AI QUALITY CHEAT SHEET

Testing & Evaluating Probabilistic Systems

Use deterministic tests for guarantees, evals for judgment, and real QA to find the cases neither system anticipated.

Core idea: Tests prove rules that should always hold. Evals measure whether probabilistic behavior is good enough. A reliable AI product needs both.

Choose the right evidence for the behavior

Layer	Best for	State example
Unit / contract test	Rules that should always hold.	Rejected Evidence never mutates Current State.
Integration test	Deterministic flows across components.	Evidence → interpretation → Review → accepted transition → History.
End-to-end test	User journeys across browser + API.	Add a Note, review change, confirm Project/History update.
LLM eval	Judgment where several outputs may be plausible.	Did the model correctly judge whether Evidence is consequential?
Manual / adversarial QA	Unexpected behavior and UX seams.	Ambiguous notes, duplicate Reviews, misleading copy, strange navigation.

Golden scenarios

A **golden scenario** is a deliberately chosen case with the expected product judgment written **before** looking at model output. It prevents the model from quietly redefining what "good" means.

GIVEN Current State + relevant rules + incoming Evidence + open Questions/Reviews.

EXPECTED INTERPRETATION Review/no Review, affected State, proposal, or explicit uncertainty.

PROHIBITED BEHAVIOR What the model/software must not infer, mutate, or hide.

AFTER HUMAN ACTION Expected State, History, review lifecycle, and downstream behavior.

A plausible model answer can still be wrong for the product. Valid JSON proves shape, not correct judgment.

AI QUALITY CHEAT SHEET

Testing & Evaluating Probabilistic Systems

Real failures are valuable because they reveal decision boundaries your planned test set probably missed.

Turn every meaningful defect into permanent coverage

Case type	Example	Why it matters
Original failure	"Password reset tickets were approved for automation."	Preserves the exact bug.
Paraphrase	"Security signed off on automating password resets."	Checks the fix is semantic, not phrase-specific.
Positive neighbor	"Automation is live in production."	Ensures real implementation evidence still works.
Negative neighbor	"We may automate password resets later."	Speculation must not become current truth.
Failure / uncertainty	Provider times out after source Evidence is saved.	Checks durable input and safe recovery.

Useful evaluation dimensions

State integrity

Was authoritative state preserved correctly?

Interpretation

Did the model identify the right consequence, items, and proposal?

User burden

Did the system create unnecessary Reviews or blockers?

Groundedness

Did the output stay within what the evidence established?

Failure behavior

Did timeouts, malformed output, stale data, and retries degrade safely?

Consistency

Is behavior acceptable across paraphrases and repeated runs?

When an eval fails

Is expected behavior specified? → **Is the model wrong?** → **Is validation/app logic wrong?** → **Is UI presenting correct behavior badly?**

Use deterministic tests for invariants, evals for judgment, and real QA to discover the cases neither of them imagined.