MemSlides: A Hierarchical Memory Driven Agent Framework for Personalized Slide Generation with Multi-turn Local Revision

Ye Jin
Beijing University of Posts and Telecommunications
13681596382@bupt.edu.cn

Yangyang Xu
Tsinghua University
yangyangxu@mail.tsinghua.edu.cn

Jun Zhu
Tsinghua University
dcszj@tsinghua.edu.cn

Yibo Yang*
Shanghai Jiao Tong University
yibo.yang93@gmail.com

Code Website Project Page Video

Abstract

Personalized presentation generation requires more than conditioning on a current prompt or template: agents must preserve stable user preferences across tasks, retain newly introduced preferences and constraints during multi-turn revision, and carry out local edits reliably. We propose MemSlides, a hierarchical memory framework for personalized presentation agents that separates long-term memory from working memory and further divides long-term memory into user profile memory and tool memory. User profile memory stores intent-conditioned profiles for round-0 personalization, working memory carries active preferences and session constraints across revision rounds, and tool memory stores reusable execution experience for reliable localized editing. MemSlides pairs this memory design with scoped slide-local revision, so targeted updates act on the smallest affected region instead of repeatedly regenerating the full deck. In controlled experiments, user profile memory improves persona-alignment judgments on a multi-persona, multi-intent profile bank, tool-memory injection improves closed-loop modify behavior in diagnostic matched-pair settings, and qualitative cases illustrate working memory's ability to carryover preferences. Taken together, these results suggest that effective personalization in presentation authoring depends on separating persistent user profiles, session-level working memory, and reusable execution experience across generation and localized revision.

1 Introduction

Automatic presentation generation aims to turn user requests into structured slide decks, and has gained increasing attention because slides are widely used, yet creating high-quality presentations remains time-consuming and cognitively demanding [40, 25, 26, 55]. Recent agentic systems further advance this task by producing complete decks through multi-modal or tool-based workflows [6, 58, 51, 49, 21, 59, 30]. Although these systems can now produce complete and visually polished decks, they still lack persistent personalization, which is essential for generating ready-to-use slide decks because users vary in domain, purpose, style, and presentation habits. For example, users may prefer different layouts, templates, and styles when creating slides for different purposes, such as academic presentations versus business presentations. An effective personalized slide generation

*Corresponding author.

Preprint.

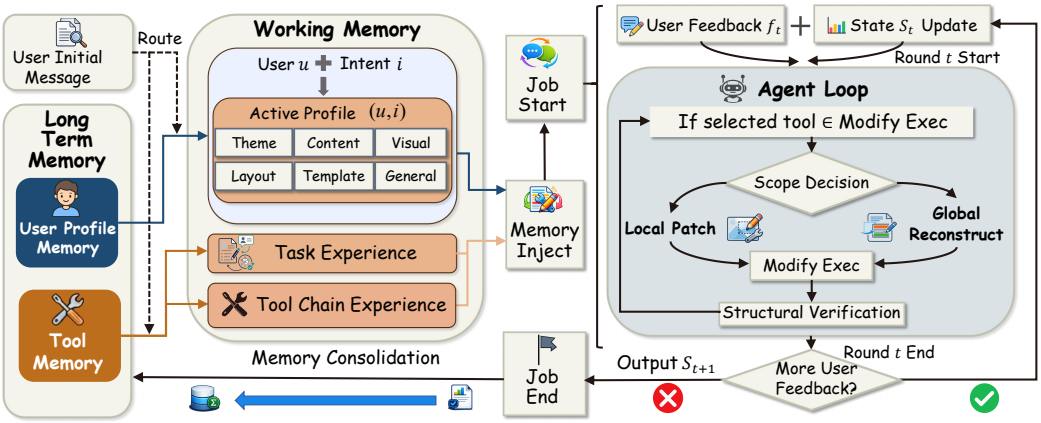


Figure 1: Overview of MemSlides. The framework comprises long-term memory and working memory. Long-term memory stores user profile memory and tool memory for persistent personalization and reusable execution experience, while working memory carries the current session state for personalized generation and localized revision. In round t, s_t is the current session state, f_t is the user feedback, and s_{t+1} is the updated state after *Modify Exec*, the memory-guided revision executor.

framework should build and maintain user profiles that capture users' long-term preferences in organizing, styling, and revising presentations across different intents, rather than requiring users to repeatedly specify their preferences in every interaction.

Prior work has progressively expanded the capability of presentation-generation systems [40, 26, 2, 25, 6, 51, 49, 21, 30]. PPTAgent moves beyond text-to-slides by coupling generation with presentation-specific evaluation [58], and DeepPresenter introduces environment-grounded reflection for agentic presentation generation [59]. These systems improve general generation and agentic refinement, but do not explicitly model user-specific personalization. SlideTailor addresses personalization by conditioning scientific slide generation on reference slides and task-time templates [55], but its personalization remains tied to provided examples or template conditions rather than to an accumulated user profile. These observations therefore expose a central gap: slide generation agents still lack a user-facing multi-turn dialogue process that converts revision requests into reusable preference memory and preserves local revision constraints across later turns.

This gap has two roots. First, personalization in presentation authoring is often revealed through revision rather than fully specified before generation, yet existing slide generation agents still handle edits by re-contextualizing or re-generating large parts of the deck. As a result, small changes must compete with deck state and feedback history for limited context, making multi-turn local modification fragile. Second, current systems mainly improve presentation-agent workflows, tools, and evaluators, but still treat personalization as an implicit byproduct of prompting rather than a direct service enabled by memory design. In a similar spirit to agent memory work [61, 31, 47], personalization of slide generation can be largely enhanced by incorporating memory as an explicit framework rather than an undifferentiated dialogue buffer.

To address these problems, in this paper, we present MemSlides that introduces scoped slide locality as its revision strategy for multi-turn slide editing. Rather than re-reading or re-writing the whole deck for each feedback turn, MemSlides projects the request onto the smallest affected slide region and operates on a bounded repair surface. It reads only a structured snapshot of that local surface, including its local layout structure, available selectors, and exposed style rules, and writes back only patches scoped to explicit selectors or to those style rules. In this way, both reading and writing stay local by construction, reducing context pressure and unintended drift while preserving already aligned content across turns.

Building on this modification strategy, MemSlides is further integrated with a hierarchy memory framework for personalized presentation generation. The framework has two levels: long-term memory and working memory. Working memory maintains the current session state and temporary feedback across revision rounds, so later local edits can preserve active temporary memory from the same deck. Long-term memory persists across jobs and is further divided into user profile memory

and tool memory. User profile memory is user-specific and intent-aware: for each user and each intent, preferences are organized by various dimensions including theme, content, visual, layout, template, and general, and are routed into working memory when each job starts. Tool memory captures reusable execution experience for later edits. This hierarchy memory framework enables generating and revising presentations according to both persistent user preferences and the active intent of the current session. The overview of MemSlides is shown in Figure 1.

In experiments, we evaluate MemSlides across multiple dimensions including slide deck quality, instruction satisfaction, and preference alignment. We develop *persona-alignment judgments* as metrics to evaluate personalization alignment. Qualitative comparisons further illustrate the effectiveness of MemSlides in aligning with users' preferences.

Our contributions are threefold:

- We introduce MemSlides, a personalized presentation agent that supports multi-turn localized revision. By maintaining session state and applying targeted slide-level updates instead of repeated full-deck regeneration, MemSlides provides the interaction substrate needed for learning user preferences from revision feedback.
- We further develop a hierarchical memory framework for MemSlides consisting of a long-term memory and a working memory. The long-term memory contains user profile memory and tool memory, allowing stable user preferences and reusable execution experience to persist across jobs, while working memory tracks session-specific constraints.
- We construct a multi-persona, multi-intent user profile bank for personalized presentation generation evaluation. Experiments show that user profile memory improves round-0 persona alignment, tool memory enhances localized modify reliability in diagnostic matched-pair comparisons, and working memory supports session-level preference carryover.

2 Related Work

Slide generation. Presentation generation has progressed from document compression and structured summarization to LLM-based systems that emphasize audience adaptation, editability, task-time preference inference, and visual refinement [40, 26, 58, 55, 59]. Presentation authoring also draws on controllable layout and design generation, including code-like layout representations, in-context layout prompting, layered or diffusion-based layout modeling, and visual preference modeling [41, 22, 9, 56, 34]. These works improve slide quality, editability, task-level controllability, and visual composition. Our focus is complementary: how user-specific preference and execution memory should persist across slide-generation and revision sessions rather than being supplied only as current-task inputs.

Memory and tool-using agents. Retrieval-augmented and external-memory language models show that stored context can support generation [16, 7, 14, 3, 10, 43]. These studies focus on persistent memory, reflection, structured updates, and long-term/short-term memory management [61, 31, 32, 50, 4, 45, 12, 47, 54]. Tool-using and reflective agents further establish patterns for interleaving reasoning with actions, using APIs or modular tools, coordinating execution, and learning from feedback [53, 39, 38, 33, 36, 18, 13, 46, 42, 24, 28, 1, 44, 52, 8]. In contrast, MemSlides targets personalized presentation authoring, where memory must distinguish user preference from execution experience and preserve the intended local edit scope during multi-turn revision.

Personalized generation and evaluation. Personalized generation has evolved from explicit persona conditioning to profile- and history-aware generation [17, 48, 37, 27, 11]. Recent surveys and alignment work characterize personalization as an agentic, retrieval-aware, and preference-sensitive problem [57, 20]. In visual domains, personalized visualization recommendation and DesignPref show that persistent expressive or design preferences can be learned from user history [35, 34]. In PPT generation, Persona-Aware-D2S and SlideTailor personalize slides through audience specifications, examples, or templates provided for the current task, whereas we study preferences accumulated across tasks and retained during multi-turn revision. Our evaluation follows rubric-based and pairwise LLM-as-judge protocols in a controlled memory/no-memory setting [23, 60, 15, 62, 19].

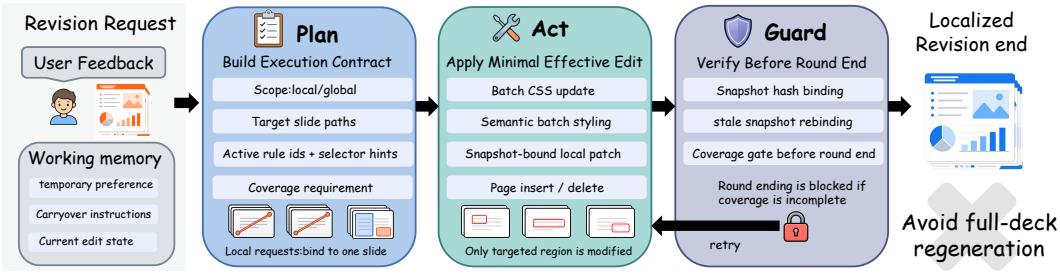


Figure 2: Localized modify execution in MemSlides. Working memory supplies active temporary preferences, carryover instructions, current edit state, and buffered tool-memory signals to the Plan–Act–Guard pipeline. Plan builds an execution contract, Act applies the minimal effective edit, and Guard verifies coverage before finalizing the localized update.

3 MemSlides

3.1 Problem Formulation

We formulate personalized presentation generation as a stateful, multi-turn authoring problem rather than a one-shot source-to-slides conversion task [40, 58, 55, 59]. Given source material x, a user profile memory P_u, and an optional task-time template τ, the system first produces an initial deck:

$$S_0 = G_{\text{init}}(x, P_u, \tau). \tag{1}$$

We refer to this initial generation stage as *round-0*. After receiving user feedback f_t at revision round t, the system updates a session state z_t and edits the current deck accordingly:

$$z_t = U(z_{t-1}, f_t; S_{t-1}), \qquad S_t = G_{\text{edit}}(S_{t-1}, x, P_u, \tau, z_t), \quad t \geq 1. \tag{2}$$

Here, z_t stores feedback-derived constraints and edit intentions that are active within the current authoring session. The objective is not only to improve the first draft, but also to move the deck toward the user's intended presentation while preserving slides that are already well aligned.

This formulation separates three personalization signals with different lifetimes. The user profile memory P_u captures recurring cross-job preferences, such as preferred themes, visual styles, layout habits, content density, and general design conventions. The task-time template τ, when provided, serves as a deck-local design constraint for the current job. The session state z_t captures temporary, turn-specific requirements, such as changing the color of new slide titles or compressing text after a particular section. When these signals conflict, explicit session feedback takes precedence for the current deck, followed by the task-time template and then the user profile memory.

The temporal structure also determines the operational scope of revision. We define a *job* as one complete deck-authoring session, a *round* (turn) as one user-triggered revision within that job, and an *operation* as a concrete edit or tool call. Revision requests may be local to a single slide, propagate across multiple slides, or modify the deck structure itself. Therefore, repeatedly regenerating the full deck is undesirable: it can overwrite already aligned content, introduce unnecessary drift, and increase context pressure. A stateful revision system should instead select the minimal effective scope for each request while carrying forward relevant session constraints.

3.2 Multi-Turn Localized Modify Execution

Plan. MemSlides converts each revision request into an explicit execution contract rather than leaving scope control implicit in the prompt. The contract records the inferred scope, target slide paths, active rule identifiers, selector hints, and whether target coverage is required. Local requests bind to the resolved slide; deck-level rules expand coverage to all slides; and hybrid requests preserve both the global rule and the local exception. Future-only rules created by an insertion request are not incorrectly forced onto existing slides, which avoids turning a structural edit into an unnecessary deck-wide rewrite.

Act. The executor chooses editing tools according to this contract and the available slide structure. If target slides share an explicit selector, it prefers a batch CSS update; if the change targets common

semantics such as titles, body text, or footers across structurally different slides, it uses semantic batch styling; if a single existing slide needs content or local structure changes, it reads a layout-first repair surface and applies non-empty patch operations to snapshot targets or exposed rules. Page insertion and deletion remain explicit page-level operations, while whole-slide rewriting is restricted to new slides or controlled recovery after a corrupted or uninspectable state.

Guard. The protocol treats completion as a checked state, not merely as the model deciding to stop. Patch calls are bound to the snapshot content hash; stale snapshots trigger rebinding hints rather than immediate full rewrite; and every patch must specify concrete operations derived from repair candidates, rules, or targets. When coverage is required, premature `finalize` calls are blocked until all target slides are modified or explicitly confirmed compliant. These guards make localized revision a constrained execution process: the agent can update the intended deck region while reducing repeated re-contextualization, uncontrolled scope expansion, and drift in already aligned slides.

Working memory is the session-scoped state layer that makes Plan–Act–Guard multi-turn rather than single-shot. It stores active temporary preferences from earlier feedback, carryover instructions that remain valid for the current deck, and edit-state records such as resolved targets, coverage status, and snapshot rebinding hints. It also buffers round-level tool-memory signals before transferable experiences are consolidated into long-term tool memory. Plan reads this state to construct the execution contract, Act uses it to restrict edits to the intended region, and Guard updates it after verification or rebinding.

As illustrated in Figure 2, localized modification gives the agent an editable substrate, but personalization still requires separating signals by lifetime and use. Rather than using one homogeneous buffer [16, 7, 14, 3, 10, 43], MemSlides instantiates hierarchy memory through two modules: user profile memory decides what the deck should reflect, and tool memory supports how edits are executed, following agent-memory work on persistent and session state [61, 31, 32, 50, 4, 45, 12, 47, 54].

3.3 User Profile Memory for Personalization

User profile memory governs personalization in MemSlides. Instead of using personalization as a static prompt prefix, we represent it as a persistent user profile plus active temporary memory for the current session:

$$\mathcal{M}_t^{\text{pref}} = \big(P_u, A_t\big), \tag{3}$$

where P_u is user u's long-term profile and A_t is the active temporary memory at revision round t. Stored profile items are organized by intent and presentation dimensions, while only the reconciled subset enters A_t. This separation is the lifecycle shown in Figure 3: user profile memory provides cross-job personalization priors, routing turns the intent-matched and request-compatible items into active temporary memory for the current deck, and job-end consolidation writes back only stable interaction signals. The figure therefore explains why profile memory is not injected as a static prompt block: it is selected, reconciled with the current request, used during round-0 and later revisions, and then updated after the round.

Before round-0 generation, MemSlides selects the profile bucket for the current intent i_0, extracts constraints from the request q_0, and routes compatible items into working memory:

$$\tilde{P}_u = \mathcal{S}(P_u, i_0)\,, \qquad C_0 = \mathcal{E}(q_0)\,, \qquad A_0 = \mathcal{R}\Big(\tilde{P}_u, C_0\Big)\,. \tag{4}$$

Here, $\mathcal{S}$ retrieves the intent-matched profile bucket, $\mathcal{E}$ extracts request constraints, and $\mathcal{R}$ reconciles them. Explicit request conflicts supersede the corresponding profile item for the current deck; compatible preferences coexist as active memory.

During revision, active temporary memory evolves with feedback:

$$A_t = \mathcal{U}(A_{t-1}, r_t)\,, \tag{5}$$

where r_t is the revision request at round t. The update operator $\mathcal{U}$ appends newly exposed preferences, supersedes true conflicts, and keeps non-conflicting items active for later rounds.

At job end, long-term user profile memory is updated by consolidation rather than by directly promoting every temporary item:

$$P_u^{+} = \mathcal{C}(P_u, H)\,, \tag{6}$$

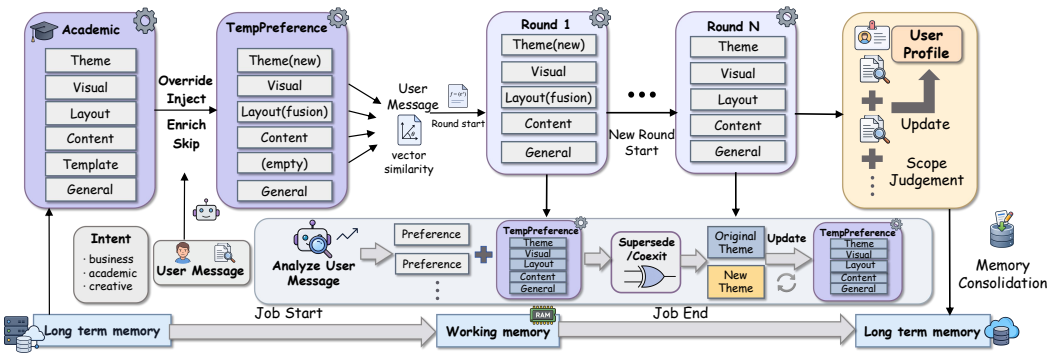


Figure 3: User profile memory lifecycle in MemSlides. Long term memory stores intent-conditioned user profile memory accumulated across jobs. At job start, user profile memory items are routed into active temporary memory by comparing them with the current user request: compatible preferences coexist, explicit conflicts are superseded, and only the active subset guides generation. At job end, stable interaction signals are consolidated back into the user profile memory.

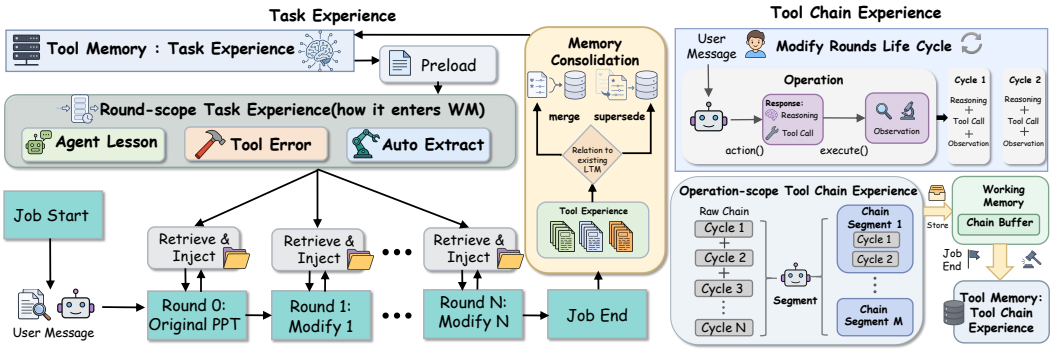


Figure 4: Tool memory flow in MemSlides. Round-scope task experience is available at job start, enters working memory during modify rounds, and is updated through agent lessons, tool-error summaries, and automatically extracted patterns before memory consolidation. Operation-scope tool-chain experience records raw reasoning–tool–observation chains as compact fragments that are retrieved and injected before similar future tool calls.

where H denotes the user messages in the current job and $\mathcal{C}$ is intent-aware profile consolidation. This prevents transient requests from being stored as persistent preferences, while allowing stable, transferable signals to improve future personalization.

3.4 Tool Memory for Reliable Editing

Tool memory addresses the execution side of personalized editing. Even when the target preference is correct, the agent may repeat ineffective trials or re-trigger known tool misuse. Prior tool-using and reflective agents motivate interleaving reasoning, tool calls, and feedback [53, 39, 38, 36, 24]. In our setting, the remembered execution unit is tied to slide-edit scope and verification rather than to raw interaction history.

In MemSlides, tool memory is organized two execution flows that match the temporal granularity of localized editing (Figure 4). Round-scope task experience is available when a modify job starts and is buffered in working memory across later modify rounds; after each round, agent lessons, tool-error summaries, and automatically extracted patterns can update this pool. Operation-scope tool-chain experience is finer grained: raw reasoning–tool–observation chains are segmented into reusable chain fragments, indexed by operation context, and retrieved before similar future tool calls.

We represent this execution support at revision round t and operation k as

$$\mathcal{M}_{t,k}^{\text{tool}} = \left(E_t^{\text{round}}, E_{t,k}^{\text{op}}\right), \tag{7}$$

Table 1: Persona-alignment judgments for first-pass generation. Scores are averaged over three personas on a 0–10 scale; higher is better. Bold marks the best score per metric.

Framework	Model	Content ↑	Structure ↑	Visual ↑	Specificity ↑
DeepPresenter	GPT-5	6.22	7.56	5.76	5.89
	GLM-5	6.67	7.61	5.28	7.22
	Gemini 3.1 Pro	6.89	8.00	6.78	7.44
SlideTailor	GPT-5	6.78	6.00	6.39	6.33
	GLM-5	4.44	4.89	4.00	3.89
	Gemini 3.1 Pro	4.48	5.00	4.03	4.67
MemSlides (Ours)	GPT-5	7.11	7.33	6.00	6.67
	GLM-5	**9.00**	**8.78**	**8.56**	**8.89**
	Gemini 3.1 Pro	7.77	8.64	8.24	8.56

Table 2: General-quality evaluation on the shared three-profile suite. Constraint, Content, Style, and Avg. use a 1–5 scale; Diversity is a suite-level normalized DINOv2-Vendi score. The rows compare independent generated decks from DeepPresenter, SlideTailor, and MemSlides under the same evaluation protocol.

Framework	Model	Constraint ↑	Content ↑	Style ↑	Avg. ↑	Diversity ↑
DeepPresenter	GPT-5	4.83	3.50	3.63	3.99	0.387
	Gemini 3.1 Pro	4.17	3.33	4.00	3.83	0.370
	GLM-5	4.00	3.57	4.00	3.86	0.366
SlideTailor	GPT-5	3.83	2.93	4.03	3.60	0.399
	Gemini 3.1 Pro	3.83	3.20	4.00	3.68	0.364
	GLM-5	3.83	2.97	4.00	3.60	0.348
MemSlides (Ours)	GPT-5	**5.00**	**3.60**	3.90	**4.17**	0.380
	Gemini 3.1 Pro	3.33	3.37	**4.10**	3.60	**0.463**
	GLM-5	3.83	3.34	4.03	3.74	0.391

where E_t^{round} denotes round-scope execution experience and $E_{t,k}^{\text{op}}$ denotes operation-scope tool chain experience for the k-th operation. Tool memory does not define what the deck should look like; it helps the agent execute that objective with fewer repeated errors, less backtracking, and more reliable local verification. Transferable experiences are consolidated into long-term tool memory at job end.

4 Experiments

We evaluate whether MemSlides improves personalized presentation generation and multi-turn localized revision. The main text reports personalization results on a controlled multi-persona, multi-intent user profile bank, a DeepPresenter-style general-quality check on the same decks, and a diagnostic matched-pair modify evaluation for localized revision. Protocol details, baseline conditions, and profile-bank construction are provided in Appendices A.1, A.2, A.4, and A.5, with protocol, profile-bank, and compute summaries in Appendix Tables 4, 6, and 5; table captions define the reported metrics.

4.1 Main Results

4.1.1 Personalization

User profile memory improves round-0 persona alignment across multiple dimensions. Under the persona-alignment judgments, OURS achieves all-column wins over both baselines on GLM-5 and Gemini 3.1 Pro. With GPT-5, it remains ahead of SlideTailor on Content, Structure, and Specificity, and ahead of DeepPresenter on Content, Visual, and Specificity, while DeepPresenter has slightly higher Structure (Table 1; Appendix Table 7 gives per-persona GPT-5 details).

On the same generated decks, the DeepPresenter-style quality metrics in Table 2 check whether persona gains remain compatible with ordinary presentation quality. The results show that MemSlides improves personalization alignment while maintaining competitive PPT generation quality.

Table 3: Tool-memory ablation on nine diagnostic modify pairs. Completion and verification are higher-is-better; time and ratio are lower-is-better.

Model	Memory Injected	Closed-Loop Completion ↑	Strict Verify ↑	First Correct Edit (s) ↓	Core Tool Time Ratio ↓
GPT-5	✓	**1.000**	**0.646**	**211.3**	**0.740**×
	×	0.667	0.294	234.2	1.000×
GLM-5	✓	**1.000**	**0.488**	**195.9**	**0.344**×
	×	0.889	0.434	500.9	1.000×
Gemini 3.1 Pro	✓	0.889	**0.469**	**309.9**	**0.137**×
	×	0.889	0.201	968.2	1.000×
Overall	✓	**0.963**	**0.534**	**242.5**	**0.327**×
	×	0.815	0.310	609.5	1.000×

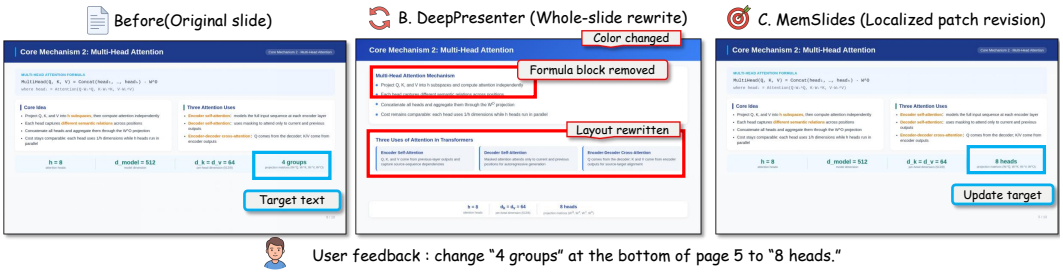


Figure 5: Qualitative illustration of localized modify execution. Given the same local edit request, DeepPresenter can satisfy the target change while altering non-target regions of the slide. MemSlides instead applies a targeted patch to the requested element, preserving already aligned slide content.

4.1.2 Localized Revision

In this diagnostic matched-pair modify setting, tool-memory injection is associated with stronger closed-loop completion and post-edit verification while reducing time to the first correct edit and core tool time (Table 3). Since each matched pair changes only tool-memory injection, these process gains support the combined tool-memory pathway for reliable multi-turn modification, while the W-L-T-NA counts expose pair-level heterogeneity; Appendix Tables 8 and 9 provide pair-level details and the paired robustness check.

The same locality-sensitive setting is illustrated qualitatively in Figure 5, which contrasts a broader edit footprint with a targeted patch on the requested element; Appendix Figure 8 shows a complementary process trajectory.

4.2 Analysis and Discussion

User profile memory yields broad, not marginal, alignment gains. Table 1 shows a consistent but not uniform pattern across model families. With GLM-5 and Gemini 3.1 Pro, MemSlides leads both baselines on all four persona-alignment dimensions. With GPT-5, the gains are strongest on Content and Specificity, while the two baselines retain isolated advantages on Structure and Visual. Averaged across model families, MemSlides improves over DeepPresenter by 1.37 points on Content, 0.53 on Structure, 1.66 on Visual, and 1.19 on Specificity, and improves over SlideTailor by 2.73, 2.95, 2.79, and 3.08 points on the same dimensions.

The gains are planning-level persona gains. The strongest evidence is the joint movement of Structure and Specificity. Structure excludes template retrieval accuracy, while Specificity uses distractor personas; moreover, judges do not see the original prompt or parsed intent. Thus the improvement cannot be read as merely better template matching or generic visual polish. It indicates that routed long-term profiles help decide page roles, ordering, layout fit, evidence emphasis, and persona-distinct framing before round-0 generation.

Persona gains remain compatible with general deck quality. Table 2 shows that OURS obtains the best Avg. on GPT-5 and remains competitive on GLM-5, while Gemini 3.1 Pro shows the best Style and Diversity but lower Constraint. These results do not claim uniform dominance on every quality

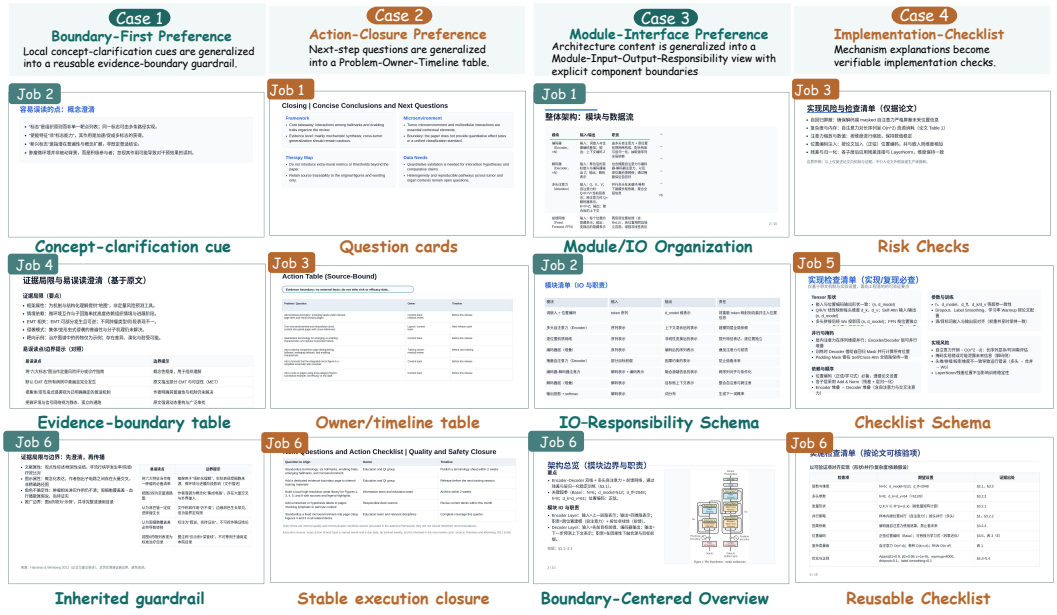


Figure 6: Qualitative cross-job profile consolidation. Across six repeated jobs, local feedback cues become reusable profile preferences and later reappear as default slide-organization patterns, including evidence-boundary guardrails, owner/timeline closure tables, module input–output responsibility schemas, and implementation checklists. The figure is diagnostic case evidence rather than a separate quantitative metric.

metric; they show that the persona-alignment improvements in Table 1 are not simply a trade-off against ordinary presentation quality.

Localized revision should be judged by locality as well as success. Figure 5 shows the qualitative difference behind the modify setting: a system may satisfy the requested change while still touching non-target regions, whereas MemSlides keeps the patch closer to the requested element. This is why the localized-revision metrics are paired with process constraints such as closed-loop completion, verification, and core-tool-time efficiency.

Scope of profile and working-memory evidence. The round-0 persona-alignment table is the main quantitative evidence for user profile memory. Figure 6 provides qualitative cross-job profile-consolidation evidence, showing how repeated local cues become reusable organization preferences in later jobs. Appendix Figure 9 provides complementary within-session evidence for delayed carryover of active temporary memory, and Appendix Figure 7 shows template-guided generation examples.

Tool memory improves overall reliability and search efficiency, but not by winning every pair. In the diagnostic matched-pair modify setting, tool-memory injection raises overall Closed-Loop Completion from 0.815 to 0.963 (W-L-T-NA 3-1-5-0) and Strict Verify from 0.310 to 0.534 (8-1-0-0). It also reduces Time to First Correct Edit from 609.5s to 242.5s (6-2-0-1) and lowers the geometric mean Core Tool Time Ratio to 0.327× (8-1-0-0), indicating less non-inspection tool work. The pair-level counts are important: one Gemini hard-modify pair loses on Closed-Loop Completion, two pairs lose on first-edit latency, and one GPT-5 pair uses more core tool time. Together with the pair-level detail in Appendix A.8, these process metrics are consistent with a more constrained localized editing pattern rather than repeated full-deck regeneration or broad exploratory edits.

5 Conclusion

We introduced MemSlides, a hierarchical memory framework for personalized presentation generation. By separating user profile memory, active temporary memory, and tool memory, MemSlides supports round-0 persona alignment and multi-turn localized revision. Controlled experiments show improved persona alignment and diagnostic gains in local modify reliability.

6 Limitations

Our evidence is scoped to controlled persona-alignment judgments, diagnostic matched-pair modify settings, and qualitative working-memory cases. The profile bank and edit requests are proxies rather than real-user deployment studies. Future work should add broader human studies, randomized edit sets, and stronger memory consent, deletion, and sensitive-preference safeguards.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

[2] Sambaran Bandyopadhyay, Himanshu Maheshwari, Anandhavelu Natarajan, and Apoorv Saxena. Enhancing presentation slide generation by LLMs with a multi-staged end-to-end approach. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 222–229, Tokyo, Japan, September 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.inlg-main.18. URL https://aclanthology.org/2024.inlg-main.18/.

[3] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2021. URL https://arxiv.org/abs/2112.04426.

[4] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. doi: 10.48550/arXiv.2504.19413. URL https://arxiv.org/abs/2504.19413.

[5] Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=g97OHbQyk1.

[6] Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, and Trevor Darrell. AutoPresent: Designing structured visuals from scratch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2902–2911, June 2025. doi: 10.1109/CVPR52734.2025.00276. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ge_AutoPresent_Designing_Structured_Visuals_from_Scratch_CVPR_2025_paper.html.

[7] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2002.08909.

[8] Jinchao Hu, Meizhi Zhong, Kehai Chen, Xuefeng Bai, and Min Zhang. Agentic tool use in large language models. *arXiv preprint arXiv:2604.00835*, 2026. URL https://arxiv.org/abs/2604.00835.

[9] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete diffusion model for controllable layout generation. *arXiv preprint arXiv:2303.08137*, 2023. URL https://arxiv.org/abs/2303.08137.

[10] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022. URL https://arxiv.org/abs/2208.03299.

[11] Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*, 2025. doi: 10.48550/arXiv.2504.14225. URL https://arxiv.org/abs/2504.14225.

[12] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

[13] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022. URL https://arxiv.org/abs/2205.00445.

[14] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019. URL https://arxiv.org/abs/1911.00172.

[15] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*, 2023. URL https://arxiv.org/abs/2310.08491.

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[17] Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1094. URL https://aclanthology.org/P16-1094/.

[18] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. *arXiv preprint arXiv:2304.08244*, 2023. URL https://arxiv.org/abs/2304.08244.

[19] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024. URL https://arxiv.org/abs/2406.11939.

[20] Xiaopeng Li, Pengyue Jia, Derong Xu, Yi Wen, Yingyi Zhang, Wenlin Zhang, Wanyu Wang, Yichao Wang, Zhaocheng Du, Xiangyang Li, Yong Liu, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. A survey of personalization: From RAG to agent. *arXiv preprint arXiv:2504.10147*, 2025. URL https://arxiv.org/abs/2504.10147.

[21] Xin Liang, Xiang Zhang, Yiwei Xu, Siqi Sun, and Chenyu You. SlideGen: Collaborative multimodal agents for scientific slide generation. *arXiv preprint arXiv:2512.04529*, 2025. doi: 10.48550/arXiv.2512.04529. URL https://arxiv.org/abs/2512.04529.

[22] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. Layout-Prompter: Awaken the design ability of large language models. *arXiv preprint arXiv:2311.06495*, 2023. URL https://arxiv.org/abs/2311.06495.

[23] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023. URL https://arxiv.org/abs/2303.16634.

[24] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

[25] Himanshu Maheshwari, Sambaran Bandyopadhyay, Aparna Garimella, and Anandhavelu Natarajan. Presentations are not always linear! GNN meets LLM for text document-to-presentation transformation with attribution. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15948–15962, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.936. URL https://aclanthology.org/2024.findings-emnlp.936/.

[26] Ishani Mondal, Shwetha S, Anandhavelu Natarajan, Aparna Garimella, Sambaran Bandyopadhyay, and Jordan Boyd-Graber. Presentations by the humans and for the humans: Harnessing LLMs for generating persona-aware slides from documents. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2664–2684, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.163. URL https://aclanthology.org/2024.eacl-long.163/.

[27] Sheshera Mysore, Zhuoran Lu, Mengting Wan, Longqi Yang, Bahareh Sarrafzadeh, Steve Menezes, Tina Baghaee, Emmanuel Barajas Gonzalez, Jennifer Neville, and Tara Safavi. Pearl: Personalizing large language model writing assistants with generation-calibrated retrievers. *arXiv preprint arXiv:2311.09180*, 2023. URL https://arxiv.org/abs/2311.09180.

[28] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

[29] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

[30] Tarik Can Ozden, Sachidanand VS, Furkan Horoz, Ozgur Kara, Junho Kim, and James M. Rehg. Narrative-driven paper-to-slide generation via ArcDeck. *arXiv preprint arXiv:2604.11969*, 2026. doi: 10.48550/arXiv.2604.11969. URL https://arxiv.org/abs/2604.11969.

[31] 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. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

[32] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023. doi: 10.48550/arXiv.2304.03442. URL https://arxiv.org/abs/2304.03442.

[33] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

[34] Yi-Hao Peng, Jeffrey P. Bigham, and Jason Wu. Designpref: Capturing personal preferences in visual design generation. *arXiv preprint arXiv:2511.20513*, 2025. doi: 10.48550/arXiv.2511.20513. URL https://arxiv.org/abs/2511.20513.

[35] Xin Qian, Ryan A. Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Nesreen K. Ahmed. Personalized visualization recommendation. *ACM Transactions on the Web*, 16(3):11:1–11:47, 2022. doi: 10.1145/3538703. URL https://doi.org/10.1145/3538703.

[36] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023. URL https://arxiv.org/abs/2307.16789.

[37] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*, 2023. doi: 10.48550/arXiv.2304.11406. URL https://arxiv.org/abs/2304.11406.

[38] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[39] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

[40] Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy X. R. Wang. D2S: Document-to-slide generation via query-based text summarization. *arXiv preprint arXiv:2105.03664*, 2021. doi: 10.48550/arXiv.2105.03664. URL https://arxiv.org/abs/2105.03664.

[41] Zecheng Tang, Chenfei Wu, Juntao Li, and Nan Duan. LayoutNUWA: Revealing the hidden layout expertise of large language models. *arXiv preprint arXiv:2309.09506*, 2023. URL https://arxiv.org/abs/2309.09506.

[42] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

[43] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*, 2023. doi: 10.48550/arXiv.2306.07174. URL https://arxiv.org/abs/2306.07174.

[44] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. URL https://arxiv.org/abs/2203.11171.

[45] Yu Wang and Xi Chen. Mirix: Multi-agent memory system for LLM-based agents. *arXiv preprint arXiv:2507.07957*, 2025. doi: 10.48550/arXiv.2507.07957. URL https://arxiv.org/abs/2507.07957.

[46] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155.

[47] Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to AI memory: A survey on memory mechanisms in the era of LLMs. *arXiv preprint arXiv:2504.15965*, 2025. URL https://arxiv.org/abs/2504.15965.

[48] Yuwei Wu, Xuezhe Ma, and Diyi Yang. Personalized response generation via generative split memory network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1956–1970, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.157. URL https://aclanthology.org/2021.naacl-main.157/.

[49] Eric Xie, Danielle Waterfield, Michael Kennedy, and Aidong Zhang. SlideBot: A multi-agent framework for generating informative, reliable, multi-modal presentations. *arXiv preprint arXiv:2511.09804*, 2025. doi: 10.48550/arXiv.2511.09804. URL https://arxiv.org/abs/2511.09804.

[50] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025. doi: 10.48550/arXiv.2502.12110. URL https://arxiv.org/abs/2502.12110.

[51] Xiaojie Xu, Xinli Xu, Sirui Chen, Haoyu Chen, Fan Zhang, and Ying-Cong Chen. PreGenie: An agentic framework for high-quality visual presentation generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3045–3063, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.165. URL https://aclanthology.org/2025.findings-emnlp.165/.

[52] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. URL https://arxiv.org/abs/2305.10601.

[53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[54] Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026. doi: 10.48550/arXiv.2601.01885. URL https://arxiv.org/abs/2601.01885.

[55] Wenzheng Zeng, Mingyu Ouyang, Langyuan Cui, and Hwee Tou Ng. SlideTailor: Personalized presentation slide generation for scientific papers. *arXiv preprint arXiv:2512.20292*, 2025. doi: 10.48550/arXiv.2512.20292. URL https://arxiv.org/abs/2512.20292.

[56] Junyi Zhang, Jiaqi Guo, Shizhao Sun, Jian-Guang Lou, and Dongmei Zhang. Layoutdiffusion: Improving graphic layout generation by discrete diffusion probabilistic models. *arXiv preprint arXiv:2303.11589*, 2023. URL https://arxiv.org/abs/2303.11589.

[57] Zhehao Zhang, Ryan A. Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, Ruiyi Zhang, Jiuxiang Gu, Tyler Derr, Hongjie Chen, Junda Wu, Xiang Chen, Zichao Wang, Subrata Mitra, Nedim Lipka, Nesreen Ahmed, and Yu Wang. Personalization of large language models: A survey. *arXiv preprint arXiv:2411.00027*, 2024. URL https://arxiv.org/abs/2411.00027.

[58] Hao Zheng, Xinyan Guan, Hao Kong, Wenkai Zhang, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. PPTAgent: Generating and evaluating presentations beyond text-to-slides. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14402–14418, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.728. URL https://aclanthology.org/2025.emnlp-main.728/.

[59] Hao Zheng, Guozhao Mo, Xinru Yan, Qianhao Yuan, Wenkai Zhang, Xuanang Chen, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. DeepPresenter: Environment-grounded reflection for agentic presentation generation. *arXiv preprint arXiv:2602.22839*, 2026. doi: 10.48550/arXiv.2602.22839. URL https://arxiv.org/abs/2602.22839.

[60] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.

[61] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023. doi: 10.48550/arXiv.2305.10250. URL https://arxiv.org/abs/2305.10250.

[62] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023. URL https://arxiv.org/abs/2310.17631.

A Appendix

The appendix provides protocol details, supplemental quantitative tables, and qualitative examples that clarify how different memory signals appear in generated presentations. The qualitative figures are intended as diagnostic illustrations, while the aggregate claims remain tied to the quantitative tables in the main text and appendix.

A.1 Evaluation Protocol Details

Section 4 defines the evaluated claims and reports the main metrics. This appendix subsection records the protocol controls used to keep those metrics attributable to the corresponding memory condition, rather than to prompt leakage, arm ordering, or unmatched source materials.

Blind profile-memory judging. The profile-memory evaluation uses complete round-0 decks as the judging unit. Before judging, every candidate deck is rendered into page-aligned images and assigned an anonymous arm label. The judge receives only the target persona summary, the aligned deck images, and the dimension rubric; the original user prompt, parsed intent, system identity, and memory condition are hidden. This separation is important because the profile claim concerns whether persistent memory changes the produced deck, not whether a judge can recover instructions that were explicitly written in the current prompt. Each persona-alignment dimension receives three blind votes. We aggregate the votes under anonymous arm labels and map arms back to systems only after aggregation.

Persona-alignment judgment operationalization. The persona-alignment judgments report four 0–10 dimensions. *Content* measures whether content selection, evidence type, emphasis, and wording match the target persona. *Structure* measures whether page order and page-type/layout fit reflect persona-conditioned deck organization; it excludes template matching because this dimension targets deck organization rather than template retrieval accuracy. *Visual* measures information density, whitespace, chart/card/diagram style, visual hierarchy, and overall visual tone. *Specificity* uses distractor personas to test whether the deck remains identifiable as the target persona rather than a generic professional presentation.

Matched tool-memory pairs. The tool-memory protocol evaluates localized multi-turn modification as paired agent executions. Within each pair, the source deck, model family, persona, and modify request are fixed; the compared condition is whether tool memory is injected into the run. The reported process metrics are computed from recorded execution traces: edit completion, verification after change, finalization, time to first correct edit, and core tool time. Inspection and markdown-conversion calls are excluded from core tool time because they reflect observation and format transfer rather than edit execution. The modify scenarios are selected before inspecting pair outcomes and are intended as a diagnostic controlled setting, not as a distribution-level estimate over all possible user revisions.

Tool-metric operationalization. Table 3 uses two higher-is-better metrics and two lower-is-better efficiency metrics. *Closed-Loop Completion* measures whether a modify run reaches a successful local edit, verifies the edited result, and finalizes the revised deck. *Strict Verify After Change* measures whether successful slide-changing edits are followed by local verification within a short tool-call window, so it rewards edit-and-check behavior rather than late or unrelated inspection. *Time to First Correct Edit* is the wall-clock interval from modify-task start to the first slide-changing edit that satisfies the requested change. *Core Tool Time Ratio* is the geometric mean of memory/no-injection core-tool time ratios after excluding inspection and markdown-conversion tools, with the no-injection arm normalized to 1.0. These definitions make the table about the editing trajectory itself: whether the agent reaches the requested local change, checks it promptly, and does so with less non-inspection tool work.

Reproducibility information. The paper reports the evaluated systems, model families, judge rubrics, evaluation configurations, matched-pair definitions, and trace-derived metrics needed to inspect the reported protocol. The source code is publicly available, and selected evaluation artifacts will be released when documentation and licensing checks are finalized. Table 4 summarizes the main protocol controls for the profile-memory and tool-memory evaluations.

Table 4: Protocol controls used to isolate memory effects in Section 4.

Control	Profile-memory evaluation	Tool-memory evaluation
Evaluation unit	Complete round-0 deck	Paired modify execution
Matched factors	Source, model family, task prompt	Source deck, model, persona, request
Varied condition	Profile-memory injection	Tool-memory injection
Hidden signals	Prompt, intent, system, memory label	Memory-condition label
Aggregation	Three blind votes per dimension	Pair-level trace metrics
Evidence scope	Main persona-alignment result	Diagnostic selected-pair setting

A.2 Baseline Conditions and Prompt/Profile Separation

The persona-alignment judgment comparison is designed to isolate persistent profile memory from prompt-level task conditioning. The memory-injected condition and its matched control use the same source material, model family, target persona, task prompt, and generation pipeline. The control condition keeps the authoring task unchanged but withholds long-term profile memory, session preference memory, and reusable tool experience from the generation context. The memory-injected condition receives the same task information plus the routed profile entry for the target persona-intent pair.

This separation matters because the task prompt already contains the source-specific request and role intent. Profile memory is therefore evaluated as an additional persistent signal: structured preferences accumulated across controlled authoring jobs, not a replacement for the user request. SlideTailor is included as an external reference/template-conditioned personalization baseline; it is evaluated from its generated presentations under the same judging protocol, but it is not treated as an internal memory ablation.

The judge-side protocol further prevents prompt or template leakage from dominating the score. The persona-alignment judge receives the target persona summary and rendered deck images, but not the original prompt, parsed task intent, system identity, or memory condition. The Structure metric deliberately excludes template matching, so the main profile table evaluates persona-conditioned organization and layout fit rather than template retrieval accuracy.

A.3 Compute Resources and Runtime Accounting

All reported experiments are inference-time agent evaluations; no model training or fine-tuning is performed. Runs were orchestrated from a local Linux workstation using Python 3.11.14. The machine has 48 Intel Xeon Silver 4214R CPU threads and 503 GiB of system memory; although the machine also has NVIDIA A40 GPUs, the reported experiments are primarily API-bound rather than GPU-bound. The local machine is used for agent orchestration, document conversion, slide rendering, image inspection, and tool-mediated HTML/PPT editing; LLM generation and judging are performed through external model APIs.

For runtime accounting, each run records model-response usage, local tool-call outcomes, tool active time, and first-to-last-event wall-clock span. Because different LLM providers expose token accounting fields with slightly different schemas, we use these records as reproducibility accounting rather than as a normalized price estimate. Table 5 reports the resulting accounting summary. The prompt and completion token columns sum provider-reported token usage; tool-call columns count successful and failed local tool invocations; tool active time sums local tool execution duration; and wall-clock span measures the elapsed time from the first recorded event to the last recorded event within each run.

The accounting rows are evidence scopes rather than additional experimental conditions. The profile-memory row summarizes the generation and analysis runs used for profile-memory experiments, so it should not be read as the number of decks used in Table 1. The tool-memory row corresponds to the diagnostic matched-pair setting reported in Tables 3 and 8, counting both the memory-injected and no-injection arms for the nine matched pairs. The working-memory row covers the four runs used to construct the two qualitative carryover cases in Figure 9. The table is intended to document API budget and local orchestration/runtime footprint, not to serve as a method-performance result or a GPU-compute claim.

Table 5: API usage and local runtime for reported evidence scopes.

Evidence scope	Runs	LLM calls	Prompt tokens	Completion tokens	Tool calls ok/err	Tool active time (min)	Wall span sum (h)	Median run span (min)
Profile-memory experiments	49	2,257	49.82M	2.74M	3,527/552	60.78	19.37	19.12
Tool-memory nine-pair	18	1,085	19.40M	0.92M	1,931/220	62.25	17.52	43.99
Working-memory cases	4	161	2.35M	0.27M	343/21	13.85	2.64	35.09

For the diagnostic matched-pair tool-memory setting, the recorded process metrics also record the core tool-time used in Table 3: 110.5 seconds for memory-injected runs and 354.8 seconds for no-injection runs, after excluding inspection and markdown-conversion tools. Under the same exclusion, the matched runs contain 779 core tool calls for memory-injected runs and 878 for no-injection runs.

A.4 Profile-Bank Construction

The profile bank is built as a controlled multi-persona, multi-intent testbed for long-term personalization. We define ten occupation-style persona profiles, each associated with three role-intent buckets, resulting in 30 persona-intent profile entries. The personas cover postsecondary teacher, software developer, management analyst, marketing manager, graphic designer, training and development specialist, financial manager, operations manager, medical and health services manager, and legislator. Each entry is used as the read/write unit for long-term profile memory during generation. Table 6 lists all 30 entries in the completed profile bank. The third column reports shortened English summaries of stored `profile_json` fields: `layout.slide_structure`, `visual.chart_type_priority`, and, where useful, `content.notes`.

Table 6: Complete 30-entry profile bank used for long-term personalization. Each row corresponds to one persona-intent profile entry. The third column reports English summaries of stored `profile_json` fields; long values are shortened for readability.

Persona	Role-intent bucket	Stored profile fields (English summaries)
Postsecondary teacher	Course unit teaching	`layout.slide_structure`: cover → learning objectives → core concept breakdown → method workflow → classroom example → recap/Q&A; `content.notes`: organize course units around objectives, concepts, methods, and examples.
Postsecondary teacher	Research method explanation	`layout.slide_structure`: two-column layout with explanation on the left and architecture/local diagrams on the right; `content.notes`: for graduate students, clarify problem definition, method steps, key experimental settings, and method rationale.
Postsecondary teacher	Academic progress review	`layout.slide_structure`: background/goals—stage progress—key results—open issues—next plan; `content.notes`: include core parameterized formulas and emphasize precise academic definitions.
Software developer	Architecture walkthrough	`layout.slide_structure`: overview/architecture diagram → module structure → data flow → design trade-offs → experimental evidence; `visual.chart_type_priority`: architecture/topology diagrams for overview pages and flowcharts for data flow.
Software developer	Sprint risk update	`layout.slide_structure`: current goal → completed capabilities → main risks → key dependencies → next steps; `visual.chart_type_priority`: status-risk comparison diagrams and milestone timelines.
Software developer	Technical value brief	`content.notes`: organize into application scenarios, core technical capabilities, performance evidence, and engineering value; `visual.chart_type_priority`: comparison tables and evidence charts.
Management analyst	Organizational diagnosis brief	`layout.slide_structure`: current issues—root-cause breakdown—capability gaps—optimization directions; `visual.chart_type_priority`: issue trees, 2×2 priority matrices, capability heatmaps, and fishbone diagrams.
Management analyst	Framework training session	`layout.slide_structure`: use structured summary pages such as 2×2 matrices or card frameworks with action orientation; `visual.chart_type_priority`: process diagrams, decision trees, and priority matrices.

Persona	Role-intent bucket	Stored profile fields (English summaries)
Management analyst	Recommendation follow-up review	`layout.slide_structure`: multi-column card layout and priority matrix for follow-up actions; `content.notes`: prioritize conclusions and logic chains over raw data stacking.
Marketing manager	Go-to-market pitch	`layout.slide_structure`: open with a hook, use numeric anchors in the middle, and close with a call to action; `content.notes`: prefer short, forceful titles and remove redundant labels such as "objective."
Marketing manager	Brand campaign showcase	`layout.slide_structure`: put the core narrative on the cover; use big numbers and conclusion cards on inner slides while reducing tables; `content.notes`: avoid meta-labels on the cover.
Marketing manager	Campaign performance review	`layout.slide_structure`: highlight key metrics with KPI cards and reduce table/screenshot clutter; `visual.chart_type_priority`: KPI cards and metric cards, avoiding table screenshots.
Graphic designer	Visual direction showcase	`layout.slide_structure`: cover with theme/signature color → mood board → color palette and usage ratio → font pairing and type scale → grid/spacing rules; `content.notes`: avoid information overload and keep the visual direction sparse and clear.
Graphic designer	Design rationale pitch	`layout.slide_structure`: cover → design goals/audience → constraints and brand tone → alternatives/trade-offs → visual rationale; `visual.chart_type_priority`: annotated design diagrams, trade-off matrices, scorecards, and before/after cards.
Graphic designer	Design iteration review	`layout.slide_structure`: emphasize clear typography hierarchy and avoid overlap between text, graphics, and containers; `visual.chart_type_priority`: side-by-side before/after cards and annotated mockups with numbered callouts.
Training and development specialist	Capability training workshop	`layout.slide_structure`: organize by numbered learning-path steps; `content.notes`: use action-oriented titles with step numbers and cover objectives, key steps, examples, and recap reminders.
Training and development specialist	Learning solution pitch	`layout.slide_structure`: consolidate information into three or four logical modules and reduce scattered bullets; `content.notes`: prefer diagrammatic expression while keeping key terms and conclusion prompts such as expert recommendations.
Training and development specialist	Rollout effectiveness review	`layout.slide_structure`: organize around training goals, execution coverage, effectiveness observations, key issues, and improvement plans; `visual.chart_type_priority`: process diagrams, highlighted tables, and comparison tables.
Financial manager	Budget and ROI assessment	`layout.slide_structure`: one table per slide for comparison/impact tables, avoiding table walls; organize around inputs, expected returns, key metrics, and constraints; `visual.chart_type_priority`: key-variable impact tables and KPI summary tables.
Financial manager	Finance control training	`layout.slide_structure`: organize around policy goals, key controls, common risks, and execution requirements; `visual.chart_type_priority`: comparison tables, checklists, process diagrams, and KPI scorecards.
Financial manager	Financial risk review	`layout.slide_structure`: use conclusion/risk-stance titles and risk indicator panels with changes in liquidity, leverage, and coverage; `content.notes`: prefer quantitative indicators over vague wording.
Operations manager	Execution alignment plan	`layout.slide_structure`: prioritize process flow, combine timelines and kanban boards, and keep work-package boundaries clear; `visual.chart_type_priority`: flowcharts, timelines, and kanban boards.
Operations manager	Process adoption training	`layout.slide_structure`: convert lists into structured visuals: horizontal process diagrams for steps, responsibility matrices for ownership, and card grids for risks; `visual.chart_type_priority`: horizontal flowcharts, card grids, and responsibility matrices.

Persona	Role-intent bucket	Stored profile fields (English summaries)
Operations manager	Delivery risk review	`layout.slide_structure`: use kanban/card layouts for baseline and configuration points; use tables for risk-remedy mapping; `content.notes`: cover delivery goal, current state, blocking risks, impact scope, and remediation plan.
Medical and health services manager	Clinical workflow evidence review	`layout.slide_structure`: structured ten-slide flow around workflow status, key evidence, metric changes, and improvement directions; `visual.chart_type_priority`: metric dashboards, horizontal clinical pathway diagrams, alert cards, and risk panels.
Medical and health services manager	Compliance protocol training	`layout.slide_structure`: use tables for checklists and comparisons, especially two-column tables in right-side modules; `content.notes`: clearly distinguish investigational from approved status for drugs or treatments.
Medical and health services manager	Quality and safety review	`layout.slide_structure`: visual-first layout with large metric cards, comparison diagrams for risks, and flowcharts for processes; `content.notes`: organize quality/safety reviews around key metrics, incidents, cause analysis, corrective actions, and next steps.
Legislator	Public issue evidence brief	`layout.slide_structure`: ten-slide evidence brief organized as issue background—core evidence—policy options—public impact; `visual.chart_type_priority`: comparison tables for policy trade-offs and tabular evidence.
Legislator	Policy communication briefing	`layout.slide_structure`: policy goals → key content → implementation points → public understanding path; `visual.chart_type_priority`: evidence-linkage tables.
Legislator	Implementation accountability review	`layout.slide_structure`: cover—commitments/framing—implementation progress and deviations—issue list and impact—accountability—remediation and supervision—milestone timeline—decision points; `visual.chart_type_priority`: responsibility matrices, issue-owner tracking tables, and compact remediation Gantt charts.

Library construction has two stages. First, each persona-intent entry receives controlled authoring interactions over the same source material but with different role-intent prompts, producing initial profile evidence for that entry. Second, because some accumulated entries remain sparse in structured fields, we apply a seeded completion step. This step uses stable persona prompts, an occupation-grounded role-preference registry, and existing profile signals to fill missing theme, visual, layout, content, and general fields. It follows only-fill-empty and intent-preservation rules: existing intent-specific signals are preserved when they already contain usable information, and generic task constraints such as page count or attachment-use instructions are filtered out.

The seeded completion is treated as profile-bank construction, not as additional interaction evidence. It updates only the structured profile entry and does not create synthetic interaction episodes, tool experiences, or template-usage records. Each filled field is accompanied by provenance tags that identify whether it came from the seed prompt, role-preference registry, current profile signal, or suite intent definition. At runtime, the generation condition reads the completed profile entry matching the current persona and role intent, while the no-injection baseline uses the same source task without profile memory. The evaluation is defined by semantic persona and role-intent conditions rather than by implementation-specific profile identifiers.

A.5 DeepPresenter-Style Quality Evaluation

We evaluate the generated decks with the general-quality metric family reported by DeepPresenter [59]. This protocol is not a persona-specific alignment judge and does not replace Table 1 in the main text. Instead, it asks whether user-profile-memory injection preserves or improves standard presentation-quality dimensions while improving persona alignment.

The metrics are reproduced as follows. *Constraint*, *Content*, and *Style* are reported on a 1–5 scale. *Content* and *Style* are computed with the released PPTEval-style prompt implementation used for presentation quality evaluation [58]. Content evaluates slide-level content clarity and image-text complementarity, while Style evaluates slide-level visual appeal and style coherence. *Constraint* is reimplemented from the DeepPresenter paper definition for this task suite: it checks hard task

constraints and instruction-level requirements that can be verified from the generated deck and source material. For SlideTailor, we report a language-adjusted Constraint score because its current generation setup does not target the requested output language; the language item is excluded while the remaining hard and semantic checks are retained. *Diversity* follows the DeepPresenter diversity definition, using deck-level visual embeddings from DINOv2 [29] and the normalized Vendi score [5]; it is a suite-level metric. *Avg.* is the arithmetic mean of Constraint, Content, and Style, and Diversity is not included in Avg. All values are re-evaluated on the generated presentations used in this work.

Table 2 in the main text reports these general-quality results; bold marks the best score in each metric column over the full table. Across the three model families, user-profile-memory injection keeps OURS competitive on the general-quality metrics while the persona-alignment table shows stronger personalization gains. The most stable pattern is on Content: OURS is above SlideTailor for all three model families and is close to or above the DeepPresenter no-injection baseline. OURS also achieves the highest Avg. on GPT-5 and GLM-5, while Gemini 3.1 Pro has a lower Avg. because of the Constraint score. On the visual side, OURS obtains the best Style and Diversity scores for Gemini 3.1 Pro and remains close to the strongest systems in the other model families. Taken together, the table suggests that user-profile-memory injection is compatible with competitive general-quality behavior, but not uniformly dominant on every metric or every model.

This quality check matters because it shows that the main persona-alignment gains do not come from a trivial trade-off that damages ordinary presentation quality. The evaluation uses the DeepPresenter quality family as an external consistency check, with Content and Style from the released PPTEval-style prompts, Constraint reimplemented from the DeepPresenter-style definition for this task suite, and Diversity computed from deck-level DINOv2 embeddings with a normalized Vendi score. It therefore supports a narrow and reviewer-safe conclusion: MemSlides improves persona-focused generation while remaining competitive on standard quality metrics, even though the gains are not monotonic across every model-metric pair.

A.6 Additional Persona-Alignment Judgment Results

The GPT-5 profile-memory details indicate that the main persona-alignment trend is not driven by a single persona. Table 7 reports ten-persona results under the persona-alignment judgments. On average, profile-memory injection improves Overall alignment by 2.42 points, with gains of 3.30 on Content, 2.30 on Structure, 3.17 on Visual, and 2.43 on Specificity. The gains are strongest when the target persona changes evidence selection, narrative organization, or visual treatment substantially; they are smaller when the no-injection baseline already happens to match the target persona reasonably well.

Table 7: GPT profile-memory details under the persona-alignment judgments. Each metric reports Ours, DeepPresenter, and their difference (Δ = Ours - DeepPresenter) on the 0–10 judge scale. *Overall* averages *Content*, *Structure*, and *Visual*; *Specificity* is reported separately.

Persona	Overall			Content			Structure			Visual			Specificity		
	Ours	DeepPresenter	Δ	Ours	DeepPresenter	Δ	Ours	DeepPresenter	Δ	Ours	DeepPresenter	Δ	Ours	DeepPresenter	Δ
Average (10)	**7.70**	**5.28**	**+2.42**	**8.35**	**5.05**	**+3.30**	**8.07**	**5.77**	**+2.30**	**7.97**	**4.80**	**+3.17**	**7.13**	**4.70**	**+2.43**
Graphic designer	7.33	3.33	+4.00	9.00	2.33	+6.67	7.33	3.33	+4.00	8.33	3.33	+5.00	6.00	2.00	+4.00
Marketing manager	5.95	2.78	+3.17	8.33	3.67	+4.67	9.00	4.00	+5.00	8.17	4.33	+3.83	8.00	5.33	+2.67
Financial manager	8.33	4.33	+4.00	8.33	4.67	+3.67	8.67	5.33	+3.33	8.50	4.00	+4.50	8.00	5.00	+3.00
Postsecondary teacher	8.67	5.33	+3.33	9.00	6.00	+3.00	9.00	6.67	+2.33	9.00	3.33	+5.67	8.67	5.33	+3.33
Operations manager	7.33	6.33	+1.00	8.00	3.33	+4.67	8.67	6.00	+2.67	8.00	3.67	+4.33	6.00	4.33	+1.67
Management analyst	8.33	5.33	+3.00	8.67	5.67	+3.00	7.00	5.67	+1.33	9.00	5.00	+4.00	8.00	5.67	+2.33
Training and development specialist	9.00	6.00	+3.00	9.00	5.33	+3.67	9.00	5.67	+3.33	8.33	4.67	+3.67	8.67	5.33	+3.33
Legislator	7.40	6.67	+0.73	6.67	7.00	-0.33	8.33	6.33	+2.00	7.67	6.00	+1.67	6.67	4.00	+2.67
Software developer	8.33	6.00	+2.33	9.00	6.33	+2.67	6.67	8.33	-1.67	6.33	7.00	-0.67	7.33	7.33	+0.00
Medical health services manager	6.33	6.67	-0.33	7.53	6.17	+1.37	7.00	6.33	+0.67	6.33	6.67	-0.33	4.00	2.67	+1.33

A.7 Template-Guided Generation Examples

Figure 7 provides qualitative examples of template-guided generation under different user roles. Each example pairs a selected template slide with a generated slide for the same source paper. The examples show templates acting as concrete task-time design constraints over layout, palette, typography, and visual organization, while persona-conditioned generation adapts the slide content and emphasis.

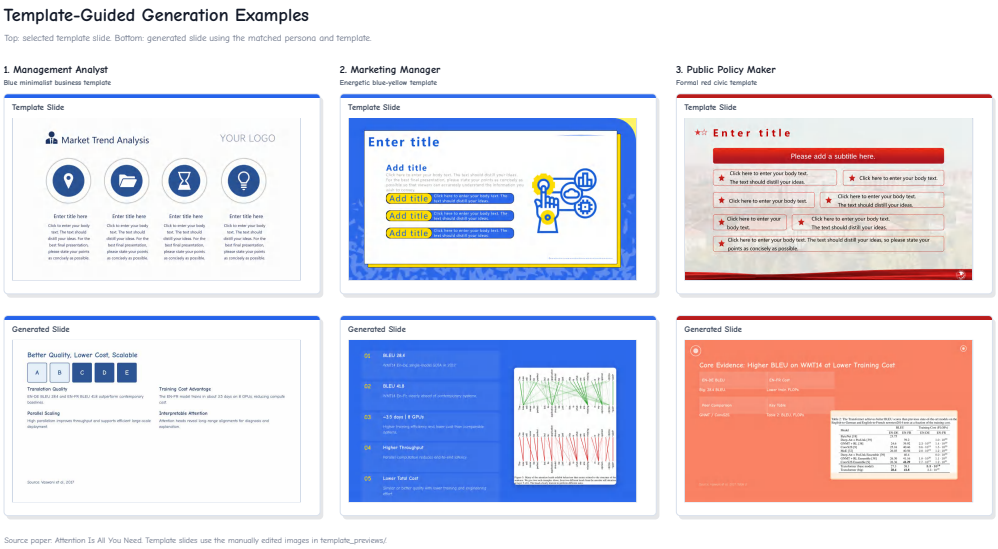


Figure 7: Template-guided generation examples: selected template slides (top) and matched persona/template generations (bottom).

A.8 Tool-Memory Pair-Level Details

The pair-level tool-memory results make the diagnostic matched-pair setting transparent. Table 8 reports the tool-memory and no-injection values for each matched pair, together with the pair verdict from the tool-memory perspective. The Gemini 3.1 Pro rows use a fixed graphic-designer hard-modify family, so the setting exposes hard cases rather than selecting pairs by favorable outcomes. The improvements are not from a single model: most pairs either improve or tie on closed-loop completion, and most improve Strict Verify. First Correct Edit time and Core Tool Time Ratio also usually decrease, although not monotonically in every pair. This supports the diagnostic conclusion that tool memory is associated with more reliable localized editing behavior, while retaining the caveat that the table summarizes a diagnostic matched-pair setting.

Table 8: Pair-level details for the diagnostic matched-pair tool-memory setting. Metric cells follow the same four metrics as Table 3 and report tool memory / no injection, with verdicts from the tool-memory perspective.

Model	Pair	Persona	Closed-Loop Completion ↑	Strict Verify ↑	First Correct Edit (s) ↓	Core Tool Time Ratio ↓
GPT-5	P1	Graphic Designer	1.000 / 1.000 (tie)	0.947 / 0.278 (win)	264.7 / 317.0 (win)	0.043× / 1.000× (win)
GPT-5	P2	Management Analyst	1.000 / 1.000 (tie)	0.579 / 0.471 (win)	158.0 / 151.3 (loss)	18.777× / 1.000× (loss)
GPT-5	P3	Postsecondary Teacher	1.000 / 0.000 (win)	0.412 / 0.133 (win)	93.3 / NA (NA)	0.502× / 1.000× (win)
GLM-5	P4	Graphic Designer	1.000 / 0.667 (win)	0.512 / 0.278 (win)	128.3 / 215.0 (win)	0.763× / 1.000× (win)
GLM-5	P5	Graphic Designer	1.000 / 1.000 (tie)	0.548 / 0.457 (win)	177.0 / 224.3 (win)	0.064× / 1.000× (win)
GLM-5	P6	Graphic Designer	1.000 / 1.000 (tie)	0.405 / 0.567 (loss)	282.3 / 1063.3 (win)	0.838× / 1.000× (win)
Gemini 3.1 Pro	P7	Graphic Designer	0.667 / 1.000 (loss)	0.481 / 0.183 (win)	340.5 / 199.7 (loss)	0.146× / 1.000× (win)
Gemini 3.1 Pro	P8	Graphic Designer	1.000 / 1.000 (tie)	0.366 / 0.242 (win)	329.0 / 2303.0 (win)	0.254× / 1.000× (win)
Gemini 3.1 Pro	P9	Graphic Designer	1.000 / 0.667 (win)	0.559 / 0.179 (win)	260.3 / 402.0 (win)	0.069× / 1.000× (win)

Aggregating these rows yields Table 3. Closed-Loop Completion and Strict Verify use arithmetic means over the three pairs per model family. First Correct Edit excludes pairs with unavailable no-injection edit time. Core Tool Time Ratio uses the geometric mean of the tool-memory/no-injection ratios.

Figure 8 illustrates the localized edit trajectory behind these process metrics.

Paired robustness check. Table 9 provides a non-parametric robustness check over the same nine matched pairs. For each metric, we compute pair-level wins and losses from Table 8 and apply an exact one-sided sign test after excluding ties and unavailable pairs. The strongest paired evidence appears on Strict Verify and Core Tool Time Ratio, while Closed-Loop Completion and First Correct Edit remain directionally favorable. Because the underlying setting is a diagnostic matched-pair

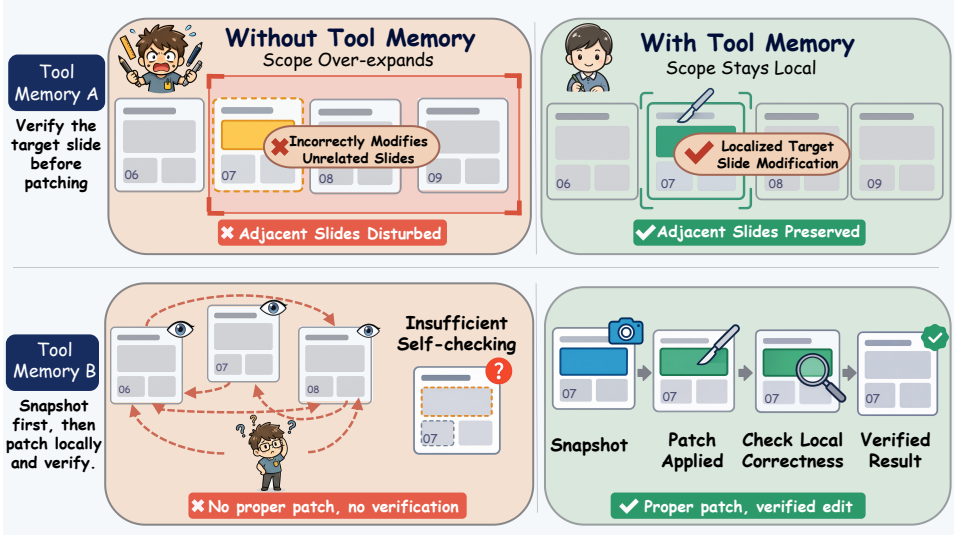


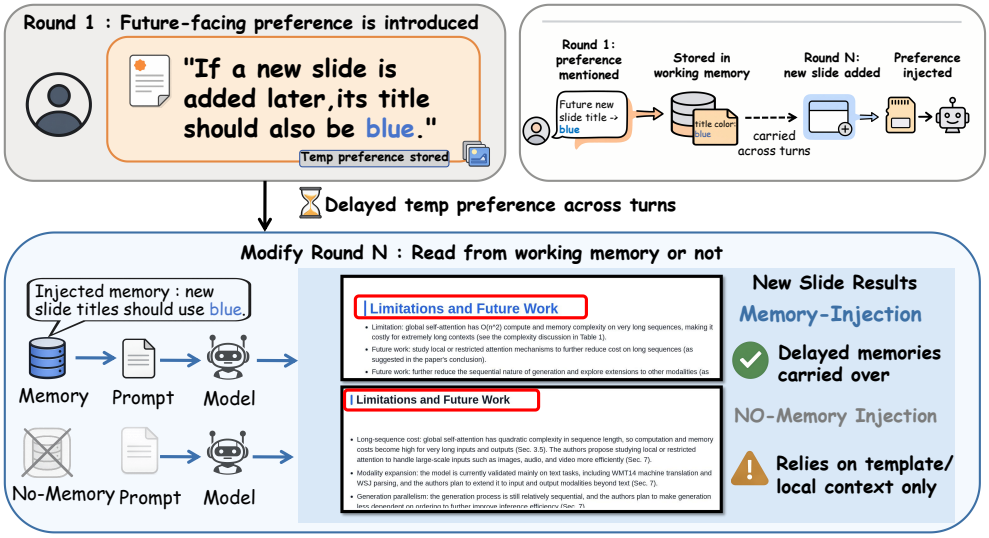
Figure 8: Illustrative qualitative trajectory for localized modify behavior. The example contrasts broad or incomplete editing behavior with a more constrained trajectory that inspects the target slide, applies a local patch, verifies the local change, and finalizes the revision. This figure is a process illustration for the diagnostic tool-memory setting rather than a separate quantitative result.

protocol, these checks are intended to characterize robustness within this controlled protocol rather than to claim significance over the full distribution of possible modify requests.

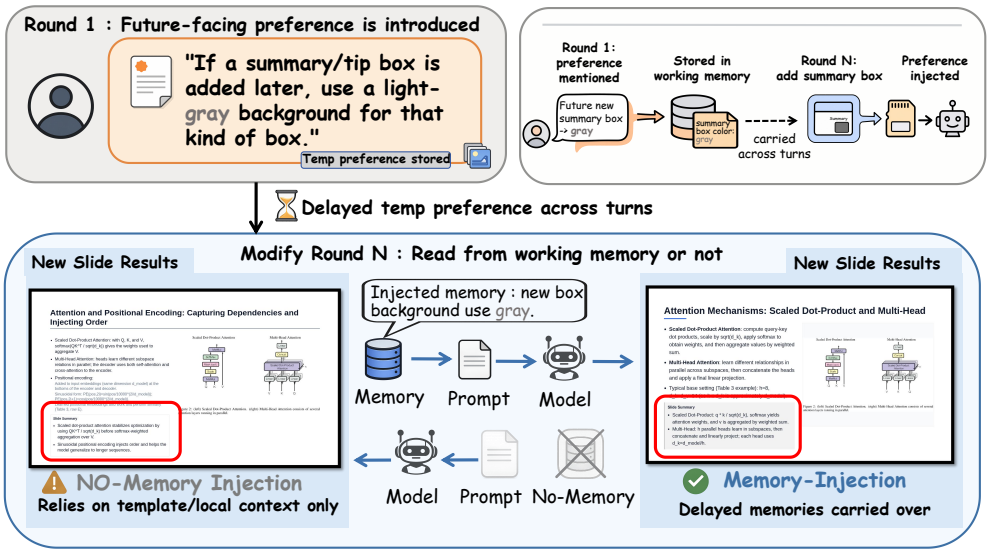
The W-L-T-NA column should be read as win–loss–tie–not-available counts from the tool-memory perspective. A win means that the memory-injected run is better than its no-injection counterpart under the metric direction: higher is better for Closed-Loop Completion and Strict Verify, while lower is better for First Correct Edit time and Core Tool Time Ratio. A tie means the two arms have the same value. NA means that one arm lacks a comparable value for that metric; for example, First Correct Edit is unavailable when a run never produces a verifiable first correct slide edit, so no latency comparison can be made. The sign test therefore uses only win/loss pairs and asks whether the number of wins is larger than would be expected under a 50–50 no-effect baseline.

Table 9: Paired robustness check for the diagnostic matched-pair tool-memory setting. Win/loss/tie/NA counts are computed from Table 8. The exact one-sided sign test uses only wins and losses, excluding ties and unavailable pairs.

Metric	W-L-T-NA	Valid W/L	Sign-test p	Interpretation
Closed-Loop Completion ↑	3-1-5-0	4	0.3125	Directional
Strict Verify ↑	8-1-0-0	9	0.0195	Paired evidence
First Correct Edit (s) ↓	6-2-0-1	8	0.1445	Directional
Core Tool Time Ratio ↓	8-1-0-0	9	0.0195	Paired evidence



(a) Title-color carryover.



(b) Summary-box style carryover.

Figure 9: Working-memory cases for delayed preference carryover. A future-facing rule is stated in an earlier turn and only becomes actionable after a later edit. Memory injection retrieves the stored rule at the trigger turn, whereas the no-memory setting relies mainly on local context.

A.9 Additional Qualitative Memory Cases

A.10 Existing Assets and Use Conditions

We use existing presentation-generation systems, hosted model APIs, visual representation models, and author-created templates for research comparison, generation, evaluation, and analysis. We do not redistribute third-party model weights, proprietary endpoints, commercial services, or third-party code as part of this submission. Table 10 summarizes the main existing assets used in the experiments and their use conditions.

Table 10: Existing assets and use conditions.

Asset category	Use and condition
Prior presentation systems	DeepPresenter, SlideTailor, and PPTAgent are cited and used for baseline generation, comparison, and evaluation alignment under their released code or project terms; third-party code is not redistributed.
Hosted LLMs	GPT-5, GLM-5, and Gemini 3.1 Pro are used for generation, modification, and LLM-as-judge evaluation through hosted APIs or institutionally provided endpoints under applicable provider terms; no model weights or endpoints are redistributed.
Visual representation models	DINOv2 is used for deck-level visual embeddings in diversity-style analysis. The standard DINOv2 model/code release is under Apache License 2.0; checkpoints are not redistributed.
Presentation templates	Author-created templates are used as task-time design constraints for template-conditioned generation examples and evaluation settings; no third-party templates are redistributed.
Evaluation materials	Profile-memory banks, prompts, generated decks, matched-pair definitions, and judge outputs are constructed for this paper's controlled evaluation. These materials are described for reproducibility. The source code is publicly available, and selected evaluation artifacts will be released when documentation and licensing checks are finalized.

A.11 Broader Impacts and Responsible Use

Personalized presentation agents can lower the cost of producing structured visual communication, helping users translate technical material into decks that better match their audience, role, and design preferences. Memory mechanisms may also reduce repeated instruction effort across authoring sessions and make multi-turn editing more efficient, especially when users need consistent style, recurring content emphasis, or localized revisions over an evolving deck.

At the same time, persistent personalization introduces risks. Stored preference profiles may encode sensitive user habits, organizational style constraints, or audience-targeting strategies, and incorrect memory consolidation may preserve outdated or unintended preferences across future tasks. Presentation generation can also be misused to produce persuasive but misleading materials, or to over-adapt content framing to a target audience. These risks suggest that personalized presentation agents should provide user-visible memory inspection, editing, and deletion controls; avoid storing unnecessary sensitive information; and separate stable user preferences from one-off task instructions. Our current work studies controlled generation and editing settings rather than deployment, and we leave stronger privacy controls, memory auditing, and real-user governance mechanisms to future work.