Appendix Overview

This appendix provides comprehensive details and additional analyses that complement our main paper on the proposed Mobile-Agent-RAG framework. We offer an in-depth look into the technical components, evaluation metrics, and a detailed case study to provide a holistic understanding of the proposed algorithm. Specifically, we present the full list of notations, a description of the Operator Agent's action space, detailed implementation specifics for our multi-agent system, the algorithms underpinning our retrieval mechanisms, and a thorough explanation of our knowledge base construction process. We also provide a complete list of the tasks in the Mobile-Eval-RAG benchmark, the full completion criteria used for task completion rate evaluation, and an expanded analysis of a case study demonstrating the Mobile-Agent-RAG's effectiveness. Specifically, it includes:

- **A. Notation Definitions**
- **B. Further Details on Experimental Setups**
- **C. Further Details on the Hierarchical Multi-agent Framework**
- **D. Retrieval Algorithms for Manager-RAG and Operator-RAG**
- **E. Further Details for Retrieval-Oriented Knowledge Base Collection**
- **F. Further Details for Mobile-Eval-RAG Construction**
- **G. Further Details on Experiment Implementations**
- **H. Further Details on Evaluation Metrics**
- **I. Completion Rate Evaluation Criteria**
- **J. Case Study**
- **K. More Analysis and Limitations**

We provide tables, algorithms, and figures in-place to keep each section self-contained for replication.

A. Notation Definitions

This appendix provides the main notations used to describe the Mobile-Agent-RAG framework and its components. The definitions, presented in Table 1, clarify the roles of each element and the information flow within our hierarchical multi-agent system for robust long-horizon multi-app mobile automation tasks.

Notation	Description
Environment	
I	User task instruction
A_t	Atomic action executed at timestep t
S_t	UI screenshot (Raw visual information) captured at timestep t
Agents	
M	Manager Agent
O	Operator Agent
P	Perceptor
R	Action Reflector
N	Notetaker
Working Memory	
V_t	Fine-grained visual information from P at timestep t
P_t	Overall plan at timestep t
T_t^{app}	Current subtask with identified app name at timestep t
G_t	Progress status at timestep t
N_t	Important notes at timestep t
F_t	Consecutive-error flag at timestep t
L^A	Action logs with outcome status
L^E	Error logs with feedback
Retrieval-Augmented Components	
MR	Manager-RAG
OR	Operator-RAG
K_{MR}	Manager-RAG knowledge base: set of D_{MR} with (I_{MR}, H_{MR}) pairs
K_{OR}^{app}	App-specific Operator-RAG knowledge base : set of D_{OR} with $(T_{OR}^{\text{app}}, S_{OR}, A_{OR})$ triplets
D_{MR}	A document with (I_{MR}, H_{MR}) pair in K_{MR}
D_{OR}	A document with $(T_{OR}^{\text{app}}, S_{OR}, A_{OR})$ triplet in K_{OR}^{app}
n_{MR}	The number of document D_{MR} in K_{MR}
n_{OR}^{app}	The number of document D_{OR} in one of the K_{OR}^{app}
I_{MR}	Task instruction text in D_{MR}
H_{MR}	Human operation steps in D_{MR}
T_{OR}^{app}	Subtask text in D_{OR}
S_{OR}	Reference screenshot in D_{OR}
A_{OR}	Atomic action (with arguments) in D_{OR}
$\mathcal{R}_M$	Top-k retrieved (I_{MR}, H_{MR}) for MR
$\mathcal{R}_O$	Top-1 retrieved $(T_{OR}^{\text{app}}, S_{OR}, A_{OR})$ for OR
$f(\cdot)$	Text encoder for embeddings by Contriever-MSMARCO

Table 1: Symbols used in Mobile-Agent-RAG, grouped by *Environment*, *Agents*, *Working Memory*, and *Retrieval-Augmented Components*. Index t denotes the interaction timestep. $\mathcal{R}_M$ returns top-k matches from K_{MR}, while $\mathcal{R}_O$ returns the top-1 match within the active app-specific K_{OR}^{app}. Embeddings $f(\cdot)$ are computed with Contriever-MSMARCO.

B. Further Details on Experimental Setups

To ensure a fair and consistent experimental setting with previous work Mobile-Agent-E, we synchronize the atomic action space, initial shortcuts, and initial tips of Mobile-Agent-RAG with those of the baseline model, Mobile-Agent-E. The shared action space is detailed in Table 2. The shared initial tips are as follows:

1. Do not add any payment information. If you are asked to sign in, ignore it or sign in as a guest if possible. Close any pop-up windows when opening an app.
2. By default, no APPs are opened in the background.
3. Screenshots may show partial text in text boxes from your previous input; this does not count as an error.
4. When creating new Notes, you do not need to enter a title unless the user specifically requests it.

Operation	Description
Open_App(app_name)	Opens the app "app_name" from the Home screen.
Tap(x, y)	Taps the current screen at position (x, y).
Swipe(x1, y1, x2, y2)	Swipes from (x1, y1) to (x2, y2) for scrolling content or navigating apps.
Type(text)	Types "text" into an active input box.
Enter(·)	Presses the Enter key.
Back(·)	Returns to the previous screen or state.
Home(·)	Returns to the home page.
Wait(·)	Pauses execution for 10 seconds to allow for page loading.
Tap_Type_Enter(x, y, text)	A composite action: taps an input box at (x, y), types the "text", then presses Enter.

Table 2: A detailed list of the atomic actions and composite shortcuts available to the agent. This identical action space is adopted from the baseline model, `Mobile-Agent-E`, to ensure that performance gains are attributed to our retrieval-augmented components rather than a more expressive action set.

C. Further Details on the Hierarchical Multi-agent Framework

This section provides a more detailed look into the implementation of our proposed `Mobile-Agent-RAG`'s hierarchical multi-agent framework, which is inherited from `Mobile-Agent-E`. Notably, all core agents and support modules, *with the exception of the Perceptor*, are categorized as reasoning agents.

Reasoning Agents: Manager, Operator, Action Reflector, and Notetaker Our framework's four reasoning agents are powered by specific API versions of leading large language models. For our experiments, we use Gemini-1.5-pro-latest, GPT-4o, and Claude-3.5-Sonnet-latest as the underlying inference engines.

Perceptor The Perceptor module is largely based on the implementation in `Mobile-Agent-E`. For fine-grained text information, we leverage the DBNet[1] model from the ModelScope platform for Optical Character Recognition (OCR) text detection, and the ConvNextViT-document[2] model for character recognition. For fine-grained icon information, we employ GroundingDINO for icon grounding and use Qwen-VL-Plus to generate descriptive captions for each cropped icon.

D. Retrieval Algorithms for Manager-RAG and Operator-RAG

This section provides a detailed description of the retrieval algorithms central to our framework. The `Mobile-Agent-RAG` system leverages two distinct retrieval components: Manager-RAG and Operator-RAG. The former is designed to retrieve high-level strategic guidance to inform the agent's overall plan, while the latter focuses on retrieving app-specific operational knowledge to enable precise atomic actions. We outline the algorithms for each component below.

[1]https://modelscope.cn/models/iic/cv_resnet18_ocr-detection-db-line-level_damo

[2]https://modelscope.cn/models/iic/cv_convnextTiny_ocr-recognition-document_damo

Algorithm 1: Manager-Retrieve Algorithm

Input: Task instruction I_{query}
Parameter: Manager-RAG knowledge base $K_{MR} = \{(I_{MR}^{(i)}, H_{MR}^{(i)})\}_{i=1}^{n_{MR}}$; embedding function $f(\cdot)$; number k
Output: Top-k retrieved results $\mathcal{R}_M = \{(I_{MR}^{(j)}, H_{MR}^{(j)})\}_{j=1}^{k}$

$v_{\text{query}} \leftarrow f(I_{\text{query}})$
Initialize $V_{MR}^{\text{sim}} \leftarrow [\,]$
for all $(I_{MR}^{(i)}, H_{MR}^{(i)}) \in K_{MR}$ **do**
  $v_{MR}^{(i)} \leftarrow f(I_{MR}^{(i)})$
  $\text{sim}_{MR}^{(i)} \leftarrow \cos(v_{\text{query}}, v_{MR}^{(i)})$
  Append $(\text{sim}_{MR}^{(i)}, I_{MR}^{(i)}, H_{MR}^{(i)})$ to V_{MR}^{sim}
end for
Sort V_{MR}^{sim} by descending $\text{sim}_{MR}^{(i)}$
$\mathcal{R}_M \leftarrow$ Top-k documents from V_{MR}^{sim}
return $\mathcal{R}_M$

Manager-RAG The Manager-RAG's retrieval algorithm is designed to provide high-level guidance for plan generation. Its knowledge base (K_{MR}), which is manually curated, contains n_{MR} ($n_{MR} = 25$ in our experiment) documents (D_{MR}). Each document consists of a mobile task instruction (I_{MR}) and a sequence of human-annotated operation steps (H_{MR}). When a new task instruction (I_{query}) is provided, the system retrieves the top-k (We set $k = 3$ in our experiment) most semantically similar documents from K_{MR} using embeddings from Contriever-MSMARCO. The retrieval process is formally described in Algorithm 1.

Operator-RAG The Operator-RAG's retrieval algorithm is responsible for retrieving app-specific operational knowledge to support the generation of atomic actions. Its knowledge base (K_{OR}^{app}) is semi-automatically constructed and partitioned by application domain (e.g., YouTube, Maps) to prevent cross-app interference. Each knowledge base contains n_{OR}^{app}. This parameter's value is specific to each application. Each knowledge base document (D_{OR}), contains a subtask description ($T_{OR}^{\text{app}(i)}$), a reference screenshot ($S_{OR}^{(i)}$), and the

Algorithm 2: Operator-Retrieve Algorithm

Input: Current subtask and identified app name $T_{\text{query}}^{\text{app}}$
Parameter: App-specific knowledge base $K_{OR}^{\text{app}} = \{(T_{OR}^{\text{app}(i)}, S_{OR}^{(i)}, A_{OR}^{(i)})\}_{i=1}^{n_{OR}^{\text{app}}}$; embedding function $f(\cdot)$
Output: Top-1 retrieved result $\mathcal{R}_O = (T_{OR}^{app(j)}, S_{OR}^{(j)}, A_{OR}^{(j)})$

$v_{\text{query}} \leftarrow f(T_{\text{query}}^{\text{app}})$
Initialize $V_{OR}^{\text{sim}} \leftarrow [\,]$
for all $(T_{OR}^{app(i)}, S_{OR}^{(i)}, A_{OR}^{(i)}) \in K_{OR}^{\text{app}}$ **do**
  $v_{OR}^{(i)} \leftarrow f(T_{OR}^{app(i)})$
  $\text{sim}_{OR}^{(i)} \leftarrow \cos(v_{\text{query}}, v_{OR}^{(i)})$
  Append $(\text{sim}_{OR}^{(i)}, T_{OR}^{app(i)}, S_{OR}^{(i)}, A_{OR}^{(i)})$ to V_{OR}^{sim}
end for
Sort V_{OR}^{sim} by descending $\text{sim}_{OR}^{(i)}$
$\mathcal{R}_O \leftarrow$ Top-1 document from V_{OR}^{sim}
return $\mathcal{R}_O$

corresponding atomic action and arguments ($A_{OR}^{(i)}$). Given a current subtask, the system retrieves the top-1 relevant document from the active app's knowledge base. The process is outlined in Algorithm 2.

E. Further Details for Retrieval-Oriented Knowledge Base Collection

This section details the construction and collection process for the two distinct knowledge bases, which are critical for the retrieval-augmented components of our proposed framework. Both the Manager-RAG and Operator-RAG knowledge bases are built through a combination of manual and semi-automated strategies to ensure high quality and relevance.

Manager-RAG Knowledge Base Collection The Manager-RAG knowledge base is meticulously compiled to provide high-level strategic guidance. Its construction involves a multi-stage process:

1. **Task Execution by Experimenters:** As illustrated in Table 6 and Table 7, we engage multiple human experimenters to perform, on real mobile devices, the top 50% of Mobile-Eval-RAG tasks from each category. This setup ensures that operations are effective and transferable to real-world conditions. During each run, we log screens, timestamps, and atomic actions to produce raw, verifiable trajectories.
2. **Rigorous Sequence Filtering:** From the raw trajectories, we remove erroneous or incomplete trials, deduplicate near-identical runs, and apply a *minimal-success* criterion: for each task, we retain the shortest real-world action sequence that reliably completes the task. This filtering keeps optimized pathways and eliminates suboptimal or redundant operations.
3. **Data Structuring:** We structure the curated data into a schema retrievable by the Manager-RAG module. Each knowledge-base document includes: (1) **Task Instruction**, representing the overall task objective in text; and (2) **Human Steps**, representing the corresponding concise operation steps that provide high-level guidance.

We normalize terminology, encode the *Task Instruction* field into embeddings, and store entries in a vector database to enable efficient similarity search during inference. Table 3 shows several examples of the structured data entries for the Manager-RAG knowledge base.

Operator-RAG Knowledge Base Collection The Operator-RAG knowledge base is carefully constructed to provide accurate, executable operational guidance. Its construction involves a multi-stage process:

1. **Data Collection:** As shown in Table 6 and Table 7, we collect Operator-RAG data by recording precise agent actions while executing Mobile-Eval-RAG tasks. **Strategic Minimization of Human Intervention:** To reduce manual overhead and better reflect autonomous behavior, we strategically minimize human involvement. Human guidance is only provided when the agent fails to perform correctly, ensuring correctness while maintaining a high degree of autonomy in the recorded behavior. **Instance Recording:** Each recorded instance includes (1) the current subtask described in text, (2) a screenshot of the corresponding UI state, and (3) the atomic action required to achieve the subtask. A representative subset of correctly executed operational instances for the Maps App is presented in Table 5.
2. **Rigorous Data Cleansing:** Following initial data collection, we perform strict cleansing to eliminate erroneous and redundant entries. This step ensures that the knowledge base maintains only high-quality, accurate, and reliable operational examples.
3. **Data Structuring:** We structure the cleaned data into a schema retrievable by the Operator-RAG module. Each knowledge-base document comprises three key components: (1) **Subtask**, a textual description of the specific subtask objective; (2) **Screenshot**, the visual UI state corresponding to the subtask, stored as a local file path; and (3) **Action**, the atomic action along with its arguments required to accomplish the subtask.

We encode the *Subtask* field into embeddings and store the structured entries in a vector database, enabling efficient similarity-based retrieval during inference.

F. Further Details for Mobile-Eval-RAG Construction

This section provides a detailed overview of the proposed benchmark dataset, namely **Mobile-Eval-RAG**, which is specifically designed to assess multi-app collaboration in long-horizon mobile agents. We outline the key features of the dataset and highlight its unique design philosophy, which makes it particularly suitable for evaluating the generalization capabilities of the proposed RAG systems. Additionally, we provide a detailed comparison with the popular benchmark, e.g., **Mobile-Eval-E**, to clarify our evaluation focus.

Task Instruction	Human steps
Find the best ramen place in Chicago Loop with at least 500 reviews and rating over 4.5. Write a review summary in Notes.	open Maps app, tap on the search bar, type "the best ramen place in Chicago Loop", enter, swipe down to find more results, tap on the "RAMEN-SAN Whisky Bar", swipe down to find more reviews, tap home, tap Notes, tap "+", tap "text", type the review summary
Look for a family-friendly restaurant in Urbana suitable for kids. Write a short summary in Notes.	open Maps app, tap on the search bar, type "family-friendly restaurant in Urbana", enter, tap the filter, tap "good for kids", tap apply, tap on the first result, swipe down to find more information, swipe down to find more information, swipe down to find more information, tap home, tap Notes, tap "+", tap "text", type the short summary
Search for breakfast buffet places near me with good reviews. Compare 2 and write in Notes.	open Maps app, tap on the search bar, type "breakfast buffet places", enter, tap filter, tap "Distance", tap "Apply", tap on the first result has review, swipe down to find more information, back, tap on the second result has review, swipe down to find more information, tap home, tap Notes, tap "+", tap "text", type the summary to compare the two restaurant
Find a hotpot restaurant near a university campus. Write the address and the average user rating into Notes.	open Maps app, tap on the search bar, type "hotpot restaurant near a university campus", enter, tap the first result, swipe down to find more information, tap home, tap Notes, tap "+", tap "text", type the address and the average user rating
Find a Chinese restaurant in Chicago with rating over 4.5 that offers takeout. Save 3 dishes and their prices in Notes.	open Maps app, tap on the search bar, type "Chinese restaurant in Chicago", enter, tap filter, tap "4.5 star", tap "Takeaway", tap "Apply", tap the first result, tap menu, swipe down to find more information, swipe down to find more information, swipe down to find more information, tap home, tap Notes, tap "+", tap "text", type the 3 dishes and their prices

Table 3: Representative documents with **task instructions** and **human steps** from *Restaurant Recommendation* tasks used to construct the **Manager-RAG** knowledge base.

More Key Features of Mobile-Eval-RAG Mobile-Eval-RAG is a comprehensive benchmark dataset designed to evaluate the capability of multi-app collaboration across mobile agents. It simulates real-life scenarios that require agents to use multiple applications simultaneously to complete complex, daily tasks. The core idea is to test a system's practical operational ability in a cross-app mobile device environment, moving beyond single-app proficiency. Additionally, a significant feature of Mobile-Eval-RAG is its emphasis on information integration and cross-platform collaboration. Most tasks require collecting and synthesizing data from multiple sources before performing analysis, comparison, and summarization. For example, in a restaurant recommendation task, the agent must search for restaurants in a map application, review ratings and comments, and then write a summary in a notes app. Similarly, an online shopping task might involve searching for products on a platform like Walmart, watching related reviews on YouTube, and finally summarizing the pros and cons in a notes app.

Comparison Between Mobile-Eval-RAG and Existing Benchmarks Table 4 presents a comparative analysis between our proposed benchmark, **Mobile-Eval-RAG**, and several representative mobile automation benchmarks. Mobile-Eval-RAG is divided into two subsets: **Mobile-Eval-RAG (Simple)** and **Mobile-Eval-RAG (Complex)**, which correspond to relatively simple and complex task scenarios within the same evaluation framework.

Compared to existing datasets such as Mobile-Eval, DroidTask, and AppAgent, Mobile-Eval-RAG offers notable advantages in several key dimensions. First, it contains a higher proportion of cross-app tasks (100% in both subsets), which more accurately reflects real-world multi-app collaboration. Second, it features significantly longer task horizons, with an average execution length of 14.05 steps for simple tasks and up to 18.80 steps for complex tasks—highlighting its suitability for evaluating long-horizon reasoning capabilities. Third, unlike most prior datasets (with the exception of Mobile-Eval-E), our benchmark incorporates well-defined **Completion Rate Evaluation Criteria** (see **Appendix I**), enabling more objective and fine-grained evaluation of task progress.

Furthermore, while both Mobile-Eval-RAG and Mobile-Eval-E aim to support complex automation tasks, they differ significantly in their design focus. Mobile-Eval-E features a broader range of task types and a progressively distributed difficulty spectrum, making it well-suited for evaluating self-learning and lifelong learning in open-ended scenarios. In contrast, Mobile-Eval-RAG emphasizes generalization under controlled task variations, making it especially suitable for benchmarking **RAG-based systems**. Each task category (e.g., "Restaurant Recommendation") in Mobile-Eval-RAG consists of highly similar applications and interaction patterns, but with subtle variations in content and context. This concentrated design yields two key benefits: (1) it

Benchmark	Multi-App / All	Apps	Avg Steps	CR Crit.	LongH.	RAG Eval.
Mobile-Eval	3 / 33	10	5.55	-	-	-
DroidTask	0 / **158**	13	5.56	-	-	-
Mobile-Eval-v2	4 / 44	10	5.57	-	-	-
AppAgent (General)	0 / 45	9	6.31	-	-	✓
AndroidWorld	10 / 116	**20**	9.13	-	-	-
Mobile-Eval-E	19 / 25	15	14.56	✓	✓	-
AppAgent (Long)	0 / 5	5	15.40	-	✓	✓
Mobile-Eval-RAG (simple)	20 / 20	4	14.05	✓	✓	✓
Mobile-Eval-RAG (complex)	**30** / 30	7	**18.80**	✓	✓	✓

Table 4: Comparison of **Mobile-Eval-RAG** (ours) with existing mobile automation benchmarks. **Mobile-Eval-RAG (simple)** and **Mobile-Eval-RAG (complex)** represent the simple and complex task subsets of our proposed benchmark, respectively. **Multi-App / All** denotes the number of cross-app tasks versus the total number of tasks. **Apps** is the number of unique applications involved. **Avg Steps** is the average number of steps per task. **CR Crit.** indicates whether Completion Rate Evaluation Criteria (see **Appendix I**) are defined. **LongH.** denotes whether the benchmark supports long-horizon task execution. **RAG Eval.** shows if the benchmark is suitable for RAG-based evaluation.

enables the reuse of generalized retrieval strategies across tasks within a category, and (2) it prevents systems from relying on rote memorization, thereby requiring deeper understanding and adaptive reasoning. As a result, Mobile-Eval-RAG serves as a more rigorous testbed for evaluating the generalization capabilities of mobile agents in RAG settings.

G. Experiment Implementations

We use Android Debug Bridge[3] (ADB) to allow the Operator Agent to perform atomic actions on real mobile devices. We select 8 widely used mobile applications and design corresponding tasks for multi-application scenarios. Each task allows up to 30 steps, with no more than five consecutive repetitions of the same action. To ensure reproducibility, MLLM API calls are limited to 2048 tokens with a temperature of 0. Human annotators monitor and evaluate system performance in real time using predefined metrics. For consistency with our `Mobile-Agent-RAG`'s implementation, we use only the default "Shortcuts" and "Tips" components of `Mobile-Agent-E` across all frameworks. For methods requiring pre-training, such as `Mobile-Agent-RAG` and `AppAgent`, we allocate 50% of the Mobile-Eval-RAG dataset for knowledge base construction and use the remaining 50% as a unified test set to ensure fair and consistent evaluation. Empirically, we set $k = 3$ in all exmperiments.

H. Further Details on Evaluation Metrics

To comprehensively evaluate the performance of mobile agents in complex automation tasks, we draw inspiration from classic frameworks such as `Mobile-Agent-E` and `Mobile-Agent-v2` to construct a multidimensional evaluation metric system. This system encompasses task completion effectiveness (**Success Rate** and **Completion Rate**) and fine-grained operational accuracy (**Operator Accuracy** and **Reflector Accuracy**), aiming to assess whether the agent can perform tasks correctly and with high quality. While these metrics effectively reflect the agent's competence, they are insufficient for capturing execution efficiency.

Considering the practical importance of efficiency in real-world scenarios such as mobile device automation, we further introduce two complementary metrics: **Steps** and **Efficiency**, which measure the operational cost of task completion and the contribution of each step to the task completion progress. Together, these metrics form a systematic, fine-grained, and objective evaluation framework that assesses agent performance from three perspectives—task effectiveness, action-level accuracy, and execution efficiency—making it suitable for the evaluation of long-horizon, multi-step tasks.

1. **Success Rate (SR, %):** The Success Rate evaluates whether a task is completed successfully under three essential conditions: (1) The task is completed within 30 steps; (2) The agent does not make any erroneous task completion judgments; and (3) The agent does not repeat the same action more than five consecutive times. A task is counted as a "success" only if all three of these conditions are met. If any one of the conditions is not satisfied, the task is counted as a "failure". This metric reflects whether the agent can accomplish a goal efficiently, without becoming trapped in repetitive behavior or terminating prematurely.

2. **Completion Rate (CR, %):** Completion Rate is used to quantify the degree to which a task has been completed. Metrics like SR have been adopted in prior work, such as `Mobile-Agent-v2`. However, as illustrated in **Appendix I**, its original definition exhibits certain limitations in terms of accuracy and applicability. To address this, we tackle the shortcomings of existing metrics by introducing a tailored **Completion Rate Evaluation Criteria** for each task. Based on these criteria, the Completion Rate is recalculated to more precisely reflect task progress. The formal definition of Completion Rate is

[3]https://developer.android.google.cn/tools/adb

provided as follows.

$$\text{CR} = \frac{\text{Number of completed items}}{\text{Number of total items}} \quad (1)$$

This metric is particularly useful when tasks are partially completed or vary in structure and length.

3. **Operator Accuracy (OA, %):** Operator Accuracy measures how accurately the Operator module selects and executes the correct atomic actions required to advance each subtask. An action is considered correct if it is successfully executed on screen and contributes to progress. The metric is defined as:

$$\text{OA} = \frac{\text{Number of correct operations in the task}}{\text{Total steps in the task}} \quad (2)$$

This metric directly reflects the precision of the agent's action selection and execution.

4. **Reflector Accuracy (RA, %):** Reflector Accuracy evaluates whether the Action Reflector module can correctly judge the outcome of the Operator's action. It captures the proportion of steps in which the reflection correctly determines whether the action has advanced the current subtask. The metric is defined as:

$$\text{RA} = \frac{\text{Number of correct reflections in the task}}{\text{Total steps in the task}} \quad (3)$$

This metric is essential for understanding the system's capacity for self-assessment and timely error correction.

5. **Steps:** This metric records the number of core operational steps required to complete a task. It serves as a direct measure of task execution cost. In our framework, the maximum number of steps is capped at 30 in accordance with the Success Rate condition.

6. **Efficiency:** Efficiency captures the average per-step contribution to task completion. It measures how effectively the agent advances toward the goal with each step. A higher efficiency value indicates that each step contributes more significantly to task progress. It is defined as:

$$\text{Efficiency} = \frac{\text{CR of the task}}{\text{Total steps in the task}} \quad (4)$$

This metric reflects the overall quality of the agent's exploration and decision-making process.

In summary, these evaluation metrics constitute a rigorous and comprehensive framework for assessing mobile agent systems. They jointly capture various aspects of performance, including task-level success, progression detail, operation precision, reflection correctness, and behavioral efficiency. This framework is especially well suited for evaluating Mobile-Agent-RAG under human-in-the-loop settings, where understanding nuanced task behavior and adaptation strategies is essential.

I. Completion Rate Evaluation Criteria

To more accurately assess agent performance in multi-step mobile tasks, this work introduces a fine-grained evaluation mechanism referred to as the **Completion Rate Evaluation Criteria**. This mechanism is designed to address the limitations of the traditional Success Rate (SR) metric, which provides only a coarse-grained view of task outcomes in complex scenarios.

Conventional SR metrics evaluate task execution using a binary outcome—either "success" or "failure"—for the entire task. This approach fails to capture whether an agent has made partial progress or completed key subcomponents of the task. In real-world mobile settings, tasks often involve a sequence of operations, such as opening applications, searching for content, filling out forms, or interacting across multiple apps. Even when an agent does not complete all steps, it may successfully carry out a majority of essential actions. Therefore, relying solely on SR lacks the granularity needed to fully understand agent performance and hinders targeted model improvements.

To overcome this limitation, each evaluation task is decomposed into a set of clearly defined and independently verifiable subgoals, referred to as *completion items*, which collectively constitute the task's Completion Criteria. The number of subgoals is determined by task complexity: tasks involving two applications are assigned 8 completion items, while tasks involving three applications are assigned 10. Each subgoal can be individually assessed, enabling a fine-grained quantification of task progress. The number of completed subgoals directly maps to the agent's **Completion Rate (CR)**, providing a more nuanced indicator of intermediate performance.

The initial set of completion items is automatically generated using the large language model **Gemini-2.5-pro**, which is prompted to identify objective and critical progress points. These model-generated items were then rigorously reviewed and refined by human annotators to ensure clarity, accuracy, and comprehensive task coverage. All subgoals are assigned equal weight, ensuring that the final CR score is both fair and comparable across different tasks.

This evaluation framework brings three key benefits:

- **Fine-grained Measurement:** Offers detailed insight into agent behavior by quantifying intermediate progress, improving objectivity and expressiveness.
- **Complexity-aware Comparison:** Accounts for task complexity by using a standardized structure, enabling fair cross-task comparisons.
- **Interpretability and Reproducibility:** Human-validated subgoals make results more transparent and replicable, helping researchers diagnose performance bottlenecks and improve system design.

In summary, the Completion Criteria provide a more stable, precise, and extensible foundation for systematic evaluation of mobile agents, addressing the limitations of binary metrics in multi-step task environments.

J. Case Study

In this section, we perform case studies to evaluate `Mobile-Agent-RAG` in a multi-app mobile setting that mirrors real-world "app-hopping" scenarios.

Case Study of End-to-End Task Execution In this case, we provide a comprehensive, step-by-step walkthrough of how the agent completes a complex long-horizon cross-app task on a real-world smartphone in Figures 1 and 2. Throughout the process, the system repeatedly follows an iterative "**Planning → Execution → Reflection**" loop, as illustrated below.

- **Phase 1 - Planning (*Manager Agent + Manager-RAG*)** Before any UI action is issued, Manager-RAG intervenes; guided by the overall task objective, it queries its knowledge base and retrieves relevant Manager-RAG documents (To ensure a clear demonstration in the figure, we set $k = 1$ in this case.) that contains the task instruction together with the corresponding human steps, as shown in the "Manager-RAG" section of the figure. The exemplar constrains the search space, steers the Manager's high-level strategy, and prevents sub-optimal plans.
- **Phase 2 - Execution (*Operator Agent + Operator-RAG*)** After the Manager provides the high-level plan and current sub-goal, the Operator must emit atomic UI actions to satisfy that sub-goal. It first captures the live screenshot shown on the left under "Current Screenshot"; it then consults Operator-RAG, which returns the top-1 (subtask, screenshot, action) triplet. The screenshot is located under the "Retrieved Image from Operator RAG" section on the right, while the retrieved action guidance is within the "Operator-RAG" section. By fusing the live screenshot with the retrieved exemplar, the Operator accurately locates the target UI widget and generates a precise atomic action with all required arguments. The full list of actions and their arguments is enumerated in the "Operator" section of the figure.
- **Phase 3 - Reflection (*Action Reflector*)** Once an atomic action completes, the Action Reflector evaluates the result by analysing the post-action screenshot and its fine-grained metadata, confirms whether the desired effect was achieved, updates the progress tracker, and decides whether to proceed, retry, or trigger a plan revision. The "Action Reflector" panel logs every verdict and the updated task status, forming a dynamic feedback loop that allows `Mobile-Agent-RAG` to adapt to real execution conditions.

Case-study Comparison with `Mobile-Agent-E` To verify `Mobile-Agent-RAG`'s effectiveness of retrieval augmentation on complex tasks, we conduct a head-to-head case study, as shown in Figure 3. In this scenario, `Mobile-Agent-E`'s Operator frequently misidentifies visually similar buttons and drifts into local exploration, while its Manager occasionally produces sub-optimal plans on unseen screens, both of which lower overall efficiency. By introducing Operator-RAG and Manager-RAG—thereby injecting human-annotated experience—`Mobile-Agent-RAG` issues more accurate atomic actions and maintains a coherent plan, significantly accelerating execution and demonstrating improved robustness and generalisation.

K. More Analysis and Limitations

This section presents a deeper analysis of the mechanisms that drive the effectiveness of `Mobile-Agent-RAG`, identifies key limitations observed during experimentation, and compares the retrieval-based framework with evolutionary approaches. These insights provide guidance for future system improvements and the exploration of hybrid designs.

How Retrieval Enhances Planning and Execution The strong performance of `Mobile-Agent-RAG` is rooted in its ability to ground agent behavior in verified human trajectories, which substantially mitigates the hallucination problem that commonly affects autonomous agents.

- **Manager-RAG** strengthens high-level planning by retrieving similar task patterns that offer strategic guidance. These examples effectively reduce the planning search space and prevent the agent from engaging in inefficient or suboptimal strategies. The retrieved task templates are not static—they are adapted to fit the nuances of novel task variations while maintaining consistency with proven strategies.
- **Operator-RAG** enhances low-level execution by retrieving contextually relevant examples of successful UI interactions. This is especially crucial in visually complex mobile environments, where subtle differences in UI layout can significantly impact action validity. The retrieved cases help the agent infer correct actions from the current visual context, providing robust grounding for precise operations.

Observed Limitations and Failure Modes Despite its overall effectiveness, `Mobile-Agent-RAG` reveals two primary types of failure that highlight important limitations:

- **Limitations in Knowledge Base Coverage:** Failures may arise when the agent encounters task contexts or UI states that are underrepresented or missing from the retrieval corpus. These cases highlight the need for a more comprehensive and diverse knowledge base. Leveraging active learning to identify and fill such gaps could significantly enhance retrieval effectiveness and overall system robustness.
- **Challenges in Visual Perception:** In some cases, the agent struggles to interpret complex or unfamiliar UI layouts, even when relevant examples are successfully retrieved. These failures indicate that retrieval alone is insufficient to overcome fundamental visual understanding limitations, emphasizing the need for more capable visual perception modules to support retrieval-based reasoning.

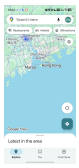
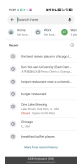
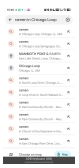
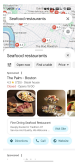
 **Subtask:** Tap Maps app. **Action:** Open_App at {“app_name”: “Maps”}	 **Subtask:** Tap the search bar. **Action:** Tap at {“x”: 404, “y”: 260}	 **Subtask:** Type “ramen in Chicago Loop”. **Action:** Type at {“text”: “ramen in Chicago Loop”}	 **Subtask:** Tap Enter. **Action:** Enter at null
 **Subtask:** Tap the search bar. Type “ramen in Chicago Loop”. Tap Enter. **Action:** Tap_Type_and_Enter at {“x”: 200, “y”: 250, “text”: “ramen in Chicago Loop”}	 **Subtask:** Tap “Filter”. **Action:** Tap at {“x”: 110, “y”: 1068}	 **Subtask:** Tap “4.5 Stars & up”. **Action:** Tap at {“x”: 1034, “y”: 902}	 **Subtask:** Tap “Apply. **Action:** Tap at {“x”: 917, “y”: 2642}
 **Subtask:** Swipe up to see more results if needed. **Action:** Swipe at {“x1”: 630, “y1”: 1400, “x2”: 630, “y2”: 280}	 **Subtask:** Tap on a ramen place with at least 500 reviews and rating over 4.5. **Action:** Tap at {“x”: 250, “y”: 1600}	 **Subtask:** Tap Home. **Action:** Home at null	 **Subtask:** Tap a restaurant result. **Action:** Tap at {“x”: 355, “y”: 1162}
 **Subtask:** Tap Liuyishou Hotpot(Chicago). **Action:** Tap at {“x”: 609, “y”: 2420}	 **Subtask:** Tap “Takeaway”. **Action:** Tap at {“x”: 335, “y”: 1905}	 **Subtask:** Tap the first restaurant in the results. **Action:** Tap at {“x”: 300, “y”: 1300}	 **Subtask:** Tap “Photos”. **Action:** Tap at {“x”: 717, “y”: 2207}
 **Subtask:** Tap Maps app. **Action:** Tap at {“x”: 197, “y”: 333}	 **Subtask:** Tap “Top rated”. **Action:** Tap at {“x”: 1102, “y”: 1069}	 **Subtask:** Tap “Attractions”. **Action:** Tap at {“x”: 777, “y”: 153}	 **Subtask:** Tap on “Live & Kicking Lobsters” from the search results. **Action:** Tap at {“x”: 100, “y”: 650}

Table 5: Representative documents with **subtasks**, **screenshots**, and **actions** collected from executing Maps app task and used to construct the “Maps” part of the **Operator-RAG** knowledge base.

Category	Apps	Task Instruction
Information Searching	Chrome, Notes	Research the latest green energy innovations from 2025 and summarize top 3 technologies in Notes.
	YouTube, Notes	Find 3 healthy breakfast recipes on YouTube that are under 10 mins to cook. Write the summary in Notes.
	Chrome, Notes	Find recent space discoveries from NASA or SpaceX. Summarize 2 major ones in Notes.
	Chrome, Notes	Find recommended books for beginners in machine learning. Summarize book titles and author recommendations in Notes.
	Chrome, Notes	Research top 5 internet safety tips for teenagers in 2025. Write a short guideline in Notes.
	Chrome, Notes	Search for electric car models under $35k in Chrome. Note 3 options and main features in Notes.
	YouTube, Notes	Find a beginner-friendly daily yoga video on YouTube. Note down the video title and channel name and write a routine summary in Notes.
	Chrome, Notes	Find 2023–2024 news or articles about global plastic bans. Summarize 3 countries' policies in Notes.
	Chrome, Notes	Research next 3 major space missions. Write a timeline summary in Notes.
	Chrome, Notes	Find articles or guides comparing coffee brewing methods. Summarize key differences and ideal use cases for each method in Notes.
What's Trending	X, Notes	Search for 3 fun or useful mobile apps trending on X in 2025. Summarize features in Notes.
	Chrome, X, Notes	Look for trending tech startups in 2025. Use X and Chrome to summarize 3 promising ones.
	YouTube, Notes	Find the top trending music video on YouTube. Analyze the comments and summarize what people like in Notes.
	X, Notes	Check what's trending for holidays in Tokyo. Search on X and summarize top places or events.
	YouTube, Notes	Find 3 popular vloggers who post daily life or travel content. Summarize what makes their videos engaging.
	X, Notes	Search on X for discussions on 2025 metaverse chat tools. Summarize 3 tools in Notes.
	X, Notes	Check recent posts about AI-generated music. Find 2 popular songs or tools and write a summary.
	Chrome, Notes	Research what types of games are trending in 2025. Use Chrome and summarize 3 trends in Notes.
	YouTube, Notes	Find 3 YouTube Shorts creators with viral content in 2025. Summarize what makes their content engaging.
	X, Notes	Look for hype around 2024 memecoins on X. Note top 2 trending coins and community sentiment.

Table 6: Examples of **simple** operation tasks proposed by **Mobile-Eval-RAG**, covering two categories: *Information Searching* and *What's Trending*. Each task specifies a real-world mobile scenario involving 2–3 apps and requires users to retrieve, analyze, and summarize content in Notes.

Category	Apps	Task Instruction
Restaurant Recommendation	Maps, Notes	Find the best ramen place in Chicago Loop with at least 500 reviews and rating over 4.5. Write a review summary in Notes.
	Maps, Notes	Look for a family-friendly restaurant in Urbana suitable for kids. Write a short summary in Notes.
	Maps, Notes	Search for breakfast buffet places near me with good reviews. Compare 2 and write in Notes.
	Maps, Notes	Find a hotpot restaurant near a university campus. Write the address and the average user rating into Notes.
	Maps, Notes	Find a Chinese restaurant in Chicago with rating over 4.5 that offers takeout. Save 3 dishes and their prices in Notes.
	Maps, Notes	Search for nearby vegan breakfast spots. Pick one with best rating and write a short review in Notes.
	Maps, Notes	Find a seafood restaurant suitable for a romantic dinner. Include menu highlight in Notes.
	Maps, X, Notes	Find a trending brunch place in Chicago in Maps. Check user posts on X and summarize in Notes.
	Maps, Notes	Find 3 burger restaurants within 5km. Write a comparison of reviews and prices in Notes.
	Maps, Notes	Search for a dessert shop open after 10pm. Check user reviews and note recommended items.
Online Shopping	Walmart, Notes	Find a tablet under $150 on Walmart. Compare 2 brands and summarize specs in Notes.
	Walmart, YouTube	Search for a portable speaker under $100. Watch 1–2 YouTube reviews and write pros/cons in Notes.
	Walmart, Notes	Look for an affordable desk lamp for study with eye-care mode. Compare ratings & product descriptions and note in Notes.
	Walmart, Notes	Find top-rated pet supplies under $30 for dogs on Walmart. Compare options, prices and descriptions in Notes.
	Walmart,YouTube, Notes	Find a student laptop under $400. Check YouTube reviews and summarize 3 models in Notes.
	Walmart,YouTube, Notes	Find an ergonomic chair under $120 on Walmart. Watch YouTube reviews and write pros/cons in Notes.
	Walmart, Notes	Find a blender under $50 with good reviews on Walmart. Save a note with specs and use cases.
	Walmart, Notes	Find a mechanical keyboard with WiFi wireless under $80. Compare two models and write a summary in Notes.
	Walmart, Notes	Look for a budget monitor for under $150 on Walmart. Research reviews and compare specs in Notes.
	Walmart, Notes	Find a student-friendly printer under $100. Summarize pros/cons and printing cost in Notes.
Travel Planning	Booking, Notes	Find a breakfast and Free-WiFi including hotel in Florida under ¥1750/night for 3 people. Summarize final choice in Notes.
	Tripadvisor, Notes	Plan a weekend foodie trip to Chicago. Select 3 top-rated food places from Tripadvisor and write your plan in Notes.
	Tripadvisor, Notes	Use Tripadvisor to plan a route to visit 4 museums in Washington, D.C. Include notes on entry fees and hours.
	Tripadvisor, Notes	Find 3 top-rated beautiful spots in Arizona using Tripadvisor. Summarize cost, facilities, and activities in Notes.
	Tripadvisor, Maps, Notes	Search for a hotel in Seoul close to cultural landmarks. Summarize the hotel info and nearby attractions in Notes.
	Booking, Maps, Notes	Find a top-rated hotel in Boston under ¥1840/night, and note nearby attractions in Notes.
	Booking, Maps, Notes	Book a hotel in Orlando for tonight under ¥800. Check on Maps if it's close to amusement parks. Summarize in Notes.
	Tripadvisor, Notes	Plan a 3-day itinerary to San Francisco. Use Tripadvisor to find places to visit, eat, and stay. Summarize in Notes.
	Booking, Notes	Find 2 top-rated hotels in Vail, Colorado. Write a comparison of the hotels in Notes.
	Tripadvisor, Notes	Find 3 trendy restaurants in Tokyo on Tripadvisor. Summarize their highlights in Notes.

Table 7: Examples of **complex** operation tasks proposed by **Mobile-Eval-RAG**, covering three categories: *Restaurant Recommendation*, *What's Trending* and *Travel Planning*. Each task specifies a real-world mobile scenario involving 2–3 apps and requires users to retrieve, analyze, and summarize content in Notes.

Task	Completion Items	Task	Completion Items
Research the latest green energy innovations from 2025 and summarize top 3 technologies in Notes.	• Opened Chrome • Searched for "2025 green energy innovations" • Identified at least 3 innovations • Checked technical descriptions or use cases • Compared key features or applications • Opened Notes • Created a new note • Wrote a summary of 3 innovations	Search for electric car models under $35k in Chrome. Note 3 options and main features in Notes.	• Opened Chrome • Searched for "electric cars under $35k" • Checked top 3 results • Compared specs of at least 3 models • Verified prices are under $35k • Opened Notes • Created a new note • Wrote key features of 3 models
Find 3 healthy breakfast recipes on YouTube that are under 10 mins to cook. Write the summary in Notes.	• Opened YouTube • Searched for "healthy breakfast under 10 mins" • Identified 3 suitable recipes • Verified recipe durations • Identified key ingredients • Opened Notes • Created a new note • Wrote a summary of 3 recipes	Find a beginner-friendly daily yoga video on YouTube. Note down the video title and channel name and write a routine summary in Notes.	• Opened YouTube • Searched for "daily yoga beginner video" • Selected one video • Verified it's beginner-friendly • Verified title, channel name and routine • Opened Notes • Created a new note • Summarized the routine structure
Find recent space discoveries from NASA or SpaceX. Summarize 2 major ones in Notes.	• Opened Chrome • Searched for "recent discoveries NASA / SpaceX" • Tap into a relevant article • Selected 2 major discoveries • Checked dates and agencies involved • Opened Notes • Created a new note • Summarized both discoveries	Find 2023–2024 news or articles about global plastic bans. Summarize 3 countries' policies in Notes.	• Opened Chrome • Searched for "plastic ban policies in 2023-2024" • Tap into a relevant article • Identified 3 country-specific sources • Noted banned items and start dates • Opened Notes • Created a new note • Summarized each policy
Find recommended books for beginners in machine learning. Summarize book titles and author recommendations in Notes.	• Opened Chrome • Searched for "beginner machine learning books" • Visited at least one curated list • Selected 3 recommended books • Checked author information • Opened Notes • Created a new note • Listed book titles and authors	Research next 3 major space missions. Write a timeline summary in Notes.	• Opened Chrome • Searched for "upcoming space missions 2025" • Tap into a relevant official article • Found missions from different agencies • Noted launch dates and goals • Opened Notes • Created a new note • Wrote mission timeline
Research top 5 internet safety tips for teenagers in 2025. Write a short guideline in Notes.	• Opened Chrome • Searched for "internet safety tips teenagers 2025" • Tap into a relevant article • Read multiple guides • Selected 5 actionable tips • Opened Notes • Created a new note • Listed 5 safety tips	Find articles or guides comparing coffee brewing methods. Summarize key differences and ideal use cases for each method in Notes.	• Opened Chrome • Searched for "coffee brewing methods comparison" • Found a detailed article • Found more detailed articles • Compared time, flavor, ease • Opened Notes • Created a new note • Wrote a method-by-method summary and compare differences and use cases

Table 8: Examples of **Completion Rate Evaluation Criteria** for the *Information Searching* task category in the Multi-Eval-RAG dataset. Each task is decomposed into a sequential list of atomic **Completion Items**, serving as step-wise indicators for objectively evaluating task Completion Rate (CR). The table is structured into pairs of columns: the **Task** column shows the natural language instruction, while the corresponding **Completion Items** column enumerates the necessary steps to fulfill the task. Tasks involving two applications (e.g., Chrome and Notes) contain 8 items, while those involving three applications contain 10 items, enabling more fine-grained progress tracking for long-horizon or multi-app tasks.

Task	Completion Items	Task	Completion Items
Search for 3 fun or useful mobile apps trending on X in 2025. Summarize features in Notes.	• Opened X • Searched for "trending mobile apps 2025" • Identified 3 different apps • Reviewed posts describing app features • Verified popularity through likes • Opened Notes • Created a new note • Wrote short feature summary for the 3	Search on X for discussions on 2025 metaverse chat tools. Summarize 3 tools in Notes.	• Opened X • Searched for "metaverse chat tools 2025" • Found 3 frequently mentioned tools • Read user discussions • Identified key features and opinions • Opened Notes • Created a new note • Wrote a short summary of the 3 tools
Look for trending tech startups in 2025. Use X and Chrome to summarize 3 promising ones.	• Opened Chrome • Searched for "trending tech startups 2025" • Identified 3 promising companies • Opened X • Searched each startup name • Reviewed user sentiment or press mentions • Opened Notes • Created a new note • Summarized startup strengths • Mentioned industries	Check recent posts about AI-generated music. Find 2 popular songs or tools and write a summary.	• Opened X • Searched for "AI-generated music in 2025" • Found 2 popular songs/tools • Reviewed likes/comments for verifying popularity • Opened Notes • Created a new note • Mentioned AI tool or artist • Summarized feedback
Find the top trending music video on YouTube. Analyze the comments and summarize what people like in Notes.	• Opened YouTube • Searched for "top trending music video" • Clicked top video • Reviewed likes/multiple user comments for checking trending • Identified common praise points • Opened Notes • Created a new note • Summarized viewer highlights	Research what types of games are trending in 2025. Use Chrome and summarize 3 trends in Notes.	• Opened Chrome • Searched for "game trends 2025" • Found 1–2 gaming articles • Identified 3 distinct trends • Reviewed game titles as examples • Opened Notes • Created a new note • Wrote short trend summary
Check what's trending for holidays in Tokyo. Search on X and summarize top places or events.	• Opened X • Searched "holidays in Tokyo" • Found relevant trending posts • Identified 2–3 locations/events • Checked photo/video content • Opened Notes • Created a new note • Summarized top places or events	Find 3 YouTube Shorts creators with viral content in 2025. Summarize what makes their content engaging.	• Opened YouTube • Searched for "YouTube Shorts creators with viral content" • Tap into Shorts • Identified 3 viral creators • Noted visual/editing style • Opened Notes • Created a new note • Summarized engaging features
Find 3 popular vloggers who post daily life or travel content. Summarize what makes their videos engaging.	• Opened YouTube • Searched for "top vloggers 2025 travel/-daily life" • Picked 3 with high views/subscribers • Identified common themes • Opened Notes • Created a new note • Summarized engagement factors • Mentioned creator names	Look for hype around 2024 memecoins on X. Note top 2 trending coins and community sentiment.	• Opened X • Searched for "2024 memecoins" • Identified 2 frequently mentioned coins • Reviewed comments for popular threads • Analyzed tone and hype level • Opened Notes • Created a new note • Wrote short summary for top 2 trending coins and community sentiment

Table 9: Examples of **Completion Rate Evaluation Criteria** for the *What's Trending* task category in the Multi-Eval-RAG dataset. Each task is decomposed into a sequential list of atomic **Completion Items**, serving as step-wise indicators for objectively evaluating task Completion Rate (CR). The table is structured into pairs of columns: the **Task** column shows the natural language instruction, while the corresponding **Completion Items** column enumerates the necessary steps to fulfill the task. Tasks involving two applications (e.g., Chrome and Notes) contain 8 items, while those involving three applications contain 10 items, enabling more fine-grained progress tracking for long-horizon or multi-app tasks.

Task	Completion Items	Task	Completion Items
Find the best ramen place in Chicago Loop with at least 500 reviews and rating over 4.5. Write a review summary in Notes.	• Opened Maps • Searched for "ramen restaurants in Chicago Loop" • Identified restaurants with $>$ 500 reviews • Identified restaurants with rating > 4.5 • Selected the best-rated candidate • Opened Notes • Created a new note • Wrote a review summary mentioning key strengths, rating and review count in summary	Search for nearby vegan breakfast spots. Pick one with best rating and write a short review in Notes.	• Opened Maps • Searched for "vegan breakfast near me" • Evaluated ratings and reviews • Verified vegan menu items • Selected the top-rated spot • Opened Notes • Created a new note • Wrote a short review
Look for a family-friendly restaurant in Urbana suitable for kids. Write a short summary in Notes.	• Opened Maps • Searched for "family-friendly restaurant Urbana" • Checked at least one restaurant's rating • Verified that restaurant offers kids menu • Selected one preferred restaurant • Opened Notes • Created a new note • Wrote a summary about the selected restaurant	Find a seafood restaurant suitable for a romantic dinner. Include menu highlight in Notes.	• Opened Maps • Searched for "seafood restaurant" • Checked ambiance-related reviews for suitability • Checked menu options • Identified one with good romantic setting • Opened Notes • Created a new note • Wrote menu highlights in Notes
Search for breakfast buffet places near me with good reviews. Compare 2 and write in Notes.	• Opened Maps • Searched for "breakfast buffet near me" • Identified a buffet place • Identified two buffet places with good reviews • Compared menu offerings and prices • Opened Notes • Created a new note • Wrote a comparison between the two places	Find a trending brunch place in Chicago in Maps. Check user posts on X and summarize in Notes.	• Opened Maps • Searched for "trending brunch place in Chicago" • Identified 1-2 candidate restaurants • Opened X • Searched restaurant names on X • Reviewed relevant user posts • Collected highlights or common opinions • Opened Notes • Created a new note • Wrote summary of user opinions
Find a hotpot restaurant near a university campus. Write the address and the average user rating into Notes.	• Opened Maps • Searched for "hotpot restaurant near university campus" • Verify at least one university on the map • Checked restaurant ratings • Selected one hotpot restaurant • Opened Notes • Created a new note • Wrote address and rating into note	Find 3 burger restaurants within 5km. Write a comparison of reviews and prices in Notes.	• Opened Maps • Searched for "burger restaurants" • Applied distance filter • Selected 3 distinct restaurants • Reviewed user comments and prices • Opened Notes • Created a new note • Wrote review and price comparison
Find a Chinese restaurant in Chicago with rating over 4.5 that offers takeout. Save 3 dishes and their prices in Notes.	• Opened Maps • Searched for "Chinese restaurant in Chicago" • Filtered for 4.5 star • Filtered for takeout • Checked menu • Opened Notes • Created a new note • Wrote a summary explaining why it's liked	Search for a dessert shop open after 10pm. Check user reviews and note recommended items.	• Opened Maps • Searched for "dessert shop open after 10pm" • Verified business hours • Checked user reviews • Identified recommended items • Opened Notes • Created a new note • Wrote summary of user recommendations

Table 10: Examples of **Completion Rate Evaluation Criteria** for the *Restaurant Recommendation* task category in the Multi-Eval-RAG dataset. Each task is decomposed into a sequential list of atomic **Completion Items**, serving as step-wise indicators for objectively evaluating task Completion Rate (CR). The table is structured into pairs of columns: the **Task** column shows the natural language instruction, while the corresponding **Completion Items** column enumerates the necessary steps to fulfill the task. Tasks involving two applications (e.g., Chrome and Notes) contain 8 items, while those involving three applications contain 10 items, enabling more fine-grained progress tracking for long-horizon or multi-app tasks.

Task	Completion Items	Task	Completion Items
Find a tablet under $150 on Walmart. Compare 2 brands and summarize specs in Notes.	• Opened Walmart • Searched for "tablet under $150" • Found at least two brand options • Compared screen size and storage • Reviewed technical specs • Opened Notes • Created a new note • Wrote brand comparison	Find an ergonomic chair under $120 on Walmart. Watch YouTube reviews and write pros/cons in Notes.	• Opened Walmart • Searched for "ergonomic chair under $120" • Chose 1–2 chairs • Open YouTube • Searched for selected chairs • Reviewed videos comment on YouTube • Identified comfort and features • Opened Notes • Created a new note • Summarized strengths and weaknesses
Search for a portable speaker under $100. Watch 1–2 YouTube reviews and write pros/cons in Notes.	• Opened Walmart • Searched for "portable speaker under $100" • Selected 1–2 speaker models • Opened YouTube • Searched for selected speakers • Reviewed videos comment on YouTube • Identified pros and cons • Opened Notes • Created a new note • Wrote pros and cons summary	Find a blender under $50 with good reviews on Walmart. Save a note with specs and use cases.	• Opened Walmart • Searched for "blender under $50" • Filtered by high ratings • Read reviews • Verified blender features • Opened Notes • Created a new note • Wrote down key specs and use cases
Look for an affordable desk lamp for study with eye-care mode. Compare ratings & product descriptions and note in Notes.	• Opened Walmart • Searched for "eye-care desk lamp for study" • Found multiple lamp options • Checked product descriptions • Reviewed customer ratings • Opened Notes • Created a new note • Wrote lamp comparison	Find a mechanical keyboard with WiFi wireless under $80. Compare two models and write a summary in Notes.	• Opened Walmart • Searched for "mechanical keyboard under $80" • Filter Wifi wireless • Choose 2 models • Compared features • Opened Notes • Created a new note • Summarized model comparison
Find top-rated pet supplies under $30 for dogs on Walmart. Compare options, prices and descriptions in Notes.	• Opened Walmart • Searched for "dog supplies under $30" • Filtered by rating • Selected multiple items • Checked item descriptions • Opened Notes • Created a new note • Compare top 2–3 products	Look for a budget monitor for under $150 on Walmart. Research reviews and compare specs in Notes.	• Opened Walmart • Searched for "budget monitor under $150" • Found multiple options • Compared screen resolution and size • Checked product details • Opened Notes • Created a new note • Summarized and compared specs
Find a student laptop under $400. Check YouTube reviews and summarize 3 models in Notes.	• Opened Walmart • Searched for "laptop under $400" • Identified 3 potential models • Compared specs (RAM, CPU, storage) • Opened YouTube • Searched for selected laptops • Reviewed videos comment on YouTube • Opened Notes • Created a new note • Wrote summary of 3 laptops	Find a student-friendly printer under $100. Summarize pros/cons and printing cost in Notes.	• Opened Walmart • Searched for "student-friendly printer under $100" • Read a product page • Checked prices, printing speed and type • Reviewed comments for pros/cons • Opened Notes • Created a new note • Summary the selected printers' pros/-cons and cost

Table 11: Examples of **Completion Rate Evaluation Criteria** for the *Online Shopping* task category in the Multi-Eval-RAG dataset. Each task is decomposed into a sequential list of atomic **Completion Items**, serving as step-wise indicators for objectively evaluating task Completion Rate (CR). The table is structured into pairs of columns: the **Task** column shows the natural language instruction, while the corresponding **Completion Items** column enumerates the necessary steps to fulfill the task. Tasks involving two applications (e.g., Chrome and Notes) contain 8 items, while those involving three applications contain 10 items, enabling more fine-grained progress tracking for long-horizon or multi-app tasks.

Task	Completion Items	Task	Completion Items
Find a breakfast and Free-WiFi including hotel in Florida under ¥1750/night for 3 people. Summarize final choice in Notes.	• Opened Booking • Filter for 3 people • Searched for "Florida" • Filter for breakfast and Free-WiFi • Reviewed at least 2 options under the prices • Opened Notes • Created a new note • Wrote hotel summary	Find a top-rated hotel in Boston under ¥1840/night, and note nearby attractions in Notes.	• Opened Booking • Searched for "Boston" • Applied top-rated filters • Selected best matching hotel • Open Maps • Searched for selected hotel • Checked nearby attractions • Opened Notes • Created a new note • Summarized hotel and nearby spots
Plan a weekend foodie trip to Chicago. Select 3 top-rated food places from Tripadvisor and write your plan in Notes.	• Opened Tripadvisor • Searched for "top-rated food places in Chicago" • Applied filters for cuisine or rating • Selected 3 restaurants • Reviewed descriptions and ratings • Opened Notes • Created a new note • Wrote a foodie trip plan	Book a hotel in Orlando for tonight under ¥800. Check on Maps if it's close to amusement parks. Summarize in Notes.	• Opened Booking • Searched for "Orlando" • Selected a hotel under ¥800 • Open Maps • Searched for the selected hotel • Checked distance to amusement parks • Opened Notes • Created a new note • Wrote booking summary • Listed name, price and the distances to amusement parks
Use Tripadvisor to plan a route to visit 4 museums in Washington, D.C. Include notes on entry fees and hours.	• Opened Tripadvisor • Searched for "museums in Washington D.C." • Selected 4 nearby museums • Checked entry fees and hours • Planned a route • Opened Notes • Created a new note • Wrote museum route plan and • Listed fees and open times	Plan a 3-day itinerary to San Francisco. Use Tripadvisor to find places to visit, eat, and stay. Summarize in Notes.	• Opened Tripadvisor • Searched for "San Francisco" • Found places to visit for 3 days • Looked for dining and hotel options • Opened Notes • Created a new note • Wrote 3-day plan • Listed sites, restaurants, and hotels
Find 3 top-rated beautiful spots in Arizona using Tripadvisor. Summarize cost, facilities, and activities in Notes.	• Opened Tripadvisor • Searched for "Arizona" • Selected 3 top-rate spots • Reviewed costs and facilities • Checked available activities • Opened Notes • Created a new note • Summarized info for each spots	Find 2 top-rated hotels in Vail, Colorado. Write a comparison of the hotels in Notes.	• Opened Booking • Searched for "Colorado" • Filtered for "Vail" • Filtered for top-rated • Selected 2 relevant options • Opened Notes • Created a new note • Listed features for each
Search for a hotel in Seoul close to cultural landmarks. Summarize the hotel info and nearby attractions in Notes.	• Opened Tripadvisor • Searched "Seoul hotels near cultural landmarks" • Found relevant listings • Open Maps • Searched selected hotels • Verified landmarks near the hotels • Opened Notes • Created a new note • Summarized hotel info • Listed nearby cultural spots	Find 3 trendy restaurants in Tokyo on Tripadvisor. Summarize their highlights in Notes.	• Opened Tripadvisor • Searched "trendy restaurants Tokyo" • Selected 3 distinct options • Read user comments • Noted dish highlights or decor themes • Opened Notes • Created a new note • Summarized 3 restaurants

Table 12: Examples of **Completion Rate Evaluation Criteria** for the *Travel Planning* task category in the Multi-Eval-RAG dataset. Each task is decomposed into a sequential list of atomic **Completion Items**, serving as step-wise indicators for objectively evaluating task Completion Rate (CR). The table is structured into pairs of columns: the **Task** column shows the natural language instruction, while the corresponding **Completion Items** column enumerates the necessary steps to fulfill the task. Tasks involving two applications (e.g., Chrome and Notes) contain 8 items, while those involving three applications contain 10 items, enabling more fine-grained progress tracking for long-horizon or multi-app tasks.

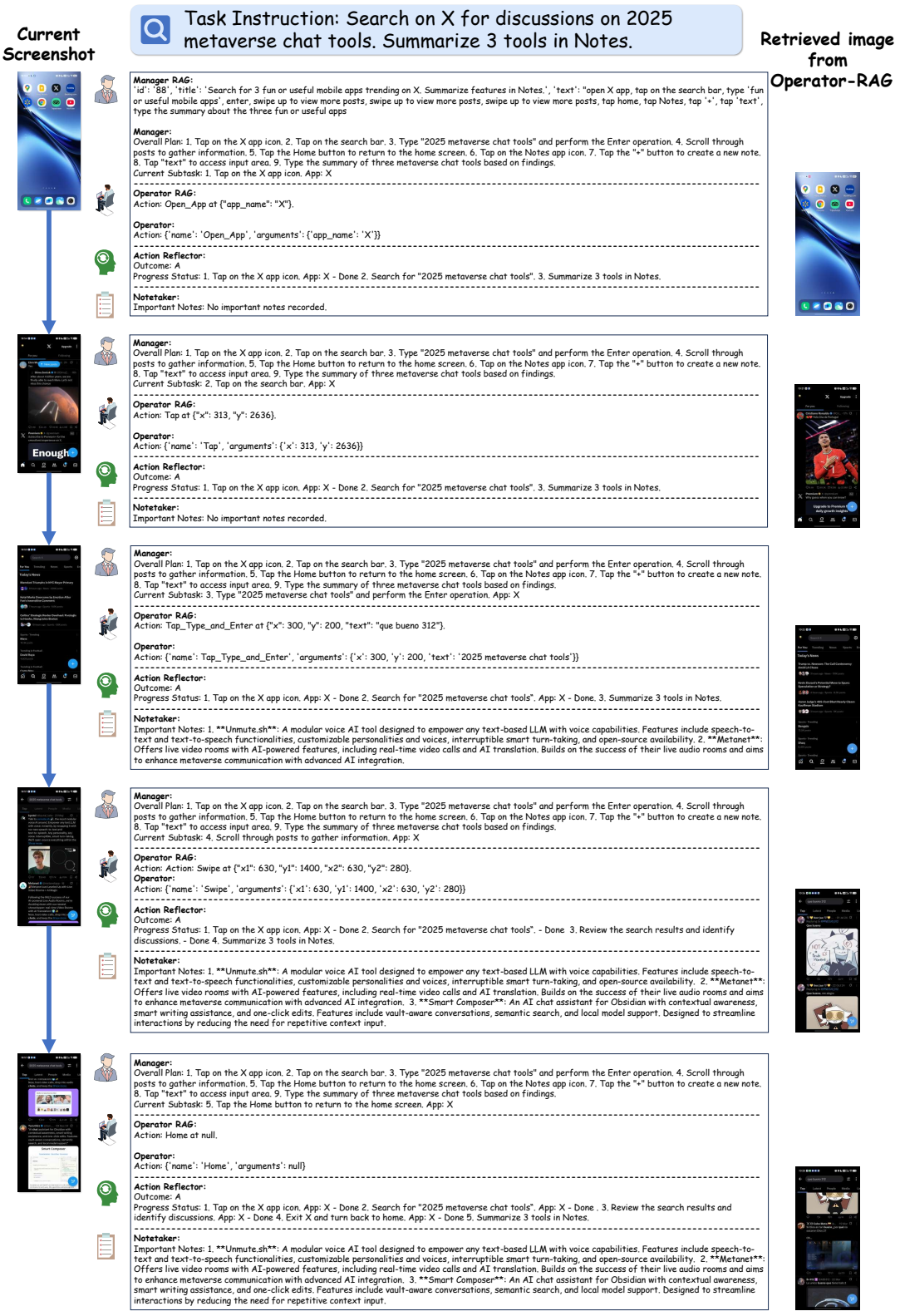


Figure 1: An end-to-end execution case of a task. Each box represents an iterative loop, progressing chronologically from top to bottom. The "**Current Screenshot**" column on the left shows the mobile device's real-time UI screen state. The central section details the inter-agent communication, where a "**Manager-RAG**" retrieves the most relevant document (we set k=1 here) to provide high-level guidance. The "**Manager**" section presents the overall plan and the current subtask. The "**Operator-RAG**" section shows the best-matched action retrieved from its knowledge base for the current subtask, while the "Retrieved image from Operator-RAG" on the right provides reference screenshot. The "**Operator**" then synthesizes this information to generate a precise action with its argument. After that, The "**Action Reflector**" records the outcome and updates the progress status. Finally, the "**Notetaker**" records important information as "Important Notes" for future context. This process clearly depicts how the agents, through a "**Planning → Execution → Reflection**" loop, retrieval augmentation, step-by-step execution, and continuous feedback, progressively complete complex, cross-app tasks. (Process continued with next figure)

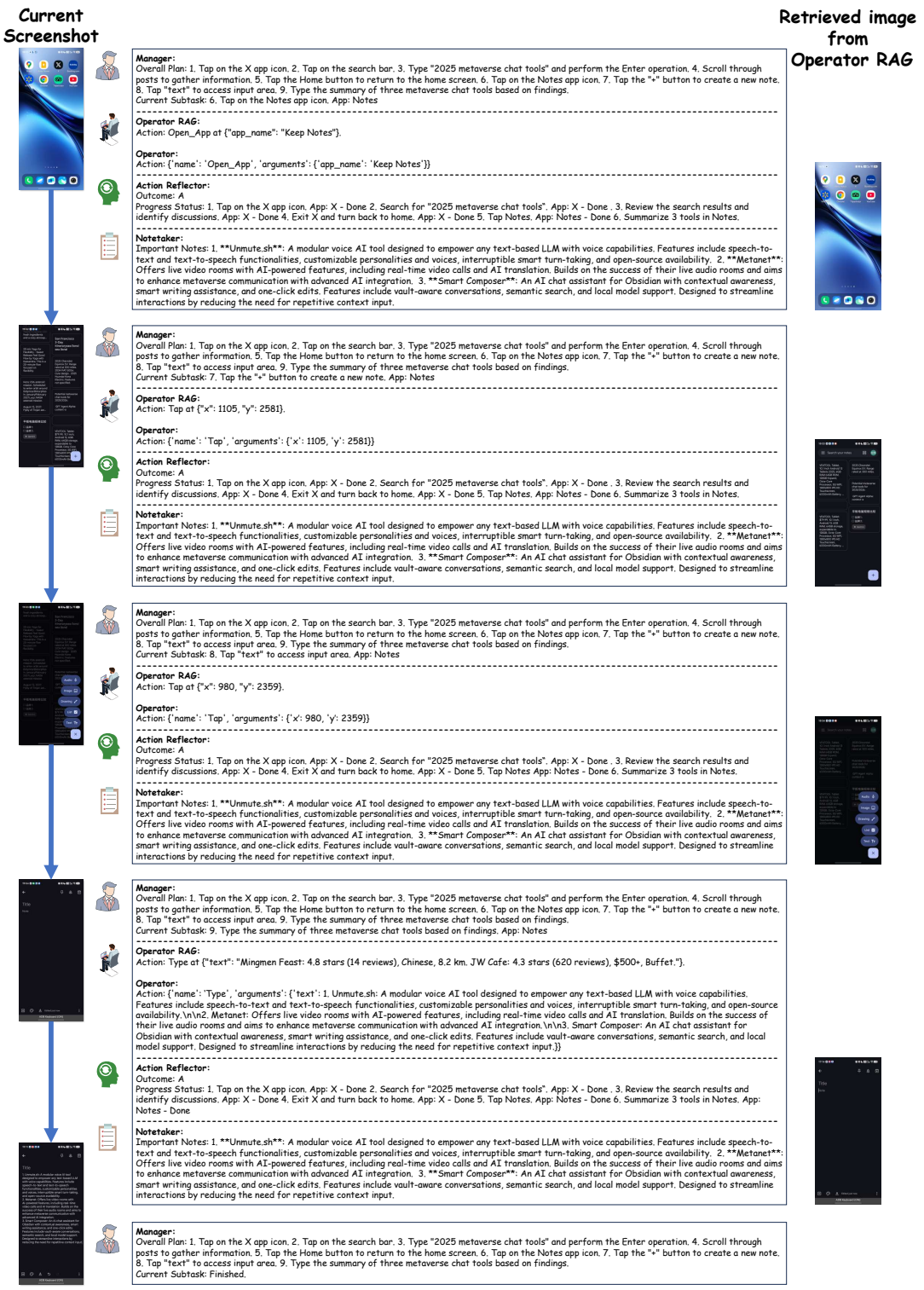


Figure 2: (Process continued from previous figure) An end-to-end execution case of a task. Each box represents an iterative loop, progressing chronologically from top to bottom. The "**Current Screenshot**" column on the left shows the mobile device's real-time UI screen state. The central section details the inter-agent communication. The "**Manager**" section presents the overall plan and the current subtask. The "**Operator-RAG**" section shows the best-matched action retrieved from its knowledge base for the current subtask, while the "Retrieved image from Operator-RAG" on the right provides reference screenshot. The "**Operator**" then synthesizes this information to generate a precise action with its argument. After that, The "**Action Reflector**" records the outcome and updates the progress status. Finally, the "**Notetaker**" records important information as "Important Notes" for future context. This process clearly depicts how the agents, through a "**Planning → Execution → Reflection**" loop, retrieval augmentation, step-by-step execution, and continuous feedback, progressively complete complex, cross-app tasks.

Figure 3: Qualitative comparison on a challenging cross-app task. Without retrieval augmentation, `Mobile-Agent-E` misidentifies visually similar buttons, enters local exploration, and requires a greater number of atomic steps and replans, which impacts task execution efficiency. Augmented by **Manager-RAG** and **Operator-RAG**, `Mobile-Agent-RAG` completes the same task with precise and decisive actions, demonstrating higher action accuracy, plan coherence, and overall efficiency.