FREE · NO EMAIL REQUIRED

The 30-point production-readiness checklist.

What I check before I will call an AI system ready for real users. Six areas, thirty checks, drawn from the failure modes that actually show up once a demo meets production traffic.

Thirty checks across six areas. If you can tick fewer than twenty-five, you have a launch risk rather than a launch date.

01

Retrieval

Most "the model is wrong" bugs are retrieval bugs wearing a costume.

- [ ] 01 **Chunking was chosen by measurement, not by default**
  You tested at least two chunk sizes and overlaps against a labelled query set, and you can say how much the winner won by.
- [ ] 02 **Retrieval is measured separately from generation**
  Recall@k and precision exist for the retriever alone, so a bad answer can be traced to the wrong document rather than blamed on the model.
- [ ] 03 **The index rebuild is reproducible and versioned**
  Rebuilding from source yields the same index, the embedding model is pinned, and you can roll back without re-ingesting everything.
- [ ] 04 **Metadata filtering happens before the vector search, not after**
  Tenant, permission and date filters are applied in the query itself, so a user can never retrieve a chunk they are not entitled to see.
- [ ] 05 **There is a defined answer for "it is not in the corpus"**
  Low-relevance retrievals are detected and the system says it does not know, rather than passing weak context along and letting the model improvise.

02

Evaluation

If you cannot measure a regression, you will ship one and hear about it from a customer.

- [ ] 06 **A labelled evaluation set exists, and it predates the fix you are shipping**
  Held out, written from real user queries, and not generated by the same model you are testing.

- [ ] 07 **Every prompt change runs the eval suite before merge**
  Automated in CI with a pass threshold, so a one-word prompt edit cannot silently regress the whole system.

- [ ] 08 **At least one metric is something a user would recognise**
  Task success or answer correctness — not only cosine similarity, BLEU, or another number that moves without the product improving.

- [ ] 09 **LLM-as-judge is calibrated against human labels**
  You have measured how often the judge agrees with human raters on a sample, so you know the error bars on your own scoreboard.

- [ ] 10 **Regression failures are diffable**
  When the suite fails you can see the exact input, the previous output and the new output side by side, without re-running anything by hand.

03

Cost & latency

A system that works but costs more than it earns is not in production, it is in trouble.

- [ ] 11 **You know the cost per request, not just the monthly bill**
  Broken down across model calls, retrieval and re-ranking, so you know which component to attack first.

- [ ] 12 **Token growth has a ceiling**
  Context assembly runs to a hard budget, so a long conversation or an oversized document cannot quietly triple the unit cost.

- [ ] 13 **Caching exists wherever the traffic is repetitive**
  Prompt or semantic caching is in place and you track the hit rate, because an uninstrumented cache is a guess.

- [ ] 14 **p95 latency is a tracked objective, not an anecdote**
  Measured end to end, including retrieval and time-to-first-token, on the same percentile you would defend in a customer meeting.

- [ ] 15 **There is a cheaper fallback path and you know its quality cost**
  The smaller model has been run against the eval suite, so degrading under load is a decision rather than a surprise.

04

Observability

You will be asked why it said that. You need to be able to answer.

- [ ] 16 **Every generation is traced end to end**
  One request id links retrieval, prompt, model, token counts, latency and output into a single record you can pull up.

- [ ] 17 **Prompts and outputs are logged under a retention and redaction policy**
  You can debug a complaint from last Tuesday without holding personal data you were never supposed to keep.

- [ ] 18 **You can answer "what changed?" in under five minutes**
  Model version, prompt version and index version are stamped on every trace, so a quality cliff has a suspect list.

- [ ] 19 **User feedback joins back to the trace**
  A thumbs-down resolves to the exact request that caused it, rather than to a date and a vague description.

- [ ] 20 **Alerts fire on quality, not only uptime**
  Refusal rate, empty-retrieval rate and schema-validation failures are alarmed — a system can be fully available and completely wrong.

05

Safety & failure modes

Assume the input is hostile and the provider is down. Both will be true eventually.

- [ ] 21 **Prompt injection is tested, not assumed handled**
  You keep an adversarial input set, including injections that arrive inside retrieved documents rather than from the user.

- [ ] 22 **Untrusted content cannot reach a privileged tool**
  Any tool call with a side effect is gated by validation or explicit confirmation, so a retrieved document cannot trigger an action.

- [ ] 23 **Output is validated against a schema before it reaches another system**
  Structured output is parsed and rejected on failure, never passed downstream on the assumption that it parsed correctly.

- [ ] 24 **Rate limits and spend caps exist per tenant**
  One enthusiastic customer, or one retry loop, cannot exhaust the budget for everyone else.

- [ ] 25 **There is a documented degradation path**
  When the provider is slow or unavailable the product does something defined and tested, rather than hanging until the request times out.

06

Operations & handover

The system is not ready if it only runs while one person is awake.

- [ ] 26 **Someone other than the author can ship a prompt change**
  Prompts are versioned in the repository, reviewed, and released like code rather than edited live in a console.

- [ ] 27 **The runbook covers the three most likely failures**
  Provider outage, stale index and quality regression, each with a named owner and a first step.

- [ ] 28 **Provider keys rotate without a redeploy**
  Secrets live in a secret store, scoped per environment, so rotating one is a routine operation rather than a release.

- [ ] 29 **There is a staging environment with realistic data**
  Evaluation and manual QA run somewhere that is not production, against data shaped like the real thing.

- [ ] 30 **The team can explain the system without you**
  Architecture, the decisions behind it and the known trade-offs are written down, including what you deliberately chose not to build.

WHO WROTE THIS

Nitish Kumar

AI SYSTEMS CONSULTANT · NOETIC

I make AI systems survive contact with real users. This checklist is the short version of what I look for in a *production-readiness review*.

Work with me · noeticitservices.com nitish@nits.dev