CRMArena: Understanding the Capacity of LLM Agents to Perform Professional CRM Tasks in Realistic Environments

Kung-Hsiang Huang Akshara Prabhakar Sidharth Dhawan Yixin Mao
Huan Wang Silvio Savarese Caiming Xiong Philippe Laban Chien-Sheng Wu
Salesforce AI Research
{kh.huang, akshara.prabhakar, sidharth, y.mao,
huan.wang, ssavarese, cxiong, wu.jason}@salesforce.com

Abstract

Customer Relationship Management (CRM) systems are vital for modern enterprises, providing a foundation for managing customer interactions and data. Integrating AI agents into CRM systems can automate routine processes and enhance personalized service. However, deploying and evaluating these agents is challenging due to the lack of realistic benchmarks that reflect the complexity of real-world CRM tasks. To address this issue, we introduce **CRMArena**, a novel benchmark designed to evaluate AI agents on realistic tasks grounded on professional work environments. We worked with CRM experts to design nine customer service tasks distributed across three personas: service agent, analyst, and manager. We synthesize a large-scale simulated organization, populating 16 commonly-used industrial objects (e.g., account, order, knowledge article, case) with high interconnectivity, and uploading it into a real Salesforce CRM organization. UI and API access to the CRM is provided to systems that attempt to complete the tasks in CRMArena. Experimental results reveal that state-of-the-art LLM agents succeed in less than 58% of the tasks with ReAct prompting, and less than 65% even when provided manually-crafted function-calling tools. Our findings highlight the need for enhanced agent capabilities in function-calling and rule-following to be deployed in real-world work environment. CRMArena is an open challenge to the community: systems that can reliably complete tasks showcase direct business value in a popular work environment.[1]

1 Introduction

Customer Relationship Management (CRM) systems are pivotal in modern enterprises, serving as the backbone for managing interactions with current and potential customers (Winer, 2001; Payne and Frow, 2005). The integration of intelligent agents based on large language models (LLMs) into CRM systems promises to automate routine tasks, enhance operational efficiency, and revolutionize customer experiences. However, evaluating LLM agents in real-world professional environments remains a challenge, due to the absence of robust benchmarks that faithfully capture the complexity of tasks encountered in real-world CRM environments, largely due to data privacy concerns within enterprises.

Prior benchmarks on evaluating LLM agents on work-related tasks, such as WorkArena (Drouin et al., 2024), Workbench (Styles et al., 2024), and Tau (Yao et al., 2024) tend to focus on basic functionality, and fall short in two key areas. First, the complexity of the objects (e.g., tables in databases) and dependencies (e.g., foreign keys) between these objects is often overly simple, lacking the complexity of real-world scenarios. Second, the tasks included in the benchmarks, such as navigating web pages and filtering lists, are typically too straightforward and do not represent real-world work tasks.

To address these limitations, we introduce **CRMArena**, a comprehensive benchmark tailored to evaluate LLM agents on performing realistic CRM tasks in real-world work environments. CRMArena features a realistic sandbox environment modeled after Salesforce's schema, developed using an extensible data generation pipeline powered by LLMs (top left of Figure 1). Specifically, the pipeline tackles two key challenges: (1) *Object connectivity*: reflecting the complex relationships between data objects (e.g., ACCOUNT associated with CASE and ORDER) by mirroring Salesforce's Service Cloud schema[2]. (2) Introducing *latent variables* to better simulate realistic data dynamics, such as influenc-

[1] Our code and benchmark have been released at https://github.com/SalesforceAIResearch/CRMArena.

[2] https://architect.salesforce.com/diagrams/data-models/service-cloud/service-cloud-overview

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 3830–3850

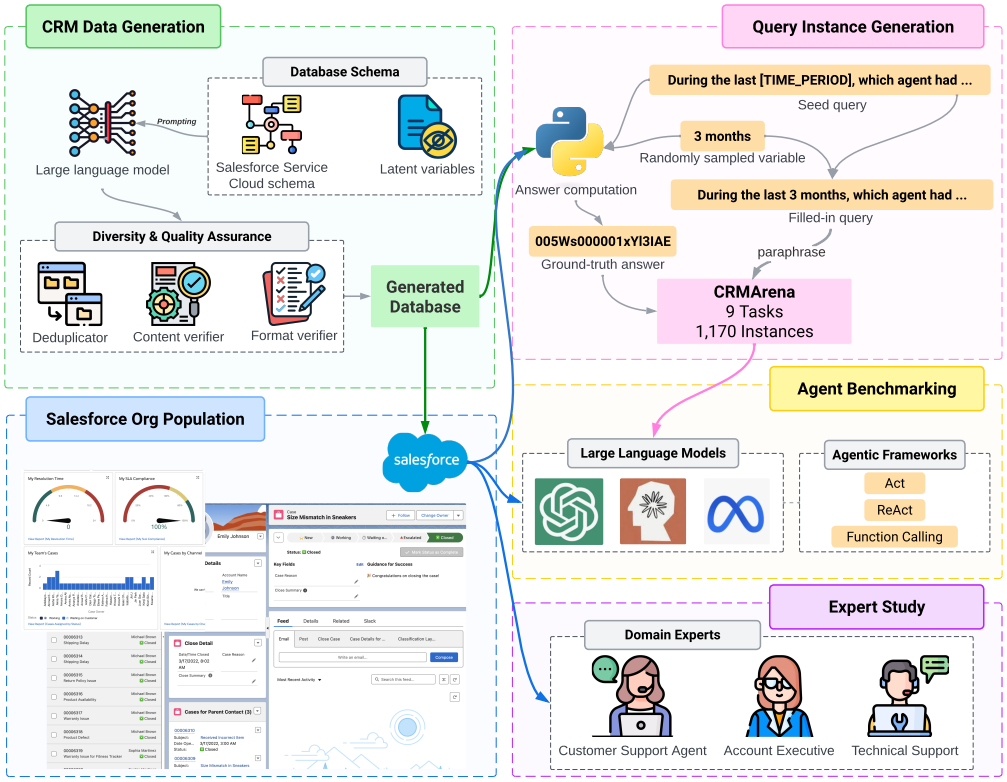


Figure 1: An overview of the contribution of this work. We begin by generating realistic CRM data based on the Salesforce Service Cloud schema, ensuring both quality and diversity through rigorous verification processes. This verified data is then stored locally and uploaded to a Salesforce organization (Org). An expert study, conducted with domain experts, validated the data's realism. Using this Org as a sandbox environment, we create query instances and benchmark various LLMs across different agentic frameworks.

ing case-filing behavior and modeling deviations from company guidelines.

Moreover, CRMArena defines tasks based on actual customer service personas. By consulting CRM experts experienced with Salesforce, we identified nine tasks representative of CRM use cases (§2.1). These tasks span three personas: Service Manager, Service Agent, and Service Analyst. For example, Service Managers focus on agent performance and strategic resource allocation. Table 1 compares CRMArena with previous datasets.

CRMArena seamlessly integrates with Salesforce,[3] enabling interaction via both the user interface and API access (see bottom of Figure 1). This integration facilitated an expert study with CRM professionals to assess the quality of our synthesized organization (§2.5). Study findings revealed that **90% of domain experts found the test environment to be *Realistic* or better**, underscoring the benchmark's fidelity to real-world CRM scenarios. Upon verifying the realism of CRMArena, we then assess various agentic systems through API access. We develop two sets of tools *general-purpose* vs. *task-specific* tools, combine them with three agentic frameworks and various LLMs. Findings indicate that **all LLM agents struggle to reliably complete tasks when using general-purpose tools**, with top performing systems completing less than 40% of the tasks. Incorporating manually designed tools can enhance performance, with top LLM agents solving up to 55% of the tasks. However, we discover that **weaker LLMs often do not benefit from manually-crafted tools due to their weaker function calling capabilities**.

In summary, our main contributions are:

- Introducing CRMArena, a realistic CRM agent benchmark with tasks validated by domain experts to evaluate LLM agents in real-world business scenarios.
- Developing a data generation strategy anchored in a real-world CRM schema, incorporating latent variables, deduplication, and rigorous data validation to ensure diversity and quality.
- Demonstrating through experiments that even state-of-the-art LLM agents do not reliably

[3] https://www.salesforce.com/crm/

Datasets	# Objects	# Dependencies/ Object	Real-world Environment	Realistic Work Tasks	Expert Validation
WorkBench (Styles et al., 2024)	5	0	✗	✗	✗
Tau-Bench (Yao et al., 2024)	3	0.67	✗	✗	✗
WorkArena (Drouin et al., 2024)	7	0.86	✓	✗	✗
CRMArena (Ours)	**16**	**1.31**	✓	✓	✓

Table 1: **A comparison between our benchmark with prior datasets.** CRMArena is the most complex benchmark given the highest number of objects and object dependencies involved. Furthermore, CRMArena is the only expert-validated benchmark that encompasses both a real-world environment and realistic work tasks.

complete CRMArena tasks, emphasizing the benchmark's value and challenges.

2 CRMArena

Motivated by tasks commonly addressed by CRM personas: service manager, service agent, and service analyst, CRMArena comprises nine tasks that reflect real-world CRM scenarios. Verified by domain experts as common occurrences in CRM, an overview of these tasks is presented in Figure 2. Below, we provide detailed illustrations of each task.

2.1 Tasks

The tasks in CRMArena are designed to accurately reflect the variety of challenges encountered in real-world CRM environments. They span the responsibilities of three key personas: the Service Manager, who focuses on strategic resource allocation; the Service Agent, who addresses customer inquiries; and the Service Analyst, who analyzes data trends and performance metrics to improve service operations.

New Case Routing (NCR) The goal of this task is to assign the best human agent to an incoming case, aiming to optimize various performance metrics. The input consists of a case subject and description, and the output is the ID of the recommended human agent. This task assesses LLM agent's ability to match cases to the most suitable human agent based on case histories and the skills and availability of these agents.

Handle Time Understanding (HTU) This task involves identifying the agent with the shortest/longest average handle time. Given the case history data, the objective is to determine the human agent who handled the previous cases the fastest/slowest.

Transfer Count Understanding (TCU) In this task, the LLM agent must find out which human agent transferred cases to others the least/most given a period of case history. Both HTU and TCU evaluate LLM agent's capacity to analyze performance based on predefined metrics accurately.

Name Entity Disambiguation (NED) The LLM agent must disambiguate named entities related to customer transactions. Here, we focus on disambiguating product names. Given the query shown in Figure 2, the agent needs to identify the specific order corresponding to running shoes bought by the mentioned customer within the given time frame. This tests the understanding of product names and customer order histories.

Policy Violation Identification (PVI) In this task, the LLM agent is given a case with interaction between a customer and an agent and must determine if any company policies have been breached. This involves analyzing the case details and comparing them against policy rules outlined in knowledge articles to identify violations.

Knowledge Question Answering (KQA) The goal here is for the LLM agent to answer a specific question based on knowledge articles. This evaluates the agent's capacity to look for accurate and relevant information from the CRM knowledge repository.

Top Issue Identification (TII) This task requires the LLM agent to identify the most reported issue for a particular product. Given case history, the agent must determine which issue has the highest frequency. This tests the ability to analyze issue reports for trend analysis.

Monthly Trend Analysis (MTA) The LLM agent must determine which months have the highest number of cases for a given product and a given time period. By analyzing the case history in a given period of time, the LLM agent identifies the month with the most cases, demonstrating its ability to recognize trends and patterns over time.

Best Region Identification (BRI) In this task, the LLM agent's objective is to identify the regions where cases are closed the fastest. The agent must analyze case closure times across various regions and indicate which regions perform best.

Figure 2: An overview of the nine distinct tasks introduced in CRMArena.

2.2 Sandbox Environment

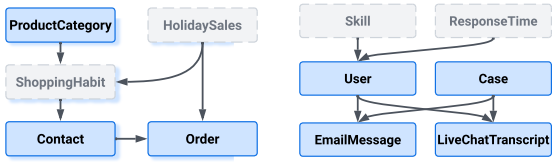
Creating a sandbox environment for CRMArena poses unique challenges, particularly related to privacy concerns and the need for realistic data without using real enterprise data. To this end, we develop a scalable data generation pipeline capable of producing diverse and realistic CRM data across various domains. In our setup, we model 16 business objects. The complete list of objects and their descriptions can be found in Appendix D. There are two major challenges for building such a pipeline: object connectivity and hidden casual relationship. In the following subsections, we illustrate how we address these challenges.

Object Connectivity Real-world company data is characterized by complex interconnections between objects like CASE and ACCOUNT. Our data generation approach, based on Salesforce's Service Cloud schema, ensures high connectivity. For instance, the CASE object is connected to objects like ACCOUNT, CONTACT, and USER. Figure 7 displays these interdependencies, creating meaningful data environments. Table 1 highlights our benchmark's much higher object connectivity compared to existing work.

Hidden Causal Relationship Replicating the implicit causal relationships found in real-world data presents a significant challenge. To address this, we introduce latent variables that simulate various underlying factors, creating data that mirrors the subtle dependencies and patterns in authentic CRM databases. These latent variables are crucial for facilitating certain tasks and generating desired scenarios. As shown in the example in Figure 3, the SHOPPINGHABIT variable allows us to more realistically simulate a customer's purchasing patterns based on time periods or holiday seasons. Similarly, the SKILL latent variable for the USER (Agent) object enables our simulations of EMAILMESSAGE and LIVECHATTRANSCRIPT to include situations where an agent is unable to resolve an issue and must transfer it to another agent. Without this latent variable, we would lack scenarios essential for our Transfer Count Understanding task. The full data generation flow is shown in Figure 8.

Quality and Diversity Assurance We generate data in JSON format, with each JSON object representing one entry of an object, to ensure higher controllability (Huang et al., 2024; Laban et al., 2024). Due to the large volume of objects (e.g., 500 PRODUCT entries paired with 40+ PRICEBOOK entries resulting in over 20,000 PRICEBOOKENTRY items) and the limited maximum output tokens of LLMs, directly prompting LLMs to generate all entries of an object is infeasible. To address this, we employ mini-batch prompting with a batch size of 10. However, this approach can lead to duplicated or highly similar content across batches. To mitigate this issue, we implement a two-phase deduplication strategy. First, for all objects, we include all previously generated entries in the prompt during mini-batch prompting and instruct the LLM not to generate the same content. After data generation, we use string exact matching to remove duplicate entries for fields and objects crucial to certain tasks (e.g., the email of USER).

Additionally, we subject the data to a rigorous quality assurance process involving a dual-layer verification. The *format verifier* ensures all data entries conform to predefined schemas by checking whether each entry in the generated mini-batch contains all required fields for the object. Mini-batches that fail this verification are discarded and regenerated. The *content verifier* checks for feasibility for



(a) (b)

Figure 3: Examples of latent variables (gray) influencing object (blue) generation. (a) The SHOPPINGHABIT variable affects when and what type of products a customer buys. (b) The SKILL variable determines if an agent can handle a case or needs to transfer it.

tasks, focusing on objects crucial for specific tasks. For example, in the Named Entity Disambiguation task, we verify that the paraphrased ambiguous product name (1) does not deviate too much from its original name and (2) is not too similar to other products the customer has purchased. In this scenario, the content verifier provides an LLM with a list of products the customer has purchased and the paraphrased product name. If the LLM correctly identifies the product, we retain the entry; otherwise, it is discarded.[4]

Upload to Org Once the data is generated, we populate it into a clean Simple Demo Org (SDO)[5] on Salesforce without latent variables. This exclusion serves two purposes: it mirrors the typical scenario where companies do not have access to such information, thus providing a more realistic testing environment, and it adds an extra layer of challenge compared to testing on the generated databases. Moreover, utilizing Salesforce's SDO as the sandbox eliminates the necessity and complexity for local environment setup, which is required in many related work (Styles et al., 2024; Drouin et al., 2024; Yao et al., 2024; Zhou et al., 2024). This approach not only facilitates testing but also encourages scientific rigor and future research on our benchmark. The details of the sandbox environment can be found in Appendix D.

Environment Specification The input to our data generation pipeline are company name, company description, database schema, and the scale of the objects (e.g., number of cases and products). We choose to create an Org for a fictional shoe company due to the diverse product range and customer service scenarios typical in the footwear industry. The scale of our generated data is designed to reflect a mid-sized enterprise, with thousands of orders and hundreds of products and support cases spanning a 4-year period. The total number of entries per object is shown in Appendix D.

Extensibility Our data generation pipeline is designed for flexibility and can be easily adapted to other industries through changes in user-specified input parameters. For instance, by specifying the industry in the company description, our pipeline can automatically generate realistic CRM data tailored to that specific industry, such as finance. Furthermore, to accommodate other use cases beyond customer service, such as sales, users would only need to provide the corresponding schema (e.g., Salesforce Sales Cloud schema for sales). This flexibility ensures that the pipeline can be extended to meet a wide range of business needs and LLM agent benchmarking purposes.

Note that our current setup reflects a simplified version of CRM scenarios, where each CASE is linked to both an ISSUE and a PRODUCT. This simplification helps manage the complexity of tasks like Top Issue Identification, which would otherwise require LLM agents to individually analyze every case, making the tasks too infeasible for the current state of LLMs. Our benchmark can be adjusted to create more complex settings by removing such dependencies as LLM capabilities advance.

2.3 Query Instance Generation

Following the creation of the sandbox environment, we generate natural language query instances and their ground-truth answers to benchmark our tasks. For the Knowledge QA tasks, queries can be naively constructed by prompting an LLM each knowledge article to generate question answer pairs (Laban et al., 2022; Huang et al., 2024). For the remaining tasks, we construct query instance through a four-step process: (1) seed query construction, (2) ground-truth computation, (3) ID mapping, and (4) query paraphrasing.

We manually create 14 seed queries in total with placeholders for corresponding variables, such as time period or product name. This facilitates the development of functions that compute the ground truth answers on the generated database by leveraging the latent variables that are only visible there. For example, an agent's policy violation during a live chat is verifiable only within the generated database. Upon obtaining the answers, we map the IDs in the generated database to their counterparts in Salesforce Org, thereby establishing the ground truths for our queries in the sandbox environment.

[4] We utilize `gpt-4o` as the LLM for data synthesis and content verification for its cost efficiency.

[5] https://partners.salesforce.com/s/education/general/Salesforce_Orgs

Finally, to ensure diversity in the test queries, we employ an LLM to paraphrase the seed queries, enhancing the robustness and variety of our benchmarking process. An example of this process is shown in the top right of Figure 1.

Additionally, to simulate real-world scenarios where some questions may be *unanswerable*, we construct non-answerable queries. Inspired by the non-answerable question schema outlined in (Brahman et al., 2024), we focus on *False Presuppositions* queries, which are most relevant in CRM settings. For example, a query may request the identification of an agent who transfers the most cases during a given period, despite no agents transferring cases in that period. We include non-answerable queries in five tasks: Transfer Time Understanding, Handle Time Understanding, Top Issue Identification, Named Entity Disambiguation, and Policy Violation Identification. For these instances, we expect models to produce "None" as outputs. In summary, non-answerable queries account for 30% of the total queries per corresponding task. Overall, we produce 130 query instances per task, totaling 1,170 queries for `CRMArena`. Details and seed queries are provided in Appendix B.

2.4 Tools: APIs and Functions

Salesforce Orgs naturally support a variety of *general-purpose* APIs, such as the Apex API, REST API, and Tooling API, which are designed to cover a broad set of functionalities within the Salesforce ecosystem. For the scope of our tasks and their integration with a Python environment, we choose to utilize SOQL and SOSL queries[6]. SOQL queries are intended for obtaining a specific subset of objects using exact matches or filtering criteria, typically formatted as `"SELECT Id ..."`, while SOSL queries enable fuzzy searching in objects like knowledge articles and product names, formatted as `"FIND ..."`. These two types of queries can theoretically support a wide range of query instances, eliminating the necessity to manually design actions for function calls.

In addition to general-purpose APIs, we also develop *task-specific* tools in the form of Python wrapper functions to facilitate the evaluation of function-calling agents. These functions optimize task performance by providing structured and logical operations directly mapped to typical CRM tasks. We manually define 27 such Python wrapper functions on top of SOQL and SOSL (complete list in Appendix C) to streamline function calls and target the key components needed for each task. These task-specific functions are designed to maximize reusability across various tasks, minimizing the need for task-specific customizations.

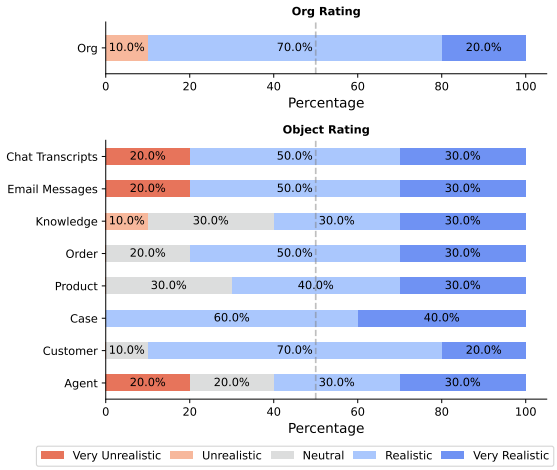


Figure 4: Expert study results. The plots illustrate domain experts' realism ratings for the overall Org structure (top) and individual objects we generated (bottom).

2.5 Expert Study

To ensure the realism and practicality of the sandbox environment we developed, we conducted a user study involving ten experts with diverse professional backgrounds who have experience working on Salesforce Orgs daily. These experts were recruited via the User Interviews platform[7]. Details of the expert study can be found in Appendix F.

Each session of the expert study was structured into three parts. First, we provided the experts with an overview of our sandbox, highlighting key objects such as CASE and CONTACT, and allowing them access through relevant URLs. This initial orientation was designed to familiarize them with the organization. Second, we assigned them five query instances sampled from `CRMArena`, each representing a different task, to complete. This task completion phase was aimed at evaluating the practical application and operational coherence of the sandbox in executing real-world CRM tasks. Finally, the experts rated the realism of our Org environment compared to the real-world systems they are accustomed to. They also provided detailed rationales for their ratings, giving insights into how our environment aligns with actual CRM scenarios.

The results of our expert study are presented

[6] https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_sosl_intro.htm

[7] https://www.userinterviews.com/

in Figure 4. The findings are highly encouraging: **90% of the experts rated our populated Org as either *Realistic* or *Very Realistic*.** This positive assessment extended to the individual objects within the Org, with more than 77% of experts giving them similarly high ratings for realism. These results strongly suggest that our sandbox environment closely mirrors real-world CRM systems. We provide the qualitative feedback and rationale from the experts we interviewed in Table 14.

3 Benchmarking Experiments

3.1 Experimental Settings

Models We evaluate state-of-the-art proprietary and open-source LLMs, including `gpt` models (`gpt-4o` and `gpt-3.5-turbo`); `claude` models (`claude-3.5-sonnet` and `claude-3-sonnet`), and the `llama` models (`llama-3.1-405b` and `llama-3.1-70B` (Dubey et al., 2024)).[8] Additionally, we tested inference-time scaling models for enhanced reasoning capabilities, including `o1` and `deepseek-r1` (Guo et al., 2025). With these models, we tested three common agentic frameworks: Act, ReAct (Yao et al., 2023), and Function Calling (FC). ReAct is a prompt-based method, with each step characterized by a *thought* and *action* process, while Act is ReAct without the *thought* component. The details of these settings are described in the following paragraphs and Appendix G.

Action Space Every task can be formulated as a Partially Observable Markov Decision Process (POMDP) $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$ with instruction space $\mathcal{U}$, state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, and reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$. In the Act and ReAct settings, the action space is rich, i.e. $\mathcal{A}$ = {`execute <query>`, `submit <result>`}. Given a user query $u \in \mathcal{U}$ in natural language, an agent can `execute <query>` to issue a SOQL or SOSL query to interact with the instance to receive the observation $o_t \in \mathcal{O}$ of executing the query in the environment. This continues until the agent chooses to `submit` and receives a binary reward $r = \mathcal{R}(s_T, \texttt{submit}) \in \{0, 1\}$. In the Function Calling setting, the agent interacts with the environment via API tools implemented as Python functions. In this case the agent is not directly exposed to the Salesforce environment and the object dependencies are kept hidden. Internally the APIs interact with the environment in a controlled manner defined by us. An action a is of the form `tool_call{**kwargs}`. The system prompts for these three setups are described in Appendix E.

Observation Space Actions are executed on the sandbox environment through the Simple Salesforce package[9]. If an action succeeds, the environment will return the queried data in the CRM system; otherwise, an error message, such as incorrect function calling parameters, is returned.

Evaluation Metrics For the knowledge QA task, since it is an open-ended text generation task, we use F1 scores. For the remaining tasks, we only need to compare the predicted and ground truth object IDs; therefore, an exact match is used.

3.2 Results

The main results are summarized in Table 2. We made the following observations. First, **real-world CRM tasks remain challenging for top LLM agents**. Using the ReAct framework, the best model (`o1`) only achieves an overall score of 57.7%. Even when equipped with human-crafted functions, the overall performance is still only 64.3%. These findings highlight the challenges of our `CRMArena`. Second, **stronger and weaker LLMs show opposite trend on different agentic frameworks**. In particular, models like `gpt-4o` and `claude-3.5-sonnet` score higher in the FC setting, while their weaker counterparts performs worse when equipped with function calling capabilities. This indicate that human-defined functions may not always help LLM agents, as weaker models may not be able to properly utilize the functions, resulting in lower performance. An intriguing exception is `deepseek-r1`. Though `deepseek-r1` is recognized as a strong reasoning model, its tool-calling capabilities seem lacking, primarily due to its (1) inadequate adherence to user instructions and (2) poor ability to adjust previous responses based on external feedback. **function calling might be unnecessary with a sufficiently strong reasoning model**, as evidenced by `o1` in the ReAct setting outperforming all other models in the FC setting. Nevertheless, integrating human-crafted functions can still offer performance benefits to strong reasoning models like `o1`. Finally, **open-source models**

[8] We observed the native function-calling mode to perform poorly and hence report the prompt mode performance for Llama 3.1 models.

[9] https://github.com/simple-salesforce/simple-salesforce

Model	NCR	HTU	TCU	NED	PVI	KQA	TII	MTA	BRI	ALL
					Act					
gpt-4o	43.1	10.0	17.7	30.8	28.5	29.3	68.5	29.2	7.7	29.4
gpt-4o-mini	0.8	38.5	23.8	9.2	0.0	43.1	26.9	3.8	3.8	16.7
claude-3.5-sonnet	78.5	24.6	15.4	51.5	28.5	44.7	45.4	20.8	26.9	37.4
claude-3-sonnet	9.2	26.9	24.6	30.8	23.8	16.6	16.2	1.5	0.0	16.6
llama3.1-405b	46.2	17.7	17.7	13.9	30.0	47.0	15.4	5.4	6.9	22.2
llama3.1-70b	28.5	20.0	24.6	6.2	30.0	47.9	8.5	0.0	1.5	18.6
llama3.1-8b	0.0	3.1	0.0	6.2	4.6	4.5	2.3	0.0	1.5	2.5
					ReAct					
gpt-4o	70.0	39.2	22.3	30.8	35.4	50.2	64.6	20.9	10.8	38.2
gpt-4o-mini	40.8	36.9	25.4	31.5	24.6	52.8	30.0	6.2	6.2	28.3
claude-3.5-sonnet	62.9	20.0	11.5	52.3	30.0	45.0	43.9	20.8	21.5	34.3
claude-3-sonnet	7.7	24.6	26.9	29.2	28.5	16.0	22.3	0.8	0.0	17.3
llama3.1-405b	81.5	22.3	15.4	33.9	34.6	55.3	34.6	13.9	13.1	33.8
llama3.1-70b	48.5	20.0	13.9	33.1	37.7	48.7	23.9	13.9	10.8	27.8
llama3.1-8b	0.0	0.0	1.5	6.2	15.4	4.0	0.0	0.0	0.8	3.1
o1	70.0	51.5	54.6	34.6	30.0	58.8	81.5	75.4	63.1	57.7
deepseek-r1	53.8	23.1	30.1	40.8	34.6	61.2	46.9	3.1	22.3	35.1
					Function Calling					
gpt-4o	60.0	47.7	81.5	46.2	39.2	30.4	97.7	27.7	59.2	54.4
gpt-4o-mini	0.8	10.8	10.8	17.7	13.8	39.7	60.0	0.0	21.5	19.5
claude-3.5-sonnet	4.6	33.1	82.3	52.3	30.0	40.5	69.2	26.9	36.9	41.8
claude-3-sonnet	0.8	1.5	30.0	25.4	41.5	23.2	12.3	1.5	0.0	15.1
llama3.1-405b (prompt)	16.2	31.5	64.6	50.0	26.9	47.6	95.4	86.9	42.3	51.3
llama3.1-70b (prompt)	1.5	23.1	44.6	53.8	37.4	42.4	93.8	43.8	29.2	41.1
llama3.1-8b (prompt)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
o1 (prompt)	60.8	68.5	66.9	60.0	24.6	39.2	99.2	84.6	74.8	64.3
deepseek-r1 (prompt)	0.8	0.8	2.3	0.8	24.6	34.6	0.0	13.8	3.1	9.0

Table 2: Overall performance (%) of various LLMs under different agentic frameworks on CRMArena. The evaluation metric is F1 score for the knowledge question answering (KQA) task and exact match for all other tasks. ALL represents the average performance across all tasks.

	Model	# Completion Tokens	# Turns	Cost ($)
ReAct	gpt-4o	48,568.73	5.4	0.182
	claude-3.5-sonnet	70,814.75	6.9	0.228
	llama3.1-405b	35,647.29	7.3	0.125
FC	gpt-4o	78,305.38	6.8	0.305
	claude-3.5-sonnet	105,248.43	8.1	0.371

Table 3: The cost of top-performing agents averaged across queries and tasks.

are catching up the proprietary LLMs. Across three settings, we see the llama models score similar, and sometimes higher, than the gpt and claude models. This indicate a closing gap between the open and closed-source models. From Figure 6, we observe how llama models tend to show higher scope for error recovery based on execution feedback than the closed-source models.

3.3 Discussions

What is the most cost-effective solution? Excluding the two reasoning models, in two-third of the agentic frameworks, gpt-4o performs the best. The efficiency of gpt-4o is also reflected in Table 3, which shows that gpt-4o has the lowest cost per instance and requires the least number of turns to complete a query. Therefore, the most cost-effective solution is using gpt-4o under the function calling setting.

How does the type of function affect model performance? In Table 2, we observe that equipping LLM agents with function calling capabilities does not necessary results in increased performance. To better understand this phenomenon, we categorizes the functions based on two dimensions: functionality and functional dependency. Functionality refers to whether the function solely queries the CRM system via SOSL or SOQL (QUERY) or if it includes additional operations such as mathematical calculations or aggregations (CALCULATION). Functional dependency, on the other hand, classifies functions into those that rely on the outputs of other functions (DEPENDENT)and those that are independent (INDEPENDENT). This is crucial because our benchmark requires LLM agents to perform a sequence of calls, with each call dependent on the output of the previous ones (Qin et al., 2023; Lu et al., 2024). Table 15 shows the list of functions and tasks we tested.

We sampled four function-task pairs from each category to evaluate the performance of gpt-4o, gpt-4o-mini, and claude-3-sonnet when specific functions were removed from the toolset, sub-

Functionality	Dependency	gpt-4o	gpt-4o-mini	claude-3-sonnet
QUERY	INDEPENDENT	-6.6	-6.9	2.3
QUERY	DEPENDENT	-2.9	3.0	7.5
CALCULATION	INDEPENDENT	-9.4	4.6	-3.3
CALCULATION	DEPENDENT	-26.7	4.0	3.3

Table 4: Performance difference (%) when removing each category of functions. A lower number indicates more useful functions to the LLM agents.

stituting two generic functions, `execute_soql` and `execute_sosl`, to execute arbitrary queries. The findings, summarized in Table 4, indicate that while all function types enhance `gpt-4o`'s performance, they do not have the same effect on `gpt-4o-mini` or `claude-3-sonnet`. This suggests that stronger models are better at utilizing human-crafted functions effectively, whereas weaker models might struggle. Interestingly, CALCULATION functions, hypothesized to benefit LLMs weak in mathematical operations, may actually decrease performance in weaker models due to their limited function calling capabilities.

How consistent are the agents across multiple trials? Consistency is important for LLM agents, especially when deployed in work environments. We evaluate the consistency of LLM agents through multiple trials of prompting. Here, we adapt the pass^k metric proposed by Yao et al. (2024). pass^k computes the probability that all k independent and identically distributed task attempts are successful, averaged over all tasks. We run ten trials across all tasks in `CRMArena` except for KQA, as the reward for KQA is not binary. The results are shown in Figure 5, we found that, surprisingly, pass^k for all three agentic frameworks we tested drop at the nearly same rate as k increases. This indicates that the consistency for these three frameworks are similar and that the top-performing LLM cannot reliably solve the tasks with any of the three agentic frameworks we evaluated.

4 Related Work

Agent Benchmark Several benchmarks have been developed to evaluate LLM-based agents (Yao et al., 2022; Liu et al., 2024; Jimenez et al., 2024). Recently, major efforts have focused specifically on web agents, which challenges LLMs to navigate and perform actions on websites. These websites are often about everyday scenarios, such as e-commerce, and social discussion form (Deng et al., 2023; He et al., 2024; Zhou et al., 2024; Lù et al., 2024; Yoran et al., 2024). Another line of work focus on evaluating the safety of deploying agents

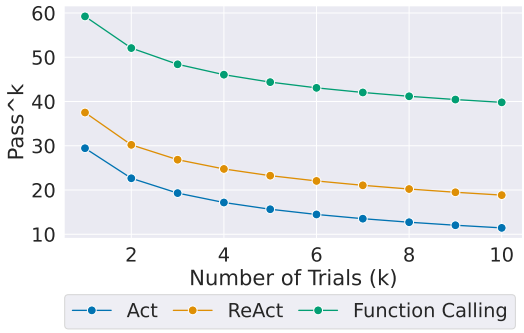


Figure 5: The consistency of `gpt-4o` using different agentic frameworks.

(Ruan et al., 2024; Yuan et al., 2024; Yin et al., 2024; Qiu et al., 2025).

Work-oriented Datasets A few studies have developed datasets specifically for work-oriented tasks. The CRM Benchmark (Salesforce, 2024) aims to assess LLMs' text generation and summarization abilities in business applications. WorkBench (Styles et al., 2024) consists of five databases designed to evaluate LLM agents' performance in simple work tasks, such as sending emails, creating calendar invites, and counting traffic sources for a website. τ-Bench (Yao et al., 2024) creates tasks that require interactions with users to obtain relevant information and authorization, achieved by using LLMs to simulate users. WorkArena (Drouin et al., 2024) builds a web-based work environment that allows for testing agents with visual capabilities.

5 Conclusion

This work introduces `CRMArena`, a novel benchmark for evaluating LLM agents in performing realistic CRM tasks within professional work environments. By incorporating expert-validated tasks and modeling intricate data interconnections typical of CRM systems, `CRMArena` offers a comprehensive and realistic challenge for LLM agents. Our experiments demonstrate that even state-of-the-art LLMs struggle with these realistic tasks, achieving limited success rates even with function-calling capabilities. These findings highlight the gap between current LLM capabilities and the requirements of real-world CRM scenarios. `CRMArena` serves as a foundational step towards more sophisticated evaluations of LLM agents in realistic work environments.

6 Ethical Considerations

This work introduces a benchmark for evaluating LLM agents within the context of CRM systems. While the data used is synthetically generated, it is modeled after real-world CRM data structures and tasks. Thus, it is important to consider the ethical implications of this work, particularly regarding data biases and privacy concerns.

Data Bias Although synthetic, the data is generated by models trained on real-world data, which may contain inherent biases. These biases, related to customer demographics, purchase behavior, or case resolution, could be inadvertently reflected in the generated data, potentially perpetuating stereotypes or discriminatory practices. Thankfully, after conducting a thorough manual inspection of the generated data to identify potential biases, we did not observe such patterns.

Privacy Concerns While our benchmark does not use any real customer data and therefore does not have access to personal information, the structure and nature of CRM data itself can raise privacy concerns. The tasks in our benchmark involve accessing sensitive customer information, albeit synthetic. To ensure responsible handling of this data, even though synthetic, we performed a thorough manual inspection to verify the absence of any personally identifiable information and to confirm that the data cannot be used to infer private information about individuals. This meticulous review process reinforces our commitment to ethical data practices and mitigates potential privacy risks.

7 Limitations

The CRMArena comprises nine tasks that thoroughly assess the ability of LLM agents to perform duties typically associated with three primary roles within a realistic environment: Service Manager, Service Agent, and Service Analyst. Nonetheless, this study does not encompass other common personas in CRM, such as sales representatives. We aim to incorporate these additional roles in our future studies.

References

Faeze Brahman, Sachin Kumar, Vidhisha Balachandran, Pradeep Dasigi, Valentina Pyatkin, Abhilasha Ravichander, Sarah Wiegreffe, Nouha Dziri, Khyathi Chandu, Jack Hessel, et al. 2024. The art of saying no: Contextual noncompliance in language models. *arXiv preprint arXiv:2407.12043*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? In *Forty-first International Conference on Machine Learning*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Kung-Hsiang Huang, Philippe Laban, Alexander Fabbri, Prafulla Kumar Choubey, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2024. Embrace divergence for richer insights: A multi-document summarization benchmark and a case study on summarizing diverse information from news articles. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 570–593, Mexico City, Mexico. Association for Computational Linguistics.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Philippe Laban, Alexander R Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context llms and rag systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Philippe Laban, Chien-Sheng Wu, Lidiya Murakhovs' ka, Xiang'Anthony' Chen, and Caiming Xiong. 2022. Discord questions: A computational approach to diversity analysis in news coverage. *arXiv preprint arXiv:2211.05007*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2024. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. *Preprint*, arXiv:2408.04682.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

Adrian Payne and Pennie Frow. 2005. A strategic framework for customer relationship management. *Journal of marketing*, 69(4):167–176.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Haoyi Qiu, Alexander R. Fabbri, Divyansh Agarwal, Kung-Hsiang Huang, Sarah Tan, Nanyun Peng, and Chien-Sheng Wu. 2025. Evaluating cultural and social awareness of llm web agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of lm agents with an lm-emulated sandbox. In *The Twelfth International Conference on Learning Representations*.

Salesforce. 2024. Salesforce announces the world's first llm benchmark for crm.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. 2024. Workbench: a benchmark dataset for agents in a realistic workplace setting. In *First Conference on Language Modeling*.

Russell S Winer. 2001. A framework for customer relationship management. *California management review*, 43(4):89–105.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2024. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. Tau-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Da Yin, Haoyi Qiu, Kung-Hsiang Huang, Kai-Wei Chang, and Nanyun Peng. 2024. Safeworld: Geo-diverse safety alignment. In *Thirty-eighth Conference on Neural Information Processing Systems*.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. Assistantbench: Can web agents solve realistic and time-consuming tasks? *Preprint*, arXiv:2407.15711.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. 2024. R-judge: Benchmarking safety risk awareness for llm agents. *arXiv preprint arXiv:2401.10019*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

Model	HTU	TCU	NED	TII	PVI
			Act		
gpt-4o	15.4	48.7	94.9	87.2	92.3
gpt-4o-mini	94.9	79.5	30.8	79.5	74.4
claude-3.5-sonnet	25.6	28.2	82.1	33.3	84.6
claude-3-sonnet	84.6	79.5	100.0	51.3	74.4
llama3.1-405b	56.4	51.3	46.2	38.5	0.0
llama3.1-70b	46.2	76.9	20.5	20.5	100.0
			ReAct		
gpt-4o	64.1	48.7	100.0	84.6	74.4
gpt-4o-mini	97.4	82.1	97.4	61.5	71.8
claude-3.5-sonnet	12.8	7.7	87.2	30.8	82.1
claude-3-sonnet	79.5	84.6	94.9	69.2	94.9
llama3.1-405b	53.8	38.5	97.4	41.0	64.1
llama3.1-70b	64.1	41.0	97.4	17.9	17.9
			Function Calling		
gpt-4o	59.0	84.6	74.4	100.0	35.9
gpt-4o-mini	15.4	7.7	0.0	0.0	0.0
claude-3.5-sonnet	52.6	74.4	100.0	100.0	100.0
claude-3-sonnet	2.6	15.4	59.0	38.5	56.4

Table 5: Performance (%) of various LLMs under different agentic frameworks on CRMArena for the non-answerable queries.

A Further Discussions

Reward vs number of turns In Figure 6, we show the distribution of the number of turns it takes for agents to successfully complete an user query.

Non-answerable query analysis In Table 5, we present the performance of each LLM agents. Overall, LLM agents are good at handling such queries, compared to standard queries. Interestingly, a trend shown in Table 2 is observed in this experiment as well: function calling only benefit stronger LLMs, while weaker LLMs like claude-3-sonnet and gpt-4o performs worse when equipped with function calling capabilities.

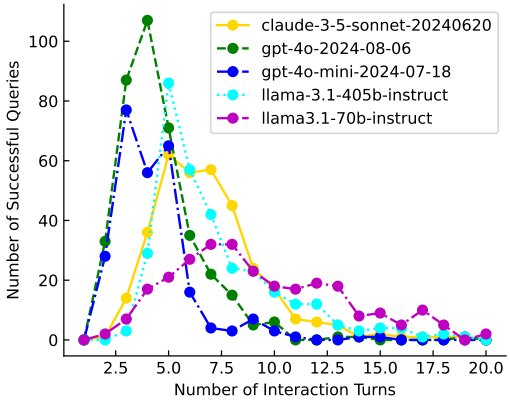


Figure 6: Distribution of the number of turns it takes for agents to reach the goal under *ReAct*.

B Query Generation Details

Table 6 show the complete list of seed queries used in our experiments. More examples of how the final queries are constructed can be found in Table 9.

C Action Space Details

In the text-based agent settings (i.e. ReAct and Act), the actions include (1) executing SOSL queries, (2) executing SOQL queries, and (3) submitting the answer. In the function-calling settings, the actions are a list of carefully designed functions, shown in Table 7.

D Sandbox Environment Details

We show the objects and dependencies in Figure 7. These objects, except for Knowledge__kav are densely connected, reflecting the complexity of real-world work environment. The total number of entry per objects is shown in Table 8. Our data generation flow is shown in Figure 8.

D.1 Object Details

Below, we describe the details of each object.

- **ProductCategory**: Represents the category that products are organized in.

1. In [YEAR] [MONTH/QUARTER/SEASON], identify the agent who managed more than [NCASES] cases and had the [EXTREMA] handle time.
2. In the past [TIMEPERIOD], find the agent with the [EXTREMA] handle time among those who managed more than [NCASES] cases.
3. During the last [TIMEPERIOD], which agent had the [EXTREMA] average handle time for those handling over [NCASES] cases?
4. In [YEAR] [MONTH/QUARTER/SEASON], identify the agent who managed more than [NCASES] cases and had the [EXTREMA] transfer counts.
5. In the past [TIMEPERIOD], find the agent with the [EXTREMA] transfer counts among those who managed more than [NCASES] cases.
6. During the last [TIMEPERIOD], which agent had the [EXTREMA] average transfer counts for those handling over [NCASES] cases?
7. Which knowledge article did the agent violate policy?
8. Today is [TODAY]. Is there any month in which the cases we received for [PRODUCT] is much more than other months over the past [TIMEPERIOD]?
9. Today is [TODAY]. For [PRODUCT], what is the most common issue in the last [TIMEPERIOD].
10. Today is [TODAY]. In [YEAR] [MONTH/QUARTER/SEASON], what is the most common issue for [PRODUCT].
11. Today is [TODAY]. In which states do we close cases the fastest in the last [TIMEPERIOD]?
12. Today is [TODAY]. In [YEAR] [MONTH/QUARTER/SEASON], which states do we close cases the fastest.
13. What is the best agent to assign to for this case?
14. Today is [TODAY]. Show me the [PRODUCT] that I ordered [PERIOD] ago.

Table 6: The full set of seed queries used for query generation.

Functions	Description
get_agents_with_max_cases(subset_cases)	Returns a list of agent IDs with the maximum number of cases from the given subset of cases.
get_agents_with_min_cases(subset_cases)	Returns a list of agent IDs with the minimum number of cases from the given subset of cases.
calculate_average_handle_time(cases)	Calculates the average handle time for each agent based on a list of cases.
get_start_date(end_date, period, interval_count)	Calculates the start date based on the end date, period, and interval count.
get_period(period_name, year)	Calculates the start and end date based on the period name and year.
get_agent_handled_cases_by_period(start_date, end_date)	Retrieves the number of cases handled by each agent within a specified time period.
get_qualified_agent_ids_by_case_count(agent_handled_cases, n_cases)	Filters agent IDs based on the number of cases they have handled.
get_cases(start_date, end_date, agent_ids, case_ids, order_item_ids, issue_ids, statuses)	Retrieves cases based on various filtering criteria.
get_non_transferred_case_ids(start_date, end_date)	Retrieves the IDs of cases that were not transferred between agents within a specified date range.
get_agent_transferred_cases_by_period(start_date, end_date, qualified_agent_ids)	Retrieves the number of cases transferred between agents within a specified date range.
get_shipping_state(cases)	Adds shipping state information to the given cases.
calculate_region_average_closure_times(cases)	Calculates the average closure times for cases grouped by region (shipping state).
get_order_item_ids_by_product(product_id)	Retrieves the order item IDs associated with a given product.
get_issue_counts(start_date, end_date, order_item_ids)	Retrieves the issue counts for a product within a given time period.
find_id_with_max_value(values_by_id)	Identifies the ID with the maximum value from a dictionary.
find_id_with_min_value(values_by_id)	Identifies the ID with the minimum value from a dictionary.
get_account_id_by_contact_id(contact_id)	Retrieves the Account ID associated with a given Contact ID.
get_purchase_history(account_id, purchase_date, related_product_ids)	Retrieves the purchase history for a specific account, date, and set of products.
get_month_to_case_count(cases)	Counts the number of cases for each month from a list of cases.
search_knowledge_articles(search_term)	Searches for knowledge articles based on a given search term.
search_products(search_term)	Searches for products based on a given search term.
get_issues()	Retrieves a list of issue records.
get_email_messages_by_case_id(case_id)	Retrieves the email exchanges for a given case.
get_livechat_transcript_by_case_id(case_id)	Retrieves the live chat transcript for a given case.
submit(content)	Returns the response content.

Table 7: The complete list of functions for the function calling settings.

Object	Number of Entries
USER	100
CONTACT	196
PRODUCTCATEGORY	12
PRODUCT	500
ORDERITEM	71,00
PRICEBOOK	44
PRICEBOOKENTRY	22,000
CASE	977
ORDER	2,071
EMAILMESSAGE	3,234
LIVECHATTRANSCRIPT	387
KNOWLEDGE	45

Table 8: The number of entries per object.

- **Product2**: Represents a product that your company sells.
- **ProductCategoryProduct**: Holds the relation between product and product category to assign products to a category.
- **Pricebook2**: Represents a price book that contains the list of products.
- **Pricebook Entry**: Represents a product entry (an association between a Pricebook2 and Product2) in a price book.
- **Order**: Represents an order associated with a contract or an account.
- **Order Item**: Represents an order product that the company sells.
- **Knowledge**: Documentation or information articles that are accessible to users or customers.
- **Contact**: Refers to an individual or party related to an account.
- **Issue**: Represents a type of problem raised by a customer.
- **Account**: An entity, company, or individual your company does business with. In B2C setting, an account represents a customer.
- **User (agent)**: System user, often representing customer support agents.
- **Case**: A record that describes a customer inquiry or issue.
- **CaseHistory**: A log of the changes and updates made to a case over time.
- **EmailMessage**: Email communication related to cases or customer inquiries between an agent and a customer.
- **LiveChatTranscript**: A conversation from a live chat session between an agent and a customer.

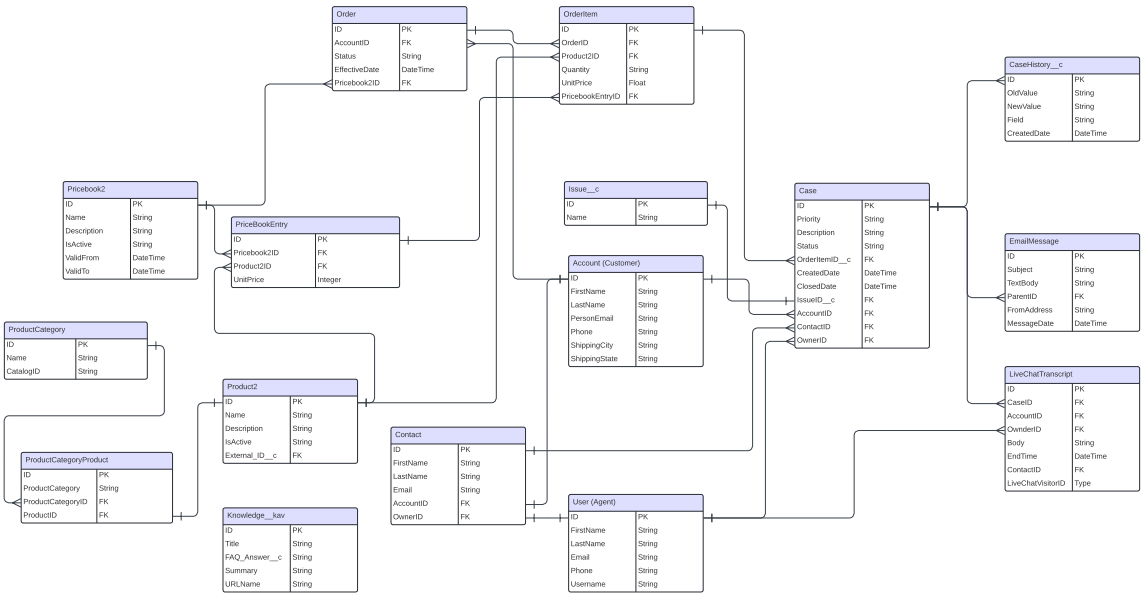


Figure 7: The objects and their dependencies in our sandbox environment.

Handle Time Understand **Seed query**: In [YEAR] [MONTH/QUARTER/SEASON], identify the agent who managed more than [NCASES] cases and had the [EXTREMA] handle time. **Filled-in query**: In 2021 February, identify the agent who managed more than 2 cases and had the highest handle time. **Paraphrased query**: In February 2021, determine the agent with the longest handle time who managed more than 2 cases.
Top Issue Identification **Seed query**: In [YEAR] [MONTH/QUARTER/SEASON], what is the most common issue for [PRODUCT]? **Filled-in Query**: In 2023 Q2, what is the most common issue for Flex Yoga Mat? **Paraphrased query**: What was the most frequent issue with Flex Yoga Mat in the second quarter of 2021?

Table 9: Examples of the query generation process.

E Prompts

In this section, we display the prompts used in our experiments. Table 10, Table 11, Table 12 show the system prompt for the Act, ReAct, and Function Calling settings, respectively.

F Expert Study Details

As detailed in Table 13, we recruited a diverse range of domain experts for our study. The participants varied in age, gender, and professional backgrounds.

F.1 Recruitment Criteria

Using the User Interviews platform, we set the job filter such that the participants of our survey must have a job title of one of the following:

- Account Manager
- Technical Support Engineer
- Support Engineer
- Technical Support Specialist
- Technical Support Manager
- Technical Support Technician

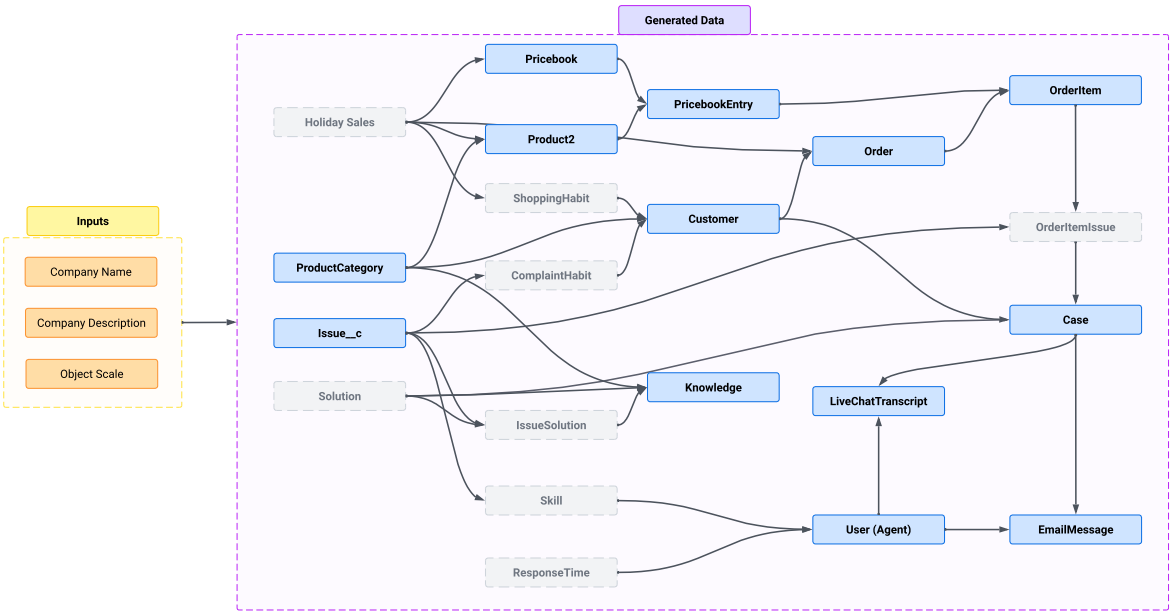


Figure 8: Data generation overview. The generation of each object is conditioned on the previously generated objects with arrows pointing to it. Blue boxes represent standard object, while gray boxes denote latent variables that are not uploaded to the Salesforce Org.

- Technical Support Agent
- Technical Support Expert
- Account Manager/Agent
- Account Manager/Analyst
- Customer Service Advisor/Customer Service Associate
- Customer Service Associate
- Customer Service Representative

In addition, we have created a screener survey. The most important question in the survey is "How often do you use Salesforce CRM?". The valid candidate must select the option "Several times a day" to be able to participate in our study.

F.2 The study

We use Google Form to conduct expert studies due to its ease to use. The study is broken down into three parts:

- **Part 1**: Familiarizing the Org [5 minutes]. This is for a broad overview of some of the objects in this Org.
- **Part 2**: Task Completion [45 minutes]. At this stage, they are be given tasks regarding customer service. They should try to accomplish as many as possible within the 45-minute time frame.
- **Part 3**: Quality Rating [10 minutes]. Based on their experience with the first two parts of this study, rate the quality of the Org and objects.

Below, we illustrate how each part is executed.

Part 1 In this part, we provide interviewee the log in credentials to our created Org (sandbox environment). Once they log in, they are instructed to spend 5 minutes to read the objects in the Org that are relevant to the tasks they will be completing later. The instructions and interface for this part are shown in Figure 9.

Part 2 After familiarizing with our created Org, participants are then asked to complete the tasks. They are required to complete 5 query instances from `CRMArena`. An example of the query is shown in Figure 10.

Part 3 Upon completing the first two parts of the expert study, in the final stage, participants are asked to rate the realism of our Orgs and data. In addition to providing ratings, they also need to provide rationales for their ratings. An example question is shown in Figure 11.

Below, we provide the rating and descriptions for participants to choose from.

You are an expert in Salesforce and you have access to a Salesforce instance.
Instructions
- You will be provided a question, the system description, and relevant task context.
- Interact with the Salesforce instance to build Salesforce Object Query Language (SOQL) or Salesforce Object Search Language (SOSL) queries as appropriate, to help you answer the question.
- Salesforce Object Search Language (SOSL) can be used to construct text-based search queries against the search index.
- Your generation should always be an Action command and NOTHING ELSE. Generate only one Action command.
- DO NOT generate ANY system observation, you will receive this based on your Action command.
- If no record is found matching the requirements mentioned, just return 'None'.

Here is a description of how to use these commands:
Action
- Can be 'execute' or 'submit'.
- execute, to execute SOQL/SOSL that will return the observation from running the query on the Salesforce instance.
- submit, to return the final answer of the task to the user.
- Format: <execute> a valid SOQL/SOSL query </execute> or <submit> response to user </submit>

Guidelines
- Execute SOQL/SOSL queries to understand the Salesforce instance that will help you find the answer to the question.
- When you are confident about the answer, submit it.
- Always end with a submit action containing ONLY the answer, NO full sentences or any explanation.

Example 1
Question: What is the total number of opportunities?
Output:
<execute> SELECT COUNT() FROM Opportunity </execute>
(If the observation from the Salesforce instance 100, your next step can be)
<submit> 100 </submit> NOT <submit> The total number of opportunities is 100 </submit>
Example 2

[... Hide details for space...]

Salesforce instance description
The objects available in the Salesforce instance are:
User, Account, Contact, Case, Knowledge__kav, ProductCategory, Product2, ...
The fields available for the objects along with their descriptions and dependencies are:
User
- FirstName: First name of the agent
- LastName: Last name of the agent
- Email: Email address of the agent

[... Hide details for space...]

Additional task context
Handle/Transfer Times Policies
[... Hide details for space...]

Table 10: The system prompt used in the Act setting.

Org ratings:

- Very Unrealistic: The organization structure and setup felt highly artificial, with no resemblance to typical Salesforce setups.
- Unrealistic: The organization had some familiar elements, but significant parts were not convincingly structured.
- Neutral: The organization felt somewhat realistic, with a mix of plausible and implausible elements.
- Realistic: The organization largely mirrored a real-world Salesforce setup, with minor inconsistencies.
- Very Realistic: The organization felt entirely authentic, closely resembling a real-world Salesforce configuration.

Object ratings:

- I don't know/I'm not familiar with the object.
- Very Unrealistic: The objects seemed fundamentally flawed or invented with little regard for typical Salesforce objects.

You are an expert in Salesforce and you have access to a Salesforce instance.
Instructions
- You will be provided a question, the system description, and relevant task context.
- Think step by step and interact with the Salesforce instance to build Salesforce Object Query Language (SOQL) or Salesforce Object Search Language (SOSL) queries as appropriate, to help you answer the question.
- Salesforce Object Search Language (SOSL) can be used to construct text-based search queries against the search index.
- Your generation should always be a Thought followed by an Action command and NOTHING ELSE. Generate only one Thought and one Action command.
- DO NOT generate ANY system observation, you will receive this based on your Action command.
- If no record is found matching the requirements mentioned, just return 'None'.

Here is a description of how to use these commands:
Thought
- A single line of reasoning to process the context and inform the decision making. Do not include any extra lines.
- Format: <thought> your thought </thought>

Action
- Can be 'execute' or 'submit'.
- execute, to execute SOQL/SOSL that will return the observation from running the query on the Salesforce instance.
- submit, to return the final answer of the task to the user.
- Format: <execute> a valid SOQL/SOSL query </execute> or <submit> response to user </submit>

Guidelines
- Always start with a Thought and then proceed with an Action.
- Generate only one Thought and one Action command at a time.
- Execute SOQL/SOSL queries to understand the Salesforce instance that will help you find the answer to the question.
- When you are confident about the answer, submit it.
- Always end with a submit action containing ONLY the answer, NO full sentences or any explanation.

Example 1
Question: What is the total number of opportunities?
Output:
<thought> I need to find the total number of opportunities in the system. </thought>
<execute> SELECT COUNT() FROM Opportunity </execute>
(If the observation from the Salesforce instance 100, your next step can be)
<thought> I have found the total number of opportunities. </thought>
<submit> 100 </submit> NOT <submit> The total number of opportunities is 100 </submit>

Example 2

[... Hide details for space...]

Salesforce instance description
The objects available in the Salesforce instance are:
User, Account, Contact, Case, Knowledge__kav, ProductCategory, Product2, ...
The fields available for the objects along with their descriptions and dependencies are:
User
- FirstName: First name of the agent
- LastName: Last name of the agent
- Email: Email address of the agent

[... Hide details for space...]

Additional task context
Handle/Transfer Times Policies
[... Hide details for space...]

Table 11: The system prompt used in the ReAct setting.

- Unrealistic: The objects had recognizable features but were generally not representative of actual Salesforce objects.
- Neutral: The objects were moderately realistic, combining elements of both realistic and unrealistic features.
- Realistic: The objects were mostly realistic and aligned well with typical objects used in Salesforce, with minor issues.
- Very Realistic: The objects felt entirely authentic and perfectly matched real-world Salesforce objects.

Instructions
- You are an expert in Salesforce and you have access to a Salesforce instance.
- You will be provided a question, the system description, and relevant task context.
- Interact with the Salesforce instance using the tools provided to help you answer the question.
- You should ALWAYS make ONLY ONE tool call at a time.
If you want to submit your final answer, use the 'submit' tool.
If not, you should call some other tool. But ALWAYS make a tool call.
- Always end by calling 'submit' tool containing ONLY the answer, NO full sentence or any explanation.
- If your answer is empty that is there are no records found matching the requirements mentioned,
just return 'None' to the 'submit' tool.
Additional task context
Case Routing Policy
The case routing policy determines the best agent to assign the given new case based on the following criteria
- Issue Expertise: The agent who has closed the most cases
associated with the issue most similar to the given case.
- Product Expertise: If there is a tie in issue expertise, the best agent is the one who has solved the most cases
associated with the product most relevant to the given case.
- Workload: If there is still a tie, the best agent is the one that has least cases with Status not 'Closed'.
Domain Details
Quarters of the Year
- Q1: January 1 to March 31 (both inclusive).
- Q2: April 1 to June 30 (both inclusive).
- Q3: July 1 to September 30 (both inclusive).
- Q4: October 1 to December 31 (both inclusive).
Seasons
- Winter: December 1 to February 28/29 (both inclusive).
- Spring: March 1 to May 31 (both inclusive).
- Summer: June 1 to August 31 (both inclusive).
- Autumn/Fall: September 1 to November 30 (both inclusive).
Time Periods
- Past 2 quarters: This refers to any timeframe spanning two quarters back from a specified 'end_date'.
That translates to a six-month period retrospectively from the 'end_date'.
- Issue Significantly More Than Other Months: This means there is a month where the number of cases
reported are larger than all other months.

Table 12: The system prompt used in the Function Calling setting.

Profession	Gender	Age
Customer Service Associate	Female	23
Customer Service Associate	Female	25
Customer Service Agent	Male	39
Customer Service Associate	Male	29
Customer Service Advisor	Male	49
Customer Service Manager	Male	39
Account Executive	Female	25
Technical Support	Female	38
Customer Service Advisor	Female	25
Customer Service Agent	Female	35

Table 13: The background of the participants in our expert study.

F.3 Qualitative Feedback

In Table 14, we present qualitative feedback and rationale from the experts we interviewed, as they determine whether our Organization and Object are perceived as Realistic or Unrealistic.

G Implementation Details

We use the OpenAI API for the `gpt` models; Amazon Bedrock API for the `claude` models; and the Together API for the `llama3.1` models. Below we provide the version of the model we tested:

- `o1`: o1-2024-12-17
- `gpt-4o`: gpt-4o-2024-08-06
- `gpt-3.5-turbo`: gpt-3.5-turbo-0125
- `deepseek-r1`: deepseek-ai/DeepSeek-R1
- `claude-3.5-sonnet`: anthropic.claude-3-5-sonnet-20240620-v1:0
- `claude-3-sonnet`: anthropic.claude-3-sonnet-20240229-v1:0
- `llama3.1-405b`: meta-llama/Meta-Llama-3.1-405B-Instruct-Turbo
- `llama3.1-70b`: meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo
- `llama3.1-8b`: meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo

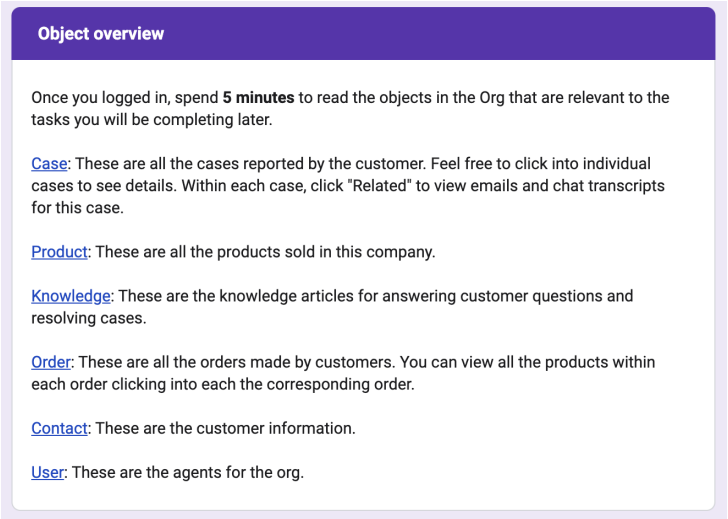


Figure 9: The instructions and interface of Part 1 of our expert study.

We choose the ReAct setting over Plan based approaches that decompose the task into more manageable steps as prior works showed that in SQL based database querying tasks, planning strategy is less flexible to altering its plan by adjusting to execution feedback (Yang et al., 2024). We set the max actions for each instance to 20, temperature to 0, and top_p to 1 for all experiments.

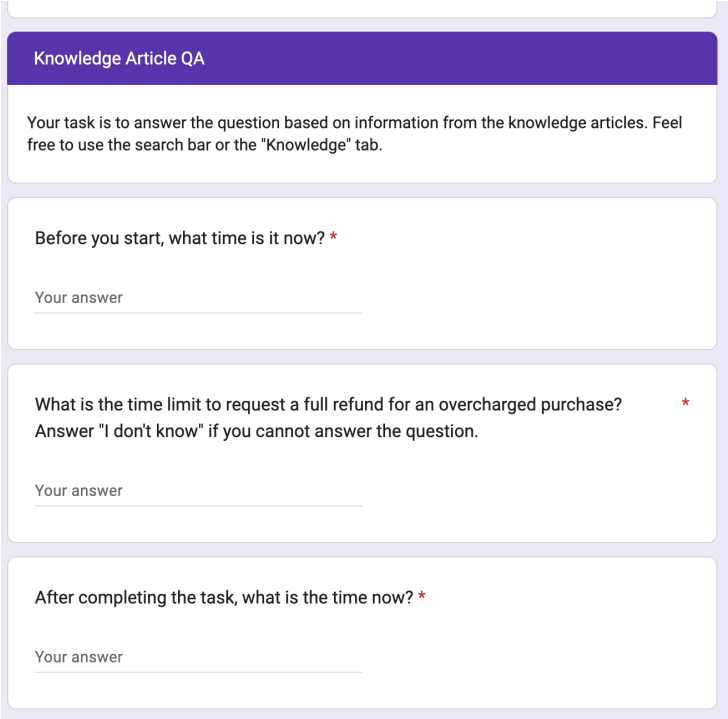


Figure 10: An example query instance for the part 2 of expert study.

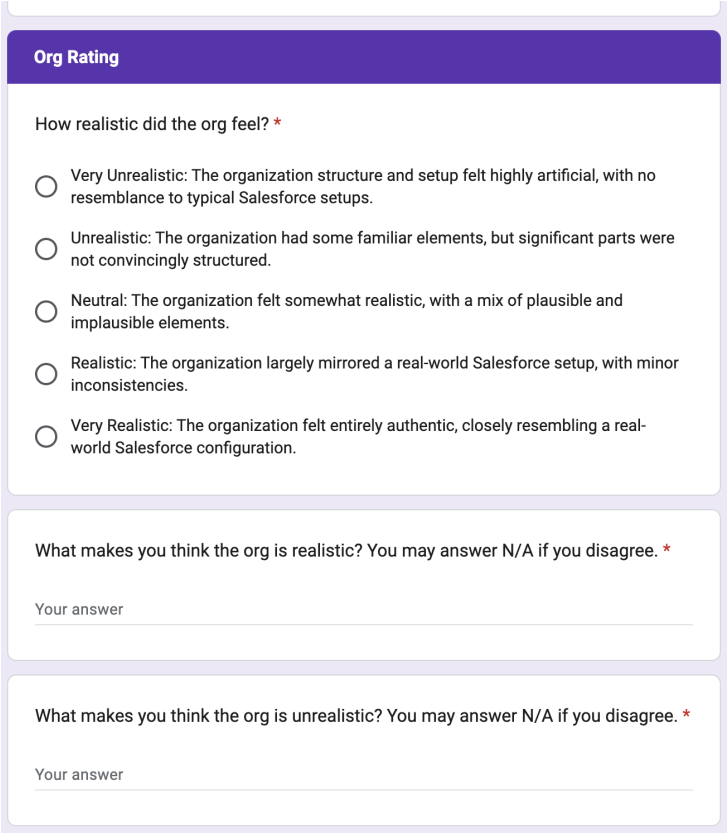


Figure 11: An example question for the part 3 of our expert study.

Rated Instance	Rating	Rationale
Org	Realistic	1. This is really similar to what a normal Salesforce instance looks like (i.e. the one we use at our company). However, there are a few missing details in some of the pages like when you click into a contact or account. 2. It feels like my usual Salesforce Dashboard for my current job, I could more or less get a feel for the general navigation of the simulation. 3. This is what salesforce looks like for me to find case numbers and information about each of the cases that were indentified by customers. 4. Knowing nothing about the org I was able to fumble my way around and find what I needed to.
	Unrealistic	1. The lack of customer data/information filling out the rest of the fields. There is no semblance of a system that's been "worked in" and everything feels very empty and confusing with nothing to fill the interface.
Object	Realistic	1. Case management, customer interactions, knowledge base, and the transcripts were what made it realistic. 2. I think the email correspondence wasn't perfect, but it did feel rather authentic. 3. I feel like the cases and customers issue are real life issue so I feel like they are realistic. 4. They have similar details and structures as a typical salesforce deployment (at least in my company). A lot of those elements have the same fields that are in their expected places (like Details, additional context on the right side)
	Unrealistic	1. The unrealistic ones are finding the agent information. This is unrealistic because I should be able to filter and find each of the agent transfers and handle time with the customers.

Table 14: Example rationales provided by domain experts for their ratings of our sandbox environment's realism.

Functionality	Dependency	Function	Task
QUERY	INDEPENDENT	`get_order_item_ids_by_product(product_id)`	MTA
		`get_order_item_ids_by_product(product_id)`	NCR
		`search_products(search_term)`	NED
		`get_account_id_by_contact_id(contact_id)`	NED
QUERY	DEPENDENT	`get_non_transferred_case_ids(start_date, end_date)`	HTU
		`get_cases(start_date, end_date, agent_ids, case_ids,`	NCR
		`get_cases(start_date, end_date, agent_ids, case_ids,`	BRI
		`get_cases(start_date, end_date, agent_ids, case_ids,`	HTU
CALCULATION	INDEPENDENT	`get_start_date(end_date, period, interval_count)`	TCU
		`get_start_date(end_date, period, interval_count)`	BRI
		`get_start_date(end_date, period, interval_count)`	TII
		`get_period(period_name, year)`	TCU
CALCULATION	DEPENDENT	`calculate_region_average_closure_times(cases)`	BRI
		`get_qualified_agent_ids_by_case_count(agent_handled_cases, n_cases)`	TCU
		`calculate_average_handle_time(cases)`	HTU
		`get_agents_with_max_cases(subset_cases)`	NCR

Table 15: The list of functions and tasks tested in Table 4.