SOURCE CODE & PROJECT REVIEW

Project Health Assessment

PawNest — Pet-Sitting Marketplace · Web, iOS, Android

Prepared for	**PawNest Ltd.**
Prepared by	NUS Technology
Review package	High-level Audit
Codebase	`github.com/pawnest/pawnest-app @ 8c41e2d (412 commits)`
Date	1 November 2026
Version	1.0 — SAMPLE (fictional client)

Confidential. This document contains confidential information about PawNest Ltd.'s software and is intended solely for PawNest Ltd. It must not be shared with third parties without written consent. Secrets and personal data found during the review have been masked.

NUS Technology · nustechnology.com

1 Executive Summary

PawNest connects pet owners with vetted local sitters for home visits and overnight stays, earning a 15% commission per booking. The web app was generated with Lovable and the mobile app with Expo and Cursor; together they cover most of the core journey (search, book, pay, chat) and are used by around 140 beta users. The product looks finished on the surface, but the backend has no effective access control, payments can be faked, and there are no database backups. These issues must be closed before the public launch planned for January 2027.

TAKEOVER VERDICT

Refactor

Continue | **Refactor** | Partial rewrite | Rewrite

Overall health **1.9 / 5**

The frontend is worth keeping: it uses one consistent component library, the routing and screen structure are sensible, and most user-facing flows already work. The serious problems sit in the backend — database access rules, payment confirmation, environments and backups — and in missing marketplace operations such as refunds and sitter payouts. These can be fixed without a rewrite. We estimate about 2 weeks to remove the critical exposure and 5–6 further weeks to reach a safe public launch.

Findings at a glance

4	7	9
Critical	High	Medium

This is a high-level audit: findings are grouped by root cause, and Low-severity and informational issues are not included in this report.

Top risks

1. Any logged-in user can read or change every booking, message and pet record, including owners' home addresses and door codes.
2. A booking can be marked as paid without paying, simply by opening the payment-success page.
3. The OpenAI API key is visible to every website visitor and can be used to run up charges on PawNest's account.
4. There are no database backups: one mistaken query or a provider incident could permanently erase all users and bookings.
5. Development and the live app share one database, so every test or mistake during development directly affects real customers.

Key strengths

- ✔ Consistent UI built on shadcn/ui and Tailwind — easy to extend and restyle
- ✔ Core booking journey (search → book → pay → chat) is implemented end to end
- ✔ Codebase is fully exported to GitHub and runs without the Lovable platform
- ✔ Supabase Auth is used instead of a home-grown login system
- ✔ Clear product scope with no significant unused features

2 Scorecard

Each area is scored 1 (Critical) to 5 (Excellent). Red = 1–2, Amber = 3, Green = 4–5. Counts show findings by severity: Critical, High, Medium. Low-severity and informational issues are not included in this report.

#	AREA	SCORE	FINDINGS	ASSESSMENT
CQ	**Code Quality & Maintainability**	2 · Poor	1 3	Clean UI layer, but inconsistent data access, an out-of-support mobile SDK and an unmanaged database schema raise the cost and risk of every change.
SEC	**Security**	1 · Critical	3 1	Access control is effectively missing at the database level, payment status is trusted from the browser, and a paid API key is public.
PERF	**Performance & Scalability**	3 · Fair	1	Fine at today's volume; search and listings will slow down sharply beyond a few thousand sitters or bookings.
TEST	**Testing & Monitoring**	2 · Poor	1 1	No error tracking and no meaningful automated tests — failures in production are invisible until a user complains.
DOC	**Documentation**	2 · Poor	1	Only the default generated README; a new developer needs most of a day to get the project running.
BIZ	**Business & Feature Fit**	2 · Poor	2 1	The booking journey works, but refunds, sitter payouts and trust & safety tooling are missing for a real marketplace.
INFRA	**Infrastructure**	1 · Critical	1 1 1	No database backups, one shared environment for development and production, and free-tier services in production.
COMP	**Compliance & Legal**	2 · Poor	2	The mobile apps miss store requirements (account deletion, accurate privacy disclosures) that will block the next releases.

3 Scope & Methodology

In scope

- Web app (React + Vite) and mobile app (Expo / React Native) source code
- Supabase schema, access rules, storage and Edge Functions
- Stripe and OpenAI integrations
- Full git history (412 commits) for exposed secrets
- Read-only access to Supabase and Netlify dashboards provided by the client

Out of scope

- Low-severity and informational issues (not included in this report)
- Line-by-line code review — findings are grouped by root cause
- Penetration testing, load testing and UI/UX design review
- Legal advice — compliance findings flag risks for PawNest's legal counsel to confirm

Limitations

- No access to the Apple Developer or Google Play accounts; store-related findings are based on the build configuration and public listings.
- Stripe dashboard not shared; payment setup inferred from code.

Project inventory

Web frontend	React 18.2, Vite 5, TypeScript 5.2, Tailwind, shadcn/ui
Mobile	Expo SDK 49, React Native 0.72
Backend	Supabase (Postgres 15, Auth, Storage, 9 Edge Functions)
Payments	Stripe Checkout (no Stripe Connect)
AI	OpenAI API (pet profile bio generation)
Hosting	Netlify (web), Supabase Free plan
Size	~21,400 lines of code (TS/TSX 19,800; SQL 900; other 700)
Git history	412 commits, 2 contributors, Mar–Oct 2026
Users	~140 beta users, 310 bookings (client data)

Methodology

This is a **high-level audit**: automated scans (secrets across the full git history, static analysis, dependency vulnerabilities, size and duplication) plus a walkthrough of the critical user journeys from UI to database. Findings are grouped by root cause and focus on what matters for taking over and launching the product. Each is marked **Verified** or **Suspected** (needs runtime or account access to confirm). **Low-severity and informational issues are not included in this report.**

Rating scale

Critical	Exploitable now, or causing data loss, money loss or legal exposure.
High	Major risk at launch or at scale; blocks going live safely.
Medium	Slows development, raises maintenance cost or degrades UX.

Effort S	Less than 1 day
Effort M	1–5 days
Effort L	More than 1 week
P0	Fix now — all Critical
P1	Before launch — High, and Medium quick wins
P2	Backlog — remaining Medium

4 Findings by Area

CQ Code Quality & Maintainability

2/5 · Poor

Clean UI layer, but inconsistent data access, an out-of-support mobile SDK and an unmanaged database schema raise the cost and risk of every change.

CQ-02

Mobile app on an out-of-support Expo SDK

High | Effort M | P1 | Verified

`mobile/package.json: expo ~49`

ISSUE	The mobile app uses Expo SDK 49, which no longer receives fixes and targets an Android API level below Google Play's current minimum.
BUSINESS IMPACT	New Android releases will be rejected and security fixes are no longer available.
RECOMMENDATION	Upgrade the Expo SDK step by step and re-test native features.

CQ-01

Inconsistent architecture with business logic spread across screens

Medium | Effort L | P2 | Verified

`src/pages/*` `mobile/screens/*`

ISSUE	Data is loaded in three different ways, pricing and availability rules are re-implemented in several screens, and key screens are very large (up to ~1,500 lines) with about 14% duplicated code.
BUSINESS IMPACT	Every change costs more and behaves differently by screen; price inconsistencies between listing and checkout are already visible.
RECOMMENDATION	Standardize on one data layer and move business rules into shared modules, incrementally during feature work.

CQ-03

Database schema not under version control; unsafe data types

Medium | Effort M | P2 | Verified

`supabase/migrations/ (outdated)` `bookings.price (float)` `bookings.start_time (no time zone)`

ISSUE	Most schema changes were made by hand in the dashboard, so the repository cannot recreate the database. Prices are stored as floating-point numbers and booking times without time zones.
BUSINESS IMPACT	No reliable way to create test environments or review database changes; rounding errors in money and wrong visit times across time zones.
RECOMMENDATION	Generate a baseline migration from production and make migrations the only way to change the schema; store money as integer cents and times with time zone.

CQ-04

Demo data shown to real users

Medium | Effort S | P1 | Verified

`src/data/mockSitters.ts` `production database: test accounts`

ISSUE	When search fails, the app silently shows hard-coded demo sitters, and test accounts and bookings live in the production database.
BUSINESS IMPACT	Customers may try to book sitters who do not exist; metrics and payouts are skewed.
RECOMMENDATION	Remove mock data from production code, show proper empty/error states, and clean test data once backups exist.

SEC Security

1/5 · Critical

Access control is effectively missing at the database level, payment status is trusted from the browser, and a paid API key is public.

SEC-01

Any logged-in user can read and change other users' data

Critical | Effort M | P0 | Verified

`Supabase: RLS disabled on bookings, messages, pets` `supabase/functions/get-booking`

ISSUE	Database access rules (row-level security) are switched off on three core tables and wide open on profiles, and one backend function returns any booking by ID without checking who is asking. Home addresses, door codes, pet medical notes and private messages are all exposed to every registered user.
BUSINESS IMPACT	This is a personal-data exposure that customers and regulators would treat as a breach, and a serious trust risk for a business that holds home-access details.
RECOMMENDATION	Enable and write owner/sitter-scoped access rules on every table, enforce ownership checks in backend functions, and store home-access details separately with restricted access. Ask legal counsel whether the past exposure must be notified.

```
bookings | rowsecurity = f
messages | rowsecurity = f
pets     | rowsecurity = f
profiles | policy USING (true)
```

SEC-02

Bookings can be marked as paid without paying

Critical | Effort S | P0 | Verified

`src/pages/PaymentSuccess.tsx`

ISSUE	The app sets a booking to 'paid' from the browser when the payment-success page opens, instead of waiting for confirmation from Stripe.
BUSINESS IMPACT	Sitters can complete jobs that were never paid; PawNest loses the booking value and its commission.
RECOMMENDATION	Confirm payments only through a signed Stripe webhook handled on the server; the success page should only display the status.

SEC-03

Paid API keys exposed to the public

Critical | Effort S | P0 | Verified

`src/lib/ai.ts (web bundle)` `git history: .env (commit 3f9a1c2)`

ISSUE	The OpenAI key is shipped inside the website code where any visitor can read it, and the database master key was committed to git history (rotated during this review at our request).
BUSINESS IMPACT	Anyone can run up charges on PawNest's OpenAI account, and the leaked master key could have given full control of the database.
RECOMMENDATION	Rotate the OpenAI key, move AI calls to the server with per-user limits and a spending cap, purge secrets from git history, and add automated secret scanning.

SEC-04

No abuse protection and unvetted dependencies

Medium | Effort S | P1 | Verified

`AI generation, OTP and messaging endpoints` `package.json / mobile/package.json`

ISSUE	Costly or abusable features have no rate limits, several dependencies have known high-severity advisories, and the mobile app relies on an unofficial, barely used authentication package.
BUSINESS IMPACT	Easy to abuse for spam or cost attacks; supply-chain risk on user sessions.
RECOMMENDATION	Add per-user rate limits, upgrade flagged packages, replace the unofficial package with the official Supabase client, and enable automated dependency updates.

PERF Performance & Scalability

3/5 · Fair

Fine at today's volume; search and listings will slow down sharply beyond a few thousand sitters or bookings.

PERF-01

Search and listings will not scale

Medium | Effort M | P2 | Verified

`src/pages/Search.tsx` `bookings / messages tables`

ISSUE	Search downloads every sitter with full-size photos and filters in the browser; lists have no pagination and the main query columns have no indexes.
BUSINESS IMPACT	Fine with 60 sitters today; with a few thousand, pages will take several seconds on mobile and hosting costs will rise.
RECOMMENDATION	Move filtering to the database with pagination, add indexes on the main query columns, and serve resized images.

TEST Testing & Monitoring

2/5 · Poor

No error tracking and no meaningful automated tests — failures in production are invisible until a user complains.

TEST-01

High | Effort S | P1 | Verified

No error tracking — failures are invisible

`web, mobile and backend functions`

ISSUE	Errors are not reported anywhere, and many are silently ignored in the code, so the app just appears to do nothing when something fails.
BUSINESS IMPACT	Failed bookings or payments go unnoticed until a customer complains — or leaves.
RECOMMENDATION	Add error tracking (e.g. Sentry) to web, mobile and backend with alerts for payment failures.

TEST-02

Medium | Effort M | P2 | Verified

No automated tests on critical journeys

ISSUE	Only two placeholder tests exist; sign-up, booking, payment and access rules are untested.
BUSINESS IMPACT	Every fix risks breaking something else unnoticed.
RECOMMENDATION	Add tests for access rules and payments, plus a few end-to-end tests of the main journeys, run on every change.

DOC Documentation

2/5 · Poor

Only the default generated README; a new developer needs most of a day to get the project running.

DOC-01

Medium | Effort S | P1 | Verified

No setup guide or technical documentation

`README.md (generated default)`

ISSUE	There is no setup guide, no list of the 14 environment variables, and no description of the backend functions or data model. Our reviewer needed 5.5 hours to run both apps locally.
BUSINESS IMPACT	Every new developer pays this cost again, and knowledge stays with whoever built it.
RECOMMENDATION	Write a README with setup, environment variables and deployment steps, plus a short architecture overview.

BIZ Business & Feature Fit

2/5 · Poor

The booking journey works, but refunds, sitter payouts and trust & safety tooling are missing for a real marketplace.

BIZ-01

High | Effort M | P1 | Verified

No cancellation or refund flow

ISSUE	Bookings cannot be cancelled in the app and there is no refund logic; refunds are done manually in Stripe today.
BUSINESS IMPACT	Cancellations are routine in pet sitting; manual handling does not scale and commission accounting becomes wrong.
RECOMMENDATION	Define a cancellation policy and implement cancellation, automatic refunds and notifications.

BIZ-02

High | Effort L | P1 | Verified

Sitter payouts not automated

ISSUE	All payments go to PawNest's account and sitters are paid manually by bank transfer from a spreadsheet.
BUSINESS IMPACT	The 15% commission model is not enforced by the system; manual payouts are slow, error-prone and may create obligations for holding third-party funds.
RECOMMENDATION	Move to Stripe Connect with automatic commission and sitter onboarding/verification.

BIZ-03

Medium | Effort L | P2 | Verified

No trust & safety or admin tooling

ISSUE	Sitter verification is a manual database flag, there is no report/block feature, and all operations are done by editing the database directly.
BUSINESS IMPACT	Trust is the core product of a home-access marketplace; operations are slow and risky.
RECOMMENDATION	Add ID verification, report/block and a small admin panel for bookings, users and refunds.

INFRA Infrastructure

1/5 · Critical

No database backups, one shared environment for development and production, and free-tier services in production.

INFRA-01

Critical | Effort S | P0 | Verified

No database backups

`Supabase Free plan`

ISSUE	The production database has no backups of any kind, and a restore has never been tested.
BUSINESS IMPACT	A mistaken query or a provider incident would permanently erase all users and bookings.
RECOMMENDATION	Take a manual backup today; before launch, enable point-in-time recovery and a nightly off-site backup with a tested restore.

INFRA-02

Development and production share one environment

High | Effort M | P1 | Verified

`.env` `Netlify preview deploys`

ISSUE	Local development, preview deploys and the live app all use the same database and keys.
BUSINESS IMPACT	Every experiment or bug during development directly affects real customers and their data.
RECOMMENDATION	Create separate development, staging and production environments with their own databases and keys.

INFRA-03

Production runs on free tiers with rate-limited email

Medium | Effort S | P1 | Verified

`Supabase Free plan` `Supabase default SMTP`

ISSUE	The database plan pauses on inactivity and sign-up/password emails go through a built-in sender limited to a few emails per hour.
BUSINESS IMPACT	At launch, new users will not receive confirmation emails and cannot sign up.
RECOMMENDATION	Upgrade the database plan and configure a transactional email provider on PawNest's domain.

COMP Compliance & Legal

2/5 · Poor

The mobile apps miss store requirements (account deletion, accurate privacy disclosures) that will block the next releases.

COMP-01

No in-app account deletion

High | Effort S | P1 | Verified

`mobile/screens/Settings.tsx`

ISSUE	Apple and Google require apps with account sign-up to let users delete their account from within the app; PawNest does not offer this.
BUSINESS IMPACT	The next iOS and Android submissions are likely to be rejected, blocking all updates including security fixes.
RECOMMENDATION	Add account deletion that removes or anonymizes personal data, with handling for active bookings.

COMP-02

Privacy disclosures do not match how data is used

High | Effort S | P1 | Verified

`/privacy` `mobile/ios (no privacy manifest)` `Google Play data-safety form`

ISSUE	The privacy policy is a generic template that does not mention OpenAI or Google Maps, the iOS app has no privacy manifest, and the Google Play form states that no data is shared.
BUSINESS IMPACT	Store rejection and possible enforcement for inaccurate disclosures; a transparency issue under data-protection law.
RECOMMENDATION	Update the privacy policy (with legal counsel), add the iOS privacy manifest and correct the store privacy forms.

5 Remediation Roadmap

Effort ranges are for one NUS Technology developer familiar with the stack and include regression testing. Phases can overlap with additional developers.

Phase 0 — Fix now

Weeks 1–2

6–8 days

Close active data and money exposure and protect the data.

SEC-01	Any logged-in user can read and change other users' data
SEC-02	Bookings can be marked as paid without paying
SEC-03	Paid API keys exposed to the public
INFRA-01	No database backups

Phase 1 — Before public launch

Weeks 3–8

25–30 days

Make the product safe and operable for launch: environments, payments, store requirements and visibility of errors.

INFRA-02	Development and production share one environment
CQ-03	Database schema not under version control; unsafe data types
BIZ-01	No cancellation or refund flow
BIZ-02	Sitter payouts not automated
CQ-02	Mobile app on an out-of-support Expo SDK
COMP-01	No in-app account deletion
COMP-02	Privacy disclosures do not match how data is used
TEST-01	No error tracking — failures are invisible
CQ-04	Demo data shown to real users
SEC-04	No abuse protection and unvetted dependencies
PERF-01	Search and listings will not scale
DOC-01	No setup guide or technical documentation
INFRA-03	Production runs on free tiers with rate-limited email

Phase 2 — After launch

Weeks 9–12

15–20 days

Reduce the cost of future features and build trust & operations tooling.

BIZ-03	No trust & safety or admin tooling
CQ-01	Inconsistent architecture with business logic spread across screens
TEST-02	No automated tests on critical journeys

6 Next Steps

1. Walk-through call (45 minutes) to go over the findings and answer questions.
2. PawNest takes the immediate owner actions: set an OpenAI spending cap and take a manual database backup.
3. NUS Technology sends a fixed-price proposal for Phase 0 and a time-and-materials estimate for Phase 1.
4. Kick-off of Phase 0 within 3 working days of approval; we provide a short weekly progress report against this document's finding IDs.

A Appendix — Findings Index

ID	TITLE	SEVERITY	EFFORT	PRIORITY	STATUS
INFRA-01	No database backups	Critical	S	P0	Verified
SEC-01	Any logged-in user can read and change other users' data	Critical	M	P0	Verified
SEC-02	Bookings can be marked as paid without paying	Critical	S	P0	Verified
SEC-03	Paid API keys exposed to the public	Critical	S	P0	Verified
BIZ-01	No cancellation or refund flow	High	M	P1	Verified
BIZ-02	Sitter payouts not automated	High	L	P1	Verified
COMP-01	No in-app account deletion	High	S	P1	Verified
COMP-02	Privacy disclosures do not match how data is used	High	S	P1	Verified
CQ-02	Mobile app on an out-of-support Expo SDK	High	M	P1	Verified
INFRA-02	Development and production share one environment	High	M	P1	Verified
TEST-01	No error tracking — failures are invisible	High	S	P1	Verified
BIZ-03	No trust & safety or admin tooling	Medium	L	P2	Verified
CQ-01	Inconsistent architecture with business logic spread across screens	Medium	L	P2	Verified
CQ-03	Database schema not under version control; unsafe data types	Medium	M	P2	Verified
CQ-04	Demo data shown to real users	Medium	S	P1	Verified
DOC-01	No setup guide or technical documentation	Medium	S	P1	Verified
INFRA-03	Production runs on free tiers with rate-limited email	Medium	S	P1	Verified
PERF-01	Search and listings will not scale	Medium	M	P2	Verified
SEC-04	No abuse protection and unvetted dependencies	Medium	S	P1	Verified
TEST-02	No automated tests on critical journeys	Medium	M	P2	Verified

NUS Technology
nustechnology.com
Questions about any finding? We are happy to walk you through this report on a call.