Fernlight: production-readiness report

Prepared for Sam, 12 October 2026

Summary

Overall RED (2 not verified)

What could hurt you now

- Your login check accepts any value in one HTTP header, so anyone on the internet can pretend to be any of your customers (including you) and read, change or delete their data `03-6787bc9a1e00`.
- Any logged-in customer can see every other customer's booking, including names, phone numbers and health notes, just by changing a number in the address bar `04-75d67ea7af1b` `04-c28b396bc513`.
- Your AI class-description feature is open to anyone, has no limit, and your AI provider has no spending cap, so a stranger could run up a very large bill overnight `13-e09364d9fbce` `13-ef71a77623ac`.

What's already done well

- Prices come from your server, so customers can't change what they pay for a class pack `08-1d1e2dad45de`.
- Daily backups are on with seven days of rollback, held outside your app `10-dabcac2daedb`.
- Only the keys that are meant to be public reach your customers' browsers `02-aeb6876ef898`.

What we recommend first

	Fix	Why	Who	Price
1	Replace the broken login check with your real login provider	Right now anyone can act as any customer	Us, as a pull request	$900
2	Stop customers seeing other customers' bookings, and turn the database's own protection back on	Personal and health data is reachable today	Us, as a pull request	$900

	Fix	Why	Who	Price
3	Limit the AI feature and set a spending cap with your AI provider	One person could run up a huge bill	Us (limit) and you (cap)	$400 + $150

What we couldn't check: whether anything in your live app leaks secrets or personal data into logs (send a sample of a recent server log with the user data redacted), and whether any of the code libraries you use have known security holes (we need the yarn or pnpm lockfile, whichever you use).

Secrets in code history	Secrets in the app	Who can call what	Who can see what	Files and uploads	Personal data	Payment messages
A	NV	R	R	A	R	A

Prices and access	Test and live apart	Backups exist	Restore rehearsed	Dependencies	Abuse and runaway cost	Who can change production
G	R	G	R	NV	R	R

R = needs fixing · A = weak spot · G = checked and OK · NV = not verified · — = doesn't apply

---

The detail

Each finding has a confidence: **Confirmed** (we saw it directly), **Likely** (strong evidence, one link unseen) or **Unverified** (plausible, needs checking). Anything you told us counts only where you sent proof, and is marked *you told us, with proof*.

1. Secrets

What could happen: if a password or key gets into your code history, anyone who ever had a copy of the code still has a working key, even after you delete the file.

What we found: a file that normally holds live passwords was committed to your code in October 2025 and removed about a month later. Nothing in the project since then shows the keys were replaced (Confirmed, `01-f65d326239fe`). The automated scan did not match the exact format, but that doesn't make the file safe.

Done well: the automated scan found no other secret-shaped strings in the full history (Confirmed, `01-4c3daeb2762e`). Only the keys that are meant to be public reach your customers' browsers (Confirmed, `02-aeb6876ef898`).

2. Access

What could happen: anyone can pretend to be any customer, and anyone logged in can see every other customer's bookings and health notes.

What we found: your login check only reads a single HTTP header and trusts whatever value is in it, with no production login logic underneath. Anyone can set that header to the owner's user id and act as the owner (Confirmed, `03-6787bc9a1e00`). The page that shows a booking does not check whose booking it is, and booking numbers simply count up (Confirmed, `04-75d67ea7af1b`). The database's own safety net on bookings is switched off with a comment calling it temporary (Confirmed, `04-c28b396bc513`). Your AI description route is on a public allowlist with no caller check (Confirmed, `03-e9cfa14c0945`). Signed waivers are stored on permanent public links; the links can't be guessed, but anyone who sees one in an email can open the waiver forever (Confirmed, `05-701824b565de`).

3. Personal data

What could happen: if a customer asks you to delete their data, you cannot fully do it today, which is a legal problem in the UK and the EU.

What we found: deleting an account removes the login only. The customer's bookings (with name, phone and health notes) and their signed waiver are left behind (Confirmed, `06-6cff6d24a78b`). You told us, with proof, that this is exactly what happened with the one deletion request you've had.

4. Payments

What could happen: your tests could take real money from real customers, or move real data; and if your payment provider retries a message, a customer could get class credits twice.

What we found: your preview versions of the app share your live payment keys and your live database (Confirmed, `09-4b7962e756b6`), which is dangerous whenever anyone opens a pull request. Your payment webhook checks the signature properly, but doesn't remember whether it has already seen the same message, so retries would add credits a second time (Confirmed, `07-c863ae0549b2`).

Done well: prices come from your server, never from the customer's browser (Confirmed, `08-1d1e2dad45de`).

5. Backups and recovery

What could happen: if your data were lost, you'd have backups, but you don't know yet whether you could restore from them under pressure.

What we found: daily backups with seven days of rollback are on (Confirmed, *you told us, with proof*, `10-dabcac2daedb`). You've never restored one, and there's no written step-by-step for doing it (Confirmed, `11-01f11f8ed158`).

6. Changes and running costs

What could happen: a stranger could make your AI bill explode, and anyone with push access (or anyone's AI coding assistant) could change your live app without an approval step.

What we found: your AI description route is public with no rate limit (Confirmed, `13-ef71a77623ac`) and your AI provider has no spending cap or alert (Confirmed, *you told us, with proof*, `13-e09364d9fbce`). Every push to your main branch goes live immediately: there's no branch protection, no approval, and nothing in the project tells an AI assistant to stay away from the deploy and database commands (Confirmed, `14-361debcd507d`; Likely, `14-81e40eb6d0d3` `14-a9a4b50fc3e7` `14-e180fac22143` `14-f9cb9b5a5202`).

Fixes that left a gap

Earlier fixes that protected new customers but left existing ones exposed.

Finding	The fix	What it left behind	Who is still exposed
`01-f65d326239fe`	The file holding live passwords and keys was removed from the code in November 2025	The keys themselves were never replaced (as far as we can see), so any copy made before the removal still works	Everything those keys unlock, until each one is replaced
`04-c28b396bc513`	A note on the bookings table says the database's own protection is off "while we debug the booking flow"	Debugging has long since finished, but the protection was never turned back on	Every booking row, if the app layer is ever bypassed

All recommended fixes

1. **Replace the broken login check with your real login provider** (M, closes `03-6787bc9a1e00`) Why: anyone can pretend to be any customer by sending one header. Done when: a test calls a private route with the test header and gets refused; the same test with a real Supabase session succeeds.
2. **Stop customers seeing other customers' bookings, and turn the database's own protection back on** (M, closes `04-75d67ea7af1b` `04-c28b396bc513`) Why: personal and health data is reachable today by any customer. Done when: logged in as one customer, trying to open another customer's booking id gets "not found"; and the database refuses the same read even if the app layer is bypassed. The tests fail on today's code and pass after the fix.

3. **Limit the AI description feature** (S, closes `13-ef71a77623ac`, together with fix 4) Why: one person could run your AI feature in a loop. Done when: a test makes 21 requests in a minute from the same caller and the 21st is refused; the feature requires either a login or a short-lived site token.
4. **Set a spending cap with your AI provider** (you, with our notes, closes `13-e09364d9fbce`) Done when: you send a screenshot of the cap and the alert address.

After those:

- Give previews their own test payment keys and a test database (M) `09-4b7962e756b6`.
- Make account deletion remove bookings, waiver files and payment customer records (M) `06-6cff6d24a78b`.
- Replace the credentials from the removed file and send a screenshot showing the new keys' creation date (you, with our notes) `01-f65d326239fe`.
- Make the AI description route require a login or a server token (S) `03-e9cfa14c0945`.
- Record each Stripe event id and refuse duplicates (S) `07-c863ae0549b2`.
- Serve waivers through short-lived signed links (S) `05-701824b565de`.
- Rehearse a restore into a spare database and keep a short written step-by-step (you, with our notes) `11-01f11f8ed158`.
- Add branch protection, a reviewer on the deploy step, and project settings that keep an AI assistant away from deploy and database commands (S) `14-361debcd507d` `14-81e40eb6d0d3` `14-a9a4b50fc3e7` `14-e180fac22143` `14-f9cb9b5a5202`.
- Audit your yarn or pnpm lockfile for known advisories (S) `12-fbd77e8add0a`.

What we couldn't check

- **Secrets in the app at runtime** `02-318ae78ba455`: whether your live logs or shipped browser bundle leak secrets or personal data. To settle it, send a sample of a recent server log with any user data redacted, and we'll check for the common leaks.
- **Dependencies** `12-fbd77e8add0a`: whether any of the libraries you use have known security holes. The automated audit needs an npm lockfile and your project doesn't have one. Tell us whether you use yarn or pnpm and we'll audit that lockfile instead.

Appendix: technical evidence

For whoever maintains the code. One entry per finding cited above.

Finding	Confidence	Evidence
01-4c3daeb2762e	Confirmed	gitleaks full-history scan: 0 findings.
01-f65d326239fe	Confirmed	`git log`: a production env file was added in commit `c248241` (2 Oct 2025) and removed in commit `50d3b2c` (5 Nov 2025). No commit, note or document after the removal mentions rotation. Values not read.
02-aeb6876ef898	Confirmed	`src/lib/supabase.js:2-3` exports only `NEXT_PUBLIC_SUPABASE_URL` and `NEXT_PUBLIC_SUPABASE_ANON_KEY`. No service-role key is exported here.
02-318ae78ba455	Unverified	No log redaction configuration in the repo. Runtime logs and the built client bundle were not reviewed.
03-6787bc9a1e00	Confirmed	`src/lib/auth.js:2-5`: `currentUser` returns `{ id: request.headers.get("x-test-user") }` with no production branch. `src/middleware.js:15` uses it to accept or reject.
03-e9cfa14c0945	Confirmed	`src/middleware.js:4-10`: `/api/describe-class` is on the public allowlist. `src/app/api/describe-class/route.js:6-12` calls OpenAI `gpt-4o` with no caller check or rate limit.
04-75d67ea7af1b	Confirmed	`src/app/api/bookings/[id]/route.js:5-10`: `db().bookings.find(Number(params.id))` with no owner filter. `supabase/migrations/0003_bookings.sql:2`: ids are `bigserial`.
04-c28b396bc513	Confirmed	`supabase/migrations/0003_bookings.sql:9-10`: `alter table bookings disable row level security` with the comment "Temporarily off while we debug the booking flow." The table holds `name`, `phone`, `health_notes`.
05-701824b565de	Confirmed	`src/lib/storage.js:6-9`: upload to the `waivers` bucket then `getPublicUrl()`. The returned URL is stored on the user's row and used for emails.

Finding	Confidence	Evidence
06-6cff6d24a78b	Confirmed	`src/app/api/account/route.js:5-9`: `db().auth.deleteUser(user.id)` only. No cascade to bookings, waiver files or Stripe customer. Founder confirmed (intake Q4) that only the login was removed on the one request so far.
07-c863ae0549b2	Confirmed	`src/app/api/webhooks/stripe/route.js:10` verifies the signature on the raw body. Line 16 calls `db().credits.add(...)` with no event-id dedupe table.
08-1d1e2dad45de	Confirmed	`src/app/api/checkout/route.js:7-14`: the client sends only `packId`; the server resolves `pack.stripePriceId` server-side.
09-4b7962e756b6	Confirmed	`docs/deploy.md:5-6`: "Preview deployments share the production environment variables (Stripe keys and the database URL)". `.github/workflows/deploy.yml` deploys on every push to `main`.
10-dabcac2daedb	Confirmed	Founder screenshot (`supabase-backups.png`): point-in-time recovery on, 7-day retention, latest backup 02:00 on review date.
11-01f11f8ed158	Confirmed	Founder (intake Q2, with the backup screenshot above): no restore has ever been performed. No runbook in the repository.
12-fbd77e8add0a	Unverified	No `package-lock.json` or `npm-shrinkwrap.json`. `npm audit` could not run. The yarn/pnpm lockfile has not been audited.
13-e09364d9fbce	Confirmed	Founder screenshot (`openai-limits.png`): no cap or usage alert. `src/app/api/describe-class/route.js:6-12` calls OpenAI `gpt-4o` per request.
13-ef71a77623ac	Confirmed	`src/app/api/describe-class/route.js`: no rate limiter, no per-caller counter, no token check. Route is public per `src/middleware.js:4-10`.
14-81e40eb6d0d3	Likely	No `.claude/settings.json` in the repo. Nothing project-level stops an AI assistant running the GitHub Actions deploy step (which calls `npx vercel --prod` with a stored token).

Finding	Confidence	Evidence
14-a9a4b50fc3e7	Likely	`.github/workflows/deploy.yml` deploys on every push to `main` with no GitHub environment and no reviewer, so nothing in the workflow requires an approval.
14-e180fac22143	Likely	No `.claude/settings.json`; nothing project-level stops an AI assistant running the `package.json` script `deploy` (`vercel --prod`).
14-f9cb9b5a5202	Likely	No `.claude/settings.json`; nothing project-level stops an AI assistant running the `package.json` script `db:push` (`supabase db push`).
14-361debcd507d	Confirmed	Founder screenshots (`github-branches.png`, `vercel-team.png`): branch protection off on `main`; two owners on Vercel, Supabase and GitHub. Combined with the ungated workflow above.

Scope: Fernlight repository at commit `b572976c958d7460d9c395665cf0b67fef85a640`, reviewed read-only on 12 October 2026 with Thetis 0.5.1. We didn't access your live systems, run the app against live services, or read files containing passwords or keys. Ratings reflect what we could see at that commit.
