Addressing trust requirements in the design of an open-source multi-agent LLM-based domain-specific chatbot

Requirements Engineering (2026) 31:3
https://doi.org/10.1007/s00766-026-00457-w

ORIGINAL ARTICLE

Addressing trust requirements in the design of an open-source multi-agent LLM-based domain-specific chatbot

Jonatan Axetorn[1] · Felix Edholm[1] · Felix Dobslaw[2] · Lucas Gren[1]

Received: 30 August 2025 / Accepted: 12 January 2026

Abstract

Large Language Models (LLMs) have the potential to automate knowledge-intensive interactions in enterprise systems, yet their adoption is often limited. One reason is a lack of user trust. This study examines how trust can be *systematically engineered* into an LLM-driven, multi-agent chatbot that handles routine human-resources (HR) queries. We follow a two-cycle Design Science Research methodology. Cycle 1 triangulated a systematic literature review with a thematic analysis over semi-structured interviews of six employees at a global firm and a confirmatory workshop with five AI experts to elicit and validate *trust requirements*. Cycle II instantiated these requirements in a multi-agent LLM chatbot prototype artifact and evaluated whether the artifact satisfies them through controlled user sessions and expert walkthroughs, emphasizing perceived usefulness and *trust* captured in post-task interviews ($n = 11$) and operationalizing trust via alignment-oriented measures (faithfulness, answer relevancy, and adversarial robustness). The study yields a refined taxonomy of *external* (transparency, organizational safeguards, third-party security) and *internal* (model provenance, bias risk, reliability) trust factors, identifying *reliability* as the primary determinant of adoption. The implemented design achieved ≥ 0.86 on trust-aligned metrics and was endorsed by 9/11 participants as ready for field deployment. These findings demonstrate that trust can be proactively addressed through design and offer prescriptive guidelines for software engineers seeking to embed LLMs safely and responsibly in socio-technical contexts.

Keywords Large Language Models · multi-agent LLMs · software system design · trust

1 Introduction

The application of Software solutions integrating Large Language Model (LLM) across various business areas enables the use of natural language to address repetitive tasks. Text-based interactions with LLMs are increasingly augmenting [44] or even replacing traditional human-to-human interactions [6]. LLM-based systems are software systems that integrate one or more large language models as core components (via API or local runtime) together with functionality such as retrieval, memory, added guardrails, and user interfaces.

Despite their potential to improve organizational efficiency, the introduction of domain-specific software solutions integrating artificial intelligence (AI) such as LLMs often encounters reluctance. Factors such as fear of job displacement, distrust of AI's perceived human qualities, and general skepticism contribute to delays in adopting these systems [44, 46]. To overcome these challenges, it is crucial to design LLM-based systems that actively build user trust. Key performance-related factors — such as accuracy and low frequency of hallucinations — have been shown to positively influence trust [1, 36]. Since these factors are directly shaped by system design, thoughtful design emerges as a vital strategy for fostering trust in LLM-based technologies.

✉ Felix Dobslaw
felix.dobslaw@miun.se

✉ Lucas Gren
lucas.gren@lucasgren.com

Jonatan Axetorn
jonatan.axetorn@gmail.com

Felix Edholm
felix.edholm@gmail.com

[1] Department of Computer Science and Engineering, Chalmers University of Technology and the University of Gothenburg, Gothenburg, Sweden

[2] Department of Communication, Quality Management and Information Systems, Mid Sweden University, Östersund, Sweden

Published online: 16 March 2026

Developing autonomous agent systems based on LLMs, where *agents* refer to AI-based entities that have capabilities such as planning, social interaction, and memory [49], holds significant potential for positively impacting trust factors such as the reliability of the system. Additionally, LLM-based autonomous agent systems have demonstrated significant versatility [52], highlighting their potential to address a wide range of organizational needs. Ideally, a unified, general-purpose agent-based platform would satisfy the diverse requirements of employees across all organizational roles. In practice, however, developing such comprehensive LLM-driven solutions remains stubbornly out of reach [26, 35]. A more pragmatic approach is to engineer task- or domain-specific systems that apply LLMs for limited scopes. When designing such systems, developers face a strategic decision: whether to host LLMs on-premise or depend on external state-of-the-art model providers. External providers typically offer more capable models than those available and affordable for self-hosting, but sharing sensitive organizational data with external actors raises trust and compliance concerns. Open-source LLMs, while potentially less capable, offer greater control over data location and processing, making them a viable option for organizations prioritizing privacy and compliance [37].

Multi-agent architectures leverage the collaborative abilities of multiple specialized LLM agents, enabling role-based coordination and task allocation that differ fundamentally from single-agent systems [22]. In a multi-agent architecture, specialized agents can be assigned distinct roles (e.g., retrieval, generation, verification), allowing for layered quality checks, iterative refinement processes, and explicit source attribution—capabilities that single-agent systems cannot easily achieve. This architectural approach introduces new trust considerations: users must trust not only individual agents but also the coordination mechanisms, the reliability of inter-agent communication, and the overall system's ability to manage task allocation and error recovery.

While trust in single-agent LLM systems has been extensively studied and trust dimensions for standalone models have been identified (e.g., Huang et al. [24]), this study focuses specifically on *multi-agent* LLM-based systems, exploring the factors that influence trust when multiple specialized agents collaborate to accomplish tasks. Multi-agent architectures introduce unique trust considerations—such as coordination transparency, agent specialization, and distributed responsibility—that differ from single-agent trust dynamics. Additionally, multi-agent structures can compensate for limitations in individual LLM capabilities through collaborative problem-solving [16]. There remains a gap in understanding how to systematically engineer trust into *multi-agent* LLM-based systems. Existing trust frameworks primarily evaluate standalone models and do not account for the interplay between internal trust factors (such as model provenance, bias risk and reliability) and external factors (such as transparency and organizational safeguards) in socio-technical systems with multiple collaborating agents. Research on multi-agent orchestration highlights task allocation and memory management challenges [22], yet empirical studies rarely assess how these design choices affect user trust or adoption.

This paper aims to systematically engineer trust into multi-agent LLM-based (MALLM) systems by: (i) identifying trust factors specific to multi-agent contexts through empirical investigation, (ii) instantiating trust requirements in a concrete multi-agent architecture, and (iii) evaluating whether the implemented mechanisms satisfy those requirements. Through a combination of literature review, interviews, and expert workshops, we identify key trust factors relevant to MALLM systems. Employing a design science research approach, we present an artifact: an open-source multi-agent LLM-based HR chatbot designed to answer questions related to HR guidelines and employment information, with trust factors integrated into its design. The artifact is evaluated using both qualitative and quantitative methods.

This work differentiates itself from existing research in several key ways. Unlike trust frameworks that evaluate standalone LLM models [24, 36], we focus on complete multi-agent systems and the interplay between internal and external trust factors in socio-technical contexts. Unlike multi-agent orchestration research that emphasizes scalability and fault tolerance [22, 34, 48], we systematically map design choices to trust requirements and evaluate their impact on user trust. This differentiation contributes a practical methodology for developing and evaluating trustworthy MALLM systems, with explicit guidelines for mapping requirements to design decisions and assessing trust outcomes.

Our study adopts a design-science stance: we first identify trust requirements for LLM-based HR support (via literature, interviews, and a workshop), then instantiate those requirements in a multi-agent artifact, and finally evaluate whether the artifact meets them using alignment-oriented metrics and qualitative feedback. The evaluation is thus requirement-driven rather than competitor-driven.

This paper is outlined as follows: Sect. 2, Background, introduces the fundamentals of trust and its relationship to LLMs while Sect. 3, Related Work, presents the literature and research gap, including studies on trust in AI and multi-agent architectures. The study follows a two-cycle design science approach detailed in Sect. 4, Method, capturing an understanding of trust in LLMs by literature review, the collection of perceptions in support of thematic analysis,

workshops, as well as the creation and evaluation of a MALLM chatbot prototype. Sections 4.4 and 5 present the two design science cycle designs and findings. Section 7, Discussion, then explores the findings implications and notes threats to validity of the study. Finally, Sect. 8, Conclusions, summarizes and concludes the paper raising future work directions.

2 Background

We introduce here foundational concepts relevant to our study, including definitions of trust, large language models and prompt engineering, hallucinations and retrieval augmentation, multi-agent systems and their orchestration, and evaluation techniques.

2.1 Trust and digital systems

Trust is the relationship between a party that relies on something (the trustor) and the entity being trusted (the trustee). It has been studied in organizational theory and sociology to understand how uncertainty, vulnerability and dependence create the conditions for trust to emerge [29, 38]. Trust in automation has further been extensively studied in human factors research. Lee and See's foundational work emphasizes designing systems for appropriate reliance—where users neither over-trust nor under-trust automated systems [31]. Hoff and Bashir's comprehensive review integrates empirical evidence on factors that influence trust in automation, identifying key antecedents such as system performance, transparency, and user experience [23]. In enterprise settings, these elements manifest when employees depend on automated systems for sensitive or complex information. Trust is often discussed at different levels—individual, interpersonal, relational, and societal—but the focus here is on the one-way relationship between a user and a technical artifact. A system is trusted when it is perceived as reliable, transparent and safe, and when organizational safeguards protect the trustor from errors or misuse.

2.2 Large language models and prompt engineering

LLMs are neural networks trained on vast text corpora to create statistical models that predict sequences of tokens. Text is first tokenized—broken into discrete units (tokens) that may represent words, subwords, or characters—and the model learns patterns from these token sequences. Given an input sequence (such as an incomplete sentence, question, or command), an LLM predicts the most likely next tokens based on statistical patterns learned during training. This autoregressive process—generating one token at a time, conditioning each prediction on the previous tokens—enables LLMs to produce coherent text that appears to understand and respond to natural language, even though the underlying mechanism is probabilistic token prediction in high-dimensional numeric spaces rather than semantic interpretation.

Models such as BERT and ChatGPT popularized different architectural approaches: BERT uses bidirectional attention to understand context in both directions, while ChatGPT and similar generative models use unidirectional attention to generate text sequentially [9, 13]. Despite their probabilistic nature, LLMs can effectively answer questions, summarize documents, and perform complex reasoning tasks, making them attractive for automating knowledge work. The performance of an LLM depends heavily on the prompt—the input sequence that conditions the model's token predictions. Prompt engineering techniques, such as using delimiters, providing examples, specifying steps or output length, and including precise details, help improve the relevance and quality of responses by shaping the statistical context that guides token prediction, without modifying the underlying model [33].

2.3 Hallucinations and retrieval augmentation

LLMs occasionally generate unfaithful or incorrect content, a phenomenon often called hallucination [27]. Hallucinations occur for several reasons: (i) LLMs are trained to generate plausible-sounding text based on statistical patterns rather than factual knowledge, leading them to produce confident but incorrect statements; (ii) training data may contain errors, biases, or outdated information that the model reproduces; (iii) the model may overgeneralize from training examples or fill gaps in knowledge with plausible but false information; and (iv) the probabilistic nature of token prediction means the model can generate sequences that are syntactically correct but factually wrong. Hallucinations manifest in various ways: the model may invent facts not present in the input, fabricate citations or sources, produce contradictory information, or confidently assert incorrect details. In sensitive domains such as HR, hallucinations pose particular risks: they may reveal or infer personal information from training data, generate incorrect guidance that leads to poor decisions, or produce outputs that appear authoritative but contain private details that should not be disclosed.

One way to mitigate hallucinations is retrieval-augmented generation (RAG), which grounds responses in external knowledge sources. In RAG, "external sources" refer to knowledge bases that are *external to the LLM's pre-trained parameters*—such as organizational documents,

databases, or up-to-date information repositories that are separate from the model's training data. The typical RAG pipeline works as follows: (i) *indexing* converts external sources into vector embeddings and stores them in a searchable database; (ii) *retrieval* identifies the most relevant pieces from these external sources based on a query; and (iii) *generation* conditions the LLM on both its pre-trained knowledge and the retrieved context [21, 32]. "Conditioning" an LLM means providing additional context (the retrieved documents) alongside the user's query as input, so the model's token predictions are influenced by both its pre-trained knowledge and the specific external information provided. This process allows the model to generate responses that are grounded in the retrieved context rather than relying solely on potentially outdated or incomplete training data. By grounding answers in up-to-date external data, RAG reduces hallucinations and enhances factual correctness [3]. However, RAG systems introduce their own risks regarding sensitive information. While RAG can reduce hallucinations by grounding responses in controlled external sources, it also means that any sensitive or personal information present in those external sources becomes accessible to the system. If the external knowledge base contains personal data, employee records, or confidential information, the RAG system may retrieve and include such information in its responses, potentially exposing sensitive details that should remain private. This risk is particularly acute in enterprise contexts where RAG systems are deployed over internal documents that may contain personal information. Therefore, careful access control, data filtering, and privacy safeguards are essential when implementing RAG systems in sensitive domains.

2.4 Multi-agent systems and orchestration

Artificial agents are software entities that act autonomously, interact with one another and the environment, respond to changes, and proactively pursue goals [51]. The integration of LLMs into agents enables natural language reasoning, memory, and tool use, fostering multi-agent systems in which specialized agents collaborate on complex tasks [49]. Coordination involves orchestrating prompts, data retrieval, API calls and state management; frameworks such as LangChain and LangGraph provide libraries for constructing sequential or graph-based workflows.[1] These orchestration tools simplify the development of multi-agent applications by offering reusable components, while *guardrails* monitor inputs and outputs to filter harmful or biased content [4, 15]. However, current safety mechanisms provide only limited protection against adversarial prompts [45], and researchers continue to explore methods for ensuring reliable and secure agent behavior [20].

2.5 Evaluation techniques

Evaluating the quality of an LLM system requires both intrinsic metrics and human-aligned judgment. Traditional software testing typically verifies correctness by comparing program outputs against expected values via simple assertions. While LLM-based systems are non-deterministic and may produce multiple acceptable responses, non-determinism itself is not the primary verification challenge—stochastic systems have been successfully verified long before LLMs were engineered [14]. The fundamental challenge in verifying LLM outputs lies in the nature of their output: natural language is unstructured and subject to interpretation, making it difficult to establish objective correctness criteria. Unlike structured outputs (e.g., numerical results, boolean values, or well-defined data structures), natural language responses can be semantically equivalent while using different wording, can be correct in multiple valid ways, and require interpretation to assess quality. As a result, evaluation must focus on the *semantic* quality of the output rather than exact string equality. Dobslaw et al. [14] propose a faceted taxonomy for testing LLM-based software that distinguishes between atomic oracles—which check specific conditions in a single output—and aggregated oracles that assess semantic intent across multiple outputs or by using other models as judges.

2.5.1 LLM-as-a-judge

LLM-as-a-judge frameworks exemplify these aggregated semantic oracles: one model evaluates another model's output according to a rubric [53]. Coined by Zheng et al. [53], this approach uses LLMs as evaluators for tasks that typically require human judgment, such as assessing the quality of chatbot responses in open-ended dialogue. This addresses a key limitation of traditional benchmarks, which often fail to capture how well models align with human preferences. By contrast, LLM-based judges can offer a scalable and efficient alternative to human evaluation.

To test the viability of this approach, Zheng et al. [53] developed two benchmarks. Their findings show that GPT-4, when used as a judge, agrees with human preferences over 80% of the time—comparable to the agreement rate between human annotators themselves. While promising, the study also highlights several limitations, including susceptibility to biases (e.g., favoring the first-listed response or more verbose answers) and occasional failures

[1] https://langchain-ai.github.io/langgraph/

in evaluating complex tasks requiring precise reasoning. Despite these issues, the results suggest that, when carefully applied, LLM-as-a-judge can serve as a practical and surprisingly reliable proxy for human evaluation in many settings. However, these limitations necessitate complementary human validation, particularly to identify false positives (cases where the LLM judge incorrectly rates an answer as correct).

2.5.2 DeepEval framework

Frameworks such as DeepEval offer features that build on the LLM-as-a-judge idea to provide standardized metrics for retrieval-augmented systems [11]. DeepEval is an open-source evaluation framework designed to assess the performance of LLM-based systems. Among the evaluation metrics it offers for RAG scenarios are faithfulness, answer relevancy, and contextual relevancy. Originally introduced in the RAGAS framework by Es et al. [18], these metrics are defined as follows:

- *Faithfulness* measures how accurately the generated answer reflects the retrieved context, aiding in identifying hallucinations. This metric assesses whether the answer is grounded in the provided context rather than generated from the model's pre-trained knowledge.
- *Answer relevancy* evaluates the degree to which the generated response directly addresses the user's question. The metric does not take into account factuality but instead focuses on completeness and focus, penalizing responses that are irrelevant, incomplete, or verbose.
- *Contextual relevancy* assesses how relevant the retrieved context used to generate the answer is to the input question. The context should be focused and contain as little irrelevant information as possible.

DeepEval also provides the capability to create custom evaluation metrics through the use of G-Eval [12]. G-Eval is a framework that enables the evaluation of outputs based on user-defined criteria. For instance, it can be employed to assess the correctness of a given output. This is achieved by specifying both the evaluation criteria and the corresponding evaluation steps. In our study, we defined a custom correctness metric with the following criteria and evaluation steps:

- *Criteria:* Determine whether the actual output is factually correct based on the expected output.
- *Evaluation steps:*
  - Check whether the factual claims in the actual output contradict any factual claims in the expected output (factual contradictions are not acceptable).
  - Heavily penalize omission of detail.
  - Vague language is acceptable. Differences in opinions, preferences, or interpretations (as opposed to factual claims) are acceptable and do not constitute contradictions.

This approach enables the creation of metrics that are not predefined in the DeepEval framework, offering greater versatility when evaluating the outputs of LLM systems. Together, these approaches emphasize semantic correctness and robustness over simple assertion-based testing, bridging the gap between traditional program evaluation and the assessment of language generation models.

3 Related work

This section reviews prior research on foundational trust models from human factors research, trust factors in LLM systems, challenges in multi-agent LLM systems, collaboration frameworks, and advances in retrieval-augmented generation filtering.

3.1 Foundational trust models from human factors research

Trust has been extensively studied in human factors and organizational psychology, providing mature theoretical frameworks that inform our understanding of human-technology interactions. Mayer et al.'s integrative model of organizational trust identifies three key antecedents: ability (competence and skills), benevolence (positive intentions), and integrity (adherence to principles) [38]. This model has been widely applied across different contexts and provides a robust foundation for understanding trust dynamics in organizational settings.

Rotter's work on interpersonal trust emphasizes the role of trustworthiness and gullibility, highlighting how individual differences in trust propensity affect technology adoption [42]. In digital contexts, Kelton et al. identify four levels of trust: individual (inherent trust based on experience), interpersonal (social connections), relational (emergent from ongoing relationships), and societal (community-wide trust) [29]. These frameworks establish that trust is context-dependent and requires uncertainty, vulnerability, and dependence to be relevant.

3.2 Trust factors in LLM systems

Building on these foundational models, recent work on trust in AI identifies several dimensions that influence whether users accept machine outputs. Liu et al. [36] and Huang et al. [25] conducted extensive literature reviews and developed taxonomies of trust factors while designing benchmarks to evaluate LLMs.

Huang et al. [24] provide a comprehensive synthesis of trustworthiness in large language models, identifying eight trust dimensions based on numerous primary studies: truthfulness, safety, fairness, robustness, privacy, machine ethics, transparency, and intellectual property. Their work synthesizes evidence from a wide range of primary studies and evaluates their framework across various LLMs, providing benchmarks and evaluation practices for each dimension. While their work focuses on assessing the models themselves rather than complete systems incorporating them, the trust dimensions they identify remain relevant as they ultimately relate to how users perceive and trust LLM-generated content. Their taxonomy includes truthfulness, which emphasizes providing correct information; privacy, which is treated as a separate category rather than a subset of safety; and transparency, which relates to how openly a system communicates how it generates its outputs. These dimensions provide a structured framework for understanding trust in LLM-based systems, though they primarily address model-level properties rather than system-level design considerations.

Other authors propose complementary perspectives. Liu et al. [36] categorize trust into reliability, safety, and explainability & reasoning, where reliability refers to accuracy and consistency while minimizing errors, safety involves protecting sensitive information, and explainability & reasoning focuses on how well a system can justify its responses. Schwartz et al. [43] emphasize openness about system capabilities, limitations, and reliability, as well as the importance of a positive first impression to build a trajectory of trust. These taxonomies, while varying in their specific categorizations, underpin design guidelines for systems that seek to engender trust through clear communication and robust safeguards.

While our background has emphasized LLM technology, trust has long been theorized and operationalized within software engineering and requirements engineering contexts. Recent work in software engineering calls for grounding studies of "trust in AI assistants" in established trust theory and for moving beyond equating trust with mere acceptance of generated artifacts [5]. Closer to RE practice, Borg et al. elicited concrete requirements for a domain chatbot and demonstrated how such requirements can be turned into verifiable QA tests—evidence that trust concerns can (and should) be captured as requirements and evaluated systematically [7]. In regulated, safety-critical settings, ethics-assurance cases provide another pattern to make confidence claims explicit, linking ethical principles, evidence, and system constraints in a form familiar to requirements engineering practice [28]. Complementing these application- and process-oriented perspectives, *TrustLLM* synthesizes eight trustworthiness dimensions (e.g., robustness, fairness, privacy, transparency) and curates benchmarks and evaluation practices for LLMs [24]. In this paper, we treat *trust requirements* as requirement engineering-level properties that map to these dimensions (e.g., reliability → robustness/testing; bias risk → fairness/data governance) and we aim to make the linkage explicit by (i) eliciting trust-related requirements, (ii) mapping them to design decisions and safeguards, and (iii) evaluating whether the implemented mechanisms satisfy those requirements.

3.3 Challenges in multi-agent LLM systems

As LLM agents collaborate, they introduce unique challenges that have implications for trust. Research highlights four areas requiring attention: (i) allocating tasks to agents in a way that exploits their specialized capabilities; (ii) fostering robust reasoning through debate or discussion among agents to refine intermediate results; (iii) managing layered context and memory so that agents maintain awareness of global objectives, local tasks and shared knowledge; and (iv) handling different types of memory (long-term, working, episodic) coherently across interactions [22]. These challenges complicate the orchestration of multiple agents and call for systematic approaches to task planning, context propagation and memory management. Surveys of MALLM systems synthesize these challenges within broader workflow frameworks encompassing agent profiles, perception, self-action, mutual interaction, and evolution [34].

These multi-agent challenges create trust concerns that extend beyond those addressed by existing trust frameworks. While Huang et al. [24] provide a comprehensive synthesis of trust dimensions (truthfulness, safety, fairness, robustness, privacy, machine ethics, transparency, and intellectual property) for standalone LLM models, their framework does not account for how these dimensions manifest in multi-agent systems. For instance, task allocation challenges relate to robustness and truthfulness: poor allocation decisions can lead to agents operating outside their capabilities, producing incorrect outputs that undermine trust. Reasoning through debate among agents touches on transparency users need to understand how multiple agents arrive at a consensus, yet current transparency frameworks focus on single-model explainability rather than multi-agent coordination. Context and memory management directly impacts privacy

and safety: agents may inadvertently expose sensitive information through shared memory or context propagation, creating privacy risks that single-agent systems do not face. Memory coherence across interactions affects robustness and truthfulness: inconsistent memory handling can lead to contradictory outputs that erode user trust. These trust implications of multi-agent challenges are not addressed by existing trust taxonomies that evaluate standalone models, creating a gap in understanding how to design trustworthy MALLM systems.

3.4 Collaboration frameworks

Multi-agent collaboration can take several forms. In cooperation, agents jointly pursue a shared goal; in competition, agents optimize their own objectives, sometimes leveraging adversarial reasoning; and in coopetition, they combine collaboration and competition. Collaboration can be governed by rules (predefined protocols), roles (specialized responsibilities) or models (probabilistic decision-making under uncertainty). Three communication structures are commonly discussed: a centralized architecture connects all agents through a coordinator; a decentralized architecture distributes control among peers with local knowledge; and a hierarchical architecture organizes agents in layers with intermediate supervisors. Coordination and orchestration architectures can be static, relying on domain expertise to define collaboration channels upfront, or dynamic, adapting roles and interactions in real-time to evolving tasks [34, 48]. Each approach offers different trade-offs in scalability, fault tolerance and implementation complexity.

3.5 Advances in retrieval-augmented generation filtering

Recent work seeks to further improve RAG by introducing self-criticism and multi-agent filtering. In self-retrieval, the model decides when to look up additional information and evaluates its own outputs, reducing irrelevant citations and improving factuality [2]. An extension called multi-agent retrieval introduces specialized agents: a predictor fetches documents and proposes an answer; a judge filters out irrelevant sources based on the query and initial answer; and a final predictor synthesizes the final response from the curated evidence. These mechanisms operate without retraining the base language model and have shown improved performance on benchmark datasets [10].

3.6 Research gap

Despite extensive work on trust taxonomies, multi-agent collaboration and retrieval-augmented generation, there remains a gap in systematically engineering trust into a multi-agent, LLM-driven chatbot for enterprise use. This gap emerges from the limitations of existing work in these areas:

Trust frameworks for standalone models. Huang et al. [24] evaluation practices for LLMs, synthesizing evidence from numerous primary studies and evaluating their framework across various standalone LLMs. Similarly, Liu et al. [36] categorize trust and Schwartz et al. [43] emphasize openness about system capabilities and reliability, all in single-model contexts. As established in Sect. 3.2, the challenges extend beyond those addressed by these single-model trust frameworks. These frameworks do not account for the interplay between internal trust factors and external factors in socio-technical systems with multiple collaborating agents.

Multi-agent orchestration without trust assessment. Research on multi-agent orchestration [22] highlights four key challenges: task allocation, robust reasoning through debate, context and memory management, and memory coherence. However, empirical studies in this area rarely assess how these design choices affect user trust or adoption, nor do they map these challenges to trust dimensions. Surveys of MALLM systems [34, 48] examine collaboration frameworks and system workflows, but they focus on scalability, fault tolerance and implementation complexity rather than trust implications. The trust concerns raised by multi-agent challenges (as detailed in Sect. 3.2) are not systematically addressed in this body of work.

RAG and filtering advances not applied to sensitive enterprise domains. Recent work seeks to improve RAG through self-criticism and multi-agent filtering [2, 10]. Chang et al. [10] demonstrate improved performance on benchmark datasets through specialized agents (predictor, judge, final predictor), but these advances have not been applied to sensitive domains like human-resources support, where accuracy, privacy and reliability are paramount. The evaluation focuses on technical performance metrics rather than user trust or adoption in enterprise contexts.

Software Engineering trust literature without multi-agent system focus. Recent work in Software Engineering calls for grounding studies of trust in AI assistants in established trust theory [5], and requirements engineering practice demonstrates how trust concerns can be captured as requirements and evaluated systematically [7, 28]. However, these approaches have not been applied to MALLM systems, where the complexity of multiple collaborating agents

introduces additional trust considerations beyond those addressed in single-agent or traditional software contexts.

This paper addresses this gap by adopting a design science research methodology to derive trust requirements from literature, employee interviews and expert workshops, and by implementing an open-source multi-agent HR chatbot that integrates retrieval augmentation, guardrails and orchestration. We evaluate the prototype using both intrinsic quality metrics and perceived trust measures, refining the trust taxonomy and providing prescriptive guidelines for embedding LLMs responsibly in socio-technical contexts.

4 Method

This section outlines the design science research (DSR) methodology used to develop and evaluate our multi-agent HR chatbot. DSR focuses on building a functional artifact that solves a real problem while simultaneously generating knowledge about the design process [50]. Our artifact comprises two components—one that answers questions grounded in employment data and another that references HR guideline documents—designed to explore how particular design choices influence user trust.

4.1 Overview of the HR chatbot artifact

The artifact developed in this study is a MALLM-based HR chatbot that answers questions using either HR guideline documents or structured employment data. The chatbot comprises two multi-agent components: a *guidelines component* that implements an enhanced retrieval-augmented generation pipeline with specialized agents for judging document relevance, generating answers, and checking quality; and an *employment component* that handles structured data queries through field identification and data retrieval. Agents coordinate via LangGraph and adopt a cooperative, role-based strategy [48]. An overview of the architecture is shown in Fig. 1. This overview provides context for the methodology described in the remainder of this section. Detailed descriptions of the two multi-agent components and their implementation appear in Sect. 5.

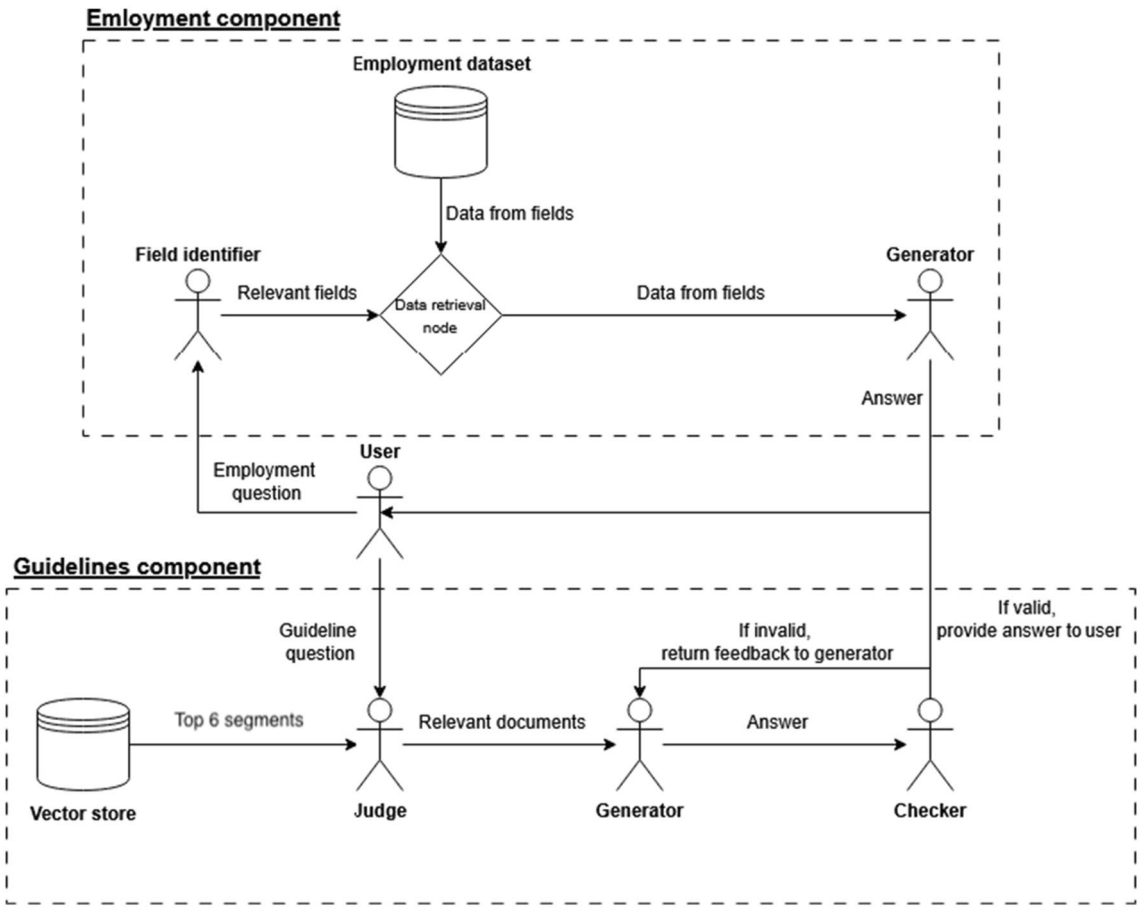

Fig. 1 Overview of the HR chatbot: two components (guidelines and employment), user interaction flow, and coordination via LangGraph among role-based agents

4.2 Design science research approach

We follow the regulative cycle by Wieringa [50], grouping its phases into three categories after Knauss [30]: *problem investigation*, *solution design and validation*, and *evaluation*. Rather than being strictly sequential, these activities interact iteratively; insights from each phase shape the artifact's further development. Our research proceeded through two cycles: the first focused on understanding trust factors and devising preliminary solutions (Sect. 4.4), while the second refined the prototype and conducted formal evaluations (Sect. 5).

4.3 Research questions

To guide our investigation, we address the following research questions:

- *RQ1:* What are the main trust factors that exist in the usage of MALLM-based systems?
- *RQ2:* What potential solutions can be integrated into the system design of a MALLM-based HR chatbot to address the relevant trust factors identified in RQ1?
- *RQ3:* To what extent can the relevant trust factors identified in RQ1 be addressed through the design solutions implemented in a MALLM-based HR chatbot?

These questions structure our research into three phases: problem investigation (RQ1), solution design (RQ2), and evaluation (RQ3), corresponding to the design science research phases outlined above.

Research context

The study was conducted in collaboration with a large multinational corporation with software and data engineering roles distributed across multiple departments (e.g., HR, engineering, operations, cybersecurity). This organizational structure enabled access to staff with technical responsibilities embedded in various functions, which explains the availability of AI experts for the workshop. Interview participants represented diverse functions, seniority levels (junior to senior roles), and prior AI exposure (low to high), providing varied perspectives on trust in LLM-based systems. The company's employees are generally tech-savvy, with many having experience using LLM tools in their work, though levels of technical expertise and AI familiarity varied across roles and departments.

Problem investigation

To elicit trust requirements (RQ1) we conducted six remote, semi-structured interviews with employees at the collaborating company (about 60 min each; ≈ 6 hours total) and a two-day on-site workshop with five company AI experts (≈ 16 hours). The interviews were recorded with consent and transcribed; insights were thematically analyzed and fed into the workshop discussions on design choices for an LLM-based HR chatbot. During the evaluation phase (Cycle II), five of the original six interviewees returned for follow-up interviews (≈ 45 minutes each; ≈ 3.75 hours total) to assess the artifact against the identified trust factors. In sum, the study involved 11 unique participants (six employees and five AI experts) contributing 16 participants across activities (6 problem-investigation interviews, 1 expert workshop, 5 evaluation interviews) and ≈ 25.75 hours of contact time. The AI expert workshop cohort was distinct from the interview cohort, while the evaluation interviews intentionally overlapped with the first interview round (5/6 returning) to enable within-participant comparison of perceptions.

Solution design and validation

Potential solutions (addressing RQ2) were devised through literature review, design activities, and feedback from the AI workshop. The design evolved iteratively as we explored different architectural choices and guardrails. Detailed prompts, agent configurations and other artefacts are provided in the supplementary material via Zenodo[2].

Implementation

The final artifact is a terminal-based Python application consisting of two multi-agent subsystems (detailed in Sect. 4.1). The *guidelines component* implements an enhanced retrieval-augmented generation pipeline: documents are stored in a vector database, a *judge agent* ranks and filters the most relevant segments, a *generator agent* crafts an answer using only the selected segments, and a *checker agent* verifies that the answer addresses the question, grounds its content in the context, and omits extraneous information. The checker may request revisions up to three times before returning the answer to the user. All prompts, including those for the judge, generator and checker, are available as supplementary material.

The *employment component* uses a field-identifier agent to determine which data fields (e.g., department, manager, salary) are needed to answer a question. A data retrieval module fetches these fields from a CSV dataset, and a generator agent composes a concise response. If needed data are unavailable, the system informs the user rather than improvising an answer.

Users pose a question and specify whether it concerns general HR policies or personal employment data. This simple selection is made via a text prompt rather than a graphical menu and directs the query to the appropriate component.

As discussed in Sect. 1 (see paragraph on multi-agent architectures), we adopted a multi-agent structure rather

[2] Link to supplementary material.

than a single LLM with RAG. While a single LLM with RAG can retrieve relevant documents and generate responses, it lacks the layered quality checks and iterative refinement processes that multi-agent systems enable. For an HR chatbot handling sensitive information where accuracy and trust are paramount, the multi-agent architecture provides critical capabilities: the judge agent filters irrelevant or low-quality retrieved segments before generation, reducing hallucination risk; the checker agent verifies answer quality and can request revisions, enabling iterative improvement that a single-pass RAG system cannot achieve; and the separation of concerns (retrieval, generation, verification) allows each agent to specialize, improving overall system reliability. This architectural choice was particularly important given our use of an open-source LLM (Llama3), which benefits from the collaborative problem-solving capabilities of multi-agent systems to compensate for limitations in individual agent capabilities [16].

Evaluation

Our evaluation combined quantitative simulation and qualitative feedback to answer RQ3. The trust factors identified in Cycle I through interviews and workshop (Sect. 5.2)—particularly reliability, transparency, and the need for accurate, contextually relevant responses—informed our selection of quantitative evaluation metrics. We mapped these trust factors to measurable dimensions: reliability was operationalized through faithfulness (measuring whether answers reflect retrieved context, addressing hallucination concerns) and a custom correctness metric (assessing factual accuracy); transparency was evaluated through answer relevancy (assessing how well responses address questions) and contextual relevancy (evaluating the pertinence of retrieved sources). These metrics, drawn from the RAGAS framework [18] and extended with a custom correctness metric, align with the trust requirements specified in Sect. 5.2.3.

Quantitatively, the prototype was exercised across 2 860 evaluation runs—360 each for answer relevancy, faithfulness and contextual relevancy; 500 each for the corresponding robustness metrics; and 280 for correctness. These runs yielded precision scores for each metric and guided parameter tuning. Qualitatively, the second set of five interviews explored how participants perceived the usefulness, reliability and transparency of the system, directly assessing the trust factors identified in Cycle I. Discussion protocols and additional evaluation materials can be found in the supplementary material.

4.4 Overview of research cycles

Cycle 1 is centered on problem investigation and preliminary design. Interviews and the workshop provided insight into trust requirements, which we translated into design principles and an initial prototype. Cycle II, described in Sect. 5, refined the prototype, implemented the final multi-agent architecture, and executed quantitative and qualitative evaluations. Figure 2 summarizes the activities undertaken during these two cycles. Cycle II built on the first one, allowing us to refine our trust taxonomy and evaluate the impact of specific design choices on perceived trust.

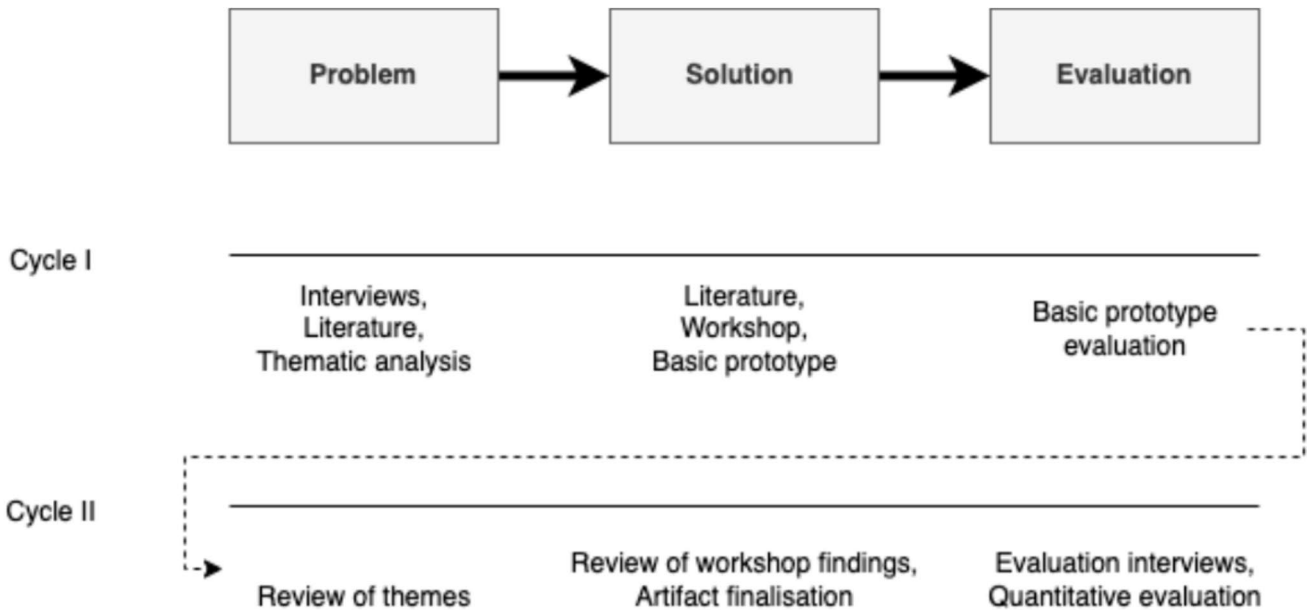

Fig. 2 The two design science cycles in the study are detailed with the respective performed activities

4.4.1 Evaluation data

To avoid disclosing proprietary material and to enable reproducibility, we evaluated the guidelines component on synthetic HR guideline documents. Fourteen Markdown documents were generated with an LLM using prompts seeded from public HR policy templates and common benefits/leave topics; outputs were redacted for company identifiers and lightly edited for internal consistency. For the employment component, we used a small synthetic employee dataset with controlled ground truth. Treating these corpora as ground truth allowed us to run repeatable tests while isolating design effects from organization-specific content.

5 Cycle I: understanding trust challenges

Cycle I focuses on understanding the trust challenges of LLM-based systems and exploring possible design solutions, addressing RQ1 and RQ2. We adopted a problem-driven investigation to identify trust factors relevant to a multi-agent HR chatbot. Semi-structured interviews with potential users served as the primary data source; two of the authors independently performed a thematic analysis before merging results to reduce bias. To complement the interviews, we surveyed the literature on multi-agent chatbot architectures and potential safeguards. The literature survey focused on identifying design patterns, orchestration frameworks, and safety mechanisms relevant to MALLM systems. We searched academic databases and recent preprints for work on MALLM architectures, retrieval-augmented generation systems, trust factors in LLM-based systems, and safety mechanisms such as guardrails and verification approaches. Given the rapid pace of development in LLMs, we included both peer-reviewed publications and recent preprints to capture current state-of-the-art approaches. The literature review informed the design choices explored in the expert workshop and helped identify candidate architectural patterns and safeguards to address trust concerns.

Candidate design solutions were further scrutinized during a two-day workshop with five experts in building LLM-based systems. The workshop facilitated structured discussions on architectural options, orchestration frameworks, and safety mechanisms, and the experts' feedback provided an initial validation of our design choices.

Although the emphasis of this cycle was on problem understanding and solution exploration, we built a simple prototype to test high-level considerations such as framework selection, the feasibility of an agentic RAG system, and compatibility with available LLMs. The prototype was evaluated informally by the authors, guided by insights from the literature on best practices for HR chatbots.

In summary, Cycle I combined qualitative data collection and expert feedback to derive a preliminary taxonomy of trust factors and to outline a set of design principles. Detailed descriptions of the interview methodology, thematic analysis, and workshop setup are provided in the supplementary material, and Sect. 5.2 reports the trust factors elicited from users along with key insights from the expert discussions. Concretely, we identified cooperation-style, role-based coordination among agents, refusal on low-relevance retrieval as a guardrail against hallucination, and explicit source citation as safeguards to support trust.

5.1 Method: qualitative data collection

To understand user trust in LLM-based systems and explore possible solution designs, Cycle I relied on two qualitative approaches: semi-structured interviews with employees and a workshop with domain experts.

5.1.1 Interviews

Six employees from the partner company were interviewed to identify trust factors relevant to an HR chatbot. We adopted a standardized, open-ended format [39, 41] in which the same open questions were posed to each participant, with follow-ups as needed. Snowball sampling ensured a diverse set of informants: an industry supervisor recommended information-rich participants from different roles and with varying levels of AI knowledge. Reported demographics indicated variation across functions (e.g., HR, engineering, operations), seniority (junior to senior roles), and prior AI exposure (low to high). Questions covered four topics—demographics, AI/LLM experience, attitudes toward and trust in AI, and HR-system specifics—and the interview guide is available in the supplementary material. Conversations lasted about 60 min, were conducted remotely, and were recorded and transcribed with consent. Each author independently analyzed the transcripts and merged their findings to mitigate bias.

Thematic analysis

Interview transcripts were analyzed using the five-phase thematic analysis process by Braun and Clarke [8]: repeated reading to familiarize ourselves with the data, systematic coding of meaningful features, clustering codes into preliminary themes, reviewing and refining those themes for coherence, and finally defining and naming the themes. This recursive process allowed us to identify patterns in participants' perceptions of trust and to map them to design requirements for the chatbot.

5.1.2 Workshop

A two-day on-site workshop further explored design and implementation choices for LLM-based systems. Seven participants—including the authors and AI specialists from sales, HR, and cybersecurity—discussed two use cases: (1) using LLMs to manage large volumes of internal documentation and (2) deploying an LLM-based chatbot to answer employee questions about employment matters, HR guidelines and company policies. The workshop provided empirical insight into how experts approach AI implementation in organizational settings, highlighted design trade-offs, and assessed the feasibility and scalability of candidate architectures. These discussions informed both the preliminary design of our artifact and the refinement of trust factors uncovered during the interviews.

5.2 Findings

We here present the findings from the interviews regarding RQ1 and the findings from the workshop related to RQ2 which then informed the design of the artifact for Cycle II.

5.2.1 Trust factors in LLM-based systems (RQ1)

The thematic analysis produced five themes, but only two relate directly to trust in LLM-based systems (Fig. 3). The other three (*AI as a helping hand*, *Concerns about replacing human interaction*, and *Critical thinking and revising outputs*) reflect general attitudes and are not addressed further here.

The trust factors identified here align with established frameworks from human factors research. Lee and See's emphasis on appropriate reliance [31] and Hoff and Bashir's factors influencing trust in automation [23]—such as system performance, transparency, and user experience—remain relevant for LLM-based systems. However, the LLM era introduces unique challenges: the non-deterministic nature of outputs creates greater uncertainty than traditional automated systems, the opacity of model reasoning complicates transparency efforts, and the risk of hallucinations introduces novel reliability concerns that differ from predictable automation failures. These factors—particularly reliability emerging as paramount—reflect both continuity with established trust theory and the distinctive challenges posed by generative AI systems.

External trust factors

These refer to influences outside the technical design that nonetheless shape user trust. They encompass transparency, organizational change management and education, and perceptions of vendor or provider security.

Transparency Participants want clarity on how the system is built, what data it uses, and its limitations. Such openness helps calibrate expectations: a tool need not be perfect but should be honest about its fallibility.

Interviewee 3: "That's the thing that I'm saying, I don't trust 100%, but I still think we can implement things acknowledging that probably 100% is impossible ... we can gain trust and confidence of all the people that are gonna use it."

Change management and usage Trust builds through use [23]. Four interviewees reported overcoming early skepticism once they saw the system improve their work. Organizational support—promoting adoption, demonstrating value beyond existing HR support, and ensuring easy access—is therefore essential.

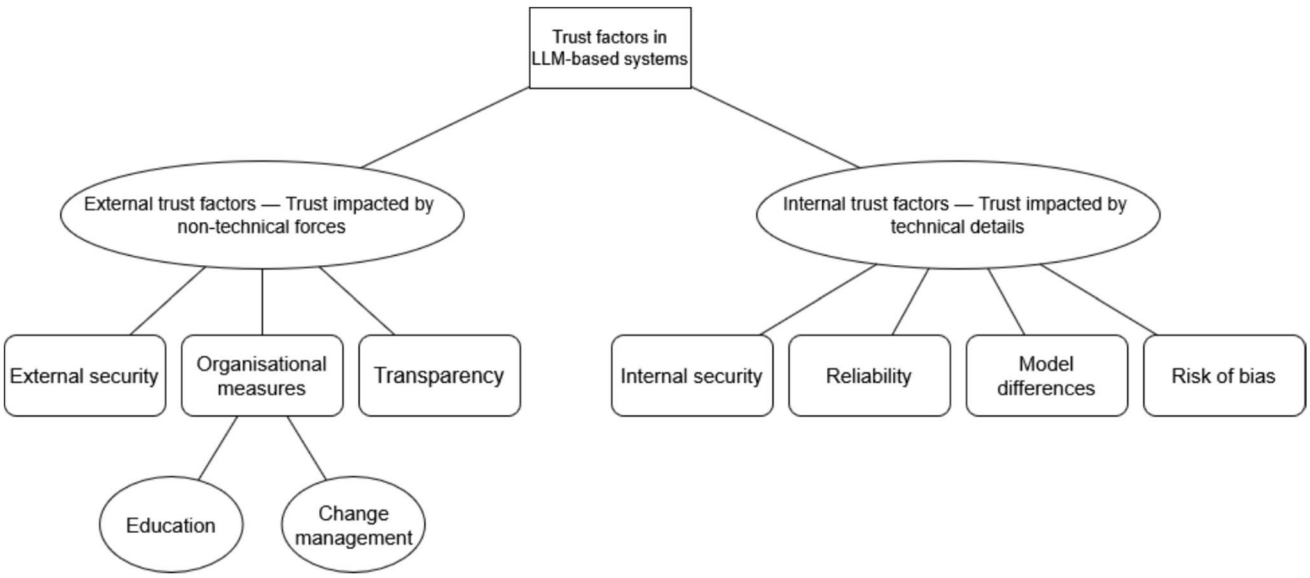

Fig. 3 Identified trust factors in LLM-based systems

Interviewee 5: "Can you trust what it's telling you? Will it make mistakes? ... As I've used [an LLM tool] and seen it evolve ... I really see the opportunities."

Education plays a pivotal role in this process. All participants believed that training on LLM-based chatbots would enhance trust and utility.

Interviewee 4: "I really think that I would benefit from educating myself more in optimizing the usage [of LLM-based chatbots]."

External security Trust also depends on the origin of the model. Several respondents hesitated to share sensitive information with external providers or overseas services; confidence increases when corporate IT hosts and vets the tools.

Interviewee 3: "For instance, obviously I played around with Deepseek, and I knew that using Deepseek was basically sending information to China ... The moment I'm able to actually get Deepseek working on a server, I would probably use it in a different way."

Interviewee 1: "I have no real limitations as long as I know that these tools have been embedded by corporate IT from a security standpoint."

Internal trust factors

These factors stem from the design and behavior of the system itself: internal security, risk of bias, model differences and reliability.

Internal security Users worry that an HR chatbot could reveal personal data or fail to meet legal and IT compliance. Strong safeguards are therefore imperative.

Interviewee 4: "We have to be compliant with all the rules that exist. We have IT processes and legal processes that must be followed."

Risk of bias Several respondents voiced concerns that LLMs might perpetuate biases or limit diversity, especially in recruitment contexts. The EU AI Act classifies AI systems used in recruitment and selection as high-risk applications, requiring strict compliance with transparency, human oversight, and bias mitigation requirements. While our study focuses on informational queries (e.g., policy questions, employment data retrieval) rather than decision-making in recruitment, these concerns highlight the importance of designing LLM-based systems with bias awareness and mitigation strategies, particularly when such systems might be extended to sensitive decision-making contexts.

Interviewee 5: "And then also, like I mentioned before, the ethics around the information that people use from it, and how these models have been built ... is information it gives representative of the wider population?"

Model differences Participants observed clear disparities between internal and public LLM tools. Five interviewees found corporate models less capable than services like ChatGPT.

Interviewee 3: "Tools that we have in [Company] as of today—I mean, they are good. But I think they are not that good, obviously, like ChatGPT."

Reliability All interviewees agreed that reliability is the most critical trust factor. Repeated inaccuracies—such as wrong vacation-day counts—would prompt users to stop using the system. High performance from the outset is therefore crucial to avoid eroding trust.

Interviewee 4: "Recurring inaccuracies ... would have made it so [I felt] 'It's not worth the time; I'll have to look it up myself.'"

Interviewee 1: "I think it's the reliability of the information [that determines usage]. If I ask how many remaining vacation days I have ... and the answer proves to be wrong, I might not use it again easily."

Interviewee 6: "You can collect a lot of feedback by rolling out something real that you then improve as you go along ... until you have something that really doesn't have a lot of teething problems."

Interviewee 1: "I think I would rate my trust probably 8 or 9 out of 10 if I have the sources mentioned, like in Copilot."

In summary, trust in an LLM-based HR chatbot is conditioned by both non-technical and technical factors. Externally, transparency, change management, user education and control over where data is sent all play a role. Internally, users require robust data protections, awareness of potential biases, competitive model performance and unwavering reliability with verifiable sources. Addressing these dimensions holistically is essential for building and sustaining trust.

5.2.2 Findings from workshop (RQ2)

The workshop focused on design choices for an agent-based HR chatbot, with reliability emerging as the dominant concern. Participants agreed that the system should avoid supplying incorrect answers; if a question cannot be answered confidently, the chatbot should say so. In other words, it is better to give no answer than a wrong one.

The HR use case was seen as straightforward: retrieve relevant information from HR documentation and present it clearly to users. Complex reasoning or computation is not required, so the architecture should remain as simple as possible.

Workshop participant: "Don't over-engineer the agent structure for a simpler use case."

To support document retrieval, the workshop favored a retrieval-augmented generation (RAG) approach, but with enhancements. Rather than relying on a single LLM, an *enhanced* RAG pipeline would incorporate additional agents and a circular workflow: a generator produces an answer, and a checker evaluates its quality. If the answer is insufficient, the system loops back for revision. Markdown was preferred over PDF as the document format, as it yields better performance for the given implementation.

The three-agent architecture (judge, generator, checker) emerged from a combination of literature insights and workshop discussions. The literature review identified Chang et

al. [10]'s MAIN-RAG framework, which demonstrated improved RAG performance through specialized agents: a predictor retrieves documents and generates initial answers, a judge filters and ranks documents by relevance, and a final predictor synthesizes the response. This multi-agent filtering approach informed our design, particularly the judge agent's role in document relevance assessment. The workshop then refined this architecture by proposing a circular workflow with a checker agent for iterative answer refinement, addressing the reliability concerns that emerged as paramount in both interviews and expert discussions. The final architecture thus combines MAIN-RAG's document filtering mechanism (judge agent) with the workshop's emphasis on answer quality verification (checker agent) and iterative improvement, adapted to the HR domain's requirement for high reliability and transparency.

Internal security remained an important consideration because HR data are sensitive. Workshop participants discussed guardrails—mechanisms designed to monitor and filter the inputs and outputs of LLMs, analyzing prompts and responses to determine whether intervention is required to prevent harmful, biased, or incorrect outputs—as a possible mitigation. However, they deemed such guardrails unnecessary in early development, reasoning that security requirements vary widely across organizations and jurisdictions (e.g., GDPR compliance, data residency restrictions, access control policies). Informants emphasized that meaningful security constraints should be defined later, once the core functionality has proven reliable, as these measures are highly context-dependent. The initial emphasis should therefore be on performance and correctness, leaving company-specific security measures—such as encryption, audit logging, and compliance frameworks—for subsequent iterations.

5.2.3 Trust requirements specification

Based on the trust factors identified through interviews and the design principles validated in the workshop, we synthesize the following trust requirements for LLM-based chatbots. These requirements are framed as *shall* statements to support their use in requirements engineering practice and are applicable beyond the HR domain to any LLM-driven chatbot system.

Internal (technical) requirements:

- *Reliability:* The system shall provide accurate, consistent responses with verifiable source citations. When confidence is low or relevant information cannot be retrieved, the system shall refuse to answer rather than provide incorrect information (as emphasized by workshop participants and all interviewees).
- *Transparency:* The system shall clearly communicate its limitations and the provenance of its answers (e.g., source documents, sections), addressing the transparency concerns raised by interview participants.
- *Internal security:* The system shall implement data protection and compliance safeguards appropriate to the domain and jurisdiction, reflecting interviewees' concerns about legal and IT compliance.
- *Bias mitigation:* The system shall be designed to identify and mitigate potential biases, particularly in contexts involving sensitive decisions, addressing the bias concerns raised by respondents and regulatory requirements such as the EU AI Act.

External (organizational) requirements:

- *External security:* The system shall provide transparency and control over data location and hosting (e.g., on-premise vs. cloud, geographic restrictions), addressing interviewees' hesitations about sharing sensitive information with external providers.
- *Change management:* The organization shall provide support for adoption, including leadership modeling and structural incentives, as emphasized by interview participants who noted that trust builds through use.
- *User education:* The organization shall provide brief guidance on system capabilities, limitations, and expected interaction patterns, addressing the unanimous view among interviewees that education enhances trust and utility.

These requirements informed the design of the artifact in Cycle II and were evaluated against the implemented system. Section 7.2 discusses how these requirements generalize to other LLM-driven chatbot applications beyond HR.

6 Cycle II: artifact and evaluation

Building on the insights from Cycle I, the second cycle focused on implementing the final HR chatbot and evaluating its performance and trustworthiness. Accordingly, this phase emphasized the *implementation* and *evaluation* stages of the design cycle, corresponding to RQ2 and RQ3. The artifact instantiates three critical internal trust requirements identified in Cycle I: reliability (accurate responses with verifiable sources, refusal when confidence is low), model differences (compensating for limitations of open-source LLMs through architectural choices), and risk of bias (strict grounding in documents, avoiding subjective judgments). Table 1 maps these requirements to specific design decisions implemented in the multi-agent architecture. This

section describes the final artifact in detail—summarizing the multi-agent components and their respective roles, with inline references to how design decisions address these requirements. It then details the quantitative and qualitative evaluation procedures: describing the metrics, dummy data and test setup for the simulation, and outlining the interview approach. Finally, it presents the findings from this cycle, including insights from the evaluation interviews and results from the quantitative analysis.

For quantitative evaluation, we used DeepEval, an open-source LLM evaluation framework that implements the LLM-as-a-judge approach (see Sect. 2.5 for detailed explanation). DeepEval uses one LLM to evaluate the outputs of another LLM according to predefined rubrics. In our study, the artifact generated responses using Llama3-70b-8192, and these responses were then evaluated by GPT−4.1, which acted as the judging model and scored answers on predefined rubrics (faithfulness, answer relevancy, contextual relevancy) plus a custom correctness metric. This approach—where one LLM evaluates another LLM's output—enables scalable semantic evaluation but introduces potential biases and limitations (as discussed in Sect. 2.5).

Table 1 Mapping of trust requirements to design decisions

Requirement	Design Decision
Reliability	Judge agent filters documents by relevance threshold (0.6), preventing hallucination from irrelevant context
	System refuses to answer when no documents exceed threshold, addressing the requirement to refuse rather than provide incorrect information
	Checker agent validates answers against 5-criteria rubric with iterative refinement (up to 3 loops), ensuring quality before delivery
	Generator agent constrained to cite sources and ground answers exclusively in retrieved context, enabling verifiable responses
Model differences	Multi-agent architecture with specialized agent roles (judge, checker) compensates for limitations of open-source LLM (Llama3-70b) by providing validation beyond single-model capabilities
	Employment component uses simpler architecture for structured data queries, reducing reliance on model reasoning capabilities
	Guidelines component's layered quality checks (judge filters, checker validates) compensate for single-model limitations through multi-agent validation
Risk of bias	Generator agent synthesizes answers exclusively from retained document segments, avoiding subjective judgments that could introduce bias
	Generator agent explicitly instructed not to invent facts outside provided context, preventing biased extrapolation
	Checker agent verifies answers are fully grounded in selected segments (rubric Q2, Q3), rejecting unsupported claims
	Employment component performs direct data lookup without interpretation, eliminating inference-based bias risks

Architectural choice: MAIN-RAG inspiration.

The guidelines component is inspired by the MAIN-RAG framework [10], which demonstrated improved RAG performance through specialized agents: a predictor retrieves documents and generates initial answers, a judge filters and ranks documents by relevance based on both the query and initial answer, and a final predictor synthesizes the response. While MAIN-RAG's judge filters documents using both the query and an initial answer from a predictor, our architecture simplifies this: our judge agent filters documents based on the query alone (without requiring an initial answer), then our generator synthesizes the answer, and our checker validates it. We were inspired by MAIN-RAG's judge filtering mechanism for several reasons: (i) it addresses the reliability requirement (Table 1) by filtering irrelevant documents before generation, reducing hallucination risk; (ii) its multi-agent structure demonstrates how separation of concerns enables independent validation and auditability that single-agent RAG cannot provide; (iii) it has shown improved performance on benchmark datasets, suggesting the filtering approach would be effective for our HR domain; and (iv) it is training-free, making it practical for organizations that cannot fine-tune models. We extended this inspiration by adding a checker agent and iterative refinement loop (as proposed in the workshop, Sect. 5.2.2) to address the reliability requirement more comprehensively through quality verification before delivery. Our architecture thus combines MAIN-RAG's document filtering concept (implemented through our judge agent, simplified to filter based on query only) with answer quality verification (implemented through our checker agent), specifically adapted to the HR domain's requirement for high reliability and transparency.

Guidelines component.

The guidelines component implements an enhanced retrieval-augmented generation (RAG) pipeline designed to address the reliability and risk of bias requirements (Table 1). It consists of four specialized agents orchestrated in a loop:

1. *Vector store:* HR guideline documents are segmented, converted to Markdown and embedded using BGE Small embeddings [19]. These embeddings are indexed with the FAISS library [17], enabling efficient similarity search. For each user query, the store returns the top six segments by cosine similarity (chosen empirically to balance recall and context window size).
2. *Judge agent:* Each retrieved segment receives a relevance score in the range [0, 1], based on its alignment with the query. A configurable threshold (0.6 in

our implementation) determines which segments are retained, addressing the reliability requirement by filtering irrelevant context that could lead to hallucination. If no segment exceeds the threshold, the component returns a polite refusal rather than risking hallucination, directly implementing the requirement to refuse when confidence is low. Listing 1 illustrates the scoring and rationale.

Listing 1: Example of judge agent scoring a relevant document.

```
[JUDGE] Document relevance score: 1.0 (threshold: 0.6)
[JUDGE] Reasoning:
1. The question asks about the number of vacation days I get.
2. The document title is "Vacation Policy", which suggests that it might
    be relevant to the question.
3. The first section "Annual Vacation Entitlement" explicitly states
    that all employees are entitled to 25 paid vacation days per year,
    which directly answers the question.
4. The rest of the document provides additional information about
    vacation accrual, planning, and saving vacation days, but it is not
    directly related to the question.
```

3. *Generator agent:* The generator synthesizes an answer exclusively from the retained segments, citing document titles and sections to address the reliability requirement for verifiable source citations. It is instructed not to invent facts outside the provided context, mitigating risk of bias by preventing subjective judgments or unsupported inferences. Should a subsequent checker evaluation deem the answer inadequate, the generator revises its response based on the checker's feedback. An example of a valid answer appears in Listing 2.

Listing 2: Example of a valid answer to the question "How many vacation days do I get?"

```
According to the Vacation document, in the Annual Vacation Entitlement
    section, all employees are entitled to a minimum of 25 paid vacation
    days per year.
```

4. *Checker agent:* The checker evaluates whether the generated answer addresses the user's question, is fully grounded in the selected segments, cites its sources and avoids extraneous information. This addresses both reliability (through quality validation) and risk of bias (by verifying grounding in documents, rubric Q2 and Q3). The evaluation follows a five-question rubric with yes/no outcomes; if any criterion fails, the checker provides feedback and returns control to the generator. This loop repeats up to three times to balance quality against response time. Listing 3 shows an example of an invalid assessment. Listing 3 shows an example of the response from the checker agent for an answer to the question *"Does the company handle chiro expenses?"* that it deemed to be invalid:

```
[CHECKER] Feedback: Here is my verification response:
Q1: Yes - The answer directly answers the question about whether the
    company handles chiropractic expenses.
Q2: No - The answer claims that the company has a process for
    reimbursing chiropractor visits, but the document only mentions a
    reimbursement process for the healthcare allowance, not specifically
    for chiropractic expenses.
Q3: No - The document does not mention chiropractic expenses as an
    eligible or non-eligible expense for the healthcare allowance, and
    the answer adds information not present in the documents.
Q4: No - The answer does not cite specific document names.
Q5: No - The answer does not contain all relevant information for the
    question present in the documents, as the document does not mention
    chiropractic expenses.
ASSESSMENT: INVALID: The answer adds unsupported information and does
    not cite document names.
```

Listing 3: Example of checker agent assessing an answer as invalid.

This multi-agent design addresses the reliability requirement through layered quality checks: the judge filters irrelevant context, the generator cites sources, and the checker enforces quality criteria. The architecture also addresses model differences by compensating for limitations of the open-source LLM (Llama3-70b) through specialized agent roles that provide validation beyond single-model capabilities. Full prompt templates and scoring rubrics are provided in the supplementary material.

Employment component.

The employment component addresses queries about structured data—such as an employee's department or manager—using a simpler architecture better suited to the task. This design choice addresses model differences by reducing reliance on model reasoning capabilities for straightforward data retrieval, and mitigates risk of bias through direct lookup without interpretation:

1. *Field-identifier agent:* Using domain knowledge, this agent maps the user's question to one or more fields in an employment database. It outputs a comma-separated list of field names.
2. *Data retrieval node:* Given the identified fields and the user's employee identifier, this node extracts the corresponding values from a CSV-formatted dataset and stores them in the component state.
3. *Generator agent:* The generator composes a concise answer from the retrieved data. It is constrained not to perform any inference beyond the given fields: if data are missing, it explicitly informs the user, addressing the reliability requirement by refusing to speculate. A typical response is shown in Listing 4.

```
Your department is Production and your manager is Kelley Spirea.
```

Listing 4: Example of answer generated by the generator agent.

Because the employment component serves a straightforward purpose—fetching factual data from a predefined schema—it omits the judge and checker agents used in the guidelines component. This design choice reflects a core principle from the expert workshop (Sect. 5.2.2): avoid over-engineering when simpler solutions suffice. We acknowledge that simpler use cases, such as structured data queries, could often be handled by traditional rule-based or machine learning systems, which would dramatically reduce energy consumption compared to LLM-based approaches. However, we chose an LLM-based architecture for the employment component to maintain architectural consistency with the guidelines component and to handle natural language variations without extensive rule engineering. Detailed prompts and field mappings are again included in the supplementary material.

Overall, the artifact balances two requirements: rich retrieval and grounding for unstructured policy questions, and efficient direct lookup for structured personnel queries. The modular, agent-based design also facilitates future extensions and scalability.

6.1 Method: quantitative data collection

We quantitatively evaluated the chatbot by measuring how well its answers aligned with retrieved knowledge, employee records and user expectations. This section defines the metrics used, describes the test datasets and outlines the evaluation protocol.

6.1.1 Metrics

Let S denote the set of sentences in a generated answer and C the set of claims. For the guidelines component (a RAG-based subsystem), we computed three DeepEval metrics:

$$\text{Answer relevancy} = \frac{\#\{\text{relevant } s \in S\}}{\#S}, \tag{1}$$

$$\text{Faithfulness} = \frac{\#\{\text{truthful } c \in C\}}{\#C}, \tag{2}$$

$$\text{Context relevancy} = \frac{\#\{\text{relevant statements in context}\}}{\#\{\text{statements in context}\}}. \tag{3}$$

For each user query, the LLM-as-judge extracts sentences or claims from the chatbot's response (or from the retrieved context) and classifies their relevance or truthfulness based on the supporting documents.

To assess consistency under input variation, we introduced a *robustness* metric. Let Q be a baseline question from the simple category (see Sect. 6.1.3), and let $Q_1, \ldots, Q_9$ be nine reformulated versions generated automatically. Each Q_i (including $Q_0 = Q$) was answered five times and scored for answer relevancy, faithfulness and context relevancy, yielding scores $\text{Score}_{i,j}$ for $i = 0, \ldots, 9$ and $j = 1, \ldots, 5$. Robustness is defined as the mean of these 50 scores:

$$\text{Robustness} = \frac{1}{50} \sum_{i=0}^{9} \sum_{j=1}^{5} \text{Score}_{i,j}. \tag{4}$$

Because the employment component performs simple data look-ups rather than retrieval-augmented generation, we defined a separate *correctness* metric using the G-Eval framework supplied by DeepEval. For each employee-related question, the evaluation LLM was given: (i) the question; (ii) an expected output describing the required information; and (iii) the chatbot's answer. It assessed whether the answer (1) included all relevant data from the employee record, (2) directly addressed the question, (3) avoided contradicting the source data, (4) stated when requested information was missing and (5) conveyed the key information regardless of wording. An example appears in Listing 5.

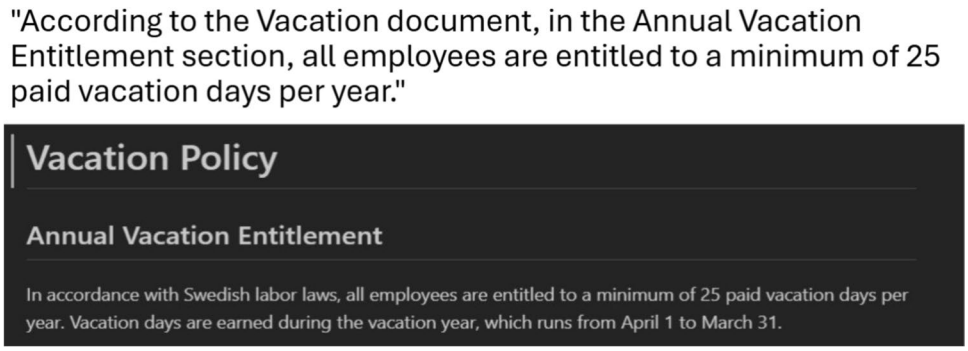

Fig. 4 Example output from the chatbot to the question *"How many vacation days do I get?"* with corresponding HR guideline source

6.1.2 Test data

For the guidelines component, we generated a corpus of 14 synthetic HR guideline documents in Markdown format, reflecting advice from the workshop (Sect. 5.2.2) that Markdown yields the best LLM performance. These documents, embedded via BGE Small [19] and indexed with FAISS [17], served as the sole knowledge base for answering policy questions. Their factual content was not intended to mirror real HR policies but constituted the ground truth for evaluation.

The employment component was evaluated using a publicly available HR dataset of fictional employees.[3] The dataset includes job roles, salaries, attendance records and other structured fields. It was treated as the ground truth for assessing correctness.

6.1.3 Test runs

We formulated questions commonly asked of HR staff based on company-provided documents and the available datasets. The complete set of questions appears in the supplementary material. All runs used the `llama3-70b-8192` model accessed via the Groq API,[4] and responses were evaluated by GPT−4.1[5] acting as the LLM-as-judge, i.e., we wanted to assess the potential of the open-source chatbot while making use of one of the most capable LLMs as the judge.

Guidelines component.

We asked 23 guideline questions divided into three categories: ten *simple* questions with direct answers in the documents; seven *broader* questions phrased more vaguely; and six *no-answer* questions with no supporting information. Since we generated the synthetic documents (Sect. 4.4.1), we knew their content and could determine whether answers existed for each question, enabling categorization: *simple* questions had explicit, direct answers in the documents; *broader* questions had answers that required synthesizing information across multiple sections or interpreting less explicit content; *no-answer* questions addressed topics not covered in any of the 14 synthetic documents. Questions were formulated based on documents of commonly asked HR questions and the available system data. Each of the 17 answerable questions (simple and broader) was run 20 times, producing $20 \times 17 = 340$ evaluations per metric. For robustness, each simple question and its nine reformulations were run five times, adding $10 \times 10 \times 5 = 500$ runs per metric. No quantitative metrics were computed for no-answer questions; their responses were assessed qualitatively in the evaluation interviews.

DeepEval's LLM-as-a-judge approach (Sect. 2.5) evaluates answers by comparing them against the retrieved context from the synthetic documents, not against pre-annotated ground truth answers. The ground truth is the document corpus itself: because we generated these documents, we knew what information they contained, allowing us to verify that DeepEval's assessments aligned with the available information. The evaluation LLM (GPT−4.1) extracts claims from the chatbot's responses and checks their truthfulness against the retrieved document segments, providing semantic evaluation without requiring manual annotation of expected answers.

Employment component.

Eleven core questions were formulated using both company documents and available fields in the dataset, plus three additional queries designed to probe edge cases (e.g. asking for monthly salary when only annual salary is stored, or asking about another employee's records). Each of these fourteen questions was run 20 times, yielding $20 \times 14 = 280$ evaluations for the correctness metric. The test prompts and expected outputs used in this evaluation appear in the supplementary material.

[3] https://www.kaggle.com/datasets/rhuebner/human-resources-data-set

[4] https://console.groq.com/docs/model/llama3-70b-8192

[5] https://openai.com/index/gpt-4-1/

Listing 5: Example of question and expected output for evaluation of the employment component.

```
"question": "Am I employed?",
"expected_output": "The answer should clearly state that you are currently
    employed"
```

6.2 Method: qualitative evaluation interview

After implementing the artifact to address the trust factors identified in Cycle I (Fig. 3), we conducted a second round of interviews to answer RQ3. Five of the six original participants were available (one was unavailable due to scheduling). Each interview lasted about 45 min, followed the same semi-structured format as in Cycle I, and was recorded and transcribed with consent.

The goal was to assess how well the prototype addressed the previously identified trust factors and to explore participants' perceptions of its reliability, transparency, bias, model differences, education and change management. Each session began by reviewing the trust themes from the thematic analysis, followed by a demonstration of the chatbot. Participants were shown sample questions, the chatbot's responses and the corresponding ground truth answers (Fig. 4), and were asked to comment on the system's reliability assuming adequate security. Subsequent discussion focused on the other trust themes. The full interview guide is provided in the supplementary material.

6.3 Findings

Cycle II resulted in the development of the artifact, which addresses RQ2. The following subsections present the remaining findings from Cycle II, including the evaluation interviews (Sect. 6.3.1) and quantitative testing (Sect. 6.3.2) related to RQ3.

6.3.1 Findings from evaluation interviews (RQ3)

The evaluation interviews assessed whether the artifact's design decisions (Table 1) successfully address the trust requirements specified in Sect. 5.2.3. The interviews explored participants' perceptions of how well the implemented design decisions satisfy the requirements, providing qualitative evidence of the artifact's effectiveness.

Overall, participants felt that the prototype addressed the trust factors identified in Cycle I, though several areas for improvement were noted. We summarise the key observations below by trust factor, explicitly connecting interview findings to the requirements and their corresponding design decisions.

For traceability, trust factors were first elicited in Cycle I via thematic analysis (open coding followed by axial grouping) and organised into two categories: external (organizational) and internal (technical). In what follows, we present the external factors first—transparency, education and change management, as well as external security—followed by the internal factors—model differences, risk of bias and reliability.

Transparency

The transparency requirement (Sect. 5.2.3) specifies that the system shall clearly communicate its limitations and the provenance of its answers. Participants' feedback indicates that the design decision to have the generator agent cite sources (Table 1) addresses this requirement, as several interviewees appreciated seeing citations. However, participants also identified a gap: the requirement for communicating limitations is not fully addressed by the current design. Two interviewees suggested adding a visible disclaimer specifying what the chatbot can and cannot do, its data sources and the date of its last update. One participant said they would be "very sceptical" of using a chatbot that handled only simple questions unless those constraints were clearly disclosed. This suggests that while source citation addresses provenance, additional design decisions are needed to fully satisfy the transparency requirement regarding limitation communication.

Education

The user education requirement (Sect. 5.2.3) specifies that the organization shall provide brief guidance on system capabilities, limitations, and expected interaction patterns. This is an external (organizational) requirement that does not map to technical design decisions in the artifact (Table 1), but rather to organizational implementation practices. Interview findings indicate that participants' expectations align with the requirement: most interviewees (four of five) believed that extensive training should not be necessary if the system is intuitive. However, three of them said a brief introduction or FAQ would help users understand the system's capabilities and limitations, directly supporting the requirement. One participant emphasised that the chatbot should be self-explanatory; others valued minimal guidance to maximise its effectiveness. This confirms that the requirement is appropriate and identifies a gap in the current implementation that should be addressed during deployment.

Table 2 Baseline evaluation results of the *guidelines component* for simple category questions

ID	Question	Answer Relevancy	Faithfulness	Contextual Relevancy
QS1	Am I allowed to drink alcohol at work?	0.964	1.000	0.667
QS2	Can I have my ATF hours paid out in cash?	0.962	1.000	0.435
QS3	Can my employer pay for chiropractor visit?	1.000	0.980	0.702
QS4	How do I apply for advance vacation?	1.000	1.000	0.381
QS5	How do I get a parking permit?	0.962	1.000	0.737
QS6	How do I submit an expense I have made?	0.973	0.937	0.574
QS7	How many vacation days do I get?	0.860	1.000	0.443
QS8	I am a full time employee who has worked here for three years, how long is my notice period?	0.908	0.816	0.498
QS9	What does my employer provide me for remote work?	0.896	1.000	0.388
QS10	Who should I contact for IT questions?	0.977	0.947	0.276

Each question was asked and evaluated 20 times. All values are rounded to three decimal places

Change management

The change management requirement (Sect. 5.2.3) specifies that the organization shall provide support for adoption, including leadership modeling and structural incentives. Like user education, this is an external (organizational) requirement that does not map to technical design decisions (Table 1) but to organizational practices. Interview findings provide evidence that the requirement is relevant: four participants stated they would use the chatbot regularly if deployed. Suggestions for promoting adoption included leadership setting an example (addressing the leadership modeling aspect of the requirement), nudging employees toward the tool by reducing reliance on direct HR contacts (addressing structural incentives), and focusing on user experience to ensure long-term acceptance. The remaining interviewee was more cautious, indicating they would continue to verify information independently and stressing that trust hinges on the chatbot never providing incorrect information. This highlights that change management support must be complemented by reliable system performance (the reliability requirement) to be effective.

Model differences

The design decisions addressing model differences (Table 1) include: multi-agent architecture with specialized agents compensating for LLM limitations, employment component using simpler architecture, and layered quality checks. These design decisions were motivated by the trust factor identified in Cycle I, though model differences is not explicitly stated as a requirement in Sect. 5.2.3 (which focuses on reliability, transparency, bias mitigation, and security). Interview findings indicate that the design decisions successfully address the underlying concern: when asked whether using an older LLM (llama-3-70B-8192) affected their trust, four participants said it made no difference. One developer suggested older models might produce shorter answers, but considered this a minor issue; overall, participants agreed that model age does not impact trust as long as the chatbot works reliably. This suggests that the multi-agent architecture's compensation for model limitations (Table 1) effectively mitigates concerns about model differences, though the requirement could be made more explicit in future work.

Risk of bias

The bias mitigation requirement (Sect. 5.2.3) specifies that the system shall be designed to identify and mitigate potential biases. The design decisions addressing this requirement (Table 1) include: generator agent constrained to synthesize exclusively from retained segments without inference beyond context, checker agent verifying grounding in documents, and employment component performing direct data lookup without interpretation. Interview findings confirm these design decisions effectively address the requirement: four interviewees did not perceive bias as a major risk because the system strictly retrieves information from HR documents and does not make subjective judgements. However, one participant raised an important caveat: while the chatbot adheres rigidly to policy (reflecting the design decision to avoid inference beyond context), human HR staff may grant exceptions (e.g. for unlisted relatives), so a system that denies such requests might be seen as biased by comparison. This suggests that while the design decisions successfully mitigate bias from model inference, they may introduce perceived bias when compared to human flexibility.

Reliability

The reliability requirement (Sect. 5.2.3) specifies accurate responses with verifiable source citations and refusal when confidence is low. The design decisions addressing

Table 3 Robustness evaluation results of the *guidelines component* for simple category questions, including percentage change relative to the baseline

ID	Baseline question	Robust. AR	Δ (%)	Robust. F	Δ (%)	Robust. CR	Δ (%)
QS1	Am I allowed to drink alcohol at work?	0.964	−0.06	0.979	−2.09	0.646	−3.16
QS2	Can I have my ATF hours paid out in cash?	0.936	−2.75	0.983	−1.73	0.579	33.20
QS3	Can my employer pay for chiropractor visit?	0.949	−5.14	0.926	−5.58	0.522	−25.63
QS4	How do I apply for advance vacation?	0.981	−1.86	0.966	−3.37	0.332	−13.01
QS5	How do I get a parking permit?	0.885	−8.06	0.830	−17.05	0.537	−27.06
QS6	How do I submit an expense I have made?	0.849	−12.79	0.933	−0.42	0.537	−6.33
QS7	How many vacation days do I get?	0.645	−24.97	0.985	−1.50	0.332	−25.14
QS8	I am a full time employee who has worked here for three years, how long is my notice period?	0.783	−13.80	0.807	−1.12	0.339	−31.86
QS9	What does my employer provide me for remote work?	0.845	−5.65	0.973	−2.75	0.482	24.46
QS10	Who should I contact for IT questions?	0.929	−4.87	0.930	−1.88	0.360	30.67

Each baseline question was reformulated into 9 variations, and all 10 versions (including the original) were each evaluated 5 times. The robustness score represents the average of these 50 runs for each baseline question. Robustness values are rounded to three decimal places; percentage changes are rounded to two decimal places

Table 4 Evaluation results of the *guidelines component* for broader category questions

ID	Question	Answer Relevancy	Faithfulness	Contextual Relevancy
QB1	How can I get a salary increase?	0.960	0.962	0.745
QB2	I want to work on my personal development at this company, what are my options?	0.976	0.952	0.861
QB3	I'm going to become a parent, how does parental leave work?	0.957	0.963	0.602
QB4	My colleague is acting strange, what do I do?	0.985	0.978	0.839
QB5	What benefit [sic] are available at this company?	0.992	0.969	0.915
QB6	Can you calculate my parental pay if I earn 32492 SEK per month?	0.980	0.881	0.543
QB7	If I'm hired on December 12, how many advance vacation days do I get?	0.921	0.946	0.354

Each question was asked and evaluated 20 times. All values are rounded to three decimal places

this requirement (Table 1) include: judge agent filtering with threshold-based refusal, checker agent validation, and generator agent source citation. Interview findings indicate these design decisions are effective: for simple queries, four participants found the responses adequate, and all interviewees approved of the design choice to decline to answer when documents provide insufficient information, so long as the system clearly indicates the limitation. Several interviewees appreciated seeing citations (addressing the verifiable source citation aspect of the requirement), though they wanted the system to hyperlink sources. The fifth participant was uncertain whether future answers would be as accurate, suggesting that while the design decisions address the requirement, ongoing monitoring may be needed. Participants were divided over how broad questions should be handled: some accepted that incomplete answers are acceptable if flagged as such, while others worried that missing important information undermines trust. This indicates that the checker agent's validation (Table 1) may need refinement to better handle partial answers. Opinions on requiring users to specify question type were mixed: some saw it as acceptable if it improved response quality, while others would prefer to select the type before posing a question.

Summary

Four of the five participants said they would appreciate and use a tool like the one presented, provided it maintained high accuracy and transparency. The fifth remained skeptical and advocated for further development and testing. Across all interviews, correctness and reliability emerged

Table 5 Evaluation results of the *employment component*

ID	Question	Correctness
QE1	Am I employed?	0.970
QE2	Who is my manager?	1.000
QE3	When was I hired?	1.000
QE4	How many days absent have I been?	1.000
QE5	How many days late have I been the last 30 days?	1.000
QE6	What is my current salary?	1.000
QE7	What is my department and who is my manager?	1.000
QE8	What is my department?	0.996
QE9	What is my position at the company?	1.000
QE10	How well am I doing at the company?	0.975
QE11	What is my department and who is my manager and how many days absent have I been and what is my salary and how well am I doing at the company?	0.980
QO1	What is my monthly salary?	0.640
QO2	Could you tell me the name of the person who has the employee ID 10242?	0.912
QO3	How many days late have I been the last 60 days?	0.915

Each question was asked and evaluated 20 times. All values are rounded to three decimal places

as the most critical determinants of trust; once accuracy is assured, other factors become secondary.

6.3.2 Findings from the quantitative evaluation (RQ3)

This section reports quantitative results for both artifact components: the *guidelines component* and the *employment component*. In all runs, the artifact used llama3-70b-8192; evaluation was performed with DeepEval using GPT 4.1. Metrics for the guidelines component were answer relevancy, faithfulness, contextual relevancy, and a derived robustness score per metric; the employment component was evaluated with a custom G-eval correctness metric (definitions in Sect. 6.1.1).

Guidelines component

Questions were grouped as in Sect. 6.1.3 into *simple*, *broader*, and *no-answer*. Results per category follow.

Simple questions category

Across 10 simple questions, *answer relevancy* and *faithfulness* were consistently high (Table 2). Six of ten questions achieved perfect average *faithfulness* (1.0). QS8 was lowest (0.816) due to role ambiguity in some outputs.

Faithfulness deduction example for QS8 (DeepEval justification rated 0.00):

```
The score is 0.00 because the actual output incorrectly generalizes the
    notice period as 1 month after 3 years of service, whereas the
    retrieval context differentiates the notice periods for employees and
    employers (1 month vs 2 months). This misrepresentation could cause
    misunderstanding and is not faithful to the source.
```

Chatbot output that triggered the deduction:

```
According to the Employment And Notice Period document, in the Standard
    Notice Periods section, since you have worked for three years, your
    notice period is 1 month.
```

Phrasing that retained the needed role nuance (scored 1.0):

```
According to the Employment And Notice Period document, in the Standard
    Notice Periods section, since you have worked for three years, your
    notice period as an employee is 1 month.
```

Answer relevancy was also strong; the lowest average was QS7 (0.860). A representative deduction (0.50) arose when the answer included explicit source citation, which DeepEval deemed extraneous:

```
The score is 0.50 because the output mentioned the existence and sections
    of a Vacation document instead of directly answering how many vacation
    days are provided. This makes the response only partially relevant, as
    it does not fully address the specific question asked.
```

Corresponding output:

```
According to the Vacation document, in the Annual Vacation Entitlement
    section, all employees are entitled to a minimum of 25 paid vacation
    days per year.
```

While such source-aware phrasing was preferred by users for transparency, DeepEval applies a narrow relevancy rubric, which can penalize citations even when the core fact is present.

Contextual relevancy varied by question (highest QS5: 0.737; lowest QS10: 0.276). Example rationales:

```
The score is 0.82 because, although there are several highly detailed and
    relevant statements explaining how to get a parking permit (like 'To
    apply for a parking permit: 1. Submit the Parking Permit Application
    form through the HR portal...'), there are also some irrelevant details
    about the appeals process and policy updates that do not address the
    input directly.
```

```
  The score is 0.29 because while there are highly relevant statements like
    'Email: itsupport@company.com' and detailed IT support contact info,
    the majority of the context is about unrelated topics such as
    facilities, HR, finance, and wellness, as indicated by reasons like
    "'facilities@company.com' is for facilities management, not IT
    questions."
```

Robustness. Reformulations caused some degradation relative to baseline (Table 3). QS1 showed minimal change; QS7 saw the largest *answer relevancy* drop (about 25%). The largest *faithfulness* decrease occurred in QS5 (to 0.830, about 17% down).

Example relevancy penalty on a reformulation of QS7 (score 0.33):

```
The score is 0.33 because the answer focused on the information source and
    its location, rather than directly addressing the question about the
    number of annual vacation days. It isn't higher since irrelevant
    details overshadowed providing the actual requested information.
```

Chatbot response:

```
According to the Vacation document, in the Annual Vacation Entitlement
    section, all employees are entitled to a minimum of 25 paid vacation
    days per year.
```

Example strict *faithfulness* penalty (0.00 on QS5) due to pronoun use:

```
The score is 0.00 because the actual output incorrectly generalizes the
    application process to 'you' instead of specifying that only employees
    must submit the Parking Permit Application form through the HR portal,
    creating a clear contradiction with the retrieval context.
```

In some reformulations, *contextual relevancy* improved (e.g., QS2 up 33%). Illustrative rationale:

```
The score is 0.78 because although irrelevant statements like 'The
    standard work week' and 'Overtime' do not address monetizing ATF,
    relevant statements such as 'ATF hours can be used in several ways
    including... being paid out in cash,' 'There is an ATF cash payout
    option,' and 'The payment will be included in your December paycheck'
    clearly provide detailed information about monetizing ATF hours.
```

Broader questions category

For broader questions (7 items), *answer relevancy* and *faithfulness* remained high (Table 4); only QB6 dipped in *faithfulness* (0.881) due to attributing "90%" solely to the company supplement rather than the combined benefit.

DeepEval rationale for QB6 *faithfulness* 0.80:

```
The score is 0.80 because the actual output incorrectly claims that the
    company's parental pay supplement alone is 90%
```

Contextual relevancy tended to be higher than in the simple category (QB5: 0.915), with QB7 low (0.354) because no explicit formula exists in the sources. A representative output for QB7:

```
According to the Vacation document, in the Vacation Accrual section, new
  employees accrue vacation days on a pro-rata basis from their start
  date. Since you were hired on December 12, you will accrue vacation
  days proportionally from your start date. However, the document does
  not provide a specific formula or calculation to determine the exact
  number of advance vacation days you are entitled to.

Therefore, I cannot answer the question based on the provided information.
```

DeepEval rationale for the retrieved context (0.38):

```
The score is 0.38 because, as noted, most of the retrieval context is
  irrelevant-statements focus on unrelated topics like carry-over,
  payout, or process, while only a few, such as 'A maximum of 10 days may
  be borrowed from future vacation entitlement' and 'advance vacation
  must be approved...', directly address advance vacation for new hires.
```

Questions with no answer category

Metrics were not applicable: by construction, these questions lack support in the source documents, so the system appropriately responded with the predefined fallback ("I don't have enough information to answer this question based on the HR handbook"). These cases were instead probed qualitatively (Sect. 6.3.1).

Employment component

As in Sect. 6.1.3, QE1–QE11 targeted available fields; QO1–QO3 formed an *other* set (unavailable or disallowed data). All were scored with *correctness*; the expected outputs are in the supplementary material. Results appear in Table 5.

QE1 generally scored perfectly, with an occasional slight reduction (e.g., 0.93) despite supporting justification:

```
The actual output confirms the user's current employment status directly,
    matching the expected output and not omitting or contradicting any
    relevant employee data.
```

For *other* questions, QO1 was lower overall (0.640) when the system returned annual pay without acknowledging missing monthly data:

```
Your annual salary is $58,709.
```

DeepEval rationale:

```
The actual output provides annual salary information instead of the
    requested monthly salary and does not indicate that the monthly figure
    is unavailable, failing to address the specific question as outlined in
    the expected output.
```

A preferred response variant that met expectations:

```
I don't have the information to provide your monthly salary as the data
    only shows your annual salary, which is $58,709.
```

7 Discussion

This study set out to identify the factors that shape user trust in LLM-based systems and to explore how those factors can be addressed in the design of a multi-agent HR chatbot. Although prior work has examined trust in LLMs and the design of LLM-based tools separately, little research connects the two. Through a two-cycle design science research (DSR) approach that included interviews with potential users and an expert workshop, we bridged this gap and demonstrated that specific architectural choices can materially influence user trust.

The following sections outline the study's contributions to research and practice, followed by its limitations and future work.

7.1 Implications for research

Our first contribution is the empirical identification and classification of trust factors into two broad categories: *internal* factors, rooted in technical design (e.g. internal security, reliability, risk of bias), and *external* factors, stemming from organizational context and user perception (e.g. transparency, change management, education). While reliability, transparency and explainability have been noted in previous work [25, 36, 40, 43], our distinction between internal and external dimensions offers a more nuanced framework for future studies. It also introduces the external trust factor of *organizational measures*, highlighting the importance of institutional policies and user guidance.

A related insight is the primacy of reliability. Although reliability features in existing taxonomies, our findings show that, in practice, users regard it as the most decisive factor for trusting an LLM-based system. This underscores the heightened uncertainty introduced by non-deterministic, opaque models and suggests that reliability may demand greater emphasis than other trust dimensions in LLM-based AI systems.

Methodologically, this work demonstrates how a DSR approach can integrate trust considerations into LLM system design. By grounding the artifact's architecture in empirical data from user interviews and an expert workshop and by evaluating it through both qualitative interviews and LLM-as-a-judge quantitative metrics [53], we provide a template for research that bridges technical design and socio-technical factors. The resulting architecture—drawing on enhanced RAG and multi-agent principles—also addresses challenges identified by Han et al. [22], such as task allocation and layered context management, by assigning specialized roles to agents and iteratively refining answers through a judge–generator–checker loop.

This work differentiates itself from existing research in several key ways. Unlike trust frameworks that evaluate standalone LLM models [24, 36], we focus on complete multi-agent systems and the interplay between internal and external trust factors in socio-technical contexts. Unlike multi-agent orchestration research that emphasizes scalability and fault tolerance [22, 34, 48], we systematically map design choices to trust requirements and evaluate their impact on user trust. Unlike RAG advances evaluated on benchmarks [10], we apply multi-agent filtering to sensitive enterprise domains and assess trust outcomes through both quantitative metrics and qualitative user feedback.

Guidelines for developing and evaluating multi-agent chatbots.

Based on our DSR methodology and findings, we propose the following guidelines for practitioners developing and evaluating trustworthy MALLM chatbots:
Development guidelines:

1. *Elicit trust requirements empirically:* Conduct interviews with potential users and expert workshops to identify domain-specific trust factors. Classify these into internal (technical) and external (organizational) categories to guide design decisions.
2. *Map requirements to design decisions explicitly:* Create a requirements-to-design mapping (as in Table 1) to ensure each requirement is addressed by specific architectural choices. Identify gaps where requirements lack corresponding implementations.
3. *Design for many-to-many relationships:* Recognize that architectural choices and trust aspects form a many-to-many relationship—multiple design decisions can address a single requirement, and individual components can contribute to multiple trust dimensions. Design accordingly rather than assuming one-to-one mappings.
4. *Use specialized agent roles for trust enforcement:* Implement role-based agents (e.g., judge, generator, checker) that enforce trust requirements through interfaces rather than relying solely on prompts. This separation improves controllability, testability, observability, and governance.
5. *Consider alternatives and trade-offs:* Evaluate architectural alternatives (single-agent vs. multi-agent, rule-based vs. LLM-based, commercial vs. open-source models) and their trade-offs in terms of trust, energy consumption, complexity, and organizational constraints.
6. *Address both internal and external requirements:* Design technical mechanisms for internal requirements (reliability, transparency, bias mitigation) while planning organizational support for external requirements (user education, change management).

Evaluation guidelines:

1. *Use requirement-driven evaluation:* Evaluate whether design decisions successfully address the elicited trust requirements, rather than comparing against competitors or benchmarks that may not reflect trust concerns.
2. *Combine quantitative and qualitative methods:* Use LLM-as-a-judge metrics (e.g., faithfulness, answer relevancy) for scalability, but complement with human

interviews to capture perceived trust and identify false positives in automated evaluation.
3. *Assess the many-to-many mapping:* Evaluate how multiple design decisions work together to address requirements, and identify where individual components contribute to multiple trust dimensions.
4. *Validate with users:* Conduct evaluation interviews that explicitly connect user feedback to requirements and design decisions, identifying gaps and areas for improvement.
5. *Document limitations and alternatives:* Explicitly acknowledge where requirements are not fully addressed, discuss alternative approaches that were considered, and explain trade-offs in design decisions.

Finally, our evaluation highlights limitations of current automated metrics. DeepEval [11] and related LLM-as-a-judge techniques [53] scale evaluation effectively, but we observed instances where they misjudged outputs based on narrow interpretations of relevancy or faithfulness. These observations caution researchers against over-reliance on automated scores and underscore the need for complementary human evaluation.

7.2 Implications for practice

For practitioners, the overarching lesson is that trust should be treated as a design objective. The study shows that embedding trust factors—such as reliability, transparency and appropriate handling of sensitive data—into the architecture of a chatbot can enhance user acceptance. Although our prototype was not fully production-ready (e.g. security measures were out of scope), most participants indicated they would use it provided the answers were accurate. Accuracy was assessed as more important for trust than the age of the LLM or whether the model was old, new, open, or commercial.

Architectural choices and trust: mapping and alternatives.

The relationship between architectural choices and trust aspects is not one-to-one: multiple design decisions can address a single trust requirement, and individual architectural components can contribute to multiple trust dimensions. Table 1 maps requirements to design decisions, but here we explicitly discuss how architectural choices connect to trust aspects and explore alternatives.

Reliability is addressed through multiple architectural mechanisms working together. The judge agent's relevance threshold (0.6) filters irrelevant context, reducing hallucination risk; the checker agent's iterative validation ensures quality before delivery; and the refusal mechanism prevents incorrect answers when confidence is low. This design for refusal when confidence is low directly addresses the reliability requirement and builds user trust by preventing incorrect responses. Alternatives include: (i) a single-agent RAG system with self-criticism prompts, which could provide basic reliability but lacks the layered validation that multi-agent separation enables; (ii) a simpler two-agent system (judge and generator) without a checker, which would reduce reliability by eliminating iterative refinement; (iii) adjusting the relevance threshold (higher values increase refusal rate but may miss valid answers, lower values increase hallucination risk). Our multi-agent approach provides stronger reliability guarantees than single-agent alternatives, as the separation of concerns enables independent validation and auditability.

Transparency is primarily addressed by the generator agent's source citation requirement, but this alone is insufficient. The checker agent verifies that citations are present, contributing to transparency. However, interview findings revealed a gap: the transparency requirement also calls for communicating system limitations, which is not fully addressed by current architectural choices. Alternatives include: (i) adding a dedicated "limitations agent" that appends disclaimers to responses; (ii) implementing UI-level transparency features (disclaimers, last-update dates) that complement the architectural design; (iii) enhancing the checker agent to explicitly flag when answers are incomplete or based on partial information. The current architecture addresses provenance transparency but requires additional design decisions to fully satisfy the transparency requirement.

Bias mitigation is addressed through multiple architectural constraints: the generator agent is explicitly instructed not to infer beyond provided context; the checker agent verifies grounding in documents (rubric Q2, Q3); and the employment component performs direct data lookup without interpretation. Alternatives include: (i) using rule-based systems for structured queries (as discussed in Sect. 6, employment component paragraph), which eliminate inference-based bias but sacrifice natural language flexibility; (ii) implementing explicit bias-detection agents that scan outputs for potential bias indicators; (iii) using multiple generator agents with different prompts and selecting outputs through consensus, though this increases complexity and energy consumption. Our approach balances bias mitigation with practical usability, though the employment component's use of LLMs for simple queries represents a trade-off between bias risk and architectural consistency.

Model differences (addressing limitations of open-source LLMs) are mitigated through the multi-agent architecture's layered quality checks. The judge and checker agents compensate for single-model limitations by providing validation beyond what a single LLM can achieve. Alternatives include: (i) using a more capable commercial LLM, which

would reduce the need for multi-agent compensation but raise privacy/compliance concerns; (ii) fine-tuning the base model, which could improve performance but requires resources and may not address all limitations; (iii) hybrid architectures that route simple queries to rule-based handlers while reserving LLMs for complex cases, reducing energy consumption but increasing system complexity. Our multi-agent approach provides a practical solution for organizations prioritizing on-premise deployment and open-source models.

This analysis demonstrates that architectural choices and trust aspects form a many-to-many relationship: reliability benefits from judge filtering, checker validation, and refusal mechanisms working together; transparency requires both architectural (source citation) and non-architectural (UI disclaimers) decisions; and bias mitigation involves constraints across multiple agents. Understanding these relationships enables practitioners to make informed trade-offs when designing trustworthy LLM systems.

Implementing a similar system in practice requires careful attention to source materials. The guidelines component relies on HR documents converted to Markdown; any inaccuracies or outdated content in those documents will propagate into the chatbot's responses. Organizations must therefore maintain high-quality source documents: ensure source materials are accurate, consistent, and up-to-date, as inaccuracies in source documents will propagate into system responses. Plan for ongoing maintenance and evaluation to sustain trust as organizational policies evolve.

Transparency emerged as a powerful trust builder: users valued knowing the system's limitations and the provenance of its answers. Simple measures like visible disclaimers and source citations can significantly bolster confidence. Education and change management also play crucial roles. Interviewees wanted brief guidance on how to interact with the system and what to expect, and they recommended leadership modeling (e.g., managers promoting and using the chatbot) and structural nudges (such as limiting direct HR contact) to encourage adoption.

Our prototype adapts the MAIN-RAG architecture [10], extending its judge-based filtering with a checker agent and feedback loop. This offers a concrete pattern for organizations seeking to design trustworthy on-premise multi-agent LLM systems in HR and similar domains. The novel trust factor of *organizational measures* further implies that deploying LLM tools may require additional institutional support—such as training programs and user-friendly policies—beyond what is typical for conventional software.

In summary, the study illustrates how technical design choices, coupled with transparent communication and supportive organizational practices, can meaningfully enhance trust in LLM-based systems. The explicit mapping of architectural choices to trust aspects, along with discussion of alternatives, provides practitioners with a framework for making informed design decisions. It provides a replicable methodology, a refined trust taxonomy and a practical architecture that can be adapted and extended in future work.

7.3 Limitations, scope, and threats to validity

Our evaluation is requirement-driven (do the mechanisms realize trust properties?), not competitor-driven. In the absence of a public HR benchmark and standardized baselines, ad-hoc comparisons would conflate document quality, retrieval settings, and LLM choice. We therefore focus on alignment metrics, robustness, and user perceptions that directly reflect the elicited trust requirements. We further investigate MALLM instead of basic LLMs. A single LLM prompt can request citations and carefulness, but it provides weak guarantees: the same component retrieves, composes, and self-judges. By contrast, the multi-agent design makes trust requirements *enforceable* through interfaces: the judge constrains evidence, the generator must cite judged evidence, and the checker can veto release. This separation improves controllability (policy changes per role), testability (role-scoped checks), observability (who failed where), and governance (auditable traces), which are central to engineering trust beyond best-effort prompting.

Construct validity

Our quantitative constructs target trust-centric qualities: *answer relevancy*, *faithfulness*, and *contextual relevancy* for document-grounded answers, and a *correctness* metric for structured employment queries. A threat is a possible mismatch between these operationalizations and the latent construct of user trust. Mitigation: (i) trust requirements were elicited via interviews and an expert workshop and mapped to metrics (e.g., faithfulness to reliability and hallucination avoidance); (ii) we triangulated automated scores with post-task interviews to capture perceived usefulness and trust. Residual risk remains that automated judges may value concision differently from users who prefer source-aware explanations; we report such divergences in the results and discussion.

Internal validity

Several factors may affect the internal validity of our findings. First, given the fast pace of LLM research, we augmented peer-reviewed literature with preprints and practitioner sources to capture current developments; however, these sources lack formal vetting and may contain unverified claims. Second, our quantitative evaluation relied on LLM-as-a-judge (DeepEval), which occasionally produced judgments that differed from human expectations. Human validation was limited to low-scoring cases to understand false negatives. Systematic agreement rates between human

evaluators and the LLM judge were not calculated. False positives (where DeepEval incorrectly rates answers as correct) were not systematically checked due to the evaluation scale (2,860 runs) and the study's focus on requirement-driven evaluation triangulated with qualitative interviews. This limitation should be addressed in future work through systematic human validation of both high- and low-scoring cases. Third, the set of evaluation questions was finite; although we carried out robustness tests with multiple rephrasing, the question space is vast and different formulations might yield different results. Fourth, thematic analysis was performed by the two authors, whose coding and interpretation may have introduced bias despite independent coding and iterative reconciliation [8]. Finally, the evaluation examined the artifact holistically; individual design choices (e.g., judge thresholds or prompt wording) were not tested in isolation, so their specific contributions to performance remain uncertain.

External validity

The study's generalizability is also limited. We used a synthetic HR dataset and mock guidelines to simulate a corporate environment; real organizations may have different data characteristics, document structures and user behavior [47]. Our artifact targets HR queries specifically; while the trust factors identified may extend to other domains, the design may not. We interviewed only six employees from a single company, albeit selected to cover diverse roles, ages and genders. Their responses may reflect company-specific culture, and individuals with more critical views of LLMs may have self-selected out of participation. Consequently, both the qualitative themes and quantitative performance may not fully extrapolate to other organizations or to skeptical user populations.

Conclusion validity

We summarize central tendencies across repeated runs per question but do not perform hypothesis testing between competing systems (none included by design). The main inference risk is over-interpreting small differences that are within LLM stochastic variability.

8 Conclusion and future work

This study sought to connect user trust factors in LLM-based systems with the design of an open-source multi-agent HR chatbot. Through interviews and thematic analysis, we distinguished between internal trust factors (such as reliability, internal security and bias) and external factors (such as transparency, organizational measures and education). Reliability emerged as the most critical determinant of trust. Guided by these insights, we built a prototype comprising two components—one for employment data and one for HR guidelines—and evaluated it with quantitative metrics and qualitative feedback. The guidelines component achieved high answer-relevancy and faithfulness scores, while the employment component scored highly on correctness. Interviewees generally deemed the chatbot trustworthy enough for adoption, though they noted that further refinement and rigorous testing are needed before real-world deployment.

Our findings reinforce the principle that trust is a design objective: incorporating reliability and other trust factors early in development can yield LLM systems that users are more willing to adopt. Methodologically, the work demonstrates how a design science research approach can integrate user input, expert guidance and multi-agent architecture to address socio-technical concerns. Practically, it offers a template for building domain-specific, trustworthy AI tools and highlights the importance of high-quality source documents and transparent communication.

Future work should address the study's limitations and extend its scope. Recruiting participants from multiple organizations would test whether the identified trust factors generalize beyond a single company. Research on the engineering process, for example, embedding transparency and fairness considerations throughout the development life cycle, could complement our design-focused approach. Longitudinal studies would reveal how trust evolves with sustained use. Research on maintenance and evolution of LLM-based systems over time would address how to keep documentation current, adapt to changing organizational policies, and maintain trust as the system and its context evolve. Finally, further investigations into security, domain-specific fine-tuning and alternative multi-agent structures with more capable LLMs (open-source and commercial) could help optimize LLM-based systems for reliability and user confidence. The effect on trust of open-source versus commercial models would also be interesting to investigate from a security perspective even if our results mainly show that practitioners are concerned with reliability and accuracy.

Author contributions J.A. and F.E. designed the study and conducted the experiments as well as the analysis under the supervision of L.G. F.D. and L.G. led the framing of the work as a scientific contribution and wrote the main manuscript text. F.D. contributed to data analysis/validation and manuscript editing and handled the revision. All authors discussed the results, revised the manuscript, and approved the final version. L.G. is the corresponding author.

Funding Open access funding provided by Mid Sweden University.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Afroogh S, Akbari A, Malone E et al (2024) Trust in ai: progress, challenges, and future directions. Humanit Soc Sci Commun 11(1):1–30
2. Asai A, Wu Z, Wang Y, et al (2024) Self-rag: learning to retrieve, generate, and critique through self-reflection. In: International conference on learning representations
3. Ayala O, Bechard P (2024) Reducing hallucination in structured outputs via retrieval-augmented generation. In: Proceedings of the 2024 conference of the north American chapter of the association for computational linguistics: human language technologies (Volume 6: Industry Track). Association for Computational Linguistics, pp 228–238, https://doi.org/10.18653/v1/2024.naacl-industry.19
4. Ayyamperumal SG, Ge L (2024) Current state of LLM Risks and AI guardrails. arXiv:2406.12934
5. Baltes S, Speith T, Chiteri B, et al (2025) On the need to rethink trust in ai assistants for software development: a critical review. https://doi.org/10.48550/arXiv.2504.12461, arXiv preprint, arXiv:2504.12461
6. Barone AM, Stagno E (2023) Chatbots. In: Artificial intelligence along the customer journey: a customer experience perspective. Springer, pp 37–54
7. Borg M, Bengtsson J, Österling H, et al (2022) Quality assurance of generative dialog models in an evolving conversational agent used for swedish language practice. In: Proceedings of the 1st international conference on ai engineering – software engineering for AI (CAIN '22). ACM, New York, NY, USA, pp 22–32, https://doi.org/10.1145/3522664.3528592
8. Braun V, Clarke V (2006) Using thematic analysis in psychology. Qual Res Psychol 3:77–101. https://doi.org/10.1191/1478088706qp063oa
9. Brown T, Mann B, Ryder N et al (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
10. Chang CY, Jiang Z, Rakesh V, et al (2024) Main-rag: multi-agent filtering retrieval-augmented generation. arXiv preprint arXiv:2501.00332
11. Confident AI (2025a) DeepEval. https://www.deepeval.com/docs/metrics-introduction, Accessed 20 April 2025
12. Confident AI (2025b) Llm evaluation metrics - deepeval. https://www.deepeval.com/docs/metrics-llm-evals, Accessed 03 Aug 2025
13. Devlin J, Chang MW, Lee K, et al (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: human language technologies, vol 1 (Long and Short Papers), pp 4171–4186
14. Dobslaw F, Feldt R, Yoon J, et al (2025) Challenges in testing large language model based software: a faceted taxonomy. arXiv preprint arXiv:2503.00481
15. Dong Y, Mu R, Zhang Y, et al (2024) Safeguarding large language models: a survey. arXiv preprint arXiv:2406.02622
16. Dorfner FJ, Jürgensen L, Donle L, et al (2024) Is open-source there yet? A comparative study on commercial and open-source LLMs in their ability to label chest X-ray reports. arXiv preprint arXiv:2402.12298
17. Douze M, Guzhva A, Deng C, et al (2025) The Faiss library. arXiv:2401.08281
18. Es S, James J, Anke LE, et al (2024) Ragas: automated evaluation of retrieval augmented generation. In: Proceedings of the 18th conference of the European chapter of the association for computational linguistics: system demonstrations, pp 150–158
19. FlagEmbedding (2024) BAAI/bge-small-en-v1.5. https://huggingface.co/BAAI/bge-small-en-v1.5, Accessed 09 May 2025
20. Gao C, Chen X, Zhang G (2025) Sva-icl: improving llm-based software vulnerability assessment via in-context learning and information fusion. Inf Softw Technol pp 107803
21. Gao Y, Xiong Y, Gao X, et al (2024) Retrieval-Augmented generation for large language models: a survey. arXiv:2312.10997
22. Han S, Zhang Q, Yao Y, et al (2024) Llm multi-agent systems: challenges and open problems. arXiv preprint arXiv:2402.03578
23. Hoff KA, Bashir M (2015) Trust in automation: integrating empirical evidence on factors that influence trust. Hum Factors 57(3):407–434. https://doi.org/10.1177/0018720814547570
24. Huang Y, Sun L, Wang H, et al (2024a) Position: TRUSTLLM: Trustworthiness in large language models. In: Proceedings of machine learning research (ICML 2024), pp 20166–20270, https://proceedings.mlr.press/v235/huang24x.html, also available as arXiv:2401.05561, https://doi.org/10.48550/arXiv.2401.05561
25. Huang Y, Sun L, Wang H, et al (2024b) Trustllm: trustworthiness in large language models. arXiv preprint arXiv:2401.05561
26. Jayakumar T, Farooqui F, Farooqui L (2023) Large language models are legal but they are not: making the case for a powerful legalllm. Proc Nat Legal Lang Process Workshop 2023:223–229
27. Ji Z, Lee N, Frieske R et al (2023) Survey of hallucination in natural language generation. ACM Comput Surv 55(12):1–38
28. Kaas MH, Porter Z, Lim E, et al (2023) Ethics in conversation: building an ethics assurance case for autonomous ai-enabled voice agents in healthcare. In: Proceedings of the first international symposium on trustworthy autonomous systems, pp 1–13
29. Kelton K, Fleischmann KR, Wallace WA (2008) Trust in digital information. J Am Soc Inform Sci Technol 59(3):363–374
30. Knauss E (2021) Constructive master's thesis work in industry: guidelines for applying design science research. In: 2021 IEEE/ACM 43rd international conference on software engineering: software engineering education and training (ICSE-SEET), IEEE, pp 110–121
31. Lee JD, See KA (2004) Trust in automation: designing for appropriate reliance. Hum Factors 46(1):50–80. https://doi.org/10.1518/hfes.46.1.50_30392
32. Lewis P, Perez E, Piktus A et al (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv Neural Inf Process Syst 33:9459–9474
33. Li W, Wang X, Li W, et al (2025) A survey of automatic prompt engineering: an optimization perspective. arXiv:2502.11560
34. Li X, Wang S, Zeng S et al (2024) A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. Vicinagearth 1(1):9
35. Liang T, He Z, Jiao W, et al (2024) Encouraging divergent thinking in large language models through multi-agent debate. In: Proceedings of the 2024 conference on empirical methods in natural language processing, pp 17889–17904

36. Liu Y, Yao Y, Ton JF, et al (2024) Trustworthy LLMs: a survey and guideline for evaluating large language models' alignment. arXiv:2308.05374
37. Manchanda J, Boettcher L, Westphalen M, et al (2024) The open source advantage in large language models (LLMs). arXiv preprint arXiv:2412.12004
38. Mayer RC, Davis JH, Schoorman FD (1995) An integrative model of organizational trust. Acad Manag Rev 20(3):709–734
39. McNamara C (2017) General guidelines for conducting research interviews. http://managementhelp.org/businessresearch/interviews.htm, Retrieved 6 March 2025
40. Pandey SK, Chand S, Horkoff J et al (2025) Design pattern recognition: a study of large language models. Empir Softw Eng 30(3):69
41. Patton MQ (2014) Qualitative research and evaluation methods: integrating theory and practice, 4th edn. SAGE Publications, Thousand Oaks, CA
42. Rotter JB (1980) Interpersonal trust, trustworthiness, and gullibility. Am Psychol 35(1):1
43. Schwartz S, Yaeli A, Shlomov S (2023) Enhancing trust in llm-based ai automation agents: new considerations and future challenges. arXiv:2308.05391
44. Sergeyuk A, Golubev Y, Bryksin T et al (2025) Using ai-based coding assistants in practice: state of affairs, perceptions, and ways forward. Inf Softw Technol 178:107610
45. Shen X, Chen Z, Backes M, et al (2024) "Do Anything Now": characterizing and evaluating in-the-wild jailbreak prompts on large language models. In: Proceedings of the 2024 on ACM SIGSAC conference on computer and communications security, pp 1671–1685
46. Shettigar R (2024) AI in human resource: an empirical research on the impact, adoption, and employee perspectives. In: 2024 International conference on trends in quantum computing and emerging business technologies, pp 1–4
47. Stol KJ, Fitzgerald B (2018) The abc of software engineering research. ACM Trans Softw Eng Methodol (TOSEM) 27(3):1–51
48. Tran KT, Dao D, Nguyen MD, et al (2025) Multi-agent collaboration mechanisms: a survey of llms. arXiv preprint arXiv:2501.06322
49. Wang L, Ma C, Feng X et al (2024) A survey on large language model based autonomous agents. Front Comp Sci 18(6):186345
50. Wieringa R (2009) Design science as nested problem solving. In: Proceedings of the 4th international conference on design science research in information systems and technology, pp 1–12
51. Wooldridge M, Jennings NR (1995) Intelligent agents: theory and practice. The Knowl Eng Rev 10(2):115–152
52. Xi Z, Chen W, Guo X et al (2025) The rise and potential of large language model based agents: a survey. Sci China Inf Sci 68(2):121101
53. Zheng L, Chiang WL, Sheng Y et al (2023) Judging llm-as-a-judge with mt-bench and chatbot arena. Adv Neural Inf Process Syst 36:46595–46623