ORG-AGENT: BEYOND PERSONAL ASSISTANTS TOWARDS ORGANIZATIONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language model agents serving organizations must coordinate requests from multiple users while using knowledge distributed across their interactions. We identify two complementary capabilities for this setting, namely cross-user interaction and decision-making, as well as cross-user memory and knowledge use. Both capabilities are governed by organizational constraints across three aspects: user identity, authority, and access permissions; the attribution and temporal validity of information; and rules for resolving conflicting requirements across users and completion requirements for joint decisions. These constraints shape what information or decisions must be obtained before an action can proceed and what conditions must be satisfied during its execution. Motivated by this, we introduce ORG-AGENT, a unified constraint-centric reasoning framework that organizes task execution in three stages. Specifically, ORG-AGENT decomposes a task into atomic subtasks and constructs a task dependency graph whose edges encode the dependencies among them. Building on this graph, it schedules the subtasks in dependency order through topological sorting. It then executes each subtask while accounting for the task's constraints, supported by evidence-acquisition and memory-management tools. Experiments on MUSES-Bench and GroupMemBench demonstrate the effectiveness of ORG-AGENT on both capabilities, and ablations further support the contributions of dependency modeling and tool use. Our code is available at https://anonymous.4open.science/r/ORG-AGENT-80C3.

1 INTRODUCTION

Language model agents (Zhou et al., 2026; Singh et al., 2025; DeepSeek-AI, 2026) offer a path toward automating organizational workflows (Xu et al., 2025). These workflows often involve multiple users, each holding part of the information and authority needed to complete tasks. However, existing agent systems are largely designed and evaluated for a single user (Huang et al., 2026; Chhikara et al., 2025; Yao et al., 2025). Extending these agents to organizational settings is not merely a matter of scale, but of coordinating requests across users and using knowledge distributed among them. This motivates our study of *agents for organizations*, where one shared agent serves multiple users and supports the completion of their tasks.

We identify two complementary capability dimensions for such agents, as illustrated in Figure 1. *Cross-user interaction and decision-making* involves interacting with multiple users to coordinate their requests and make decisions. For example, scheduling a product launch requires bringing together the manager's proposed timeline and the progress reported by the engineer and designer to develop a joint launch plan. *Cross-user memory and knowledge use* involves organizing and retrieving knowledge accumulated through interactions with different users to answer subsequent questions. For example, answering a question about a product's login method requires referring back to earlier project discussions among team members. Recent benchmarks make these requirements concrete. MUSES-Bench, introduced in *Multi-User Large Language Model Agents* (Yang et al., 2026b), evaluates agents on tasks involving instruction selection and following, cross-user information access, and meeting scheduling, while GroupMemBench (Yang et al., 2026a) examines memory and knowledge use in multi-user conversations. These studies establish evaluation settings and show that current agents and memory systems still struggle with both capabilities.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

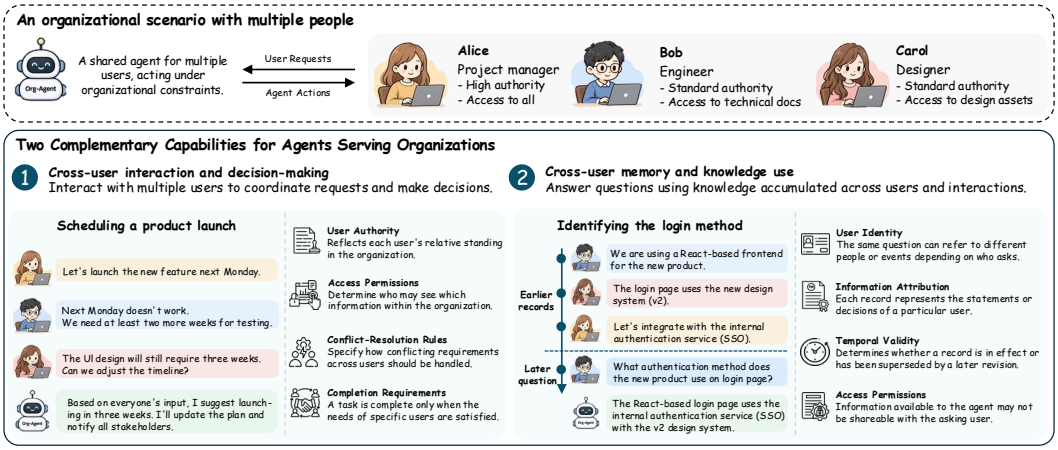


Figure 1: **Two complementary capabilities of agents for organizations.** Left: *cross-user interaction and decision-making*, illustrated by coordinating a product launch based on multiple users' requests. Right: *cross-user memory and knowledge use*, illustrated by answering a question about the login method using prior interaction history.

Both capabilities are governed by organizational constraints across three aspects. Regarding users, identity concerns who issues a request, authority reflects the user's relative standing in the organization, and access permissions concern who may see which information. Regarding information, attribution identifies whose statements or decisions the information represents, while temporal validity concerns whether a record is still in effect or has been superseded by a later revision. Regarding joint decisions, conflict-resolution rules specify how conflicting requirements across users should be handled, while completion requirements specify which users' needs must be satisfied for the task to be considered complete. Simply concatenating all users' requests and interactions does not explicitly represent or enforce these constraints, while handling each user's requests independently can overlook the dependencies that these constraints induce across users.

To this end, we propose ORG-AGENT, a constraint-centric reasoning framework with three stages. First, *task decomposition and graph construction* decomposes the task into executable subtasks and organizes them into a directed acyclic graph (DAG) (Thost & Chen, 2021), whose edges encode the dependencies among subtasks, and the graph is updated as needed when new user inputs change the pending subtasks. Second, *dependency-aware scheduling* topologically sorts the graph into an execution order in which every node comes after the nodes it depends on. Third, *constraint-aware execution* invokes tools according to each node's objective and the constraint rules to support decision-making or acquire information from the interaction history. Specifically, we design evidence-acquisition tools that rank records by similarity, filter them by metadata conditions, and traverse explicit links from a given record to retrieve related records, as well as memory-management tools that maintain an execution memory of accumulated information and intermediate results for subsequent nodes. Together, these stages equip ORG-AGENT with both capabilities within a unified framework, extending a single-user agent to an organizational agent. Our contributions are fourfold:

- We identify *cross-user interaction and decision-making*, as well as *cross-user memory and knowledge use* as complementary capabilities of organizational agents, highlighting that both are governed by organizational constraints. This motivates the design of ORG-AGENT, a unified framework that organizes execution around these constraints and thereby supports both capabilities.
- ORG-AGENT constructs a dynamic task dependency graph, termed TDG, which is a directed acyclic graph whose nodes are atomic subtasks and whose edges encode the dependencies among them, and updates the graph as needed when new user inputs change the pending subtasks.
- On top of the constructed graph, ORG-AGENT schedules the subtasks in dependency order by topological sorting and executes each of them under the constraint rules, with evidence-acquisition tools for retrieving and linking relevant records from the interaction history and memory-management tools for maintaining accumulated execution memory across subtasks.

- Experiments on MUSES-Bench and GroupMemBench demonstrate the effectiveness of ORG-AGENT in cross-user interaction and decision-making as well as cross-user memory and knowledge use, respectively. Ablation studies further show the contributions of dependency-aware scheduling and tool-augmented execution on both benchmarks.

2 RELATED WORK

Recent work has begun to examine LLM agents in multi-user environments. MUSES-Bench (Yang et al., 2026b) evaluates the ability of a single agent to serve multiple users within a shared interaction setting, while PeopleJoin (Jhamtani et al., 2025) studies how agents identify and communicate with relevant users to gather distributed information. In parallel, agent memory systems (Zhang et al., 2025) have evolved to help agents store and retrieve information from past interactions. Early work such as MemGPT (Packer et al., 2023) manages hierarchical memory to extend the effective context window. Subsequent systems extract and structure salient information from conversations. Mem0 (Chhikara et al., 2025) consolidates key facts from conversations, while Zep (Rasmussen et al., 2025) and Hindsight (Latimer et al., 2025) further organize memory into temporal knowledge graphs and structured memory networks, respectively. More recent systems such as LightMem (Fang et al., 2026) and SimpleMem (Liu et al., 2026) improve efficiency through offline consolidation and compact memory representations. Despite this progress, GroupMemBench (Yang et al., 2026a) reveals persistent limitations of existing memory systems in using information from multi-user conversations. Rather than focusing solely on memory storage and retrieval, ORG-AGENT makes these constraints explicit during execution and supports cross-user interaction and decision-making as well as cross-user memory and knowledge use.

3 PRELIMINARIES

We consider an agent for an organization that serves a set of N users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$. Interaction proceeds in discrete turns indexed by $t \in \{1, 2, \ldots, T\}$. At each turn, users may submit requests, provide information, or respond to the agent. We denote the input from user u_i at turn t by $I_{i,t}$, which is empty if the user provides no input. Let $\mathcal{H}_t$ denote the interaction history before turn t, which consists of records of conversations among users and their exchanges with the agent. Given $\mathcal{H}_t$, the current user inputs, and the organizational constraints $\mathcal{R}$, the agent produces a set of actions

$$\mathcal{A}_t = \pi\big(\mathcal{H}_t, \{I_{i,t}\}_{u_i \in \mathcal{U}}, \mathcal{R}\big), \tag{1}$$

where π denotes the agent policy. Actions may include retrieving information, requesting input from users, and providing responses or task outcomes. The interaction ends once all pending requests are addressed. Next, we describe the interaction process for each of the two capabilities.

Cross-user interaction and decision-making. The agent handles requests submitted by multiple users over multiple turns. Here $\mathcal{H}_t$ mainly records the exchanges between the agent and the users in earlier turns of the task. The user inputs and agent responses at turn t are added to the history to form $\mathcal{H}_{t+1}$. Moreover, each input $I_{i,t}$ may contain multiple requests submitted by user u_i at turn t.

Cross-user memory and knowledge use. We formulate this setting as a single-turn question-answering task, where the only input $I_{i,1}$ is a question from user u_i. Here $\mathcal{H}_1$ mainly consists of earlier conversations among users that precede the question, and $\mathcal{A}_1$ consists of actions that retrieve and examine relevant records from $\mathcal{H}_1$, followed by a single answer to u_i.

4 METHOD

4.1 OVERVIEW

We analyze that the constraints in organizational tasks confront the agent with two fundamental issues. **(i) The prerequisites of each action are distributed across users and the interaction history.** For example, a coordination action may require approval from a higher-authority user

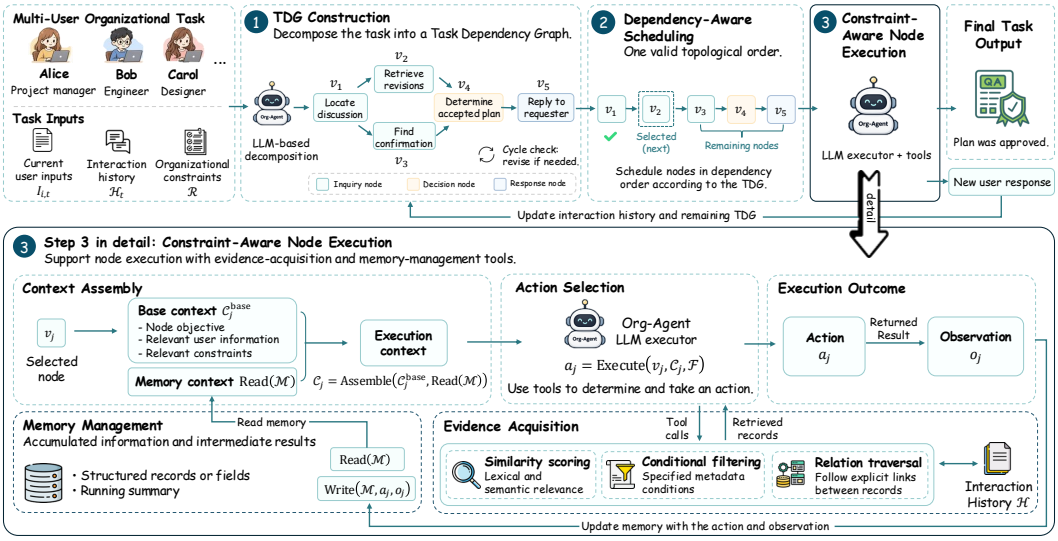


Figure 2: **Overview of ORG-AGENT.** Given user inputs, interaction history, and organizational constraints, ❶ **TDG construction** decomposes the task into subtasks and represents their dependencies in a Task Dependency Graph; ❷ **dependency-aware scheduling** orders the nodes through topological sorting; and ❸ **constraint-aware node execution** uses the LLM to determine and take actions toward node objectives based on the assembled execution context, with evidence-acquisition tools providing relevant records and memory-management tools supplying memory for context assembly and updating it with information from actions and observations.

before it can proceed. Similarly, an information-retrieval action may require identifying the project referred to in a question before retrieving the relevant records. The agent must therefore identify the prerequisites of each action and organize execution according to the resulting dependencies. **(ii) Each action must satisfy the relevant constraints during execution.** The agent must therefore determine which constraints in $\mathcal{R}$ apply to each action and ensure that the action satisfies them. For example, an answer to a question must not disclose information the user is not authorized to access.

Accordingly, our core idea is to decompose a task into executable subtasks and represent their dependencies as a directed acyclic graph. We address (i) by using this graph to execute subtasks in dependency order (Sections 4.2 and 4.3), and (ii) by applying the relevant constraint rules in $\mathcal{R}$ during each subtask's execution, with tools supporting evidence acquisition and memory management (Section 4.4). The overall framework is illustrated in Figure 2.

4.2 TASK DEPENDENCY GRAPH CONSTRUCTION

At turn t, we decompose the current pending requests into executable subtasks and organize them into a directed acyclic graph, termed the Task Dependency Graph (TDG). We then describe the details of the graph formulation and construction process below.

Graph Modeling. Let $\mathcal{D}_t = \{d_{t,1}, \ldots, d_{t,K_t}\}$ denote the set of subtasks at turn t, where K_t is the number of subtasks. We represent their dependencies as a directed acyclic graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, where $\mathcal{V}_t = \{v_{t,1}, \ldots, v_{t,K_t}\}$ and each node $v_{t,j}$ corresponds to subtask $d_{t,j}$. Specifically, we distinguish three types of nodes, namely an inquiry node that retrieves relevant records from the interaction history $\mathcal{H}_t$ or requests information from a user, a decision node that makes a task-specific decision, such as outputting the conclusion of accepting or rejecting a user's instruction, and a response node that replies to a user request. The directed edge set $\mathcal{E}_t \subseteq \mathcal{V}_t \times \mathcal{V}_t$ encodes dependencies among subtasks. An edge $(v_{t,j}, v_{t,k}) \in \mathcal{E}_t$ indicates that executing $v_{t,k}$ requires the output of $v_{t,j}$.

Graph Construction. Building on this formulation, we describe how the framework constructs the graph from the current pending requests in three steps.

- **Subtask decomposition.** An LLM identifies the atomic subtasks needed to fulfill the pending requests from the inputs $\{I_{i,t}\}_{u_i \in \mathcal{U}}$ and the history $\mathcal{H}_t$, taking the constraint rules in $\mathcal{R}$ into account. Each subtask is represented by a concrete objective and associated attributes.
- **Dependency identification.** Given the subtask descriptions and relevant task rules, the LLM identifies which nodes must be resolved before others can be executed and constructs the edge set $\mathcal{E}_t$ accordingly. Cycles are checked and, if detected, revised by the LLM.
- **Graph update.** As nodes are completed, the framework advances through the execution plan. If new user inputs change the pending subtasks, it adjusts the remaining plan, reusing existing dependencies where applicable and reconstructing the graph when needed.

4.3 Dependency-Aware Scheduling

Although the Task Dependency Graph captures dependencies among subtasks, these dependencies must still be translated into an execution order. This stage derives such an order from the acyclic structure of the graph, so that every node is executed only after its prerequisites are available. For brevity, we omit the turn index t from the graph notation hereafter.

Topological Sorting. Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with K nodes, we perform a topological sort and index the nodes by their positions in the resulting sequence $\langle v_1, v_2, \ldots, v_K \rangle$, so that

$$(v_j, v_k) \in \mathcal{E} \quad \Longrightarrow \quad j < k. \tag{2}$$

This ensures that every prerequisite node precedes the nodes that depend on its output. Nodes can be executed sequentially in this order or organized into topological layers. In layer-wise execution, each layer begins after the preceding layers have completed, and nodes within the same layer may execute in parallel since they have no dependencies on one another.

4.4 Constraint-Aware Node Execution

Rather than relying solely on the model's reasoning to deal with the constraint rules, we support node execution through two categories of predefined tools, namely evidence-acquisition tools and memory-management tools. Let $\mathcal{F}$ denote the tool set. Given its context $\mathcal{C}_j$, each node v_j uses the LLM and the available tools in $\mathcal{F}$ to determine and take an action toward its objective:

$$a_j = \text{Execute}\,(v_j, \mathcal{C}_j, \mathcal{F})\,. \tag{3}$$

After taking action a_j, the node receives the corresponding result as observation o_j, and the actions produced by nodes executed at turn t are included in the action set $\mathcal{A}_t$. The two categories of tools used to support node execution are described below.

Evidence Acquisition. Evidence-acquisition tools retrieve evidence from $\mathcal{H}$ and are invoked by the LLM as needed. They include the following three operations.

- **Similarity scoring** takes a textual query and selects candidate records ranked by a combination of lexical and semantic relevance, which identifies useful records related to the node's objective.
- **Conditional filtering** restricts the search to records that satisfy specified metadata conditions, such as author, role, project phase, topic, and channel, so that record selection accounts for contextual conditions as well as relevance to the query.
- **Relation traversal** starts from a specified record and follows explicit links in a chosen direction to retrieve related records, such as replies, the record being replied to, or subsequent decision records, helping trace how a discussion developed and how its decisions changed.

During node execution, the LLM selects tools and specifies their arguments based on the node's objective and execution context to refine the search scope or follow links from previously identified records, rather than relying on textual similarity alone.

Memory Management. The memory-management tool maintains an accumulated execution memory $\mathcal{M}$ through read and write operations invoked by the framework. It stores task-specific information derived from user interactions together with intermediate results produced during execution. Depending on the task, memory is kept as structured records or fields or as a running summary of the records gathered and conclusions reached so far. Both representations are accessed and updated through the Read and Write operations described below.

Before node v_j executes, the read operation provides the available memory for constructing its execution context as

$$\mathcal{C}_j = \mathrm{Assemble}\left(\mathcal{C}_j^{\mathrm{base}}, \mathrm{Read}(\mathcal{M})\right), \tag{4}$$

where $\mathcal{C}_j^{\mathrm{base}}$ contains the node's objective, relevant user information, and relevant constraints from $\mathcal{R}$. Read returns the available memory content, including the outputs of the predecessors of v_j, and $\mathrm{Assemble}$ combines this information with the base context in the node's prompt template.

After v_j takes its action $a_j \in \mathcal{A}_t$, the write operation incorporates relevant information from the action and its corresponding observation o_j, into memory:

$$\mathcal{M} \leftarrow \mathrm{Write}(\mathcal{M}, a_j, o_j), \tag{5}$$

where, depending on the memory representation, Write updates structured records or fields with new information obtained during the current execution, or uses the LLM to revise the running summary.

5 Experiments

To comprehensively evaluate Org-Agent, we design experiments around four questions. **Q1:** How does Org-Agent compare with baseline methods in the capability of cross-user memory and knowledge use, as evaluated on GroupMemBench? **Q2:** How does Org-Agent compare with the vanilla baseline in the capability of cross-user interaction and decision-making, as evaluated on MUSES-Bench? **Q3:** What are the contributions of dependency-aware scheduling and auxiliary tools? **Q4:** How does performance change as the number of users increases? (Additional experimental evaluations of Org-Agent are presented in Appendix E.)

5.1 Experimental Setting

Datasets. We evaluate Org-Agent on two benchmarks covering the two complementary capabilities. GroupMemBench (Yang et al., 2026a) evaluates cross-user memory and knowledge use through questions grounded in multi-user interaction histories. It contains 745 questions across four domains, namely Finance, Healthcare, Manufacturing, and Technology, and six query categories, namely Multi-Hop, Update, Ambiguity, Implicit, Temporal, and Abstention. MUSES-Bench (Yang et al., 2026b) evaluates cross-user interaction and decision-making through instruction selection (Queue), instruction following (Instruct), cross-user information access (Cross-user Access), and meeting scheduling (Meeting). More details are provided in Appendix A.

Baselines. On GroupMemBench, we compare Org-Agent with two groups of baselines. (i) RAG-based methods include BM25 (Robertson & Zaragoza, 2009) for sparse lexical retrieval, text-embedding-3-large for dense semantic retrieval, and HippoRAG2 (Gutiérrez et al., 2025) and GraphRAG (Edge et al., 2024) for graph-based retrieval. (ii) Memory-based methods include MemGPT (Packer et al., 2023), LightMem (Fang et al., 2026), SimpleMem (Liu et al., 2026), and Hindsight (Latimer et al., 2025). On MUSES-Bench, we compare Org-Agent with the vanilla agent using the same backbone LLM, treating Org-Agent as a plug-and-play enhancement to the backbone for handling requests from multiple users.

Evaluation Metrics. On GroupMemBench, we report accuracy for each query category and overall accuracy across all questions, with answer correctness assessed by an LLM judge against reference answers. On MUSES-Bench, Queue F_1 measures the precision–recall balance between the predicted and reference sets of accepted instructions. Instruct accuracy measures the proportion of applicable instruction requirements satisfied by the generated responses, as determined by rule-based checkers. For cross-user access, Privacy measures the proportion of unauthorized users for whom no access violation is detected, while Utility measures the proportion of authorized users

Table 1: **Results (%) of baselines and ORG-AGENT on GroupMemBench across six query categories using GPT-4o-mini.** The best result for each category is highlighted in **bold**, while the second best is indicated with an underline. Avg. denotes the average performance.

Method	Multi-Hop	Update	Ambiguity	Implicit	Temporal	Abstention	Avg.
RAG-based Methods							
BM25	36.26	15.89	14.15	42.86	41.36	68.35	37.72
text-embedding-3-large	32.42	12.15	23.58	34.69	25.31	66.91	33.29
HippoRAG2	28.57	18.69	14.15	34.69	33.33	**79.86**	36.11
GraphRAG	8.24	0.93	6.60	14.29	1.85	78.42	19.06
Agent Memory Systems							
MemGPT	24.18	10.28	23.58	30.61	14.20	60.43	27.11
Hindsight	43.41	10.28	**33.96**	38.78	**44.44**	51.80	38.79
LightMem	6.04	0.00	5.66	6.12	3.70	72.66	17.05
SimpleMem	23.63	7.48	16.98	20.41	24.69	69.78	28.99
ORG-AGENT (Ours)	**49.45**	**28.97**	25.47	**48.98**	42.59	62.59	**44.03**

judged to have received access to the requested information. For meeting scheduling, success rate measures the proportion of scenarios in which the agent finalizes a schedule that accommodates all required participants within their stated availability windows.

Implementation Details. On GroupMemBench, all methods use GPT-4o-mini (Hurst et al., 2024) for their LLM-based components, while GPT-5.1 (Singh et al., 2025) serves as the judge model. On MUSES-Bench, we evaluate GPT-4o-mini, DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), and Qwen3-32B (Yang et al., 2025), where each backbone drives both the agent and the simulated users. Multi-turn tasks are limited to at most $T = 10$ turns. Further settings are given in Appendix B.

5.2 PERFORMANCE ON CROSS-USER MEMORY AND KNOWLEDGE USE (Q1)

To address Q1, we conduct a comprehensive comparison of various baseline methods with ORG-AGENT on GroupMemBench. The detailed experimental results are presented in Table 1. Based on our analysis, we derive the following key observations.

Obs. 1. Existing agent memory systems do not consistently outperform basic retrieval baselines on cross-user memory and knowledge use. Although these systems are designed to organize and reuse interaction history, most of them fall behind BM25 in average accuracy. Specifically, SimpleMem and MemGPT reach 28.99% and 27.11%, falling behind BM25 by 8.73 and 10.61 points, and LightMem drops to 17.05%. We analysis that compact memory representations omit contextual details needed to interpret and select relevant records in cross-user tasks. Moreover, although Hindsight relies on costly memory construction, it exceeds BM25 by only 1.07 points.

Obs. 2. ORG-AGENT demonstrates improved cross-user memory and knowledge utilization capabilities compared with both retrieval baselines and agent memory systems. On GroupMemBench, ORG-AGENT attains the best average accuracy among all compared methods. Specifically, it reaches 44.03%, exceeding BM25 and text-embedding-3-large by 6.31 and 10.74 points, and the strongest memory system, Hindsight, by 5.24 points. Taken together, these comparisons support the effectiveness of the approach across the two baseline groups. Notably, its gains do not require constructing a memory graph before the query is received.

5.3 PERFORMANCE ON CROSS-USER INTERACTION AND DECISION-MAKING (Q2)

To analyze the performance in cross-user interaction and decision-making scenarios, we compare ORG-AGENT with the vanilla baseline on MUSES-Bench across four tasks and three backbones. The results in Table 2 yield the following observations.

Table 2: **Results (%) of Vanilla and ORG-AGENT on MUSES-Bench across three backbones.** The best result for each metric within each backbone is shown in **bold**. Avg. denotes the average performance. $\Delta\uparrow$ denotes the difference between ORG-AGENT and Vanilla in percentage points.

Model	Method	Multi-user Instruction		Cross-user Access		Meeting	Avg.
		Queue (F_1)	Instruct (Acc.)	Privacy	Utility	Success Rate	
GPT-4o-mini	Vanilla	62.97	58.69	96.94	**64.96**	37.04	64.12
	ORG-AGENT (Ours)	**74.24**	**73.41**	**97.59**	56.52	**60.19**	**72.39**
	$\Delta\uparrow$	+11.27	+14.72	+0.65	−8.44	+23.15	+8.27
DeepSeek-V4.1-Flash	Vanilla	78.16	81.18	71.26	97.35	75.93	80.78
	ORG-AGENT (Ours)	**88.21**	**83.78**	**75.25**	**99.17**	**78.70**	**85.02**
	$\Delta\uparrow$	+10.05	+2.60	+3.99	+1.82	+2.77	+4.24
Qwen3-32B	Vanilla	62.38	63.46	**64.28**	91.10	36.11	63.47
	ORG-AGENT (Ours)	**64.67**	**81.77**	47.45	**98.05**	**51.85**	**68.76**
	$\Delta\uparrow$	+2.29	+18.31	−16.83	+6.95	+15.74	+5.29

Obs. 3. ORG-AGENT improves cross-user interaction and decision-making capability over the vanilla baseline. On MUSES-Bench, ORG-AGENT raises the average score with every backbone we evaluate. Specifically, with GPT-4o-mini it reaches 72.39% on average, exceeding the vanilla baseline by 8.27 points, with Instruct accuracy rising from 58.69% to 73.41% and Meeting success rate from 37.04% to 60.19%. The largest gains appear on Meeting, reaching 23.15 percentage points with GPT-4o-mini, as this task most clearly illustrates how constraints of organizational tasks affect task completion through interactions among users over multiple turns.

Obs. 4. The improvements hold across backbones of different capabilities. ORG-AGENT improves the average score with all three backbones by 8.27 points with GPT-4o-mini, 4.24 points with DeepSeek-V4.1-Flash, and 5.29 points with Qwen3-32B. The gain is larger for the two weaker backbones, whose vanilla scores are around 64, than for DeepSeek-V4.1-Flash, whose vanilla score is 80.8, which suggests that explicit constraints handling compensates for weaker reasoning in the backbone. Moreover, these backbones span proprietary and open-source models at different scales, supporting the generality of ORG-AGENT as a model-agnostic framework.

5.4 ABLATION STUDIES (Q3)

To address Q3, we conduct ablation studies on the two core components of ORG-AGENT across GroupMemBench and MUSES-Bench, using GPT-4o-mini as the backbone in all cases. We consider two variants: (i) *w/o Dependencies* removes all edges of the TDG, so that every subtask is executed independently without topological order. (ii) *w/o Tools* keeps the TDG and its execution order but disables conditional filtering, relation traversal, and memory management, so that each node retrieves records by similarity scoring alone and receives the unprocessed results of earlier nodes instead of the organized execution memory. Results on both benchmarks are reported in Figure 3.

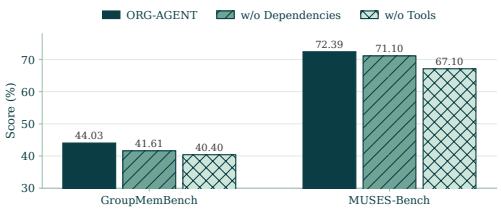


Figure 3: **Ablation study on key components of ORG-AGENT across GroupMemBench and MUSES-Bench with GPT-4o-mini.** The y-axis represents the average score of the full model and its two variants on each benchmark.

Obs. 5. Both dependency-aware scheduling and constraint-aware node execution contribute to the effectiveness of ORG-AGENT. Removing either component reduces the average score on both benchmarks. On GroupMemBench, the average score drops from 44.03% to 41.61% without dependencies and to 40.40% without tools. On MUSES-Bench, it decreases from 72.39% to 71.10% and 67.10%, respectively.

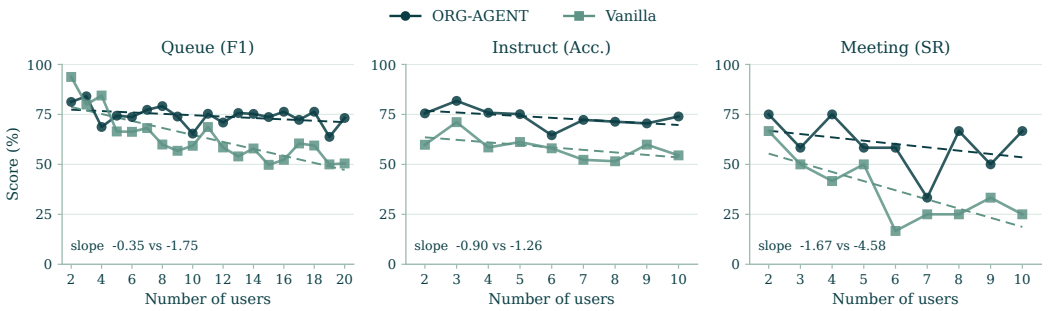


Figure 4: **Performance across different user group sizes on MUSES-Bench using GPT-4o-mini.** The x-axis represents the number of users, and the y-axis represents the metric of each task. Dashed lines show a least-squares trend weighted by the number of scenarios at each group size.

These results highlight the contributions of dependency-aware scheduling and tool-supported node execution to performance on multi-user tasks.

5.5 Performance across Different User Group Sizes (Q4)

To address this question, we examine how the size of the user group affects performance on MUSES-Bench. We report evaluation results for three representative tasks, namely Queue, Instruct, and Meeting. Using GPT-4o-mini as the backbone for both ORG-AGENT and the vanilla baseline, we then compare their scores as the number of users grows.

Obs. 6. ORG-AGENT degrades more slowly than the vanilla baseline as the number of users grows. Both methods decline as groups become larger, but at different rates, shown by the dashed trends in Figure 4. The dashed lines are fitted by least squares, weighted by the number of scenarios at each group size. Specifically, ORG-AGENT loses 0.35 points per additional user on Queue and 1.67 on Meeting, whereas the vanilla baseline loses 1.75 and 4.58. Instruct follows the same pattern with a smaller difference, 0.90 against 1.26. The fitted trends therefore indicate a widening performance gap as the number of users increases within the evaluated range. As the user group grows, the agent must account for requests and constraints involving more users. The slower performance decline suggests that ORG-AGENT is better able to handle this increased coordination burden within the evaluated range of group sizes.

6 Conclusion

We presented ORG-AGENT, a unified framework for agents serving multiple users in organizational environments. It addresses two complementary capabilities, namely cross-user interaction and decision-making, as well as cross-user memory and knowledge use. Both are governed by organizational constraints on users, information, and joint decisions that shape the prerequisites of an action and the conditions that must be satisfied during its execution. ORG-AGENT makes the former explicit in a Task Dependency Graph that is updated as needed when new user inputs arrive, schedules the subtasks in dependency order by topological sorting, and accounts for the latter during each node's execution, with support from evidence-acquisition and memory-management tools. Experiments on GroupMemBench and MUSES-Bench demonstrate the effectiveness of this approach on both capabilities. On GroupMemBench, ORG-AGENT improves overall accuracy over the baselines. On MUSES-Bench, it improves average performance across all three evaluated backbones. Ablation studies support the contributions of dependency modeling and tool-supported execution, while analyses of different user group sizes show slower performance degradation than the vanilla agent as the number of users grows in the evaluated settings. These findings highlight the value of organizing task execution around organizational constraints, and suggest a path for LLM agents to move beyond personal assistants toward organizational agents.

Reproducibility Statement

All code and relevant resources can be accessed from the anonymous repository referenced in the abstract. A thorough README document within the repository supplies full guidance to reproduce our experimental setups and outcomes.

AI Use Statement

We used AI tools to assist with grammar checking and language refinement. The authors take full responsibility for the accuracy and integrity of the final manuscript.

References

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026. URL https://arxiv.org/abs/2609.19969.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515. PMLR, 2025.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, Henry Peng Zou, Wanjia Zhao, Song Wang, Wujiang Xu, Zixuan Ke, Zheng Hui, Dawei Li, Yaozu Wu, Langzhou He, Chen Wang, Xiongxiao Xu, Baixiang Huang, Juntao Tan, Shelby Heinecke, Huan Wang, Caiming Xiong, Ahmed Metwally, Jun Yan, Chen-Yu Lee, Hanqing Zeng, Yinglong Xia, Xiaokai Wei, Ali Payani, Yu Wang, Haitong Ma, Wenya Wang, Chenguang Wang, Yu Zhang, Xin Eric Wang, Yongfeng Zhang, Jiaxuan You, Hanghang Tong, Xiao Luo, Xue Liu, Yizhou Sun, Wei Wang, Julian McAuley, James Zou, Jiawei Han, Philip S. Yu, and Kai Shu. A survey of agent memory in the second half: Towards self-evolving and long-horizon agents. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=XycbogUAeJ. Survey Certification.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Harsh Jhamtani, Jacob Andreas, and Benjamin Van Durme. LLM agents for coordinating multi-user information gathering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17800–17826, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.916. URL https://aclanthology.org/2025.findings-acl.916/.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *arXiv preprint arXiv:2512.12818*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for LLM agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=oBgLvd5YC6.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Veronika Thost and Jie Chen. Directed acyclic graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=JbuYF437WB6.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0315.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jingbo Yang, Kwei-Herng Lai, Xiaowen Wang, Shiyu Chang, Yaar Harari, and Evgeniy Gabrilovich. Groupmembench: Benchmarking llm agent memory in multi-party conversations. *arXiv preprint arXiv:2605.14498*, 2026a.

Shu Yang, Shenzhe Zhu, Hao Zhu, José Ramón Enríquez, Di Wang, Alex Pentland, Michiel A Bakker, and Jiaxin Pei. Multi-user large language model agents. *arXiv preprint arXiv:2604.08567*, 2026b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025.

Hanzhang Zhou, Panrong Tong, Xu Zhang, Quyu Kong, Chenglin Cai, Tianyu Xia, Gongjie Zhang, Jianan Zhang, Long Li, Long Chen, et al. Qwen-ui-agent technical report: Toward next-generation real-world centric foundation gui agents. *arXiv preprint arXiv:2607.28227*, 2026.

A DATASETS

We conduct our experiments on two benchmarks corresponding to the two capabilities studied in this paper. GroupMemBench evaluates cross-user memory and knowledge use through question answering over multi-user conversations, while MUSES-Bench evaluates cross-user interaction and decision-making through tasks involving a shared agent and multiple users. Our evaluation includes 745 questions from GroupMemBench and 1,183 scenarios from MUSES-Bench. For MUSES-Bench, we evaluate four tasks and use the partial-disclosure subset for Meeting, with group sizes ranging from 2 to 20 users across tasks. Table 3 summarizes the statistics of both benchmarks.

GroupMemBench provides conversation histories annotated with user identities and timestamps, with each question associated with an asking user. The questions cover six categories. *Multi-Hop* requires linking evidence across multiple messages or threads. *Update* tests the tracking of revised facts and decisions, including their current values, previous values, or reasons for change. *Ambiguity* requires interpreting role-dependent terminology and connecting expressions used by different speakers. *Implicit* requires resolving first-person references using the asking user's identity and history. *Temporal* tests reasoning about event timing and ordering using timestamps and relative time expressions. *Abstention* requires recognizing that the requested information is absent from the history rather than fabricating an answer. These categories contain 182, 107, 106, 49, 162, and 139 questions, respectively.

Table 3: **Statistics of the benchmarks in evaluation.** Samples correspond to questions for GroupMemBench and scenarios for MUSES-Bench.

Benchmark	Category	# Samples
GroupMemBench	Multi-Hop	182
	Update	107
	Ambiguity	106
	Implicit	49
	Temporal	162
	Abstention	139
	Total	**745**
MUSES-Bench	Queue	304
	Instruct	555
	Cross-user Access	216
	Meeting	108
	Total	**1,183**

MUSES-Bench evaluates a shared agent under task-specific objectives, authority rules, and information-access constraints. *Queue* contains 304 scenarios with 2 to 20 users. The agent must select which instructions to accept according to their alignment with the global objective and the users' authority hierarchy, rejecting instructions that violate the objective or lose a conflict under these rules. *Instruct* contains 555 scenarios with 2 to 10 users, comprising 187 aligned and 368 conflict scenarios. User requests include automatically verifiable requirements, such as length and wording constraints. In aligned scenarios, the agent should satisfy all users' requirements; in conflict scenarios, evaluation focuses on those issued by the highest-authority users. *Cross-user Access* contains 216 scenarios with 2 to 10 users. Each scenario specifies a protected resource and an authorized-user set. The agent must support authorized access while preventing unauthorized disclosure, with performance evaluated through Privacy and Utility scores. *Meeting* uses 108 partial-disclosure scenarios with 2 to 10 users. Participants have preferred and secondary availability, which may need to be elicited through interaction. The agent must gather the necessary information, negotiate scheduling conflicts, and finalize a time slot that all required participants can attend.

B IMPLEMENTATION DETAILS

Table 4: Hardware configuration of our local experimental server.

Component	Specification
GPU	4× NVIDIA GeForce RTX 4090 (24 GB each)
	4× NVIDIA GeForce RTX 4090 D (24 GB each)
CPU	2× Intel Xeon Gold 6426Y (16 cores each)

Hardware. The hardware configuration of our local experimental server is summarized in Table 4.

Prompting settings. On GroupMemBench, all baselines use the same question-answering prompt template to ensure a consistent evaluation setup. On MUSES-Bench, all methods share the same task-specific system instructions and user-simulation prompts.

C Details of Baselines

We provide additional descriptions of the baseline methods used in our experiments. On GroupMemBench, the baselines cover RAG-based methods and agent memory systems, representing different approaches to accessing and organizing information from interaction histories. On MUSES-Bench, we compare against a vanilla agent with the same backbone LLM to examine the contribution of the proposed framework beyond the backbone's existing capabilities.

C.1 RAG-Based Methods

Retrieval-based methods identify information relevant to a question and provide it as context for answer generation (Lewis et al., 2020). The selected baselines cover sparse lexical retrieval, dense semantic retrieval, and retrieval over graph-structured representations.

BM25. BM25 (Robertson & Zaragoza, 2009) is a sparse retrieval method that ranks records according to their lexical relevance to a query. Its scoring function combines term frequency, inverse document frequency, and document-length normalization.

text-embedding-3-large. This baseline uses text-embedding-3-large to encode queries and historical records into dense vector representations. Records are ranked by similarity in the embedding space and supplied as evidence for answering the question.

HippoRAG 2. HippoRAG 2 (Gutiérrez et al., 2025) combines graph-based retrieval with dense representations to support access to interconnected information. It builds on Personalized PageRank and incorporates passage information into the retrieval structure, while using an LLM during query processing to improve evidence selection. This baseline evaluates whether graph-based associations can help recover information distributed across multiple records, beyond what can be obtained through direct lexical or semantic matching alone.

GraphRAG. GraphRAG (Edge et al., 2024) uses an LLM to extract entities and relationships from source documents and organize them into a graph index. It also constructs summaries of graph communities to support answering questions over related information.

C.2 Agent Memory Systems

Agent memory systems maintain representations derived from previous interactions and make this information available for subsequent queries. The selected methods differ in how they manage memory, consolidate information, and retrieve relevant content.

MemGPT. MemGPT (Packer et al., 2023) introduces a hierarchical memory-management architecture inspired by operating systems. It manages information across memory tiers and transfers relevant content into the LLM's limited context window when needed. This design supports interactions whose accumulated history exceeds the available context.

LightMem. LightMem (Fang et al., 2026) is a memory framework that separates online information processing from offline consolidation. It first filters and groups incoming information, then organizes related content into compact memory representations. An offline update process further consolidates long-term memory without placing all maintenance operations on the inference path.

SimpleMem. SimpleMem (Liu et al., 2026) constructs compact memory units through semantic structured compression and integrates related information through online semantic synthesis, merging related context within a session to reduce redundancy. At query time, its intent-aware retrieval planning determines what information to retrieve and how broadly to search.

Hindsight. Hindsight (Latimer et al., 2025) organizes memory into distinct logical networks for world facts, agent experiences, entity summaries, and evolving beliefs. Its retain, recall, and reflect operations support information ingestion, retrieval, and reasoning over the resulting memory bank. The architecture incorporates temporal and entity information to support the interpretation of accumulated knowledge. It provides a structured-memory baseline for evaluating questions that require connecting and distinguishing information across users and interactions.

C.3 VANILLA AGENT ON MUSES-BENCH

The vanilla baseline uses the backbone LLM within the benchmark's original interaction protocol, without the Task Dependency Graph or the execution tools introduced by ORG-AGENT. It receives the task instructions and user inputs and generates responses or decisions directly. For multi-turn tasks, it can continue interacting with users through the benchmark environment rather than being restricted to a single response. We compare the vanilla agent and ORG-AGENT using the same backbone LLM under each of the three evaluated backbones.

D TASK-SPECIFIC NODE EXAMPLES

The content of a node depends on the task being performed. On GroupMemBench, a node may retrieve records relevant to a user's question. On MUSES-Bench, nodes may evaluate an instruction, generate a response, handle an information-access request, or ask a participant for confirmation. Table 5 illustrates these task-specific node objectives. The examples describe individual subtasks rather than complete task workflows.

Table 5: **Examples of task-specific nodes in ORG-AGENT across GroupMemBench and MUSES-Bench.** Each node represents a subtask with a concrete objective.

Benchmark / Task	Example node
GroupMemBench	Find User_13's decision about the UAT validation scope
MUSES-Bench: Queue	Decide whether to accept Alice's instruction to prioritize urgent requests
MUSES-Bench: Instruct	Respond to Alice in English using at most three sentences
MUSES-Bench: Cross-user Access	Handle Bob's request for the project budget
MUSES-Bench: Meeting	Ask David whether he can attend on Friday at 14:00

E ADDITIONAL EXPERIMENTS

To further analyze ORG-AGENT, we conduct additional experiments and case studies addressing four questions. **Q5:** Does ORG-AGENT remain effective on GroupMemBench with a different backbone model? **Q6:** How sensitive is its performance to the hybrid-retrieval weight α that balances lexical and semantic relevance? **Q7:** How much LLM cost does ORG-AGENT incur compared with the baselines, and how does this cost relate to accuracy? **Q8:** How do constraint-aware node execution and dependency-aware scheduling operate in concrete multi-user cases?

E.1 EFFECTIVENESS OF DIFFERENT BACKBONES (Q5)

In this section, we study the effectiveness of ORG-AGENT across different model backbones. Since the main experiments on GroupMemBench use GPT-4o-mini, we additionally evaluate ORG-AGENT and the baselines with Qwen3-8B as the backbone model to assess whether its performance advantage over these baseline methods extends to another backbone.

Obs. 7. ORG-AGENT remains effective with a different model backbone. As shown in Table 6, ORG-AGENT achieves an overall accuracy of 40.40% with Qwen3-8B, outperforming the strongest

Table 6: **Results (%) of baselines and ORG-AGENT on GroupMemBench across six query categories using Qwen3-8B.** The best result for each category is highlighted in **bold**, while the second best is indicated with an underline. Avg. denotes the average performance.

Method	Multi-Hop	Update	Ambiguity	Implicit	Temporal	Abstention	Avg.
RAG-based Methods							
BM25	36.81	**28.04**	18.87	36.73	38.89	41.73	34.36
text-embedding-3-large	29.12	22.43	22.64	42.86	17.28	33.09	26.31
HippoRAG2	29.67	23.36	17.92	38.78	24.07	**64.03**	32.89
GraphRAG	7.69	0.93	5.66	6.12	4.32	51.80	13.83
Agent Memory Systems							
MemGPT	20.33	14.95	20.75	32.65	10.49	35.97	21.21
Hindsight	34.07	11.21	28.30	**48.98**	38.27	38.85	32.75
LightMem	16.48	3.74	15.09	14.29	13.58	40.29	18.12
SimpleMem	25.27	14.02	13.21	14.29	29.01	46.04	25.91
ORG-AGENT (Ours)	**39.01**	25.23	**34.91**	34.69	**39.51**	61.15	**40.40**

baseline, BM25, by 6.04 percentage points and the strongest memory system, Hindsight, by 7.65 percentage points. These results show that its overall advantage is maintained with Qwen3-8B, providing evidence that the effectiveness of the framework is not limited to GPT-4o-mini.

E.2 HYPER-PARAMETER SENSITIVITY (Q6)

In this section, we examine the sensitivity of ORG-AGENT to the weight α in the similarity scoring tool introduced in Section 4.4, evaluated on GroupMemBench using GPT-4o-mini. This tool ranks candidate records by combining BM25 lexical relevance and dense semantic similarity computed with text-embedding-3-large, and α controls their relative contribution. For a retrieval query x and a candidate record h in the interaction history $\mathcal{H}$, the combined score is defined as

$$\text{Score}(x,h) = (1-\alpha)\,\widetilde{s}_{\text{BM25}}(x,h) + \alpha\,\widetilde{s}_{\text{dense}}(x,h), \tag{6}$$

where $\widetilde{s}_{\text{BM25}}$ and $\widetilde{s}_{\text{dense}}$ denote the BM25 score and the dense cosine similarity, respectively, each min-max normalized over the current candidate set. Thus, $\alpha = 0$ ranks records by lexical relevance only, while $\alpha = 1$ ranks them by semantic similarity only. Figure 5 reports the average score for $\alpha \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$.

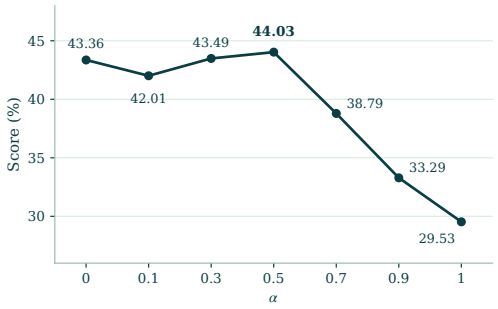


Figure 5: **Effect of the hybrid-retrieval weight α on GroupMemBench with GPT-4o-mini.** The x-axis is α, and the y-axis is the weighted average accuracy.

Obs. 8. Balanced lexical and semantic weighting achieves the best performance among the evaluated settings. ORG-AGENT achieves its highest average score at $\alpha = 0.5$, while larger dense weights lead to lower performance. Specifically, the average score reaches 44.03 at $\alpha = 0.5$, compared with 43.36 at $\alpha = 0$ and 29.53 at $\alpha = 1$. These results suggest that lexical relevance remains important for evidence selection in this setting, supporting our default choice of $\alpha = 0.5$.

E.3 COST ANALYSIS (Q7)

To address Q7, we measure the LLM tokens consumed by ORG-AGENT and the baselines on GroupMemBench using GPT-4o-mini, covering both the build stage and the question-answering stage. Figure 6 plots the number of tokens over against accuracy.

Obs. 9. ORG-AGENT delivers the highest accuracy at moderate LLM cost. As shown in Figure 6, ORG-AGENT achieves the highest accuracy of 44.03% while consuming only 5.5M tokens, which is 3.9 times fewer than LightMem (21.6M) and 21.4 times fewer than the strongest baseline, Hindsight (117.5M). Although BM25 is cheaper at 1.2M tokens, it reaches only 37.72% because it selects records by lexical matching alone. Among the remaining baselines, higher accuracy generally comes at a higher cost, as HippoRAG2 and Hindsight consume more than 100M tokens to reach 36.11% and 38.79%, respectively. Yet more tokens do not guarantee better accuracy, since GraphRAG consumes 62.8M tokens but reaches only 19.06%, below SimpleMem with 35.7M tokens. ORG-AGENT departs from this trend because it does not construct memories or graphs over the entire history in advance, and instead resolves each question through a Task Dependency Graph at query time.

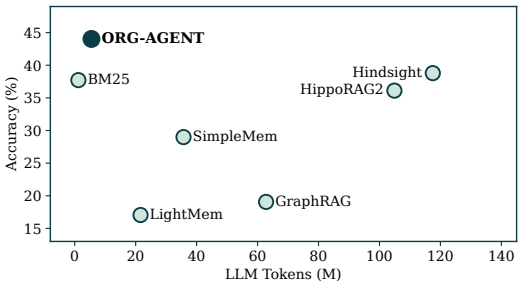


Figure 6: **LLM cost and accuracy on GroupMemBench with GPT-4o-mini.** The x-axis is the number of LLM tokens consumed in the build and question-answering stages, and the y-axis is the average accuracy.

E.4 CASE STUDY (Q8)

To illustrate how ORG-AGENT behaves on multi-user tasks, we examine two cases in Figure 7. The first case, from GroupMemBench, shows how constraint-aware node execution uses the asking user's identity to guide record selection through tools. The second case, from MUSES-Bench, shows how dependency-aware scheduling orders interactions with different users.

Obs. 10. Constraint-aware node execution uses the asking user's identity to select relevant records. In Figure 7 (a), User_13 asks which decision they are trying to finalize about the UAT scope in the Risk: Calculation Discrepancy phase. The history contains two decisions in this phase from users with the same role, the validation scope stated by User_13 and the release timing stated by User_12. The BM25-based baseline ranks the record of User_12 higher because of its lexical overlap with the question and answers with that decision, although the question concerns User_13's own decision. ORG-AGENT records the asking user's identity as a node attribute and uses it to specify conditional filtering with `author = User_13` and the phase name, so only records from User_13 in this phase enter the candidate set, and the answer is the validation-scope decision. This case shows that ORG-AGENT accounts for the asking user's identity and information attribution when selecting records from the relevant project phase.

Obs. 11. Dependency-aware scheduling resolves a required participant's availability before wider coordination. In Figure 7 (b), the task is to schedule a meeting for multiple users whose attendance status is not given to the agent in advance. The vanilla baseline pursues Wednesday at 15:00, which most participants support, and finalizes it although Eve has repeatedly stated that she cannot attend, so the schedule excludes a required participant and fails the task. ORG-AGENT first asks the participants and learns that Eve and David are required, and that David lists Friday at 14:00 as a backup slot. It then places the interaction with David before those with the other participants to clarify whether he can use Friday at 14:00, since his confirmation is a prerequisite of proposing that slot. After David replies that he can attend if necessary, the TDG is updated to reflect the satisfied prerequisite, the remaining confirmation requests can proceed in parallel, and the meeting is finalized for Friday at 14:00 with both required participants. This case shows that ORG-AGENT obtains prerequisite confirmations before the interactions that depend on them.

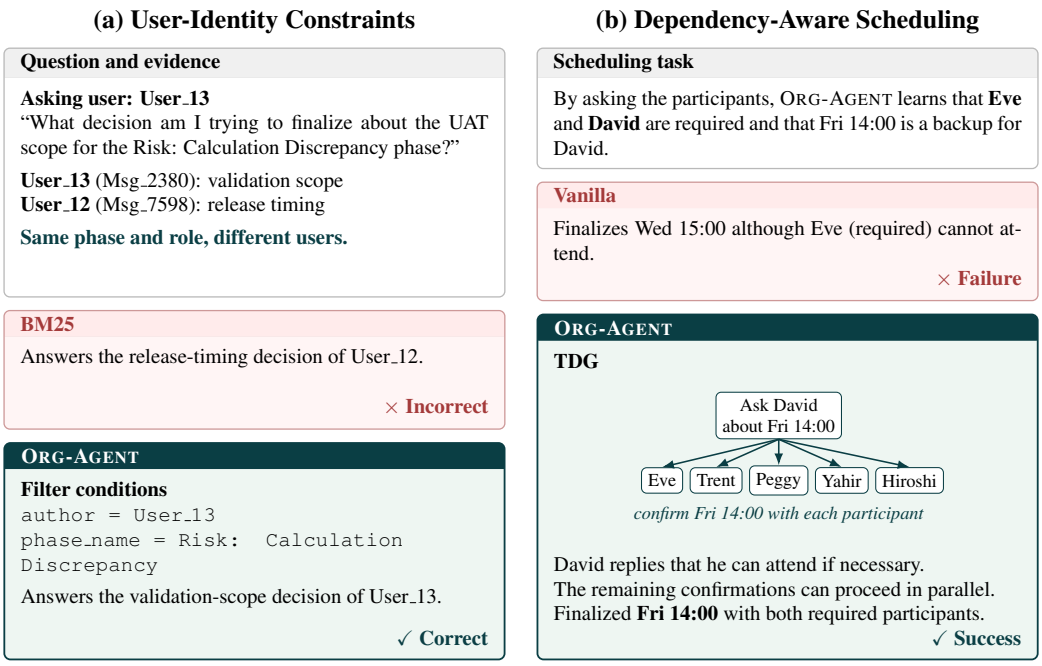


Figure 7: **Case studies on GroupMemBench and MUSES-Bench.** (a) ORG-AGENT identifies the asking user's decision rather than another user's related decision by filtering records according to the user's identity and the relevant project phase. (b) With David identified as a required participant, the ORG-AGENT queries him before proposing the slot to the others, and his confirmation allows the remaining confirmation requests to proceed.