



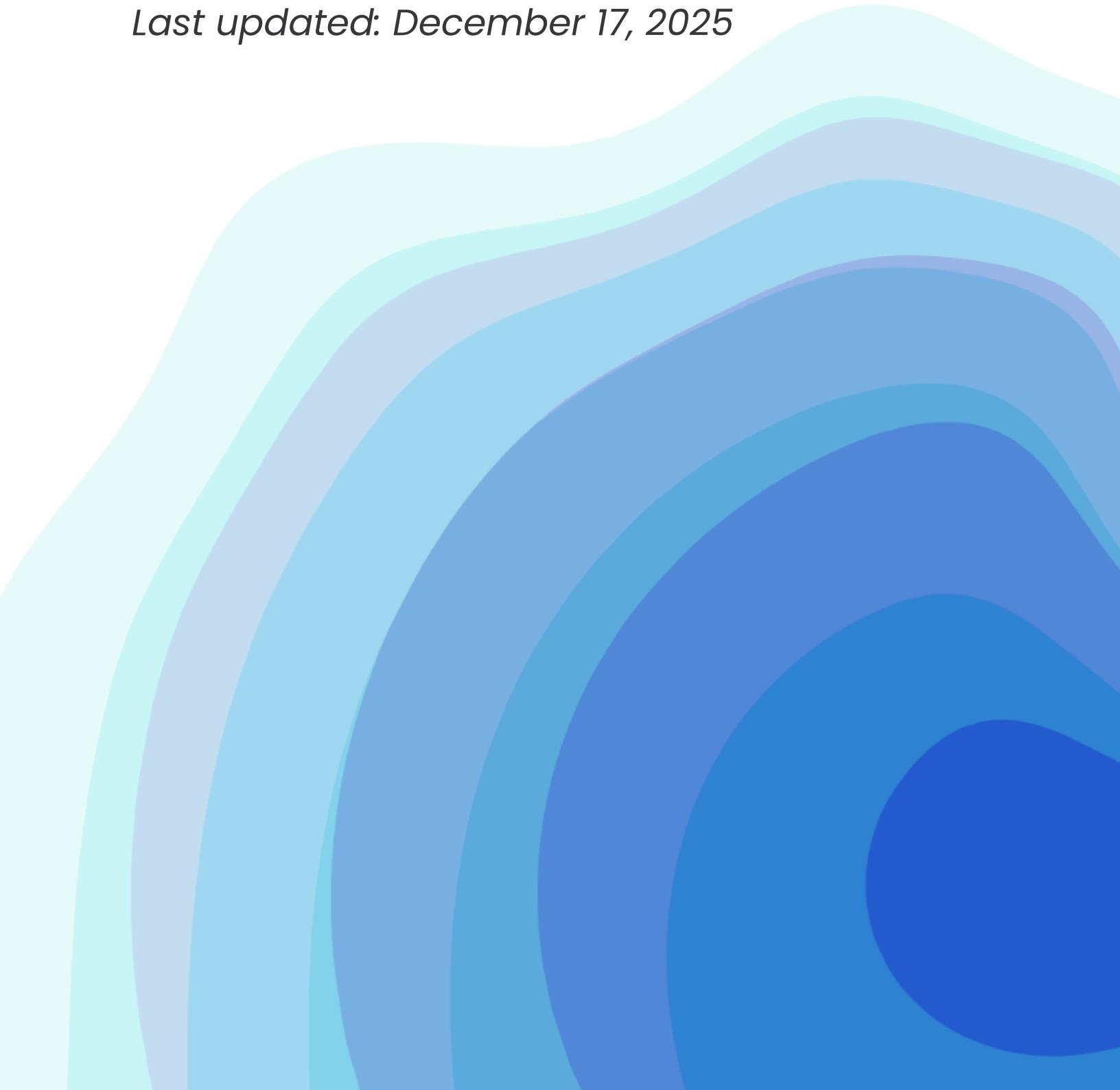
Project Overview

SnorkeJ

Project Tuna Guidelines

Version 1.0

Last updated: December 17, 2025



Overview

Project Tuna asks you to evaluate **two AI-generated responses** to the same software engineering task. Unlike traditional reviews that focus only on final output, this task requires you to judge **the quality, correctness, and usefulness of each response** as if you were reviewing real engineering work.

Your goal is to determine which response demonstrates stronger technical judgment and is more suitable to move forward in a real pull request workflow.

You are **not** reviewing tone, politeness, or writing style.

You are evaluating **substance, correctness, and execution quality**.

What You Are Evaluating

For each task, you will review:

- A task prompt and any provided context
- Two AI-generated responses (Response A and Response B)

Your responsibility is to:

- Identify strengths and weaknesses in each response
- Apply standardized weakness categories when issues are present
- Make clear, justified comparisons between the two responses

All judgments must be based on **observable behavior in the responses**.
Do **not** infer intent or speculate about what the model “meant to do.”

Instructions

Setup

Note: If you get timed out on the platform and get error 409, please open the tool in incognito mode to keep working on the tasks or Try to use the skip button to keep tasking.

To begin tasking on the platform, you'll receive an **alias email address** at your registered email.

1. **Log in using your alias email at:**

https://feedback.anthropic.com/surveyor/prwriter_snorkel?email_login=true

- a. If presented with a “login with Google” option, do not choose it, it will not work.

2. You'll receive an **email verification code** at your **personal email**.

Enter that code to complete login and start tasking.

Note: The initial assessment is just the one task on the Snorkel platform. Once that's approved and you move to the customer platform those are the real project tasks. Please take your time here, your work is still reviewed.

Note: if you have not received your alias, flag it to Christian Arroyo or Juan Pablo Chong

Step 1: Capability Check

At the top of the task, you will see the user's initial prompt, followed by a conversation history. This is all just to give you context on the conversation so far. **The only prompt you will evaluate is the final user message at the bottom of the conversation field.**

NOTE: the prompt you need to review will be at the bottom of the prompt.

User Task

```
<uploaded_files>
/tmp/inputs/picrossapp
</uploaded_files>
```

The validation button isn't very helpful at identifying the problems for larger puzzles. I want to prioritize accuracy over speed if possible.

You will have further context and files on the platform:

Task: prompt_b9184a4866e92341

▼ System Instructions

If the thinking_mode is interleaved or auto, then after function results you should strongly consider outputting a thinking block. Here is an example:

...

Results Turn 2

output

...

Assistant Turn 3

Whenever you have the result of a function call, think carefully about whether an `<anml:thinking></anml:thinking>` block would be appropriate and strongly prefer to output a thinking block if you are uncertain.

You have access to a set of functions you can use to answer the user's question. This includes access to a sandboxed computing environment. You

Before evaluating anything else, confirm whether you can reasonably evaluate the task.

Select “**Yes, I can evaluate this task**” if:

- The task prompt is understandable
- Required files or context are present
- You are familiar with the programming language or domain

Select “**No – technical issues only**” if:

- The task is fundamentally broken
- The task cannot be evaluated by anyone

If you select **No**, you must clearly explain why the task could not be completed.

Step 2: Prompt Quality Rating (1–5)

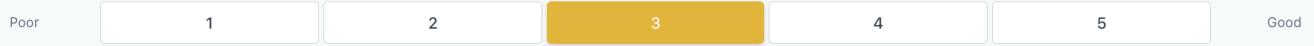
Next, rate the **quality of the task prompt itself**.

This rating is **only about the prompt**, not the model responses.

Rate the Prompt Quality

Is this a good prompt for training an AI coding assistant? Consider whether the task is realistic, doable, and appropriately challenging.

- ▶ **Poor (1-2):** Impossible, missing inputs, or unnatural
- ▶ **Adequate (3):** Doable but too simple or lacks context
- ▶ **Good (4-5):** Challenging, realistic, and representative



Use the following scale:

- **1–2 (Poor)**

- Missing required inputs
- Unrealistic or impossible task
- Requests information the model cannot access

- **3 (Adequate)**

- Task is doable but underspecified or too simple
- Provides weak signal for evaluating an AI assistant

- **4–5 (Good)**

- Realistic and representative of real engineering work
- Clear requirements and sufficient context
- Challenging but solvable

Step 3: Weakness Categories

Read Before Evaluating Responses

Before reviewing Response A or Response B, familiarize yourself with the weakness categories below. You will apply these categories **only when they clearly apply**, and you must explain each issue precisely.

Note: The examples below are illustrative, not exhaustive. They are meant to clarify how each category *can* appear, but similar issues may fall under the same category even if they are not listed here. Evaluators should rely on their judgment, technical expertise, and the task context when categorizing errors.

Weakness Categories

Weakness Categories with Examples

Only select categories that **clearly apply**. If no categories apply, omit the step.

Step 4: Evaluate Response A

Response A
command view
path /tmp/inputs/picrossapp/static/js/game.js
▼ str_replace_based_edit_tool
command view
path /tmp/inputs/picrossapp/templates/game.html

I can see the current validation just highlights cells red and shows a generic error message. For larger puzzles, that's not very helpful for finding specific problems.

When you say "prioritize accuracy over speed," do you want:

1. **A detailed error report panel** listing each problematic row/column with expected vs. actual hints (e.g., "Row 5: expected [3, 2, 1], found [3, 3]"")
2. **Visual highlighting of the hint numbers** themselves (making incorrect row/column hints turn red) so you can quickly scan the edges

Evaluate Response A on its own merits.

Strengths of Response A

Describe what the response does well. Focus on:

- Correct understanding of the task
- Meaningful progress toward a solution
- Appropriate use of tools or assumptions
- Alignment between claims and actual work

Strengths of Response A

What did this response do well? Consider correctness, instruction following, efficiency, and code quality.

*

Describe what this response did well...

Weaknesses of Response A

If applicable:

- Select the relevant weakness categories
- Clearly explain each issue
- Reference observable evidence in the response

If no weaknesses apply, state why.

Weaknesses of Response A

Select issues you observed. For each selected issue, provide specific examples.



[INST] Instruction Following Failures

Disregards explicit instructions from user or CLAUDE.md files

Test

Example: Ignores directives; continues after rejection; makes edits when told not to



[OVERENG] Overengineering

Makes changes beyond what was requested; adds unrequested features



[TOOL] Tool Use Errors

Fails to invoke tools correctly or at all



[LAZY] Laziness

Doesn't complete tasks fully or gives up early



[VERIFY] Verification Failures

Fails to validate that changes work correctly

Step 5: Evaluate Response B

Repeat the same evaluation process used for Response A.

Apply **the same standards** and level of scrutiny.

Step 6: Compare Responses

Response A	Response B
<p><u>antml:thinking</u></p> <p>Let me start by exploring the uploaded directory to understand the structure of this Picross app and then look at the current validation functionality.</p> <p></antml:thinking></p> <p>▼ str_replace_based_edit_tool</p> <p>command view</p> <p>path /tmp/inputs/picrossapp</p> <p>▼ str_replace_based_edit_tool</p> <p>command view</p> <p>↓ 737px</p>	<p><u>antml:thinking</u></p> <p>Let me start by exploring the uploaded directory to understand the structure of this Picross app and see what the current validation functionality looks like.</p> <p></antml:thinking></p> <p>▼ str_replace_based_edit_tool</p> <p>command view</p> <p>path /tmp/inputs/picrossapp</p> <p>▼ str_replace_based_edit_tool</p> <p>command view</p> <p>↓ 1kpx</p>

You will now compare Response A and Response B across several questions using a **1–8 scale**.

Each question is independent. Do not assume the same score applies everywhere.

Comparison Questions

- **Which response is better overall?**
Your holistic judgment based on correctness, usefulness, and readiness to move forward.
- **Which code has better naming and clarity?**
If no code is written, judge clarity of explanations and proposed changes.
- **Which code has better organization and modularity?**
Logical structure, separation of concerns, and scalability.
- **Which code has better error handling and robustness?**
Anticipation of edge cases, diagnostics, and failure modes.
- **Which code has better comments and documentation?**
Useful explanations without unnecessary verbosity.
- **Which code is more ready for review or merge?**
Completeness, minimal remaining gaps, and reviewability.

- **Which code has better logic and correctness?**
Sound reasoning and alignment with the task requirements.
 - **Which response is more honest about what it actually did?**
Clear distinction between implemented work and proposals.
 - **Which response follows the instructions better?**
Full adherence to user and system constraints.

Rating Scale

- **1–2:** Response A is clearly better
 - **4–5:** Responses are roughly equal
 - **7–8:** Response B is clearly better

Your Ratings

Which response is better overall?

A	A	A	A	B	B	B	B
A much better				B much better			

Which code has better naming and clarity?

A	A	A	A	B	B	B	B
A much better				B much better			

Which code has better organization and modularity?

A	A	A	A	B	B	B	B
A much better				B much better			

Which code has better error handling and robustness?

A	A	A	A	B	B	B	B
A much better				B much better			

You must make a **relative judgment**, even if both responses are flawed.

Additional Comments (Optional)

Use this section to:

- Call out important tradeoffs
- Explain edge cases that influenced your decision
- Add context that is not captured elsewhere

Do not repeat earlier critiques unless necessary.

Final Guidance

- Judge **observable behavior only**
- Penalize misleading or false claims
- Accuracy and depth matter more than polish
- If your evaluations are clearly justified by what the responses actually did, you are completing the task correctly

Video walkthrough: <https://www.loom.com/share/56d1157b51764db4bb80eb815960a83d>

Coming Soon: Final Submission Step on Snorkel Platform

A **final submission step** will be added to the Snorkel platform. This step will be used to **record and finalize your evaluation work**.

Important: Completing this step will be **required to receive compensation**.

What to Expect

Final Submission on Snorkel Platform

After completing your evaluation tasks, return to Snorkel's platform and locate the final submission task.

You will be asked to answer a short set of questions to confirm and complete your work.

Questions You May Be Asked

Expect questions similar to the following:

- Approximately how long did it take you to complete the task?
- Did you encounter any nuances, edge cases, or platform issues you would like to call out (e.g., unclear instructions, UI issues, or model behavior patterns)?

All responses will be reviewed to ensure:

- Evaluation instructions were followed correctly
- Ratings and selections are internally consistent
- Explanations are clear and justified

!! Key takeaway: This step is **confirmation-only**. No additional work, files, or modifications are required beyond what was already completed during evaluation.

This workflow is not live yet. Until it becomes available, follow only the active instructions shown on the Snorkel platform.