

# MemAura: Structured Context Memory for Personalized LLM Reasoning in Smart Environments

SIYUAN LIU, Hong Kong University of Science and Technology (Guangzhou), China

HUANGXUN CHEN\*, Hong Kong University of Science and Technology (Guangzhou), China

Personalized services in smart environments based on historical context have become an important demand. A promising direction is to leverage the strong reasoning abilities of Large Language Models (LLMs). However, enabling personalization requires persistent context memory. The key challenge lies in how to efficiently organize long-term device and sensor logs, and how to extract user behavior patterns and preferences from them. To address this challenge, we propose MemAura, a memory management system for smart environments. MemAura consists of three core modules: (1) Memory Graph Manager, which efficiently organizes contextual memory, supports fast retrieval, and provides user patterns and preference information to the LLM; (2) Periodic Pattern Predictor, which predicts periodic behavioral patterns and preferences at a given timestamp; (3) User Profiling Scheme, which transforms commands into explicit intent units and incrementally updates and forgets high-level profiles over time.

We evaluate MemAura against multiple baselines across two command datasets. With GPT-4o as the backbone, it achieves 100% accuracy and 38%–43% personalization rate. When deployed with lightweight local models, MemAura still achieves up to 95% accuracy, with an average personalization rate of up to 37.5%. However, when using the same lightweight LLM, vanilla only achieves 28%–56% accuracy, with a personalization rate of at most 21%. Vector-RAG achieves 54–83% accuracy and up to 25% personalization with lightweight models. While it outperforms vanilla, it still lags behind MemAura. The results show that MemAura maintains high accuracy and personalization across LLMs of different sizes, with practical token usage and latency. We also conduct a series of experiments to validate the robustness of MemAura. A user study with 8 participants further confirms its usability. The participants comprehensively experienced and evaluated its superior performance and service quality. Our work enables LLMs to better understand user intent and efficiently extract patterns and preferences from contextual memory, thereby making smart living spaces more efficient, context-aware, and user-centric.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**.

Additional Key Words and Phrases: Large Language Model, Personalized Memory Management

## ACM Reference Format:

Siyuan Liu and Huangxun Chen. 2026. MemAura: Structured Context Memory for Personalized LLM Reasoning in Smart Environments. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 10, 2, Article 55 (June 2026), 43 pages. <https://doi.org/10.1145/3810196>

## 1 INTRODUCTION

Enabling a truly intelligent living environment has long been a visionary pursuit, with intelligent assistants [1, 2, 13, 50] such as Alexa and Siri representing early efforts toward this goal. They handle explicit commands effectively, for example, “*Lower the living room air conditioner temperature by two degrees*”. However, they often

\*Corresponding author.

Authors’ addresses: Siyuan Liu, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong, China, sliu268@connect.hkust-gz.edu.cn; Huangxun Chen (corresponding author), Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong, China, huangxunchen@hkust-gz.edu.cn.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2474-9567/2026/6-ART55

<https://doi.org/10.1145/3810196>

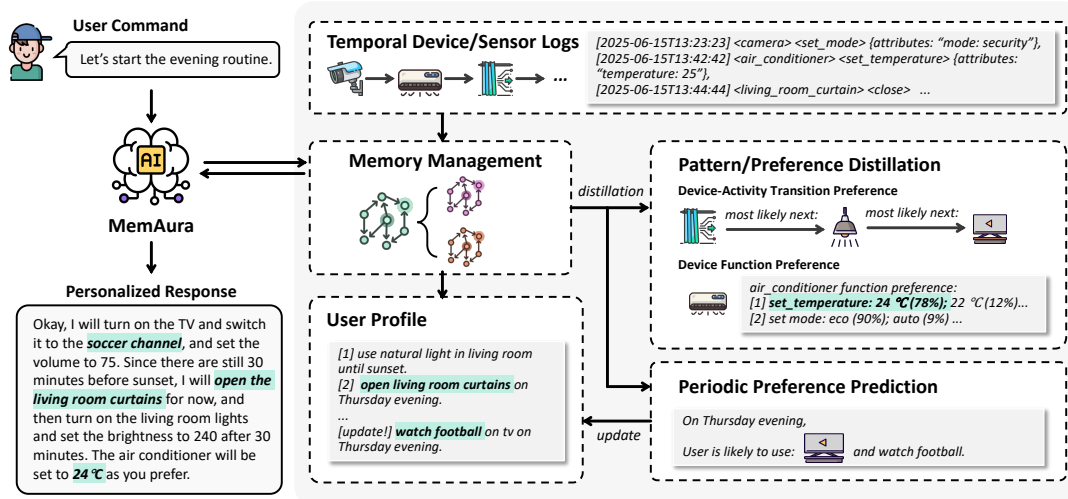


Fig. 1. Overview of a usage scenario. MemAura derives its memory from raw logs and is able to distill and predict behavior patterns and preferences to respond to user commands. After each interaction, MemAura further updates the user profile based on the response outcome, allowing the system to adapt to the evolution of the user's habits.

struggle with vague or compound requests such as “*Let's start the evening routine*”. Such commands neither specify the target devices nor the intended actions and lack a single definitive response, making it difficult for the system to accurately infer user intent. If a human butler were to respond to a similar request, he would infer the employer's specific needs based on an understanding of their preferences and daily habits. For instance, knowing that “*today is Thursday, and the employer usually watches football and drinks beer on evening*”, the butler would quickly identify the relevant devices and take actions according to the contextual semantics of the request. Such a response is inherently personalized, enabled by the butler's deep understanding of the employer, by the expense of employing one. Our goal is to develop an intelligent system capable of achieving a similar level of contextual reasoning and personalized adaptation, as shown in Fig. 1.

Recent advances in Large Language Models (LLMs) have demonstrated promising reasoning capabilities to enable personalized services. Several studies [31, 49] have explored this on individual personal devices such as smartphones, where user interaction histories are well organized and stored, enabling LLMs to extract preference patterns and reason about personalized responses. However, extending such personalization to an environment remains more challenging and less mature. Existing efforts [15, 23, 28, 42] primarily focus on interpreting user commands based on the immediate environmental state, with limited attention to effectively managing historical user interactions across multiple devices or systematically distilling user preferences to deliver more tailored responses. Based on our investigation, we identify three key challenges in enabling a smart environment capable of understanding user preferences and delivering personalized services in response to user commands.

- **CH1: Multi-user Multi-device Interaction History Management.** To deliver personalized services, it is essential to record user behaviors and operations in an organized manner. User interactions on a personal smartphone can be continuously collected from a single device, however, interactions within a smart environment involve multiple devices and exhibit inherent spatiotemporal characteristics. Moreover, such environments are often shared among multiple residents. Therefore, an effective scheme is required to

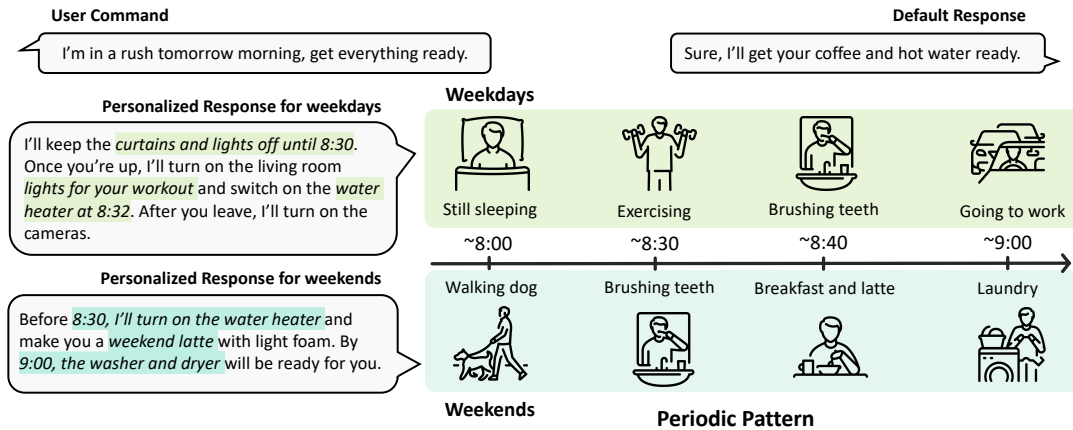


Fig. 2. A representative use case highlighting the importance of predicting periodic patterns is essential for delivering personalized smart services. With explicit predictions of a user's diverse periodic morning routines across different days, the system can disambiguate identical yet underspecified routine-preparation requests, enabling accurate responses and the proactive provisioning of appropriate resources for the corresponding set of routine events.

manage historical interaction traces across multiple users and devices, enabling LLMs to efficiently retrieve precise and contextually relevant information when needed.

- CH2: User-specific Preference and Pattern Distillation.** As illustrated in Fig. 2, when the system has explicit knowledge of a user's preferences and periodic behavioral patterns across contexts (e.g., weekdays vs. weekends), it can effectively disambiguate identical yet underspecified routine-preparation requests and deliver accurate responses. Specifically, on a weekday, it predicts that the user typically sleeps until 8:30a.m., exercises for 10min, and then washes up before leaving for work. The system can therefore proactively pre-provision resources for these routine events. In contrast, on a weekend, the predicted user behavior exhibits a different pattern, e.g., waking up earlier to walk the dog, washing up, having breakfast at home, and doing laundry, so the resources prepared should be correspondingly adjusted. Raw interaction logs collected in smart environments can, in principle, capture comprehensive user information. However, without effective distillation and an explicit prediction, they impose substantial burdens on downstream reasoning models to identify user preference and infer periodic patterns, and often still yield unsatisfactory performance. Our investigation as shown in Fig. 3 reveals a trade-off between LLM size and the precision of the provided user preference context: supplying raw logs directly (*vanilla* in Fig. 3) in requires a powerful LLM (e.g., an online API such as GPT-4o) to infer personalized responses, whereas providing distilled preference contexts (MemAura in Fig. 3) allows smaller LLMs to achieve comparable performance. Thus, it is important to effectively distill explicit user preference and periodic patterns from raw interaction logs to enable a locally deployable, more affordable, and privacy-friendly solution.
- CH3: User Profile Management.** Each user exhibits multiple dimensions of preferences, among which certain preferences could be stable and reflect long-term habits, while others could be temporary and may change rapidly. To enable LLMs to reason about and deliver personalized services effectively, an efficient scheme is required to manage these preferences and dynamically adjust their priorities over time in response to users' evolving behaviors and issued commands.

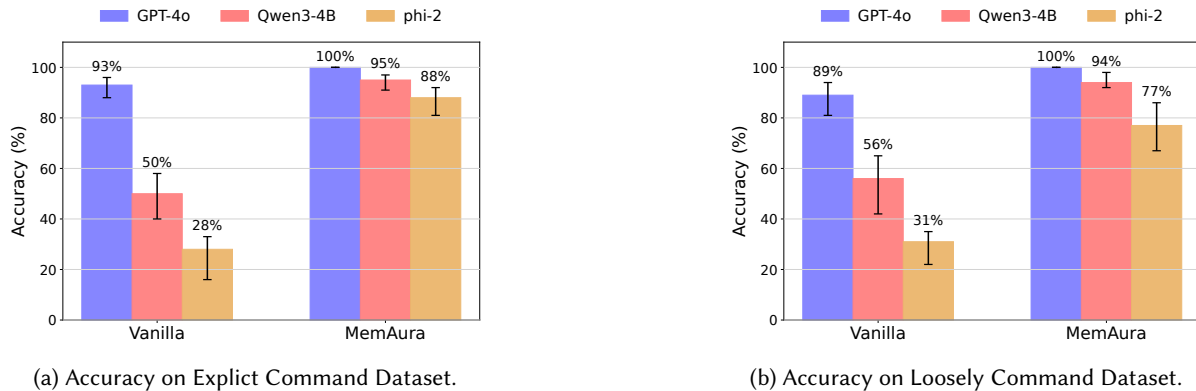


Fig. 3. A comparison of accuracy between vanilla (i.e., supplying raw logs directly) and MemAura. When employing a large-scale LLM such as GPT-4o, both vanilla and MemAura achieve consistently high accuracy. However, when lighter LLMs are used, the accuracy of vanilla drops sharply, whereas MemAura, benefiting from its well-structured and context-aware memory management, maintains an accuracy level comparable to that of large-scale LLMs.

To address these key challenges, we propose MemAura, a system that enables persistent and personalized context memory for LLM-based services in smart environments. To address **CH1**, MemAura features a **Memory Graph Manager**, a graph-based framework designed to efficiently organize multi-user, multi-device interaction histories as contextual input for LLM services. The graph structures all raw logs into a *User-Device Temporal Event Memory*, enabling the LLM to quickly locate, retrieve, and extract relevant historical information. Furthermore, two subgraphs, the *Device and Activity Transition Preference Graph* and the *Device Function Usage Graph* are derived to better represent user-specific patterns, facilitating LLM reasoning. To address **CH2**, MemAura introduces a **Periodic Pattern Predictor**, which extracts user-specific event sequences from the memory graph and applies a BERT-based model to predict a user's periodic preferences at a given timestamp, thereby providing more explicit and context-aware guidance for LLM reasoning. To address **CH3**, MemAura introduces a **User Profiling Scheme**, which consists of a *two-tier user preference queue*, a *user intent decomposer*, and an *interplay mechanism between intent understanding and profile updating*. Specifically, upon receiving a user command, the system decomposes it into a set of fine-grained intent units for sequential LLM reasoning. It then invokes the *Periodic Pattern Predictor* to obtain the user's current contextual preferences. After reasoning, preferences found to be useful are either inserted into the user preference queue or promoted in priority if they already exist in the queue. Over time, the queue's top- $k$  entries naturally form a dynamic user profile that captures the user's representative preferences, effectively guiding the LLM's personalized reasoning and adaptive decision-making.

Our main contributions are summarized as follows:

- (1) We identify the importance of managing user-specific interaction histories to enable smart environments and highlight the key challenges involved. Accordingly, we propose MemAura, a system that enables persistent and personalized contextual memory to enhance LLM-based services in smart environments.
- (2) MemAura mainly features a Memory Graph Manager for efficiently organizing multi-user, multi-device interaction sequences, a Periodic Pattern Predictor for inferring users' most probable preferences at a given timestamp, and a User Profiling Scheme for structuring and evolving users' various preferences over time.
- (3) We extend the existing living-space (smart-home) command dataset [23] by including richer commands that require personalization, and we further construct a new command dataset for working spaces (smart offices). Each dataset contains 40 explicit commands and 28 loose (underspecified) commands. In addition, we build a memory-context dataset to diversely emulate user interaction histories in smart environments.

The fidelity of our memory-context dataset is grounded in a real-world dataset [38]. We leverage an LLM to enrich its diversity and then replay the generated traces in the VirtualHome physics engine [40] for validation, retaining only those that satisfy real-world interaction logic and thereby ensuring the realism of our dataset.

- (4) We conduct a comprehensive evaluation to demonstrate the effectiveness and usability of MemAura. Using our dataset, we evaluate how the precise contextual information provided by MemAura enables LLMs of various sizes, particularly smaller ones, to perform satisfying personalized reasoning within acceptable latency and token cost. With GPT-4o as the backbone, MemAura achieves 100% task accuracy and a personalization rate of 38%–43%. When using lightweight LLMs such as Qwen3-4B and Phi-2, MemAura achieves up to 95% accuracy with an average personalization rate of up to 39%. However, the vanilla approach, directly feeding raw logs into the model, despite achieving reasonably strong performance with GPT-4o as the backbone, attains only 28%–56% accuracy and at most a 21% personalization rate when using lightweight LLMs. For another baseline vector-RAG [25], with GPT-4o, it achieves accuracy comparable to vanilla and up to 30% personalization, while lightweight models, it reaches 54–93% accuracy and 15–25% personalization rate. While vector-RAG outperforms the vanilla baseline, it still falls clearly short of MemAura. In addition, we design a series of stress tests to emulate real-world constraints, including cold-start scenarios with shorter memory lengths, device outages, and logging noise (erroneous log entries). The results show that MemAura remains robust under these conditions. Furthermore, our ablation study validates the effectiveness of each individual design component, and we also recruit a group of participants to assess the personalization quality and overall user experience.

## 2 MEMAURA DESIGN

### 2.1 Overview and Workflow

Fig. 4 presents an overview of MemAura. When a user command is received, MemAura first employs the Intent Decomposer (§2.4.2) to segment the instruction into fine-grained intent units, which are then processed sequentially. For each reasoning and service step, MemAura first queries the Memory Graph Manager (§2.2) to retrieve contextual information, primarily by identifying highly correlated subgraphs as relevant context. In addition, the system invokes the Periodic Pattern Predictor (§2.3) to obtain user-specific preference predictions for the current moment. Finally, LLM also incorporates a high-level User Profile (§2.4.1), distilled from past interactions, to generate adaptive and personalized responses for each intent unit. The responses are then integrated to produce the final comprehensive output. After the reasoning process, a post-hoc procedure (§2.4.3) updates the two-tier user preference queue. Preferences identified as meaningful are either inserted into the queue or promoted to a higher position if they already exist, while outdated entries may be removed as the user’s habits evolve. Consequently, the leading top- $k$  entries in the queue dynamically evolve over time, serving as an effective reference for subsequent reasoning and personalization.

### 2.2 Memory Graph Manager

Daily interaction logs in smart environments capture the preferences and habits of individual residents. Unlike conversation-based user interaction data, interactions with smart devices inherently exhibit spatiotemporal characteristics shaped by the user’s physical context and activity patterns. Motivated by this observation, we design the Global Memory Graph, as illustrated in Fig. 5, an expressive and efficient structure that organizes raw device logs to enable fast retrieval and contextual access. We further derive two subgraphs as shown in Fig. 6 to explicitly characterize user behavior patterns, enabling more efficient querying and utilization. Memory Graph Manager provides convenient interfaces for LLMs to retrieve precise contextual information, facilitating the generation of personalized responses.

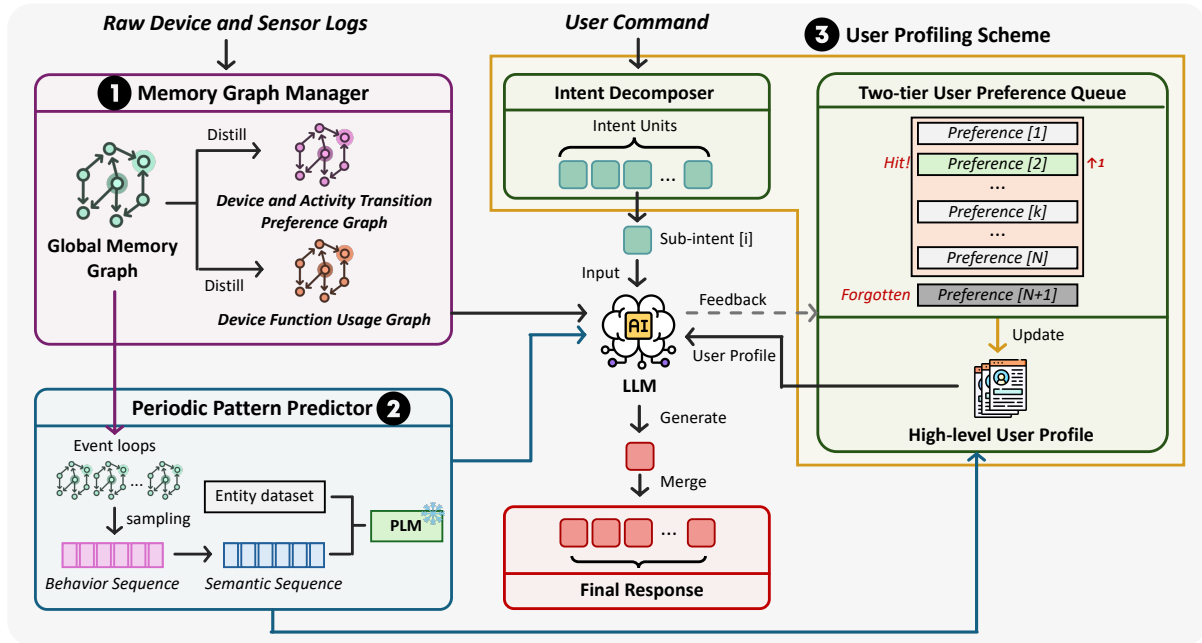


Fig. 4. Overview of MemAura System. The system consists of three core modules: (1) Memory Graph Manager, which efficiently organizes contextual memory, supports fast retrieval, and provides user patterns and preference information to the LLM; (2) Periodic Pattern Predictor, which predicts periodic behavioral patterns and preferences at a given timestamp; (3) User Profiling Scheme, which consists of a two-tier user preference queue and a user intent decomposer. It incrementally updates and forgets high-level profiles over time.

2.2.1 *Global Memory Graph Construction.* We propose a directed and labeled heterogeneous graph  $G$ , constructed from logs collected by smart devices and sensors. It includes eight vertex classes:

$$V = \bigcup_X V_X, X \in \{U, D, F, AC, E, I, AR\}$$

These vertex classes denote users (e.g., Alice), devices (e.g., coffee machine), sensors (e.g., kitchen\_entry\_sensor), functions (e.g., turn\_on), activities (e.g., cooking), events (e.g., Alice cooks, Alice turns on the coffee machine), intent (e.g., prepare for sleep in bedroom) and areas (e.g., kitchen), respectively. Thus, we define node attribute function  $\ell_V(V_X) = X$ . For events, the following attribute functions map each event to its timestamp in  $\mathbb{T}$  and its sequence number in  $\mathbb{N}$ , respectively:

$$\tau : V_E \rightarrow \mathbb{T}, \sigma : V_E \rightarrow \mathbb{N}$$

The graph edges  $E$  have nine classes and associated labels  $\ell_E$  as shown in Table 1. The resulting graph  $G = (V, \ell_V, \tau, \sigma, E, \ell_E)$  unifies static entities, such as devices, devices' functions, residents, and routine activities, and dynamic event node expansion along with their interplay.

Fig. 5 presents a partial view of the global graph  $G$ , showing several of its initial Event nodes as examples. Each Event node records an interaction event. For example, *event1* in the figure represents the event “Alice turns on the kitchen light.” Each Event node forms a unique directed cycle, either  $Event \rightarrow Device \rightarrow Function \rightarrow User$  or  $Event \rightarrow Activity \rightarrow User$ , which records every participant and detail associated with that event. For example, *event1*

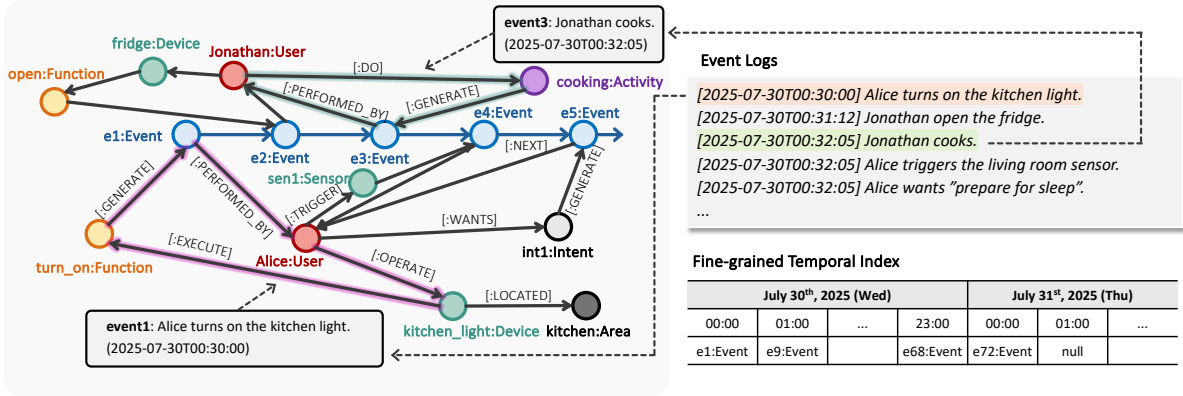


Fig. 5. An example of the structure of the global memory graph. The global graph stores the long-term interaction logs shared among multiple residents. Each Event node represents an action recorded by a device or sensor, and forms a unique directed cycle, thereby capturing all participants involved in that event. A Temporal Index is employed to enable fast retrieval of specific events.

Table 1. Edge Types in Global Memory Graph.

Notations	Specifics	Label
$E_{OP}$	User $\rightarrow$ Device	OPERATE
$E_{NX}$	User $\rightarrow$ Sensor	TRIGGER
$E_{LO}$	(Device $\cup$ Sensor) $\rightarrow$ Area	LOCATED
$E_{EX}$	Device $\rightarrow$ Function	EXECUTE
$E_{DO}$	User $\rightarrow$ Activity	DO
$E_{DO}$	User $\rightarrow$ Intent	WANTS
$E_{GN}$	(Function $\cup$ Activity $\cup$ Sensor) $\rightarrow$ Event	GENERATE
$E_{PB}$	Event $\rightarrow$ User	PERFORMED_BY
$E_{NX}$	Event $\rightarrow$ Event	NEXT

corresponds to the unique directed cycle  $e_1: Event \rightarrow kitchen\_light: Device \rightarrow turn\_on: Function \rightarrow Alice: User \rightarrow e_1: Event$ . By connecting these nodes with appropriate prepositions, the event can be expressed as the natural language description “Alice turns on the kitchen light.” Since devices and sensors in a smart environment may be distributed across different physical spaces, the graph  $G$  also includes a supernode called *Area*. Each Device or Sensor node is linked to its corresponding area via an edge  $E_{LO}$ , distinguishing their spatial distribution. In a nutshell, our graph design is LLM-friendly. All nodes and edges are semantically named, and the event cycles can be converted into natural language descriptions containing complete contextual information, enabling LLMs to efficiently access and interpret the underlying data.

**2.2.2 Fine-grained Temporal Index.** Over the graph  $G$ , we incorporate a two-level index  $I: \mathcal{D} \times \mathcal{H} \rightarrow V_E \cup \perp$  to support efficient daily/hourly event retrieval. Specifically,

$$I(d, h) = \arg \min_{v_e} \{\tau(e) \mid \tau(e) \in [T_{d,h}, T_{d,h+1}]\}$$

$$= \perp, \quad \text{if no such event } e \text{ exists}$$

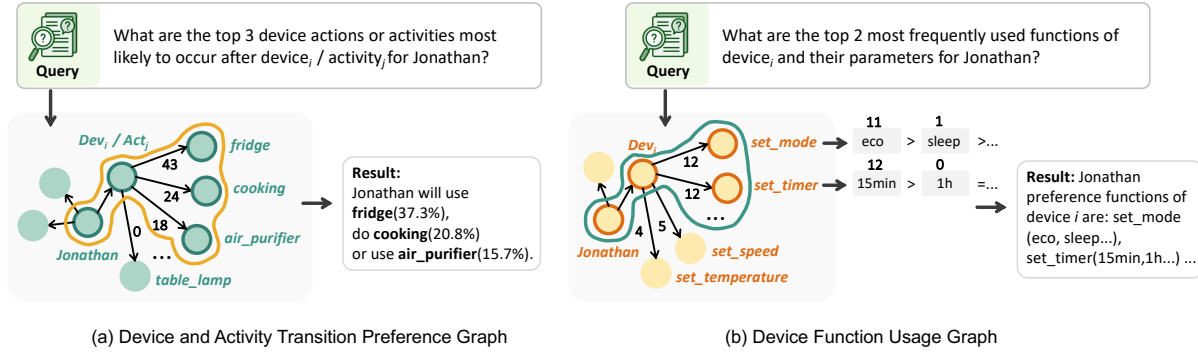


Fig. 6. An example of the structure of two derived subgraphs. The two subgraphs are generated from the global graph. The left side is Device and Activity Transition Preference Graph, identifies the most probable next device operation or activity on statistical transition probabilities. The right side subgraph, the Device Function Usage Graph, efficiently captures the functions and parameter configurations that a user most frequently employs when interacting with Device  $i$ .

which maps a given date  $d$  and hour  $h$  to the first event in this duration, where  $T_{d,h}$  denotes the starting timestamp of hour  $h$  on date  $d$ .

**2.2.3 Device and Activity Transition Preference Graph.** Each user’s event profile can be represented as a sequence of device operations, sensor logs, and activity executions. Among these, device operations and activity executions contain abundant contextual details, that encapsulate rich behavioral patterns and user preferences. While the global graph  $G$  is comprehensive, it can sometimes be overly detailed, making it difficult to directly infer transition probabilities between different operations and activities for a specific user. To enable the LLM to efficiently access such statistical insights into user behavior patterns, we derive a **Device–Activity Transition Preference Graph**  $G_T^u$  from the global graph  $G$ .

As illustrated in Fig. 6(a), this graph mainly includes device vertices  $V_D$  and activity vertices  $V_{AC}$ . We examine each edge  $\{e_m, e_n\} \in E_{NX}$  and identify the associated device or activity nodes connected to the endpoints. The definition is as follows:

$$\pi(e_m) = v_i, ; \pi(e_n) = v_j$$

where  $v_i, v_j \in V_D \cup V_{AC}$ . We then establish a directed edge  $(v_i, v_j) \in E_T^u$  and increment the associated edge attribute  $w$  by 1 to reflect the transition frequency. The resultant graph  $G_T^u = (V_D \cup V_C, E_T^u, w_T^u)$  summarizes the residents’ historical activity transition patterns, enabling the LLM to more easily extract meaningful information.

Fig. 6 illustrates an example of graph  $G_T^u = (V_D \cup V_C, E_T^u, w_T^u)$ , and its usage, specifically, how the system can efficiently answer an LLM query such as “What are the top 3 device actions or activities most likely to occur after device  $i$  and activity  $j$  for Jonathan?” We first identify all device  $i$  or activity  $j$  nodes connected to the Jonathan node, then locate the corresponding Device or Activity nodes involved in each adjacent event. The occurrences are counted and ranked to obtain the most frequent transitions. As shown in this example, after using device  $i$ , Jonathan most frequently interacts with the fridge (43 times), begins cooking (24 times), or uses the air purifier (18 times). Such user preferences can be interpreted as individualized probability distributions. Therefore, the nodes with the highest transition probabilities represent Jonathan’s Top-K preferences for subsequent actions after operating device  $i$ .

**2.2.4 Device Function Usage Graph.** Different users may exhibit distinct preferences over various device functions, even for the same device. To make such patterns more pronounced for LLM reasoning, we extract a user-specific

device usage weighted graph  $G_F^u$  from the global graph  $G$ , as shown in Fig. 6(b). This graph primarily consists of device vertices  $V_D$  and function vertices  $V_F$ . We consolidate all edges  $\{v_d, v_f\} \in E_{EX}$  with the same endpoints and assign the edge attribute  $w$  as the total number of such edges, reflecting how many times user  $u$  has invoked function  $f$  on device  $d$ . The attribution field within the metadata of each Function node is also weighted and counted after each use, allowing the system to record more fine-grained preference settings.

For example, as illustrated in Fig. 6, when LLM queries Jonathan’s preferred functions and parameters for device  $i$ , it can obtain this information directly from graph  $G_F^u$ . Specifically, all edges connecting device  $i$  to its associated Function nodes are first ranked according to their weights. Then, the parameters within each Function node are sorted by their own weighted frequencies. The resulting rankings represent Jonathan’s Top-K preference settings for device  $i$ . In this example, the preferred configuration is setting the temperature to 24 °C and adjusting to eco mode.

### 2.3 Periodic Pattern Predictor

Besides historical data recorded in above graph structure, to facilitate the LLM in generating personalized responses, it is highly beneficial to know what the user is most likely to do at the moment a user command is issued. This prediction task can be approached in multiple ways, for instance, by feeding all contextual information into a LLM. However, considering inference cost, latency, and data privacy, we instead formulate it as a **masked token prediction task** and train a lightweight Periodic Pattern Predictor to perform the prediction efficiently. An overview of our design is illustrated in Fig. 7.

**2.3.1 Predictor Training.** Our basic idea is to adopt BERT [7] as the predictor backbone, leveraging its strong capability to capture semantic dependencies. We then extract temporal–semantic information from the global graph  $G$  to construct a domain-specific dataset for fine-tuning the base BERT model.

Specifically, as previously described, each event in the global graph corresponds to a specific timestamp and encapsulates a complete set of entities and relations. Thus, we segment the global graph temporally into a sequence of historical subgraphs  $G_{t_1:t_n}$ , each representing events occurring within the continuous time window from  $t_1$  to  $t_n$ . From each subgraph, we identify the entities requiring prediction (Device, Activity, and Intent) and compile them into an entity list. Next, we perform frequency-based sampling over event cycles, assigning higher sampling weights to entities that occur more frequently in the list. This process yields a time-ordered sequence in which each event is represented as a quadruple (*user, relation, entity, timestamp*). We then transform each quadruple into a semantically coherent sentence  $s_i$ , formulated as:

$$s_i = \text{time, user relation entity}$$

Multiple sentences  $s_i$  are then concatenated in chronological order, with special tokens [CLS] and [SEP] added at the beginning and end, respectively, to form a complete sample  $S_q$  for BERT:

$$S_q = [\text{CLS}] s_1, s_2, \dots, s_i. [\text{SEP}]$$

We apply a sliding time window to generate multiple such sample  $S_q$ , forming a domain-specific dataset used to fine-tune the base BERT model to obtain our Periodic Pattern Predictor.

**2.3.2 Predictor Inference Process.** As illustrated on the right side of Fig. 7, when a sequence of historical events is known and the goal is to predict *what Jonathan will do at a specific time*, this can be formulated as a query  $q$ . We define three types of such queries as follows:

$$\begin{aligned} q_w(\text{time,user}) &= \text{time, user wants ?} \\ q_o(\text{time,user}) &= \text{time, user operates ?} \\ q_d(\text{time,user}) &= \text{time, user does ?} \end{aligned}$$

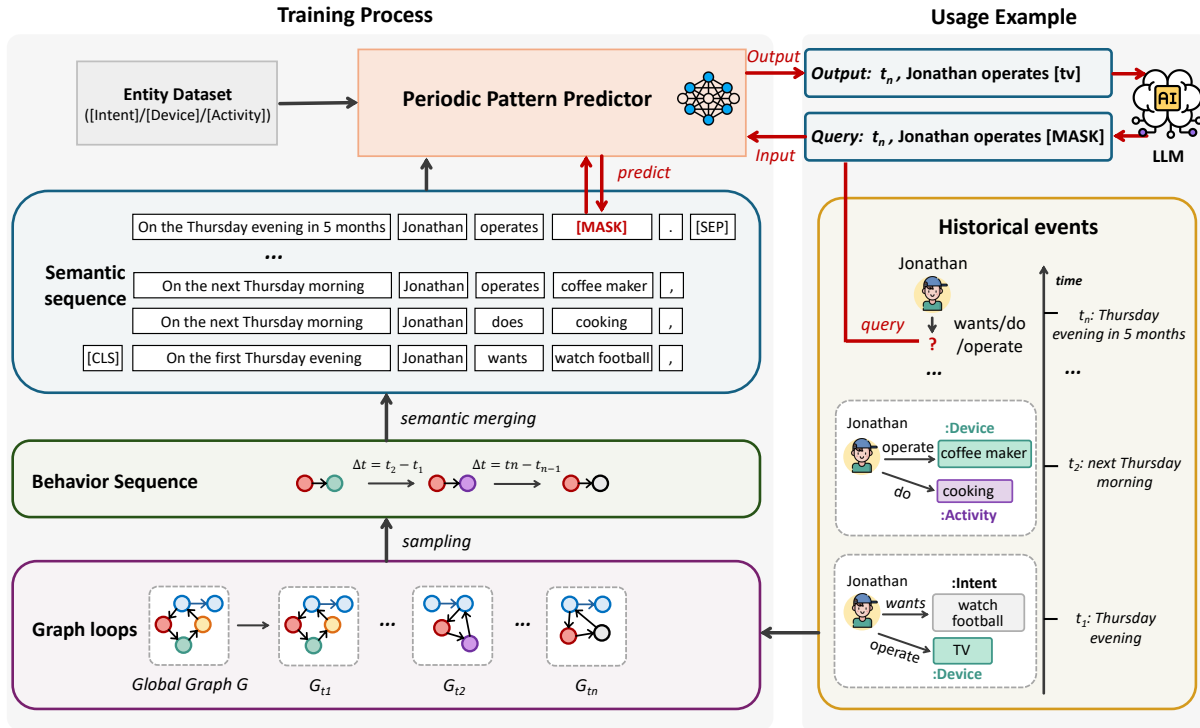


Fig. 7. Overview of the Periodic Pattern Predictor. We perform frequency-based sampling over the behavioral cycles in the global graph and compute the time intervals between sampled events to construct behavioral sequences. These sequences are then organized into semantically coherent sentences, forming semantic sequences. The user’s potential action at a specific time point is treated as the masked entity to be predicted within the semantic sequence. We extract all possible entities and fine-tune them together with the semantic sequence by BERT. Finally, the Periodic Pattern Predictor can make the prediction.

The trained Predictor can process a query  $q$  to make a prediction, denoted as  $P_t^{S,q}$ :

$$P_t^{S,q} = \text{Predictor}(S_q)$$

The prediction output is regarded as a user-specific periodic pattern, indicating what the user is likely to do at a given time. Compared to invoking an LLM for such reasoning, our predictor is compact, efficient, and achieves high accuracy as well. Moreover, since the entire process can be affordable in local servers, it mitigates risk of privacy leakage using cloud services.

## 2.4 User Profiling Scheme

**2.4.1 Two-tier User Preference Queue.** From historical behavior records, multiple periodic patterns can be inferred for a specific user. However, human behavior and preferences often evolve dynamically. For example, recent history in the global graph may show that Jonathan usually watches football on Thursday evenings, while also enjoying natural light and fresh air. Therefore, an effective structure is needed to capture various aspects of user preferences, manage their priority and recency, and update them dynamically over time. Thus, as illustrated in Fig. 8, we design a two-tier user preference queue composed of multiple entries that record user preferences. Meanwhile,

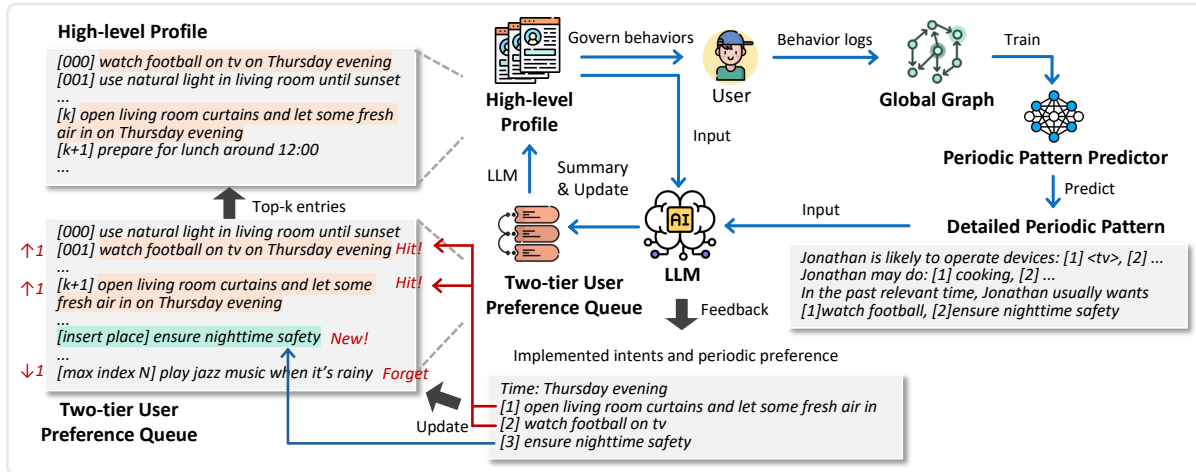


Fig. 8. Overview of the Two-tier User Preference Queue. A user's preferences cause certain behaviors to occur frequently, which results in these behaviors becoming increasingly represented in the global graph, and thus more likely to be predicted by the Periodic Pattern Predictor. These preferences also manifest in repeated command intents. After the LLM completes its reasoning, it provides feedback indicating which preference items were adopted in the response. The Two-tier User Preference Queue maintains a cache-like structure: each time the LLM reports a used preference item, that item is added to the queue. If the item already exists, its priority is increased. The queue has a fixed size, so items that have not been accessed for a long time are removed, that is, they are forgotten. The top- $k$  remaining items after updating form the effective high-level profile for the next round of inference. The update process is a background task performed by the LLM during system idle time.

the top- $k$  entries are regarded as a high-level profile to characterize the user's representative preferences, which guide the LLM's personalized reasoning and adaptive response generation.

**2.4.2 User Intent Decomposer.** To update the user preference queue accurately, we argue that intent understanding during system runtime is an essential component. To achieve precise intent interpretation, we design an Intent Decomposer module, particularly aimed at handling ambiguous commands such as "Let's start the evening routine" mentioned earlier. This module decomposes a user command into explicit intent units. We employ a carefully designed few-shot prompting strategy, as illustrated in Fig. 9, to guide the LLM in performing effective initial intent decomposition. Given the strict output constraints, Intent Decomposer rarely generates the impractical results observed in our evaluation. Nevertheless, we provide users with flexible options to audit and refine the system's inferences. As illustrated in Fig. 9, Intent Decomposer explicitly outputs intent units, then users may execute all units or only a subset, or request a revision of a specific unit. After user confirmation, the reasoning LLM processes the approved units in parallel and then merges the results into the final output.

**2.4.3 Interplay Mechanism between Intent Understanding and Profile Updating.** As illustrated in Fig. 8, in MemAura, user intent reasoning drives the updating of the user profile, while a well-maintained profile, in turn, facilitates more accurate and personalized LLM reasoning.

Specifically, after each intent inference, the system reflects on the process and identifies the preference entries that contributed to generating the personalized response, either from the existing high-level profile or newly discovered preferences provided by the trained Periodic Pattern Predictor. The system then updates their order accordingly in the user preference queue. Newly identified preferences are inserted at a predefined place (*insert\_place*) near the middle of the queue to keep the representation timely and reflective of recent behavior.

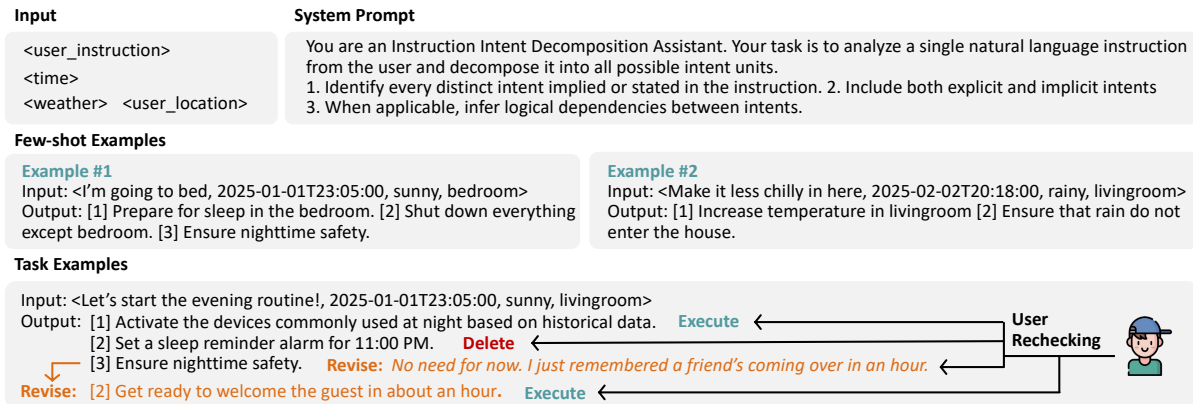


Fig. 9. Prompt Design for Intent Decomposer. In some cases, an API-based LLM may be employed to support this component; therefore, to minimize the risk of privacy leakage, the input data are kept concise and free of any personal information. After Intent Decomposer outputs the intent units, the user can audit them, deciding which units to execute or discard, and which to revise by prompting the LLM to regenerate, thereby balancing automation with user control.

The queue has a maximum capacity ( $max\_index$ ). Adding a new item pushes all subsequent entries down, and those exceeding the limit are removed, representing forgotten preferences due to long inactivity. For existing entries, their priority is raised by one position in the queue. The top- $k$  entries after each update form the next active high-level profile used in subsequent LLM personalized reasoning. The parameters  $k$ ,  $insert\_place$ , and  $max\_index$  are tunable hyperparameters.

The rationale behind this interplay mechanism between intent understanding and profile updating is inspired by the human short-term memory system: only a limited number of entries can be retained. Those that are seldom reactivated gradually fade away, while those frequently activated persist over time. In our scenarios, if certain behaviors are repeatedly performed by the user, they are captured in the global memory graph, allowing our predictor to extract them as preferences. Similarly, if specific commands are frequently issued by the user, they indicate potential preferences and are gradually promoted upward in the queue to form part of the high-level profile. In our design, only preferences that persist and accumulate over time are considered part of the high-level profile. It is worth noting that the system performs preference updating as a background task, introducing no additional latency during inference.

### 3 EVALUATION

In this section, we conduct a detailed evaluation of MemAura. We begin by introducing the command dataset and the context memory dataset used in our experiments. We then describe the evaluation metrics and baseline methods, as well as the deployment configuration. Finally, we present experimental results measuring the accuracy and overall performance of MemAura. We aim to answer the following research questions:

**RQ1:** Is there a trade-off among MemAura's accuracy, token consumption, and latency, with state-of-the-art LLMs?

**RQ2:** Does MemAura maintain high accuracy when using lightweight local models?

**RQ3:** When MemAura fails, which type of error occurs more frequently?

**RQ4:** Does MemAura provide significant personalization services?

**RQ5:** To what extent is the effectiveness of MemAura influenced by its individual modules?

**RQ6:** Is MemAura robust under real-world constraints, including device failures, missing or corrupted logs, and cold-start scenarios where context memory is limited?

### 3.1 Datasets

We use two types of datasets in our evaluation: one command dataset for assessing the system’s responses to different user commands, and one context memory dataset that simulates user behavior in the environment.

- **Command Dataset.** We consider two representative smart-environment settings: a living-space scenario (smart home) and a workspace scenario (smart office), which exhibit diverse user query needs. For smart home scene, we adopt a subset of smart-home command datasets from prior work [23], consisting of **40 explicit commands** and 18 loosely defined commands. To better assess personalization performance, we further design 15 additional loosely defined commands, leading to a total of **28 loosely defined commands** when combined with those from [23]. For the smart-office scenario, We collect common workplace needs from an open dataset [19] and expand them to construct a command dataset tailored to smart-office settings. The complete command datasets for both scenarios are provided in Appendix A.
- **Context Memory Dataset.** We utilize the corresponding environment layout (environment *h3* in [23]) to ensure fair comparison. To populate the context memory in MemAura, we construct a multi-resident shared event-log dataset that diversely emulates user interaction histories in smart environments, comprising 360 distinct 24-hour records in total. The fidelity of our memory-context dataset is grounded in a real-world dataset [38]. We leverage an LLM to enrich its diversity and then replay the generated traces in the VirtualHome physics engine [40] for validation, retaining only those that satisfy real-world interaction logic and thereby ensuring the realism of our dataset. The specific dataset sampling and simulation procedures are documented in Appendix B.

### 3.2 Evaluation Metrics

We propose a comprehensive set of metrics to evaluate the performance of MemAura. For clarity, we define the notation as follows: the command dataset is denoted as  $C_{\text{com}}$ , and the context memory dataset as  $C_{\text{mem}}$ . Let  $D$  be the set of all devices. For each device  $d \in D$ , let  $F_d$  denote the set of function nodes associated with  $d$ . For each user  $u$  and command  $com \in C_{\text{com}}$ , there exists a minimal set of devices  $D_{\text{goal}}$  and a corresponding set of functions  $F_{\text{goal}}$  that together satisfy the intended task, sets of  $D_{\text{res}}$  and  $F_{\text{res}}$  that an LLM’s response, and a set of devices  $D_{\text{pre}}$  and functions  $F_{d_{\text{pre}}}$  that match the user’s preferences. Based on these definitions, our evaluation metrics cover the following dimensions:

- **Device Error(DE).** When the system produces a final response, an error is considered to have occurred if it selects an incorrect device or fails to include a required target device.

$$DE = \frac{1}{|C_{\text{com}}|} \sum_{com \in C_{\text{com}}} de(com, D_{\text{goal}}, D_{\text{res}}). \quad de(com, D_{\text{goal}}, D_{\text{res}}) = \begin{cases} 1, & \text{if } \exists d \in D_{\text{goal}} \text{ such that } d \notin D_{\text{res}} \\ 1, & \text{if } \exists d \in D_{\text{res}} \text{ such that } d \notin D_{\text{goal}} \\ 0, & \text{otherwise.} \end{cases}$$

- **Function Error(FE).** On the other hand, when no Device Error occurs but the system selects an incorrect function in its final response, a Function Error is considered to have occurred.

$$FE = \frac{1}{|C_{\text{com}}|} \sum_{com \in C_{\text{com}}} fe(com, F_{\text{goal}}, F_{\text{res}}). \quad fe(com, F_{\text{goal}}, F_{\text{res}}) = \begin{cases} 0, & \text{if } de(com, D_{\text{goal}}, D_{\text{res}}) = 1 \\ 1, & \text{if } \exists d \in D_{\text{goal}} \text{ such that } F_{\text{res}}(d) \notin F_{\text{goal}}(d) \\ 0, & \text{otherwise.} \end{cases}$$

- **Fail Rate(FailR).** The occurrence of either type of error indicates a task failure.

$$FailR = DE + FE.$$

- **Accuracy(Acc).** When neither type of error occurs, the task is considered successful.

$$Acc = 1 - FailR$$

- **Latency.** We measure the total response latency by recording the time from when the user issues a command to when the LLM returns the final output.
- **Token Consumption.** For each module that invokes the LLM, we also calculate the total token consumption, which allows us to estimate the monetary cost of using the LLM to complete tasks in realistic deployment scenarios.
- **Personalization Rate(PR).** To evaluate whether the system’s response is sufficiently creative, we consider personalization to have occurred when the output includes device selections or function choices that align with the user’s preferences.

$$PR = \frac{1}{|C_{com}|} \sum_{com \in C_{com}} pr(com). \quad pr(com) = pr(com) = \begin{cases} 1, & \text{if } (D_{res} \cap D_{pre}) \neq \emptyset \\ & \text{or } \exists d \in D_{goal}, F_{res}(d) \in F_{d_{pre}}, \\ 0, & \text{otherwise.} \end{cases}$$

### 3.3 Baseline and Backbones

We compare our system against three baselines: (1) the prior work **Sasha** [23], (2) **vanilla**: Sasha [23] augmented with raw event logs in Context Memory Dataset, (3) **vector-RAG (Retrieval-Augmented Generation)** [25]: It works by storing a corpus as dense vector embeddings in a vector database. For each user query, it computes a query embedding, retrieves the most semantically similar chunks via nearest-neighbor search, and feeds the retrieved content into LLM as context. We replace the context memory management module in MemAura with a vector-RAG pipeline. We use the embedding model `sentence-transformers/all-MiniLM-L6-v2` and empirically determine its optimal hyperparameters in our evaluation: the maximum length of each log is 180, the maximum embedding input length is 256, and the batch size is 32. For each intent unit, we first fetch 3K candidates and then retrieve the top- $k$  results with  $k=10$ .

In addition, to examine the impact of different LLM types on system performance, we evaluate all baselines and MemAura using the following models: **GPT-4o** [34], a large closed-source model provided by OpenAI and accessed via API; **phi-2** [32], a lightweight model from Microsoft with 2.7 billion parameters (approximately 3B) deployed locally; and **Qwen3-4B** [46], a recent lightweight open-source causal language model (specifically, Qwen3-4B-Instruct-2507) with approximately 4B parameters, also deployed locally.

### 3.4 Implementation

In the Memory Graph Manager, we use Neo4j to construct and manager the global graph, then the LLM can issue query statements to quickly locate the relevant information within it. In the Periodic Pattern Predictor, we use bert-base-cased [7] as the pre-trained model, with all text processed using the standard tokenizer. Considering that bert-large-cased supports a maximum input length of 512 tokens, only the most recent 512-token behavioral sequence is retained to ensure compliance with the model’s input constraints. During batch construction, we apply Bernoulli masking to the sequence with a masking probability of  $p_{mlm} = 0.15$ . The optimization settings are as follows: learning rate  $5 \times 10^{-5}$ , batch size of 8, and 3 training epochs. All experiments are conducted on an NVIDIA A6000 GPU.

In the following, we will demonstrate the detailed evaluation results to answer RQ1-RQ5.

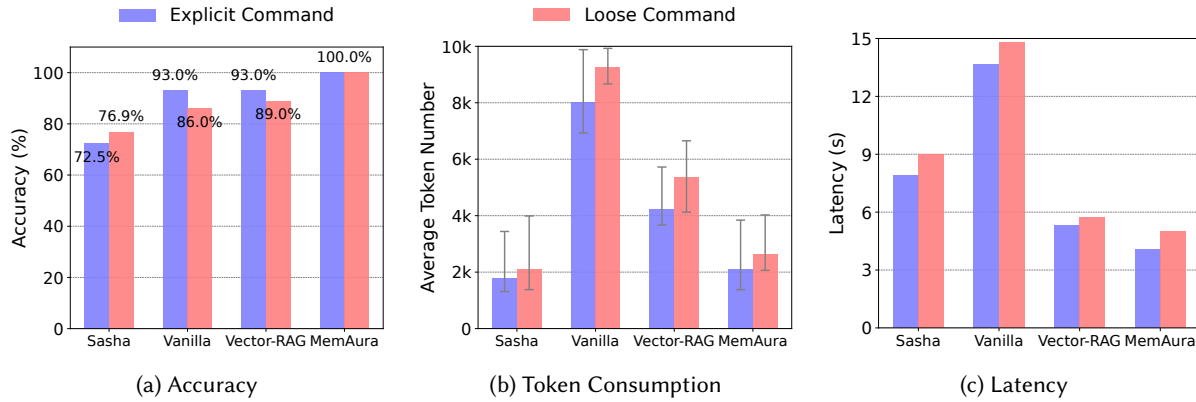


Fig. 10. Overall Performance against baselines using GPT-4o as the backbone model. (a) shows the results for accuracy, (b) presents the comparison of token consumption, and (c) reports the response latency.

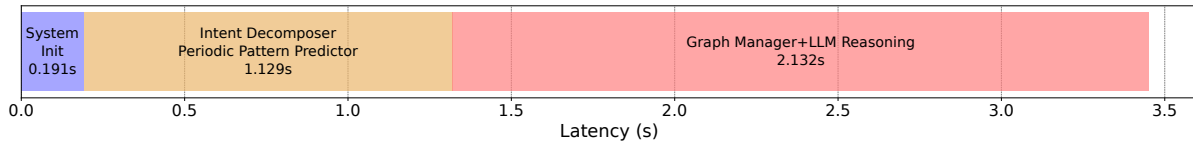


Fig. 11. Latency breakdown of MemAura using GPT-4o as the backbone. We run components without sequential dependencies in parallel. Overall, the latency can be grouped into three stages to achieve the best practical performance.

### 3.5 Overall Performance with large LLM backbone (RQ1)

We first conduct an experiment with large LLM backbone to compare MemAura with the baselines using three key quantitative metrics, including Accuracy, Token Consumption, and Latency. All experiments in this stage use GPT-4o as the backbone. Fig. 10 presents the results.

In terms of accuracy, results show that Sasha (with GPT-4o) achieves only 72.5% accuracy on explicit commands and 76.9% on loosely commands. When context memory is added in vanilla baseline, accuracy increases by 22.5% and 15.4% on the two command datasets respectively, compared to Sasha. More notably, MemAura with GPT-4o achieves 100% accuracy across all experiments. These results indicate that incorporating contextual memory can substantially improve system accuracy. When we replace the graph-based memory management and retrieval in MemAura with an industry-standard vector-RAG approach, the resultant accuracy is comparable to the vanilla baseline, but still remains 7% and 11% lower than MemAura, respectively. This implies that our graph-based memory design is more effective than vector RAG at retrieving concise contextual memories to improve LLM inference accuracy.

In terms of token consumption, it is important to note that while the vanilla approach improves accuracy, it also significantly increases token consumption, rising from roughly 2k tokens to 8k–9.5k tokens, because the large volume of stored memory expands the context window. Vector RAG reduces token usage compared to the vanilla baseline, but still uses about 1.9k–2.7k more tokens than MemAura. This is because vector-RAG retrieves logs that are less concise and less accurate than graph-based retrieval, which introduces redundant context. The structured organization and selective retrieval process of MemAura limits the amount of context passed to the LLM. As a result, token consumption is maintained at approximately 3k, with only an additional 0.5k tokens compared to Sasha, which does not consider context memory at all.

In terms of execution efficiency, MemAura is also superior to the baselines. Vanilla is the slowest because it includes the most contextual content to process, resulting in a response time of 12.2–14.6 seconds. Both vector RAG and MemAura achieve a similar level of execution efficiency, where the former is about 2 seconds slower due to the larger number of vectors it searches to find similar entries. Fig. 11 shows a detailed breakdown of MemAura inference latency. Since Periodic Pattern Predictor does not depend on the decomposed intent units, both can run in parallel to reduce end-to-end latency. Specifically, Intent Decomposer takes 1.129 s (32.71%) and Periodic Pattern Predictor takes around 0.76s. MemAura then queries preference information through the Memory Graph Manager, invokes LLMs to process each individual intent unit in parallel and finally aggregates their responses into the final result, which takes 2.132s. Overall, with system-level optimizations and acceleration, MemAura achieves an end-to-end latency of 4.5–4.9s.

We acknowledge that MemAura does not yet provide near-instant feedback (e.g., within 1 second). However, the primary bottleneck is LLM inference latency. This also partially explains why Sasha incurs an 8–9s delay, as it requires multiple GPT API calls. Accelerating the LLM itself is important but beyond the scope of our work. We argue that MemAura remains valuable because it remains minimal additional inference overhead while substantially improving accuracy, up to a 28% gain over the baselines, making a few seconds of waiting worthwhile for better services.

### 3.6 Performance with lightweight LLM backbone (RQ2/RQ3/RQ4)

In this section, we provide a detailed evaluation of how MemAura, vanilla and vector-RAG perform under different settings after incorporating contextual memory, examining each metric on two scenarios comprehensively. The results are shown in Table 2 and Table 3.

**Observation 1: MemAura maintains high accuracy when using lightweight backbone models. (RQ2)** As shown in Table 2 and Table 3, we evaluate vanilla, vector-RAG and MemAura under three different backbone models. Among them, phi-2 (3B) and Qwen3-4B are lightweight models that can feasibly be deployed on a home device or local server. We observe that when GPT-4o is used as the backbone, the vanilla approach achieves reasonably good accuracy. However, when the backbone is replaced with Qwen3-4B or phi-2, the accuracy of vanilla drops sharply—by approximately 43% and 65%, respectively. This indicates that without the large context window and strong reasoning ability of a large model, simply supplying raw historical logs is insufficient to guide effective inference. The vector-RAG achieves 20–33% higher accuracy than vanilla on phi-2 and Qwen3-4B, but still underperforms MemAura. MemAura maintains strong performance across models. With Qwen3-4B as the backbone, accuracy remains around 94–95%, and even with phi-2, the accuracy only decreases by about 12–23% compared to GPT-4o. With a lightweight LLM backbone, the performance gap between MemAura and the baselines is larger for loose commands. For instance, vector RAG is about 23–26% less accurate than MemAura. A likely reason is that the context retrieved by vector RAG, while more concise than feeding all logs, is still less accurate and less concise than the context produced by MemAura’s graph-based memory management. Due to their limited reasoning capacity compared to large models (e.g., GPT-4o), lightweight models such as Qwen3-4B and phi-2 require more concise and targeted context like that provided by MemAura to maintain reasoning accuracy. This demonstrates that the effectiveness of MemAura does not primarily depend on the inherent capabilities of the backbone LLM. It results from the structured memory representation and retrieval strategy provided by MemAura.

**Observation 2: MemAura rarely incurs device selection errors, in contrast to vanilla and vector-RAG. (RQ3)** We categorize errors into Device Errors and Function Errors. As shown in Table 2 and Table 3, the vanilla baseline exhibits these two types of errors at roughly similar frequencies. The vector-RAG method shows more device errors on the loosely command for both the smart-home and smart-office scenarios. However, MemAura almost never produces Device Errors—that is, MemAura’s responses do not incorrectly select or omit required devices. All remaining failures are attributable only to inaccuracies in function selection. This outcome

Table 2. Summary of MemAura and baseline results with lightweight LLMs on the smart-home command datasets. We evaluate several metrics, including Accuracy (Acc), Device Error (DE), Function Error (FE), Latency, Token Consumption, and Personalization Rate (PR).

Configuration			Overall			Latency			Token Consumption			Personalization
Method	Model	Command	Acc	DE	FE	Min	Max	Avg	Min	Max	Avg	PR
vanilla	phi-2	Explicit	0.28	0.48	0.25	13.24	17.88	14.60	2008	2048	2021	0.10
		Loose	0.31	0.54	0.15	14.29	20.15	16.95	2019	2048	2051	0.15
	Qwen3-4B	Explicit	0.50	0.30	0.20	12.98	19.11	16.14	7980	9761	8799	0.18
		Loose	0.56	0.23	0.23	11.27	18.87	14.85	8024	10902	9570	0.16
	GPT-4o	Explicit	0.93	0.00	0.07	10.27	14.38	12.15	7425	7982	8142	0.10
		Loose	0.89	0.04	0.07	12.60	18.62	15.05	7902	9867	9363	0.19
vector-RAG	phi-2	Explicit	0.78	0.15	0.08	4.86	6.37	5.26	1740	2048	1806	0.25
		Loose	0.54	0.36	0.11	5.05	6.38	5.64	1939	2048	2002	0.15
	Qwen3-4B	Explicit	0.83	0.10	0.08	4.99	6.60	5.28	2797	3639	3244	0.20
		Loose	0.68	0.00	0.32	5.04	6.15	5.33	3152	5270	4165	0.18
	GPT-4o	Explicit	0.93	0.00	0.07	4.83	6.72	5.06	3931	5546	4328	0.30
		Loose	0.89	0.04	0.07	5.16	6.20	5.70	4126	6880	5285	0.25
MemAura	phi-2	Explicit	0.88	0.03	0.10	2.92	4.25	3.02	1678	2048	1890	0.38
		Loose	0.77	0.00	0.23	3.07	4.96	3.48	1897	2048	1992	0.31
	Qwen3-4B	Explicit	0.95	0.00	0.05	2.90	4.88	3.57	1024	1984	1453	0.25
		Loose	0.94	0.00	0.07	2.85	4.33	3.03	1628	3637	2234	0.35
	GPT-4o	Explicit	1.00	0.00	0.00	2.79	4.46	2.83	2004	3990	2004	0.38
		Loose	1.00	0.00	0.00	2.68	4.87	3.08	2595	4390	2987	0.43

benefits from MemAura’s structured memory management: the graph-based organization explicitly encodes the relationships between devices and their functions, and provides LLMs with an efficient method for targeted retrieval. While LLMs are capable of reading JSON-formatted logs, when the number of devices and functions scales up, the raw JSON data become cumbersome, making it difficult for an LLM to reliably extract and map the correct execution units. In contrast, MemAura’s structured graph representation enables precise and efficient device reasoning.

**Observation 3: MemAura effectively reduces token consumption while maintaining practical latency.**

Complex tasks and lengthy inputs typically lead to higher response time and token usage in LLM-based systems, especially when the backbone model is lightweight and has limited capacity. If a single instruction required tens of seconds and tens of thousands of tokens, the system would clearly be impractical in real deployments. We measure the latency and token consumption of MemAura, vector-RAG method and vanilla across different tasks and model settings. Phi-2 has a relatively small context window (about 2048 tokens), whereas Qwen3-4B supports a much larger context window (over 30K tokens). Accordingly, as shown in the results, vanilla with phi-2 nearly saturates the entire context window, and with Qwen3-4B it still requires ~8799–9570 tokens. It is reasonable to infer that the vanilla input exceeds phi-2’s context capacity and is therefore truncated, which directly explains the substantial drop in accuracy. Even with Qwen3-4B, the token consumption remains high and is impractical for real-time use. In contrast, MemAura performs notably better. Although MemAura occasionally approaches the

Table 3. Summary of MemAura and baseline results with lightweight LLMs on the smart-office command datasets. We evaluate several metrics, including Accuracy (Acc), Device Error (DE), Function Error (FE), Latency, Token Consumption, and Personalization Rate (PR).

Configuration			Overall			Latency			Token Consumption			Personalization
Method	Model	Command	Acc	DE	FE	Min	Max	Avg	Min	Max	Avg	PR
vanilla	phi-2	Explicit	0.30	0.35	0.35	12.37	16.08	14.50	2029	2048	2040	0.13
		Loose	0.29	0.58	0.13	16.33	20.90	18.39	1995	2048	2015	0.21
	Qwen3-4B	Explicit	0.50	0.30	0.20	14.20	18.92	16.14	7027	9782	8263	0.18
		Loose	0.54	0.21	0.25	13.60	15.16	14.58	7936	10926	9610	0.21
	GPT-4o	Explicit	0.93	0.00	0.07	12.04	15.33	13.66	6928	9882	8010	0.18
		Loose	0.86	0.11	0.04	12.98	16.42	14.82	8667	9926	9255	0.21
vector-RAG	phi-2	Explicit	0.78	0.18	0.05	4.79	6.32	5.14	1722	2048	1927	0.23
		Loose	0.57	0.21	0.21	4.90	6.48	5.46	1927	2048	2010	0.15
	Qwen3-4B	Explicit	0.78	0.18	0.05	5.17	6.53	5.02	2635	3895	3026	0.20
		Loose	0.71	0.00	0.29	5.12	6.15	5.30	3386	5113	4106	0.21
	GPT-4o	Explicit	0.93	0.00	0.07	4.85	6.68	5.30	3668	5723	4244	0.30
		Loose	0.89	0.07	0.04	5.14	6.80	5.74	4126	6652	5360	0.25
MemAura	phi-2	Explicit	0.93	0.05	0.03	3.09	5.03	3.99	1699	2048	1834	0.35
		Loose	0.82	0.00	0.18	3.30	5.74	4.38	1842	2048	1902	0.32
	Qwen3-4B	Explicit	0.95	0.00	0.05	3.29	5.21	4.22	1007	1877	1531	0.25
		Loose	0.93	0.00	0.07	3.30	5.11	3.31	1639	3620	2028	0.39
	GPT-4o	Explicit	1.00	0.00	0.00	3.09	4.82	3.07	1380	3842	2100	0.38
		Loose	1.00	0.00	0.00	3.10	5.35	4.00	2063	4026	2638	0.43

context limit when using phi-2, its accuracy experiences only a modest decrease, indicating that minimal essential information is lost. When using Qwen3-4B, MemAura keeps token consumption within a reasonable 1453–2234 token range. Vector-RAG uses significantly fewer tokens than vanilla, but still more than MemAura with Qwen3-4B and GPT-4o. On the smart-home dataset, it uses about 1.8k and 2.3k more tokens than MemAura, respectively; on the smart-office dataset, it uses about 1.5k–2.1k and 2.1k–2.7k more tokens, respectively.

**Observation 4: MemAura produces more personalized and creative responses. (RQ4)** Personalization Rate reflects the extent to which the system’s responses align with user preferences. As shown in the last column of Table 2 and Table 3, incorporating context memory allows the vanilla baseline to achieve around 10–19% personalization, whereas MemAura achieves 25–43%, representing an improvement of 15–23% compared with vanilla, and shows stable performance across the command datasets in both scenarios. Vector-RAG achieves a personalization rate that is about 5–16% lower than MemAura, with a larger drop on loosely commands. This is a clear and meaningful gain, indicating that MemAura’s structured memory representation and effective pattern extraction mechanism provide the LLM with richer and more accurate personalized signals.

It is necessary to provide a clarification: even though MemAura achieves its best average personalization rate of around 43% when using GPT-4o, this is because a considerable portion of the command dataset does not allow for personalization, particularly the explicit commands. Further command details can be found in the Appendix A.

We present an example from the experimental results, including the outputs of each module and the final response generated by MemAura:

Command: "I'm about to start an important online meeting."

**Intent Decomposer:**

[1] turn on lights for meeting [2] enable quiet mode in the room

**LLM Selected Devices:**

[1] bedroom light [2] table lamp [3] bedroom curtain

**Periodic Pattern Predictor:**

User is likely to operate devices: [1] bedroom curtain [2] table lamp [3] air purifier

User may do: [1] speaking [2] typing [3] flipping a book

In the past relevant time, user usually wants: [1] provide a quiet environment [2] ensure safety after going out [3] make a cup of cappuccino

**Memory Graph Manager:**

Device and Activity Transition Preference:

bedroom light → bedroom curtain (66%)/ table lamp (23%), table lamp → typing (37%)/bedroom curtain (30%)/air purifier (21%), bedroom curtain → air purifier (78%)/ air conditioner (12%).

Relevant devices and corresponding functions:

[1] bedroom light <set\_brightness: 220 (68%);255 (24%)> [2] table lamp <set\_brightness: 200 (66%),255 (10%)> [3] air purifier <set\_mode: eco (90%);sleep (5%), set\_wind\_speed: low (100%)> [4] air conditioner <set\_temperature: 26°C (67%);24°C (21%), set\_mode: auto (92%);eco (7%), set\_timer:8h (18%); 6.5h (14%); 2h (10%)>

**High-level Profile:**

[1]... [7]close bedroom curtains when user's working. [8]...

**MemAura:** Understood. According to your preferences, I will turn on the bedroom light and set its **brightness to 220**. Then, I will turn on the table lamp and set its **brightness to 200**. After that, I will **close the bedroom curtains**, turn on the air purifier, and set its **mode to eco with low wind speed**. Noticing that the indoor temperature is currently high and the bedroom is enclosed, I will also turn on the air conditioner to **auto mode** and set the **temperature to 26°C**.

When processing the command shown above, the Intent Decomposer first breaks it down into two executable intent units. In the meanwhile, the Periodic Pattern Predictor provides the predicted user pattern at the current timestamp. Next, the LLM processes each intent unit in parallel. After identifying the devices relevant to the command, the LLM queries their corresponding preference information through the interface provided by the Memory Graph Manager. The High-level Profile is also fed into the LLM as guiding input. Finally, the LLM performs reasoning based on this information and delivers personalized services. In the example, the parts highlighted in green indicate information that may contribute to the LLM's personalized reasoning, which is also reflected in its response. This example demonstrates that MemAura not only fulfills the functional requirements of the task but also incorporates user-specific preferences into the final action plan, resulting in responses that are more personalized and more aligned with the user's long-term habits.

### 3.7 Ablation Study (RQ5)

In this section, we present ablation studies on each module of MemAura and provide detailed analyses based on representative examples. All experiments in this part are conducted on the smart-home command dataset, using Qwen3-4B as the backbone.

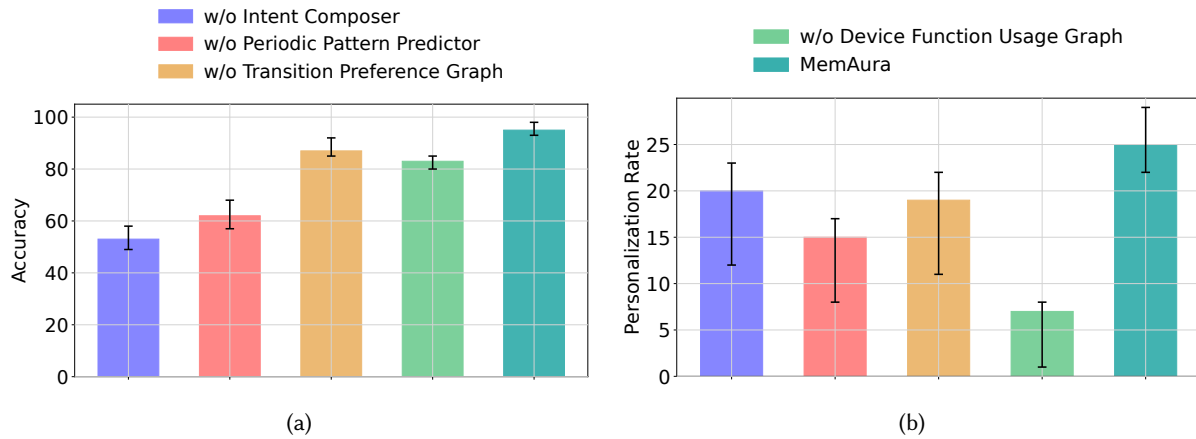


Fig. 12. Overview of the ablation study results. (a) shows the impact on the accuracy of MemAura when each component is removed individually, (b) shows the impact on the personalization rate of MemAura when each component is removed individually.

**3.7.1 Impact of the Intent Decomposer on Accuracy.** Complex tasks that are not pre-processed can substantially hinder reasoning performance. As shown in Fig. 12, when the intent decomposition module is removed, the accuracy of MemAura drops by nearly 40%, while the personalization rate decreases only slightly (around 5%). This indicates that, when encountering complex or ambiguous instructions, failing to decompose them into executable intent units significantly impairs performance, especially for models with limited reasoning capabilities.

**3.7.2 Impact of the Pattern Prediction Predictor on Performance.** We examine whether the model can still complete the task if it is required to infer user patterns directly from memory, without the pattern extraction process. As shown in Fig. 12, removing the Pattern Prediction Predictor leads to a 33% decrease in task success rate, and importantly, a 10% drop in personalization rate. This indicates that without providing summarized behavioral patterns, lightweight models such as Qwen-3-4B struggle to extract personalized information from memory and also have difficulty identifying the correct execution logic. As a result, both personalization and accuracy are significantly reduced.

**3.7.3 Impact of the Device and Activity Transition Preference Graph and the Device Function Usage Graph.** To examine the effect of removing user habit information, we independently ablate the two subgraphs and evaluate performance. As shown in Fig. 12, when the Device and Activity Transition Preference Graph is removed, the accuracy of MemAura decreases to 87%, a reduction of about 8%, and the personalization rate drops to 19%, a decrease of 6%. This suggests that the transition preference graph contributes more to personalization, likely because it provides behavioral pattern information that helps the model infer what the user is likely to do next. When the Device Function Usage Graph is removed, the impact on personalization is even more pronounced: the personalization rate drops to 7%, a reduction of 18%. This indicates that without knowledge of which functions and parameter settings a user typically prefers, the system struggles to generate responses that align with user habits. The effect on accuracy is smaller but still noticeable, with a decrease of about 12%. Overall, both the Device and Activity Transition Preference Graph and the Device Function Usage Graph play critical roles in the performance of MemAura, and their influence is especially significant in improving the personalization rate.

## 3.8 System Stress Evaluation (RQ6)

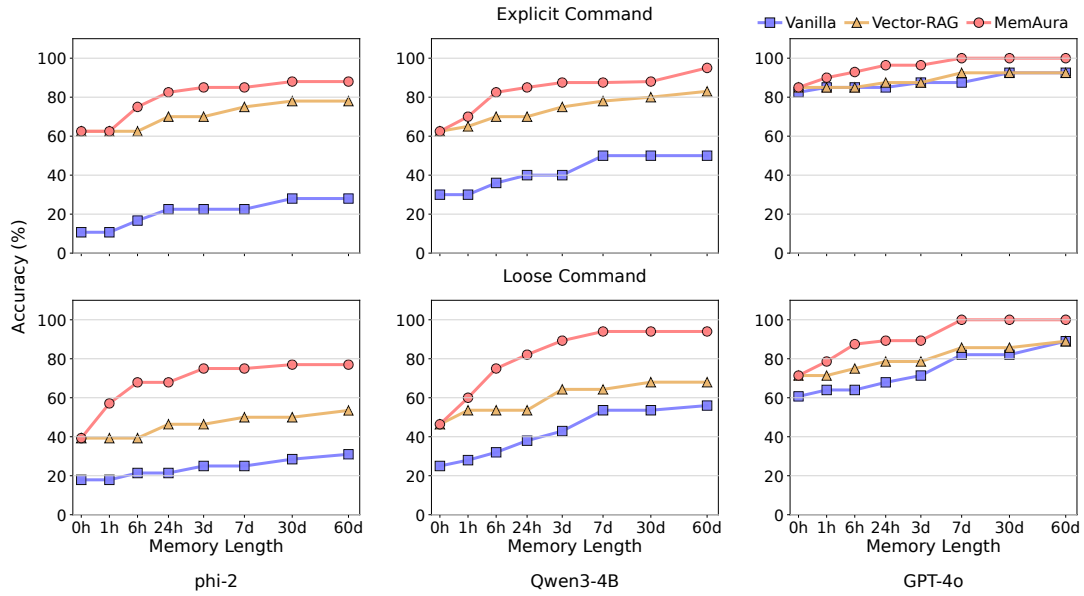


Fig. 13. Stress Test 1 (Cold-start): Impact of different memory lengths on accuracy.

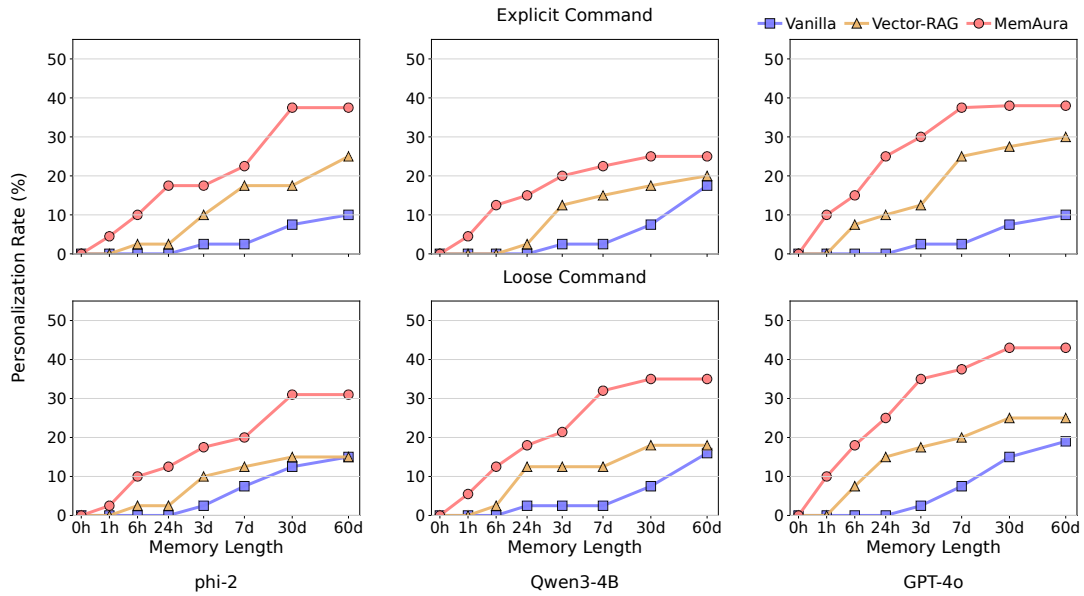


Fig. 14. Stress Test 1 (Cold-start): Impact of different memory lengths on personalization rate.

In this section, we design a series of stress tests to emulate real-world constraints and evaluate MemAura’s robustness against them.

**3.8.1 Cold-start Scenario: Impact of Memory Length on Performance.** We first emulate a cold-start scenario. MemAura organizes memory and supports accurate and efficient retrieval to enable personalized services, but at the beginning, when the log history is still small, we wonder whether MemAura still deliver meaningful gains. To study the effect of memory length, we apply a sliding window that retains only logs within  $L \in \{0h, 1h, 6h, 24h, 3d, 7d, 30d, 60d\}$  prior to the current time, where h denotes hours and d denotes days. The high-level profile and the periodic pattern predictor are also constructed using only the logs within the same window. When  $L = 0$ , we treat it as cold start and provide no additional information beyond the command, and  $L = 60d$  uses the full memory.

In terms of accuracy, as shown in Fig. 13, under cold start, vector-RAG and MemAura are equivalent. With phi-2 and Qwen3-4B, their accuracy is about 62% on explicit commands and about 40% on loose commands. With GPT-4o, their cold-start accuracy reaches 82.5% and 71.4%, respectively. Vanilla has much lower cold-start accuracy because it follows the Sasha-based pipeline with only device selection, device JSON input, and command parsing, leading to a larger gap from MemAura and our vector RAG implementation. While accuracy generally decreases as  $L$  becomes shorter, MemAura consistently outperforms vector-RAG at all lengths. With about one week logs ( $L=7d$ ), MemAura achieves performance comparable to using 60 days logs, which is affordable in practical deployments.

In terms of personalization rate, limited memory has a more severe impact than it does on accuracy, as shown in Fig. 14, which is not surprising. For vanilla, personalization is almost unattainable when  $L < 24h$ . When  $L > 7d$ , increasing  $L$  significantly improves vanilla's personalization rate, and extending the window from  $30d$  to  $60d$  further brings a clear gain. Vector-RAG method shows a clear turning point between  $L = 6h$  and  $L = 24h$ , after which personalization rises markedly. With phi-2 or Qwen3-4B, its personalization is only about 0–2.5% at  $L = 6h$ , but increases rapidly once  $L > 6h$ ; moving from 6h to 24h yields a 7.5–10% improvement. With GPT-4o, the turning point appears earlier, starting around  $L = 1h$ . MemAura shows a turning point at  $L = 1h$  and its personalization rate increases steadily with longer memory. At  $L = 24h$ , it reaches about half of its final personalization, and around  $L = 30d$  it is already close to the  $L = 60d$  level. Overall, MemAura demonstrates better memory utilization than the baselines: even 1 hour of memory enables consistent personalization gains across backbone models.

**3.8.2 Impact of Device Outages/Missing Logs.** Our second stress test emulates device outages. Smart environments include diverse devices with different aging rates, and in practice some devices may go offline, resulting in missing logs. To emulate device outages and missing logs, we inject failures as follows: each device in our evaluation dataset (22 devices in total) has a probability  $p_{out}$  to experience an outage event. If an outage occurs, we remove all logs produced by that device within a contiguous outage block of duration  $\delta \in \{8h, 24h, 7d, 15d, 30d\}$ , where the block is placed uniformly at random within the memory window. Here,  $p_{out}$  controls outage frequency and  $\delta$  is the outage duration. For each  $p_{out}$ , we repeat the random outage injection using five different seeds and report the average.

Fig. 15 reports MemAura's performance under device outages on the smart-home command dataset with Qwen3-4B. In terms of accuracy, for explicit commands, outages with  $\delta \leq 24h$  have little effect on accuracy: it drops only when  $p_{out} > 10\%$ , and stays around 90% even at  $p_{out} = 30\%$ . With  $\delta = 7d$ , accuracy remains above 85%. For the extreme cases  $\delta = 15d$  and  $\delta = 30d$ , accuracy is above 85% when  $p_{out} \leq 10\%$ , and remains 78.5% at  $p_{out} = 30\%$ . The impact is slightly larger for loose commands: with  $\delta = 8h$ , accuracy stays above 89%, and even with 30% of devices having  $\delta = 30d$  outages, it remains above 71%. In terms of personalization rate, for explicit commands, when  $\delta \leq 7d$ , personalization declines only when  $p_{out} > 10\%$ ; at  $p_{out} = 30\%$ , it remains about 17.5%, 12.5%, and 10% for  $\delta = 8h, 24h, 7d$ , respectively. When  $\delta \geq 15d$ , outages have a larger effect, but personalization still retains about half of the original level when  $p_{out} \leq 15\%$ . For loose commands, with  $\delta = 8h$ , personalization is affected only when  $p_{out} > 15\%$  and remains at least 67% of the original; when  $\delta \geq 24h$ , it starts to decline once

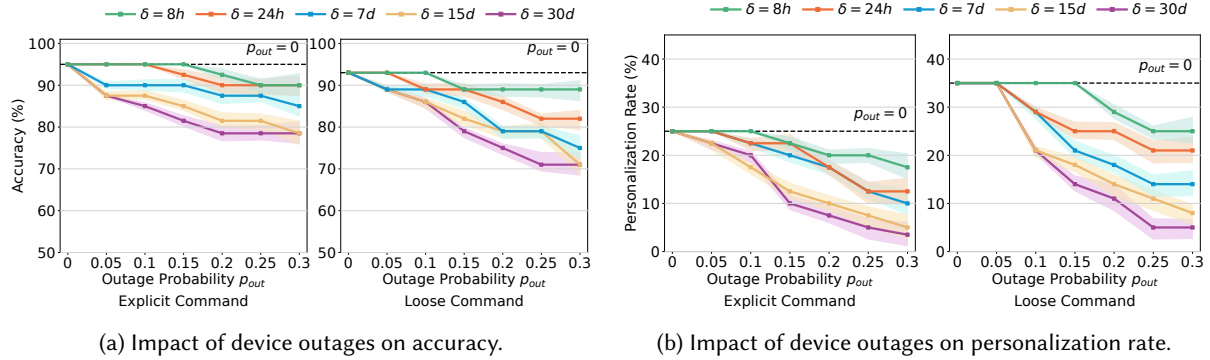


Fig. 15. Stress Test 2 (Device Outages+Missing Logs):  $p_{out}$  denotes the probability that a device experiences an outage, and  $\delta$  denotes the outage duration, during which no logs are available for that device.

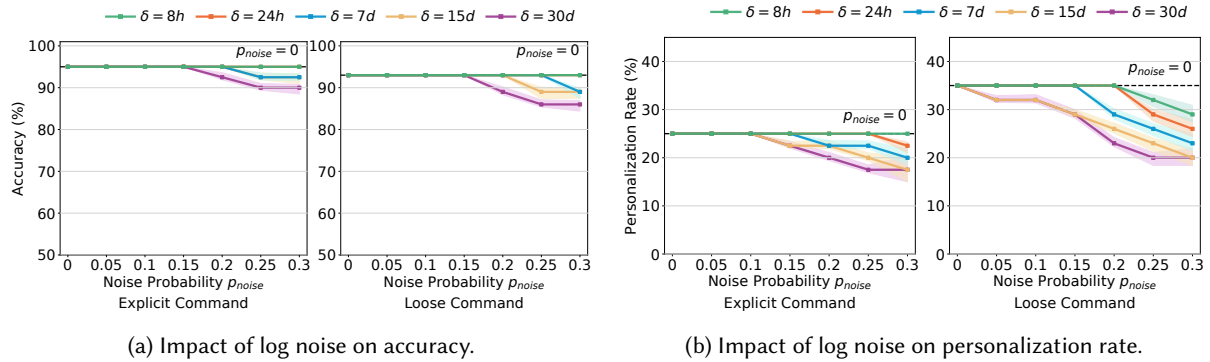


Fig. 16. Stress Test 3 (Logging Noise):  $p_{noise}$  denotes the noise injection ratio, and  $\delta$  is the time window where noise logs are injected.

$p_{out} > 5\%$ . Overall, when  $p_{out} < 10\text{--}15\%$  and  $\delta \leq 24h$ , MemAura’s accuracy and personalization rate are almost unchanged. This indicates that MemAura maintains a reasonable level of robustness to device outages and the resulting missing logs.

**3.8.3 Impact of Logging Noise.** Our third stress test emulates logging noise. In practice, logging is not always accurate and may contain erroneous events. To emulate this scenario, we inject erroneous log events as follows: each existing log is, with probability  $p_{noise}$ , followed by an additional spurious entry. When it is sampled, we draw an event according to the empirical event distribution of our dataset and insert it between the current log and its original successor.

Fig. 16 reports MemAura’s performance under logging noise on the smart-home command dataset with Qwen3-4B. In terms of accuracy, injected logs has little effect for both explicit and loose commands. When  $\delta \leq 7d$ , accuracy is almost unchanged. Only in extreme cases ( $p_{noise} > 15\text{--}20\%$  and  $\delta \geq 15d$ ) does accuracy drop to 86–92.5%. In terms of personalization rate, noise has a larger impact on loose commands than on explicit commands. For explicit commands, personalization shows no clear drop when  $\delta < 7d$ , but decreases by about 5–7.5% when  $\delta \geq 15d$  and  $p_{noise} > 15\text{--}20\%$ . For loose commands, personalization starts to decline when  $p_{noise} \geq 20\%$  for  $\delta = 8h$  or  $24h$  (about 6–9% at  $p_{noise} = 30\%$ ). When  $\delta \geq 15d$ , it keeps declining to around 20% at  $p_{noise} = 30\%$ , though this

is rare. Overall, under reasonable settings ( $\delta \leq 7d$  and  $p_{\text{noise}} \leq 15\%$ ), noisy logs have little impact, indicating that MemAura is reasonably robust.

## 4 USER STUDY

In addition to quantitative performance evaluations, we conducted a user study to assess the usability of MemAura. We invited  $N = 8$  participants to rank the performance of MemAura and the baselines. Subsequently, participants were allowed to freely specify scenarios and time contexts to issue commands to MemAura and evaluate its responses. This user study aims to investigate this research question:

**RQ7:** Does MemAura provide a better user experience compared to the baselines?

### 4.1 Study Procedure

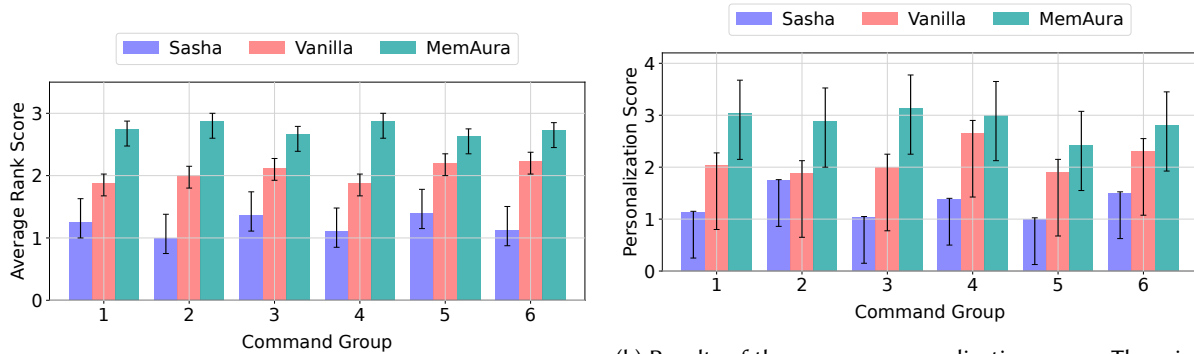
**4.1.1 Participants.** We recruited  $N=8$  participants (ages  $M = 23.6$  years,  $SD = 3.4$  years). Their demographic information is shown in Table 4.

Table 4. Demographic characteristics of the participants in user study.

ID	Age	Education	Smart Environment Familiarity	Familiar Smart Assistant
P1	18–24	Graduate	Moderately	Siri, ChatGPT Voice
P2	25–34	Bachelor's	Slightly	Siri
P3	35–44	Graduate	Slightly	Google Assistant
P4	25–34	Bachelor's	Very	Siri, ChatGPT Voice
P5	45–54	High school or equivalent	Moderately	Siri, Xiaoai Assistant
P6	55–64	Bachelor's	Slightly	Siri
P7	18–24	Bachelor's	Very	Siri, Gemini Home
P8	25–34	Graduate	Slightly	Xiaoai Assistant

**4.1.2 Study Design.** Each participant was provided with a computer displaying the room layout and device distribution, while running both MemAura and the baseline systems to present response results. The specific workflow was as follows:

- (1) **Demographic Questionnaire:** Collect participants' demographic information, with particular attention to their education background and familiarity with smart environment services.
- (2) **Background Learning:** Each participant was given a 15-minute introduction to the overview of MemAura, including a detailed explanation of the Context Memory Dataset, such as each user's traits and preferences. Several examples of command executions by MemAura were also demonstrated afterward.
- (3) **Anonymous Ranking and Personalization Scoring Task:** In this task, after being presented with a predefined user and command, participants were shown responses generated by MemAura and the baselines—without being informed of their sources and with the order randomized. Participants were asked to rank the responses from best to worst based on overall performance, then score each response for its degree of personalization, and provide brief qualitative feedback.
- (4) **Open-ended Q&A task:** In this stage, participants could select any resident in the environment and freely issue commands to MemAura. Afterwards, they evaluated MemAura's performance based on its responses.
- (5) **Post-Task Questionnaire**
- (6) **Semi-Structured Interview**



(a) Scores from the anonymous ranking task across six command groups. Responses were scored from 3 to 1, corresponding to rankings from best to worst, with ties permitted.

(b) Results of the average personalization scores. The minimum score is 0, indicating no personalization, while the maximum score is 4, representing highly creative and personalized responses.

Fig. 17. Results of Anonymous Ranking and Personalization Scoring for MemAura

Between each task, participants were allowed to take a break. After the study, participants were compensated at a rate of \$12 per hour. This study has been approved by the Institutional Review Board (IRB) of our institution.

## 4.2 Anonymous Ranking and Personalization Scoring Task

**4.2.1 Anonymous Ranking.** We first asked participants to complete the anonymous ranking task. In this task, commands were grouped according to six different residents, with each group containing ten commands randomly sampled from the Explicit Command Dataset. The responses from MemAura and the two baselines were presented to participants anonymously and in random order. Participants ranked the responses from best to worst, corresponding to scores of 3 to 1, respectively.

As shown in Fig. 17 (a), MemAura consistently outperformed both Sasha and vanilla in the ranking task, achieving average scores between 2.63 and 2.88, which is approximately 0.7 points higher than vanilla. These results indicate that, from a subjective user perspective, MemAura delivers noticeably better overall performance than the two baselines.

**4.2.2 Personalization Scoring.** Next, participants rated each response for personalization. The lowest score, 0, indicated that the response contained no personalized content, while the highest score, 4, indicated that the response provided sufficiently rich and appropriate personalization aligned with the resident’s preferences. As shown in Fig. 17 (b), MemAura achieved an average personalization score of approximately 2.63, significantly higher than the baselines. Although there remains a gap from the maximum score, this is reasonable—some commands inherently lack opportunities for personalization, or the user’s historical data may not provide enough evidence to support personalized responses. Nevertheless, the comparison clearly shows that Sasha and vanilla scored only around 1.26 and 2.0, respectively, demonstrating MemAura’s distinct advantage in delivering personalized services.

## 4.3 MemAura’s User Experience Study

**4.3.1 SUS Questionnaire Result.** To collect user ratings on their experience with MemAura, We employed System Usability Scale (SUS) [4] questionnaire, which is widely used to evaluate system usability (Appendix C). The standard SUS questionnaire consists of 10 items rated on a 5-point Likert scale. We computed the raw scores for each participant following the scoring method described in [20], i.e., converting the range from 1–5 to 0–4. The results are shown in Fig. 18 (a). We also subsequently calculated both the overall SU score and the usability

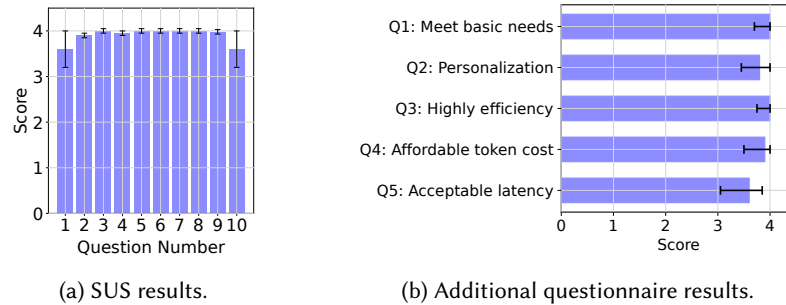


Fig. 18. Results of Standard SUS and Our Self-Designed Questionnaires for MemAura

subscore for each group, with scores ranging from 0 to 100. The non-expert group achieved an SU score of 98.00 ( $\pm 3.00$ ), with a usability subscore of 98.75 ( $\pm 1.875$ ). This result indicates that MemAura is highly practical and provides a positive user experience.

**4.3.2 Self-design Questionnaire Result.** In addition to the standard SUS questionnaire, we were also interested in issues specific to the user experience of MemAura. Thus, we designed an additional questionnaire with the following questions:

- MemAura fulfills the fundamental requirements of user commands.
- MemAura provides high-quality personalized services.
- MemAura operates with high efficiency.
- The token consumption of MemAura is within an acceptable range.
- The latency of MemAura is acceptable for practical use.

Except for the last question, all of the above questions were answered using a 5-point scale to indicate participants' level of agreement with the statements, where the lowest score means "strongly disagree" and the highest score means "strongly agree". The 5-point scale was adopted to maintain consistency with the SUS questionnaire. We also converted the range to 0–4.

The results of our self-designed questionnaire are shown in Fig. 18 (b). Most participants agreed that MemAura effectively fulfills the fundamental requirements of user commands and operates with high efficiency, with token consumption remaining within a reasonable range. In addition, participants positively evaluated the quality of personalization and response latency provided by MemAura, indicating that the system demonstrates strong overall performance and practical feasibility.

**4.3.3 Open-ended Q&A Task.** In addition to the quantitative evaluations, we invited participants to engage in an open-ended Q&A session. Participants could select any resident and, after becoming familiar with their profile and habits, freely issue commands to MemAura. Afterwards, they were encouraged to evaluate MemAura's responses and provide feedback.

**Personalization quality reflected in the details of functional parameters.** Although individuals may define personalization differently, true personalization is often reflected in the precise handling of preference details. When faced with participants' vague commands, MemAura can still infer user preferences and provide detailed, contextually appropriate responses:

**Resident Zoe: a home fitness enthusiast**

**P4:** Time for Pamela’s exercise! Let me get the right space ready.

**MemAura:** No problem! Based on your historical preferences, you usually start exercising in the center of the living room where the air conditioner does not blow directly. I will set the air conditioner to 26°C with the lowest fan speed, turn on the living room light at 240 brightness, and start the audio player with your jazz playlist. I will also close the living room curtains to keep privacy.

**P4 Feedback:** This answer really gets Zoe. I’ve noticed she often works out while listening to jazz, and for a girl, closing the curtains during a workout is such a thoughtful detail. Sometimes personalization isn’t about big gestures, it’s these small touches that make things feel warm.

**Resident James: a coffee lover**

**P6:** Can I get a cup of coffee?

**MemAura:** Of course. Based on your coffee preferences, you usually enjoy a double-shot latte on weekday mornings. I will start the coffee maker, set the mode to latte, adjust the grind level to 13, and select the double-cup output. Since the weather is pleasant and you prefer a well-ventilated environment, I will also open the kitchen curtains to let in fresh air, and gradually dim the kitchen lights until they turn off.

**P6 Feedback:** For someone like me who’s into coffee, this is spot on. It nails all the little preference details. We’re getting older; even my wife forgets how to adjust the settings sometimes, but system remembers.

In the two examples above, the participants’ commands contained no explicit details, yet MemAura successfully inferred and applied them, which explains the positive user feedback. Therefore, the ability to accurately extract and apply residents’ preferred functional parameters is a key factor in achieving personalization. Meanwhile, for elderly users, this capability can also help them retain everyday setting details, thereby reducing inconvenience and improving usability.

**Take recent history into consideration.** Participants designed several commands that could be influenced by recent events to observe how MemAura would respond:

**Resident Harry: a pet owner**

**Key event two hours ago:** Harry gave his dog a bath.

**P1:** The house is a mess, we need a proper clean.

**MemAura:** Understood. Based on your recent history, you just bathed and dried your dog, so it’s important to clean the bathroom and hallway thoroughly. I will start the vacuum robot using your usual settings — mode set to sweep then mop, order: bedroom, living room, then hallway, with suction power at maximum. I will also open the bathroom window and curtains for ventilation. After 30 minutes, the robot will clean the bathroom again to prevent hair entanglement.

**P1 Feedback:** It saved me some hassle by remembering what happened earlier. It knew which spots needed more attention and cleaned the bathroom last, after the water dried — really considerate.

In this scenario, MemAura generated a personalized service because it was aware of the preceding key event, effectively avoiding potential inconveniences. Another participant designed a different command for the same situation:

### Resident Harry: a pet owner

**Key event two hours ago:** Harry gave his dog a bath.

**P2:** I'm going to work.

**MemAura:** You left your dog at home, so I'll minimize all potential risks. I will turn off the water heater, idle smart plugs, and lights in other rooms—keeping only one living room light on at 55 brightness. The camera will be set to pet mode. Also, since the records show you recently bathed your dog, I will adjust the air conditioner to 27°C with the lowest fan speed to prevent it from catching a cold.

**P2 Feedback:** It's very thoughtful, using the history to spot what needs attention.

From these cases, it can be inferred that prior events represent a form of implicit preference, as they directly influence subsequent environmental decisions. Therefore, MemAura, equipped with a memory management system capable of efficiently retrieving historical context, can highlight such events and incorporate them into its responses—forming the foundation of personalization.

**Distinguishing multiple intents and tasks within loosely commands** People often prefer to express a series of requests all at once—after all, no one wants to speak one command at a time and wait for a reply, as that would be inefficient. Participants were also interested in whether MemAura could effectively distinguish multiple intents within complex or loosely structured commands while still providing personalized services:

### Resident Jonathan

**P8:** I'm gonna head inside to play some games. Gotta tidy up the place before my mom comes, and take care of her when she gets here.

**MemAura:** Sure. Based on your usual gaming preferences, I will turn on the bedroom light and set its brightness to 180, close the bedroom curtains, and turn off the living room lights. In addition, I will immediately start the robot vacuum cleaner in quick mode with maximum suction power. According to the settings you typically use when your mother visits, I will set a trigger so that when fingerprint ID 2 unlocks the door, the living room light turns on at brightness 255, and the audio player starts playing a soothing music playlist at volume 60.

**P8 Feedback:** It was quite a long chain of tasks, but the system managed everything smoothly and added personalization to each part.

Since the Intent Decomposer in MemAura first breaks down a command into executable intent units, even a complex instruction is merely composed of multiple intents executed in parallel. This design not only ensures MemAura's capability to handle complex tasks but also allows it to distinguish user preferences for each intent, enabling more precise and personalized services.

## 5 RELATED WORK

### 5.1 Memory Management for LLMs

Large Language Models (LLMs) have recently demonstrated exceptional capabilities in language understanding, generation, and reasoning, especially as their context windows continue to expand, allowing them to handle increasingly complex reasoning tasks. To enhance an LLM's knowledge and guide its behavior, context organization, retrieval, management, and integration have become crucial. Earlier studies explored various context management strategies [11, 35, 47, 59], including retrieval and generation mechanisms, as well as context processing, compression, and optimization methods. These later evolved into systems based on Retrieval-Augmented

Generation (RAG) to effectively leverage external knowledge [10, 14, 25, 45]. For example, FlashRAG [21] introduced a modular toolkit for efficient retrieval, while Self-RAG [3] proposed a self-reflective retrieval strategy. More recently, researchers have begun treating context as a form of memory, inspired by human cognition, enabling long-term and short-term information evolution, updating, and optimization [9, 27, 33, 51]. Examples include Memory Bank [59], which uses long-term memory to enhance response quality and personalization, and Sandbox [17], which manages long conversational histories interactively. However, when applying memory management to real-world environments for perception, interaction, and continuous updating, these approaches face significant limitations. Most existing memory-based systems are primarily designed for chatbot-style conversational scenarios rather than for smart environments like MemAura. For example, THEANINE [33] introduces a timeline-based memory management mechanism for dialogue understanding but supports only natural language text memories, making it unsuitable for handling data common in IoT and environmental settings. Similarly, A-MEM [54] and MemoRAG [41] are tailored for conversational or document-retrieval tasks, but their stored content remains in raw or unstructured formats, requiring the LLM to interpret it at runtime rather than enabling direct, structured, and ready-to-use access as needed in MemAura.

## 5.2 Personalized LLM Services

The concept of personalized service can be traced back to recommender systems (RS), which predict a user's interest in an item based on historical records and related factors, thereby providing tailored products or services. Traditional recommender systems often rely on self-supervised learning or other data-driven techniques [52, 60]. Recently, LLM-driven recommendation systems, or more broadly, LLM-based personalized services, have brought new possibilities to this domain. A straightforward approach is to embed a portion of user information directly into the prompt to help the LLM make behavioral predictions [6, 36, 43, 48]. Some studies employ user profile prompting to improve recommendations [30, 44]. However, as the amount of user information increases, this approach faces scalability challenges such as exceeding the context length, handling incomplete or insufficient data [8, 39], and addressing the cold-start problem. Therefore, LLM personalization is inherently tied to context memory management. For example, Li et al. [26] constructed long-term and short-term conversational memory as well as role memory to build an LLM-driven personalized agent, while SeCom [37] designed a fragment-level memory bank to enhance retrieval efficiency. OS-1 [55] uses eyewear devices to collect data, which are then organized into real-time memory and historical context memory, and summarized into a user profile to help the LLM deliver more personalized behavior in dialogue tasks. Inspired by these approaches, we designed a specialized memory management framework in MemAura to enable LLM-powered personalized services tailored to smart environments.

## 5.3 Smart Environment Personalized Services via LLM

The defining feature of a smart environment lies in its ubiquitous sensing and computation. Networked smart devices, wearable sensors, and embedded systems together form an intelligent ecosystem that continuously interacts with human residents [16, 53, 57]. Recent studies have leveraged the reasoning capability of large language models to interpret and act upon the vast and heterogeneous data generated within such environments [24, 29, 58]. One representative application of smart environments is home healthcare. For example, Kim et al. [22] fed physiological signals collected from wearable devices into an LLM to enhance diagnostic accuracy. DrHouse [56] uses sensor signals from home devices as input to construct a diagnostic decision tree, assisting in in-home medical consultations. This represents another scenario of personalized service, where discrete sensor data must be transformed into clinically useful contextual information, requiring a well-designed memory storage structure and information extraction mechanism. More commonly, smart environments are embodied in smart home assistants. Sasha [23] proposes a new paradigm for a smarter LLM-based smart assistant, exploring how

Table 5. Comparison of recent LLM-driven services in smart environments. The 'Data Content' and 'Data Format' columns describe the type and representation of data input to the LLM, respectively. The 'Memory Duration' column indicates the time span of stored data — where Episodic refers to non-temporal data without an explicit time dimension.

Approach	Data Content	Data Format	Memory Duration
GreenIFTTT [12]	Devices Infomation	Json	Episodic
Sensor2Text [5]	Wearable Data	Parametric Form	1-2 hours
Sasha [23]	Devices Infomation	Json	Episodic
TaskSense [29]	Sensor & Device Data	Json (→ Graph)	6-12 hours
ContextAgent [58]	Sensor & Device Data	Json	Episodic
JARVIS [24]	HVAC Sensor Data	Json (→ RAG)	1 year / 1 minute resolution
<b>MemAura</b>	Sensor & Device Data	Json (→ Graph)	1440 hours

large language models can reason creatively and execute complex tasks. Although it performs well in handling ambiguous commands, Sasha is limited to real-time information and lacks any temporal context memory, which leads to weaker personalized response. In this work, we also include Sasha as one of the baselines for detailed evaluation. Sensor2Text [5] tracks daily activities using wearable sensor data and enables natural-language interaction. GreenIFTTT [12] integrates device and sensor data in an IFTTT-like framework and employs an LLM-based chatbot to help users design automation routines. Commercially, systems such as Siri [2], Alexa [1], and Gemini for Home [13] demonstrate practical implementations of this paradigm. However, as summarized in Table 5, prior research still faces an unresolved challenge: how to distill long-term user behavior patterns and preferences from continuous sensor and device logs, in order to enhance personalized reasoning and service quality in complex tasks. To address this challenge, we propose MemAura, a system designed to efficiently manage context memory and extract behavioral patterns and preferences from raw IoT data, to improve LLM-driven personalization in smart environments.

## 6 DISCUSSION

The proposed MemAura introduces a new approach to LLM-based personalized services with context memory in smart environment. In this section, we discuss MemAura's potential for real-world deployment, its privacy and security considerations, and the limitations of the current work.

### 6.1 Deploying MemAura in Real-World Environments

We designed MemAura with real-world deployment in mind, ensuring that several of its components and design choices are well-suited for home environments. First, the choice of backbone model offers flexibility. As shown in our evaluations, MemAura maintains stable performance and output quality even when built upon lightweight models. Thus, in practical deployment, users can either (1) call advanced LLMs via API (e.g., GPT-4o, Gemini) to achieve streamlined workflows and high accuracy, or (2) adopt lightweight local models for fully on-device operation. The models tested in our experiments, phi-2 and Qwen-3-4B, can both be feasibly deployed on home servers or personal computers, and many other compact models with strong reasoning capabilities can also serve as viable alternatives. Moreover, the Periodic Pattern Predictor, which is based on BERT, can be trained locally on a home server. The predictor trained on 1440 hours of IoT logs is approximately 4.5 GB in size, allowing inference tasks to run smoothly even on personal laptops, making it highly home-friendly. Finally, the data storage requirement is modest: the raw multi-user IoT log covering 1440 hours occupies only a few kilobytes,

imposing minimal storage overhead. Therefore, MemAura demands relatively low computational resources and can be fully deployed locally, offering a feasible and privacy-preserving solution for real-world smart environments.

## 6.2 Privacy and Security Considerations of LLM-Enabled MemAura

Since smart environments inevitably collect large amounts of sensitive data from devices and sensors, privacy and security must be carefully addressed before any real-world deployment. Privacy risks are a critical concern for LLM-driven systems. The most direct mitigation strategy is to process all sensitive information locally, and, when necessary, use cloud-based LLMs only for non-sensitive data. This approach aligns closely with our earlier discussion on local deployment, which naturally eliminates most privacy concerns. In particular, the Intent Decomposer module can safely invoke API-based LLMs because its inputs consist exclusively of global, non-private data. However, if a system chooses to rely entirely on advanced cloud-connected LLMs, it should include mechanisms to periodically remind users of potential risks when collecting sensitive data and to actively monitor for possible data leakage. Overall, MemAura offers deployment flexibility that allows balancing privacy, performance, and convenience, ensuring that sensitive user data remains protected in intelligent environments.

## 6.3 Limitation and Future Work

Our work has several limitations. First, the IoT data used in our system are not multimodal. Many multimedia IoT devices, such as cameras and voice recorders, collect information in video and audio modalities. Although the current single-modality memory in MemAura already enables high-quality personalized services, the incorporation of multimodal information could unlock even more creative and expressive possibilities. Incorporating these new modalities into the memory raises open questions about how to align, store, and coordinate heterogeneous data effectively, which deserves further exploration. In addition, the latency of MemAura still has room for improvement. Although we have employed parallel processing of intent units to reduce response time, achieving near-zero-latency service remains challenging. Future work could explore ways to make the graph-structured memory directly interpretable by the model through pretraining or joint embedding, thereby minimizing the time spent on knowledge distillation and input processing.

## 7 CONCLUSION

In this work, we present MemAura, a memory management system for smart environments that organizes device and sensor memories, distills user behavior patterns and preferences, and continuously updates user profiles through three key modules. Experimental results demonstrate that when powered by a large LLM, MemAura achieves 100% task success and a personalization rate of 38%–43%. Even when using lightweight LLMs, MemAura maintains up to 95% accuracy with an average personalization rate of 37.5%. In contrast, vanilla equipped with the same lightweight LLMs achieves only 28%–56% accuracy and a maximum personalization rate of 17.5%, while vector-RAG achieves 54–83% accuracy and up to 25% personalization rate, highlighting the robustness and consistency of MemAura under constrained model capacity. Furthermore, MemAura shows favorable token usage and latency, making it both efficient and practical for real-world deployment.

## Acknowledgments

We sincerely thank the anonymous AEs and reviewers for their valuable comments and suggestions. This work is supported by Guangzhou-HKUST(GZ) Joint Funding Program (No. SL2024A03J01192), Guangzhou Municipal Science and Technology Project (Young Scholar Set-Sail Research, No. SL2024A04J01392), Guangdong General Higher Education Institution (Young Innovative Talents, No. 2025KQNCX113), the Department of Education of Guangdong Province (Innovation Team, No. 2024KCXTD008), and Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No. 2023B1212010007).

## References

- [1] Amazon. 2025. alexa. <https://www.amazon.com/alexaplus>
- [2] Apple. 2025. apple-siri. <https://www.apple.com/siri/>
- [3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Self-reflective retrieval augmented generation. In *NeurIPS 2023 workshop on instruction tuning and instruction following*.
- [4] Aaron Bangor, Philip T Kortum, and James T Miller. 2008. An empirical evaluation of the system usability scale. *Intl. Journal of Human-Computer Interaction* 24, 6 (2008), 574–594.
- [5] Wenqiang Chen, Jiaxuan Cheng, Leyao Wang, Wei Zhao, and Wojciech Matusik. 2024. Sensor2text: Enabling natural language interactions for daily activity tracking using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 4 (2024), 1–26.
- [6] Konstantina Christakopoulou, Alberto Lalama, Cj Adams, Iris Qu, Yifat Amir, Samer Chucrri, Pierce Vollucci, Fabio Soldo, Dina Beseio, Sarah Scodel, Lucas Dixon, Ed H. Chi, and Minmin Chen. 2023. Large Language Models for User Interest Journeys. arXiv:2305.15498 [cs.CL] <https://arxiv.org/abs/2305.15498>
- [7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 <http://arxiv.org/abs/1810.04805>
- [8] Horatiu Dumitru, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, and Mehdi Mirakhorli. 2011. On-demand feature recommendations derived from mining public product descriptions. *2011 33rd International Conference on Software Engineering (ICSE)* (2011), 181–190. <https://api.semanticscholar.org/CorpusID:2779250>
- [9] Jizhan Fang, Xinle Deng, Haoming Xu, Ziyang Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. 2025. LightMem: Lightweight and Efficient Memory-Augmented Generation. *arXiv preprint arXiv:2510.18866* (2025).
- [10] Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. 2024. Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. *arXiv preprint arXiv:2407.21059* (2024).
- [11] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945* (2023).
- [12] Mathyas Giudici, Luca Padalino, Giovanni Paolino, Ilaria Paratici, Alexandru Ionut Pascu, and Franca Garzotto. 2024. Designing home automation routines using an LLM-based chatbot. *Designs* 8, 3 (2024), 43.
- [13] Google. 2025. Google home-gemini. <https://home.google.com/get-inspired/bring-home-the-magic-of-gemini/>
- [14] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779* (2024).
- [15] Jiawei Hu, Hong Jia, Mahbub Hassan, Lina Yao, Brano Kusy, and Wen Hu. 2025. Lightllm: A Versatile Large Language Model for Predictive Light Sensing. In *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*. 158–171.
- [16] Yongzhi Huang, Kaixin Chen, Yandao Huang, Lu Wang, and Kaishun Wu. 2023. A Portable and Convenient System for Unknown Liquid Identification With Smartphone Vibration. *IEEE Transactions on Mobile Computing* 22 (2023), 1894–1911. <https://api.semanticscholar.org/CorpusID:239284340>
- [17] Ziheng Huang, Sebastian Gutierrez, Hemanth Kamana, and Stephen MacNeil. 2023. Memory sandbox: Transparent and interactive memory management for conversational agents. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–3.
- [18] ICATUS. 2023. Time-use activities classificationRegional Workshop on Time Use Statistics: Methods and Uses.
- [19] IFTTT. 2024. IFTTT:Automation for Business and Home. <https://ifttt.com/>, Accessed:2024-09-06.
- [20] Brian Kenji Iwana and Seiichi Uchida. 2020. An Empirical Survey of Data Augmentation for Time Series Classification with Neural Networks. *CoRR* abs/2007.15951 (2020). arXiv:2007.15951 <https://arxiv.org/abs/2007.15951>
- [21] Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*. 737–740.
- [22] Yu Han Kim, Xuhai Xu, Daniel McDuff, Cynthia Breazeal, and Hae Won Park. 2024. Health-LLM: Large Language Models for Health Prediction via Wearable Sensor Data. In *ACM Conference on Health, Inference, and Learning*. <https://api.semanticscholar.org/CorpusID:266998674>
- [23] Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. 2024. Sasha: Creative Goal-oriented Reasoning in Smart Homes with Large Language Models. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 1 (2024), 1–38.
- [24] Sungmin Lee, Minju Kang, Joonhee Lee, Seungyong Lee, Dongju Kim, Jingi Hong, Jun Shin, Pei Zhang, and JeongGil Ko. 2025. LLM-based Question-Answer Framework for Sensor-driven HVAC System Interaction. *ArXiv* abs/2507.04748 (2025). <https://api.semanticscholar.org/CorpusID:280148125>
- [25] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information*

- processing systems* 33 (2020), 9459–9474.
- [26] Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. 2025. Hello Again! LLM-powered Personalized Agent for Long-term Dialogue. arXiv:2406.05925 [cs.CL] <https://arxiv.org/abs/2406.05925>
  - [27] Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, et al. 2025. MemOS: A Memory OS for AI System. *arXiv preprint arXiv:2507.03724* (2025).
  - [28] Kaiwei Liu, Bufang Yang, Lilin Xu, Yunqi Guo, Guoliang Xing, Xian Shuai, Xiaozhe Ren, Xin Jiang, and Zhenyu Yan. 2025. TaskSense: A Translation-like Approach for Tasking Heterogeneous Sensor Systems with LLMs. In *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*. 213–225.
  - [29] Kaiwei Liu, Bufang Yang, Lilin Xu, Yunqi Guo, Guoliang Xing, Xian Shuai, Xiaozhe Ren, Xin Jiang, and Zhenyu Yan. 2025. TaskSense: A Translation-like Approach for Tasking Heterogeneous Sensor Systems with LLMs. *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems* (2025). <https://api.semanticscholar.org/CorpusID:278326090>
  - [30] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172 [cs.CL] <https://arxiv.org/abs/2307.03172>
  - [31] Karla Jaimes Merazzo, Joseba Totoricaguena-Gorriño, Eduardo Fernández-Martín, F Javier Del Campo, and Eva Baldrich. 2021. Smartphone-enabled personal diagnostics: Current status and future prospects. *Diagnostics* 11, 6 (2021), 1067.
  - [32] Microsoft. 2023. phi-2. <https://huggingface.co/microsoft/phi-2>
  - [33] Kai Tzu-iunn Ong, Namyoun Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seung-won Hwang, Dongha Lee, and Jinyoung Yeo. 2024. Towards lifelong dialogue agents via timeline-based memory management. *arXiv preprint arXiv:2406.10996* (2024).
  - [34] OpenAI. 2025. Hello GPT-4o. <https://openai.com/index/hello-gpt-4o/>, Accessed:2025-04-26.
  - [35] Charles Packer, Vivian Fang, Shishir\_G Patil, Kevin Lin, Sarah Wooders, and Joseph\_E Gonzalez. 2023. MemGPT: Towards LLMs as Operating Systems. (2023).
  - [36] Dario Di Palma, Giovanni Maria Biancofiore, Vito Walter Anelli, Fedelucio Narducci, Tommaso Di Noia, and Eugenio Di Sciascio. 2024. Evaluating ChatGPT as a Recommender System: A Rigorous Approach. arXiv:2309.03613 [cs.IR] <https://arxiv.org/abs/2309.03613>
  - [37] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. 2025. On Memory Construction and Retrieval for Personalized Conversational Agents. arXiv:2502.05589 [cs.CL] <https://arxiv.org/abs/2502.05589>
  - [38] Maithili Patel and Sonia Chernova. 2023. Proactive Robot Assistance via Spatio-Temporal Object Modeling. In *Conference on Robot Learning*. PMLR, 881–891.
  - [39] Luis G. Pérez, Manuel J. Barranco, and Luis Martínez-López. 2007. Building User profiles for Recommender Systems from incomplete preference relations. *2007 IEEE International Fuzzy Systems Conference (2007)*, 1–6. <https://api.semanticscholar.org/CorpusID:567609>
  - [40] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating Household Activities via Programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8494–8502.
  - [41] Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591* 1 (2024).
  - [42] Dmitry Rivkin, Francois Hogan, Amal Feriani, Abhisek Konar, Adam Sigal, Xue Liu, and Gregory Dudek. 2024. AIoT Smart Home via Autonomous LLM Agents. *IEEE Internet of Things Journal* (2024).
  - [43] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large Language Models are Competitive Near Cold-start Recommenders for Language- and Item-based Preferences. arXiv:2307.14225 [cs.IR] <https://arxiv.org/abs/2307.14225>
  - [44] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. arXiv:2302.00093 [cs.CL] <https://arxiv.org/abs/2302.00093>
  - [45] Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441* (2025).
  - [46] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] <https://arxiv.org/abs/2505.09388>
  - [47] Cangqing Wang, Yutian Yang, Ruisi Li, Dan Sun, Ruicong Cai, Yuzhu Zhang, and Chengqian Fu. 2024. Adapting llms for efficient context processing through soft prompt compression. In *Proceedings of the International Conference on Modeling, Natural Language Processing and Machine Learning*. 91–97.
  - [48] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. arXiv:2304.03153 [cs.IR] <https://arxiv.org/abs/2304.03153>
  - [49] Gary M Weiss and Jeffrey W Lockhart. 2012. The impact of personalization on smartphone-based activity recognition. In *AAAI workshop on activity context representation: techniques and languages*. Toronto., 98–104.
  - [50] Xiaomi. 2025. Xiaomi Xiaoi. <https://xiaoi.mi.com/>
  - [51] Junjie Xiong, Changjia Zhu, Shuhang Lin, Chong Zhang, Yongfeng Zhang, Yao Liu, and Lingyao Li. 2025. Invisible Prompts, Visible Threats: Malicious Font Injection in External Resources for Large Language Models. *arXiv preprint arXiv:2505.16957* (2025).
  - [52] Jinfeng Xu, Zheyu Chen, Shuo Yang, Jinze Li, Hwei Wang, and Edith C. H. Ngai. 2024. MENTOR: Multi-level Self-supervised Learning for Multimodal Recommendation. arXiv:2402.19407 [cs.IR] <https://arxiv.org/abs/2402.19407>

- [53] Lilin Xu, Chaojie Gu, Rui Tan, Shibo He, and Jiming Chen. 2023. MESEN: Exploit Multimodal Data to Design Unimodal Human Activity Recognition with Few Labels. *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems* (2023). <https://api.semanticscholar.org/CorpusID:268857170>
- [54] Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110* (2025).
- [55] Zhenyu Xu, Hailin Xu, Zhouyang Lu, Yingying Zhao, Rui Zhu, Yujiang Wang, Mingzhi Dong, Yuhu Chang, Qin Lv, Robert P. Dick, Fan Yang, Tun Lu, Ning Gu, and Li Shang. 2024. Can Large Language Models Be Good Companions? An LLM-Based Eyewear System with Conversational Common Ground. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 2, Article 87 (May 2024), 41 pages. doi:10.1145/3659600
- [56] Bufang Yang, Siyang Jiang, Lilin Xu, Kaiwei Liu, Hai Li, Guoliang Xing, Hongkai Chen, Xiaofan Jiang, and Zhenyu Yan. 2024. DrHouse: An LLM-empowered Diagnostic Reasoning System through Harnessing Outcomes from Sensor Data and Expert Knowledge. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8 (2024), 1 – 29. <https://api.semanticscholar.org/CorpusID:269929777>
- [57] Bufang Yang, Wenxuan Wu, Yitian Liu, and Hongxing Liu. 2022. A Novel Sleep Stage Contextual Refinement Algorithm Leveraging Conditional Random Fields. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–13. <https://api.semanticscholar.org/CorpusID:247131606>
- [58] Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. 2025. ContextAgent: Context-Aware Proactive LLM Agents with Open-World Sensory Perceptions. *ArXiv abs/2505.14668* (2025). <https://api.semanticscholar.org/CorpusID:278769319>
- [59] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19724–19731.
- [60] Xin Zhou, Aixin Sun, Yong Liu, Jie Zhang, and Chunyan Miao. 2023. SelfCF: A Simple Framework for Self-supervised Collaborative Filtering. *ACM Transactions on Recommender Systems* 1, 2 (June 2023), 1–25. doi:10.1145/3591469

## A COMMAND DATASET

### A.1 Smart-home scenario command dataset

Command Number	Command
<b>Explicit Command</b>	
1	make it less chilly in here
2	help me cool off
3	make it less stuffy in here
4	turn the AC off when it's cold outside
5	turn the heat off when it's hot outside
6	make it comfortable at night
7	make it less bright
8	get ready for bed
9	help me see better
10	keep it well-lit after sundown
11	use natural light once the sun is up
12	keep the lights off when I'm gone
13	check the front door
14	lock up
15	let the guest in
16	lock the door when I leave
17	let me know when there's a visitor
18	keep the home safe during the day
19	turn off the thermostat when I don't need it
20	save energy when I'm gone

Command Number	Command
21	help me save energy
22	help me lower my power bill
23	set up for a party
24	make it cozy in here
25	help me wind down
26	make it cozy when it rains
27	let me know when the weather is bad
28	help me sleep better
29	clean up the bedroom
30	stop cleaning
31	tidy up the house
32	stop cleaning after sunset
33	don't clean when people are here
34	clean up while I'm gone
35	give me some privacy
36	let some sun in
37	finish the coffee
38	keep it humid at night
39	I need coffee in the morning
40	let some sun in if the weather is nice
Loosen Command	
1	I'm too hot after my shower and I would like to get some cleaning done
2	I want to have coffee at the dining room table and look outside
3	I'm done watching the match and I'm gonna have some coffee and my dinner
4	People are leaving, let's wind the party down
5	I'm going to bed
6	I'm going to relax now
7	Can you change the reading lamp to a cooler tone
8	I'm heading out, please make sure everything is turned off
9	I'm leaving for work
10	My friend threw up in the kitchen
11	I'm not sure if I want coffee or tea. Pick one for me
12	Help me make my girlfriend happy
13	My parents are coming to visit
14	I'm hungry, let's get something ready
15	Watch something just like always
16	Congratulate my birthday
17	My dog just got the floor dirty. I'm gonna go give it a bath, and you take care of the rest
18	I'm about to start an important online meeting
19	I'm not taking a break this weekend. I'll be working all day
20	I'll do some exercise in living room
21	I want to take a nap
22	The weather outside sucks. Don't let me see it

Command Number	Command
23	I'm going out and won't be back for two days. You take care of everything at home
24	I'm home. And I'm going to sleep now. Go air out the rest of the place
25	Is someone outside the door? Make sure I'm safe
26	Start the night routine
27	I'll come back late, remember to take care of my cats
28	Let's have fun

## A.2 Smart-office scenario command dataset

Command Number	Command
Explicit Command	
1	it's a bit dark, can you help me see better
2	make it a little brighter in here
3	can you dim the lights a touch
4	make it cozy in here
5	give the room a calmer vibe
6	make the lighting more neutral
7	switch to a cooler light, please
8	turn on the desk lamp for me
9	turn off my desk lamp
10	make it a double espresso, please
11	it feels stuffy, get some air moving
12	make it a bit cooler in here
13	warm it up a little
14	set it to a comfortable temperature
15	turn on the AC
16	turn off the AC
17	put the fan on high
18	set the HVAC to auto
19	there's glare on my screen, close the blinds
20	open the blinds a bit
21	close the blinds halfway
22	let in more daylight
23	I need some coffee
24	the air feels dusty, turn on the air purifier
25	set the purifier to quiet mode
26	turn on the fresh air system
27	turn off the fresh air system
28	turn the room volume up a little
29	turn the volume down, it's too loud
30	lower the background music volume in the lounge
31	if there is no one watching the TV, then turn it off

Command Number	Command
32	we need warm water from the bathroom
33	turn off the water heater to save energy
34	the water from bathroom is too hot. Make it more comfortable
35	turn off the washing machine when it's work time
36	please unlock the room door
37	lock the room door
38	open the main entrance for a visitor
39	clean the working place
40	clean the realxing place
Loosen Command	
1	I feel gross and tired, can you help me reset
2	it's way too stuffy in here, but I don't want it cold
3	I can't focus with this vibe, make it better
4	make this place feel cozy, but not sleepy
5	I want it brighter, but I hate glare on the screen
6	it feels noisy and chaotic, can you make it calmer
7	I'm freezing, but the air feels dry
8	I'm about to nap, but I don't want to mess up my sleep tonight
9	I'm craving coffee, but I might regret it later
10	this place is a mess, I need it to look decent fast
11	I want to clean, but I don't know where to start
12	I need the kitchen to be guest-ready in 20 minutes
13	something smells off, can you deal with it
14	I spilled something earlier, help me fix it
15	I need to wind down, but I still have stuff to finish
16	I'm exhausted, but tomorrow's going to be intense, help me prep
17	this room feels off today... can you fix it
18	I'm feeling anxious, can you help me calm down
19	I want the dishes taken care of, but I don't want loud noises right now
20	I want my office ready by tomorrow morning, figure out the best way
21	I need a fresh start in here
22	I need to get into "deep work mode" for an hour, then switch me off
23	I want it quiet, but not dead silent
24	make it private in here
25	I'm leaving soon, make sure I didn't forget anything important
26	I want to feel safe and settled tonight
27	I'm going to bed, make it sleep-friendly, but don't mess up my alarms
28	I'm about to have a call, make the space look and feel professional

## B GENERATION PROCEDURE OF THE CONTEXT MEMORY DATASET

**LLM-Augmented Real-World Behavior Dataset.** We build our memory dataset based on a real-world dataset HOMER [38]. The original dataset records daily routines from five households, with 60 records per household (300

records in total). However, it only contains timelines of 22 high-level activities (e.g., brushing teeth or cooking) and lacks detailed device-interaction logs. We first fill overly long blank intervals in the original dataset. We follow the International Classification of Activities for Time-Use Statistics [18] from the United Nations Statistics Division, which includes around 400 sub-activities. After filtering out indoor activities that do not overlap with HOMER, we instruct an LLM to insert them into the original dataset to keep blank segments shorter than 20 minutes. The LLM prompt is shown below:

#### Prompt for expanding original dataset.

You are a smart home assistant. You need to fill in the blank periods in a daily activity timeline, ensuring semantic coherence before and after each inserted activity.

The raw time series is: {timeline}

Some gaps exceed 20 minutes; do NOT fill gaps between 'leave\_home' and 'come\_home'.

Below are several example fillings in CSV format: {example\_csv\_content}

Please follow these rules when choosing activities to fill:

1. If the entire series does NOT contain any of: {original\_activity} → prioritize inserting from this list.
2. Otherwise, consider inserting from: {additional\_activity}

Next, to obtain more diverse records from the limited real-world data, we summarize a base pattern for each household and then expand it into 10 extended patterns to simulate longer-term records. To model more fine-grained behaviors, we use few-shot prompting to guide the LLM to generate more realistic and detailed action units. The LLM prompt is shown below:

#### Prompt for pattern generation and behaviors generation.

You are a smart home assistant.

{raw\_content} is a base pattern of the {activity\_name} activity.

Please generate 10 extended patterns based on this base pattern. These extended patterns may vary in total duration scene transitions, selection of furniture and smart devices, usage scenarios, etc.

I would like you to preserve the original pattern's personal behavior characteristics while expanding. There are some smart device in this house: {device\_list}

**Real-World Fidelity Guarantees.** The fidelity of our dataset is rooted in a real-world dataset (HOMER), and LLMs help substantially to make it more comprehensive and diverse. Nevertheless, to ensure that the resulting sequences are plausible under an embodied interaction model, rather than merely linguistically plausible, we validate them in VirtualHome [40], a widely used simulation engine. VirtualHome provides an execution engine that runs routines as atomic actions in a 3D scene while maintaining explicit object/agent states (e.g., open/closed, on/off) and enforcing three constraints at each step: object affordances (the target supports the action), state-based preconditions and effects (the required state must hold and is updated accordingly), and reachability (the agent must be able to navigate to and physically interact with the target). Invalid steps therefore fail to execute. We use this as a feasibility filter and retain only routines whose full sequences execute successfully under the simulator's rules. For example, attempting to place an item into a closed fridge is rejected unless the routine first opens the fridge (and navigates into reach), and similarly, actions that mismatch an object's supported interactions (e.g., "open the television") are disallowed. To execute the routines in our dataset in VirtualHome, we convert them into action units in the required format using scripts, simulate each routine, and assign timestamps based on the simulation to validate the fidelity.

**Multi-resident behavior interleaving.** The above mentioned step produces single-user simulation traces. To study multi-user settings, we further construct multi-user households. We first compute the DTW average distance for each pair of users over time segments,  $\delta = avg(DTW(A_i, B_i))$ , where a lower DTW distance indicates higher similarity. We guide the LLM's merging strategy based on user similarity. For user pairs with low DTW distance (i.e., more similar behaviors), temporally overlapping activities are merged into a jointly performed event. For less similar pairs, we instead treat the two users as performing the activities sequentially. Finally, we feed the merged traces into VirtualHome again for validation. We then assign each record a timestamp derived from the simulation, ensuring temporal consistency and realism across the dataset.

## C QUESTIONNAIRE

## 1. DEMOGRAPHIC QUESTIONS

### How old are you?

- 18-24 years old
- 25-34 years old
- 35-44 years old
- 45-54 years old
- 54+ years old

### What is your highest level of education?

- High School or less
- Bachelor's degree
- Graduate or professional degree (MA, MS, PhD etc.)

### How familiar are you with smart environments?

- Frequently use services in smart environments
- Occasionally use some smart services
- Have little or no familiarity with smart environments.

### *(Multiple selections allowed)* Do you have any smart assistants you often use?

- None
- Siri
- Alexa
- Gemini Home
- Others: \_\_\_\_\_

## 2. SYSTEM USABILITY SCALE QUESTIONNAIRE

Based on your actual experience during the experiment, please answer the following questions using a scale from 1 to 5, where 1 means Strongly Disagree and 5 means Strongly Agree.

**I think that I would like to use this system frequently.**

1  2  3  4  5

**I found the system unnecessarily complex.**

1  2  3  4  5

**I thought the system was easy to use.**

1  2  3  4  5

**I think that I would need the support of a technical person to be able to use this system.**

1  2  3  4  5

**I found the various functions in this system were well integrated.**

1  2  3  4  5

**I thought there was too much inconsistency in this system.**

1  2  3  4  5

**I would imagine that most people would learn to use this system very quickly.**

1  2  3  4  5

**I found the system very cumbersome to use.**

1  2  3  4  5

**I felt very confident using the system.**

1  2  3  4  5

**I needed to learn a lot of things before I could get going with this system.**

1  2  3  4  5

### 3. ADDITIONAL QUESTIONNAIRE

Based on your actual experience during the experiment, please answer the following questions using a scale from 1 to 5, where 1 means Strongly Disagree and 5 means Strongly Agree.

1. **MemAura meets the basic needs described in the command.**

1  2  3  4  5

2. **MemAura meets the basic needs described in the command.**

1  2  3  4  5

3. **MemAura was highly efficient.**

1  2  3  4  5

4. **The token consumption of MemAura is affordable.**

1  2  3  4  5

5. **The response latency of MemAura is acceptable.**

1  2  3  4  5

6. **Please briefly describe your experience using MemAura.**