SAMPLE REPORT

Security and Production-Readiness Audit

AI-built customer feedback app (Bolt, React, Supabase)

Date	19 Nov 2026	**Method**	17-point checklist: security, reliability, deployment
Stack	React + Vite, Supabase (Postgres, Auth, Storage, Edge Functions)	**Scope**	Code, database policies, edge functions, live behaviour, dependencies

1. Summary

The app is feature complete and its database layer is solid: Row Level Security is on for every table and sensitive functions are locked down. The weak points are the **server functions and operations**. Two high-severity issues let any visitor spend the owner's AI quota and flood the public form. Both are fixable in about a day of work.

Pass	Partial	Fail	Unknown	N/A
7	1	6	2	1

2. Findings, ranked by severity

#	Severity	Finding and impact	Recommended fix	Est.
1	High	**AI endpoint usable without signing in** The function accepts the public anon key as a valid login, so anyone can call it. An anonymous request ran the function and reached the AI provider on the owner's key.	Require a real user session for authenticated actions; rate limit the public one; validate input.	3-4 h
2	High	**No rate limiting on the public form, uploads or votes** Unlimited submissions, 5 MB uploads and roadmap votes from anonymous visitors. Spam, storage cost and vote stuffing.	Move anonymous writes behind a server function with rate limits; add database throttles; optional captcha.	4-6 h
3	Medium	**CORS allows any origin (*)** Every edge function accepts browser calls from any website.	Allow-list the real app origin(s) via an environment variable.	1 h
4	Medium	**Notification function trusts a bare record id** Callable without a login; a failed Slack call permanently loses the notification.	Validate the id, only act on fresh records, release the claim on failure.	1-2 h
5	Medium	**Known vulnerable dependency** `npm audit` reports a high-severity issue in a production package.	Update dependencies and re-audit.	0.5 h
6	Low	**Invite link built from client-supplied origin; unescaped HTML in email** A malicious inviter could send links to another site from the app's sender address.	Use a server-side base URL; escape values in HTML.	1 h
7	Low	**Raw error messages returned to callers** Internal details can leak through error text.	Return generic errors; log details server-side.	0.5 h
8	Ops	**No CI, README, error boundary, error tracking or backups** One bad commit or render error can break the app with no warning and no way back.	Add CI, an error boundary, error tracking, a README and backups.	3-4 h

Estimates are for implementation and testing of each fix. Total for all findings: about 15-19 hours.

3. Checklist results

Every item the audit covered, with the result and a short note.

Check	Result	Note
Security		
Secrets not in frontend or git	**Pass**	Only public keys reach the browser.
service_role key never shipped to client	**Pass**	Used only inside edge functions.
RLS enabled and correct on every table	**Pass**	All tables protected; sensitive functions locked down.
Protected routes enforced server-side; no IDOR	**Pass**	Policies check workspace membership and role.
Input validation and sanitisation	**Pass**	Database constraints; no unsafe HTML rendering.
CORS configured properly	**Fail**	Wildcard origin on all functions.
Rate limiting on auth and expensive endpoints	**Fail**	None on AI, uploads, form or votes.
Stripe webhooks verified	**N/A**	No payments in this app.
Reliability		
Error handling; no white screens	**Partial**	Friendly errors, but no error boundary or tracking.
Environment variables separated dev/prod	**Pass**	Secrets live in function secrets.
Database indexes; no N+1 queries	**Pass**	Indexes on main query paths.
Backups configured	**Unknown**	Needs a dashboard check; depends on plan.
Dependencies current, no known vulnerabilities	**Fail**	High-severity finding in `npm audit`.
Deployment		
Custom domain plus HTTPS	**Unknown**	Not hosted at audit time.
CI/CD with build checks	**Fail**	No pipeline.
Uptime monitoring and error tracking	**Fail**	None configured.
README with setup and deploy steps	**Fail**	No README.

4. Recommended order of work

Week 1, security. Fix findings 1 and 2 first. They are the only ones with direct cost or abuse risk.

Week 1, hardening. Findings 3 to 7: CORS allow-list, notification function, dependency updates, invite flow, generic errors.

Week 2, operations. CI, error tracking, uptime monitoring, backups, README, custom domain.

5. Scope and limits

This audit reviewed the application code, database policies, edge functions, dependencies and live behaviour against the checklist above. It is a point-in-time review, not a penetration test, and it does not certify the app as secure. Items marked Unknown need dashboard access or a decision from the owner.

Want this for your app? Fix packages cover the critical issues, and a production launch package adds deployment, CI/CD, monitoring, backups and handover docs. Message me with your stack and what the app does before ordering.