Design and Development of a Controlled Multi-agent Conversational System Using Large Language Models for Public Services: A Case Study

Zaid Rustamov [1], Mir Amir Pashayev [1], Sanan Jafarov [1], Ulvi Aslanli [1], Aghasalim Mammadov [1], Alakbar Valizada [1], and Samir Rustamov [2,*]

[1]AI Laboratory, Megasec LLC, Baku, Azerbaijan
[2] School of IT and Engineering, ADA University, Baku, Azerbaijan
Email: zaid.r@megasec.ai (Z.R.); amir.p@megasec.ai (M.A.P.); sanan.j@megasec.az (S.J.); ulvi.a@megasec.ai (U.A.); aghasalim.m@megasec.ai (A.M.); alakbar.v@megasec.ai (A.V.); srustamov@ada.edu.az (S.R.)
*Corresponding author

***Abstract*—Large Language Model (LLM)-based conversational systems are increasingly adopted as front-facing interfaces for public and institutional services. However, end-to-end and tool-augmented approaches often rely on implicit, prompt-driven decision making, which limits controllability, transparency, and safety in regulated environments. This paper presents a case study of a controlled multi-agent conversational architecture deployed within a single institutional public-service environment, explicitly separating intent classification, service selection, and response generation into independent and auditable decision stages. The proposed design introduces an intent-driven control mechanism and a deterministic service routing process that constrains response generation to supported service contexts while enabling systematic handling of ambiguity and out-of-scope queries. To support reliable deployment, we propose an iterative development workflow in which prompts are treated as evolving behavioral specifications validated through structured testing and monitoring. The system is evaluated on 1255 real conversational queries spanning multi-turn interactions collected from the deployed system. Experimental results demonstrate high reliability across all modules, achieving 99.2% accuracy in intent classification, 97.7% accuracy in single-service routing, and 98.8% correctness for service-grounded responses. Overall, the system produces correct responses for 97.6% of evaluated queries while maintaining deterministic handling of unsupported and ambiguous requests. Within the scope of this single-institution case study, the findings highlight the importance of explicit decision boundaries and modular design for deploying LLM-based assistants in safety-critical and public-service domains and provide practical guidance for building controllable, service-grounded conversational systems.**

***Keywords*—conversational AI, intent classification, multi-agent systems, public-service chatbots, safety-critical AI, tool-augmented systems**

I. Introduction

Recent advances in Large Language Model (LLMs) have significantly accelerated the adoption of conversational systems across a wide range of domains. Prior work has shown that LLMs such as Generative Pre-trained Transformer (GPT)-4 and related architectures can automate cognitive tasks and improve context-aware support in customer service settings [1]. These models are increasingly integrated into enterprise customer relationship management systems to enhance automation, response time, and personalization [2]. Owing to their strong language understanding and generation capabilities, LLM-based chatbots are consequently being explored as front-facing interfaces for accessing institutional information and services. However, deploying such systems in public-sector and regulated environments introduces substantial challenges, particularly with respect to response reliability, hallucination risks, ambiguity handling, and compliance with predefined service boundaries [3].

Existing approaches to conversational artificial intelligence are commonly divided into end-to-end language-model-driven systems and tool-augmented or agent-based frameworks [4]. End-to-end LLM-based chatbots rely on direct sequence-to-sequence generation and therefore produce fluent and flexible responses [5]. However, they expose little control over intent interpretation and response grounding, which often leads to unsupported or inconsistent outputs [6]. More recent tool-augmented and agent-based approaches extend LLMs with access to external resources and application programming interfaces, enabling more reliable task execution and broader functional capabilities [7, 8].

Manuscript received March 5, 2026; revised May 15, 2026; accepted June 9, 2026; published September 14, 2026.

Despite these advances, decision making in many existing systems remains largely implicit and prompt driven. Consequently, system behavior can be difficult to constrain, analyze, and systematically validate, particularly in safety-critical or policy-sensitive application domains [9]. Fig. 1 contrasts these architectural paradigms with the modular design proposed in this work.

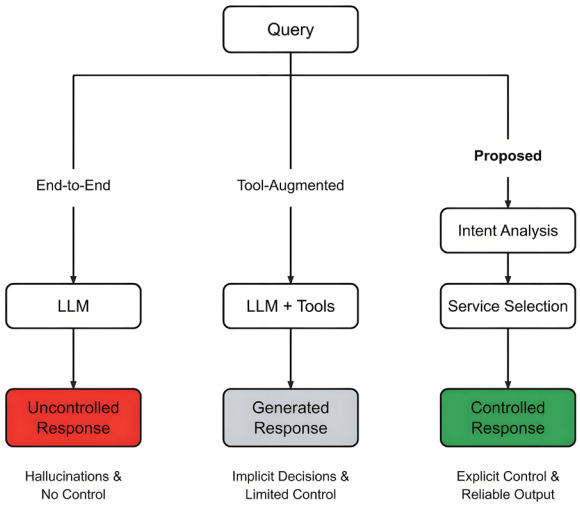

Fig. 1. Comparison of conversational system architectures. End-to-end LLM-based systems rely on direct generation and may produce uncontrolled responses. Tool-augmented approaches improve functionality but embed decision-making implicitly. The proposed modular architecture introduces explicit intent analysis and service selection, enabling controlled and reliable responses.

In this paper, we present a case study of a modular conversational assistant architecture that explicitly addresses the need for controlled and transparent behavior in institutional settings. The system has been designed and deployed within a single public-service environment, and the analysis reported here is bounded to that institutional context. The system is built around role-specialized LLM agents, each responsible for a well-defined function within the conversational pipeline, including intent classification, service selection, handling of irrelevant or out-of-scope queries, response generation, and user clarification. A key characteristic of the proposed architecture is an intent-driven control mechanism that determines system behavior before any response is generated. This mechanism is coupled with explicit service selection logic, which distinguishes between no-match, single-match, and multi-match scenarios and governs whether the system should respond, request clarification, or defer action. Together, these design choices enable service-grounded interactions while maintaining clear decision boundaries and operational transparency.

Beyond the system architecture, this paper describes the iterative development and evaluation workflows used to build and refine both the intent classification and service selection modules. Rather than treating prompts as fixed instructions, we treat them as living behavioral specifications and refine them through structured testing and monitoring. The main contributions of this work are threefold:

- A controlled multi-agent conversational architecture designed for public-service applications;
- Concrete and reproducible development workflows for intent classification and service selection in LLM-based systems;
- Practical insights into the deployment and maintenance of such systems under real-world operational constraints.

The remainder of this paper is organized as follows. Section II reviews related work on conversational systems and safety-critical AI. Section III presents the proposed system architecture and its functional modules. Section IV describes the experimental setup, evaluation results, and an analysis of remaining errors. Section V concludes the paper and discusses limitations and future research directions.

II. Literature Review

A. End-to-End LLM-Based Conversational Systems

End-to-end conversational systems based on LLMs have attracted significant attention due to their strong language understanding and generation capabilities. Transformer-based architectures such as GPT-style models demonstrate high linguistic fluency and generalization across a wide range of conversational tasks, enabling open-domain interaction without task-specific training [5, 10]. These systems typically rely on direct sequence-to-sequence generation, allowing flexible responses conditioned on conversational context.

Despite these advantages, end-to-end LLM-based chatbots expose little control over intermediate decisions such as intent interpretation, service applicability, or response grounding [11]. Prior studies show that these models are prone to hallucinated content and unsupported assumptions, especially when operating outside well-defined knowledge boundaries [6, 12]. These limitations are especially problematic in task-oriented or institutional settings, where outputs must comply strictly with predefined services, policies, and domain constraints.

B. Tool-Augmented and Agent-Based LLM Systems

To improve task execution and grounding, recent work has built tool-augmented and agent-based LLM systems that give language models access to external tools, application programming interfaces, and specialized modules. Representative approaches include reasoning-and-acting frameworks that interleave natural language reasoning with tool invocation [7], systems that learn to select and use tools automatically [13], and modular reasoning architectures that combine language models with symbolic or functional components [14].

More recently, multi-agent conversational frameworks have been proposed to coordinate multiple role-specialized agents through structured interaction protocols [8, 15, 16]. These systems demonstrate improved performance on complex, multistep tasks and enable decomposition of reasoning, planning, and execution across agents. However, decision routing in many existing agent-based systems remains largely prompt-driven, with control logic implicitly encoded in natural language instructions. As a

result, explicit separation between intent recognition and service resolution is often absent, complicating systematic validation and conservative handling of ambiguous user queries.

C. Controlled and Safety-Critical Conversational Systems

The increasing deployment of LLM-based conversational systems in safety-critical and regulated domains has motivated extensive research on reliability, grounding, and risk mitigation. Prior work has documented risks related to hallucination, inconsistent behavior, and lack of transparency in generative models, particularly when applied to high-stakes decision-making contexts [6, 17]. Proposed mitigation strategies include retrieval-based grounding, rule-based safeguards, constrained decoding, and human-in-the-loop supervision [18–20].

These approaches improve robustness at the model or runtime level, but engineering-level system design for controlled conversational assistants has received less attention. The literature offers little guidance on how to structure decision stages, handle ambiguity conservatively, or support iterative development and evaluation workflows.

In parallel, recent research has explored natural language processing applications for low-resource languages, including Azerbaijani. Studies such as those of Alizada *et al.* [21] investigate the development of contextualized word embeddings using transformer-based models, while Guliyev *et al.* [22] analyzes public sentiment in Azerbaijani news and social media using a range of machine learning and deep learning approaches. These works highlight both the progress and the challenges associated with building language technologies in underrepresented linguistic settings.

III. System Design and Functional Modules

A. Design Objectives and System Architecture

LLM-based conversational systems have shown strong natural language understanding and generation capabilities. Existing end-to-end and agent-based approaches, however, rely heavily on implicit, prompt-driven decision making, which makes their behavior difficult to constrain, analyze, and validate in public-service and other regulated environments. The proposed system addresses these limitations by treating explicit control, transparency, and conservative decision-making as primary design goals.

Fig. 2 illustrates the overall system architecture, which decomposes conversational processing into a sequence of well-defined decision stages. Incoming user queries are first processed by an Intent Detection Module, which performs explicit intent assignment before any service resolution or response generation. This separation prevents premature generation and ensures that unsupported or policy-restricted intents, such as greetings, termination requests, or out-of-scope queries, are handled through predefined system responses rather than freeform generation.

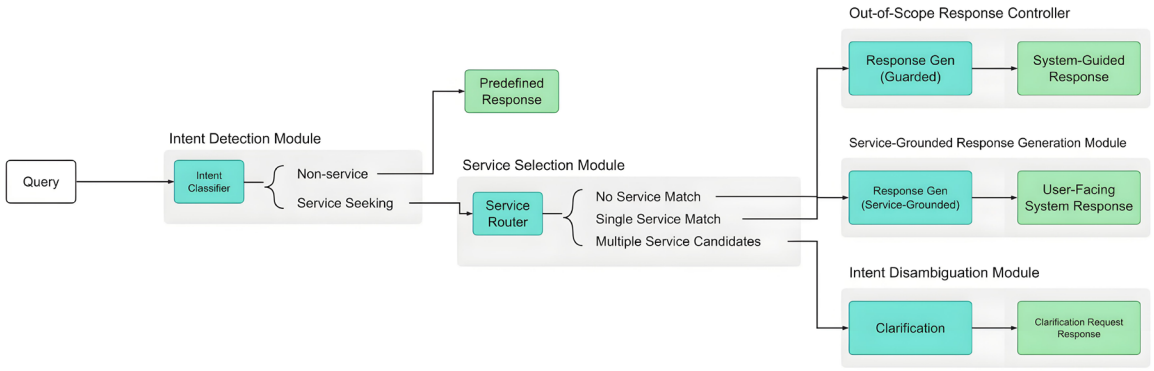

Fig. 2. High-level system architecture illustrating the modular conversational pipeline, including intent detection, service selection, out-of-scope handling, service-grounded response generation, and intent disambiguation, together with explicit decision and failure paths.

For service-seeking intents, the query is forwarded to the Service Selection Module, which uses a structured catalog of service descriptions and keywords to make its decision. Unlike prompt-embedded tool selection used in many agent-based frameworks, service selection is performed explicitly and results in one of 3 outcomes: no service match, a single service match, or multiple candidate services. This explicit branching enforces conservative behavior and prevents unsupported service assumptions.

When multiple candidate services are identified, the system routes the interaction to an Intent Disambiguation Module, which generates a clarification request instead of attempting to resolve the ambiguity implicitly. Treating ambiguity as an explicit case lets the system prioritize correctness and safety over coverage—a requirement that matters especially in institutional settings.

Only after a service has been uniquely resolved does the system proceed to the Service-Grounded Response Generation Module, where responses are generated based on the resolved service context. This design ensures that generated outputs remain aligned with supported services and reduces the risk of hallucinated or non-compliant responses. Queries that fail service resolution or violate scope constraints are handled by an Out-of-scope Response Controller, which produces system-guided responses or refusals as appropriate.

Together, these modules form a controlled conversational pipeline in which all response generations are preceded by explicit intent- and service-level decisions. The clear separation of responsibilities and decision boundaries enables systematic evaluation, transparent failure analysis, and iterative refinement, as discussed in subsequent sections.

B. Intent Classification Module

The intent classification module is the first routing gate of the pipeline. Its role is deliberately coarse-grained: it separates non-service conversational intents from service-seeking requests so that only eligible queries reach service resolution. This design reduces unnecessary downstream processing and prevents policy-restricted or unsupported interactions from entering the service selection and response generation stages.

Table I summarizes the intent categories and their routing outcomes. In particular, the classifier does not attempt fine-grained service identification; instead, it functions as an explicit eligibility filter that enables later modules to operate within a well-defined scope.

TABLE I. INTENT CLASSIFICATION AND INITIAL QUERY ROUTING

Intent Category	Description	Typical Query Examples	Routing Outcome
Greeting/Opening	User initiates conversation without requesting a service or action.	"Hello", "Good morning", "Hi, are you there?"	Return a predefined greeting response.
Conversation Termination	User explicitly signals the end of the interaction.	"Thanks, that's all", "Goodbye", "I'm done"	Return a predefined closing response.
Unsupported Language	Query is written in a language not supported by the system.	Queries written in unsupported languages.	Return a predefined system response.
Policy-Restricted	User requests information or actions disallowed by institutional policy.	Requests involving sensitive data access, illegal actions, or restricted procedures.	Return a predefined policy-compliant response.
Service-Seeking	User requests information or actions supported by the system's service set.	"How can I renew my ID?", "I want to check my fine", "How do I apply for a document?"	Forward query to the Service Selection Module.

C. Service Selection Module

The service selection module is the central decision-making component of the architecture. Its role is to determine which predefined services are relevant to a given user query before any service-specific response is generated. By explicitly separating service resolution from both intent classification and response generation, this module addresses a key limitation of end-to-end and prompt-embedded agent systems, where service assumptions are often made implicitly during generation.

Service selection is formulated as a constrained decision process operating over the current user query, conversational context, and a structured representation of supported services. Let:

- Q denote the current user query,
- $D_h = \{(q_1, r_1) \ldots (q_n, r_n)\}$ denote the dialogue history,
- $S = \{s_1, s_2 \ldots s_m\}$ denote the set of supported services, and
- i denote the system-level routing instructions encoded in the service router prompt.

Each service $S_j \in S$ is represented as a tuple: as defined in Eq. (1):

$$S_j = (key_j, desc_j, kw_j) \tag{1}$$

where key_j is a unique service identifier, $desc_j$ is a natural language description of the service scope, and kw_j is a set of representative keywords. In other words, every supported service is described to the system through 3 pieces of information: a stable name, a short text description of what the service covers, and a list of trigger keywords used to recognize it.

The service selection module implements a deterministic mapping f_{svc} (the service selection function), defined in Eq. (2):

$$f_{svc}(Q, D_h, S, i) \rightarrow P(S) \tag{2}$$

where $P(S)$ denotes the power set of available services. Intuitively, the module takes the user query, the dialogue history, and the catalog of supported services, and returns any subset of those services as the routing decision. This subset can be empty, contain one service, or contain several, allowing the system to explicitly represent three mutually exclusive outcomes:

- $|S^*| = 0$: no supported services apply,
- $|S^*| = 1$: a single service is selected, and
- $|S^*| > 1$: multiple candidate services are identified.

D. Response Generation and Failure Handling

Response generation in the proposed architecture is strictly governed by explicit intent and service-resolution decisions produced by upstream modules. Let I denote the resolved intent label and let $S^* \subseteq S$ denote the output of the service selection function $f_{svc}(Q, D_h, S, i_{svc})$.

Unlike end-to-end conversational systems, free-form generation is not used in every case. Instead, the system selects a specific response mode based on the intent type and the cardinality of S^*, so that every user-facing output is traceable to an explicit decision.

The response mode is determined as follows (SS denotes Service-Seeking) as expressed in Eq. (3)

$$Mode(Q)\begin{cases} Intent - Predefined, & I \neq SS \\ Out - of - Scope, & I = SS \wedge |S^*| = 0 \\ Clarification, & I = SS \wedge |S^*| > 1 \\ Service - Grounded, & I = SS \wedge |S^*| = 1 \end{cases} \tag{3}$$

In plain terms, if the user is not asking for a service, the system returns a predefined reply; if the user is asking for a service but no supported service matches, it returns an out-of-scope message; if several services could apply, it

asks the user to clarify; and only when exactly one service is identified does it generate a free-form, service-grounded answer. For non-service-seeking intents, responses are returned through predefined system templates and do not invoke language model generation. For service-seeking intents, response generation is model-mediated and parameterized by the resolved execution context.

When no supported service is identified ($|S^*| = 0$), an out-of-scope response is generated using only the current query, dialogue history, and system-level constraints, as expressed in Eq. (4), where the notation $R \sim M(\cdot)$ means that the response R is sampled from a language model M, with M_{oos} denoting the dedicated out-of-scope response model:

$$R \sim M_{oos}(Q, D_h, i_{oos}) \tag{4}$$

That is, the out-of-scope response is produced by a dedicated language model invocation that is not given access to any service description, so it cannot fabricate references to unsupported services.

When multiple candidate services are identified ($|S^*| > 1$), the system generates a clarification request conditioned on the set of candidate service descriptions, as expressed in Eq. (5), where M_{clar} denotes the clarification response model and i_{clar} denotes the system-level instructions provided to the clarification model.

$$R \sim M_{clar}(Q, S^*, i_{clar}) \tag{5}$$

In other words, the system explicitly lists the candidate ser-vices to the clarification model so that the resulting question reflects only the services that genuinely matched the query. When a single service is uniquely resolved ($|S^*| = 1$), the system produces a service-grounded response conditioned on the resolved service context s^*, as expressed in Eq. (6), where M_{sg} denotes the service-grounded response model, s^* is the single resolved service (the unique element of S^*), and i_{sg} denotes the system-level instructions provided to the service-grounded response model.

$$R \sim M_{sg}(Q, D_h, S^*, i_{sg}) \tag{6}$$

That is, the answer is generated by a service-grounded model invocation that is restricted to a single, validated service context.

In the service-grounded case, the system enforces the invariant in Eq. (7), where the symbol ∀ denotes "for all", so that every response component r must reference the resolved service:

$$\forall r \in : Referenced\ Service(r) = s^* \tag{7}$$

Ensuring that generated responses do not reference unsupported or unrelated services. Intuitively, every response component must point back to the same resolved service; any reference to a different service is treated as a violation and discarded.

By separating response generation modes and conditioning each language model call on validated intent and service-level decisions, the system enforces controlled execution, deterministic failure handling, and strict adherence to the supported service scope. This design prevents implicit service assumptions and removes the uncontrolled generation paths commonly observed in end-to-end and prompt-embedded agent systems.

IV. Results and Discussion

The evaluation is conducted on a dataset of 1255 real user queries collected from multiturn conversational interactions with the deployed system. The deployment is a government public-service chatbot operated in production. The dataset is proprietary and cannot be shared, including in anonymized form, due to the data-handling and confidentiality policies governing the service. Queries were drawn from the live operational logs of the chatbot; only anonymized queries, with personally identifying information removed at the source, were made available to the research team, and no externally sourced or third-party conversational data was used. The dataset reflects realistic usage patterns, including both single-turn requests and longer task-oriented dialogues. Each query was manually annotated by the research team for intent class, service routing outcome, and response correctness to enable both module-level and end-to-end evaluation.

Fig. 3 illustrates the distribution of conversation length measured by the number of queries per conversation. The majority of interactions are short, with single-query conversations forming the largest portion, followed by 2- and 3-turn dialogues. Longer interactions (4 or more queries) occur less frequently but represent complex or ambiguous service requests that require clarification or multi-step handling. This distribution confirms that the dataset captures both simple and multi-turn usage scenarios typical of public-service conversational systems.

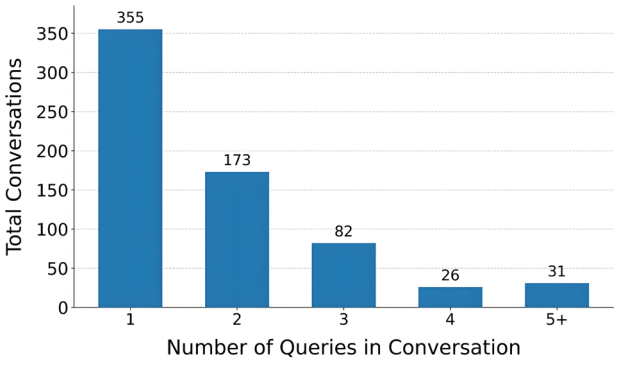

Fig. 3. Distribution of conversation lengths showing the number of conversations grouped by the number of user queries per conversation.

A. Intent Classification Module

Fig. 4 presents the evaluation of the intent classification module, showing both the distribution of queries across intent classes and the number of correctly labeled instances. The goal of this evaluation is to verify that nonservice queries are reliably filtered before downstream

processing, ensuring that only relevant requests reach the service selection stage.

The 4 non-service intents (Greeting/Opening, Con-versation Termination, Unsupported Language, and Policy-Restricted) account for approximately 18% of all queries and are classified with an accuracy of about 99.1%. These queries are handled through predefined responses, which avoids unnecessary service routing and reduces the risk of unintended generation. Accurate detection of Unsupported Language and Policy-Restricted requests matters in particular, because these responses communicate the system's scope, supported languages, and institutional boundaries to the user.

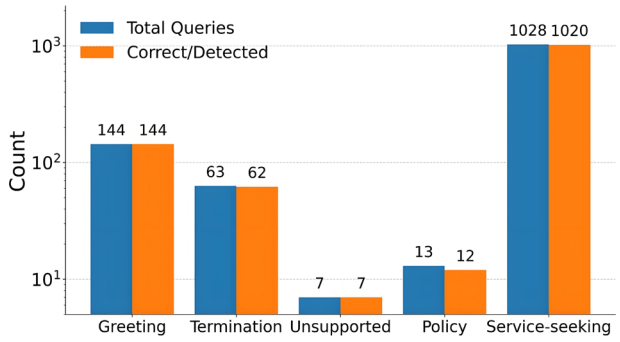

Fig. 4. Intent classification evaluation showing the distribution of queries across intent categories and the number of correctly classified instances.

The remaining queries are classified as service-seeking with 99.2% accuracy, demonstrating the reliability of the first decision stage in separating actionable requests from conversational or out-of-scope interactions.

B. Service Selection Module

Fig. 5 presents the evaluation of the service selection module, showing the distribution of service-seeking queries across routing outcomes and the number of correctly routed cases. The majority of service-seeking queries correspond to a single service (914 queries), of which 893 were correctly routed, indicating strong reliability in identifying the appropriate service context for downstream response generation.

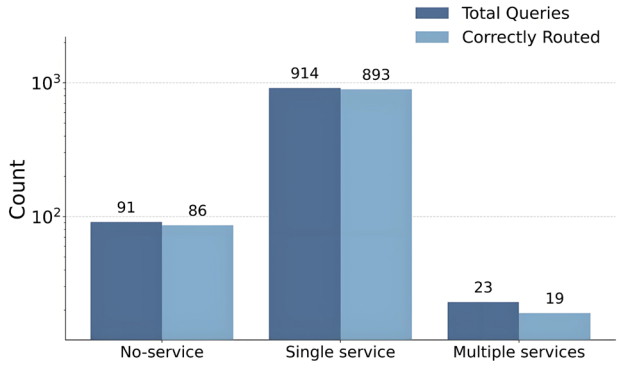

Fig. 5. Service selection module evaluation showing routing outcomes (no service, single service, multiple services) and correctly routed queries.

A smaller portion of queries resulted in no-service matches (91 queries, 86 correctly routed), representing requests for unsupported services that are subsequently handled by the scope validation component. Queries mapped to multiple services (23 queries, 19 correctly routed) reflect inherently ambiguous requests that require clarification through the disambiguation module.

C. Response Generation Module

Fig. 6 presents the evaluation of the response generation stage across the 3 response paths: scope validation, service-grounded generation, and disambiguation. The service-grounded path accounts for the majority of cases (902 queries) and is the most reliable, with 891 responses judged correct. Once a single service context is resolved, conditioning the model on the matching service information yields highly consistent and accurate outputs.

The remaining cases correspond to scope-validation responses (89 queries) and disambiguation responses (37 queries). These categories cover unsupported or ambiguous requests, which are harder by nature because the service context is limited or uncertain. The system still performs strongly in these edge cases, which suggests that the con-trolled routing and generation strategy holds up across all response types.

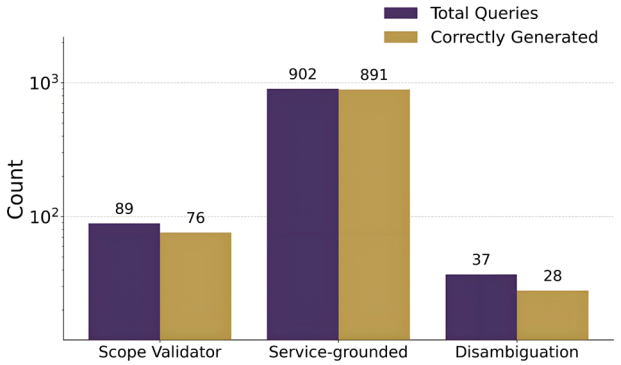

Fig. 6. Response generation module evaluation comparing total processed queries and correctly generated responses across scope validation, service-grounded generation, and disambiguation cases.

D. Error Analysis

Table II reports an apparent accuracy gap at the response generation stage between the service-grounded path (98.8%) and the combined scope-validation and disambiguation path (82.5%). Inspection of the 22 errors in this combined path shows that the gap is misleading: almost all of the failures are inherited from upstream routing rather than produced by the scope-validation or disambiguation modules themselves.

Of the 126 queries that reached the combined scope-validation and disambiguation path, only 105 were genuine no-service or multi-service cases for which clarification or a scope-bounded refusal is the correct system behavior. The remaining 21 were single-service queries that were misrouted at the service selection stage (the same 21 errors reported under "Single-service" in Table II) and propagated to this path. On the 105 legitimate cases, the modules answered 104 correctly, corresponding to a per module accuracy of approximately 99%. On the 21 misrouted queries, however, the modules cannot recover: they correctly produce a clarification

request or an out-of-scope response, but the user actually expected a direct service-grounded answer, so every one of these 21 cases is annotated as incorrect. Once these upstream-propagated failures are isolated, only a single residual error remains that is genuinely attributable to the scope-validation or disambiguation modules.

This decomposition answers the questions raised about the apparent accuracy gap. The errors are not generation failures of the scope-validation or disambiguation modules; they are routing failures that the downstream modules are not designed to detect or override. The scope detector itself does not produce systematic false positives or false negatives in this evaluation, and there is no observable pattern of the disambiguation module struggling with the queries that are actually directed to it. The only category in which the system genuinely struggles is the boundary case at the service selection stage, where queries that touch on 2 thematically adjacent services or sit between supported and unsupported scope account for essentially all of the residual end-to-end errors.

TABLE II. EVALUATION RESULTS ACROSS SYSTEM MODULES

Stage	Category	Query Count	Correct	Accuracy
Intent Classification	Service-seeking	1028	1020	99.2%
	Non-service	227	225	99.1%
Service Routing	Single-service	914	893	97.7%
	No / multiple-services	114	105	92.1%
Response Generation	Service-grounded	902	891	98.8%
	Scope & Disambiguation	126	104	82.5%

Two implications follow: First, the natural locus for further improvement is the service selection prompt and the service-catalog descriptions and keywords, not the scope-validation or disambiguation prompts. Second, the conservative design choice of handling ambiguity through clarification rather than free-form generation is functioning as intended: when a query is correctly identified as ambiguous or out-of-scope, the system reliably produces an appropriate response.

E. End-to-End System Performance

Table II summarizes the overall performance of the proposed conversational pipeline across all modules. Out of 1255 evaluated queries, the intent classification stage demonstrates very high reliability, with only 2 errors observed for non-service intents. This confirms that the system consistently prevents unnecessary downstream processing for greetings, termination requests, unsupported language usage, and policy-restricted queries, ensuring stable handling of routine and out-of-scope interactions.

Across the full pipeline, the system produced 225 predefined non-service responses, 104 scope-validation or disambiguation responses, and 891 service-grounded responses. Most user interactions reach the service-grounded generation stage, while ambiguous or unsupported requests are systematically redirected to clarification or scope-control mechanisms. The aggregated results show that the modular pipeline maintains high reliability across stages while keeping conversational behavior controlled and traceable in the studied institutional setting.

V. CONCLUSION

This paper presented a case study of a modular conversational architecture for controlled, transparent, and service-grounded interactions in public-service environments. Unlike end-to-end and prompt-embedded agent approaches, the proposed system introduces explicit decision boundaries between intent classification, service selection, and response generation, transforming conversational processing into a staged and auditable pipeline. Evaluated on 1255 real user queries, the system achieved over 99% accuracy in intent classification, 97.7% accuracy in single-service routing, and an overall end-to-end accuracy of approximately 97%.

These results indicate that explicit intent- and service-level control yields reliable and traceable conversational behavior in the studied deployment. The error analysis in Section IV further shows that the remaining errors are concentrated at service boundaries and on conservative refusals rather than on uncontrolled generation, consistent with the architecture's design intent. We do not, however, claim a quantitative improvement over implicit prompt-driven base-lines, as a controlled comparison was outside the scope of this case study.

The study is subject to several limitations. The evaluation is confined to a single institutional domain, and the intent and service routing modules rely on prompt-engineered specifications that require periodic updates as services and user behaviors evolve. The current evaluation is also based on offline annotated data and does not capture largescale deployment metrics such as user satisfaction and operational stability.

Future work will therefore explore cross domain validation, multilingual support, automated prompt optimization, controlled comparison against implicit prompt-driven baselines, and the integration of user feedback and online learning, together with longitudinal studies in real operational environments to assess long-term robustness, scalability, and user experience.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Z.R. designed the system architecture, implemented the conversational modules, and wrote the manuscript; M.A.P. and S.J. contributed to the development of the intent classification and service selection modules; U.A.

supported the experimental evaluation and dataset preparation; A.M. and A.V. supervised the project, provided technical guidance, and contributed to system design discussions; S.R. coordinated the research, supervised the methodology, and reviewed the manuscript. All authors had approved the final version.

FUNDING

This research was funded by MegaSec LLC. No specific grant number was assigned.

ACKNOWLEDGMENT

This research was conducted at the AI Laboratory of MegaSec LLC and the Center for Data Analytics Research of ADA University.

REFERENCES

[1] X. Ni, Y. Wang, T. Feng *et al.,* "Generative AI in action: Field experimental evidence on worker performance in e-commerce customer service operations," *SSRN*, 2024. doi: 10.2139/ssrn.5012601

[2] E. GołabAndrzejak, "AI-powered customer relationship management-generative AI-based CRM-einstein GPT, Sugar CRM, and MS Dynamics 365," *Procedia Computer Science*, vol. 246, pp. 1790–1799, 2024.

[3] X. Wang, H. Dai, S. Gao *et al.*, "Characteristic AI agents via large language models," in *Proc. Joint Int. Conf. Comput. Linguistics, Lang. Resources and Evaluation (LREC-COLING)*, 2024, pp. 3016–3027.

[4] K. K. Nirala, N. K. Singh, and V. S. Purani, "A survey on providing customer and public administration based services using AI: Chatbot," *Multimedia Tools and Applications*, vol. 81, pp. 22215–22246, 2022.

[5] T. Brown, B. Mann, N. Ryder *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[6] Z. Ji, N. Lee, R. Frieske *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, 248, 2023.

[7] S. Yao, J. Zhao, D. Yu *et al.*, "ReAct: Synergizing reasoning and acting in language models," arXiv preprint, arXiv:2210.03629, 2023. doi: 10.48550/arXiv.2210.03629

[8] Q. Wu, G. Bansal, J. Zhang *et al.*, "AutoGen: Enabling next-generation LLM applications via multi-agent conversations," arXiv preprint, arXiv:2308.08155, 2023. doi: 10.48550/arXiv.2308.08155

[9] L. Weidinger, J. Uesato, M. Rauh *et al.*, "Taxonomy of risks posed by language models," in *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAccT)*, 2022, pp. 214–229.

[10] H. Zhang, H. Song, S. Li *et al.*, "A survey of controllable text generation using transformer-based pre-trained language models," *ACM Computing Surveys*, vol. 56, no. 3, 64, 2023.

[11] L. Huang, W. Yu, W. Ma *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Transactions on Information Systems*, vol. 43, no. 2, 42, 2025.

[12] J. Maynez, S. Narayan, B. Bohnet *et al.*, "On faithfulness and factuality in abstractive summarization," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2020, pp. 1906–1919.

[13] T. Schick, J. Dwivedi-Yu, R. Dessi *et al.*, "Toolformer: Language models can teach themselves to use tools," *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[14] L. Gao, A. Madaan, S. Zhou *et al.*, "PAL: Program-aided language models," in *Proc. 40th Int. Conf. Machine Learning (ICML)*, 2023, pp. 10764–10799.

[15] G. Li, H. Hammoud, H. Itani *et al.*, "CAMEL: Communicative agents for mind exploration of large language model society," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51991–52008, 2023.

[16] S. Hong, M. Zhuge, J. Chen *et al.*, "MetaGPT: Meta programming for a multi-agent collaborative framework," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024.

[17] E. M. Bender, T. Gebru, A. McMillan-Major *et al.*, "On the dangers of stochastic parrots: Can language models be too big?" in *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAccT)*, 2021, pp. 610–623.

[18] P. Lewis, E. Perez, A. Piktus *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[19] T. Mu, A. Helyar, J. Heidecke *et al.*, "Rule-based rewards for language model safety," *Advances in Neural Information Processing Systems*, vol. 37, pp. 108877–108901, 2024.

[20] L. Ouyang, J. Wu, X. Jiang *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27730–27744, 2022.

[21] T. Alizada, U. Suleymanov, and Z. Rustamov, "Contextualized word embeddings in Azerbaijani language," in *Proc. IEEE 18th Int. Conf. Application of Information and Communication Technologies (AICT)*, 2024, pp. 1–6.

[22] N. Guliyev, Z. Rustamov, and S. Rustamov, "Analysis of public sentiment in Azerbaijani news and social media," in *Proc. Int. Conf. Computing, Machine Learning and Data Science*, 2024, pp. 1–6.