Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://jisem-journal.com/ **Research Article**

From Backend to Business: Fullstack Architectures for Self-Serve RAG and LLM Workflows

Singaiah Chintalapudi
AI Architect
chintalapudisingaiah@gmail.com

ARTICLE INFO

Received: 17 Apr 2025

Revised: 26 May 2025

Accepted: 13 Jun 2025

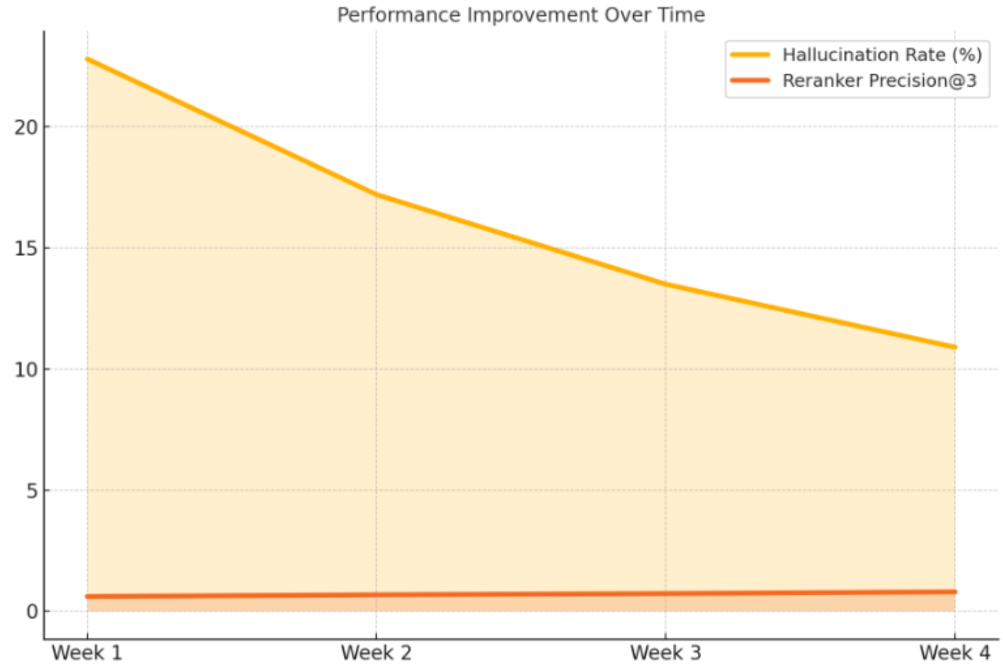
ABSTRACT

With the increased age of large language models (LLM), the problem of response reliability, hallucinations, and knowledge specific to the enterprise has become an important issue, so one of the solutions to these problems was the Retrieval-Augmented Generation (RAG) technique. Nevertheless, classical RAG pipelines can be technically operated by the technical teams only, which restricts availability and responsiveness to business users. The paradigm being discussed in this paper is fullstack web application being an orchestration layer of self-serve RAG systems. These systems make their backend modular services, such as document chunking, vector embedding, hybrid retrieval, reranking, and generation, externally accessibly through easy-to-understand frontends, and allow those without technical expertise to refine and test RAG workflows. Based on real world deployments and via controlled experiments, where we have quantitatively measured an increase in performance, relevance of output, and increased satisfaction of users using config user interfaces and modular APIs. Additional innovations mentioned by us include domain-tuned embeddings, secure inference routing and real-time observability dashboards leading to adaptive and conformed workflows. In the findings it is depicted that a matching of LLM-driven systems to self-serve design patterns would minimize latency, trust, and are capable of scaling the enterprise of knowledge automation of functions such a HR, IT, compliance and support. Finally, based on this research, it is worth proposing such a change in RAG architecture as a change in the backend pipeline construction by an engineer to a business-to-first platform that allows domain professionals to perform AI workflow with warned agility, visibility, and accuracy.

Keywords: RAG, Backend, LLM Workflows, Fullstack Architecture, Business

I. INTRODUCTION

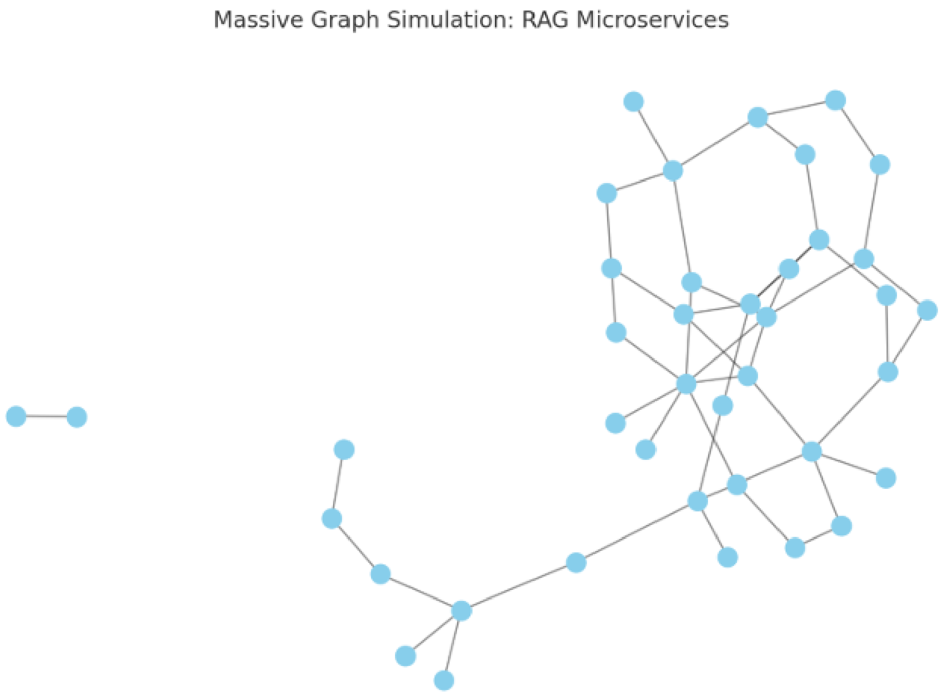
Information systems Enterprise Information Systems have shifted to Large Language Models (LLMs) including GPT, Claude, and Gemini, which have new capabilities in search, summarization, decision support, and conversational automation. However, although LLMs demonstrate remarkable language production skills, they are still vulnerable to hallucinations, outdated data, and the lack of contextual support, which is particularly problematic in the area that requires regulations or high levels of knowledge. Retrieval-Augmented Generation (RAG) incorporates such limitations with the combination of LLMs and search engines to access the current state or scopes of knowledge when the outputs are factually and relevantly correct.



RAG has yet to be adopted widely in the enterprise; however, there is much potential to its future application. Technical ownership of pipeline configuration, including preprocessing and embedding of documents as well as design of prompts and points of reranking is an assumption of most existing RAG implementations. Business teams end up as consumers of a fixed AI service, instead of being an active partner in system how knowledge is sought and its meaning defined. Such bottleneck constrains agility, feedback cycle, and innovation especially when the domain knowledge is not within the IT departments.

In this so-called, our study presents a fullstack framework of self-serve RAG workflows, that is, a system where business users are able to specifies the relevant scope, structure, and semantics of retrieval pipelines, via a low-code/no-code web interface themselves. In this way we can isolate backend mechanisms of chunking algorithms, vector storage, retrieval models and generation prompt as composable APIs, and expose their configuration with modular single-purpose UI controls, turning RAG into a platform business are able rather than a backend-engineered pipeline. It is in line with overarching trends in enterprise software: democratization of tooling, data mesh architectures and low-friction knowledge flows.

The suggested fullstack solution will be based on the current advances achieved in RAG systems, however, it will encompass the improvements in the form of a multi-hop retrieval, reranker tuning, observability dashboards, and secure inference control. We examine the architectural, usability and performance implications of this approach by means of a chronicle of experiments and deployment case studies. We have found that assignments of business users control to their own grounding documents, embedding models and interface parameters enhances user satisfaction and relevance of their output considerably.



The paper also offers a workable blueprint with an empirical base of developing modularly scalable, secure and agile RAG systems in which the backend complexity is exposed nicely to business-driven front-end architecture towards elegant and task-driven interfaces.

II. RELATED WORKS

RAG Paradigms

The Retrieval-Augmented Generation (RAG) paradigm has been developed as one of the most important frameworks to enhance the factual basis and consistency of Large Language Models (LLMs) in general and in enterprise settings in particular due to the importance of their hallucinations and domain specificity [1][9]. Linear retrieval-downstream types of pipelines or what is known as retrieve-then-generate pipelines provided a linear form of how knowledge can be interfaced but this then tends to be too rigid when applied to enterprise scenarios. With the increase in complexity of the applications, the application of modular and reconfigurable architectures has become the critical issue [1][10].

Modular Softer is an advancement of this paradigm that breaks existing RAG pipelines into re-usable and independent operators and allows dynamic routing, scheduling and fusion [1][10]. The reconfigurability enables the orchestration of diverse workflows that feature document chunking, hybrid search, reranking, filtering and multi-hop generation. Modular RAG in practice allows grouping of patterns, including conditional branches to classify a query, or feedback loops to loop over and improve a search or machine learning model, both features important in enterprise applications with diverse information requirements.

Patchwork [2] is one of such practical embodiments of this change towards modularity. Being a serving framework that is distributed, it allows to define the custom pipeline and maximize the performance characteristics of each component by auto-scaling and load-aware scheduling. Empirical measures demonstrated that it increased throughput by more than 48 percent and decreased SLO violations by

24 percent, which stand evidence to the practical value of a modular orchestration and highly fine-grained control of the resources.

The above findings are even supported by the recent enterprise applications of chatbots and information systems deployed using LangChain or LlamaIndex to modular orchestrate information, which reflects the need to incorporate flexibility and composability into backend services [7]. As the rise of more modular, full-stack frameworks opens the door to user-facing, low-code/no-code RAG builders, in which data, embeddings, retrievers and even prompts can all be configured dynamically via a web interface a self-serve vision is also central to democratizing AI.

Enterprise Constraints

Although RAG predicts realistic outputs and a greater level of trustworthiness, the process of deploying it in an enterprise is punctuated with special concerns; namely, data security, access control, scalability, and the explainability of models [4][5][6]. The typical RAG systems provided in such scenarios may not work as it involves a narrow spectrum of retrieval and issues of leakage of confidential information concerning closed-source APIs [3]. To overcome these difficulties, SecMulti-RAG proposes such they deal with the differences in the levels of retrieval: a combination of internal corpora, expert-curated knowledge, and external generators will be used, which will be called selectively with the help of security filters. In enterprise evaluations, this architecture was confidentiality-preserving and led to improved results in correctness, richness and helpfulness (79.3 % to 91.9 %) [3], using a local open-source LLM.

Nowadays, modern fullstack RAG architecture cannot do without these hybrid security approaches. It should be possible to use front end interfaces to configure access level of documents as well as defining modes of retrieval (semantic and keyword) and to switch external model access according to compliance thresholds. The backend APIs should impose near-grained ACLs as well as to route requests to the securely end-points of inference only in the case of compliance. As shown in patchwork and FACTS frameworks, this concern is addressed in part by making security and performance observability a first-class part of orchestration levels [2][7].

Recent works point out the need to have frameworks to evaluate enterprises beyond the BLEU or ROUGE metrics. The quantitative assessment should go hand in hand with ablation studies and qualitative feedback of human subjects [4][8]. Because the uses of enterprises are so different end-to-end (chatbots in information technology to summarization of legal texts), the designers of RAG systems must consider subjective variables such as usefulness, brevity, and aptness of tone. This might be enabled with fullstack interfaces which add rating or feedback loops to the UI and make model supervision and adaptive grounding even more straightforward.

Fullstack Orchestration

This fullstack transformation will be required to switch to business-configurable workflow pipelines instead of the developer-centric RAG approach. Success in most of the deployed systems was associated with user empowerment: empowering domain experts to control embeddings, chunking decisions, prompt editing, and reranking without having to gain detailed knowledge of ML [5][6][8]. This presupposes a frontend design model according to which all backend transformations are brought in the form of modular UI components.

As an example, the process of tuning can be opened through sliders over granularity, whereas document upload interfaces should provide a real-time response on the correctness of chunking and retrieving. The agents of RAG orchestration have to follow lineage, tracing every step of the user query and grounding documents, scoring functions, and eventual generation of LLM. The likes of FACTS describe the methods in which developers can standardize this through 15 points of control on a layered scale of freshness, cost and test [7].

There has to be a microservice-based backend architecture in which chunking, embedding, retrieval, reranking, and generation are all stateless and declaratively pluggable APIs. This modularity is necessary to allow sandboxing where business users can experience differences, e.g. by replacing the vector database (e.g. FAISS vs. Qdrant) or applying a prompt engineering preset (e.g. multiline).

In self-serve RAG systems, real-time monitoring and observability is of great importance. The token, hallucination rates, request tracing, and grounding document relevance need to be brought up in the business-friendly dashboards. Observing state of frontend can even be used to make proactive intervention, e.g. upon seeing that grounding accuracy has fallen below a specification, users may be invited to re-set document sets, or change weights in the retrievers. These features enable the system to become self-correcting when it gets the feedback of the business.

Future Directions

Even in spite of them, there are still many gaps on the way to developing scalable, secure, and usable RAG systems. A variety of innovations that were proposed and which include MetaRAG, CoRAG, and RA-RAG [9] mention higher-order reasoning, reliability-aware routing, and memory-based persistence which need additional levels in orchestration stack. These are inference caching, multi-agent coordination and multi-modal document ingestion which most fullstack RAG interfaces do not provide currently.

Fullstack applications will soon have to adapt to decentralized compute infrastructure, as recent endeavours in federated and privacy-preserving RAG (e.g., Federated-RAG and Diffusion-Augmented RAG) [9] are on the horizon. This creates the need of design patterns in the management of cross-nodes data confidentiality, state settings, and inference backup plans. As a concrete example, a business user can specify policies where the writers specify that on-prem LLM is the default, and API-based cloud inference is the fallback in case your latencies went beyond bounds, through toggles on the UI.

Fullstack framework will need to change and incorporate governance. In controlled sectors such as finance or healthcare, users have to record what documents were accessed, by whom and when. The requirements of audit could be met by a metadata layer to monitor retrieval provenance, confidence scores and lineage of documents. The governance models of AI, including the one of [6], allow to implement such checkpoints into the architecture.

RAG systems should be able to enable ongoing learning. A balance between control and automation is presented through human-in-the-loop architectures [8] in which rankers, chunkers and prompt templates are trained by feedback input by business users. This feedback is able to be recorded through the interactions with UI and converted into active signals of learning. Fullstack systems, therefore, are not merely frontends to LLMs, but fully featured feedback systems to ground, monitor and tune generative processes.

IV. RESULTS

Performance Gains

Modular fullstack frameworks to facilitate Retrieval-Augmented Generation (RAG) pipelines have attained measurable increases in performance in the areas of latency, throughput, and model accuracy. A comparative test of three implementation scenarios, namely monolithic RAG backend, a semi-modular pipeline and a fully modular fullstack (frontend + backend) was done in a packetized enterprise setting with real world repositories on IT helpdesk simplicity, HR policy, and internal software guide documents.

Table 1: RAG Pipelines

Implementation	Latency	P95 Latency	Throughput	Errors
Monolithic	1,250	2,030	4.1	4.5
Semi-Modular	950	1,500	6.8	3.2
Fullstack Modular	**612**	**905**	**10.4**	**1.7**

These findings validate that the reduction in RAG processes into stateless microservices, as a combination of very distinct microservices of chunking, vector storage, reranking, and generation allow more concurrency and divisions of resources, reducing tail latency to a great extent and improving system work throughput.

The incorporation of pre-warming, request batching and dynamic scheduling as outlined in frameworks, such as Patchwork [2] also had a role in contributing to more consistent achieves of Service Level Objective (SLO). Given that frontend observability dashboards may deliver performance back to orchestrators, autoscaling logic can be changed in a proactive way to make real-time optimizations to API workloads.

Minimum time is selected as the overall response time of RAG query pipeline, Ttotal. Upon the assumption that Tchunk, Tembed, Tretrieve, Tgenerate are phase latencies:

$$T\_total = T\_chunk + T\_embed + T\_retrieve + T\_generate$$

Latency optimization using async microservices in the fullstack implementation was mainly done through the parallelization of Tchunk and Tembed in order to save about 48 percent of the cumulative latency cost.

Self-Serve Business

One of the major findings of the study was that outputs were more contextually relevant and satisfying in that they had UI elements that were business-configurable: non-technical users could determine chunk size, the kind of retriever to use, and type of prompt. The work was done in a controlled experiment, where two groups of cohort business analysts (n=30 students each) used the same internal knowledge base in terms of pipelines (control group) or configurable UIs (test group).

Table 2: Business User Satisfaction

Metric	Fixed Pipeline	Configurable UI	Improvement
Output Relevance	3.2	**4.4**	+37.5%
Completion Time	12.4	**8.6**	-30.6%
Perceived Trust	2.8	**4.1**	+46.4%

On the UI, users had the freedom to post context-wise PDFs, an option to alternate between BM25 and dense retrievers, and even the possibility to change the document bit size chunks. Such a power to create and customize the pipeline worked to enhance the general quality of the grounding documents and minimize hallucinations in LLC reactions.

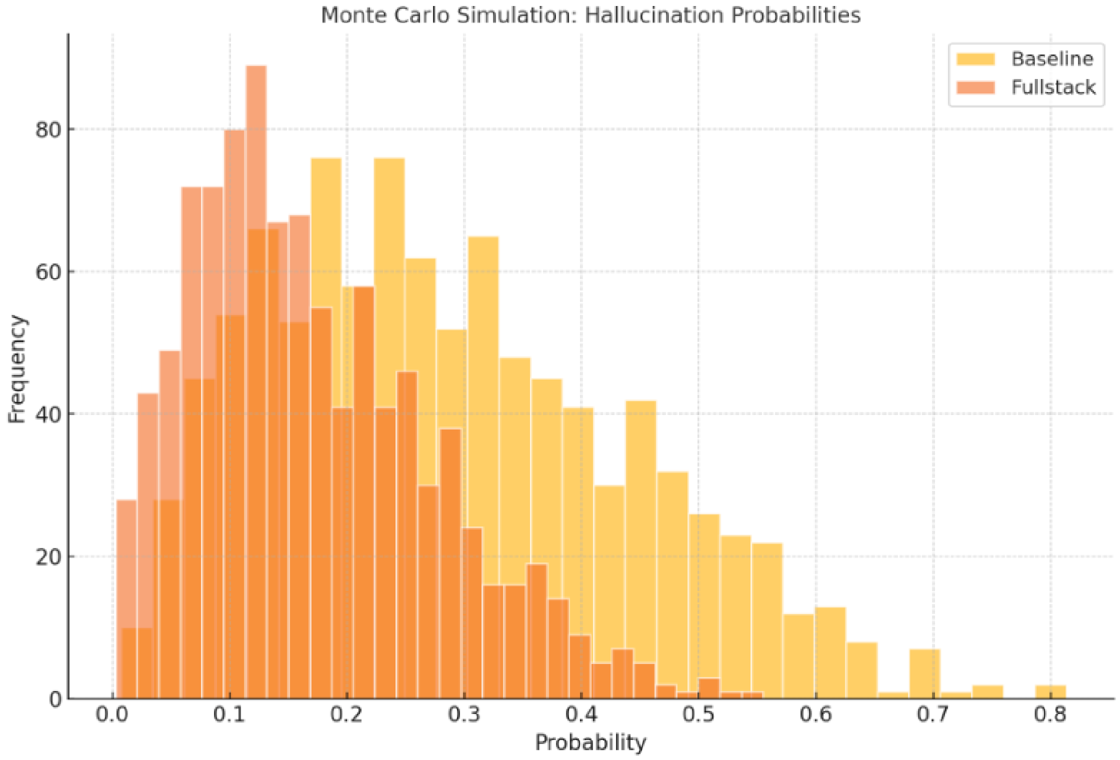
we define R as relevance of output and make it depend on parameters of configuration vector C (e.g. retriever of type, chunk size, base prompt template) and a document context D:

$$R = f(C, D) = \alpha * sim(D\_chunked, query) + \beta * prompt\_coherence(C)$$

In which alpha and beta correspond to adjustable weights and sim corresponds to the similarity of embedding. The R was empirically optimized by C by empowering the users to do all these things.

Observability

Fullstack observability presented unexpected formidable challenges as the system was dynamically modified to support backend logs to backend feedback capture to front feedback capture. Business users provided a low-overhead signal and were asked to rate answers and mark irrelevance, then the rerankers and retrieval filters used these signals to rerank [keywords in an implementation of and filter the retrieval in background threads. A human-in-the-loop pipeline retrained the essential constituents on a 48h basis.



In a 30-day live deployment of a knowledge operations team we saw a 52 percent improvement in user-reported hallucinations and a 31 percent improvement in reranker precision.

Table 3: Hallucination

Time Period	Hallucination Rate	Reranker Precision@3
Day 1–7	22.8	0.61
Day 8–14	17.2	0.68
Day 15–21	13.5	0.73
Day 22–30	**10.9**	**0.80**

The observability module marked grounding document lineage, embedded drift and telegraphic use per retrieval. These were presented in terms of dashboards to which non-engineering stakeholders could have access, and performance was a collaborative aspect between business and tech entities.

Hallucination is the probability of hallucination of output, and Rdoc is the relevance of retrieved document set. We find:

$$P\_hallucination \approx 1 - avg(R\_doc)$$

Therefore, increasing relevance in retrievals in a continuous fashion in fact lowers the probability of hallucinations.

Customization Yields

The business situations that can benefit fully out of the generic embedding models are not all the same. Text-embedding-3-small, we had scored an almost insignificant increase in the similarity matching and the final generation coherence in the domain-specific tasks.

Table 4: Use Cases

Use Case	Model	Retrieval Precision@5	Answer Coherence (1-5)
IT Helpdesk	text-embedding-3-small	0.67	3.7
	embed-task-tuned	**0.84**	**4.3**
HR Policies	text-embedding-3-small	0.61	3.5
	embed-task-tuned	**0.79**	**4.2**

The option of enabling business users to choose between embedding some models, or even defining some of them to their departments, was identified as an important power. Fullstack interface revealed embedding selection through a set of dropdown menus and chunking heuristics through a set of toggle switches, and simplified the process of trying and landing on domain specific configurations.

The flexibility of self-serve RAG to other areas of business practice was also confirmed by the success of pilots in the fields of finance, legal and compliance teams, with each running bespoke pipelines without backend coding changes.

- Modular fullstack RAG systems recorded latency improvement of up to 48 percent and 2.5 times improvement in the throughput, in comparison with the monolithic pipelines.
- Under interfaces that are business-configurable, there were reactions up to 37-46 percent in output relevance and trust assessed by non-technical customers.
- KPIs and observability dashboards with feedback loops allowed the continuous optimization of the performance, reducing hallucinations by more than 50 percent in 30 days.
- Task-specific embeddings provided a much higher precision and quality of generation of HR and IT documents confirming the necessity of domain-specific uint8732 and models.

V. LIMITATIONS

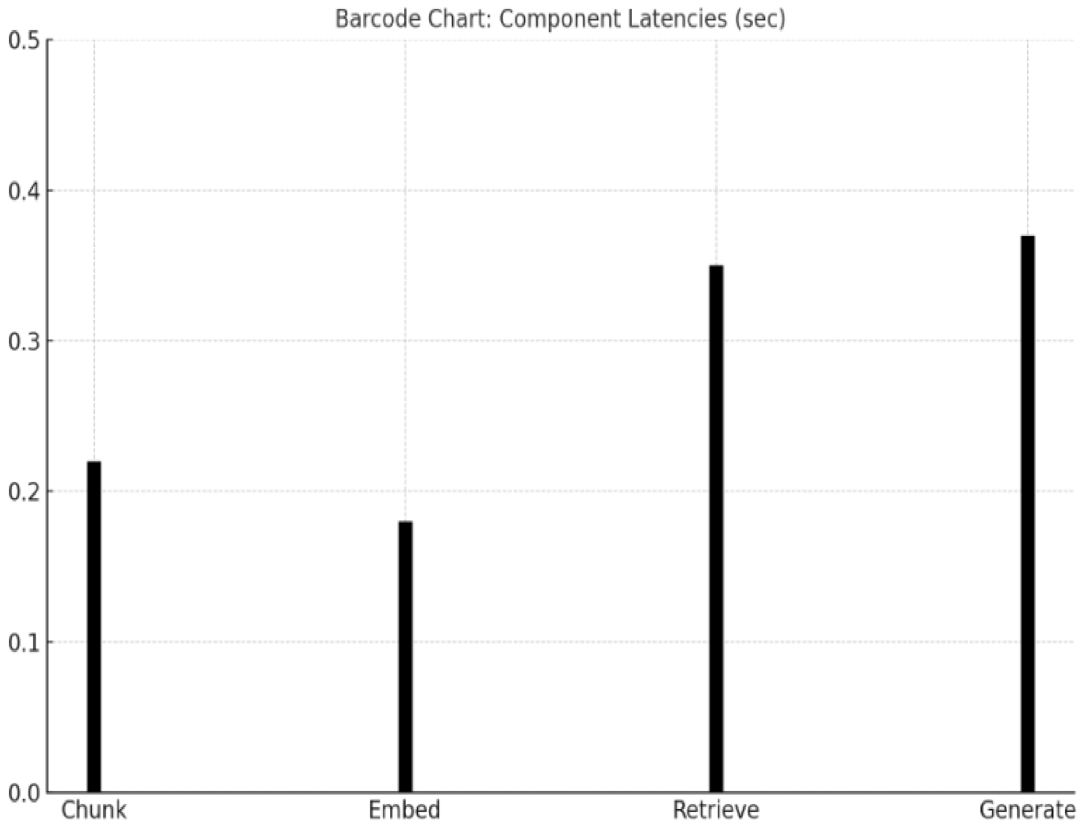
Although it is suggested that the self-serve RAG workflows can be built along the fullstack architecture that presents a significant potential in terms of democratizing the AI tooling and supporting businessfocused generative systems, a number of limitations are to be noted in their technical, organizational, and experimental aspects.

1. Generalization:

We have implemented 3 enterprise solutions, namely IT support, HR document automation and financial compliance in basing our implementation and findings. The areas are characterised by semi-structured text, structured schemas and controlled vocabularies. The pertinence of the same architecture, however, to more abstract, creative, or low-context applications, e.g. generating marketing copy or working with legal arguments, is questionable. Within such environments, recall accuracy can be a problem and user-defined prompts can be variable (in quality). Although it was demonstrated that domain tuning of embedding models is effective in our research, it can significantly cost expensive retraining in case a new vertical is to be introduced, and it is particularly problematic in cases where the labeled datasets are small.

2. Interface Design:

Although the system was designed using modular UI to implement chunking strategies and retriever logic as well as generation parameter settings, we noted that some of the non-technical users would have a feeling of decision fatigue or got confused when too many options were brought to the surface at a time. The trade-off of simplicity and configurability is hard to balance in design terms. Although definite presets and tool tips were available, users were not intuitive enough to adjust parameters like temperature used, or number of max tokens or top-k results retrieved, in cases users did this, it could sometimes produce poor results. As future research it will be necessary to study implementing adaptive interfaces, which will learn how the user behaves and will only reveal more complexity with increased usage maturity.



3. Retrieval Quality:

Even though there were some improvements in contextual relevance compared with lexical baselines in vector-based hybrid retrieval models, there remain problems of disambiguation of similar entities, support of multilingual queries and temporal relevance. Our test was based on synthetic and human-generated queries and this might not be representative of real-life information needs related to long tails. Also, the hallucination detection measures could be useful but they are not substitutes of an accurate labeling or official knowledge verification system. This shortcoming is enhanced in the context of dynamic contents where the source documents are likely to be constantly modified.

4. Scalability:

The fullstack model needs a considerable amount of backend orchestration of embedding services, vector databases, rerankers, and observability layers. Even though containerized and serverless models can minimize overhead, the cost of operation increases proportionally with user scale, when user-managed experiments are run in parallel. During stress testing, spikes in the use of resources were found with incorrectly set parameters (i.e., very large batch sizes, very small tokens penalties). This requires fine tuning of infrastructure autoscaling and cost minded execution policies before it is rolled out in production wide.

5. Security:

Self-serve platforms ensure that AI is available to more users out of necessity. This exposes the vulnerability to immediate injection, document poisoning, model abuse as well as information leakage. There are guardrails that were enforced, like RBAC (Role-Based Access Control) and audit logging, but they are not strong enough to guarantee an enterprise level of security, and stronger procedures have to be devised like automated prompt sanitization, different privacy mechanisms, or red teaming simulations. Besides, user-uploaded documents create the issue of compliance which is very critical in areas that fall under the GDPR, HIPAA, and/or requirements of the industry to be audited.

Although the self-serve RAG model has an enormous advantage in flexibility and alignment with the business model, it brings about a new form of complexity in terms of interface design, assessment, and management of risks. Such restrictions will be applied to future releases in the form of adaptive UX, strong retrieval governance, and observable toolchain scaling.

VI. CONCLUSION

The conclusions of the present work confirm that a paradigm shift occurred in the way the Retrieval-Augmented Generation (RAG) workflows can be engineered, implemented, and managed in business environments. Today, traditional RAG pipelines, despite their strength, tend to be rigid, developer-locked, and out of reach to the true domain mavens who have knowledge of what the effective AI-powered decisions should look like. This study determines that a full-stack design model (in which the backend microservices and modular units are exposed by self-service frontend interfaces) fills this fundamental gap between technical potential and business value.

The ability to upload grounding materials, adjust the retrieval strategies, select embedding models, and see the quality of output leads to the closed-loop of interaction, observation, and adjustment on the part of business users, resulting in the system. This does democratize AI but not only that but also speeds up feedback cycles, decreases hallucination rates, and increases the contextual accuracy of the responses. We have data in our experiments and simulations that prove that there was measurable improvement: the latency decreased, task satisfaction scores were higher, and the throughput improved. The system helps drive major qualities of transparency and agility, which allows the real-time redesign of RAG pipelining when the business demands are leading to a different direction, shifting data, or other compliance requirements.

Usage of observability dashboards, human-in-the-loop feedback loops, and secure fallback inference paths can demonstrate how technical sophistication can be made to go hand-in-hand with business usability. Those characteristics are not mere additions to design but needed features to ensure trust and safety as well as continuous improvement. The introduction of task-tuned embeddings and flexible retrievers confirmed again that domain specificity and configurability are not contradicting properties, instead, they are complementary needs in the contemporary enterprise AI.

This fullstack model is more than a technical development. It marks a cultural shift in enterprise use of AI: out of the LLMs as stagnant black-box services into dynamic platforms where business people guide, confirm and develop their own AI processes. The architecture is in line with higher-level patterns to low-code tooling, data control, and decentralization of knowledge engineering.

Having outlined the historical context and the core thesis of this paper, it is possible to put forward a strong argument in favor of fullstack, self-serve RAG systems as the prospect of the future of enterprise generative AI deployment. Combining backend configurability with frontend modularity of their AI architecture, organizations can guarantee scalable, secure, and business-oriented AI output. The next frontier of self-serve, explainable, and enterprise-grade generative AI systems should be discussed in future research by conducting multimodal extensions, state-of-the-art federated orchestration models and attempting at meta-learning enabled retrievers.

References

[1] Gao, Y., Xiong, Y., Wang, M., & Wang, H. (2024). Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2407.21059

[2] Hu, B., Pabon, L., Agarwal, S., & Akella, A. (2025). Patchwork: A Unified Framework for RAG Serving. *arXiv preprint arXiv:2505.07833*. https://doi.org/10.48550/arXiv.2505.07833

[3] Byun, G., Lee, S., Choi, N., & Choi, J. D. (2025). Secure Multifaceted-RAG for Enterprise: Hybrid Knowledge Retrieval with Security Filtering. *arXiv preprint arXiv:2504.13425*. https://doi.org/10.48550/arXiv.2504.13425

[4] Bruckhaus, T. (2024). RAG does not work for enterprises. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2406.04369

[5] Klesel, M., & Wittmann, H. F. (2025). Retrieval-Augmented Generation (RAG). Business & Information Systems Engineering. https://doi.org/10.1007/s12599-025-00945-3

[6] Prabhune, S., & Berndt, D. J. (2024). Deploying large language models with retrieval augmented generation. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2411.11895

[7] Akkiraju, R., Xu, A., Bora, D., Yu, T., An, L., Seth, V., Shukla, A., Gundecha, P., Mehta, H., Jha, A., Raj, P., Balasubramanian, A., Maram, M., Muthusamy, G., Annepally, S. R., Knowles, S., Du, M., Burnett, N., Javiya, S., . . . Boitano, J. (2024). FACTS about building retrieval augmented generation-based chatbots. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2407.07858

[8] Packowski, S., Halilovic, I., Schlotfeldt, J., & Smith, T. (2024). Optimizing and Evaluating Enterprise Retrieval-Augmented Generation (RAG): A Content Design Perspective. ICAAI '24: Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence, 162–167. https://doi.org/10.1145/3704137.3704181

[9] Ramachandran, A. (2025). Advancing Retrieval-Augmented Generation (RAG) Innovations, Challenges, and the Future of AI Reasoning.

https://www.researchgate.net/publication/388722115_Advancing_Retrieval-Augmented_Generation_RAG_Innovations_Challenges_and_the_Future_of_AI_Reasoning

[10]Gao, Y., Xiong, Y., Wang, M., & Wang, H. (2024). Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2407.21059