MRPilot: A Mixed-Reality System for Responsive Navigation of General Procedural Tasks

Hongliang Yang
Shenzhen University

Jin Zhou
Shenzhen University

Pengfei Xu*
Shenzhen University

Hongbo Fu
HKUST

Hui Huang
Shenzhen University

Supplemental Materials

This supplementary material provides detailed information supporting the main paper, including the full participant-task assignment table used in the user study to evaluate *Baseline* and *MRPilot*. We include real-world deployment results of the Object Anchor Module as well as implementation details of the four LLM-based agents in our system in Section 1 and 2. Additionally, we include statistical analysis of user feedback in Section 3. Specifically, we include latency statistics from repeated API calls and the complete prompt templates for the **Instruction Drafting Agent**, **Scene Analysis Agent**, **Task Structuring Agent**, and **Action Recommendation Agent** in Section 4.

1 Object Anchor

The real-world deployment results of the Object Anchor Module are illustrated in Figure 1.

2 User Study Settings

Table 1 shows the specific tasks and interface conditions that were assigned while evaluating *Baseline* and *MRPilot*.

Table 1: The study was designed to evaluate *MRPilot*, with each participant required to complete two experimental sessions across the four phases, following the sequence outlined above.

Participant	Task 1	Task 2
P1	*MRPilot* T2	*Baseline* T3
P2	*MRPilot* T3	*Baseline* T2
P3	*Baseline* T3	*MRPilot* T2
P4	*Baseline* T2	*MRPilot* T3
P5	*MRPilot* T2	*Baseline* T3
P6	*MRPilot* T3	*Baseline* T2
P7	*Baseline* T3	*MRPilot* T2
P8	*Baseline* T2	*MRPilot* T3
P9	*MRPilot* T2	*Baseline* T3
P10	*MRPilot* T3	*Baseline* T2
P11	*Baseline* T3	*MRPilot* T2
P12	*Baseline* T2	*MRPilot* T3
P13	*MRPilot* T4	*Baseline* T5
P14	*MRPilot* T5	*Baseline* T4
P15	*Baseline* T5	*MRPilot* T4
P16	*Baseline* T4	*MRPilot* T5
P17	*MRPilot* T4	*Baseline* T5
P18	*MRPilot* T5	*Baseline* T4
P19	*Baseline* T5	*MRPilot* T4
P20	*Baseline* T4	*MRPilot* T5
P21	*MRPilot* T5	*Baseline* T4

*Corresponding author, e-mail: xupengfei.cg@gmail.com

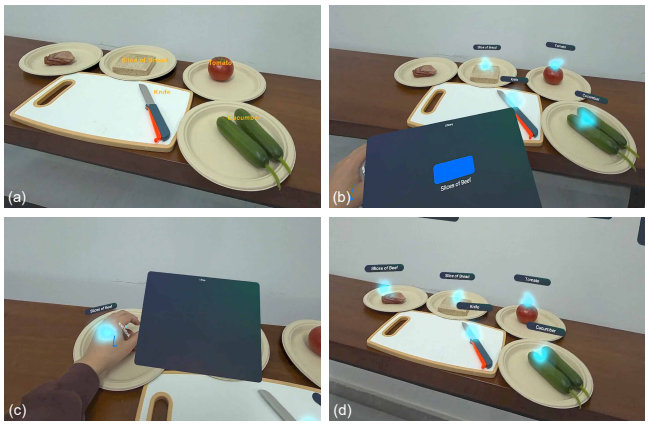

Figure 1: In automatic anchoring mode, labels displaying object names will be overlaid on recognized objects (a). Once the user confirms the anchor, the object anchor is overlaid on the corresponding object (b). Users can also use hand gestures to manually anchor objects that cannot be automatically recognized (c). All task-related objects have been properly anchored. (d)

3 Statistical Analysis of User Feedback

To evaluate user workload and system usability, we employed an adaptive version of the NASA Task Load Index (NASA-TLX) and the System Usability Scale (SUS). Both questionnaires utilized a 5-point Likert scale. For the NASA-TLX, lower scores indicate a better outcome (i.e., lower workload). For the SUS, individual items were rated on a 5-point Likert scale. To compute the overall SUS score, we first converted the responses for each item to a 0-4 scale: for positively-worded items, the score was the original rating minus 1, and for negatively-worded items, the score was 5 minus the original rating. These converted scores were then summed and multiplied by 2.5 to yield a final composite score on a 0-100 point scale, where higher scores indicate better usability. For the analysis of individual SUS sub-items, we used a converted 1-5 point scale where 1 consistently represented the worst usability contribution and 5 represented the best.

Since questionnaire evaluation for NASA-TLX and SUS score is not interval data, we use Wilcoxon Signed-Rank tests for all collected scores. We report the details of the statistical results in Table 2 and the boxplot of the overall SUS score, SUS score for each sub-items and NASA-TLX sub-items score in Figure 2, 3, and 4. Additionally, we report the mean and standard deviation of the manual switching data and user feedback data for each task category in Table 5 and the mean score of NASA-TLX and SUS of each task in Table 4. However, due to the limited number of samples within each task category, no statistical tests are conducted for these subsets.

The overall statistical results indicate that *MRPilot* significantly outperforms *Baseline* in both reducing user workload and enhancing usability. For all NASA-TLX factors, *MRPilot* consistently yielded lower mean scores, reflecting reduced mental, physical, and

Table 2: The statistical results of user feedback between *MRPilot* and *Baseline*, where the p-values ($+ : .050 < p < .100$, $* : p < .050$, $** : p < .010$, $*** : p < .001$) is reported. Here, Z denotes the results of Wilcoxon signed-rank tests for non-normal data.

Category	Factor	*MRPilot*		*Baseline*		Statistics			
		Mean	SD	Mean	SD	Z	p	d (Effect size)	Sig.
NASA-TLX (↓)	Mental	1.571	0.811	2.476	1.123	-2.551	0.00981	0.557	**
	Physical	1.667	0.913	2.524	1.209	-2.900	0.00298	0.633	**
	Temporal	1.476	0.750	2.667	1.111	-2.973	0.00260	0.649	**
	Performance	1.810	0.680	2.857	1.153	-2.762	0.00467	0.603	**
	Effort	1.952	0.805	3.048	1.161	-2.911	0.00298	0.635	**
	Frustration	1.524	0.602	2.476	1.209	-2.811	0.00417	0.614	**
SUS (↑)	Use frequently	3.476	0.928	2.190	1.123	-3.051	0.00151	0.666	**
	Simple	3.905	0.700	3.143	1.195	-2.589	0.00816	0.565	**
	Easy to use	3.857	0.727	2.667	1.111	-3.067	0.00188	0.669	**
	Without support	4.000	1.000	3.381	1.284	-1.870	0.05515	0.408	+
	Function integrated	4.143	1.062	2.476	1.289	-3.413	0.00055	0.745	***
	Consistency	4.429	0.746	3.571	1.207	-2.401	0.01491	0.524	*
	Learn quickly	4.571	0.598	3.429	1.121	-3.045	0.00198	0.664	**
	Intuitive	3.952	0.921	2.857	1.315	-2.533	0.00984	0.553	**
	Confidence	3.476	1.209	2.381	1.024	-2.585	0.00579	0.564	**
	Without learning	4.381	0.669	3.762	1.338	-1.789	0.06792	0.390	+
	Overall SUS Score	75.476	11.029	49.643	21.727	-3.280	0.00103	0.716	**

Table 3: Latency statistics of different agents.

Agent	Model	Avg Latency (s)	Min (s)	Max (s)	Std Dev (s)
Instruction Drafting Agent	gpt-4o	5.93	1.07	15.28	1.76
Scene Analysis Agent	gpt-4o	5.63	3.82	34.54	3.91
Task Structuring Agent	gpt-4o	7.45	4.53	19.40	2.63
Action Recommendation Agent	gpt-4o-mini	3.21	1.34	13.73	3.32

Table 4: Comparison of *MRPilot* and *Baseline* across tasks using NASA-TLX and SUS mean score.

Task	Category	*MRPilot*	*Baseline*
Food preparation (6)	NASA	1.556	2.444
	SUS	3.883	3.611
Healthcare (6)	NASA	1.944	2.417
	SUS	4.000	3.250
Science education (5)	NASA	1.467	3.500
	SUS	4.220	2.292
Manual assembly (4)	NASA	1.667	2.600
	SUS	4.000	2.867

temporal demands, as well as lower effort and frustration levels, with all differences reaching statistical significance ($p < .01$) and moderate to large effect sizes (e.g., Mental: 0.557; Effort: 0.635). Similarly, for the SUS metrics, *MRPilot* achieved higher ratings across key dimensions such as frequency of use, simplicity, ease of use, and functional integration, with most differences being significant (e.g., Function integrated: $p < .001, d = 0.745$; Learn quickly: $p < .01, d = 0.664$), highlighting substantial improvements in intuitiveness and learnability. Although a few factors, such as "Without support" and "Without learning," showed marginal significance ($.050 < p < .100$), the overall findings provide strong evidence that *MRPilot* effectively reduces user workload while improving usability, demonstrating its practical advantages over *Baseline*.

Across the four tasks, *MRPilot* consistently demonstrated lower cognitive workload and higher usability than *Baseline*, though the extent of improvement varied by task. The Science education and Manual assembly tasks showed the most pronounced benefits, with *MRPilot* achieving the lowest NASA-TLX scores (1.467 and 1.667) and highest SUS scores (4.220 and 4.000), suggesting that structured workflows and spatial reasoning tasks particularly benefited from its step-by-step guidance. In comparison, the Healthcare task, while still showing improvements (NASA-TLX: 1.944; SUS: 4.000), exhibited a slightly higher workload, likely due to the difficulty of performing fine operations, such as tearing a tap, under the MR environment. The Food preparation task, being more routine and physically oriented, yielded moderate workload reduction (1.556) and relatively modest usability gains (3.883). Overall, these results highlight that *MRPilot*'s advantages are more pronounced in tasks demanding clear procedural navigation and spatial support while still offering meaningful improvements in routine and precision-focused scenarios.

4 LLM Settings

4.1 LLM Latency

To ensure consistent and high-quality performance across specialized agents, we developed a systematic framework for testing agent response latency. Each agent's prompt is defined using templates in Section 4.2 and parameterized via YAML files, enabling the generation of diverse prompt instances for robust evaluation.

For agents handling visual inputs (e.g., the Scene Analysis Agent), the framework supports image-prompt pairing and integrates with Vision APIs.

For agents with prompt parameters that represent different system states (e.g., the Action Recommendation Agent), we evaluated performance under a range of conditions. Specifically, we measured response latency across multiple task progression scenarios, including initial states with all tasks pending, mid-task execution states, and near-completion stages. This comprehensive evaluation

Table 5: By task user feedback statistics between *MRPilot* and *Baseline*. Numbers in parentheses indicate participant count per task.

Task	**Category**	**Factor**	*MRPilot*		*Baseline*	
			Mean	SD	Mean	SD
Food preparation (6)	NASA-TLX (↓)	Mental	1.333	0.516	2.167	1.169
		Physical	1.167	0.408	2.500	1.643
		Temporal	1.333	0.516	2.000	0.894
		Performance	1.667	0.816	3.000	1.549
		Effort	2.333	0.516	2.667	1.033
		Frustration	1.500	0.548	2.333	1.366
	SUS (↑)	Use frequently	3.500	0.837	2.833	1.722
		Simple	3.667	0.516	3.833	1.169
		Easy to use	3.500	1.049	3.333	1.211
		Without support	4.167	0.753	4.000	1.265
		Function integrated	3.833	1.835	3.333	1.633
		Consistency	4.167	1.169	3.833	1.169
		Learn quickly	4.333	0.816	3.833	1.169
		Intuitive	3.667	0.816	3.500	1.378
		Confidence	3.333	0.816	3.000	1.414
		Without learning	4.667	0.516	4.167	1.169
Healthcare (6)	NASA-TLX (↓)	Mental	1.667	0.816	2.000	0.894
		Physical	2.167	1.169	2.167	0.753
		Temporal	1.667	1.033	2.667	1.033
		Performance	1.833	0.408	2.667	0.816
		Effort	2.500	0.837	2.833	1.169
		Frustration	1.833	0.753	2.167	1.169
	SUS (↑)	Use frequently	3.167	1.472	2.000	0.632
		Simple	4.333	0.816	3.500	1.225
		Easy to use	4.167	0.753	2.667	1.033
		Without support	3.500	1.517	3.833	1.169
		Function integrated	4.167	0.753	2.167	0.753
		Consistency	4.667	0.516	4.333	0.816
		Learn quickly	4.667	0.516	3.667	0.816
		Intuitive	4.167	0.753	3.000	1.414
		Confidence	3.167	1.835	2.000	0.632
		Without learning	4.000	0.894	4.333	1.211
Science education (5)	NASA-TLX (↓)	Mental	1.500	0.577	3.200	1.304
		Physical	2.000	1.155	2.200	1.304
		Temporal	1.500	0.577	3.000	1.000
		Performance	2.000	0.816	2.200	1.095
		Effort	1.500	0.577	3.000	1.414
		Frustration	1.500	0.577	2.000	1.000
	SUS (↑)	Use frequently	3.500	0.577	2.000	0.707
		Simple	3.750	0.500	2.600	0.548
		Easy to use	3.750	0.500	2.400	0.548
		Without support	4.500	0.577	2.800	1.483
		Function integrated	4.000	0.000	2.600	1.342
		Consistency	4.250	0.500	2.800	1.483
		Learn quickly	4.750	0.500	3.200	0.837
		Intuitive	3.500	1.291	2.600	1.517
		Confidence	3.500	0.577	2.600	0.894
		Without learning	4.500	0.577	3.400	1.140
Manual assembly (4)	NASA-TLX (↓)	Mental	1.800	1.304	2.750	0.957
		Physical	1.400	0.548	3.500	0.577
		Temporal	1.400	0.894	3.250	1.500
		Performance	1.800	0.837	3.750	0.500
		Effort	1.200	0.447	4.000	0.816
		Frustration	1.200	0.447	3.750	0.500
	SUS (↑)	Use frequently	3.800	0.447	1.750	0.957
		Simple	3.800	0.837	2.250	1.258
		Easy to use	4.000	0.000	2.000	1.414
		Without support	4.000	0.707	2.500	0.577
		Function integrated	4.600	0.548	1.500	0.577
		Consistency	4.600	0.548	3.000	0.816
		Learn quickly	4.600	0.548	2.750	1.708
		Intuitive	4.400	0.894	2.000	0.000
		Confidence	4.000	1.225	1.750	0.500
		Without learning	4.400	0.548	2.750	1.708

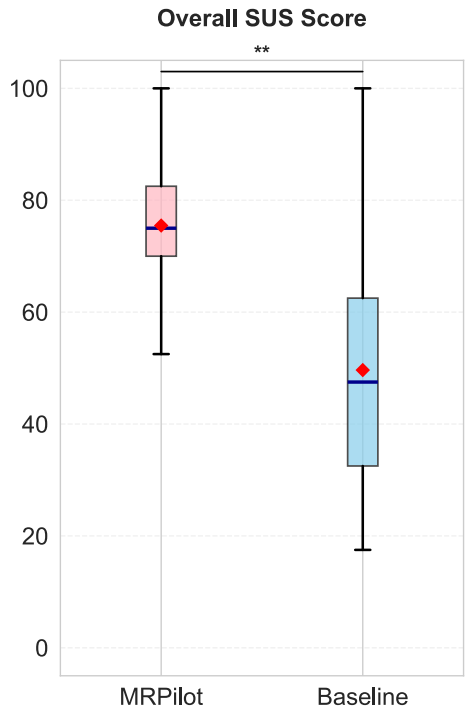

Figure 2: The boxplot of overall SUS score.

allowed us to assess the agent's responsiveness and ability to handle varying input contexts of real-world usage patterns.

All test parameters were derived from the four experimental tasks used in our user study: Food preparation, Healthcare, Science education, and Manual assembly. This ensured that the prompt evaluation process remained aligned with real-world task settings and user interaction patterns observed during empirical validation.

To quantitatively evaluate the agents' latency, we measured the API response latency of all four agents over 50 independent runs per agent. The results, summarized in Table 3, provide a comparative view of average response times across agents under standardized test conditions, thereby enabling a fair and objective assessment of their relative performance.

4.2 LLM Prompts

We show full prompt templates used for each agent, including Instruction Drafting, Scene Analysis, Task Structuring, and Action Recommendation agents passed to the LLM in Figures 5, 6, 7 and 8.

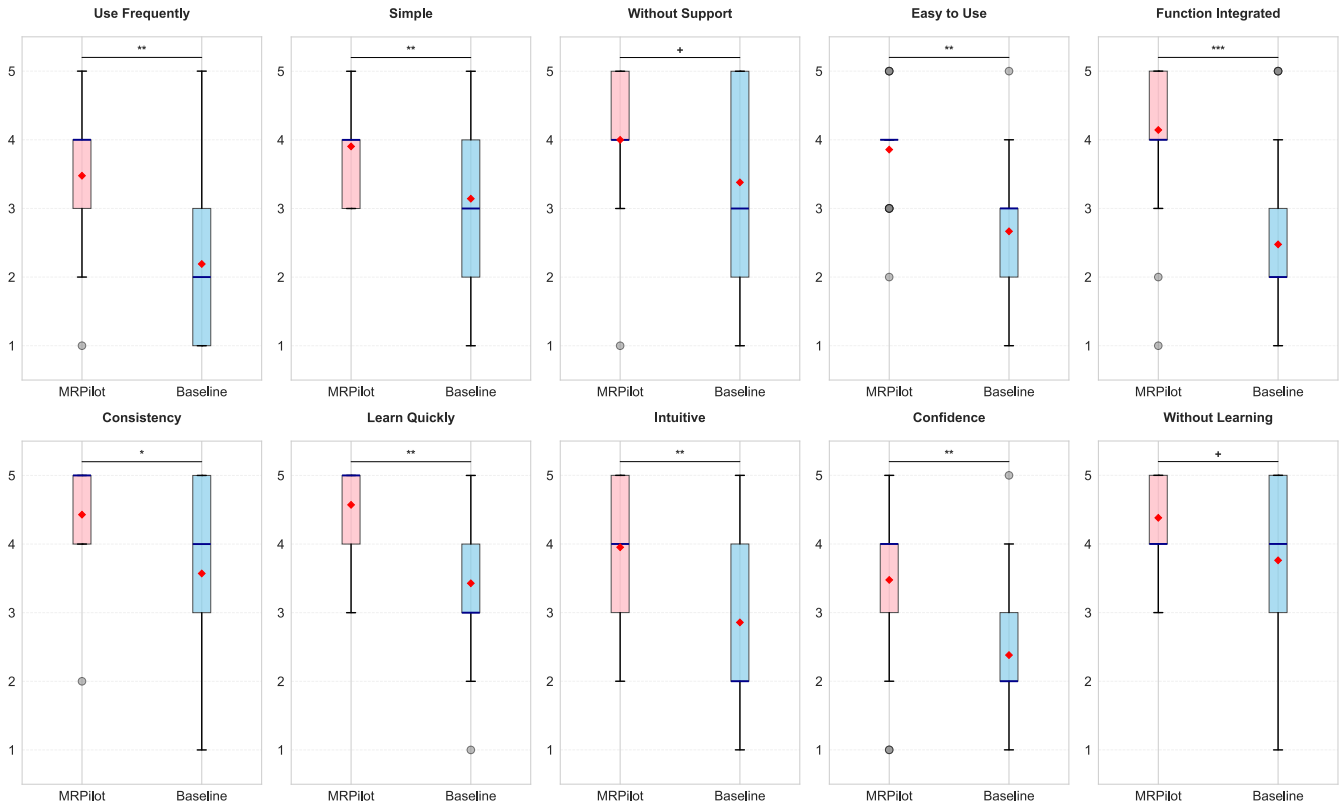

Figure 3: The boxplot of SUS score for each sub-items.

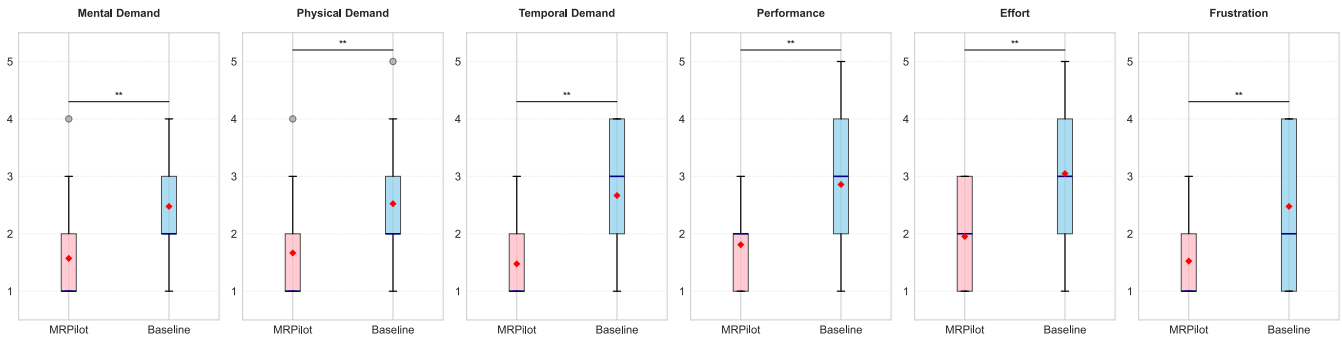

Figure 4: The boxplot of NASA-TLX score for each sub-items.

```
**Role:**
As a Task Guide Generator, your role is to assist users in completing their tasks by providing clear, detailed, and actionable
     step-by-step guides.
You will analyze the user's specific request and the items they have, then generate a tailored plan to help them accomplish their
     goal.

**Inputs:**
${something_todo}

**Instructions:**
The text in the input section describes what the user wants to accomplish, along with the items that are available.

- Based on the user's request, provide a clear and detailed step-by-step guide with precise numerical indicators for each action.
- Wherever possible, include specific quantities, counts, or measurements to further clarify each task or process. (e.g. Layer with
     2 lettuce and 3 pieces of tomato, using middle fire for 5 minutes, applying 1 pump, for at least 20 seconds)
- Generate a complete list of the required items that the user will need to accomplish the task. This list should take into account
     the items the user already has ONLY.
- Ensure that each step is in logical order, and if necessary, include tips or best practices to make the process smoother.
- Do not generate optional steps (e.g. If need, ....) or unnecessary steps (e.g. enjoy your work/ ... is ready to enjoy).


**Output Example:**
## Item List:
1. ${item_1} - {item_1_description}
2. ${item_2} - {item_2_description}
3. ${item_3} - {item_3_description}
4. ...... (may include more items based on the user's request)
## Step-by-Step Guide:
1. **Initial Setup**:
   - Place ${item_1} on a stable surface and ensure all necessary tools are within reach.
2. **Main Process Name 1**:
   - Check ${item_1} for any issues such as damage or improper setup.
   - If needed, make adjustments using ${item_2}.
3. **Main Process Name 2**:
   - Begin by using ${item_3} to perform the primary task, following any specific guidelines.
   - Ensure everything is properly aligned/connected during the process.
4. **Main Process Name 3**:
   - Make adjustments with ${item_2}.
5. ...... (may include more steps based on the user's request)
6. ...... (may include more steps based on the user's request)
7. ...... (may include more steps based on the user's request)
8. ...... (may include more steps based on the user's request)
```

Figure 5: Prompt of Instruction Drafting Agent.

```
The user wants to do: {$something_todo}

please identify the objects needed for doing this task in this image.

- ONLY output objects in the image as an unordered list.
- Do not output optional objects.
```

Figure 6: Prompt of Scene Analysis Agent.

````
**Role:**
Guide Organization Assistant - Structuring a guide into an organized JSON process with indexed tasks and related items.

**Inputs:**
```
{$guide_text}
```

**Instructions:**
Organize this guide as a process step that meets the following requirements.
- Output the organized result in JSON format.
- Subtasks/Task/Item must be indexed from index 0.
- Ensure each subtask includes related items, which cannot be null.
- Maintain a step-by-step process where each task and subtask reflects the specific instructions from the guide text as closely as
    possible, without summarizing or omitting details.
- Explicitly include any quantities, counts, measurements, or time durations mentioned in the guide for each subtask. Example:
    "Layer with 2 lettuce leaves and 3 tomato slices", "Use medium heat for 5 minutes", "Apply 1 pump for at least 20 seconds."

**Output Example:**
```json
{
  "items": [{"id": 0, "item_name": "item_name_1"}, {"id": 1, "item_name": "item_name_2"}, {"id": 2, "item_name": "item_name_3"}],
  "tasks": [
    {
      "index": 0, "task_name": "task_name_1", "subtasks": [
        {"index": 0, "subtask_name": "subtask_name_1_1", "related_items": [0]},
        {"index": 1, "subtask_name": "subtask_name_1_2", "related_items": [0, 1]}
      ]
    },
    {
      "index": 1, "task_name": "task_name_2", "subtasks": [
        {"index": 0, "subtask_name": "subtask_name_2_1", "related_items": [1]},
        {"index": 1, "subtask_name": "subtask_name_2_2", "related_items": [2]}
      ]
    },
    {
      "index": 2, "task_name": "task_name_3", "subtasks": [
        {"index": 0, "subtask_name": "subtask_name_3_1", "related_items": [2]},
        {"index": 1, "subtask_name": "subtask_name_3_2", "related_items": [0, 1, 2]}
      ]
    }
  ]
}
```
````

Figure 7: Prompt of Task Structuring Agent.

````
**Role:**
You are a procedural task optimization expert. Analyze task dependencies and resource usage to recommend the most efficient next
     steps, prioritizing parallelizable subtasks.

**Inputs:**
1. **Current Status (JSON):**
   {$current_status_json}

**Key Analysis Rules:**
1. **Dependency Check**
   - If a subtask requires specific items or steps from another task, mark it as dependent.

2. **Concurrency Criteria**
   - Recommend subtasks that:
     - Use **different items** than ongoing tasks
     - Are in a **different task branch**
     - Have no uncompleted dependencies
     - If it is the very first subtask, as long as it is `NotStarted` and has no prerequisites, it should be recommended

3. **State Filtering**
   - Ignore `Completed` or `InProgress` subtasks.

4. **First Subtask Rule**
   - If this subtask is the first `NotStarted` subtask in the entire workflow, **always recommend it** unless there is an explicit
     dependency preventing it.

**Output Requirements:**
- If no recommendations, return an empty list with `reason: "No parallelizable tasks available"`.
- There can be multiple `recommended_subtasks`

**Example Output with Context:**
```json
{
  "recommended_subtasks": [
    {
      "task_index": 2,
      "subtask_index": 0,
    }
  ]
}
```
````

Figure 8: Prompt of Action Recommendation Agent.