Points to consider:

- Frame user prompts as **real-world problems** instead of synthetic or overly simplified scenarios.
- Avoid using **generic names** like *Team A, Train XYZ, Company A*; use more realistic identifiers instead.
- Ensure that the **user query has no ambiguity leading to incorrect model responses**.
- Create a **balance of technical + non-technical tasks**.
- Add an **SI mentioning the use of markdown** in the assistant responses.
- If you are using any symbols like △ → ≅ ∠ ⟹ ≈ ∩ ∪ ∫ θ, **please add an SI explaining when to use it**. Also, you can add an example.
- If the assistant is making any **assumptions**, then it should be explicitly mentioned in the assistant response.
- In **technical tasks**, If the assistant provides the code, ensure it's **optimal, bug-free, and follows best practices**.
- Avoid **markdown** in the system prompt.
- In **mathematical tasks**, the user prompt should only have an **objective question**. Avoid asking for **suggestions or recommendations** and ensure the user query has **only one question**.
- The **assistant response should be clear in the explanation provided**.
- Before submitting tasks, please review the **SI again** to ensure you **follow all instructions**.
- Create the task while **considering the rating criteria to maximize ratings**.
- The **first turn** needs to be **complex**; the rest of the conversation can be an extension of it (**not too complex, but it should add some value**).
- **Fixing reworks is important** from a trainer's perspective because the **reworking task contains a low rating**, affecting the **trainer's overall rating**; Once we fix the issue, we will provide **new ratings**, which will increase your **overall average ratings**.
- **Bluff in user prompts has become overused, so avoid adding them to every query**.
- **Don't mention the taxonomy in the System Instructions**, e.g., "*You are an AI assistant specializing in complex reasoning within the policy optimization domain, focusing on causal reasoning.*" but "*You are an AI assistant with extensive knowledge of greedy algorithms. Your task is to assist the users in coming up with optimal solutions to user-specified problems through applying greedy algorithms.*" For further clarification, **causal reasoning** and the **metadata mentioned in the tasks**, i.e., taxonomies, use cases, and L2 taxonomies, are **ways that we use to assess the tasks and guide the direction of scenarios**. They **shouldn't be mentioned explicitly anywhere in the task**. Focus more on **solving real-life cases and scenarios rather than optimizing for the metadata**.

- **Enough context and constraints in user prompts**.
- **Assistant responses shouldn't contain low reasoning**.
- The assistant should provide **steps in reasoning before providing the code**.
- **Code explanations, code meaning, ip/op, and complexities**.

Tips:

- Write the prompt in **first person**.
- Ask **questions that a typical SDE will ask**.
- Be **natural** - as human as possible.
- Provide **some context, but not too much**.
- **Grammar mistakes are ok!** The model is supposed to understand it. (**as long it's not hampering the meaning**).
- Provide **as many details as possible** (**code errors, etc.**).
- **No synthetic signatures/AI-generated prompt**.
- **Don't skip out on complexity**.
- **If there are multiple errors, point out the first error that occurs. Ask to Fix all the errors.** The **high chances model will fix only the provided error**.
- **Having multiple logical errors but pointing out only one**.
- **Codes/template AI generated - remove comments phrases that look AI-generated, like placeholders**.
- **You can use code that does not use OOPs to look natural**.
- **You can take inspiration from GitHub issues and use unfamous libraries**.