TIER 3 - EMPLOYABLE * V1.0 -- MAY 2026

VALUATION AND TESTING AI OUTPUT

The eval framework. Golden test sets. Regression testing when prompts change. How to know "did this update help or hurt" with real signal instead of vibes. Without becoming a full QA engineer.

BY Alex Jahn / Agent Logic

v1.0 -- May 2026 **EDITION**

The person on the team responsible for an AI-powered workflow who needs to know whether a prompt change made things better or worse -- without just eyeballing the next 5 outputs and shrugging **AUDIENCE**

15-20 minutes **READ TIME**

Free. Forever. **COST**

Prepared by Agent Logic / alexanderjahn79@icloud.com / theaiguywi.com

CONTENTS

What's in here

1 **The vibe-based testing trap** 3

You've got an AI workflow running. You tweaked the prompt to fix one specific issue. The new version handles that issue well. You run it...

2 **The golden test set** 4

A golden test set is 10-30 input examples you've decided are representative of the real-world inputs your AI workflow needs to handle....

3 **The 3-tier rubric** 5

For workflows with clear ideal outputs (classification, extraction, structured generation), you can compare outputs against expected...

4 **The AI-grading-AI pattern** 6

For larger golden sets or higher-frequency releases, manually scoring every output gets expensive. The shortcut: have AI score AI...

5 **The regression test workflow** 7

15-30 minutes per prompt change. Compared to silently shipping a regression that goes unnoticed for weeks, the math is obvious.

6 **Two worked sessions** 8

You've built a prompt that generates the discovery-summary section of a proposal from rough call notes. You want to tweak it to be more...

7 **The honest limit** 9

Three categories where formal evaluation stops being worth the time:

8 **Where to go from here** 9

Ten more Tier 3 modules ahead:

SECTION 1

The vibe-based testing trap

"It seems better. I think."

You've got an AI workflow running. You tweaked the prompt to fix one specific issue. The new version handles that issue well. You run it on a couple more inputs. Looks fine.

You ship the change.

Three weeks later you realize the new prompt is silently producing worse output on a CATEGORY of inputs you didn't think to test. The output is "fine-looking" but worse than the previous version on real-world data. You don't know when it happened, or which other prompt change might have been the actual culprit.

This is the vibe-based testing problem. AI workflows are uniquely hard to test because the outputs are open-ended; there's no "compiler error" to tell you something broke. The fix is structure: a small set of test inputs you keep around, a way to compare outputs across prompt versions, and a discipline of running the tests before shipping.

This module is that structure. Lightweight enough for one-person teams. Real enough to catch the regressions vibe-testing misses.

What you'll have by page 12

By the end of this primer:

- The **golden te**
- The **3-tier rub**
- The **regressio**
- The **AI-gradin**
- **Two worked sessions** -- a proposal generator + a customer-support classifier.
- The **honest lir**

> ***Vibe-based testing fails silently. The output looks fine on the 3 cases you remembered to check; it's wrong on the 50 cases you didn't. The fix is a small, deliberate test set you actually run.***

SECTION 2

The golden test set

What it is

A golden test set is 10-30 input examples you've decided are representative of the real-world inputs your AI workflow needs to handle. Each example has either:

- An ideal/expected output (for tasks with clear right answers), OR
- A set of criteria the output must satisfy (for open-ended tasks)

You keep this set in a file. When you change the prompt, you re-run the golden set and compare new outputs against old outputs (or against the expected). You ship the prompt change only if the golden set passes.

That's it. The whole concept. Most teams that get AI eval right don't have anything fancier than this; they just have it.

What to put in the set

> **Golden set composition rules:**

- **5-10 "typical" inputs** -- the most common cases your workflow handles. These are what most outputs look like.
- **3-5 "edge case" inputs** -- weird, malformed, ambiguous cases that historically caused problems.
- **2-3 "wrong-but-plausible" inputs** -- cases that would tempt the model to give a confident-but-wrong answer. Tests robustness.

- **1-2 "should-refuse" inputs** -- cases where the right answer is "I can't / won't / shouldn't help with this." Tests appropriate refusal.

Total: 10-20 inputs. Keep it small enough that re-running through them takes minutes, not hours. The set grows as you find new failure modes -- add an example for each new bug after you fix it.

SECTION 3

The 3-tier rubric

Scoring without manual reading every output

For workflows with clear ideal outputs (classification, extraction, structured generation), you can compare outputs against expected programmatically:

- Exact match -> pass
- Close match (typo / formatting) -> pass with flag
- Different answer -> fail

For open-ended workflows (drafting, summarization, conversation), you can't string-match. Use a 3-tier rubric:

> **The 3-tier rubric:**

For each output, score one of:

- **Pass** -- output satisfies all the must-haves. Ready to use as-is.
- **Edit-and-pass** -- output is on the right track but needs human editing before use.
- **Fail** -- output misses something load-bearing. Needs to be regenerated.

When you change a prompt, you compare the new and old runs across the golden set:

- If pass-rate goes up and edit-and-pass stays roughly equal -> ship.
- If pass-rate goes up but edit-and-pass also goes up significantly -> mixed signal, dig in.
- If pass-rate stays the same and edit-and-pass goes up -> the change made outputs worse on average.

- If pass-rate goes down -> don't ship.

This is dramatically more rigorous than "eyeballed a couple, looked fine." The cost is 20 minutes of scoring per release -- but it's catching the regressions you'd otherwise ship.

SECTION 4

The AI-grading-AI pattern

Letting AI score AI outputs

For larger golden sets or higher-frequency releases, manually scoring every output gets expensive. The shortcut: have AI score AI outputs against your rubric.

The AI-grading prompt:

> *"Below is an AI-generated [type of output -- e.g., proposal recap]. Score it against this rubric: - Pass: contains [list specific must-haves] - Edit-and-pass: contains most must-haves but is missing or weak on [list things] - Fail: missing load-bearing content Output: [paste output] Reply with one of: Pass / Edit-and-pass / Fail. Then 1-2 sentences explaining your call."*

The trick: use a DIFFERENT model to grade than the one that generated. If you use Claude to generate, use GPT to grade (or vice versa). Different models catch different failure modes.

When AI-grading works

Works well:

- Classification or extraction tasks (the right answer is unambiguous)
- Pattern-match tasks ("does this contain the customer's name?")
- Format compliance ("is this valid JSON with the required fields?")

Works poorly:

- Subjective quality assessments ("is this written well?") -- AI's "well-written" doesn't match yours
- Truth-verification ("is this factually accurate?") -- AI can't verify facts it doesn't have

- Voice/tone matching ("does this sound like our brand voice?") -- AI averages voices

For the works-poorly cases, manual scoring on a smaller set is still the move. The AI-grade-AI pattern is for the unambiguous tasks where you can scale to 100+ test cases.

SECTION 5

The regression test workflow

> **The regression test workflow (before every prompt change):**

1. Save the current prompt + its golden-set scores ("baseline").
2. Make the prompt change you're testing.
3. Re-run the golden set with the new prompt.
4. Score the new outputs (manually or via AI-grading).
5. Compare: did pass-rate improve? Did anything regress?
6. If clear win -> ship. If mixed/regressed -> don't ship; iterate or revert.

15-30 minutes per prompt change. Compared to silently shipping a regression that goes unnoticed for weeks, the math is obvious.

What gets stored

For each prompt version, save:

- The prompt itself (versioned -- date or v1, v2, v3)
- The golden set you ran it against (same set across versions)
- The outputs (so you can re-grade later if scoring criteria evolve)
- The scores (Pass / Edit-and-pass / Fail per case)

This becomes your eval history. After 10-15 prompt changes, the history tells you the actual trajectory of your workflow -- whether you're improving or going sideways.

SECTION 6

Two worked sessions

Worked session 1 -- A proposal generator

You've built a prompt that generates the discovery-summary section of a proposal from rough call notes. You want to tweak it to be more concise.

Golden set (12 cases):

- 6 typical: notes from past discovery calls (your actual archive)
- 3 edge cases: very short notes, very long notes, notes with no clear pain point named
- 2 wrong-but-plausible: notes where the prospect might be a bad fit
- 1 should-refuse: notes mentioning illegal scope (rarely seen, but worth testing)

Process:

1. Run the original prompt against all 12 cases. Score outputs.
2. Tweak the prompt for conciseness.
3. Re-run against the same 12 cases. Score outputs.
4. Compare: original had 8 Pass / 3 Edit-and-pass / 1 Fail. New version has 7 Pass / 4 Edit-and-pass / 1 Fail.
5. Verdict: don't ship. The conciseness tweak moved 1 case from Pass to Edit-and-pass. Investigate why -- probably the prompt cut a section that was load-bearing for some inputs.

That's a regression you caught BEFORE 50 real proposals were generated badly. Worth the 20 minutes.

Worked session 2 -- A customer-support classifier

You've built an AI workflow that classifies inbound support emails into 5 categories. You want to add a 6th category and see if it harms accuracy on the other 5.

Golden set (40 cases -- bigger because classification is fast to grade):

- 8 cases per category for the original 5 (40 baseline)

- Add 8 examples of the new 6th category

Process:

1. Run original prompt (5 categories) against the 40 baseline cases. Compute accuracy.
2. Modify the prompt to add the 6th category. Run against all 48 cases (the 40 baseline + 8 new).
3. Compare: was accuracy on the original 5 categories maintained? Is the new 6th category being recognized?

AI-grade the classification matches against expected labels. 5 minutes of compute. You ship the change only if accuracy on the original 5 didn't drop more than 2-3 percentage points.

SECTION 7

The honest limit

Three categories where formal evaluation stops being worth the time:

- **One-off / exploratory uses.** If you're using AI for a single task today and won't use the same workflow again, building an eval set is overhead. Just review the output and ship.
- **Highly subjective tasks.** Voice, tone, brand fit -- these aren't scoreable in a useful way. Manual review by the right human is the right test.
- **Tasks where the right answer is genuinely fuzzy.** If even three humans would disagree about what "good" looks like, an eval set forces a false precision. Better to acknowledge the fuzziness and operate accordingly.

Within those limits, the golden-set + rubric + regression-test pattern is one of the most-underused AI quality practices. Most teams hand-wave it. The teams that do it ship more reliable AI workflows over years.

SECTION 8

Where to go from here

Ten more Tier 3 modules ahead:

- **RAG depth -- beyond the primer** -- chunking strategy, retrieval ranking, embedding choice, the production patterns.

Get the next module the day it drops: theaiguywi.com/training

If you want this eval discipline installed for a specific AI workflow you're already running -- the golden set built, the rubric defined, the regression-test routine adopted -- that's the consulting offer.

Reach out: alexanderjahn79@icloud.com

Closing -- the lock-in line

The vibe-based test is the silent killer of AI workflows. The fix is small: 10-20 inputs you keep around, a 3-tier rubric, a discipline of running tests before shipping. Compared to the cost of a regression silently degrading outputs for weeks, the time investment is trivial.

20

Twenty minutes per prompt change.

That's the regression-test routine cost. It catches the failures that would have shipped silently. The teams that do this have measurably more reliable AI workflows. The teams that skip it find out the hard way.

-- Alex

Agent Logic -- *theaiguyw*
Lac, WI. This is module 8 of 18 in Tier 3 (Employable).