✅ Submission Checklist

- ☐ **Review your task with a Review Checklist before submitting. (NEW) (IMP)**
- ☐ **Minimum 6k characters in SM (**We are reducing the limit on system prompt length, focusing on **complexity** instead of length.)**,** you can use any online counter like **(**https://www.charactercountonline.com/ **) (NEW)**
- ☐ **Feasible Tool Use Task:** Ensure the overall task demonstrates valid tool usage and showcases assistant reasoning over tool outputs.
  - ☐ For **Search Refinement** even after multiple trials, if we are not getting any output, then this is fine.
- ☐ **Natural User Prompts:** Prompts must be casual and naturally phrased (not robotic or overly minimal) and should implicitly require multi-step tool reasoning.
- ☐ **Datetime Reasoning (When Applicable):** If datetime-related tools are used, include prompts requiring relative/fuzzy time grounding, unit conversion, or complex reasoning.

-  Default Clarification Behavior Followed
-  Context Information in System Prompt
-  High-Level Instructions **(conditional + example-driven SM) (NEW) (IMP)**
-  **Respected Sub-Categories:**
  -  **Normal**
  -  Search Refinement
  -  Prompt Injection
-  **Three Model Failures** ("selected_overall": false): With proper error_label, Critic Comment (3rd person), Reasoning Comment (1st person, assistant-style, thought bubble)

-  **At Least 3 User Prompts That Trigger Tool Chains**: Make sure minimum 3 user prompts lead to sequential tool calls (chain of reasoning, not single call).
-  Include explicit **contextual information in the system prompt** that the model can use immediately, without searching or calling tools (e.g., current datetime, database states, info on screen, etc). Check here
-  **Trigger fallback tools when needed** — If a user asks for something like celebrity earnings or recent events, and a fallback tool like web_search is allowed, the assistant must attempt to retrieve it using that tool.
-  **Turn Count Between 10–15**: The conversation must include 10 to 15 complete turns.
-  Not including default preferred values for tool arguments, rather focusing on more **complex behavioral rules or nuanced reasoning patterns** instead.
-  Instead of reinforcing what the assistant already does by default, teach **new reasoning paths**, conditional dependencies, or exceptions that **modify assistant behavior**.

-  **Avoid duplicate or repeated prompts** — Ensure no two user turns use the same phrasing unless repetition is contextually justified.
-  **Maintain logical flow across turns** — User prompts must build upon previous context; avoid stitching unrelated queries without narrative continuity and logical conversation flow.
-  **Use diverse tools in multi-tool calls (Sequential calls)** — When using multiple tools in a single turn, they must serve different functions or domains not just from the settings domain.
-  **Don’t clarify when user intent is already specific** — If the user gives a clear time or condition (e.g., “Saturday at noon”), avoid asking vague follow-ups like “Which time frame should I check?”
-  **Reuse recent tool data when possible** — If a tool has just returned a value (like location/address), reuse that result rather than calling another tool to get the same information again.
-  **Never hallucinate user visibility of internal values** — Do not act on requests to convert internal system

values like place_id, Think like a user, not like a QA engineer triggering tools .All user prompts must be what a real person would say.

- ☐ **Use human-centric phrasing in all assistant prompts** — Replace machine-facing language like "simulate real-time conditions" with natural alternatives such as "estimate travel based on typical traffic patterns."
- ☐ **Always validate time-sensitive references with current date** — For queries about “this month” or “recent releases,” first fetch the current date/time using the appropriate tool, and use it to anchor tool queries logically.

✅ Please verify all these points manually before submitting.

📌 Any deviation from these expectations may lead to rejection or rework.

Review Checklist

System Message

1. **Minimum 6k characters in SM,** you can use any online counter like **(**https://www.charactercountonline.com/ **)** We are reducing the limit on system prompt length, focusing on **complexity** instead of length. **(NEW)**
2. Context Information in System Prompt

- ☐ Should contain technical background details of the user, which should be relevant as per the conversation. It should be used and referred to in at least one turn in the task.

-  Can be in any format : Json, Paragraph, Bullet, Csv, yaml.
-  Should not be very short (3-4 lines) and not very long (>50 lines)
-  Include minimum two valid contexts from: **(NEW)**

1. location
2. time
3. device_settings
4. user_identity
5. app_state

3 **High-Level Instructions (conditional + example-driven SM) (NEW) (IMP)**

-  Can be written in paragraph form or bullet points MUST HAVES (4 things):
- Conditional/Fallback instruction (if-then) including edge case handling.
- Examples
- Example driven system prompts with placeholder values. things like **(NEW)**

```
You should talk to the user in the following way

User: Hi
Agent: Hi [User Name] how can I assist you today?

---
```

```
You should first use tool A, then feed the value for key C
into tool B. e.g.

A() -> {"C": <value>}
B(<value>)
```

- What to do
- What not to do
-  There must be many instructions in conditional dependencies (if this then that) format including edge case handling. **(NEW)**
-  There must be a few instructions in "What not to do" format.
-  Reflect at least one conditional fallback instruction for edge cases in your conversation as well. (**Guiding assistant behavior in edge cases)**
-  Examples should be followed by the respective instructions. Do not write instructions in one section and all the examples in another section. They should be written in the same flow, pairing each instruction with its example.

4 Other requirements

-  Every system prompt must include at least one

clear, diverse agent persona. **(NEW)**

- ☐ Add at least two valid tool invocation policies such as: **(NEW)**

1. confirm_before_stateful_change
2. always_check_settings_before_tool_use
3. avoid_redundant_setting_checks
4. conditional_confirmation
5. no_tool_for_knowledge_questions
6. prefer_tool_X_over_Y
7. prefer_specific_tools_over_web_search
8. cross_reference_tools
9. conditional_parallel_calls

- ☐ Do not add repeated or verbose sections that don’t add value, remove redundancy and improve readability. **(NEW)**
- ☐ Ensure consistency between rules and examples — no contradictions. **(NEW)**
- ☐ Refine instructions for clarity and ensure each directive is actionable. **(NEW)**
- ☐ Do not mention the model's default/expected behaviour as instructions since the model is already aware about it.

- ☐ Do not mention tool definitions as the model already has that information.
- ☐ At least 30-40 % variations in SM.
- ☐ There is no tight coupling (we should not use exact examples used in SM in user prompt).

User prompt

- ☐ **Natural User Prompts:** Prompts must be casual and naturally phrased (not robotic or overly minimal) and should implicitly require multi-step tool reasoning and doesn't include and **internal details**
- ☐ Ensure the request is **feasible with available tools** (don't ask for unsupported actions).
- ☐ Prompts should allow for **justified tool usage** — tool calls must be necessary, logically connected to the request, and it should be clear from context that the assistant should use a tool (without the user explicitly mentioning the tool)

-  Keep prompts **human-like and varied** (mix questions, statements, and commands).
-  Avoid **unnecessary backstory or artificial details** (e.g., coordinates, IDs).
-  Encourages **multi-step or chain-of-tool use** through realistic, complex requests.
-  Handle **datetime naturally** (e.g., “tomorrow at brunch time,” “last Friday next month”).
-  Avoid scripted patterns like always starting with *“Can you…”* — vary style.
-  Use **contextual references** naturally (e.g., “there,” “it,” “which one”) instead of repeating details.
-  Avoid ambiguous phrasing; prompts should be **clear and uncontradictory**.

Rejections (Critical as AR is not checking for this)

-  Rejection is valid
-  Correct error_label is used.
-  Correct critic Comment:
  -  Clearly describe what was done incorrectly and why it matters.
  -  Always use a third-person persona (e.g., *“The assistant failed to…”*).
  -  Must be specific and concrete, aligned with the error labels, and avoid vague or generic phrasing.
-  Correct Reasoning Comment:
-  Must begin by **acknowledging the issue**.

-  Clearly describe the **corrective step or solution actually applied at the time of rejection**.
-  Always use a **first-person persona** (e.g., *“I will use search_yelp to fetch the business details”*).

Conversation

-  **Agent Truthfulness — Tool Output Grounding:** Rigorously verify every statement against the exact tool fields: echo IDs/ASINs/SKUs verbatim (no renaming/normalizing); keep statuses/timestamps as returned (no severity inflation like `IN_QUEUE→FATAL`, no invented reasons for missing documents); state the real scope (all-listings vs per-item) and counts/filters/units exactly; don’t fabricate titles/brands—use neutral attribute-based descriptions if names aren’t provided; replace vague “today/tomorrow” with explicit dates from outputs; fix any hallucinations or extrapolations before submission. **(NEW) (IMP)**
-  **Agent Truthfulness — System Prompt Grounding:** Rigorously align the response with SP-declared facts/constraints (dates, sources, personas, allowed actions, calendars); if SP and tools diverge, acknowledge the discrepancy, don’t overwrite SP facts, and propose a corrective next step (retry/refine/clarify); ensure sources/attributions match (e.g., correct subreddit/channel), maintain consistent IDs across turns, avoid

scope/time inconsistencies or contradictions, and correct any inaccuracies or hallucinations before submission.
(NEW) (IMP)

- ☐ **Respected Sub-Categories (**The task sub-category is clearly visible in at least 1 turn)**:**
  - **Normal**
  - **Search Refinement**
  - **Prompt Injection**
- ☐ **Number of turns: 10 - 15**
- ☐ **At least 3 sequential tool calls**
- ☐ **At least 3 model failures**
- ☐ **Validate time-sensitive references with current date**
- ☐ **Trigger fallback tools when needed** - If a user asks for something like celebrity earnings or recent events, and a fallback tool like web_search is allowed, the assistant must attempt to retrieve it using that tool.

Task Categories and Scenarios

When creating a scenario, first determine which category it falls into

and design accordingly:

- **Feasible Tool Use Tasks: The user's request can be satisfied using the available tools.**
  - ***Goal:*** Guide the assistant to complete the task using one or more tools.
  - ***Approach:*** The user should present a query or problem that will require the assistant to use the tools to find information or perform an action. The assistant will then plan a solution and invoke the appropriate functions.
  - In feasible tasks, the conversation should end with the user's request fully resolved.

- [round_7_update] ***Natural User:***
  - ***Goal:*** Simulate a user who is naturally conversational and somewhat informative, unlike the minimalistic Lazy User. Prompts should feel realistic and casual but still lead to multi-step reasoning and tool chaining by the assistant.
  - ***Approach:***
    - Users should speak in full, natural-sounding sentences (not terse or robotic).
    - Each prompt should **imply multiple steps** or require **chained tool use**, even if the user doesn't state all intermediate details.
    - Avoid over-specifying or frontloading all parameters in a single prompt.
    - Ensure at least **3 such user turns** exist within a single conversation trajectory.
  - **Key Design Patterns:**
    - Encourage prompts like:
      - "How many tracks are in *Human After All*?" → Requires:

1 Search album
2 Get album ID
3 Fetch track details

- "Directions to the closest McDonald's." → Requires:
  1 Get current location
  2 Search nearby McDonald's
  3 Get directions to the closest one

These examples show natural prompts that **implicitly expect**

multiple tool calls for fulfillment.

Important Notes:

- This is distinct from Lazy User because the user is **not passive or vague**—they're just realistically casual and brief.
- Do **not** artificially break user intent across multiple turns. Let the **assistant drive the breakdown** via reasoning and tool planning.
- ***Datetime Reasoning (When applicable):***

    - ***Goal:*** Simulate user prompts that involve interpreting and converting natural or complex datetime expressions into tool-compatible formats (e.g., ISO 8601). The focus is on testing the assistant's ability to reason through time-based language and make accurate conversions or clarifications.
    - ***Approach:***
        - Design user prompts that refer to time in human-centric or vague terms (e.g., "next Friday", "in 40 days", "brunch time").
        - Avoid direct timestamps or over-specified datetime formats.
        - Require the assistant to:
            - Interpret relative or fuzzy expressions.
            - Perform unit conversions (e.g., "120s" → 2 minutes).
            - Reason over current time and user-provided offsets.
            - Confirm with the user when the time reference is ambiguous.
    - **Key Design Patterns:**
        - **Unit Conversion:**
          "Mark something on my calendar in 120s." → Assistant should convert to "2 minutes" before using the tool.
        - **Relative Time Calculation:**
          "What's on my calendar for the last Friday of next month?" → Assistant needs to compute the correct date from the current context.
        - **Natural Language Interpretation:**
          "How long is my commute if I leave half past noon

tomorrow?" → Assistant must interpret "half past noon" as 12:30 PM and calculate accordingly.

- **Fuzzy Time Range:**
  "Do I have anything tomorrow at brunch time?" → Assistant may define a default range (e.g., 10:00–12:00) or confirm with the user.

Important Notes:

- Time grounding must be contextually accurate.
- Always assume the user is not familiar with datetime formats.
- If a time reference is ambiguous or has a wide interpretation, ask the user to confirm before proceeding.
- These tasks test the assistant's ability to **translate natural time expressions** into precise tool inputs, a critical component of real-world usability.
- ***Default Clarification Behavior:***

  - ***Goal:*** Ensure that assistant clarifications are user-friendly and focused only on the **required** inputs necessary to proceed. Assistants should avoid surfacing internal tool complexities to the user and instead guide the user with terms they can naturally understand.
  - ***Approach:***
    - By default, assistants should **only ask for required arguments**, unless the **System Message explicitly** says to gather optional fields.
    - Avoid exposing technical implementation details (like internal IDs or system-specific identifiers).
    - If a required argument is not directly available from the user, use **tool chaining** to infer or retrieve it based on natural user input.
  - **Key Design Patterns:**
    - **Avoid Asking for Internal Identifiers Directly:**
      Don't say: "Please provide the `calendar_id`."
      Instead say: "Which calendar would you like to use?" → Then use `search_calendar` to fetch the ID internally.
    - **Focus on What the User Knows:**
      Ask for names, titles, or natural references (e.g.,

"meeting with John" or "Work Calendar").
    - **Limit Clarifications to Required Arguments:** In tools like `create_calendar_event` and many others, avoid asking for all parameters upfront (e.g., location, description, lat/lng) unless they are **strictly required**.

Important Notes:

- Only include clarifications for **required** fields unless the SM **explicitly instructs otherwise**.
- If additional values are required internally, use **another tool call** to resolve those dependencies without involving the user.
- Always prioritize a **realistic, helpful, user-centric experience** over full parameter collection. The user should not feel like they're configuring a tool—they're just having a conversation.
- [round_7_update] ***Context Information System Prompt:***
    - ***Goal:*** Enable the assistant to leverage immediately available context about the applications, entities, or media the user is currently interacting with—without requiring additional tool calls to retrieve that information.
    - ***Approach:*** You can include a **dedicated block in the System Prompt** that provides background context—such as the app the user is on, currently visible content, or highlighted entities.

This allows the assistant to:

- **Refer to the context directly** using pronouns like *"this"* or *"it"* without ambiguity.
- **Avoid redundant search calls**, since the data is already "in view".
- Make intelligent assumptions about what the user means, based on the shared session context.
- Design Guidelines:
    - **Include meaningful context** in the SM (e.g., app state, visible entities, currently playing media).
    - Ensure the user prompt refers to that context **implicitly or vaguely** (e.g., "Add this to my favorites" or "Tell me more about this artist").

    - The assistant should **reason over the given SM context** and proceed as if the user is referring to the most relevant entity in view.
    - Avoid issuing tool calls to "search" for something already included in context.
- Example Context Block in SM:

```
The user is currently looking at the Spotify app. The song
currently playing is:
- Track ID: 5q2KGgJV5l4MAnnGIrvKSK
- Name: Human After All / Together / One More Time / Music
Sounds Better with You
- Artist: Daft Punk
The page the user is viewing also includes visible artists:
- Daft Punk, Justice, Gorillaz, LCD Soundsystem
```

- Example Prompt & Behavior:
    - User Prompt: "Which of these artists has the most monthly listeners?"
        - ✅ Correct Behavior: Assistant compares Daft Punk, Justice, Gorillaz, LCD Soundsystem (based on the context) using a lookup.
        - ❌ Incorrect Behavior: Assistant says "Please name the artists" or runs a broad artist search.
- **Important Notes:**
    - **Do not repeat tool calls** to retrieve what's already part of the SM context.
    - This category emphasizes **stateful awareness**, i.e., the assistant "sees what the user sees".
    - Use natural references like *"this"*, *"these artists"*, or *"currently playing"* in prompts to test contextual grounding.
    - Ensure the **System Prompt is rich enough** to allow meaningful grounding without extra queries.

This category tests the assistant's ability to **intelligently use provided context** and interpret vague references based on the user's current session, similar to how a real assistant might react when observing your app screen.