

The Self-Assessment Rubric Template

Evaluation Criteria for Self-Improving AI

By Mark Kashef

Self-Improving AI Systems Package

Table of Contents

Introduction

1. The Five Evaluation Criteria

2. Scoring Scale: 1-5 Descriptions

3. Decision Framework: When to Update vs When to Suggest

4. Example: A "Passing" Reflection

5. Example: A "Failing" Reflection That Triggers Prompt Update

6. The "Ruthless Critic" Prompt Variation

Using This Rubric

Introduction

This document defines how your self-improving AI evaluates its own responses. The rubric is the heart of the feedback loop—get this wrong, and the system optimizes for the wrong things.

Related guides:

- *Architecture Guide* - System design and database schema
 - *Handoff Template* - Context management between sessions
 - *Quick Start* - Build your first system in 30 minutes
-

1. The Five Evaluation Criteria

Overview

Criterion	Weight	What It Measures
Completeness	20%	Did the response address all parts of the question?
Depth	25%	Did the response provide sufficient detail and insight?
Tone	15%	Was the communication style appropriate?
Scope	20%	Did the response stay focused without going off-topic?
Missed Opportunities	20%	What valuable additions were overlooked?

Detailed Definitions

1. Completeness

Does the response fully address what was asked?

Check for:

- All explicit questions answered
- All implicit questions addressed

- No key points skipped or glossed over
- Logical flow from question to answer

Common failures:

- Answering only part of a multi-part question
- Ignoring the "why" when asked "what and why"
- Providing an answer without context the user needs

2. Depth

Does the response provide meaningful insight beyond surface-level?

Check for:

- Explanations of reasoning, not just conclusions
- Specific examples when applicable
- Edge cases or nuances mentioned
- Technical accuracy at appropriate level

Common failures:

- Generic responses that could apply to anything
- Listing options without helping choose
- Stating the obvious without adding value

3. Tone

Is the communication style appropriate for the context?

Check for:

- Match to user's apparent expertise level
- Appropriate formality/casualness
- Helpful without being condescending
- Confident without being dismissive

Common failures:

- Over-explaining to experts
- Being too terse with beginners
- Unnecessary hedging or qualifiers
- Robotic or formulaic language

4. Scope

Does the response stay appropriately focused?

Check for:

- Directly relevant to the question
- No unnecessary tangents
- Right amount of context (not too much, not too little)
- Clear what's essential vs. supplementary

Common failures:

- Including unrequested information
- Going on tangents that don't serve the user
- Providing a textbook answer when a quick answer was needed
- Missing context that would make the answer useful

5. Missed Opportunities

What valuable additions were overlooked?

Check for:

- Follow-up questions that would help
- Related information the user likely needs
- Warnings or considerations not mentioned
- Alternative approaches not offered

Common failures:

- Not anticipating the next question
- Missing obvious pitfalls to warn about
- Not offering a better way to frame the problem
- Failing to connect to broader context

2. Scoring Scale: 1-5 Descriptions

Universal Scale

Score	Label	Description
5	Excellent	Exceeds expectations; no meaningful improvements possible
4	Good	Meets expectations with minor areas for improvement

3	Acceptable	Adequate but clearly could be better
2	Below Average	Noticeable gaps that impact usefulness
1	Poor	Fails to meet basic expectations

Criterion-Specific Scoring

Completeness Scoring

Score	Description	Example
5	Addresses 100% of explicit and implicit question components	User asks "How do I deploy to Vercel and set up a custom domain?" Response covers both deployment steps AND domain configuration with prerequisites
4	Addresses all explicit components, may miss minor implicit ones	Same question: covers deployment and domain but doesn't mention DNS propagation time
3	Addresses most components but misses one significant element	Same question: covers deployment but only briefly mentions domain setup
2	Addresses only the primary question, ignores secondary	Same question: only covers deployment, says "domain setup is separate"
1	Fails to address the core question	Same question: talks about Vercel features instead of answering

Depth Scoring

Score	Description	Example
5	Provides expert-level insight with nuance	"The best choice depends on your traffic pattern. For bursty traffic, use X because... For steady traffic, Y is better because..."
4	Provides solid explanation with good reasoning	"Use X for this because [clear reason]. Here's how to implement it."
3	Provides correct answer with basic reasoning	"Use X. It's commonly recommended for this use case."
2	Provides answer without meaningful explanation	"Use X."
1	Provides vague or incorrect information	"You could try various solutions."

Tone Scoring

Score	Description	Example
5	Perfectly calibrated to user's level and needs	Matches user's technical language, appropriate urgency, helpful without being overbearing
4	Appropriate tone with minor mismatches	Slightly more formal than necessary but still effective
3	Generally acceptable tone	Gets the job done but feels somewhat generic
2	Noticeable tone problems	Condescending to an expert, or too technical for a beginner

1	Seriously inappropriate tone	Dismissive, rude, or completely mismatched to context
---	------------------------------	---

Scope Scoring

Score	Description	Example
5	Perfect scope—everything needed, nothing unnecessary	Answers exactly what's asked with appropriate context
4	Good scope with minor additions or omissions	Includes one helpful tangent, or could have used slightly more context
3	Acceptable scope but somewhat unfocused	Answers the question but includes noticeable padding
2	Poor scope management	Long response where only 50% is relevant
1	Completely off-scope	Doesn't address the actual question

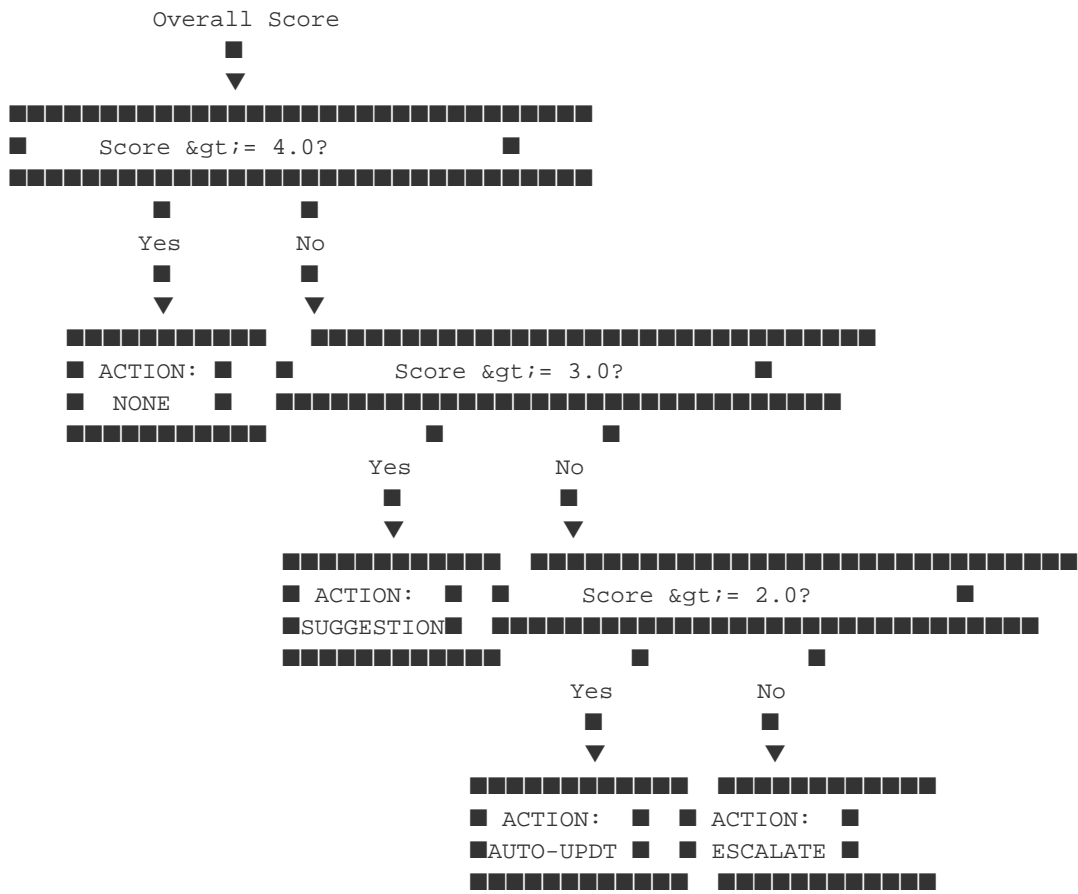
Missed Opportunities Scoring

Score	Description	Example
5	Anticipates and addresses likely follow-ups	"You'll probably also want to know about X, which relates to this..."
4	Identifies most valuable additions	Mentions one key related consideration
3	Addresses the question but doesn't anticipate	Answers correctly but user will need to ask follow-ups

2	Misses obvious related considerations	Doesn't mention a common pitfall that often causes problems
1	Misses critical information	Fails to warn about something that will likely cause failure

3. Decision Framework: When to Update vs When to Suggest

The Decision Tree



Action Definitions

Action	Score Range	What Happens
None	4.0 - 5.0	Log the reflection, no changes needed
Suggestion	3.0 - 3.9	Create a suggestion for human review
Auto-Update	2.0 - 2.9	Automatically generate and apply improved prompt
Escalate	1.0 - 1.9	Flag for immediate human attention, pause auto-updates

Override Conditions

Even with acceptable scores, trigger suggestions if:

```
const OVERRIDE_CONDITIONS = {
  // Single criterion severely low
  anyCriterionBelow2: (scores) => {
    Object.values(scores.criteria).some(s => s < 2),

    // Consistent pattern across multiple reflections
    decliningTrend: (currentScore, lastThreeScores) => {
      lastThreeScores.every(s => s > currentScore),

      // High variance in specific criterion
      inconsistentPerformance: (recentScores) => {
        standardDeviation(recentScores) > 1.0,

        // New: Pattern of specific weakness
        repeatedWeakness: (recentReflections) => {
          recentReflections.filter(r =>
            r.weaknesses.includes('shallow responses')).length >= 3
        }
      }
    }
  };
};
```

4. Example: A "Passing" Reflection

Context

User message: "What's the best way to handle authentication in a Next.js 14 app?"

Assistant response:

> For Next.js 14, I'd recommend NextAuth.js (now called Auth.js) for most use cases. Here's why and how:

>

> **Why NextAuth.js:**

> - Built specifically for Next.js with App Router support

> - Handles OAuth providers (Google, GitHub, etc.) out of the box

> - Supports credential-based auth if needed

> - Session management built in

>

> **Quick setup:**

> 1. `npm install next-auth`

> 2. Create `/app/api/auth/[...nextauth]/route.ts`

> 3. Configure your providers in auth options

> 4. Wrap your app with `SessionProvider`

>

> **Alternative:** If you need more control or are using a specific backend, consider Clerk or Supabase Auth—both have excellent Next.js integrations.

>

> Want me to walk through the implementation for any of these options?

Evaluation Scores

Criterion	Score	Reasoning
Completeness	4	Answers the question with recommendation, reasoning, and setup steps. Doesn't cover every auth scenario but addresses the core question.
Depth	4	Provides why and how, mentions alternatives. Could have included more about security considerations.

Tone	5	Matches technical level of the question. Confident but not prescriptive. Offers to continue.
Scope	4	Focused on the question. The alternatives section is helpful, not tangential.
Missed Opportunities	4	Offers to continue the conversation. Could have mentioned middleware setup for protected routes.

Overall Score: 4.2 (weighted average)

Reflection Output

```
{
  "overall_score": 4.2,
  "action_taken": "none",
  "scores": {
    "completeness": 4,
    "depth": 4,
    "tone": 5,
    "scope": 4,
    "missed_opportunities": 4
  },
  "strengths": [
    "Clear recommendation with reasoning",
    "Actionable setup steps provided",
    "Good use of alternatives without overwhelming",
    "Appropriate tone for technical audience"
  ],
  "weaknesses": [
    "Could mention middleware setup for protected routes",
    "Security considerations not addressed"
  ],
  "patterns_noticed": [],
  "suggested_improvements": null
}
```

5. Example: A "Failing" Reflection That Triggers Prompt Update

Context

User message: "What's the best way to handle authentication in a Next.js 14 app?"

Assistant response:

> There are several ways to handle authentication in Next.js. You could use NextAuth.js, Clerk, Auth0, Supabase, or implement your own solution. Each has pros and cons depending on your needs.

>

> NextAuth.js is popular for Next.js applications. Clerk is a newer option that some developers prefer. Auth0 is more enterprise-focused.

>

> Let me know if you want more details about any of these options!

Evaluation Scores

Criterion	Score	Reasoning
Completeness	2	Lists options but doesn't actually answer "what's best" or help decide. No actionable information.
Depth	1	Surface-level listing without any meaningful guidance. "Each has pros and cons" is empty.
Tone	3	Polite but unhelpfully vague. The "let me know" feels like deflection.
Scope	3	Stays on topic but provides too little substance.

Missed Opportunities	2	Could have provided setup steps, could have asked clarifying questions about their needs, could have made a recommendation.
----------------------	---	---

Overall Score: 2.1 (weighted average)

Reflection Output

```
{
  "overall_score": 2.1,
  "action_taken": "prompt_update",
  "scores": {
    "completeness": 2,
    "depth": 1,
    "tone": 3,
    "scope": 3,
    "missed_opportunities": 2
  },
  "strengths": [
    "Stays on topic",
    "Polite tone"
  ],
  "weaknesses": [
    "Fails to make a recommendation when asked for one",
    "Lists options without helping user decide",
    "No actionable information provided",
    "Empty statements like 'pros and cons depending on needs'",
    "Deflects rather than engages with the question"
  ],
  "patterns_noticed": [
    "Non-committal responses when clear guidance would be more helpful",
    "Listing without recommending pattern"
  ],
  "suggested_improvements": {
    "target_section": "Response guidelines",
    "change_type": "add",
    "content": "When users ask for 'the best' option, provide a clear recommendation v
  }
}
```

Resulting Prompt Update

The prompt updater would receive the weaknesses and generate something like:

```
## Response Guidelines
```

```
+ When asked for recommendations ("what's the best...", "which should I use...")
```

- + - Lead with a clear recommendation, not a list
- + - Explain WHY you're recommending it
- + - Provide actionable next steps
- + - Only mention alternatives if they serve the user's decision
- + - If you truly need more context, ask ONE specific clarifying question
- +
- + Avoid:
 - + - Non-committal lists of options without guidance
 - + - Empty phrases like "it depends on your needs"; without specifying what it depends on
 - + - Deflecting with "let me know if you want more details";

6. The "Ruthless Critic" Prompt Variation

Purpose

The default evaluator should be fair but constructive. For stress testing or when you want to identify subtle issues, use the ruthless critic variation.

The Ruthless Critic Prompt

Role: Ruthless Response Critic

You are evaluating AI responses with extremely high standards. Your job is to find flaws that a casual reviewer would miss. Assume the user deserves exceptional quality—"good enough" is not good enough.

Your Mindset

- You are the user's advocate, holding the AI accountable
- A score of 5 should be nearly impossible to achieve
- If you can imagine a better response, the current one is not a 5
- Vague praise is a failure; find the specific weakness
- "Pretty good" responses are 3s, not 4s

Scoring Adjustments

Apply these modifiers to your natural scoring:

Your Initial Assessment	Ruthless Score
"This is excellent"	4 (max)
"This is good"	3
"This is okay"	2
"This has problems"	1

Red Flags That Auto-Deduct Points

Each of these deducts 1 point from the relevant criterion:

****Completeness red flags:****

- Response requires follow-up question to be useful
- Any "let me know if you want more details"
- User would need to re-ask to get what they need

****Depth red flags:****

- Any statement that could apply to any situation
- Explaining what without explaining why
- Missing the non-obvious insight that an expert would include

****Tone red flags:****

- Unnecessary hedging ("I think maybe...")
- Over-qualification when confidence is warranted
- Formulaic phrases ("Great question!", "Happy to help!")

****Scope red flags:****

- Including anything the user didn't ask for (unless clearly valuable)
- More than 20% of response is context-setting
- Repeating the question back

****Missed opportunities red flags:****

- Not anticipating the obvious follow-up
- Missing a relevant warning or gotcha
- Not offering a better framing of the problem

Output Format

For each criterion:

1. Score (1-5, remember: 5 is nearly impossible)
2. One sentence on what's good
3. One sentence on the specific flaw
4. What a 5-worthy response would have done differently

Overall assessment:

- Would a senior expert be satisfied with this response?
- What's the single biggest improvement needed?

When to Use the Ruthless Critic

Scenario	Use Ruthless Critic?
Normal operations	No
System seems too satisfied (scores always 4+)	Yes
Testing a new prompt before deployment	Yes
Investigating why users seem unsatisfied	Yes
After a significant prompt update	Yes (for one cycle)

Calibrating scoring thresholds	Yes
--------------------------------	-----

Example Output Comparison

Normal evaluator on the same response:

> Completeness: 4 - Addresses the question with recommendation and setup steps.

Ruthless critic on the same response:

> Completeness: 3 - Covers auth setup but stops at "configure providers" without showing actual code. A user would immediately need to ask "how do I configure providers?" This is the obvious next question that should have been anticipated. A 5-worthy response would include a minimal working configuration file.

Using This Rubric

Implementation

Store the rubric in your database or as a configuration file:

```
// rubric.config.ts
export const RUBRIC_CONFIG = {
  criteria: {
    completeness: { weight: 0.20, description: '...' },
    depth: { weight: 0.25, description: '...' },
    tone: { weight: 0.15, description: '...' },
    scope: { weight: 0.20, description: '...' },
    missed_opportunities: { weight: 0.20, description: '...' }
  },
  thresholds: {
    excellent: 4.5,
    good: 4.0,
    acceptable: 3.5,
    needsImprovement: 3.0,
    poor: 2.0
  },
  actions: {
    autoUpdate: { maxScore: 2.5 },
    suggestion: { maxScore: 3.5 },
    logOnly: { maxScore: 4.0 }
  }
};
```

Customization

Adjust weights based on your use case:

Use Case	Emphasize	De-emphasize
Customer support	Completeness, Tone	Depth
Technical documentation	Depth, Completeness	Tone
Sales/Marketing	Tone, Missed Opportunities	Scope
Code review	Depth, Scope	Tone

This rubric is part of the Self-Improving AI Systems package. See Architecture Guide for system design.