Twenty checks between a demo and production.

The list we run before calling a product ready: authentication, data boundaries, payment correctness, observability, deployment, and evaluations. Written for AI-built apps, useful for any.

How to use it.

Score every check pass or fail against the running product, not against the plan. A check passes when someone can show the evidence in under five minutes; anything else is a fail with a nicer name.

Two rules make the list useful. First, any failure in payments or authentication stops feature work until it is fixed — those are the failures that cost money or trust. Second, the six areas are ordered by how often they are the actual cause: access, data, money, visibility, release, and model behaviour.

+ 20 checks across six areas
+ Written for AI-built and fast-shipped products
+ PDF downloads without an email
+ Used as the spine of the $4,500 assessment

01 / Authentication and access

Access is the first thing a reviewer checks and the last thing a fast build gets right. A hidden button is not a permission.

01 **Session revocation**

Sessions and tokens can be revoked server-side, they expire, and a deleted or downgraded user cannot act before the next refresh. Ask what happens in the ninety seconds after an admin removes someone.

02 **Authorization is server-side**

Every endpoint resolves the caller and checks role, tenant, and ownership against the data layer. Client-side guards are convenience, not control. Try the request directly with a valid token from a lower-privileged account; the server should refuse it.

03 **Privileged actions are explicit**

Destructive or money-moving actions carry a distinct permission, a confirmation, and an audit record naming who did it. Service-role keys never reach a component that renders in a browser.

02 / Data and tenancy

Generated applications usually get the schema right and the boundary wrong. The boundary is what customers feel.

04 **Tenancy is enforced in the database**

Row-level policies or scoped queries mean a leaked identifier returns nothing, not someone else’s record. Test with two accounts in two tenants and try to read across them.

05 **Secrets stay server-side**

No service key, admin token, or database password in the bundle, in client environment variables, or in a repository. An anon key with write access is a public write endpoint.

06 **Migrations are rehearsed**

Schema changes have a forward path, a rollback decision, and a run against a copy of real data. No manual edits in production that exist nowhere else.

07 **Retention and deletion are real**

Backups, exports, and analytics copies are accounted for. A deletion request removes the record where it actually lives, rather than setting a flag that one screen respects.

03 / Payments

The demo charges a card. Correctness is what happens on the second webhook, the refund, and the failed retry.

08 **Webhooks are idempotent**

The same event delivered twice produces one outcome, keyed by the provider event id and recorded. Test by replaying a real event rather than trusting the happy path.

09 **One state machine per money object**

Charge, authorize, capture, refund, dispute, and expire are explicit transitions with an owner and a timestamp. Status is never inferred from what a screen shows.

10 **Reconciliation runs daily**

Totals reported by the provider are compared with the system’s own numbers, and a divergence alerts a person. The month-end surprise is a design failure from three months earlier.

04 / Observability

If the first signal of failure is a customer email, the system is not observable. It is merely deployed.

11 **Errors carry context**

A failure records the user, tenant, request id, route, release, and the downstream call that failed. A stack trace alone cannot answer who was affected or how badly.

12 **Alerts have an owner**

Every threshold routes to a named person or rota, with a runbook that says what to do first. An alert nobody reads is a log line with ambition.

13 **Spend and rate limits exist**

Model calls, third-party APIs, and database work have timeouts, retries with backoff, per-tenant quotas, and a cost ceiling. A loop that can spend money needs a brake before launch.

05 / Deployment

A deploy you cannot repeat, or undo, is an incident waiting for a Tuesday afternoon.

14 **Builds are reproducible**

Dependencies are pinned, the build runs from a clean checkout, and the same artifact is promoted through environments. “Works on my machine” is a release process, not a bug report.

15 **Staging matches production**

Environment variables, schema, extensions, and integration modes are the same shape. A staging environment that skips the payment provider tests the wrong product.

16 **Rollback is rehearsed**

The previous artifact can be live within minutes and database changes are backward compatible for one release. If rollback means restoring a backup, say so out loud.

17 **Backups are restored, not just scheduled**

Someone has restored one recently and can state the recovery point and recovery time. A backup nobody has opened is a hope with a cron job.

06 / AI behaviour and evaluations

For AI-built products, the model is part of the product. It needs the same release discipline as the code around it.

18 **A golden set exists**

Representative inputs with expected behaviour, including the ambiguous and unsafe cases, versioned alongside the code. Twenty honest cases beat two hundred invented ones.

19 **Evaluations gate releases**

A change to a prompt, model, or tool runs the set before deploy, and a regression blocks the release. Otherwise every tuning change is an untested production experiment.

20 **Failure paths and tool boundaries are designed**

Timeouts, malformed output, refusals, and provider outages have a defined route. Tool permissions are scoped, and consequential actions require an approval a human can see.