start covering critical paths and regressions.
```

3

SEVERITY: LOW ×6 OCCURRENCES

ID: B5AD1FBE-03

Heavy use of `any`

The route tree file uses `any` extensively, which disables type checking for route definitions and could hide mismatches between routes and their handlers.

Found in **6 locations**:

`src/routeTree.gen.ts` `src/routes/matches.tsx` `src/lib/ai-compatibility.functions.ts`
`src/lib/journey.functions.ts` `src/routes/messages.tsx` `src/hooks/use-unread.ts`

REMEDIATION

```
Replace `any` with precise types, `unknown` + narrowing, or generated types.
```

II

Risk Report — Database, Auth & Deployment

Security rules, auth edge cases, and deployment stability — the three places AI-built apps fail hardest when real users arrive.

DATABASE

0

ISSUES FOUND

Row-level security and data access — the #1 place AI-built apps leak data.

— Not Assessed

AUTH

1

ISSUE FOUND

Session handling, role escalation, and auth edge cases.

1 LOW

DEPLOYMENT

6

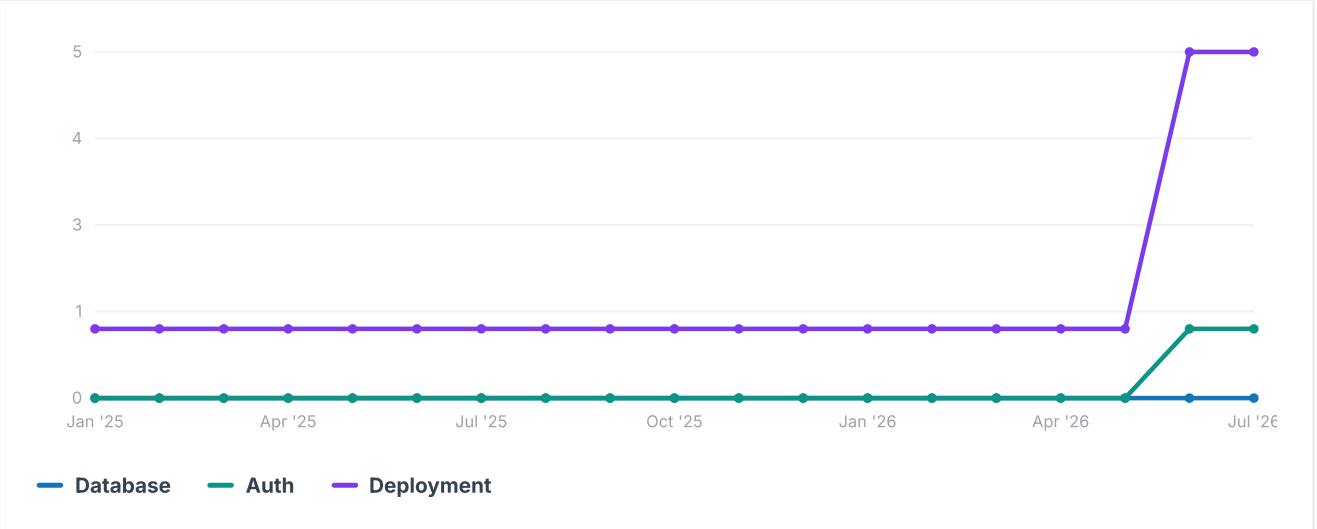
ISSUES FOUND

Exposed keys, API surface, injection surfaces, and framework defaults.

5 HIGH 1 LOW

ISSUES INTRODUCED OVER TIME

Cumulative issues per pillar, traced back to the commit that first introduced the affected file.



This chart traces each open finding back to the commit that first introduced its source file, then plots how issues accumulated across the 19 month(s) of history available. Deployment currently accounts for the most open, datable issues (5). The sharpest increase was in June 2026, when 5 new issues were introduced at once — usually a sign a specific feature push or refactor landed several risky patterns together, rather than the codebase degrading gradually. Findings without a specific source file — most notably database RLS/schema checks, which are evaluated live rather than read from a commit — aren't reflected in this chart, so its totals can run lower than the pillar counts above.

III

Priority Issue List

Every issue we find, ranked by launch impact — so you fix what matters first instead of polishing what does not.

1 SEVERITY: HIGH ID: 6693E0F6-01

Env file tracked in the repository: .env.development

.env.development is tracked, exposing VITE_PAYMENTS_CLIENT_TOKEN. Even though it's a development file, it may contain real credentials and should not be committed.

`Target: .env.development`

REMEDIATION

```
Remove from tracking, add to .gitignore, rotate anything that leaked, keep only .env.example.
```

2 SEVERITY: HIGH ID: 06F7DAFD-02

Env file tracked in the repository: .env.production

.env.production is tracked, exposing production credentials like VITE_PAYMENTS_CLIENT_TOKEN and Supabase keys. This is a serious leak; immediate rotation and removal from git history is required.

`Target: .env.production`

REMEDIATION

```
Remove from tracking, add to .gitignore, rotate anything that leaked, keep only .env.example.
```

3 SEVERITY: HIGH ID: C05D5B2E-03

Unsanitized HTML injection

Dynamic content is rendered as raw HTML (dangerouslySetInnerHTML/innerHTML/v-html) rather than a static literal. If any part of it can be influenced by user input, sanitize it with a vetted library (e.g. DOMPurify) or avoid raw HTML.

`Target: src/components/ShareablePassportCard.tsx`

REMEDIATION

```
Avoid raw HTML injection; sanitize with a vetted library (DOMPurify) if unavoidable.
```

4 SEVERITY: HIGH ID: 59157F40-04

Known vulnerability in vite@7.3.2

vite@7.3.2 matches 2 known advisory(ies) in OSV.dev: vite: `server.fs.deny` bypass on Windows alternate paths; launch-editor: NTLMv2 hash disclosure via UNC path handling on Windows. (CVE-2026-53571, CVE-2026-53632)

Target: package.json

REMEDIATION

```
Upgrade the package to a patched version (see the linked advisory for the minimum safe version).
```

5

SEVERITY: HIGH

ID: D3D0F699-05

Env file tracked in the repository: .env

The .env file is tracked in the repository, exposing credentials like SUPABASE_URL and SUPABASE_SERVICE_ROLE_KEY (though redacted in snippet). This is a real risk; the remediation to remove from tracking and rotate secrets applies.

Target: .env

REMEDIATION

```
Remove from tracking, add to .gitignore, rotate anything that leaked, keep only .env.example.
```

6

SEVERITY: MEDIUM

×5 OCCURRENCES

ID: DD18277C-06

Oversized module

The file src/routeTree.gen.ts is a generated route tree file that is inherently large due to listing all routes. Splitting it manually is impractical because it's auto-generated; instead, consider configuring the code generator to produce smaller chunks or suppress this warning for generated files.

Found in **5 locations**:

`src/routeTree.gen.ts` `src/routes/onboarding.tsx` `src/routes/chat.tsx` `src/routes/matches.tsx` `src/components/games/FreeYourMindHeartGame.tsx`

REMEDIATION

```
Split the file into smaller, single-responsibility modules with clear boundaries.
```

7

SEVERITY: MEDIUM

ID: 6AD72BC9-07

No automated tests found in the repository

No test files were found in the repository, which increases risk of regressions and makes refactoring unsafe.

Target: Code Architecture

REMEDIATION

```
Add a test runner and start covering critical paths and regressions.
```

8

SEVERITY: LOW

ID: 0E561984-08

No security headers configured

Security headers like CSP, HSTS, and X-Content-Type-Options are missing from the app/hosting configuration, leaving the app vulnerable to common attacks.

Target: API Surface

REMEDIATION

```
Add a baseline header set (CSP, HSTS, X-Content-Type-Options, Referrer-Policy).
```

9

SEVERITY: LOW ×6 OCCURRENCES

ID: B5AD1FBE-09

Heavy use of `any`

The route tree file uses `any` extensively, which disables type checking for route definitions and could hide mismatches between routes and their handlers.

Found in **6 locations**:

`src/routeTree.gen.ts` `src/routes/matches.tsx` `src/lib/ai-compatibility.functions.ts` `src/lib/journey.functions.ts` `src/routes/messages.tsx` `src/hooks/use-unread.ts`

REMEDIATION

```
Replace `any` with precise types, `unknown` + narrowing, or generated types.
```

10

SEVERITY: LOW

ID: CA2789E8-10

Session tokens in localStorage

The snippet shows localStorage is used for auth session persistence, which is standard for client-only SPAs without a server. The risk is XSS exposure, but the remediation (CSP, short token lifetimes) is appropriate. The severity is lowered because this is the documented approach and the app likely relies on RLS for data protection.

`Target: src/integrations/supabase/client.ts`

REMEDIATION

```
Add a strict Content-Security-Policy and sanitize all rendered user input to reduce XSS surface; keep access-token/refresh-token lifetimes short. A cookie-session migration is not applicable without a server to hold the session.
```

IV

DIY Fix Roadmap

Step-by-step fixes written so you can execute them yourself — with your AI builder or any developer. No agency required.

1. **Address the 5 high-severity finding(s) this week.**
2. **Schedule the 2 medium-severity item(s) into the next sprint.**
3. **Track the 3 low-severity item(s) as hardening backlog.**
4. **Complete the manual-review checklist and re-run the audit to confirm fixes.**

This roadmap sequences fixes strictly by launch impact: high-severity issues come first since they carry the greatest risk of a launch-blocking failure or data exposure, with lower-severity items following once those blockers are cleared. Each numbered step above corresponds directly to the severity counts in the Priority Issue List, so working through them top to bottom — then re-running the audit to confirm each fix actually closed the underlying finding — is the fastest, most reliable path from today's score to a clean bill of health.

V

What Was Checked & Passed

- ✓ Supabase service_role key committed to the repo
- ✓ Secret shipped via a client-exposed env var
- ✓ Unauthenticated API routes handling sensitive actions
- ✓ Verbose error responses leaking internals
- ✓ Missing server-side input validation
- ✓ Tables created without RLS by AI scaffolding
- ✓ Edge functions deployed with permissive default CORS
- ✓ Unguarded admin/debug routes from templates
- ✓ Fragile deep relative imports
- ✓ Secrets present in git history
- ✓ Third-party provider keys in code or bundle
- ✓ CORS misconfiguration
- ✓ Raw SQL built from unsanitized input
- ✓ OS command execution with user input
- ✓ Supabase client initialized with service_role in frontend
- ✓ Debug/verbose logging left enabled
- ✓ High volume of tech-debt markers

NOT ASSESSED — SCANNER MODULE DID NOT RUN

- — Tables in public schema with RLS disabled
- — Overly permissive policy (USING true)
- — Anonymous role can read sensitive tables
- — RLS enabled but no policies defined
- — Policy does not reference auth.uid()
- — Live-verified unauthenticated route

VI

Audit Scope Summary

CHECKS RUN	AUTOMATED / MANUAL
41	31 / 10
UNIQUE ISSUES	**METHODOLOGY**
10 (19 instances)	Launch Audit Checklist

Want these fixes handled for you?

Ask about our Retainer Plan — ongoing audits, fixes, and launch support from the team that found these issues.