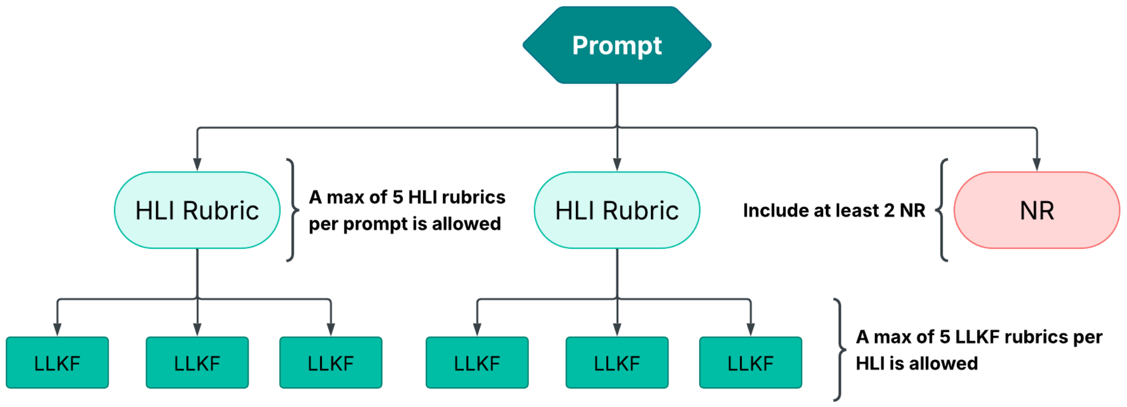
- **High-Level Intent (HLI):** This is the primary requirement. These rubrics address the user's intention on a broad scale.
- **Low-Level Key Facts (LLKF):** These are the specific, individual facts needed to fulfill the HLI.
- **Negative Rubric (NR):** Defines what the response is explicitly forbidden from doing.


Prompt
HLI Rubric
A max of 5 HLI rubrics per prompt is allowed
HLI Rubric
Include at least 2 NR
NR
LLKF
LLKF
LLKF
LLKF
LLKF
LLKF
A max of 5 LLKF rubrics per HLI is allowed

For a more detailed explanation of each type, go to HLI, LLKF, NR definitions

Formatting Instructions:

- Number each HLI with a whole number (e.g., 1, 2, 3).
- Number each supporting LLKF with a decimal that links it to the parent HLI (e.g., 1.1, 1.2, 1.3).
- Negative Rubric must begin with a capital letter (e.g., A, B, C).

📖Example of formatting:

Prompt: *Can you name only the first 3 Harry Potter books from the original series?*

Rubric:

- 1: The response must list the first three original Harry Potter books. **[HLI]**
- 1.1: The response must list 'Harry Potter and the Sorcerer's Stone' as the first book. **[LLKF]**
- 1.2: The response must list 'Harry Potter and the Chamber of Secrets' as the second book. **[LLKF]**
- 1.3: The response must list 'Harry Potter and the Prisoner of Azkaban' as the third book. **[LLKF]**
- A: The response must not list the 4th, 5th, 6th and 7th original Harry Potter books **[NR]**
- B: The response must not list any book that doesn't belong to the original Harry Potter books, for example Fantastic Beasts and Where to Find Them **[NR]**

NOTE: The examples shown here are only for teaching purposes; it's forbidden to copy these examples and use them in the tasks.

Rules for the Overall Rubric

A complete rubric **MUST BE**:

- **Comprehensive:** Includes all necessary criteria to define an ideal response, with no missing elements.
- **Relevant:** Contains no unnecessary or unrelated criteria.
- **Accurate:** Composed of objective and factually correct criteria.
- **Non-Repeating:** Avoids redundancy; the same mistake should not be penalized more than once.
- **Self-Contained:** Includes sufficient detail and examples to be easily understood without outside context.
- **Reflective of the Prompt:** Accounts for every explicit request made in the original prompt.
- **Include at most 5 HLI. The minimum is 2.**
    - **Each HLI should have at least one LLKF.**
- **Include at most 5 LLKFs per HLI. The minimum is 1.**
- **Include at least 2 NR.**

Example Rubrics

Rubric's best practices

- Write each criterion as a checklist item using the following structure:
    - *The response **must** [requirement]* or *The response **must not** [requirement]*
    - *The documentation **must** [requirement]* or *The documentation **must not** [requirement]*
- Always use this formulation with "must," **not** "should," "can," etc., like in some other projects. Example:
    - ✅*The response **must** ensure the CSV file is not empty before trying to access its data points.*
    - ⛔*The response should provide edge cases to show how the 'is_prime_number' function handles them.*
- Be specific. Each criterion should remove all reasonable ambiguity about what counts and what does not.
    - ⛔ Avoid using vague language like "mention", "suggest", "“discuss", or undefined jargon.
    - ⛔ Don't use subjective terms (like "good", "comprehensive", "general", "correct", "clean")
- If the criteria is about including examples: Say exactly how many examples are needed and whether other examples are allowed.
    - Avoid listing examples with "such as", "like", or single-item examples.
- Rubrics that deal with code elements, should mention the code element and the file, not the line numbers of code.
    - ✅ The response must reference X method in X file.
    - ⛔The response must reference lines 200-300 of X file.

- Don't be too prescriptive in code implementation rubrics: We cannot penalize valid code solutions just because our rubrics included a different naming for new code elements. Read this Common Errors to avoid section.

Step 5: Creating a golden response

- By this step you should have identified the main problems with the model's response.
- Amend the response so that it correctly answers the prompt. This can be as simple as changing a line of code in a file stored within the repo so that it runs, but it can also be more complex such as adding several files to the repo.
  - Note that you will want to make these changes locally. The changes you make to the repo will be stored using the git diff terminal command. This info will then be uploaded to the task under the file name "golden.patch"
- The end goal for the golden response is the same for every task: an answer that provides a correct solution to the prompt.
  - Once you have made the necessary changes to the local repo, use the following terminal command to create a report showing all the changes made to the repo, and save it to a file named "golden.patch"
    - `git diff > golden.patch`
- **Upload the golden.patch file to the task.**

Appendix

Common Errors to Avoid in Rubrics

CRITICAL 🔴- Having this issue will automatically lower your task score to 2
IMPORTANT 🟡- Having this issue will lower your task score with the risk of getting a 2

Status	**Common Error**	**Example**
CRITICAL 🔴	**Missing essential criteria** **Explanation:** When writing a rubric, all essential criteria to answer the prompt should be included. Missing essential criteria will automatically make	**Prompt**: *Give me 2 different Python scripts I can use to convert CSV to JSON* ⛔Bad Rubric (missing to include a criterion that evaluates whether the model includes 2 Python scripts): - The Python script must be designed to convert a CSV document into a JSON . - The response must state that the script needs to include csv.DictReader() to convert the first row to keys.

Status	Common Error	Example
	the task obtain a low quality score.	- The response must explain that each row needs to be converted into a dictionary. - The response must explain that all rows are stored in a list and then dumped to JSON using json.dump(). - The response must explain that including indent=4 makes the output human-readable. ✅Good Rubric: - The Python script must be designed to convert a CSV document into a JSON. - The response must explain that the script needs to include csv.DictReader() to convert the first row to keys. - The response must explain that each row needs to be converted into a dictionary. - The response must explain that all rows are stored in a list and then dumped to JSON using json.dump(). - The response must explain that including indent=4 makes the output human-readable. - The response must provide 2 different Python scripts.
CRITICAL 🔴	**Repetitive criteria** **Explanation:** When writing a rubric, each criterion should address a part to evaluate; therefore, repeating criteria is not allowed, as it will lead to over-penalization of the model.	**Prompt**: *Give me 2 different Python scripts I can use to convert CSV to JSON* ⛔Bad Rubric (missing to include a criterion that evaluates whether the model includes 2 Python scripts): - The Python script must be designed to convert a CSV document into a JSON. - The response must explain that the script needs to include csv.DictReader() to convert the first row to keys. - The response must explain that each row needs to be converted into a dictionary. - The response must explain that all rows are stored in a list and then dumped to JSON using json.dump(). - The response must explain that including indent=4 makes the output human-readable. - The response must provide 2 different Python scripts - The response must explain that the script needs to include csv.DictReader to convert the first row to keys **Criteria 2 and 7 are almost identical, and we must only leave one.**
CRITICAL 🔴	**Not Self-contained (Closed questions)**	**Prompt**: *What's the capital of France?* ⛔**Bad Criterion**: The response must include the capital of France.

Status	Common Error	Example
	Explanation: When you have a prompt with a closed question, the criteria should include the answer.	✅**Good Criterion**: The response must mention that the capital of France is Paris. --- **Prompt**: *I'm building a VR experience in Unreal Engine using C++ and Blueprints. I want to trigger dynamic haptics and spatial audio when the player interacts with certain world objects. I know Unreal supports force feedback and audio attenuation, but need C++-level details. Please search Unreal Engine's C++ API or headers for:* ● *Classes/methods for controller vibration.* ● *Components/properties for spatial sound.* *Provide:* ● *Full class names and method signatures.* ● *Their modules or header files.* ● *A C++ snippet showing how to trigger both haptics and spatial audio on trigger volume overlap.* ● *Any required modules/dependencies for compilation.* ⛔**Bad Criterion**: The response must provide the names of the C++ components and properties involved in spatial sound propagation. ✅**Good Criterion**: The response must provide the names of the C++ components and properties involved in spatial sound propagation: UAudioComponent, USoundAttenuation, USoundCue, USoundBase.
CRITICAL 🔴	**Missing examples (open-ended questions)** **Explanation:** Rubric criteria should be clear and objective, with specific examples that show what is expected.	**Prompt**: *Recommend some shoes for running long distances like half marathon.* ⛔Bad Criterion: The response includes running shoe models for running long distances. ✅Good Criterion: The response must include running shoe models for running long distances such as the Nike Alphafly 3, Saucony Endorphin Pro 4, Hoka Cielo X1, etc. ---

Status	Common Error	Example
		Prompt: I'm building a VR experience in Unreal Engine using C++ and Blueprints. I want to trigger dynamic haptics and spatial audio when the player interacts with certain world objects. I know Unreal supports force feedback and audio attenuation, but need C++-level details. Please search Unreal Engine's C++ API or headers for: Classes/methods for controller vibration. Components/properties for spatial sound. Provide: Full class names and method signatures. Their modules or header files. A C++ snippet showing how to trigger both haptics and spatial audio on trigger volume overlap. Any required modules/dependencies for compilation. ⛔ **Bad Criterion**: The response must explain any required dependencies or engine modules needed in the C++ build file for successful compilation. ✅ **Good Criterion**: The response must explain any required dependencies or engine modules needed in the C++ build file for successful compilation, such as Core, CoreUObject, Engine, InputCore, etc.
IMPORTANT 🟡	**Subjective criteria** **Explanation:** Criteria should be objective (True/False). Avoid criteria based on personal opinions.	⛔**Bad Criterion:** The response must state that the best coding language is C++. ✅**Good Criterion:** The response must state that C++ is the best option to design a financial system because of its ability to handle large datasets and complex calculations efficiently.
IMPORTANT 🟡	**Overly prescriptive** **Explanation:** Don't include criteria based on your own likes or dislikes. Focus on what's essential for a good response and what is stated in the prompt.	**Prompt**: *Give me some ideas of Python scripts I can use to convert CSV to JSON* ⛔**Bad Rubric:** - The Python script must be designed to convert a CSV document into a JSON. - The response must explain that the script needs to include csv.DictReader() to convert the first row to keys. - The response must explain that each row needs to be converted into a dictionary. - The response must explain that all rows are stored in a list and then dumped to JSON using json.dump().

Status	Common Error	Example
		- The response must explain that including indent=4 makes the output human-readable. - The response must provide exactly 3 different Python scripts.
IMPORTANT 🟡	**Unnecessary style/formatting** **Explanation:** Don't specify style or formatting unless the prompt requires it.	**Prompt**: *Give me 2 different Python scripts I can use to convert CSV to JSON* ⛔Bad Rubric: - The Python script must be designed to convert a CSV document into a JSON. - The response must explain that the script needs to include csv.DictReader() to convert the first row to keys - The response must explain that each row needs to be converted into a dictionary. - The response must explain that all rows are stored in a list and then dumped to JSON using json.dump(). - The response must explain that including indent=4 makes the output human-readable. - The response must provide 2 different Python scripts - The response must be formatted in a table for easier reading. **Explanation of why these are bad criteria:** The prompt is not requesting any specific formatting.
IMPORTANT 🟡	**Poor Grammar and Spelling in the Criteria** **Explanation:** The criteria in the rubric should be written with perfect spelling. Starting each criterion with a capital letter, adding accents, and punctuation marks, and correcting misspelled words**.**	⛔Bad grammar: the answer must inclde the 4 suits of a poker deck: hearts, spades, diamonds and clubs. ✅Correction: The answer must include the four suits of a poker deck: hearts, spades, diamonds, and clubs.

Words to Avoid in Rubrics 🚓🚨

Rubrics should be written with direct and objective language. This means we must avoid adjectives, adverbs or verbs that are vague or introduce subjectivity in the rubrics.

Principle to follow	Key Questions to Check in your Rubric	Instant Red Flags
Be Specific	Does the wording remove all reasonable ambiguity about what counts and what does not?	**Vague verbs**: “mention”, “discuss”, “suggest”. Undefined jargon.
Be Consistent	Would all reviewers reach the same decision with no extra context?	**Subjective adjectives**: “good”, “comprehensive”, “general”, “well”, “clean”, “effective”, “proper”, “robust”, “optimal”, “efficient”, “maintainable”, “scalable”, “best”, “high-quality”, “performant”. **Subjective adverbs**: “clearly”, “correctly”, “efficiently”, etc.
Use Must / Must Not	Is every requirement or prohibition expressed with “must” or “must not”?	**Vague modal verbs**: “should”, “may”, “could”, “shall”.
State Clear Examples	Does the rule say exactly how many examples are needed and whether other examples are allowed?	“such as”, “like”, single-item examples

HLI, LLKF, NR definitions

Category	Label	Definition
Rubric Purpose	**High-level Intent understanding Rubrics + Presentation (HLI)**	These rubrics address the user's **intention** on a broad scale. These intention rubrics can only be fulfilled with **blocks of texts** in the response (e.g. paragraphs, lists, tables). In general, each prompt will contain no more than **five** high-level intents that rubrics can check. These rubrics should not be focused on details. They must focus on a broader aspect of the response that will then guide the creation of low-level fact-based rubrics.

	Low-level key facts rubrics + Grounding (LLKF)	These rubrics define **critical, exact** and **verifiable facts** that collectively fulfill users' HLI. A response that omits this fact would fail to fulfill the user's intent. The fact should have a single clear meaning with no other interpretation(s) and must avoid subjective terms (e.g., "good" or "thorough"). For these rubrics, you should ground the reasoning to the codebase you are working whenever applicable.
	Negative Rubrics (NR)	These rubrics offer targeted, precise prohibitions to **exclude specific unwanted elements** in the responses, allowing for very specific details to **highlight exact pitfalls or errors to avoid**.
Rubric Importance	**Must Have**	These rubrics check for critical components for effective responses that match the user's main intent. Without these aspects, the model will appear unintelligent or incapable.
	Nice to Have	These rubrics check for additional details that improve the response but are less critical. The response will still be correct without them.
Rubric Aspects	**Instruction Following / Intention Understanding**	These rubrics check if the response is following the prompt's requests and intent
	Context Understanding	These rubrics check if the model understands the context (repository, files, previous question and answer in the conversation) that is needed to craft an answer.
	Content Accuracy	These rubrics check if the response content is accurate and correct.
	Grounding	These rubrics check if the response's reasoning is grounded in the codebase.
	Presentation	These rubrics check that the model presents the information in a clear and effective manner [See Presentation failure labels]
	Tool Call	These rubrics check that the model is using the relevant agent tools to reach the user's intent and instructions.

Example Rubrics Structure