



Can LLMs Help You at Work? A Sandbox for Evaluating LLM Agents in Enterprise Environments

Harsh Vishwakarma* Ankush Agarwal* Ojas Patil
Chaitanya Devaguptapu Mahesh Chandran

Fujitsu Research

EnterpriseBench - Tech Blog

{harsh.vishwakarma, ankush.agarwal}@fujitsu.com

Abstract

Enterprise systems are crucial for enhancing productivity and decision-making among employees and customers. Integrating LLM based systems into enterprise systems enables intelligent automation, personalized experiences, and efficient information retrieval, driving operational efficiency and strategic growth. However, developing and evaluating such systems is challenging due to the inherent complexity of enterprise environments, where data is fragmented across multiple sources and governed by sophisticated access controls. We present EnterpriseBench, a comprehensive benchmark that simulates enterprise settings, featuring 500 diverse tasks across software engineering, HR, finance, and administrative domains. Our benchmark uniquely captures key enterprise characteristics including data source fragmentation, access control hierarchies, and cross-functional workflows. Additionally, we provide a novel data generation pipeline that creates internally consistent enterprise tasks from organizational metadata. Experiments with state-of-the-art LLM agents demonstrate that even the most capable models achieve only 41.8% task completion, highlighting significant opportunities for improvement in enterprise-focused AI systems.

1 Background and Introduction

Large Language Models (LLMs) are fundamentally transforming how enterprises operate, driving improvements in productivity across departments (Plumb, 2025; Meta, 2024; Carlini, 2024). These models have demonstrated remarkable capabilities in automating knowledge-intensive tasks, from question answering and code generation to report writing and data analysis (Brachman et al., 2024; Jiang et al., 2024; GitHub, 2024). Recent advancements have led to emergence of Compound AI Sys-

tems (CAI) (Zaharia et al., 2024; Lin et al., 2024) (also referred to as Agents (LangChain, 2024; Anthropic, 2024a)) that can orchestrate complex workflows for solving various tasks. These systems, exemplified by tools like Devin (Labs, 2024) and Glean (Glean), can automatically search across information sources, analyze data, and even initiate actions when human intervention is needed.

However, developing effective CAI systems for enterprises faces a critical challenge: enterprise data is inherently complex and fragmented across multiple sources, including email systems, Customer Relationship Management (CRM) platforms, SharePoint sites, internal wikis, and ticketing systems. This fragmentation is further complicated by sophisticated access control mechanisms that govern who can access specific information resources. Even seemingly simple queries often require orchestrating data gathering from multiple sources, executing database calls, and performing complex reasoning across diverse information types. While current research has made progress in developing CAI systems for specific use-cases relevant to enterprises, the unique challenges of enterprise environments—particularly around data fragmentation and access control—remain largely unaddressed with current CAI systems.

To illustrate challenges and complexities of the CAI, consider an enterprise specific scenario: an employee asks, *"Create a GitHub repository named EnterpriseBench and generate a notification message to my manager informing him about the repository creation."* This seemingly straightforward request requires a complex workflow that traditional approaches like Retrieval-Augmented Generation (RAG) (Bruckhaus, 2024) and existing LLM agents (Talebirad and Nadiri, 2023; Zhang et al.; Li et al., 2019) struggle to handle. A robust enterprise-specific CAI system must orchestrate multiple sub-tasks for this: create the GitHub repository EnterpriseBench, resolve the sender and recipient details,

*Equal contribution as co-first authors.
Code | Data

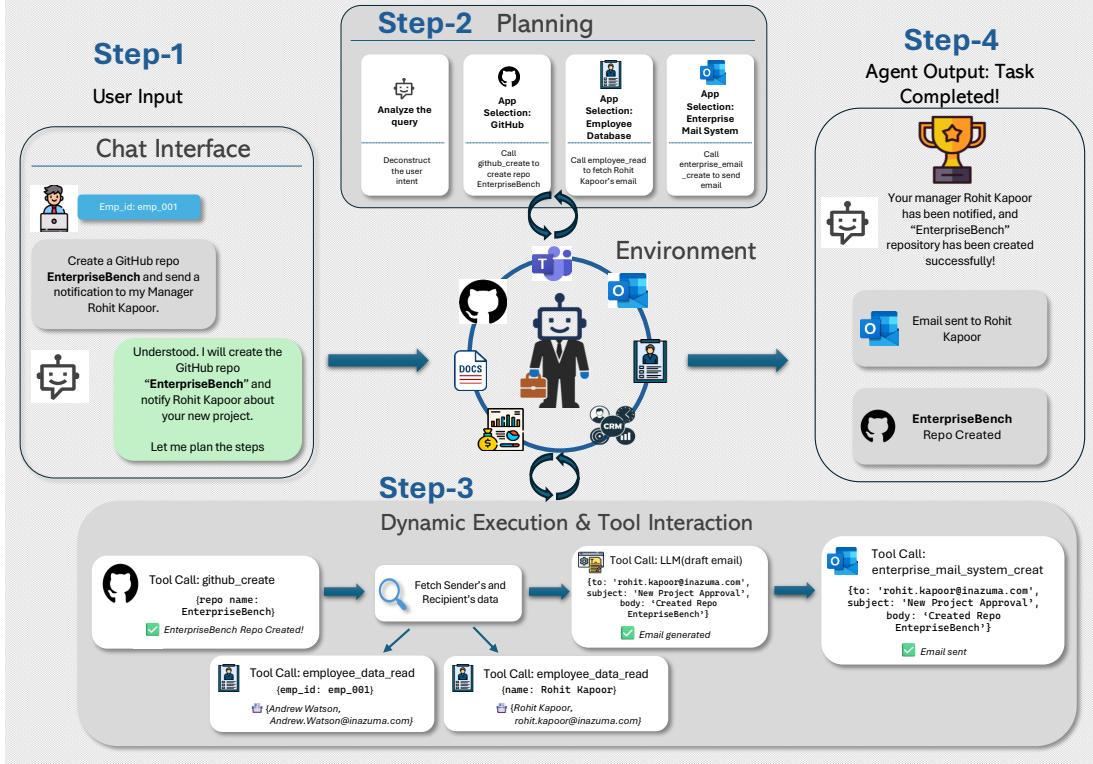


Figure 1: **Task Execution in EnterpriseBench.** This figure illustrates how an LLM-based agent interacts with the enterprise environment. Given a task, the agent perceives the available enterprise tools, applications, and data sources, formulates a reasoning plan, and executes actions to complete the task.

generate a formal notification message—all while respecting access controls and organizational hierarchies. These requirements highlight the need for sophisticated CAI systems that can (1) integrate multiple enterprise data sources and tools, (2) enforce access controls, (3) coordinate multiple tasks, and (4) maintain context across system interactions (as shown in Figure 1).

To enable development of such systems, we introduce EnterpriseBench, the first comprehensive benchmark that simulates the data from enterprise environments. By providing a benchmark that mirrors complexities of real-world scenarios without using sensitive real data, EnterpriseBench enables rapid prototyping and evaluation of CAI systems for enterprise settings. This allows organizations to validate and refine their CAI systems before deploying them on actual enterprise data. Our dataset spans multiple domains, including Software Engineering (code repositories, documentation), Sales and CRM (customer interactions), Finance (budgets, expense reports), IT support (ticketing systems, incident reports), HR (policies, employee records), and Internal Communication platforms (simulated team and email conversations). EnterpriseBench emphasizes persona-based tasks that require adherence to access controls and organiza-

tional hierarchies. Additionally, we also introduce an automated task creation framework that generates complex, multi-source tasks conditioned on persona roles and enterprise constraints.

We conduct a comprehensive evaluation of five large language models, including GPT-4o (Hurst et al., 2024), Claude 3.5 (Anthropic, 2024b), O1-mini (OpenAI, 2024), LLaMA (Touvron et al., 2023)—to assess their ability to generate complete plans for accomplishing a given task. Our evaluation spans four planning strategies, including ReAct (Yao et al., 2022b) and Chain-of-Thought (CoT) (Wei et al., 2022), implemented using two different frameworks, LangChain (LangChain, 2024) and DSPy (Khattab et al., 2024). Our key contributions are listed below.

- A comprehensive benchmark of 500 enterprise tasks across IT, HR, Sales and Finance, featuring multi-step reasoning, access controls, and cross-functional workflows.
- Our comprehensive evaluations shows a significant performance gap in current CAI systems, with even state-of-the-art models achieving only 41.8% task completion.
- A simulated enterprise sandbox environment is created for benchmark development, comprising

data domains such as chat systems, emails, and code workflows, along with representative employee information aligned with these domains.

- A persona-based task framework that generates contextually appropriate challenges, testing both technical capabilities and organizational constraints.

2 Related Work

Compound AI Systems LLMs have emerged as powerful tools, demonstrating excellence in tasks such as processing and generating human-like text (Team et al., 2023; Achiam et al., 2023), writing code (Chen et al., 2021), and performing complex reasoning (Khetan et al., 2020). Beyond these fundamental capabilities, LLMs show immense potential within Compound AI Systems, enabling collaborative problem-solving, dynamic interactions, and advanced decision-making (Yao et al., 2022b; Xi et al., 2023; Wei et al., 2022). As tasks grow in complexity and scope, leveraging multiple LLMs in a cooperative framework becomes a natural strategy to enhance their effectiveness. To evaluate these systems, specialized benchmarks are developed, which are discussed in the next module.

Evaluation of Compound AI System Compound AI Systems have been developed to address a wide range of tasks, including scientific experimentation (Ghafarollahi and Buehler, 2024; Boiko et al., 2023; M. Bran et al., 2024), embodied intelligence (Brohan et al., 2023), societal simulations (Gao et al., 2023; Li et al., 2023), and web-based environments such as Mind2Web (Deng et al.), WebArena (Zhou et al.), and WebShop (Yao et al., 2022a). Recently, benchmarks have begun to emerge for more specialized settings, such as software engineering (Jimenez et al.; Li et al., 2024), computing environments (Xie et al., 2024b; Bonatti et al.), workplace (Styles et al.), text-to-SQL workflows (Lei et al.), and real-world task planning (Yao et al., 2025; Liu et al.; Xie et al., 2024a). Despite these advancements, there remains a significant gap in the development of enterprise-simulated environments that reflect real-world, day-to-day business operations. The closest efforts in this direction such as Xu et al. (2024a); Huang et al. (2025) focus on narrow domains like database management or CRM systems. However, none of them address the challenges of managing large volumes of data spread across diverse domains, formats, and systems—a key requirement for evaluating Compound AI Systems in realistic enterprise settings.

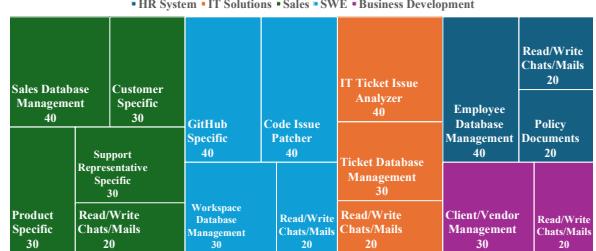


Figure 2: Classification of Tasks by Domain (counts)

To address this gap, we propose a novel benchmark, EnterpriseBench, specifically designed for enterprise scenarios. This benchmark offers a robust framework for evaluating LLM-based agents under realistic, domain-relevant conditions, thereby supporting the development of effective and reliable enterprise AI systems. A comparison with other related benchmarks is presented in Table 5.

3 EnterpriseBench: Crafting a Simulated Enterprise Benchmark

We have developed an enterprise sandbox environment that simulates a realistic company setting. This environment includes synthetic company data enriched with employee-specific details such as chat logs, emails, and GitHub activity. The data sources are constructed by gathering publicly available information from the internet and applying rule-based processing techniques, guided by domain experts to ensure authenticity. Based on this simulated data, a variety of enterprise tasks are generated within the sandbox, with strict access control policies in place to support secure and realistic interactions.

The subsequent sections elaborate on the key components of our benchmark. Section 3.1 outlines the design of enterprise tasks. Section 3.2 details the simulation of the enterprise sandbox, followed by the automatic task construction pipeline described in Section 3.3. Section 3.4 presents the API calls and functions implemented within the sandbox to support LLM agents. Finally, Section 3.5 reports an expert study conducted to assess the realism and validity of the sandbox environment and tasks.

3.1 EnterpriseBenchTasks

Our benchmark includes 500 enterprise tasks spanning five major domains: Human Resources (HR), Information Technology (IT), Software Engineering (SWE), Business Operations, and Sales. Each task is carefully designed to assess the capabilities of Compound AI systems in enterprise setting.

Domain	Task Description	User Employee	Task Category	Tools
SWE	Create a GitHub repo EnterpriseBench and notify my Manager Rohit Kapoor.	emp_001 (Level-10)	CRUD	github_create, enterprise_mail_system_create, employee_data_read
IT Management	Can you show me the ticket IDs for urgent issues I'm involved with?	emp_1106 (Level-10)	Search	it_service_management_read
Business Development	Can you help me find the thread for my recent conversations with Julian? I need it for follow-up.	emp_1180 (Level-12)	Search	conversations_read
Employee Database Management	Can you provide a breakdown of my total leaves taken so far this year?	emp_0726 (Level-09)	Search	employee_data_read
HR	Send a mail to Rahul Khanna to notify the employees regarding new PoSH policy document.	emp_0653 (Level-09)	CRUD	employee_data_read, enterprise_mail_system_create

Table 1: Examples of EnterpriseBench tasks across domains, categorized by task type and tools.

To capture a broad range of functionalities, the tasks are grouped into three primary categories: search tasks, CRUD (Create, Read, Update, Delete) tasks, and unanswerable, which account for 65%, 30%, and 5% of the benchmark, respectively. The domain-wise distribution of tasks is shown in Figure 2, and the average task complexity is defined by the number of tools required to solve a task, which is 3. Representative examples of tasks included in the benchmark are shown in Table 1.

3.2 EnterpriseBench Sandbox: Simulating Enterprise Data and Roles

The enterprise sandbox environment is developed with careful consideration of three key components: *Departments to Populate*, *Data Sources to Collect*, and *Compiling the Data to create the Simulation Environment*. We integrate both collected and synthetically generated data across multiple domains-HR, IT, Sales, Finance, and Software Development within a simulated organizational setting. Table 2 show details regarding the data sources in EnterpriseBench.

Employee data is sourced from Ayoobi et al. (2023), filtered to include only relevant departments. To reflect organizational structure, employees are categorized into four roles-Associates, Team Leads, Managers, and Directors distributed in a 4:3:2:1 ratio per department. Additional attributes such as salary, leave records, and joining dates are introduced to mimic real-world enterprise dynamics.

3.2.1 Sandbox Data Simulation

The data simulation strategy is based on two primary methodologies.

Leveraging the Collected Data We collect the enterprise-related data from different sources (de-

Application / Data Source	Content Description	# Instances
💬 Chats	Conversations between employees	3000
✉️ Enterprise Mail System	Internal and external email threads	4500
💻 Code Workspace	Source code, issues, issue git patch	1000
📞 Customer Relationship Management	Records on sales, product, customers, support chats, sentiment data, invoices, and purchase orders	30195
📄 Enterprise Policy Documents	Policy and compliance documents	24
💻 IT Service Management	Support tickets and issue logs	163
💡 Enterprise Internal Overflow	Similar to Stack Overflow	5000
🧠 Enterprise Social Blog	Internal blog posts, company news and updates	1000
👤 HR Management System	Employee records, resumes and organizational structure	1260
💻 Business and Management	Client Details, Vendor Details,	800

Table 2: Details of Data Sources/Applications in the EnterpriseBench Simulated Sandbox Environment

tails in Table 10) and use it to simulate the sandbox environment. Below, we explain how it is utilized.

- **Data Source Coverage:** Domain experts (details in Appendix A.3) identified essential data sources for each department. For example, the Sales department should include Customer Support Chats, Product Sentiment Data, Product, Customer, Sales Data, Invoices, Purchase Orders, and Shipping Records.
- **Pre-processing:** Collected data undergoes struc-

tural preprocessing, including extraction of entities (e.g., products, customers) and generation of contextual data (e.g., support conversations).

- *Mapping to Employee Personas*: Data entries are linked to employee personas based on experience, skills, and roles. For instance, customer resolutions are semantically mapped to specific support personnel.
- *Enterprise Rephrasing*: Entries are rephrased using enterprise-specific metadata to ensure contextual consistency and realism.

Generating Conversations and Emails

Following the methodology in Xu et al. (2024b), realistic conversations and emails are generated and grounded in curated data to reflect authentic enterprise communication. More details on generation are available in Appendix A.4.

3.2.2 Access Control Simulation

To emulate enterprise-level security, we implement a dynamic Role-Based Access Control system, where permissions are assigned based on organizational role levels (specifically, Levels 9 through 14), task requirements, data sensitivity, and cross-departmental relationships. For example, enterprise social platforms are accessible to all employees, while access to internal repositories (such as GitHub) is restricted to designated technical teams and their management chain. Access control policies are initially generated with assistance from a LLM and subsequently validated by human experts.

3.3 EnterpriseBench Task Generation Pipeline

We designed an LLM-based task generation pipeline to produce structured, high-quality tasks that require access to relevant data sources and tools, while also enforcing persona-specific access controls. The pipeline comprises four key stages: a) selecting the initial domain and persona for the task, b) selection from expert curated goal templates, c) generating the corresponding task based on the selected context, and d) refining the task iteratively. A stepwise explanation is provided below.

3.3.1 Domain and Persona Selection

We begin by

- *Task Domain Selection*: Among the available domains such as HR, IT, we randomly select a target domain for which task has to be generated.

- *Persona Sampling*: From a set of personas curated by domain experts for each domain, a representative persona is sampled for the selected domain to serve as a proxy for task contextualization.
- *Context Retrieval*: From the prepared data sources available in the sandbox environment, relevant contextual information associated with the sampled persona and domain is retrieved to ground the task in a realistic enterprise scenario.

3.3.2 Expert Curated Goal Templates

Creating generalizable goal templates across departments is inherently challenging due to the diversity and specificity of enterprise tasks. To address this, we leverage the O*NET 29.2¹ release (Rounds et al., 1999), a comprehensive taxonomy of occupations and task definitions developed by the U.S. Department. We manually curated goal templates (examples in Table 9) tailored to departmental tasks, refining them through iterative reviews by domain experts to ensure contextual relevance and practical applicability.

3.3.3 Task Generation

We use the persona and domain-relevant context, along with the selected goal template and available tools, to initiate the task generation process using LLM calls. We begin by

- *Entity Extraction*: Filters are applied on the persona-specific context to structure the input, reducing token count and enhancing the precision of downstream processing. This structured representation improves task grounding by highlighting salient information.
- *Subgoal Decomposition*: The expert curated high-level goal is decomposed into fine-grained subgoals, including retrieval steps and action plans, by prompting the language model to operate in a closed, tool-aware environment. This stage introduces modularity into the task planning process.
- *Task Structure*: Based on the subgoals and extracted entities, task structure is defined that can be mapped with the context entities. The structure mirror the reasoning sequence or plan that a compound AI system would follow to execute the complete task.
- *Final Task Generation*: The final task is assembled by synthesizing the goal, subgoals, entities,

¹O*NET 29.2

and task structure, resulting in a fully formed, executable task representation.

A comprehensive description of how the ground truth is established can be found in Appendix A.1.

3.3.4 Iterative Improvement

Inspired by the iterative refinement method proposed in Kim et al. (2023); Yao et al. (2025), a validation and rephrasing loop is applied. The generated task and ground truth is iteratively revised until it passes a checklist of validation criteria designed by human experts, ensuring clarity, feasibility, and alignment with task objectives.

We provide the end-to-end task generation procedure in Algorithm 1, included in Appendix A.1. The LLM prompts used for task generation are detailed in Appendix A.5. Information about the domain experts involved in designing goal templates, filtering ambiguous tasks, and other aspects of task generation is provided in Appendix A.3.

3.4 Tools: API and Functions

EnterpriseBench incorporates a suite of tools and functions designed to simulate enterprise operations across diverse domains (see Table 8 for the tool inventory). For search-based tasks, agents use domain-specific application interface tools that return results based on employee ID or semantic matching. CRUD-based tasks leverage create, read, update, and delete operations for each data source, enabling dynamic data manipulation. To reflect enterprise settings, tool and function outputs are regulated by an access control mechanism that enforces permission constraints.

3.5 Expert Study

To ensure the correctness, realism, and practicality of EnterpriseBench, we conducted a user study involving ten experts from diverse professional backgrounds. The experts were selected through a Microsoft Form circulated internally; additional details on form and experts are provided in Appendix A.3. The selection was based on their domain expertise to ensure relevant and informed feedback. During the study, participants were first introduced to the sandbox environment, followed by a set of questions designed to assess their understanding of the setup. They were then presented with search-based and CRUD-based tasks and asked to find an-

swers or perform the required operations. This step helped assess the correctness of the tasks. Subsequently, participants evaluated the realism of each task using a scale ranging from “very unrealistic” to “very realistic,” and provided justifications for their ratings. Based on the scores, we applied a filtering process to retain only those tasks that met enterprise-specific quality standards. As a result, 80% of the tasks were considered suitable, meaning they were correct, realistic, and aligned with enterprise applications, while the remaining tasks were discarded.

4 Experimental Setup

4.1 Enterprise LLM Agent Setup

To efficiently solve our enterprise search tasks, we design an LLM-based agent that follows a structured multi-step approach. Given a primary goal or task T , the agent creates a plan by decomposing it into sub-goals or sub-tasks $P = \{p_1, p_2, \dots, p_n\}$ using a reasoning-based method. These sub-goals are then refined into well-defined, solvable steps $S = \{s_1, s_2, \dots, s_n\}$. The agent, defined as $A = f(\Theta, \mathcal{K})$, where Θ and \mathcal{K} are model parameters and prior knowledge, selects the appropriate tools or API functions T to optimize information retrieval and processing. It then iteratively executes each sub-task, constructing the final answer A or executing the final task.

This setup ensures reliable execution of EnterpriseBench tasks by leveraging LLM Agents for multi-step reasoning, tool utilization, and execution.

4.2 Experimental Settings

This section outlines our experimental setup, detailing the experimental data, baseline methods used to evaluate our benchmark, the evaluation metrics employed, and the implementation specifics.

4.2.1 Experimental Dataset

We conduct experiments by building CAI systems with a range of models. For evaluation with LangChain and DSPy, we use the benchmark of 500 samples. For supervised fine-tuning (SFT), we expand the dataset to 1k samples using our task generation pipeline and split it into training and test sets with a 4:1 ratio. For Direct Preference Optimization (DPO) (Rafailov et al., 2023), we conduct experiment by creating 1200 preference pairs from SFT training examples, following the procedure described in OS-Genesis (Sun et al., 2024). The

dataset formats for SFT and DPO are illustrated in Listings 1 and 2, respectively in Appendix.

4.2.2 Baseline Methods

To evaluate the performance on the EnterpriseBench benchmark, we conducted experiments using several state-of-the-art models: GPT-4o², o1-mini³ (via Azure AI Foundry), Anthropic Claude 3.5-Sonnet⁴ (`anthropic.claude-3-5-sonnet-20240620-v1:0`) from Amazon Bedrock, as well as Llama-3.1-8B, and Llama-3.3-70B, also accessed via Amazon Bedrock. Building on these models, CAI system baselines using a variety of planning strategies: no planning, Chain-of-Thought (CoT) reasoning (Wei et al., 2022), ReAct-style reasoning (Yao et al., 2022b), and goal-aware planning. To implement these systems, we adapted state-of-the-art agent frameworks, namely LangChain (LangChain, 2024) and DSPy (Khattab et al., 2024). For DSPy, we employ an optimization-based few-shot prompting approach, while for LangChain, we provide two few-shot examples with each LLM call. Each system is designed to decompose primary goals into subgoals, select relevant data sources and tools, verify access controls, and execute tasks in an end-to-end manner, ensuring alignment with enterprise-specific requirements.

4.2.3 Implementation Details

Experiments were conducted using two NVIDIA GPUs (80 GB each) for SFT and DPO training. Additional 8 GB GPUs were employed to load retrievers such as Colpali for implementing the EnterpriseBench environment, while LLM inference was carried out through APIs.

- *Data Simulation*: We utilized GPT-4o² to generate and rephrase all components of EnterpriseBench, ensuring consistency and high-quality data synthesis.
- *Task Generation*: The task generation process was conducted using GPT-4o², implementing an end-to-end pipeline. Additionally, Anthropic Claude 3.5-Sonnet⁴ was employed for final quality assessment of the generated tasks. It took approximately 1 minutes and 20 seconds to generate a single task.
- *Tool Dependency and Execution*: Tool dependencies were defined using a structured JSON file containing detailed descriptions of all tools

²<https://platform.openai.com/docs/models#gpt-4o>

³<https://platform.openai.com/docs/models#o1>

⁴<https://aws.amazon.com/bedrock/clause/>

within EnterpriseBench. For tool execution, API calls were made to invoke various external tools. Further details on tool specifications and implementations can be found in Table 8.

- *Context Retrieval*: We implemented *id* based context retriever for text-based structured data, Colpali (Faysse et al., 2024) for PDF documents, and query-to-SQL retrievers inspired by (Zhang et al., 2025) for tabular content.
- *SFT+DPO*: We implemented SFT using LoRA (Hu et al., 2022), targeting the modules `q_proj`, `k_proj`, `v_proj`, and `o_proj`. All other hyperparameters followed the default `LoraConfig` in the TRL library from Hugging Face⁵. DPO was implemented using the `DPOTrainer` from TRL with the same hyperparameters as SFT.

The hyperparameter configurations for LLM API calls and retrievers are summarized in Table 12 and Table 13 in Appendix.

4.3 Evaluation Metric

To evaluate Compound AI systems on EnterpriseBench, we assess the correctness of the final execution of each task. For all tasks, correctness is determined using Prometheus-2⁶ with GPT-4 and Gemini-2.5 Pro, as proposed by Kim et al. (2024), which provides a rubric-based score ranging from 1 to 5. For CRUD tasks, we first call the `read()` function to verify whether the task was executed correctly, and then apply rubric-based scoring to the `read()` output. In addition to automated evaluation, we conduct human evaluation focusing on two aspects: (a) whether the agent successfully completed the task, and (b) experts are required to complete the task. A separate set of experts then assess the correctness of these human-executed tasks. Scores are averaged across three experts serving as annotators.

For the evaluation of SFT and DPO, the trained model generates planning or action steps, and the LangChain framework is used to execute the tasks. Evaluation is performed using Prometheus-2 with Gemini-2.5 Pro, consistent with the evaluation methodology applied to the CAI systems.

5 Results and Analysis

In this section, we evaluate our benchmark, EnterpriseBench, using five LLM agents built with state-of-the-art reasoning models: GPT-4o, Claude 3.5 Sonnet, Llama 3.1 8B, Llama 3.3 70B, and O1-mini. The agents are tested under different planning

⁵<https://huggingface.co/docs/trl/index>

⁶LlamaIndex Prometheus-2 Cookbook

Model	GPT-4 Evaluator			Gemini-2.5 Pro Evaluator				
	w/o Planning	CoT (Wei et al., 2022)	ReAct (Yao et al., 2022b)	w/ Gold Planning	w/o Planning	CoT (Wei et al., 2022)	ReAct (Yao et al., 2022b)	w/ Gold Planning
LangChain Framework (LangChain, 2024)								
GPT-4o	0.29	0.27	0.32	0.43	0.27	0.28	0.29	0.44
Claude-3.5-Sonnet	0.31	0.27	0.28	0.38	0.32	0.30	0.30	0.41
o1-mini	0.31	0.28	0.35	0.51	0.28	0.27	0.39	0.47
Llama-3.1-8B	0.04	0.06	0.14	0.20	0.03	0.04	0.13	0.21
Llama-3.3-70B	0.23	0.22	0.21	0.40	0.24	0.23	0.21	0.36
DSPy (Khattab et al., 2024)								
GPT-4o	0.19	0.32	0.34	0.50	0.25	0.26	0.29	0.47
Claude-3.5-Sonnet	0.19	0.24	0.30	0.50	0.21	0.29	0.29	0.44
o1-mini	0.29	0.33	0.38	0.62	0.27	0.32	0.41	0.63
Llama-3.1-8B	0.10	0.14	0.16	0.34	0.07	0.15	0.15	0.34
Llama-3.3-70B	0.20	0.27	0.30	0.47	0.24	0.25	0.28	0.48

Table 3: **EnterpriseBench Evaluation**: Comparison of performance across agents using different models and planning strategies with LangChain and DSPy frameworks, evaluated by GPT-4 and Gemini 2.5 Pro on 500 samples.

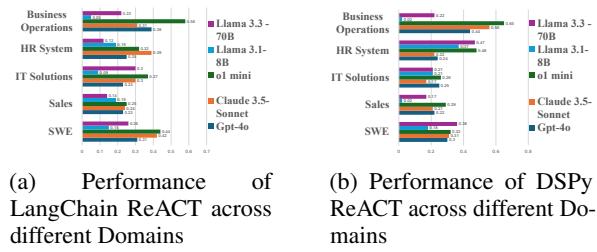


Figure 3: Comparison of different models using ReAct planning: Performance across different domains of EnterpriseBench.

strategies implemented via LangChain and DSPy. We further report results from human evaluation, assessing both the correctness of agent responses and the successful execution of tasks. In addition, we present results from a model trained on EnterpriseBench, and provide an in-depth analysis of the evaluation outcomes for CAI systems.

5.1 Evaluation on Enterprise Search Tasks

Compound AI System Evaluation Table 3 presents the evaluation of our benchmark across various models, planning strategies, and frameworks, scored using Prometheus-2 with GPT-4. ReAct-based planning outperforms both no-planning and CoT approaches across both frameworks. Among the models, O1-mini achieves the best performance, as expected given its advanced reasoning capabilities. The open-source Llama models show a significant performance drop compared to the higher-performing models, highlighting the need to improve their planning abilities. Notably, gold planning yields the highest accuracies, with approximately 40% to 50% improvements over ReAct. This substantial difference underscores the necessity for more sophisticated agents and frameworks capable of handling complex planning tasks in enterprise settings, which

require coordination across multiple sources, tools, and function calls to successfully complete the final task. We also report performance across all domains using ReAct planning in figure 3. Additionally, human evaluation was conducted on the agent built with O1-mini using ReAct planning within the LangChain framework, demonstrating an accuracy of 31%.

To further evaluate the performance of current LLM agents, we conducted a human CAI (humans acting as LLM agents) study to assess task execution. The accuracy achieved was 70%, highlighting the gap between human performance and that of LLM agents in the enterprise setting. While human agents achieved higher accuracy, this came at the cost of significantly increased average completion time—from 50 seconds with agents to 8 minutes 30 seconds per task with humans—revealing a clear trade-off between precision and efficiency. These findings suggest that there is room to improve planning strategies in current LLM agents to achieve precision levels comparable to humans while maintaining significantly faster execution times.

Trained Model Evaluation We conducted an additional experiment by training the Qwen3-8B model on data generated through our task generation pipeline. The model was fine-tuned using both supervised fine-tuning (SFT) and direct preference optimization (DPO) to predict planning or execution steps based on the task and available tools, with task execution carried out through the LangChain framework alongside GPT-4o. As shown in Table 4, Qwen3-8B achieved 27% accuracy with SFT and 29% with SFT+DPO on 1.2k samples, closely approaching GPT-4o with CoT. These results highlight the effectiveness of our benchmark and task generation pipeline, showing that even with limited

Model	GPT-4o w/ CoT	Qwen3-8B (SFT)	Qwen3-8B (SFT+DPO)
Score	0.27	0.27	0.29

Table 4: Performance comparison across GPT-4o w/ CoT and Qwen3-8B models using the LangChain framework for task execution on 200 samples. DPO results are reported with 1.2k preference pairs.

training data, small models can achieve competitive performance with, and in some cases surpass, larger LLMs such as GPT-4o. This provides a proof of concept that for domain-specific tasks, small language models (SLMs) trained with high-quality data can outperform general-purpose LLMs.

5.2 In-Depth Analysis

We conduct an error analysis of the 01-mini ReAct agent implemented with LangChain. The evaluation was performed on 100 EnterpriseBenchtasks, uniformly distributed across domains. The agent achieved an accuracy of 31%, with the remaining cases classified as failures. Below, we outline the key failure modes identified through human evaluation.

- *Wrong Tool Selection / Wrong App Selection (18):* These errors arise from the complexity of tasks requiring multiple tool calls, as well as limitations in the model architectures used by LLM agents. We observed that models such as 01-mini perform slightly better in this regard compared to GPT, Claude, and other open-source models. Domain-wise performance, presented in Table 6 and Table 7 in the Appendix, shows that GPT performs well in HR and IT tasks, Claude excels in coding tasks, and 01-mini outperforms others in several non-technical domains. Performance in this area could be improved by incorporating continual learning, which would enhance the agent’s ability to understand the environment and make more accurate tool selections.
- *Search-based Answer Hallucination (8):* The agent sometimes relies on prior knowledge instead of the retrieved context, leading to hallucinations such as fabricated policy names, incorrect dates, or non-existent entities, thereby compromising factual accuracy. This limitation could be mitigated through improved agent memory management.
- *Context Retrieval (2):* The agent sometimes retrieves incomplete or irrelevant enterprise context due to weak query formulation or mismatches between the retrieval index and task intent, which leads to incorrect responses. Improving retriever performance requires going beyond similarity

matching.

- *Task Decomposition (20):* These errors often arise from the complexity of the tasks and the agents’ limited understanding of the sandbox environment. Performance in this area could be improved by employing a trained LLM agent rather than relying solely on general knowledge and few-shot examples.
- *Partial Factual Coverage (14):* Some answers align with task goals but omit critical structured details (e.g., employee IDs, policy names, dates), reducing reliability and highlighting the need for precision in enterprise settings. Performance can be improved by using constrained decoding or function-calling approaches, which ensure that all required structured fields are consistently produced.
- *Final Step Execution (7):* Even with correct subgoals, the final synthesis step may miscombine results, leading to incorrect answers and exposing gaps in temporal or logical consistency. Performance in this area could be improved by incorporating step validation or structured reasoning mechanisms to ensure accurate integration of intermediate outputs.

Our findings highlight that enterprise agents require tighter coupling between planning, retrieval, and grounding mechanisms, along with robustness against hallucinations and tool invocation errors. These insights aim to support the development of next-generation agentic systems that meet the strict accuracy demands of enterprise environments.

6 Conclusion

In this paper, we highlight the importance of Compound AI Systems in enterprise settings and the need for a benchmark to evaluate their performance. To address this, we introduce EnterpriseBench, a novel benchmark designed to assess CAI systems on complex enterprise tasks. Our experiments show that even state-of-the-art agents face significant challenges with these tasks. To create an evaluation environment, we develop an enterprise sandbox and a task framework, enabling the construction of comprehensive benchmark with minimal input.

Limitations

The limitations of our work are as follows: 1) Our enterprise data generation process requires an initial set of real enterprise data, which can be costly to obtain. Relying solely on synthetic data may

affect the realism of generated tasks. 2) Human experts are needed to verify intermediate steps during task generation, adding to the complexity and cost. 3) While we achieve high accuracy in enterprise task generation, some errors remain, suggesting areas for future improvement. 4) The evaluation of our benchmark relies on the current capabilities of reasoning models, which are likely to improve over time. 5) Our experiments did not involve large-scale data generation with terabytes of data, which would better represent real-world enterprise-scale scenarios.

Acknowledgement

We thank the members of the AI Lab at Fujitsu Research for their valuable feedback on this work. We are also deeply grateful to the anonymous ARR reviewers, the meta-reviewer, and the ACL program chairs for their thoughtful comments and suggestions, which significantly improved the paper.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024a. Building effective agents. <https://www.anthropic.com/research/building-effective-agents>.

AI Anthropic. 2024b. Claude 3.5 sonnet model card addendum. *Claude-3.5 Model Card*, 3(6).

Navid Ayoobi, Sadat Shahriar, and Arjun Mukherjee. 2023. The looming threat of fake and llm-generated linkedin profiles: Challenges and opportunities for detection and prevention. In *Proceedings of the 34th ACM Conference on Hypertext and Social Media*, pages 1–10.

Daniil A Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*.

Rogerio Bonatti, Dan Zhao, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, et al. Windows agent arena: Evaluating multi-modal os agents at scale. In *NeurIPS 2024 Workshop on Open-World Agents*.

Michelle Brachman, Amina El-Ashry, Casey Dugan, and Werner Geyer. 2024. How knowledge workers use and want to use llms in an enterprise context. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–8.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR.

Tilmann Bruckhaus. 2024. Rag does not work for enterprises. *arXiv preprint arXiv:2406.04369*.

Nicholas Carlini. 2024. How i use "ai"? <https://nicholas.carlini.com/writing/2024/how-i-use-ai.html>.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 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, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*.

Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. S3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.

Alireza Ghafarollahi and Markus J Buehler. 2024. Protagents: protein discovery via large language model multi-agent collaborations combining physics and machine learning. *Digital Discovery*.

GitHub. 2024. Github copilot: Your ai pair programmer. Accessed: Feb. 11, 2025.

Glean. Glean: Work ai for all. Accessed: February 11, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng

Wu. 2025. Crmarena: Understanding the capacity of llm agents to perform professional crm tasks in realistic environments. In *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)*.

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

Feihu Jiang, Chuan Qin, Kaichun Yao, Chuyu Fang, Fuzhen Zhuang, Hengshu Zhu, and Hui Xiong. 2024. Enhancing question answering for enterprise knowledge bases using large language models. In *International Conference on Database Systems for Advanced Applications*, pages 273–290. Springer.

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

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, et al. 2024. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Vivek Khetan, Roshni Ramnani, Mayuresh Anand, Shubhashis Sengupta, and Andrew E Fano. 2020. Causal bert: Language models for causality detection between events expressed in text. *arXiv preprint arXiv:2012.05453*.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joon-suk Park, and Jaewoo Kang. 2023. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 996–1009.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. *Preprint, arXiv:2405.01535*.

Cognition Labs. 2024. [Introducing devin, the first ai software engineer](#). Accessed: February 11, 2025.

LangChain. 2024. What is an ai agent? <https://blog.langchain.dev/what-is-an-agent/>.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, SU Hongjin, ZHAOQING SUO, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. In *The Thirteenth International Conference on Learning Representations*.

Bowen Li, Wenhan Wu, Ziwei Tang, Lin Shi, John Yang, Jinyang Li, Shunyu Yao, Chen Qian, Binyuan Hui, Qicheng Zhang, et al. 2024. Devbench: A comprehensive benchmark for software development. *arXiv preprint arXiv:2403.08604*.

Nian Li, Chen Gao, Yong Li, and Qingmin Liao. 2023. Large language model-empowered agents for simulating macroeconomic activities. *Available at SSRN 4606937*.

Xu Li, Mingming Sun, and Ping Li. 2019. Multi-agent discussion mechanism for natural language generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6096–6103.

Matthieu Lin, Jenny Sheng, Andrew Zhao, Shenzhi Wang, Yang Yue, Yiran Wu, Huan Liu, Jun Liu, Gao Huang, and Yong-Jin Liu. 2024. Llm-based optimization of compound ai systems: A survey. *arXiv preprint arXiv:2410.16392*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, pages 1–11.

Meta. 2024. Large language models: Transforming the future of work. <https://forwork.meta.com/blog/how-large-language-models-are-changing-the-future-of-work/>.

OpenAI. 2024. Openai o1-mini: Advancing cost-efficient reasoning. Accessed: 2025-04-19.

Taryn Plumb. 2025. Here's how google is using llms for complex internal code migrations. <https://www.infoworld.com/article/3804552/heres-how-google-is-using-llms-for-complex-internal-code-migrations.html>.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

James Rounds, Thomas Smith, Lawrence Hubert, Phil Lewis, and David Rivkin. 1999. Development of occupational interest profiles for o* net. *Raleigh, NC: National Center for O* NET Development*, 8.

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

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. 2024. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. *CoRR*.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024a. Travelplanner: A benchmark for real-world planning with language agents. In *International Conference on Machine Learning*, pages 54590–54613. PMLR.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024b. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. 2024a. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*.

Weijie Xu, Zicheng Huang, Wenxiang Hu, Xi Fang, Rakesh Cherukuri, Naumaan Nayyar, Lorenzo Malandri, and Srinivasan Sengamedu. 2024b. Hr-multiwoz: A task oriented dialogue (tod) dataset for hr llm agent. In *Proceedings of the First Workshop on Natural Language Processing for Human Resources (NLP4HR 2024)*, pages 59–72.

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

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Matei Zaharia, Omar Khatab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Franke, Naveen Rao, and Ali Ghodsi. 2024. The shift from models to compound ai systems. <https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/>.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for llm agents: A social psychology view. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. Murre: Multi-hop table retrieval with removal for open-domain text-to-sql. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5789–5806.

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

A Appendix

In this section, we present additional results and analyses that could not be included in the main paper due to space constraints. It also includes visual illustrations of the sandbox environment for EnterpriseBench, and LLM prompts used for benchmark creation and baseline execution. Specifically, this appendix contains the following:

- **Algorithms, Additional Results, and Details**
- **Ablation Study**
- **Expert Study Details**
- **Details of simulating the Enterprise Sandbox**
- **LLM Prompts**

A.1 Additional Results, Algorithm, and Details

Algorithm To generate tasks tailored to individual enterprise employees, we design a pipeline that dynamically incorporates employee context, role-specific goals, and relevant enterprise entities. The process begins by retrieving contextual information based on the employee’s ID and domain of interest, followed by the selection of a suitable goal template. This goal is expanded into subgoals using contextual and entity-aware reasoning. Templates are then populated to construct a task instance, which is iteratively refined and validated using LLM capabilities. The full task generation procedure is detailed in Algorithm 1.

Additional Results Table 6 shows the evaluation of EnterpriseBench using F1 score as the metric across five domains in our benchmark: SWE, Sales, HR, IT, and Business Development. This table allows us to observe the performance of tasks within each domain, which can guide future development of better agents tailored for enterprise settings through separate domain evaluations. Additionally, Table 7 presents the evaluation results using Prometheus-2 with GPT-4 across domains.

Tools Inventory Table 8 presents the collection of tools and functions developed for our EnterpriseBench sandbox environment to support the operation of the LLM Agents.

Post Training Data format We conducted

SFT⁷ and DPO⁸ fine-tuning experiments using the standard dataset formats, illustrated in Listing 1 and Listing 2, respectively.

Listing 1: SFT data format used in training

```

1 {
2   "messages": [
3     {"role": "system", "content": "You
4       are a helpful assistant"},
5     {"role": "user", "content": "What
6       color is the sky?"}
7     {"role": "assistant", "content": "It
8       is blue."}
9   ]
10 }
11 }
```

Listing 2: DPO data format used in training

```

1 {
2   "prompt": [
3     {"role": "user", "content": "What
4       color is the sky?"}
5   ],
6   "chosen": [
7     {"role": "assistant", "content": "It
8       is blue."}
9   ],
10  "rejected": [
11    {"role": "assistant", "content": "It
12      is green."}
13  ]
14 }
```

Benchmark	Coverage	Complexity	Diversity	Expert Validation
SWEBenchmark (Jimenez et al.)	2	0.5	1	✗
WorkArena (Drouin et al.)	7	0.86	6	✗
WorkBench (Styles et al.)	5	0	5	✗
AgentBench (Liu et al.)	8	0	8	✗
τ -bench (Yao et al., 2025)	3	0.67	8	✗
CRM Arena (Huang et al., 2025)	16	1.3	9	✓
EnterpriseBench (Ours)	17	1.2	17	✓

Table 5: Comparison of benchmarks in terms of: coverage (# objects that mirror core components in the simulated environment; ER diagram nodes), environment complexity (# dependencies/object; average connections in ER diagram), and diversity (classification of tasks spread across domains).

Defining the Ground Truth Below, we summarize the step-by-step pipeline used to generate task-specific ground truth in a traceable and verifiable manner:

1. *Retrieve Context*: Relevant data are fetched from pre-defined enterprise sources using employee ID, task domain, and task category.
2. *Extract Entities and Relations*: An LLM is employed to extract (i) entities (e.g., employee, GitHub repository name, issue ID) and (ii) relations (e.g., issues linked to a repository, metadata associated with a repository).

⁷SFT Trainer Data Format

⁸DPO Data Format

Department	LangChain					DSPy				
	GPT-4o	Claude-3.5	o1-mini	LLaMA-3.1-8B	LLaMA-3.1-70B	GPT-4o	Claude-3.5	o1-mini	LLaMA-3.1-8B	LLaMA-3.1-70B
w/o Planning										
SWE	0.29	0.24	0.26	0.03	0.29	0.24	0.20	0.26	0.07	0.28
Sales	0.20	0.26	0.22	0.03	0.25	0.25	0.21	0.25	0.07	0.29
HR	0.42	0.49	0.46	0.03	0.12	0.33	0.20	0.29	0.12	0.33
IT	0.32	0.31	0.32	0.03	0.31	0.32	0.24	0.28	0.13	0.32
Business Ops	0.26	0.37	0.36	0.03	0.32	0.25	0.36	0.27	0.03	0.15
CoT (Wei et al., 2022)										
SWE	0.27	0.23	0.23	0.02	0.22	0.27	0.21	0.29	0.16	0.29
Sales	0.28	0.23	0.27	0.02	0.25	0.27	0.21	0.30	0.06	0.06
HR	0.47	0.34	0.44	0.03	0.30	0.37	0.25	0.35	0.14	0.35
IT	0.35	0.28	0.30	0.03	0.25	0.34	0.27	0.31	0.21	0.31
Business Ops	0.33	0.37	0.31	0.03	0.35	0.35	0.37	0.30	0.03	0.33
ReAct (Yao et al., 2022b)										
SWE	0.27	0.23	0.24	0.02	0.22	0.25	0.20	0.31	0.11	0.31
Sales	0.29	0.23	0.29	0.12	0.25	0.12	0.21	0.32	0.08	0.17
HR	0.48	0.29	0.43	0.14	0.31	0.47	0.30	0.38	0.03	0.40
IT	0.37	0.28	0.28	0.13	0.22	0.24	0.27	0.33	0.19	0.30
Business Ops	0.31	0.39	0.42	0.13	0.19	0.49	0.35	0.37	0.13	0.36
w/ Gold Planning										
SWE	0.36	0.40	0.35	0.14	0.42	0.44	0.38	0.48	0.37	0.42
Sales	0.26	0.27	0.28	0.13	0.27	0.32	0.25	0.44	0.13	0.28
HR	0.48	0.33	0.49	0.14	0.48	0.57	0.41	0.55	0.40	0.53
IT	0.37	0.40	0.38	0.14	0.33	0.43	0.43	0.49	0.31	0.39
Business Ops	0.45	0.56	0.61	0.24	0.34	0.51	0.47	0.51	0.33	0.47

Table 6: **EnterpriseBench Evaluation**: Domain-wise performance comparison using F1 score.

Algorithm 1: Generate Employee-Specific Task

```

1: function GENERATE(emp_id, persona, config, tools, task_domain,
2:   task_category)
3:   context ← GETCONTEXT(emp_id, config["source_paths"],
4:   task_domain, task_category)
5:   goal ← CHOOSEGOAL(config["goal_templates"], task_domain,
6:   task_category)
7:   entities ← ENTITYEXTRACTION(tools, context, goal)
8:   subGoals ← GETSUBGOAL(goal, entities, context)
9:   templates ← GETTEMPLATE(subGoals, entities, context, persona)
10:  task ← GETTASK(goal, subGoals, entities, templates, context, persona)
11:  for i = 1 to max_iter do
12:    if VALIDATE(task) then return task
13:    end if
14:    task ← REPHRASE(task)
15:  end for
16:  return task
17: end function

```

3. *Decompose Goal into Subtask Templates*: The primary task goal is decomposed into logical subtasks using LLMs, guided by domain-specific tools and the retrieved context.
4. *Fill Subtask Templates*: Extracted entities are inserted into subtask templates according to their semantic types.
5. *Ground Each Subtask*: Each subtask is linked to relevant contextual evidence (sentences or snippets) using the identified relations.
6. *Generate Final Task Ground Truth*: All subtasks and their grounded context are combined to form a complete, traceable task-level ground truth.
7. *Validation and Refinement*: The generated ground truth undergoes iterative refinement, after which human experts validate correctness and relevancy.

This process follows a *reverse task synthesis*

paradigm: rather than generating answers to predefined questions, we start from the available context and a goal template. We then frame the most appropriate task whose subgoals and answers are already embedded in the context. This ensures that each task is grounded, domain-relevant, and verifiable.

A.2 Ablation Study

We perform ablation studies to analyze the effect of planning quality, task complexity, and access control on model performance.

Gold Planning As shown in Table 3, providing models with gold plans yields substantial improvements over other planning strategies, highlighting the critical role of accurate planning in complex task execution.

Task Complexity We categorize tasks as *easy* if they contain fewer than three subtasks, and *hard* if they contain three or more. Table 11 compares performance under the ReAct baseline versus our planning-enhanced approach.

The results show a clear drop in performance as task complexity increases (*i.e.*, when the number of subtasks ≥ 3). This ablation reveals that longer interaction trajectories increase failure rates, likely due to the models' lack of prior knowledge of the sandbox environment and limited memory across steps. While planning improves robustness, it does not fully close this gap, highlighting the difficulty of long-horizon reasoning in unfamiliar environments.

Access Control In this ablation, we remove the ac-

Department	LangChain					DSPy				
	GPT-4o	Claude-3.5	o1-mini	LLaMA-3.1-8B	LLaMA-3.1-70B	GPT-4o	Claude-3.5	o1-mini	LLaMA-3.1-8B	LLaMA-3.1-70B
w/o Planning										
SWE	0.30	0.19	0.30	0.02	0.21	0.17	0.21	0.33	0.13	0.24
Sales	0.16	0.20	0.17	0.06	0.14	0.16	0.15	0.21	0.11	0.20
HR	0.36	0.62	0.39	0.05	0.47	0.14	0.13	0.18	0.12	0.18
IT	0.19	0.23	0.33	0.07	0.19	0.15	0.11	0.35	0.10	0.21
Business Ops	0.33	0.30	0.33	0.00	0.15	0.33	0.37	0.36	0.05	0.18
CoT (Wei et al., 2022)										
SWE	0.33	0.23	0.27	0.09	0.20	0.30	0.25	0.30	0.12	0.30
Sales	0.16	0.12	0.20	0.06	0.21	0.16	0.08	0.15	0.02	0.09
HR	0.44	0.40	0.40	0.05	0.40	0.45	0.30	0.48	0.15	0.48
IT	0.31	0.15	0.16	0.04	0.19	0.28	0.18	0.22	0.14	0.30
Business Ops	0.20	0.20	0.38	0.05	0.10	0.40	0.37	0.50	0.05	0.20
ReAct (Yao et al., 2022b)										
SWE	0.34	0.27	0.31	0.15	0.26	0.28	0.22	0.32	0.19	0.37
Sales	0.14	0.16	0.26	0.20	0.15	0.13	0.09	0.20	0.03	0.17
HR	0.44	0.39	0.32	0.09	0.31	0.52	0.46	0.48	0.37	0.54
IT	0.34	0.19	0.27	0.19	0.13	0.22	0.17	0.26	0.21	0.21
Business Ops	0.20	0.41	0.58	0.05	0.23	0.54	0.57	0.65	0.03	0.23
w/ Gold Planning										
SWE	0.50	0.60	0.63	0.13	0.59	0.65	0.62	0.80	0.38	0.63
Sales	0.16	0.21	0.21	0.23	0.13	0.27	0.15	0.20	0.14	0.16
HR	0.54	0.13	0.57	0.19	0.56	0.62	0.68	0.81	0.44	0.67
IT	0.46	0.47	0.53	0.12	0.42	0.58	0.51	0.68	0.50	0.49
Business Ops	0.40	0.50	0.60	0.30	0.33	0.40	0.56	0.62	0.25	0.40

Table 7: **EnterpriseBench Evaluation:** Domain-wise performance comparison with Prometheus-2 using GPT-4 score.

cess control constraint on tasks classified as *Unanswerable*. Without constraints, models attempt these tasks despite lacking the necessary permissions, leading to incorrect executions. These are counted as failures in our evaluation, confirming that access control mechanisms are essential to prevent spurious task completions.

A.3 Expert Study Details

We selected domain experts from various departments within the organization to assist in task evaluation, goal template design, and sandbox environment simulation. For the simulation and goal template creation, we engaged a group of 10 domain experts spanning the target domains. For task evaluation, we ensured relevant participation by circulating a Microsoft Form, requiring that respondents hold job titles aligned with roles defined in EnterpriseBench: Sales, Customer Support, Engineer, IT Support, and HR. Table 14 presents participant profiles involved in sandbox simulation and task validation.

Details of the MS form (screenshots in figure 4 to validate the realism of enterprise data and tasks are provided below:

- *Part-1:* (10 seconds) The participant (or expert) logs in by selecting their department and role (in figure 4a).
- *Part-2:* (15 minutes) After logging in, they are presented with instructions outlining the task: they are asked to assess the realism of the or-

ganizational environment—such as the employee flow chart and access control—and then evaluate the realism of the tasks, which are displayed on the following page (in figure 4b, 4c).

- *Part-3:* (10 minutes/task) On the following pages, elements of the enterprise environment are shown, followed by role-specific tasks—such as emails, chats, and more—tailored to the participant’s selected department and role (see Figure 4d). The user is required to perform these tasks in the sandbox and rate their realism. The participants are asked to rate the realism of environment setup and tasks using below options:
 1. *Very Unrealistic:* The organizational structure and tasks seemed very artificial and didn’t resemble how real organizations typically operate.
 2. *Unrealistic:* While the organization and tasks included some familiar elements, many aspects lacked a convincing or realistic structure.
 3. *Neutral:* The organization and tasks felt partially realistic, combining both plausible and implausible elements.
 4. *Realistic:* The organization largely resembled a real-world setup, and the tasks reflected what an employee might typically ask, though there were minor inconsistencies.
 5. *Very Realistic:* The organization appeared fully authentic, with a structure akin to real-world setups, and the tasks aligned well with

those typically posed by employees.

A.4 Details of simulating the EnterpriseBench Sandbox

In this section, we present the sandbox environment created for EnterpriseBench. To set up an enterprise sandbox, two key components are required: the ER diagram (Figure 5) and the employee hierarchy (Figure 6). The structure of these hierarchies is inspired by CRM Arena (Huang et al., 2025) from Salesforce. The hierarchy was populated based on the requirements of our benchmark, with guidance from domain experts. Building on this foundation, we now describe the statistics and design of the three main components of the sandbox: (a) collection of data sources for building enterprise applications, (b) access control mechanisms, and (c) dynamic operations within the sandbox.

A.4.1 Enterprise Data Simulation

The data simulation process is designed to align with the overall enterprise structure. To ensure authenticity, information was sourced from reliable and verified repositories. We collected relevant data and parsed it to extract key attributes. For example, from product sentiment data, we extracted customer and product information and synchronized it with the sales dataset to maintain consistency across sources. Table 10 provides a detailed overview of the data sources used in EnterpriseBench, including the number of instances and their respective origins. Example instances of enterprise data sources are shown in Figures 7, 8, and 9.

After collecting the data sources, we simulated instances for specific enterprise applications to better represent interconnected enterprise data, as summarized in Table 10. The simulation description is shown below.

Simulated Conversations The conversations generated in EnterpriseBench span various departmental teams, covering a wide range of topics—from *simple inquiries* to *comprehensive discussions about a specific GitHub repository*. These conversations are context-dependent and are designed to closely simulate real-world interactions, following the generation process of the proposed holistic pipeline. Figure 7 presents an example of a chat between two employees, Steve and John, from the engineering department, based on the GitHub repository maintained by Steve.

Simulated Customer Support Chat The customer

support conversations are generated based on product sentiment data. Persona-based interactions subjects are created by incorporating details of both the customer and a sales representative (employee from sales department). These interactions simulate a conversation where the representative responds to the customer's sentiment by proposing a potential solution to resolve the issue. Figure 8 illustrates an example of such a conversation between a customer and a sales representative.

Simulated Enterprise Mail System The email simulations are generated based on threaded conversations, where each email exchange belongs to a specific thread. Within a thread, multiple messages are exchanged between the sender and recipient, maintaining continuity and context. Figure 9 presents an example of an email thread between two employees from the HR department.

A.4.2 EnterpriseBench Security Layer Details

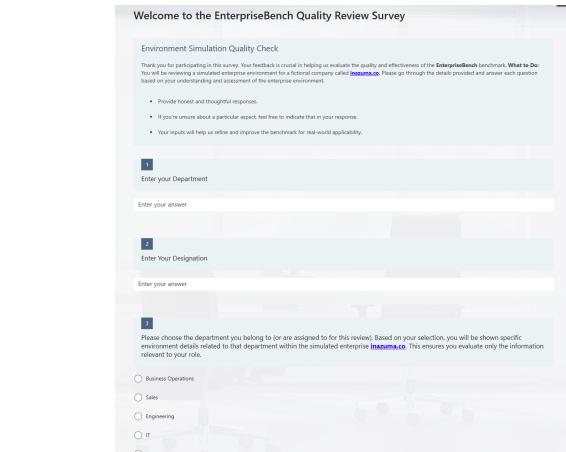
In enterprise environments, ensuring secure and regulated data access is critical. The Access Control Layer plays a fundamental role in enforcing access policies and preventing unauthorized data access. Our work, EnterpriseBench, implements a structured approach by integrating access control rules in a JSON format for each data source. A LLM Agent is responsible for verifying access permissions based on an employee's credentials and the requested data.

Access Verification Mechanism The Access Control Layer operates in conjunction with the retrieval process. When a query is processed, the retriever first gathers relevant contextual data. Before the information is presented to the user, it is passed through the Access Control Layer, where all inaccessible content is filtered out based on pre-defined rules.

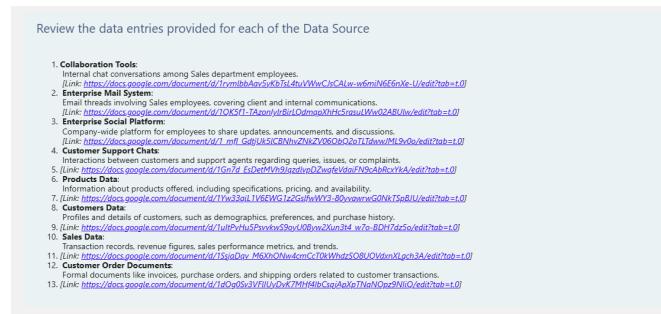
For instance, as illustrated in Figure 10, the access control rules dictate that a GitHub repository is accessible only to its owner and senior employees within the organizational hierarchy. If an employee from a different department, or even from the same department but with an `emp_id` different from the `repo_owner_id`, attempts to access the repository, the agent will respond with "Access Denied." Furthermore, if an employee at the same level attempts to perform a task requiring edit access to the repository, the agent will revoke the request, ensuring strict compliance with access policies.

Dynamic and Customizable Access Control

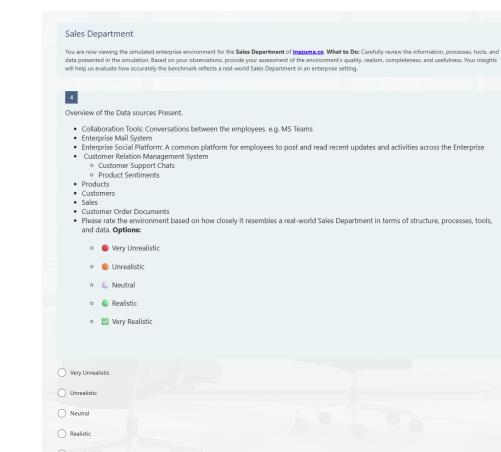
The Access Control Layer is designed to be flex-



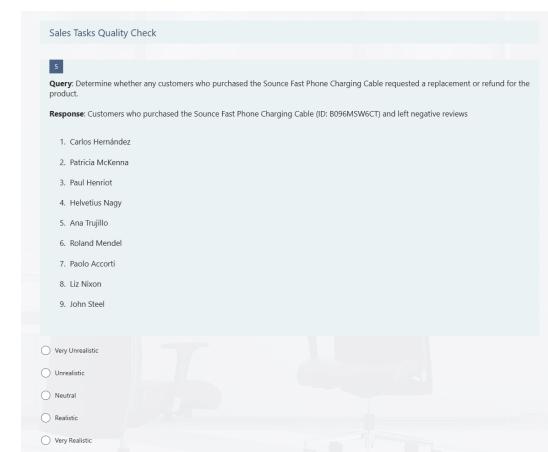
(a) First page of the Microsoft Form used to collect information about domain experts, including their department and position.



(b) Next page of the form displaying simulated data details for the selected department. This example shows sales data from the enterprise.



(c) Users are asked to rate the realism of the simulated data for the selected department, choosing from options ranging from 'Very Unrealistic' to 'Very Realistic.' They also have to provide reasons when selecting 'Unrealistic'.



(d) This page presents enterprise tasks for evaluation. Users rate each task's realism from 'Very Unrealistic' to 'Very Realistic,' and provide reasons if they select 'Neutral,' 'Unrealistic,' or 'Very Unrealistic'.

Figure 4: Domain Expert Validation in EnterpriseBench. Domain experts from all benchmark domains evaluate the realism of the generated data and created tasks. This example shows screenshots of MS form for different steps a domain expert completes during the validation process.

ible, allowing dynamic modification of access rules. This adaptability enables organizations to customize security policies according to evolving requirements while ensuring robust data protection. By maintaining granular control over data accessibility, this framework enhances security and compliance within enterprise systems.

A.4.3 Data Dynamics Operations

Data Dynamism is enabled in EnterpriseBench by allowing agents to autonomously perform CRUD operations across diverse enterprise data sources, allowing for real-time changes and interactions. By orchestrating task decomposition, access control, and data dynamism, we ensure the system is capable of handling evolving business needs, fostering enhanced operational efficiency and informed decision-making across enterprise. Below, we present the data dynamism pipeline along with pseudocode, and illustrate it using a GitHub-based example.

Data Dynamism Pipeline

```
from llmCrudOps import EnggConvCRUD
from llmCrudOps import GitHubCRUD
from llmCrudOps import GitIssuesCRUD
...

class DataDynamismPipeline:
    def __init__(self, llm):
        self.llm = AzureChatOpenAI(llm)

    def fetch_crud_control(...):
        # Returns CRUD controller for selected data source
        return control

    def run_CAI_pipeline(user_persona, user_query):
        # 1. Break down Primary Tasks into Subtasks
        prompt_CoT=ChatPromptTemplate.from_messages
        task_breakdown = prompt_CoT | self.llm
        generated_subtasks = chain_task_breakdown.invoke(...)

        for subtask in generated_subtasks:
            # 2. Determine Data Source
            prompt_ds=ChatPromptTemplate(...)
            chain_data_source = prompt_ds | self.llm
            selected_data_sources_str = chain_data_source.
                invoke(...)

            # 3. Determine Function and Parameters
            prompt_fn=ChatPromptTemplate(...)
            chain_function = prompt_fn | self.llm
            selected_function_name_str = chain_function.
                invoke(...)

            # 4. Check Access Permissions
            for function_name in selected_function_name_list:
                prompt_acc=ChatPromptTemplate(...)
                chain_access = prompt_acc | self.llm
                access_status = chain_access.invoke(...)

                # If Allowed, Execute CRUD Operation and
                # Return Response

                if access_status == "Allowed":
                    control = self.fetch_crud_control()

                    if function_name -> read:
                        result = control.read(*params)
                    elif function_name -> create:
                        result = control.create(*params)
                    elif function_name -> update:
                        result = control.update(*params)
                    elif function_name -> delete:
                        result = control.delete(*params)

                    return responses
```

```
elif function_name -> delete:
    result = control.delete(*params)

return responses
```

GitHub CRUD Script

```
from accesscontrol import GitHubAccess

class GitHubCRUD:
    def __init__(self, employees_csv_path, code_json_path):
        self.access = access_control
        self.employees_df = ...
        self.code_data = ...
        self.code_json_path = ...

    def read(self, emp_id, path):
        """Reads GitHub code."""
        check -> access.is_valid_employee(emp_id)

        if (access.path_exists(...) and
            (access.is_owner(...) or
            access.is_engg_lvl_10_or_above(...) or
            access.is_cto_or_lvl_14(...))):
            for entry in self.code_data:
                if entry["path"] == path:
                    return entry
            print("Error: Code not found.")
        else:
            print("Error: Access denied.")

    def create(repo_name, emp_id, path, ...):
        """Creates a new GitHub code entry."""
        ...

    def update(self, emp_id, path, content, ...):
        """Updates an existing GitHub code entry."""
        check -> access.path_exists(...)
        check -> access.is_valid_employee(...)

        if (access.is_owner(...) or
            access.is_engg_lvl_10_or_above(...) or
            access.is_cto_or_lvl_14(...)):
            for entry in self.code_data:
                if entry["path"] == path:
                    # update entry
                    print("Error: Code not found.")
            else:
                print("Error: Access denied for update.")

    def delete(self, emp_id, path):
        """Deletes a GitHub code entry."""
        ...
```

GitHub Access Check

```
class GitHubAccess:
    def __init__(self, employees_csv_path, code_json_path):
        self.employees_df = ...
        self.code_data = ...
        self.code_json_path = ...

    def path_exists(self, path, code_json_path) -> bool:
        """Checks if the GitHub code path exists."""
        ...

    def is_valid_employee(self, emp_id) -> bool:
        """Checks if the employee ID exists and is valid."""
        ...

    def is_owner(self, path, emp_id) -> bool:
        """Checks if the employee is the owner of the code
        path."""
        ...

    def is_engineer_lvl_10_or_above(self, emp_id) -> bool:
        """Checks if the employee is an Engineer with level
        >= 10."""
        ...

    def is_cto_or_lvl_14(self, emp_id) -> bool:
        """Checks if the employee is a CTO with level 14."""
        ...
```

Table 8: List of Apps, Tools, and their Descriptions

App	Tool	Description
HR System	employee_data_read	Reads or Fetches the employee record based on emp_id, name, email or semantic query (vector DB).
HR System	employee_data_create	Creates a new employee record.
HR System	employee_data_update	Updates an existing employee record.
HR System	employee_data_delete	Deletes an employee record (sets is_valid to False).
Enterprise Mail System	enterprise_mail_system_read	Reads email data based on sender's email, recipient's email, optional thread ID, and optional email ID or semantic query (vector DB). Retrieves user-specific emails, emails in a thread, or a specific conversation while enforcing access control.
Enterprise Mail System	enterprise_mail_system_create	Creates a new email conversation. Generates unique email and thread IDs, validates participants, and stores the email in the database.
Enterprise Mail System	enterprise_mail_system_update	Updates an existing email conversation by modifying the subject, body, importance, category, signature, or confidentiality notice. Ensures user authorization before making changes.
Enterprise Mail System	enterprise_mail_system_delete	Deletes an email conversation based on user ID, thread ID, and email ID. Ensures the user is a participant and has permission before removing the email.
Chats	conversations_create	Creates a new conversation entry in a particular software engineer team between emp1 and emp2 if the conversation ID does not already exist.
Chats	conversations_read	Reads and displays a conversation between emp1 and emp2 based on their emp_id's, conversation_id or semantic query (vector DB). Only authorized employees either emp1 or emp2 can access the conversation.
Chats	conversations_update	Updates an existing conversation entry if the conversation exists between emp1 and emp2 of software engineer team.
Chats	conversations_delete	Deletes an existing conversation entry if the conversation exists between emp1 and emp2 of software engineer team, the employee is authorized to delete it, and the repository is valid.
Workspace	github_read	Reads and displays the GitHub repository along with their issues based on specific path, repo_name or semantic query (vector DB) or fetches all repositories accessible to an employee. Ensures access control before retrieving data.
Workspace	github_create	Creates a new GitHub code entry. Ensures the path is unique, the employee is valid, and generates a hash for content integrity.
Workspace	github_update	Updates an existing GitHub code entry. Ensures the path exists, the employee has sufficient access rights, and updates content with a new hash if modified.
Workspace	github_delete	Deletes a GitHub code entry. Ensures the employee has the appropriate access rights and removes the code entry if permitted.
Sales	products_create	Creates a new product entry if the product ID doesn't already exist, and the employee has the required access level.
Sales	products_read	Reads and displays product details based on product_id or semantic query (vector DB) if the employee has the required access level and the product ID exists.
Sales	products_update	Updates an existing product entry if the product ID exists, and the employee has the required access level. Displays the product details before and after the update.
Sales	products_delete	Deletes an existing product entry if the product ID exists, and the employee has the required access level. Confirms deletion by attempting to read the product after deletion.
Sales	product_sentiment_create	Creates a new product sentiment entry if the product ID, customer ID, and employee have the necessary access rights.
Sales	product_sentiment_read	Reads and displays the product sentiment data based on product ID, customer ID or semantic query (vector DB) and employee access rights.
Sales	product_sentiment_update	Updates an existing product sentiment entry if the product ID, customer ID, and employee have the necessary access rights.

App	Tool	Description
Sales	product_sentiment_delete	Deletes a product sentiment entry if the product ID, customer ID, and employee have the necessary access rights.
Sales	sales_create	Creates a new product entry if the product does not already exist and the employee has the required permissions.
Sales	sales_read	Reads and displays product sales details based on the product ID, product_name, customer_id, customer_name, data_of_purchase or semantic query (vector DB).
Sales	sales_update	Updates an existing product sales entry if the product exists, the employee is authorized, and the required permissions are met.
Sales	sales_delete	Deletes an existing product sales entry based on access rules and authorization. Ensures the product exists and the employee has the necessary permissions.
Sales	customer_support_chats_create	Creates a new support entry with employee ID, product ID, customer ID, text, and an optional interaction date. Validates inputs before appending data to the support log.
Sales	customer_support_chats_read	Reads customer support chats between employee and customer based on employee ID, product ID, customer ID or semantic query (vector DB). Ensures employee validation and access control before retrieving records.
Sales	customer_support_chats_update	Updates an existing support record with new text and an interaction date. Ensures the entry exists and the employee has necessary permissions before modifying data.
Sales	customer_support_chats_delete	Deletes a support entry based on employee ID, product ID, and customer ID. Checks access control and validates inputs before removing the record.
IT Solutions	it_service_management_create_issue	Creates a new IT helpdesk issue if the employee has access and the issue ID doesn't exist.
IT Solutions	it_service_management_read_issue	Reads and displays a IT helpdesk issue that is raised by an employee with id raised_by_emp_id and handled by IT department employee with employee ID emp_id or semantic query (vector DB) if the employee has access.
IT Solutions	it_service_management_update_issue	Updates an existing helpdesk issue if the employee has access.
IT Solutions	it_service_management_delete_issue	Deletes a helpdesk issue if the employee has access.
Social Platform	social_platform_create	Creates a new post in the clubbed posts JSON.
Social Platform	social_platform_read	Reads a post from the clubbed posts JSON.
Social Platform	social_platform_update	Updates an existing post in the clubbed posts JSON.
Social Platform	social_platform_delete	Deletes a post from the clubbed posts JSON.
Internal Overflow	overflow_read	Reads and displays an overflow post if the user has access.
Internal Overflow	overflow_create	Creates a new overflow post if the user is a valid employee and the post ID is unique.
Internal Overflow	overflow_update	Updates an existing overflow post if the user is the owner and a valid employee.
Internal Overflow	overflow_delete	Deletes an overflow post if the user is the owner and a valid employee.
Enterprise Policy Documents	document_read	Retrieves the relevant data from the policy document pds using ColPali
LLM Call	llm_call	General purpose llm call

Domain	Goal Template
Business Operation	Send an email about scheduling a meeting
Business Operation	Send an email about Performance Management
HR	Send an email requesting for leaves
HR	Update the Salary of employee [Employee ID] to \$150000
Sales	Get the details of [Product ID or Product Name] (id, name, price, description) with the most reviews from customers.
Sales	Get the sentiment (positive/negative/neutral) from the customer's review content for [Product ID or Product Name].
SWE	Update the metadata of a particular repository
SWE	List the names of all the GitHub repositories owned by [Employee ID]
IT	Get all my tickets on [date] with 'high' priority, give ID and list them
IT	Send a message about new Asset Configurations

Table 9: Domain experts curated task goal templates for the EnterpriseBench task curation, organized by domain.

Data Source	Data Source Elements	Data Formats	Collected/Generated	# Instances	Data Origin	Collected Source Link
Collaboration Tools (Chats)	HR Business Development Sales Management IT SDE	JSON	Generated	500 500 500 500 500 500	Employees.csv + GitHub Code + Policy Documents + Sales + etc.	-
Customer Relation Management	Customer Support Chats Product Sentiments Customers Customer Orders Products Sales	JSON JSON JSON PDF JSON JSON	Generated Collected Collected Generated Collected Collected	1000 13500 832 832 1352 13511	Product Sentiments + Customer.csv + Employees + Sales	Product Sentiments Customers (Extracted from Product sentiments)
Policy Documents	Policy Documents	PDF	Collected	24	-	Documents Collected from Google Datasets
Enterprise Mail System	HR Finance Sales Management IT SDE Other Dept Emails	JSON	Generated	7000	Employees.json + GitHub Code + Policy Documents + Sales + etc.	-
Enterprise Social Platform	Tech Crunch Posts (Social Platform)	JSON	Collected	39115	-	Tech Crunch Posts
Business and Management	Clients Vendors	JSON JSON	Generated Generated	400 400	-	Open Source Business Datasets
HR Management	Employees Resumes Roles	JSON PDF PDF	Collected Generated Generated	1265 1265 1 * 32	Employees.csv	LinkedIn Profiles (Ayoobi et al., 2023)
Enterprise Overflow	Technical Posts (like StackOverflow)	JSON	Collected	8398 Posts	-	Stack Overflow Posts
IT Service Management	IT Tickets	JSON	Collected	163Tickets	-	Help Desk Tickets
Workspace	GitHub Repository	JSON	Collected	29241 957	GitHub + Employees.json	GitHub Code

Table 10: **Overview of Data Sources used in the EnterpriseBench sandbox.** The table includes data domains, elements, formats, whether the data was collected or generated, number of instances, data origin, and links to public sources (if applicable).

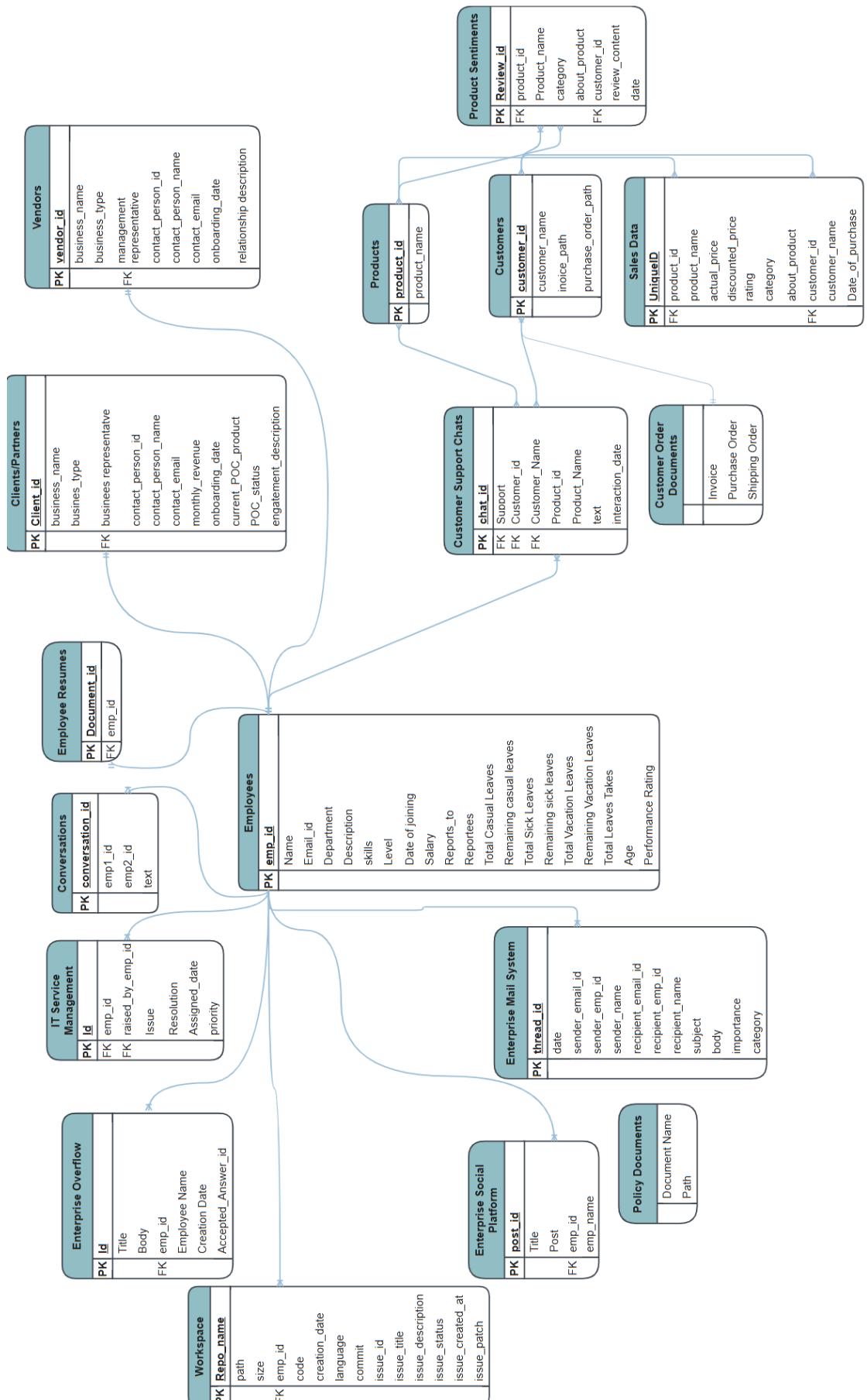


Figure 5: Expert-curated ER diagram for the EnterpriseBench sandbox

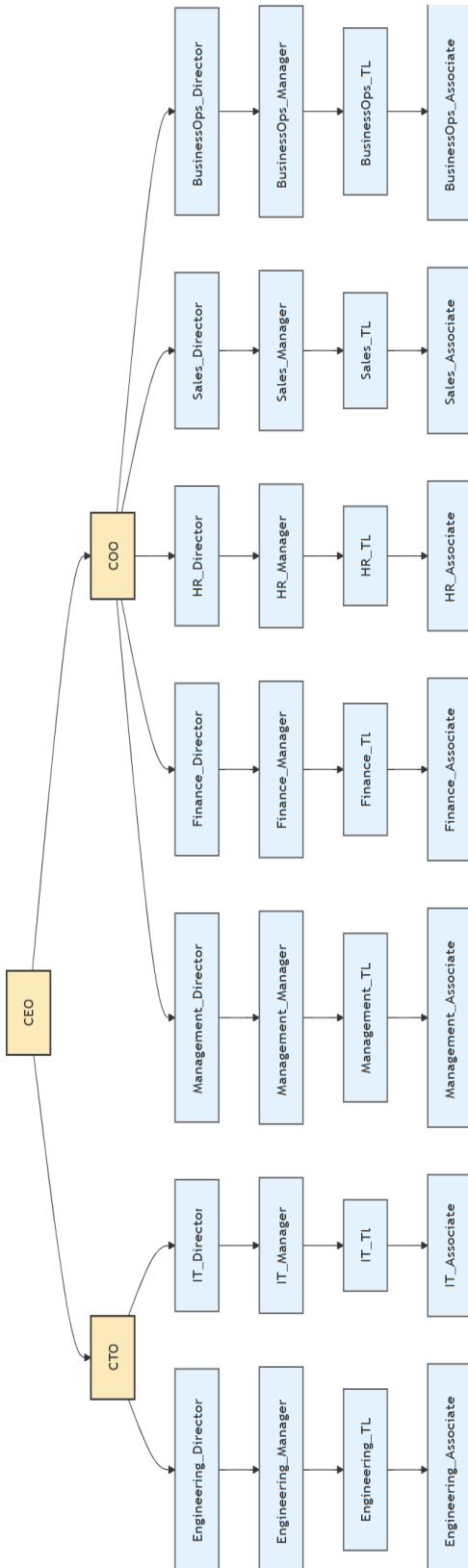


Figure 6: Expert-curated employee hierarchy for the EnterpriseBench sandbox

Model	Complexity	ReAct	w/ Planning
GPT-4o w/ LangChain	Easy	0.39	0.61
	Hard	0.21	0.35
o1-mini w/ LangChain	Easy	0.44	0.67
	Hard	0.26	0.39

Table 11: Performance comparison of models with ReAct vs. planning, grouped by task complexity.

Model Name	Model ID	Provider	Region	Temp.	Max Tokens	Max Retries
Claude 3.5 Sonnet	anthropic.claude-3-5-sonnet-20240620-v1.0	Amazon Bedrock	us-east-2	0	Default	-
LLaMA 3.8B Instruct	meta.llama3-1.8b-instruct-v1.0	Amazon Bedrock	us-west-2	0	Default	-
LLaMA 3.70B Instruct	meta.llama3-3-70b-instruct-v1.0	Amazon Bedrock	us-east-2	0	Default	-
GPT-4o	gpt-4o	Azure OpenAI	-	0	Default	2
o1-mini	o1-mini	Azure OpenAI	-	0	25,000	2

Table 12: Hyperparameter settings for API calls.

Model	Top-K	Truncation	Padding	Similarity Metric
vidore/colpali	1	-	-	Default
ColBERT	5	True	True	Cosine Similarity

Table 13: Retriever configurations for similarity-based search.

Profession	Gender	Age
Software Engineer	M	25
Senior Engineer	F	29
Sales Development Representative	M	28
Sales Manager	F	35
IT Engineer	M	29
Technical Assistant	M	32
HR Head	F	40
Lead HR	F	30
Finance Associate	M	23
Finance Manager	M	40

Table 14: **Domain Experts Information:** Profession, Gender, and Age Information



Figure 7: Example from the EnterpriseBench sandbox: Collaboration Tools chat between 2 employees of an engineering department

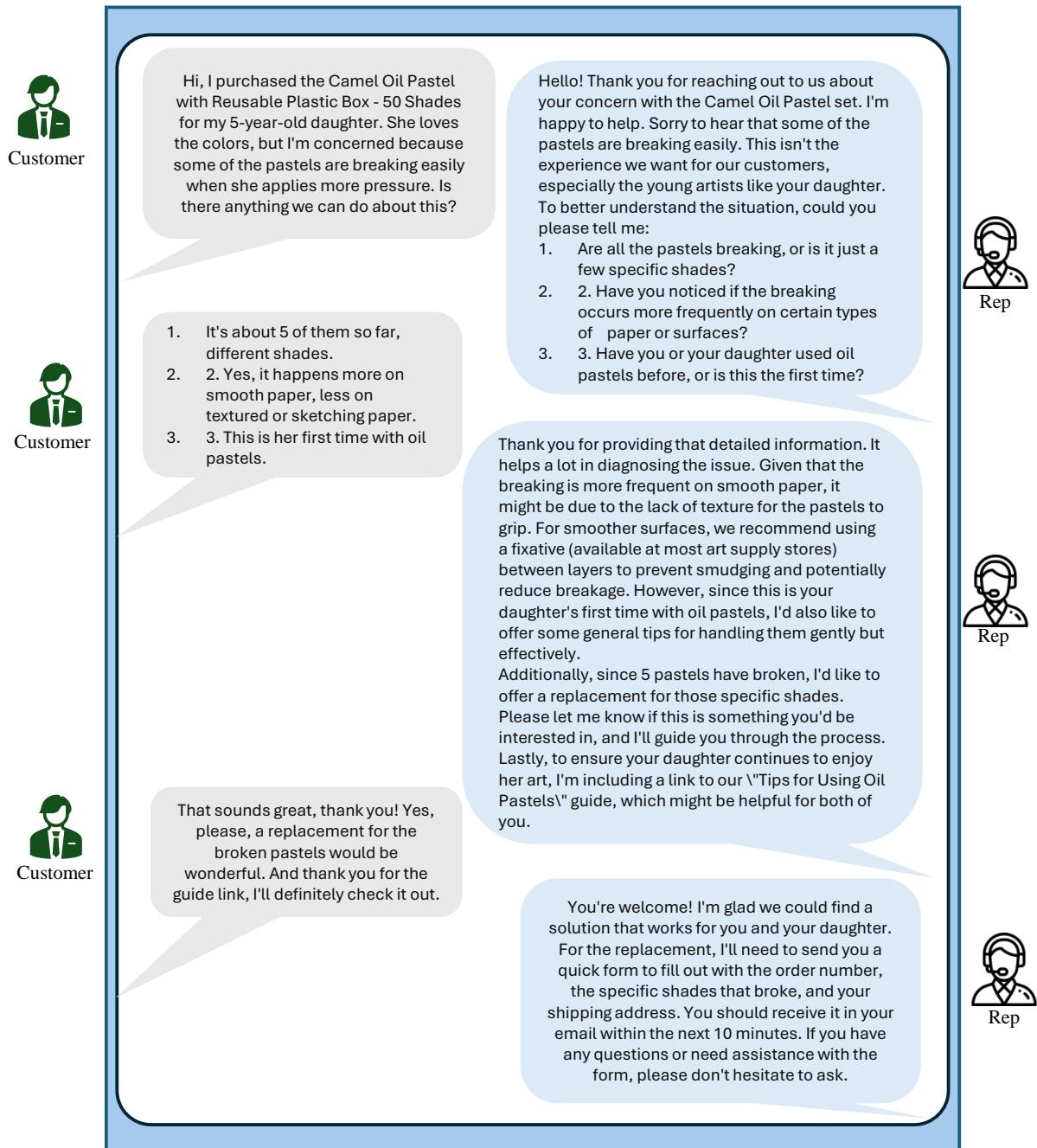


Figure 8: Example from the EnterpriseBench sandbox: Customer Support Chat between a customer and sales representative

<p>Date: 2012-03-18 06:58:29 IST Sender: aji.joseph@inazuma.com, Recipient: deepa.gangadharan@inazuma.com Subject: Re: HR Synergy: Discussing Cross-Departmental Goals for Upcoming Quarterly Reviews Body: Dear Deepa I hope this email finds you well. As we approach the upcoming quarterly reviews, I wanted to touch base with you regarding the HR synergy project. As you know, our department's goals are closely aligned with the company's overall objectives. I'd like to schedule a meeting to discuss how we can work together to achieve these goals. Please find attached the draft goals document for your review. I've highlighted the key areas where we need your input. Looking forward to hearing from you soon. Importance: High Category: INTERNAL Signature: Aji Joseph, HR Generalist, Inazuma Corporation, Bangalore Confidentiality Notice: This email contains confidential information intended only for the use of the individual or entity named above. Any unauthorized use, disclosure, or distribution is strictly prohibited.</p>
<p>Date: 2021-03-21 22:52:11 IST Sender: deepa.gangadharan@inazuma.com Recipient: aji.joseph@inazuma.com Subject: Re: HR Synergy: Discussing Cross-Departmental Goals for Upcoming Quarterly Reviews Body: Hi Aji, Thanks for reaching out and sharing the draft goals document. I've reviewed it and have some suggestions for the 'Employee Engagement' section. I've attached my feedback document for your reference. Would it be possible to schedule the meeting on next Wednesday at 2 PM? I've also invited our team lead, Rohan, to join the discussion. Looking forward to your confirmation. Importance: Normal, Category: FOLLOW-UP Signature: Deepa Gangadharan, HR Manager, Inazuma Corporation, Bangalore Confidentiality notice: This email contains confidential information intended only for the use of the individual or entity named above. Any unauthorized use, disclosure, or distribution is strictly prohibited.</p>
<p>Date: 2021-03-22 14:52:11 IST Sender: aji.joseph@inazuma.com Recipient: deepa.gangadharan@inazuma.com Subject: Re: HR Synergy: Discussing Cross-Departmental Goals for Upcoming Quarterly Reviews Body: Hi Deepa, Thanks for your prompt response and valuable feedback. I've reviewed your suggestions and incorporated them into the revised goals document (attached). I've also confirmed the meeting with Rohan for next Wednesday at 2 PM. Please note that I've marked the 'Employee Retention' section as 'High Priority' in the revised document. I'd like to discuss this further during the meeting. See you both on Wednesday.. Importance: High, Category: FOLLOW-UP Signature: Aji Joseph, HR Generalist, Inazuma Corporation, Bangalore Confidentiality notice: This email contains confidential information intended only for the use of the individual or entity named above. Any unauthorized use, disclosure, or distribution is strictly prohibited.</p>

Figure 9: Example from the EnterpriseBench sandbox: Mail delivery between an employee and HR

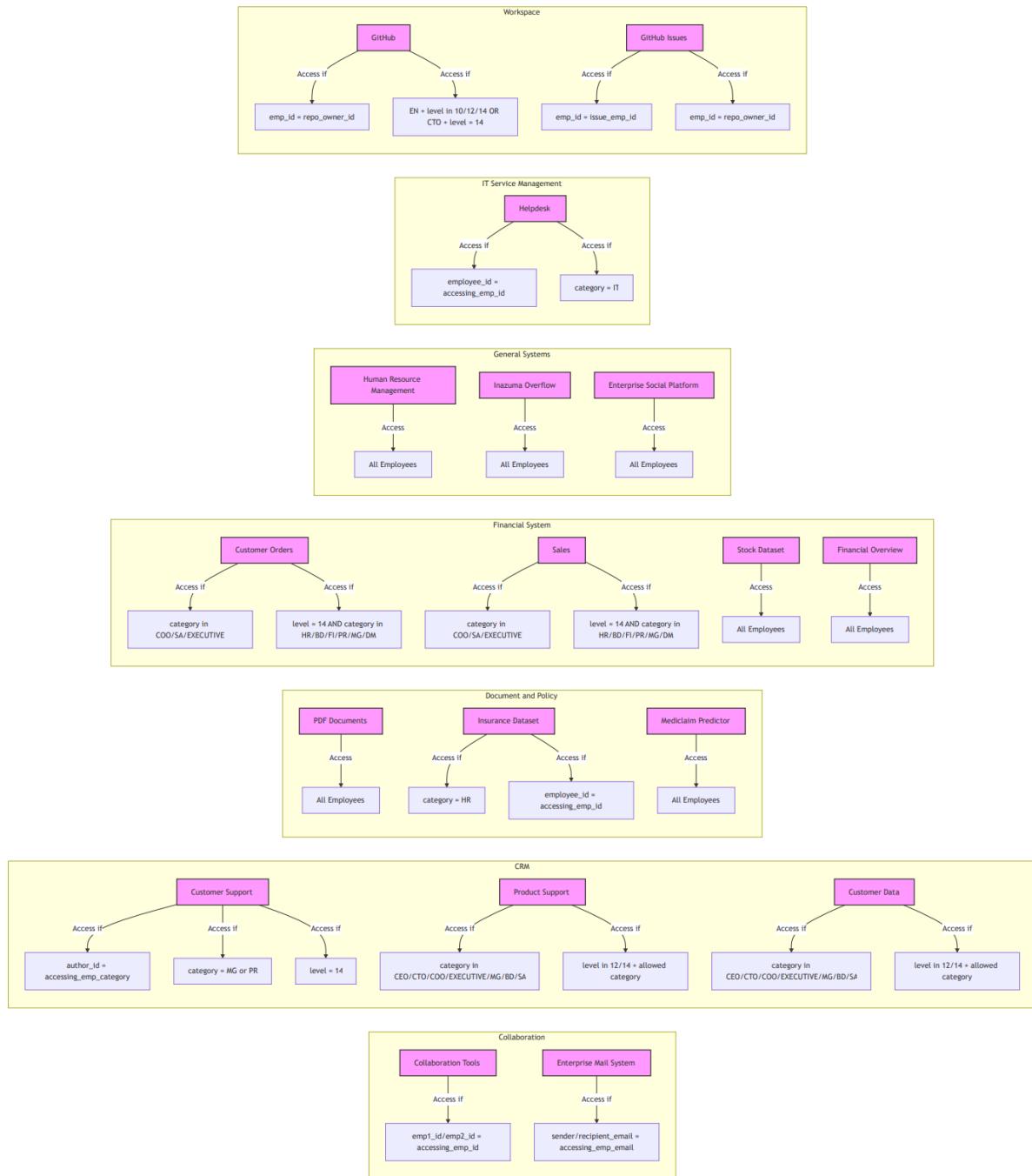


Figure 10: Access Control Design for the EnterpriseBench Sandbox

A.5 LLM Prompts

Below are the prompts used for LLM-based generation. These prompts were initially created using a system prompt and then refined through human intervention.

A.5.1 Prompts for Task Generation

Entity Extraction - Filter out entities by inference tools on context

You are a Tool Dependency Inference Agent.

Input:

```
context={context}  
tools={tools}
```

Instructions

You are given a natural language ***context*** describing what the user wants to analyze or understand, along with a list of available tools. Your job is to ***map the context to the functionality of the tools***, and describe the expected outputs if each tool were applied to this context.

- DO NOT generate verbose or overly synthetic descriptions.
- GENERATE THE FULL OUTPUT.
- Instead, for each tool, generate what the output ****should look like**** (in tabular format, dictionary, or concise key points) when used for the provided context.
- Generate the expected output of each tool given in ‘tools’.
- Make sure your response is grounded in the tool descriptions. Do NOT make assumptions beyond the capabilities defined in each tool.

Output Format:

Return a JSON structure with keys as tool names and values as the expected output structures when each tool is called on the provided context.

Examples:

.....

Subgoal Decomposition - Create plan to execute the task

You are a SubGoal Generating Agent that decomposes a high-level goal into smaller, actionable subgoals.

```
# Input:  
data_chain={data_chain}  
primary_goal={primary_goal}  
tool_inference={tool_inference}  
context={context}
```

Instructions

Your task is to break down the given primary goal into granular subgoals. Each subgoal must:

- Stay strictly aligned with the original primary goal.
- Don't assume context is direct input; first subgoal(s) must extract it using tools IF NECESSARY, with arguments derived from the primary_goal (note: emp_id is always given, for product ID and Customer ID fetch details if not mentioned in primary_goal).
- Map to **one** tool from the provided tool_inference.
- Operate on a **single data source** from the data_chain.
- Optionally use the output from the previous subgoal to enable layered analysis.
- Avoid redundancy or synthetic-sounding tasks.

DO NOT include subgoals that are irrelevant or too broad.

DO NOT combine multiple tools in a single subgoal.

Output Format:

Return a JSON with a list of precise subgoals required to achieve the primary goal.

Examples:

.....

Task Template Generation - Create Task Template for each Subgoal

You are a Task Template Generating Agent.

```
# Input:  
subgoals={subgoals}  
tool_inference={tool_inference}  
context={context}
```

Instructions

Your goal is to convert each subgoal into a natural-sounding question template that:

- Could realistically be asked by a Domain expert with details {persona} in an enterprise setting.
- Matches the intent of the subgoal.
- Uses placeholders (e.g., <product>, <issue>, <device>) that correspond to elements in the provided context.
- Is answerable using the tool mentioned in the corresponding tool dependency.

- Use first-person phrasing as if the question is being asked directly.
- Ensure the question naturally follows from the subgoal and reflects its purpose.
- Each question must map clearly to one tool dependency.
- Do not fabricate placeholders – derive them from the context.
- Avoid overly synthetic or robotic phrasing.

Output Format:

Return a list of question templates, one for each subgoal.

Examples:

.....

Final Task Generation - Create Final Task with Ground Truth

You are a Human Domain Expert tasked with curating high-quality, structured business tasks that reflect realistic analytical workflows.

Input:

```
persona={persona}
data_chain={data_chain}
primary_goal={primary_goal}
subgoals={subgoals}
tool_inference={tool_inference}
templates={templates}
Ground Truth Context={context}
```

Instructions

Your objective is to generate a well-structured, domain-relevant task and its subtasks grounded in business context, using the knowledge of a human domain expert.

Step 1: Create a single task

- Use the ‘primary_goal’ to formulate a task that reflects a realistic and meaningful question a **domain expert in the role described by ‘persona’** would ask.
- Don’t change the intent of the primary goal.
- Just rephrase the primary goal into a task specific question.
- Don’t add SUMMARIZE, ANALYZE etc. for primary goal, stick to the goal only
- Avoid generic phrasing or overly robotic structure.
- The tone should be informed, goal-directed, and tailored to the needs and context of the given persona.
- The reasoning behind the task should reflect the depth, clarity, and practicality that a professional in this domain would apply.
- Don’t synthetically include terms that changes the primary_goal

Step 2: Create subtasks

For each subgoal:

- Select the corresponding template from ‘templates’.
- Ground the template into a **realistic subtask question** by replacing

placeholders using only the ‘context’.

- Identify the appropriate tool from ‘tool_inference’.
- Select the most relevant stage from the ‘data_chain’ that helps accomplish this subgoal.
- Derive a **closed-form ‘subtask_ground_truth’** from the ‘context’. The answer should reflect how a **human domain expert** would interpret and explain the outcome.
- Provide a ‘thinking_trace’ that explains:
> “To answer this subgoal, we need to apply <tool_dependency> on <selected_stage_from_data_chain> to extract <target insight>.”
- Return the following fields for each subtask:
- ‘subgoal’
- ‘question’ (grounded from template)
- ‘subtask_ground_truth’ (explicit, closed-form, and written in the tone of a human domain expert)
- ‘thinking_trace’
- ‘data source’

Step 3: Add subtask dependency graph

- Define the chronological or logical dependencies among subtasks as a directed graph only when the thinking_trace or subtask_ground_truth of current subtask depends upon earlier subtask(s) (e.g., 1→2; 1,3→4).
- Ensure the ordering reflects realistic reasoning flow by a domain expert.

Step 4: Create final ground truth

- create the final ‘ground_truth’ from Ground Truth Context.
- The final answer should be constructive, precise, and demonstrate how a human domain expert would explain the conclusion using insights from each subtask.
- Avoid ambiguity. Focus on actionable insights and detailed, contextual explanations.
- Clearly emphasize any quantitative figures, key features, or specific stats present in the subtask_ground_truth.

Guidelines

- Do **not** invent entities or facts—rely only on what’s grounded in the ‘Ground Truth Context’.
- Tailor the language and tone to match the persona’s domain knowledge and role.
- Do not reuse the word “analyze” for every subtask—use domain-specific verbs and phrasing.
- Prioritize realism, precision, and interpretability in every output.
- Treat this as if you are preparing tasks and answers for executive review or expert-level decision-making.
- Do not include these kinds of statements in subtasks (// Additional subtasks would follow similar structure for the remaining repositories), include all, don’t do incomplete generations.

Examples:

.....

Validation - validate the task across quality checklist

You are an expert evaluator for employee-specific task generation quality. Your job is to evaluate whether a generated task and ground truth meet strict enterprise-grade standards for clarity, precision, realism, and privacy adherence.

Inputs

```
## Context Information:  
{formatted_context}  
## Generated Task:  
- Task: {current_task["task"]}  
- Ground Truth: {current_task["ground_truth"]}  
## Evaluation Criteria:
```

Please evaluate the following seven questions and provide a YES or NO answer for each, along with a detailed explanation. Each question enforces a specific quality standard for secure, grounded, and useful employee-specific task generation.

1. **Is the task meaningfully aligned with the context content [e.g., employee interactions, performance logs, internal communications, technical contributions]?**
 - The task should logically emerge from the provided context and reflect personal, employee-specific insights.
 - A good task references themes like appraisals, IT tickets, communication patterns, or personal contributions.
 - A bad task is disconnected, overly general, or focused on organization-wide information the employee shouldn't access.
2. **Is the task specific, closed-ended, and fully grounded in the context without requiring speculation?**
 - A good task is answerable based only on the provided context and targets a precise insight.
 - A bad task is vague, speculative, or assumes external knowledge not represented in the context.
3. **Does the task avoid referencing or implying the presence of context (e.g., "according to my data," "based on the logs," or "from the context")?**
 - A well-formed task should feel natural and conversational, as if the employee is asking a smart assistant.
 - It should never acknowledge that it's grounded in a hidden data layer.
4. **Does the ground truth directly and clearly answer the task using only context-derived facts, without ambiguity or open-endedness?**
 - A valid ground truth gives a complete, unambiguous response aligned with the task.
 - It must resolve the question without hedging or deferring.
5. **Does the ground truth avoid meta-references (e.g., "the system says," "based on the data," "the context shows")?**
 - A high-quality answer should stand on its own, like a statement from a knowledgeable system or assistant.
 - Meta-language reduces clarity and realism.

6. **Does the task and ground truth resemble a realistic exchange between an enterprise employee and a smart assistant?**
 - The language should reflect enterprise professionalism.
 - The task must be in first person (e.g., “Can I see,” “Do I have,” “Is there any update on my...”).
 - The tone should reflect how a real employee interacts with a system.
7. **Does the task respect employee privacy boundaries and reflect insights that only the employee is authorized to access?**
 - Tasks should refer only to the requesting employee’s data (e.g., my performance, my GitHub contributions).
 - It must not request information about peers, teams, or enterprise-wide metrics unless explicitly scoped to the individual.

```
## Format your response exactly as follows:
```
"json
"question1": <
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"question2": <
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"question3": <
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"question4":<
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"question5":<
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"question6":<
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"question7":<
"answer": "YES/NO",
"explanation": "Your detailed explanation here"
>,
"overall_pass": true/false
```

```

Improvement - improve the task for ambiguity

You are an expert in rephrasing and improving enterprise-grade employee-specific tasks for internal smart assistants. Your job is to revise a task and ground truth that failed strict enterprise validation.

Context Information: {formatted_context}

Expert who curated this task:

- Persona: {persona}

Current Task:

- Task: {current_task["task"]}
- Ground Truth: {current_task["ground_truth"]}

Evaluation Results:

```
{json.dumps(evaluation, indent=2)}
```

Instructions: You must revise the task and/or the ground truth to address specific weaknesses and ensure the highest standards for clarity, realism, and employee privacy.

Task Revisions:

- If question1, question2, or question4 is marked NO:
 - Rewrite the task to reflect employee-specific insights from the context (e.g., appraisal feedback, performance review details, ticket status, contribution history).
 - Ensure it is specific, fact-seeking, closed-ended, and answerable directly from the context.
 - Make the task sound like a natural enterprise query from a real employee to a smart assistant.
 - Avoid references like “based on the logs,” “from my context,” etc.
- If question7 is marked NO:
 - Ensure the task only accesses data the employee is authorized to see (e.g., “my tasks,” “my appraisal”).
 - Avoid questions about peers, teams, or organization-wide metrics.
 - Use first-person phrasing (e.g., “Can I check...”, “Do I have...”, “Is there any update on my...”).

Ground Truth Revisions:

- If question3 or question5 is marked NO:
 - Ensure the response is precise, complete, and self-contained.
 - Remove any meta-references (e.g., “the context says...”).
 - Avoid vague phrases like “others,” “may vary,” or “etc.”
 - Use specific, grounded details from the context.

Realism & Tone:

- If question6 is marked NO:
 - Adjust the tone to reflect a professional employee-to-assistant interaction.
 - Avoid robotic or overly formal phrasing.
 - Mimic realistic enterprise queries—natural, polite, and context-aware.

Additional Guidelines:

- Do not invent data; only use what's provided in the context.
- Use actionable verbs like "check," "determine," "summarize," "flag," or "identify."
- Maintain a first-person voice—employees are asking about themselves.
- Ensure the ground truth is verifiable and closed-form, derived from the real context.

Output Format: Return only the improved task and ground truth in the exact JSON format:

```
{  
  "task": "Your improved task statement",  
  "ground_truth": "Your improved ground truth"  
}
```