PM EVALS CHEAT SHEET

AI Product Evals

Define expected behavior, test realistic risk, and use results to make product decisions - not just produce a score.

Five rules: define expected behavior before seeing outputs; severity matters more than one accuracy number; find the first wrong step before fixing; rerun failures plus neighbors and holdouts; use evals to make a product decision.

Tests and evals answer different questions

Deterministic software test

Question: Did the software do what it was programmed to do?

Best for invariants, schemas, permissions, transitions, validation, and known logic.

AI eval

Question: Did the probabilistic behavior work well enough across realistic cases?

Best for interpretation, groundedness, ambiguity, judgment, tone, and uncertainty handling.

What the PM owns

- Which behaviors matter.
- What counts as correct.
- Failure severity and realistic risk.
- Release quality bars and tradeoffs.

Engineering usually owns: repeatable execution, instrumentation, logging, and product integration.

AI can help: draft cases, variants, summaries, and initial analysis.

Human judgment: approves expected behavior and what is acceptable.

Build the first eval set

NORMAL
Representative expected behavior.

PARAPHRASE
Same intent, different wording.

EDGE
Boundary or unusual but plausible case.

UNCERTAINTY
Unknown, tentative, conditional, conflicting.

KNOWN FAILURE
A real defect that should never quietly return.

NEGATIVE / NO-ACTION
Cases where the AI should correctly do nothing.

Write expected behavior before looking at outputs. Otherwise it is easy to redefine success around whatever the model happened to produce.

Simple scoring rubric

Dimension	Ask
Supported	Is the conclusion grounded in the available evidence?
Complete	Did it catch the important information?
Uncertainty preserved	Did unknown/pending information remain uncertain?
Human burden	Did it avoid unnecessary reviews, alerts, or duplicate work?

PM EVALS CHEAT SHEET

AI Product Evals

Treat severity, decision boundaries, and generalization as first-class parts of quality.

Severity beats one overall accuracy score

LOW
Minor inconvenience.

MEDIUM
Misleading or wasteful, but easy to catch.

HIGH
Could materially affect decisions or trust.

CRITICAL
Could directly cause unsafe, irreversible, or externally consequential action.

A strong aggregate score can still fail the release bar if the remaining failure is severe enough.

Test the decision boundary, not just the broken sentence

Case	Expected distinction
Unknown	Remain unknown.
Explicit 0	Allow 0 only when explicitly established.
Tentative 10%	Do not treat as approved.
Approved 10%	Allow a concrete proposal/value.
Observed 8%, target unknown	Do not confuse actual performance with target.
Prior target withdrawn	Return to unknown unless a replacement is established.

After a fix

Original failure → **Nearby boundary cases** → **Positive cases** → **Holdouts** → **Repeat high-risk cases**

Holdouts

Keep some cases that were not directly used to design the fix. Passing tuned examples does not prove the behavior generalizes.

Repeat where it matters

Repeat high-severity, previously failed, ambiguous, or unstable cases more heavily than routine low-risk cases.

Quality is multidimensional

A fix can improve one metric while making another worse. Check **safety, correctness, usability, and human workload** together.

PM EVALS CHEAT SHEET

AI Product Evals

When something fails, diagnose the layer before changing the model or prompt.

Trace the first wrong step

Evidence → **Context / retrieval** → **Model interpretation** → **Validation / post-processing** → **Classification** → **Product action**

Do not change the prompt until you know the failure actually occurred at the model step. If software converted null into 0, a better prompt will not fix the real defect.

What tracing should let you inspect

- ✓ Input evidence and context.
- ✓ Prompt/instruction and model version.
- ✓ Model interpretation.
- ✓ Validation or normalization.
- ✓ Action/review classification.
- ✓ Errors, timing, run or trace ID.

Find silent misses

Visible wrong answers are easier to notice than cases where the AI should have acted but did nothing.

- Sample no-action cases.
- Have humans judge whether inaction was correct.
- Track confirmed false negatives and severity.

Useful product metrics

Miss rate - important changes missed.

Unsupported-claim rate - invented/overstated claims.

Uncertainty-loss rate - uncertain became falsely certain.

Unnecessary-review rate - avoidable human work.

Practical PM eval loop

Define behavior → **Draft cases** → **Baseline** → **Score + severity** → **Trace failure** → **Targeted fix** → **Regression + monitor**

Schema-valid does not mean semantically correct. Correct AI behavior sometimes means doing nothing. Averages should not hide unacceptable failure modes.