TECHNICAL FINDINGS & REMEDIATION REPORT

FamNest — AI Wellness Coach for Busy Parents

Codebase Audit · Defect Analysis · Remediation Plan

Prepared by:

Virginia Mwega

Full-Stack Developer & Technical Author

virginiamwega2@gmail.com | virginiamwega-com.vercel.app

Document Reference	FN-TFR-2026-001
Version	1.0
Status	FINAL
Classification	CONFIDENTIAL
Date	May 2026
System Under Review	FamNest v1.0 (Production)
Audit Scope	app/, lib/, components/, docs/

1. Document Control

1.1 Version History

Version	Date	Author	Description
0.1	May 2026	V. Mwega	Initial audit findings recorded
1.0	May 2026	V. Mwega	Root-cause analysis and remediation plan finalised

1.2 Purpose

This Technical Findings & Remediation Report documents the results of a structured codebase audit of the FamNest platform (version 1.0, production). It identifies confirmed defects and feature gaps, establishes their root cause and user impact, classifies each by severity, and defines a prioritised remediation plan. The report is intended for the development team, technical reviewers, and stakeholders responsible for release readiness.

> *Methodology: Static review of the full application source tree (app/, lib/, components/) and database schema (docs/schema.sql, migrations). Each finding was traced from trigger to effect and confirmed against the actual code path. No issue is reported on suspicion alone — every finding below was verified.*

2. Executive Summary

The audit confirmed four issues and one defense-in-depth observation. Two issues are classified HIGH severity: both are silent failures — they produce no error, give the user the appearance of success, and are therefore unlikely to be caught without targeted testing. This silent-failure characteristic is the primary risk theme of this report.

2.1 Findings at a Glance

ID	Finding	Category	Severity
TFR-01	Edit & delete check-in silently affect zero rows	Data Integrity	HIGH
TFR-02	Password reset dead-ends on a 404	Authentication	HIGH
TFR-03	Preferred reminder time is ignored	Feature Gap	MEDIUM
TFR-04	Primary conversion analytics event never fires	Analytics	MEDIUM
OBS-01	Read paths rely solely on RLS (no code-level filter)	Defense in Depth	OBSERVATION

2.2 Severity Classification Scheme

Severity	Definition
HIGH	Causes data loss, data corruption, or blocks a critical user journey (auth, core action). Silent failures are escalated to HIGH.
MEDIUM	Degrades a promised feature or impairs business-critical measurement, without data loss or hard blockage.
LOW	Cosmetic or minor; no functional impact.
OBSERVATION	Not a defect under current configuration, but a latent risk worth hardening.

3. Detailed Findings

TFR-01 Edit & delete check-in silently affect zero rows [HIGH]

Category	Data Integrity
Root cause	schema.sql enables Row-Level Security on daily_checkins and recommendations but defines only SELECT and INSERT policies. No UPDATE or DELETE policy exists, and no migration adds one. Under PostgreSQL RLS, an UPDATE or DELETE with no matching policy affects zero rows and returns no error — the operation reports success while doing nothing.
Effect — delete	deleteCheckin() redirects to /dashboard as though the deletion succeeded. The check-in is never removed. The user believes their data is gone; it is not.
Effect — update	updateCheckin() does not persist edits. The original recommendation is not deleted, while a new recommendation row is inserted — producing two recommendation rows for a single check-in. The plan page subsequently calls .maybeSingle(), which errors on multiple rows, and the user is shown the “couldn’t generate your plan” fallback permanently.

Affected files	`docs/schema.sql`
	`app/actions/checkin.ts`
	`app/(app)/recommendations/[id]/page.tsx`

User impact	Data the user expects to control (their own check-ins) cannot be edited or deleted. The update path additionally corrupts the recommendation state, permanently breaking the plan view for the affected check-in.
Remediation	Add UPDATE and DELETE RLS policies scoped to the owning user for both tables.

```
CREATE POLICY "checkins_update_own" ON daily_checkins
  FOR UPDATE USING (auth.uid() = user_id) WITH CHECK (auth.uid() = user_id);
CREATE POLICY "checkins_delete_own" ON daily_checkins
  FOR DELETE USING (auth.uid() = user_id);
-- repeat equivalently for the recommendations table
```

Additional hardening: enforce a single recommendation per check-in at the schema level (UNIQUE constraint on recommendations.checkin_id) so the .maybeSingle() contract cannot be violated even if a future code path misbehaves.

TFR-02 Password reset dead-ends on a 404 [HIGH]

Category	Authentication
Root cause	forgotPassword() directs the user to /auth/callback?next=/settings/reset-password, but no route exists at /settings/reset-password. The user authenticates via the emailed link successfully, then lands on a 404 with no means to set a new password.
Affected file	`app/actions/auth.ts:100` (target route absent)
User impact	Any user who forgets their password is unable to recover their account. This blocks a critical authentication journey entirely and will generate support burden and churn.
Remediation	Build the /settings/reset-password route containing a form that calls Supabase updateUser() to set the new password; or, as an interim measure, repoint the next parameter at an existing authenticated page that exposes password update.

Compliance relevance: a functioning credential-recovery path supports the access-management expectations under HIPAA §164.312(d) (person/entity authentication) for the US market. Worth flagging in the compliance matrix once resolved.

TFR-03 Preferred reminder time is ignored [MEDIUM]

Category	Feature Gap / Expectation Mismatch
Root cause	preferred_time is collected from the user, stored, displayed back in the UI, and explicitly promised in UI and email copy (“sent within an hour of this”). However, no scheduled job reads the field. Daily reminders are dispatched to the entire eligible base on a fixed schedule of 0 14 * * * UTC.
Affected files	`app/api/cron/daily-reminder/route.ts` `app/api/cron/weekly-report/route.ts`
User impact	The product makes an explicit promise it does not keep. Reminders arrive at a fixed UTC time regardless of the user’s stated preference, undermining trust in a product whose core value is respecting the user’s time.
Remediation	Either (a) filter and dispatch reminders by preferred_time so delivery honours the stored preference, or (b) remove the per-user timing promise from UI and email copy until per-user scheduling is implemented. Option (a) is preferred; option (b) is an acceptable honest interim.

TFR-04 Primary conversion analytics event never fires [MEDIUM]

Category	Analytics / Measurement
Root cause	The checkin_submitted event is declared in the AnalyticsEvent union type but has no corresponding track() call site anywhere in the codebase. The product’s primary conversion action is therefore entirely untracked.
Affected file	`lib/analytics.ts:15`
User impact	No direct user impact. Business impact is significant: the core conversion funnel cannot be measured, making it impossible to assess activation, retention, or the effect of changes to the check-in flow.
Remediation	Fire track('checkin_submitted') after a successful submission. Recommended pattern: redirect with a ?checkin=success parameter and use a TrackOnMount component, mirroring the existing, working subscription_purchased implementation.

4. Observation — Defense in Depth

OBS-01 Read paths rely solely on RLS with no code-level user filter [OBSERVATION]

Read paths — including the recommendation view and the CSV export — contain no explicit user_id filter in application code. They depend entirely on the *_select_own Row-Level Security policies to scope results to the authenticated user.

This is not a defect under the current configuration: the RLS policies are correct and enforce ownership at the database layer. However, the safety of these read paths is contingent on RLS remaining enabled. A future migration, environment misconfiguration, or use of a service-role key that bypasses RLS would silently expose cross-user data with no second line of defense.

Recommendation

Add an explicit .eq('user_id', user.id) filter at the application layer on all read paths, as defense in depth. This ensures correct scoping even if RLS is ever disabled or bypassed, and makes the ownership contract visible in the code rather than implicit in database configuration.

> *Principle: security controls should not have a single point of failure. RLS at the database and an explicit filter in code are complementary, not redundant — each protects against a different class of mistake.*

5. Remediation Plan

5.1 Prioritised Action List

Order	ID	Action	Effort	Risk if deferred
1	TFR-01	Add UPDATE/DELETE RLS policies; add UNIQUE on recommendations.checkin_id	Low	Ongoing data corruption; broken plan views
2	TFR-02	Build /settings/reset-password route with updateUser form	Medium	Users locked out; account recovery impossible
3	TFR-04	Fire checkin_submitted on successful submit	Low	Core funnel remains unmeasurable
4	TFR-03	Honour preferred_time in cron, or remove the timing promise	Medium	Broken product promise; erodes trust
5	OBS-01	Add explicit user_id filter to all read paths	Low	Latent cross-user exposure if RLS bypassed

Sequencing rationale: TFR-01 is ordered first despite low effort because it actively corrupts data on every update — the cost of deferral grows with each affected check-in. TFR-04 is promoted above TFR-03 because it is low-effort and unblocks measurement of every subsequent fix.

5.2 Verification Criteria

Each remediation is considered complete only when the following can be demonstrated:

- TFR-01 — A user can edit a check-in and see the edit persist; a user can delete a check-in and confirm it is removed from the database; exactly one recommendation row exists per check-in.
- TFR-02 — A user can complete the full forgot-password journey end to end and authenticate with the new password.
- TFR-03 — Reminders are observed to dispatch within the promised window of the user's preferred_time; or all per-user timing copy is removed.
- TFR-04 — A checkin_submitted event is observed in the analytics destination immediately after a successful submission.
- OBS-01 — Read paths return correctly scoped data with RLS disabled in a controlled test environment.

6. Appendix

6.1 Related Documents

Document	Reference	Relationship
System Architecture Document	FN-SAD-2026-001	Describes the architecture these findings affect
Software Requirements Specification	FN-SRS-2026-001	Defines the behaviour TFR-01–TFR-04 deviate from
FamNest (Live System)	famnest-iota.vercel.app	The deployed system under review

6.2 Author

Authored by Virginia Mwega, full-stack developer and technical writer. This report documents a real audit of a live production system; all findings were verified against the actual codebase.

Writing portfolio	virginiamwega-com.vercel.app
Live system	famnest-iota.vercel.app
Email	virginiamwega2@gmail.com

> *This document is a portfolio and engineering artifact authored by Virginia Mwega. Findings reflect the FamNest v1.0 codebase at time of audit. It demonstrates structured defect analysis, root-cause reasoning, and remediation planning to a professional standard.*