BERGER SOLUTIONS · SOFTWARE STUDIO, DUBAI, UAE · 34 QUESTIONS, 7 CATEGORIES

The production audit checklist

The questions we check before an AI-built app is trusted with real customers, real money, and real data.

Ask your AI whether your app is secure and it will say yes. It is not lying. It is answering the question it was asked, about the code it can see, and it has no way to know which question you should have asked instead. The problems that end up costing founders are the ones nobody asked about.

So this list is built for coverage rather than depth. It is the surface: every area, so you can see where you have not looked. For the deeper version of a dozen of them, with the exact prompt to paste, what a real answer looks like and the follow-up when you get a confident one, the page at bergersolutions.net/guides/ask-your-ai is the companion to this.

This is not everything an audit checks, and saying so is the point of the list. These are the questions you can ask without access: put each one to your AI, or to whoever built the app. Most of what an audit examines cannot be reached by asking, because it needs the code, the database and the dashboards in front of somebody. A confident yes proves nothing on its own, but a hesitation, a "mostly", or a "we should probably" tells you exactly where to look.

HOW TO USE IT

- Work top to bottom. The order is by how fast a miss can hurt you, not by how technical it sounds.
- A question you cannot answer counts as a no. Not knowing is the finding.
- The checklist tells you what matters. It cannot verify itself. Telling a yes that sounds right from a yes that is true is what the audit does.

01 · 4 QUESTIONS

Exposures: what is leaking right now

The cheapest problems to find and the most expensive to leave. Any of these can be exploited today by someone who never logs in.

- [ ] **Are there any passwords, API keys, or tokens written into the code, or anywhere in the repository's history?**

  BAD SIGN "We moved them to a config file" (still in the repo), or a key that was deleted but lives on in old commits, where every copy of the repo still has it.

- [ ] **Can someone with the app's public key read your database directly?**

  BAD SIGN Tables with row-level security switched off, or "the app checks permissions". The app is not the database, and the public key ships to every browser.

- [ ] **Is a .env file, or any file holding credentials, committed to the repository?**

  BAD SIGN Any yes. A .env.example with placeholder values is fine; a real .env is not.

- [ ] **Is there an admin page or dashboard that can be reached without logging in?**

  BAD SIGN "It's hidden, nobody knows the URL." A hidden URL is not a lock.

02 · 5 QUESTIONS

Foundations: can the data be trusted

A wrong data model corrupts quietly, and it is the single strongest reason an app ends up rebuilt instead of repaired.

- [ ] **Does every table have a primary key, and are its relationships to other tables actually declared, not just implied by naming?**

  BAD SIGN Relationships "by convention", or ids stored as plain text with nothing linking them. Orphaned records are a matter of time.

- [ ] **Is money stored as whole minor units (cents) or a fixed decimal type, never a floating-point number?**

  BAD SIGN "It's a float, it works fine." It drifts by fractions of a cent until a total is wrong.

- [ ] **Is each customer's data separated from every other customer's by the database itself, not by a filter the code has to remember every time?**

  BAD SIGN "Every query has a WHERE clause for that." It does, until the one query that forgets, and that one is a breach.

- [ ] **Have you ever restored from a backup and confirmed the restore actually worked?**

  BAD SIGN "Backups are on" with no restore ever attempted. An untested backup is a hope, and the failure is discovered during the disaster.

- [ ] **Are changes to the database structure made through versioned migrations, not by hand in a dashboard?**

  BAD SIGN "We just change it in the admin panel." Then no environment can be rebuilt, and nobody knows what the real structure is.

03 · 5 QUESTIONS

Security: who can do what

The most common flaw in AI-built apps. The tools build the path where the right user asks, and never the path where a different user asks.

- [ ] **Does every endpoint check that the caller owns the specific record it is returning or changing, not merely that they are logged in?**

  BAD SIGN "You have to be logged in." The classic hole: change the id in the URL and read someone else's data.

- [ ] **Are privileged actions blocked on the server, or only hidden in the interface?**

  BAD SIGN "The admin button doesn't show for normal users." The endpoint behind the button is still there for anyone who calls it directly.

- [ ] **Is every input validated on the server, and is every database query parameterized rather than built from strings?**

  BAD SIGN "The form validates it" (the browser is not a control), or raw queries assembled from user input.

- [ ] **Are passwords hashed with a proper algorithm (bcrypt, argon2, or scrypt), if the app stores them at all?**

  BAD SIGN md5, sha1, plain text, or "the provider handles it" with nobody having actually checked.

- [ ] **If the app uses an AI model that can take actions (send email, spend money, write data), is that gated behind a confirmation or a limit?**

  BAD SIGN An agent with tools running on whatever a user types, or on documents users upload. Instructions hidden in a document become commands.

04 · 5 QUESTIONS

Logic: does it produce the right result

Security asks whether the app can be broken into. This asks whether it does the right thing on the paths the business actually runs on.

- [ ] **Where an action takes several steps (take the payment then grant access, create the account then send the invite, accept the booking then hold the slot), can the first step succeed while the last never happens?**

  BAD SIGN "That has never happened." It has not happened yet. Ask what happens when the second step fails after the first one already succeeded, because nothing puts the first one back.

- [ ] **If the same request arrives twice, because someone double-clicked or a provider retried, does the thing happen twice?**

  BAD SIGN No idempotency key and no deduplication. Providers retry for days, people double-click constantly, and a queue that promises delivery at least once means exactly that.

- [ ] **When something is cancelled, deleted or refunded, is every consequence reversed: the access, the linked records, the scheduled emails, and the money?**

  BAD SIGN The obvious half is undone and the rest is not, leaving a cancelled customer with working access, or a deleted record still referenced by something that now breaks.

- [ ] **Is there an automated test on the one flow the business genuinely cannot afford to have broken?**

  BAD SIGN "We test that one manually", or two hundred tests on small utilities and none on the flow the whole thing exists for.

- [ ] **What happens on the main screen with zero records, a brand-new account, or a missing optional field?**

  BAD SIGN A crash or a blank page nobody has seen, because every test account has data in it.

05 · 5 QUESTIONS

Operations: when it breaks

Everything breaks eventually. The question is whether it breaks loudly and safely, and whether a human finds out in time.

- [ ] **Are development and production truly separate, with their own databases and their own keys?**

  BAD SIGN A preview environment pointed at the production database, or the same payment key used everywhere. "Testing" then writes to real customers.

- [ ] **If checkout started failing at 2am, would anyone know before a customer complained?**

  BAD SIGN "We'd see it in the morning." No error tracking, no alert, no health check. The only signal is a support ticket.

- [ ] **Are errors ever swallowed: caught, logged, and then the app carries on as if nothing happened?**

  BAD SIGN Empty catch blocks, or a failure returned to the user as a success. Nothing downstream knows anything went wrong.

- [ ] **Can a bad release be rolled back in minutes?**

  BAD SIGN "We'd redeploy an older commit and hope", or database changes with no way back, so the code can be reverted but the data cannot.

- [ ] **Are debug mode, verbose errors, and stack traces switched off in production?**

  BAD SIGN Error pages that show file paths, framework versions, or database errors. That is a map for an attacker.

06 · 5 QUESTIONS

Performance: what happens at scale, and what it costs

An app that is fine at fifty users can fall over, or bankrupt itself, at five thousand. Cost comes first because it needs no traffic to hurt.

- [ ] **Can a single user trigger unbounded spend on an AI or paid API?**

  BAD SIGN An AI feature reachable without logging in, a loop with no cap, or no per-user limit. One malicious or buggy day and the card maxes out.

- [ ] **Is a hard spending cap set on every provider's dashboard?**

  BAD SIGN None. Most providers default to no cap at all, so the code-level limits are the only protection, and there are none of those either.

- [ ] **Does every list and every report paginate, or does it load the whole table?**

  BAD SIGN "It's fast." It is fast on five hundred rows. It will not be at fifty thousand.

- [ ] **Are there database queries running inside loops?**

  BAD SIGN Fetch a list, then one more query per item. The ORM hides it, and it is the most common performance bug in AI-written code.

- [ ] **Are independent calls made one after another when they could run at the same time?**

  BAD SIGN Ten sequential calls for one page load. Two seconds that should be a fifth of one.

07 · 5 QUESTIONS

Compliance: the rules that apply to you

Not whether there is a privacy policy. Whether the concrete obligations are actually met in the code and the data.

- [ ] **Which laws actually apply to you, based on where your users are and what data you hold (UAE PDPL, GDPR, PCI)?**

  BAD SIGN "We're a startup, that's for later." The law follows the users and the data, not the size of the company.

- [ ] **Can a user actually be erased on request, or only hidden behind a flag?**

  BAD SIGN Soft delete everywhere while the privacy policy promises deletion. A hidden flag is not erasure.

- [ ] **Do card numbers ever touch your database or your logs?**

  BAD SIGN Any yes. Only the payment provider should ever see a card number; the app should hold a token at most.

- [ ] **Is personal data (emails, names, uploaded documents) ending up in your logs?**

  BAD SIGN Logging whole requests or whole user objects. Logs are a data store too, usually with weaker protection than the database.

- [ ] **Do you own every account the business runs on: the domain, the hosting, the database, the payment provider?**

  BAD SIGN Any of them sitting in a freelancer's, an agency's, or a former co-founder's personal account. The business can be held hostage without a single line of bad code.

What a checklist cannot do

Every question here has a yes that sounds right and a yes that is true, and from the outside the two look identical. Telling them apart is verification: reading the code, the configuration, and the history, and proving what was and was not checked. That is the audit.

If you ran this list and came away with more "I think so" than you would like, that is exactly the conversation it is for. Thirty minutes, no pitch, and you leave knowing where it really stands.

Book the audit call

The audit: https://bergersolutions.net/services/mvp-to-production
Book directly: https://cal.com/bergersolutions/production-audit-call
jp@bergersolutions.net