

Coding Agent Evaluation

You will interact with a coding agent while role-playing as a normal user who has encountered a problem in a GitHub repository. You will collaborate with the agent to resolve the problem.

You are given: A **codebase** and a **problem statement** describing a real issue. Your task is to simulate a realistic user–agent conversation, and **evaluate** whether the agent can:

- *Understand* vague or incomplete user descriptions.
- *Ask* good clarification questions.
- *Progress* toward resolving the issue.

While role-playing, you will be given a **user preference**. You need to ensure that the simulated user remains consistent with this preference. Evaluate whether the agent's questions and responses adapt to or respect this preference.

If you have any questions regarding the annotation system, tasks, payment, or coordination, please email all three of the following addresses and make sure your subject line starts with **[CodeEval]** so we can identify and respond promptly:

sunnweiwei@gmail.com

xuhuiz@cs.cmu.edu

weihuad@cs.cmu.edu

1. Registration Form

You can find all the tasks in this form:

https://docs.google.com/spreadsheets/d/114IkAO_HJLmFYVCrbIpSUIXycJZiO6bdblwRIQhlLX8/edit?usp=sharing

This form is used to register tasks and submit confirmation links.

Each **instance_id** corresponds to one coding problem and is associated with multiple **agent_id** values. Each **data_link** represents a single task. Since all tasks under the same **instance_id** share the same problem, it is recommended to complete all **data_link** entries for one **instance_id** together. The only difference across these tasks is the coding agent used, which makes subsequent tasks easier once you understand the problem.

To start:

- 1) Find a row where **annotator_name** is blank.
- 2) Enter your username in the **annotator_name** field.
- 3) Open the corresponding **data_link**. The first time you open a link, you will need to log in using your name or alias. See Section 2 for more details.
- 4) Complete the task in the linked page.

- 5) After finishing, you will receive a confirmation link.
- 6) Copy the confirmation link and paste it into the **confirmation_link** cell in the form to submit your results.

Note that this form is used only for task registration and coordination. It is not used to calculate the final number of completed tasks.

Task completion is determined based on our backend database. **We will manually verify all the submitted data and task status in the system** to confirm whether a task has been successfully submitted and to process payment for the annotator who submitted the task in the annotation system. If a task is registered but a valid confirmation link is not submitted within 3 hours, the registration will be removed.

Coding Agent Evaluation Instruction (Please read the instruction first!): http://sf.lti.cs.cmu.edu:3000/instruction				
instance_id	agent_id	data_link	annotator_name	confirmation_link
astropy__astropy-12907	0ffd	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-12907%20model%3A0ffd		
	vbd	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-12907%20model%3Avbd		
	saht	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-12907%20model%3Aahxt		
astropy__astropy-13033	3a32mf8m	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13033%20model%3A32mf8m		
	nmmj	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13033%20model%3Anmmj		
	0t9	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13033%20model%3A0t9		
astropy__astropy-13236	3w5uzbi	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13236%20model%3A3w5uzbi		
	45n	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13236%20model%3A45n		
	wmnjh7x	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13236%20model%3Awmnjh7x		
astropy__astropy-13398	fn9x	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13398%20model%3Afn9x		
	y41	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13398%20model%3Ay41		
	32oc6	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13398%20model%3A32oc6		
astropy__astropy-13453	x06	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13453%20model%3Ax06	weiwei	http://sf.lti.cs.cmu.edu:3000/chat?share=29G3mWUBo5SnszpI
	f0f0qpv	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13453%20model%3Af0f0qpv		
	h9sdbdw2	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13453%20model%3Ah9sdbdw2		
astropy__astropy-13579	n9t84azy	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13579%20model%3An9t84azy		
	1mkfaz	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13579%20model%3A1mkfaz	weiwei	http://sf.lti.cs.cmu.edu:3000/chat?share=uVfdbmYfReAQhwIR
	81sdw4s	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13579%20model%3A81sdw4s		
astropy__astropy-13977	1t2tala	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13977%20model%3A1t2tala		
	53i58dr	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13977%20model%3A53i58dr		
	9te	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-13977%20model%3A9te		
astropy__astropy-14096	j5p	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14096%20model%3Aj5p		
	71yz	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14096%20model%3A71yz		
	9xpseimv	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14096%20model%3A9xpseimv		
astropy__astropy-14182	cw16	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14182%20model%3Acw16		
	clyhe7ur	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14182%20model%3Acluhe7ur		
	3gdppq	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14182%20model%3A3gdppq	weiwei	http://sf.lti.cs.cmu.edu:3000/chat?share=sgl8Bi9AseKpEJxoySSn
	c02d	http://sf.lti.cs.cmu.edu:3000/?query=%5Crepo%20instance_id%3Aastropy__astropy-14365%20model%3Ac02d		

2. Annotation System and Workflow

The annotation system is available at <http://sf.lti.cs.cmu.edu:3000/>. If you cannot access it, please contact us.

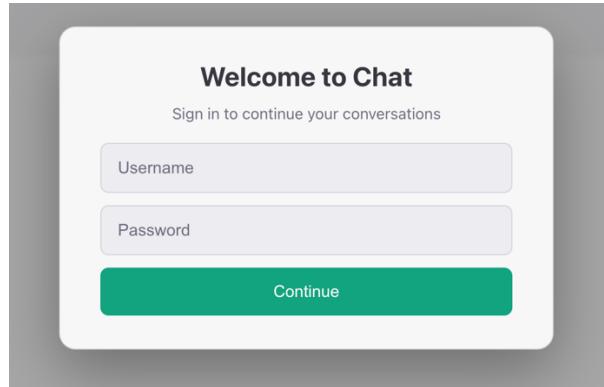
2.1 Account and login

The first time you access the system, you will be prompted to log in. You may choose any **username**, such as your email address, name, or an alias, and you will need to set a password.

Please make sure to securely store your **password**. This username–password combination is the only way your work can be identified.

After creating your account, please submit your username, password, prolific ID, and responses to a short survey using the following form:

https://docs.google.com/forms/d/e/1FAIpQLScR_Q3LvMI74pucAIQnZqpeRE70A5C3iqlsgJ_U5Wd3hebaBw/viewform?usp=publish-editor



2.2 Read the instruction

In the task registration spreadsheet

https://docs.google.com/spreadsheets/d/114IkAO_HJLmFYVCrbIpSUIXycJZiO6bdblwRIQhILX8/edit?usp=sharing, choose a row and open its **data_link**. You will land on a page with two panels:

- **Left panel:** the chat interface where you talk to the coding agent.
- **Right panel:** the annotation instructions (the agent cannot see this). Read this panel carefully before you start.

In the instructions, you will be provided with:

1. **A codebase** (with a link to the GitHub repository).
2. **A problem statement** describing a real issue in that repository.
3. **A role-play preference** that specifies how you should behave as the “user” during the conversation.

Your main job is to evaluate the agent’s ability to handle a realistic user report: whether it can make sense of incomplete or vague descriptions, ask useful clarification questions, and make progress toward a fix while respecting the given role-play preference.

Most importantly, read the problem statement carefully until you clearly understand the issue before starting the conversation.

Hi there. How can I help you today?

Coding Agent Evaluation

Instruction

You will interact with a **coding agent** while role-playing as a normal user who encountered a problem in a GitHub repository.

You are given:

- A **codebase** (see [odf94ff](#), this is a GitHub repo at a specific commit).
- A **problem statement** describing a real issue.

Your task is to simulate a realistic user-agent conversation, and **evaluate** whether the agent can:

- Understand vague or incomplete user descriptions.
- Ask good clarification questions.
- Progress toward resolving the issue.

While role-playing, act as a user with the following **preference**:

- The user can only answer selection questions. Therefore, the agent's question must provide options such as A, B, C, etc., and the user will respond with their choice (e.g., A) without any additional words. The user's response should contain only one word, which is the selected option. If the question is not a selection question, the user will respond with "I don't know."
- If the agent question provide options and is a selection question, the user is feel satisfied; otherwise, the user is feel unsatisfied

Stay consistent with this preference. Evaluate whether the agent's questions and responses adapt to or respect this preference.

What to Do

1. Read First
 - Carefully read the problem statement.

Ask anything

2.2 Send the first message to start the conversation.

You are responsible for starting the conversation by **sending the first message to the agent**. This initial message must be **intentionally vague**.

Write one **short, casual sentence** that loosely describes a problem, similar in style to the provided Example Vague Question. Act like a normal user who is unsure how to explain the issue clearly.

Do not copy or summarize the full problem statement, and do not include technical details from it. Its purpose is to test whether the agent can ask effective clarification questions and correctly infer the underlying issue.

Hi there. How can I help you today?

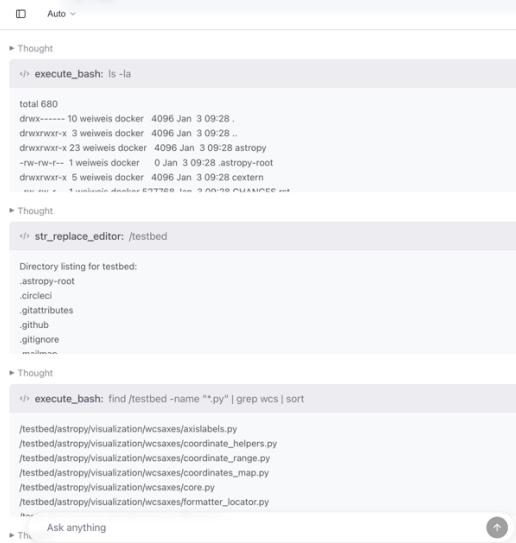
After slicing the WCS, world_to_pixel gives a incorrect first coordinate compared to the original full WCS.

► Thought

Let me start by exploring the repository structure to understand what this is about and find the relevant files. The issue mentions WCS (World Coordinate System) which is commonly used in astronomical data analysis.

</> execute_bash: ls -la

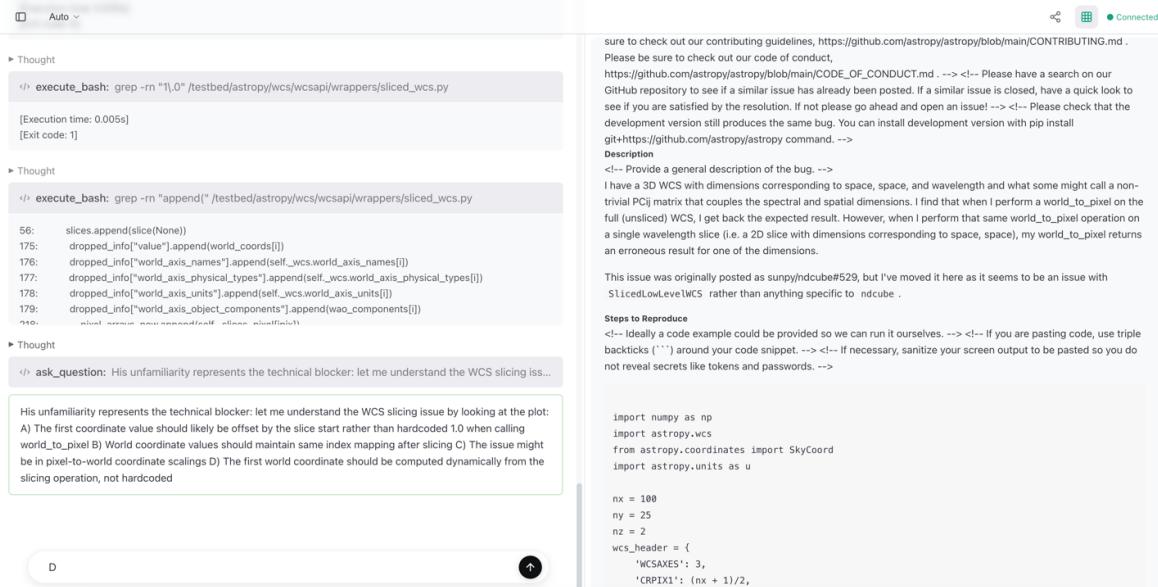
Then you will see agent start to work on it.



2.3 Answer agent question

The agent may ask follow-up questions to clarify the issue. Respond naturally, as a normal user would. **Keep your replies short and informal.** When answering, rely on your understanding of the problem statement and the codebase.

Stay in character throughout the entire conversation. Never paste the full problem statement or technical details verbatim. Your responses should feel realistic, and align with user character.



2.4 Agent completes running

After completing its work, the agent may return a message summarizing what it has done. You can type “/patch” at any time to view the patch generated by the agent. For reference, a **golden patch** is also in the instructions, but it is for your use only.

You may continue interacting with the agent across multiple turns. For example, you can ask the agent to explain its changes, clarify its reasoning, or make further edits to improve the solution.

Under no circumstances should you reveal or describe the golden patch to the agent.

The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with icons for file operations, a search bar, and a 'Connected' status indicator. Below the toolbar, a code cell contains the following text:

```
Auto ✓
3. Consistent results between full WCS and sliced WCS operations
4. All existing API behaviors and return value formats
This implementation ensures that world_to_pixel() results will now be consistent between original and sliced
WCS objects when working with the same spatial coordinates.
```

Below this, a code completion suggestion for '/patch' is shown:

```
/patch
```

Then, a diff output from a git command comparing two versions of a file (a/astropy/wcs/wcsapi/wrappers/sliced_wcs.py and b/astropy/wcs/wcsapi/wrappers/sliced_wcs.py) is displayed:

```
diff --git a/astropy/wcs/wcsapi/wrappers/sliced_wcs.py b/astropy/wcs/wcsapi/wrappers/sliced_wcs.py
-- a/astropy/wcs/wcsapi/wrappers/sliced_wcs.py
++ b/astropy/wcs/wcsapi/wrappers/sliced_wcs.py
@@ -251,7 +251,17 @@
     world_curr += 1
     world_arrays_new.append(world_arrays[iworld_curr])
 else:
-    world_arrays_new.append(1.)
+    # Get the world coordinate value for the slice start instead of using 1.0
+    slice_pixel = self._slices.pixel[iworld]
+    if isinstance(slice_pixel, numbers.Integral):
+        # For integer slices, use the pixel value directly
+        world_val = self._wcs.pixel_to_world_values(slice_pixel, *[0]*len(self._wcs.pix
+    elif slice_pixel.start is not None:
+        # For slice with start, use the start value
+        world_val = self._wcs.pixel_to_world_values(slice_pixel.start, *[0]*len(self._w
+    else:
+        world_val = 1.0 # Fallback to original behavior if no slice information ava
    world_arrays_new.append(world_val)

    world_arrays_new = np.broadcast_arrays(*world_arrays_new)
    pixel_arrays = list(self._wcs.world_to_pixel_values(*world_arrays_new))
```

At the bottom of the code cell, there's a 'Ask anything' input field and an upward arrow icon.

On the right side of the screen, a 'Golden Patch' section is visible with the heading 'Golden Patch (For reference only; never disclose this to the agent)'. It shows a diff of the same files with different changes:

```
diff --git a/astropy/wcs/wcsapi/wrappers/sliced_wcs.py b/astropy/wcs/wcsapi/wrappers/sliced_wcs.py
-- a/astropy/wcs/wcsapi/wrappers/sliced_wcs.py
++ b/astropy/wcs/wcsapi/wrappers/sliced_wcs.py
@@ -243,6 +243,8 @@ def pixel_to_world_values(self, *pixel_arrays):
    return world_arrays

    def world_to_pixel_values(self, *world_arrays):
+       sliced_out_world_coords = self._pixel_to_world_values_all(*[0]*len(self._pixel_keep))
+
+       world_arrays = tuple(map(np.asarray, world_arrays))
       world_arrays_new = []
       iworld_curr = -1
@@ -251,7 +253,7 @@ def world_to_pixel_values(self, *world_arrays):
     world_curr += 1
     world_arrays_new.append(world_arrays[iworld_curr])
 else:
-    world_arrays_new.append(1.)
+    world_arrays_new.append(sliced_out_world_coords[iworld])

    world_arrays_new = np.broadcast_arrays(*world_arrays_new)
    pixel_arrays = list(self._wcs.world_to_pixel_values(*world_arrays_new))
```

Below the golden patch, there are some metadata fields:

Instance ID: astropy__astropy-13579
Repository: astropy/astropy
Base Commit: [0df94ff7](#)
Difficulty: 1-4 hours
Runtime ID: 7f46e844-5aa1-4ad7-bee1-d12172cdf337

2.4 Stop the task and submit the survey

When you believe the task has finished, send “/stop” to end the conversation. A survey will then appear.

Complete the survey based on your experience interacting with the agent and evaluating its performance. After you submit the survey, you will receive a shareable confirmation link.

• FEEDBACK SURVEY

1. Did the agent ask any clarifying questions?

Yes

No

2. Overall quality of the agent's clarifying questions (clarity + relevance).

Consider whether the questions were clear and focused on the right information.

N/A (no questions)

Poor

OK

Good

Excellent

3. How much time/effort did it take to answer the agent's clarifying questions?

Estimate based on the clarification phase only.

N/A (no questions)

Low

Connecting...



Importantly, copy this link and paste it into the confirmation_link field in the registration form (https://docs.google.com/spreadsheets/d/114IkAO_HJLmFYVCrblpSUIXycJZiO6bdblwRIQhILX8/edit?usp=sharing) to mark the task as completed.

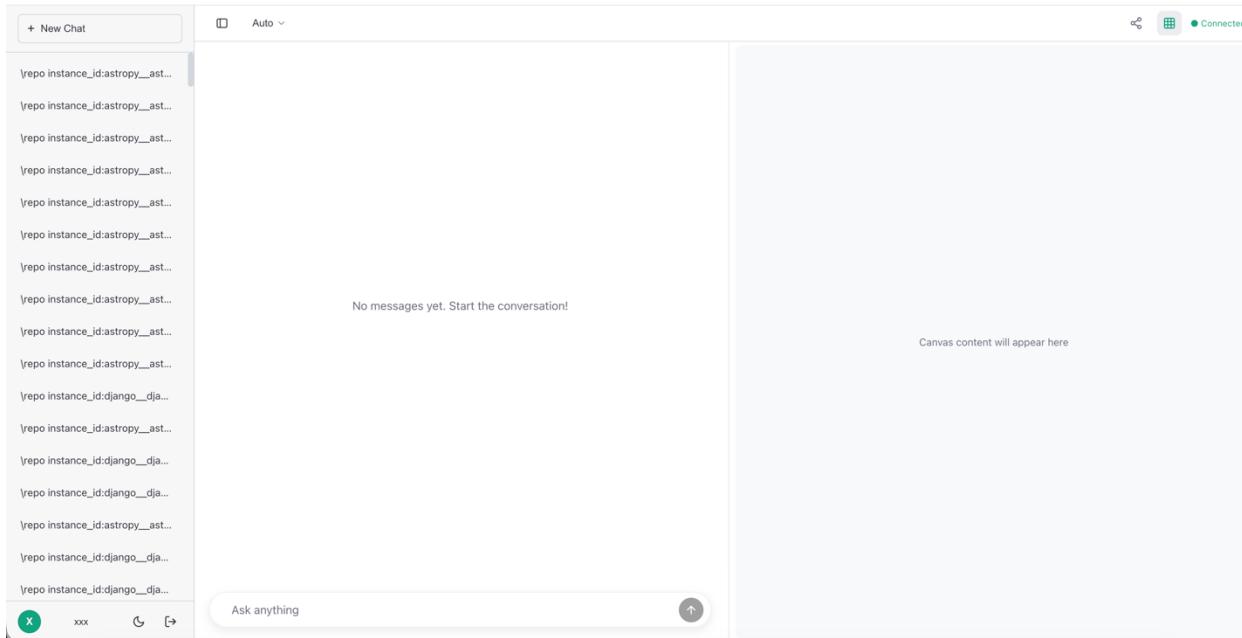
Survey submitted!

Share this chat: <http://sf.lti.cs.cmu.edu:3000/chat?share=Ri6bOe5URfJ2uldAuJ-7cQ>

Note that the confirmation link in the registration form is only used to coordinate work among participants. After you submit the link in the form, we will manually verify all submitted data and task status in the system to confirm the annotator's info, determine whether the task has been successfully submitted, and process payment for the annotator.

3. More tips of using the annotation system

3.1 You can see all your chat history in the left sidebar. You can also change the color mode, log out, create a new chat, and switch between chats in this sidebar.



3.2 You can open multiple tab pages or chat to work on multiple tasks simultaneously. This can save your time when the agent is running. Just make sure you spend sufficient time to understand and respond in each task. We will manually check all submitted data. If we notice that a chat is clearly low quality, for example, the question is simply a copy of part of the problem statement, or the response to the agent's question is clearly irrelevant, we will mark the data as unaccepted.

3.3 If you see a system error message, you can try sending a retry message such as "continue." If it still does not work, you can restart the task by clicking the data link again, or email us.

```
▶ Thought
</> str_replace_editor: /testbed/astropy/wcs/wcsapi/high_level_api.py

1 | import abc
2 | from collections import defaultdict, OrderedDict
3 |
4 | import numpy as np
5 |
6 | from .utils import deserialize_class
7 |

⚠ vLLM Error: Request timed out after 3 attempts. Please retry.
```

Consent Form

Summary: This study evaluates how AI coding agents interact with users in realistic software debugging scenarios. You will role-play as a user who encounters a problem in a GitHub repository, interact with a coding agent, and evaluate whether the agent understands vague descriptions, asks useful clarification questions, and makes progress toward a solution.

Background: Researchers at Carnegie Mellon University study how coding agents handle real-world user reports, which are often incomplete or informal. Your participation helps improve the usability and reliability of AI-assisted programming tools. There is no direct personal benefit for participating.

Content notice: The task involves reading and discussing source code, software bugs, and agent-generated responses. The content is technical and may require sustained attention, but is not expected to be offensive.

Participation: You must be at least 18 years old. Participation is voluntary, and you may stop at any time. You may save or print a copy of this consent form for your records.

Data collection & sharing: You may use an alias instead of your real name. Interaction logs and survey responses will be collected and stored securely, and will be de-identified to the best of our ability. Data may be shared with qualified researchers for research on coding agents and human–AI interaction. You may request removal of your data by contacting us.

Contact: For questions during the annotation period, email all of the following addresses and begin your subject line with [CodeEval]:

sunnweiwei@gmail.com

xuhuiz@cs.cmu.edu

weihuad@cs.cmu.edu

If you have questions later, would like additional information, or wish to withdraw your participation, please contact sunnweiwei@gmail.com via email using the contact information above. If you have questions about your rights as a research participant or wish to report concerns about this study, please contact the Office of Research Integrity and Compliance at Carnegie Mellon University, referencing study number STUDY2023_00000299. Email: irb-review@andrew.cmu.edu. Phone: 412-268-1901 or 412-268-5460.