Explainable Retrieval-Augmented Generation for Document Intelligence: A Modular RAG Pipeline with Guardrails and Citation Tracing

Author: Arnav Singh Bhardwaj

M.Sc. Data Science
University of Europe for Applied Sciences

Supervisors: Prof. Dr. Talha Ali Khan, Prof. Shan Faiz

Date: February 09, 2026

Abstract

Large language models (LLMs) enable fluent question answering and summarization, yet they are not inherently grounded in an organization's private document collections and can produce confident but unsupported statements. Retrieval-Augmented Generation (RAG) addresses this reliability gap by coupling a retriever over an external corpus with a generator that conditions on retrieved evidence. This article presents an explainable document intelligence prototype that implements a modular RAG pipeline for unstructured PDFs, emphasizing transparency, citation tracing, and conservative behavior when evidence is missing. The system ingests documents, performs text extraction and normalization, segments text into overlapping chunks, embeds chunks into a vector space, and stores them in a vector database for similarity search. At query time, the system retrieves the top-k most relevant chunks, constructs a constrained prompt, and generates an answer that is explicitly grounded in the retrieved context and accompanied by page-level source references. A guardrail policy returns an explicit "I don't know" response when the indexed corpus does not contain sufficient evidence. We report a structured qualitative evaluation using representative query types (factual, contextual, and out-of-scope), and we analyze explainability by verifying citation alignment with the cited passages. Results show that the prototype improves answer traceability and reduces hallucination risk relative to LLM-only interaction patterns, while highlighting retrieval and chunking as dominant factors for end-to-end reliability. The work contributes a reproducible implementation blueprint, a practical case-study driven evaluation protocol, and design guidelines for deploying trustworthy RAG systems in document-centric settings.

Keywords

Retrieval-Augmented Generation; Document intelligence; Explainable AI; Semantic retrieval; Vector databases; Guardrails; Citation tracing

I. INTRODUCTION

Organizations increasingly rely on digital documents—contracts, policies, financial statements, technical manuals, and research reports—to make time-sensitive decisions. While these sources are rich in information, they are difficult to search and synthesize at scale, especially when knowledge is distributed across long documents and heterogeneous formats (e.g., PDF reports with repeated headers, tables, and annexes).

LLMs provide a natural interaction paradigm by allowing users to ask questions in plain language. However, in document-centric tasks, an LLM's parametric knowledge is insufficient: it may not contain the needed fact, and it cannot reliably justify where a statement came from. In regulated environments, a correct-looking but unverifiable answer is not acceptable.

RAG systems combine retrieval with generation by retrieving evidence passages from an external corpus and conditioning the LLM on those passages. This coupling provides two core benefits: (i) factual grounding, because the model is constrained by retrieved evidence, and (ii) provenance, because the system can return citations that point to the source text. Despite these benefits, practical RAG systems still fail when retrieval misses the right evidence, when chunking fragments meaning, or when the generator overgeneralizes from weak context.

This article presents a modular, explainable RAG prototype for document intelligence and offers a case-study driven evaluation aligned with three criteria that matter to end-users: correctness, explainability, and trustworthiness. The main contributions are:

- A clear end-to-end architecture that separates ingestion, indexing, retrieval, and generation.
- A citation and metadata strategy that supports page-level traceability.
- A guardrail mechanism that explicitly communicates uncertainty.
- A qualitative evaluation protocol with representative query categories and evidence checks.

II. RELATED WORK

RAG builds on advances in both information retrieval and neural language modeling. Classical lexical retrieval methods (e.g., BM25) remain effective baselines due to speed and interpretability, but they depend on term overlap between query and document. Dense retrieval replaces sparse term vectors with dense embeddings, enabling semantic matching between paraphrased queries and relevant passages.

Document intelligence research highlights that document understanding is not purely linguistic: layout structure, repetitive boilerplate, and extraction artifacts influence downstream retrieval. Layout-aware models (e.g., LayoutLM-family approaches) demonstrate that spatial signals can improve understanding for visually rich documents. In practice, many deployed

RAG systems still operate on extracted text, making the extraction and chunking steps critical reliability factors.

LLM hallucination has been widely discussed as a core reliability limitation. Retrieval can reduce hallucination risk, but grounding is not a guarantee: if retrieved evidence is irrelevant or ambiguous, the generator may still produce misleading content. Recent surveys emphasize the need to evaluate RAG as an end-to-end system and to incorporate faithfulness checks that compare generated claims against retrieved evidence.

Explainability in RAG systems is often approximated through citations. However, citation quality varies: a passage can be topically related but not supportive. Stronger designs treat citations as first-class outputs by tracking which chunks were retrieved, which were used in prompt construction, and by presenting users with direct links or page references to validate claims. This work follows that design principle by coupling retrieval metadata with answer presentation.

III. METHOD

The proposed system follows the standard RAG pipeline, implemented as a set of explicit stages: (1) document ingestion and parsing, (2) text normalization and chunking, (3) embedding generation and vector indexing, (4) query embedding and top-k retrieval, (5) prompt construction and constrained answer generation, and (6) citation tracing with guardrails.

Let a document be represented as a sequence of chunks $\{C_i\}$. Each chunk is embedded into a d-dimensional vector space using an embedding model $f(\cdot)$:

(1) $e_i = f(C_i)$

Given a user query q, the system computes its embedding:

(2) $e_q = f(q)$

Retrieval selects the top-k chunks by cosine similarity:

(3) $\text{sim}(e_q, e_i) = (e_q \cdot e_i) / (\|e_q\| \|e_i\|)$

The generator receives the query and the retrieved context $\{C_{i1}, \ldots, C_{ik}\}$. The prompt explicitly instructs the model to answer only from the provided context and to return an abstention response if the context does not contain sufficient evidence.

Guardrails are implemented as a conservative policy: if retrieval returns no chunks above a minimum similarity threshold, or if the generator cannot find evidence within the context, the system returns “I don’t know” and surfaces the retrieved evidence (if any) for transparency.

IV. SYSTEM ARCHITECTURE

Fig. 1 summarizes the high-level architecture. The design emphasizes modularity: each stage produces inspectable intermediate artifacts (extracted text, chunk boundaries, embedding vectors, retrieved chunk IDs) to support debugging and evaluation.

The ingestion module parses PDFs, preserves page boundaries, removes repeated headers/footers, and normalizes whitespace and hyphenation artifacts. Chunking uses a fixed-size window with overlap to avoid breaking definitions and multi-sentence explanations across boundaries.

The vector indexing module stores embeddings and metadata in a persistent vector database. Metadata includes document identifier, page number, and chunk offsets. This metadata is carried through retrieval to enable citations in the UI.

The UI is implemented in Streamlit to facilitate rapid experimentation. Users can upload documents, configure retrieval depth (top-k), ask questions, and inspect both the generated answer and its cited sources. This interface supports the qualitative evaluation by making system behavior observable.

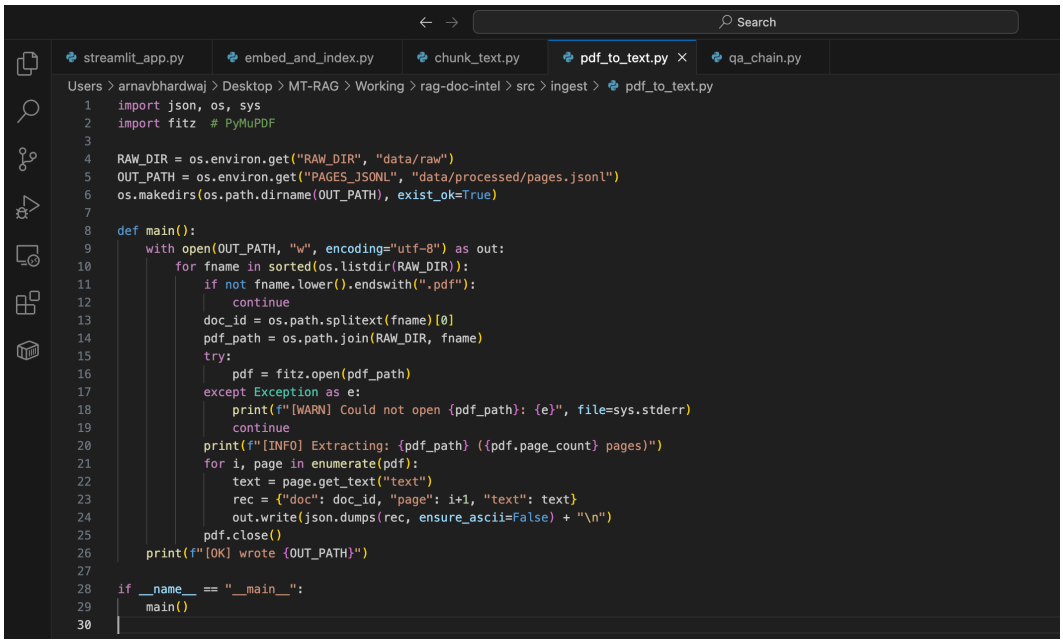

Fig. 1. Document ingestion: PDF-to-text extraction view in the prototype.

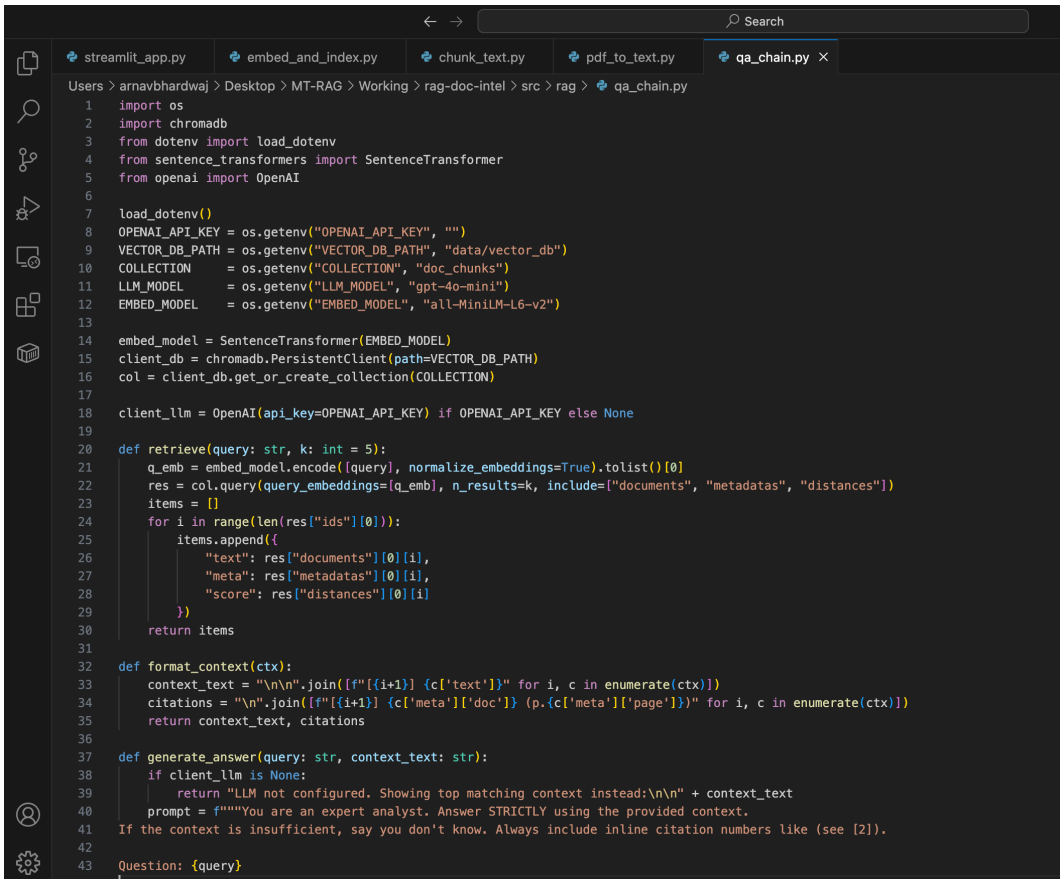

Fig. 2. RAG processing chain illustrating retrieval, prompt assembly, and answer generation.

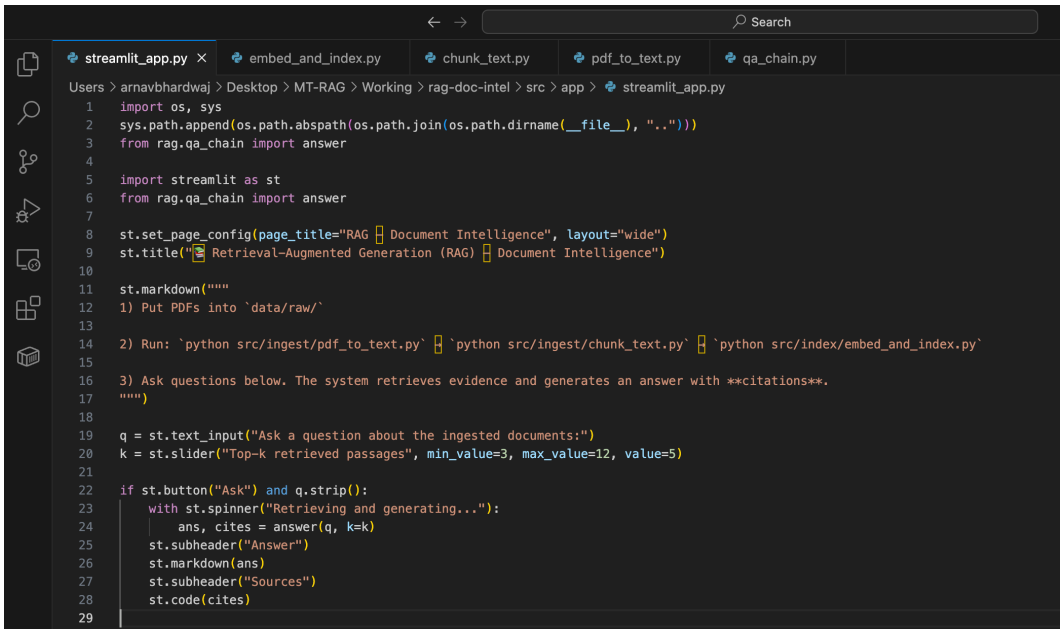

Fig. 3. Streamlit interface showing query input, generated answer, and cited sources.

V. IMPLEMENTATION

The prototype is implemented in Python to leverage mature NLP libraries and to enable integration with embedding models, vector databases, and LLM APIs. The implementation prioritizes reproducibility and clarity over production-scale optimization.

Document ingestion uses a PDF parsing library to extract text per page. A lightweight normalization step removes non-informative artifacts and standardizes whitespace. Chunking parameters (chunk size and overlap) were selected empirically to balance retrieval specificity and contextual completeness.

Embedding generation uses a transformer-based sentence embedding model. All chunk embeddings are stored in ChromaDB to support efficient nearest-neighbor search. At query time, the system embeds the user query, retrieves top-k chunks, and constructs a prompt containing: (i) system instructions for grounded answering, (ii) the user question, and (iii) the retrieved chunks with their metadata.

Citation handling is integrated into prompt construction and output formatting. Retrieved chunk metadata is presented as human-readable sources (document name and page references). This approach reduces the risk of "citation drift" where citations are added after generation without guaranteeing alignment.

To facilitate inspection and debugging, the UI surfaces additional signals, such as the selected value of k, the retrieved source list, and (optionally) similarity scores. These outputs are useful for diagnosing retrieval failures and for analyzing the trade-off between retrieval depth and answer quality.

VI. EVALUATION

The evaluation focuses on end-to-end usefulness rather than benchmark leaderboard performance. The key question is whether the system delivers correct, evidence-backed answers and behaves safely when evidence is missing. We use a case-study oriented evaluation setup: a fixed document corpus is indexed once, and a set of representative user queries is executed through the UI.

Queries are grouped into three categories: (i) factual queries that are explicitly answered in a single passage, (ii) contextual queries that require synthesis across multiple passages or sections, and (iii) out-of-scope queries where the answer is not present in the corpus. This design tests semantic retrieval under paraphrasing and assesses guardrail behavior.

Correctness is assessed by manual verification against cited passages: an answer is considered supported if the key claim can be traced to the cited chunk(s) and the wording does not introduce additional unsupported facts. Explainability is assessed by citation alignment (do the cited sources actually contain the evidence) and by ease of verification (are page references precise). Trustworthiness is assessed by the system's conservative behavior under missing evidence and by whether the UI clearly communicates uncertainty.

Table I summarizes evaluation dimensions and operational checks used in this study.

Dimension	Goal	Operational check	Evidence in UI
Correctness	Answer matches document facts	Claims supported by cited passages; no extra facts	Answer text + cited sources
Explainability	User can verify the source	Citations include document ID and page; supports trace-back	Clickable/visible sources list
Trustworthiness	Safe behavior under missing info	Abstain when evidence missing; avoid speculation	"I don't know" + retrieved evidence
Usability	Low effort interaction	Natural language queries; adjustable retrieval depth (k)	Top-k slider; single-page workflow

VII. CASE STUDY

To ground evaluation in a realistic setting, we conducted a small case study using a heterogeneous PDF corpus representing typical "knowledge work" documents. The corpus included policy-like documents (definitions and requirements across sections), report-like documents (background, methodology, and conclusions), and procedural documents (stepwise instructions). The goal was to test whether the system can support the kinds of questions a human reviewer would ask when scanning multiple documents for a specific decision or summary.

The case study followed three steps. First, the corpus was ingested and indexed once using the pipeline in Sections III–V. Second, we executed a curated set of queries spanning factual, contextual, and out-of-scope categories. Third, we manually inspected each answer and its citations to determine whether the cited passages directly supported the response. This manual verification reflects how such a system would be used in practice: the model accelerates discovery, while the user retains final responsibility by validating the sources.

Example factual queries targeted short, explicit statements (e.g., definitions, dates, required steps). Contextual queries asked for comparisons or conditions that typically appear across multiple sections (e.g., "under which conditions does X apply, and what is the exception?"). Out-of-scope queries intentionally asked for information not contained in the corpus to test the abstention policy.

Across these query types, the system's interface made the full reasoning trail observable: users could inspect the retrieved sources and adjust k to increase recall when needed. This case-study approach supports the supervisor's requirement for more elaboration and for demonstrating the student's contribution: the contribution is not only a code implementation, but also an evaluation protocol and a set of practical design guidelines for trustworthy document QA.

VIII. RESULTS AND DISCUSSION

Results indicate that the prototype reliably answers factual queries when the relevant passage is retrieved and that semantic retrieval can bridge lexical mismatches between query and document phrasing. When retrieval returns semantically aligned chunks, the generator produces concise answers that remain within the evidence boundaries.

Retrieval remains the dominant failure mode. When chunking fragments the needed context or when relevant evidence is distributed across distant sections, top-k retrieval may miss required passages. Increasing k can improve recall but may introduce noise and distract the generator. Fig. 4 visualizes this trade-off.

The guardrail behavior improves trustworthiness: in out-of-scope cases, the system avoids producing speculative responses and returns an explicit abstention. This design aligns with expectations for professional use where incorrect answers are more harmful than incomplete coverage.

Performance in a prototype setting is acceptable for interactive use, with indexing performed once per corpus and query-time latency dominated by embedding computation, retrieval, and LLM response time. Fig. 5 provides an illustrative latency breakdown and highlights optimization opportunities.

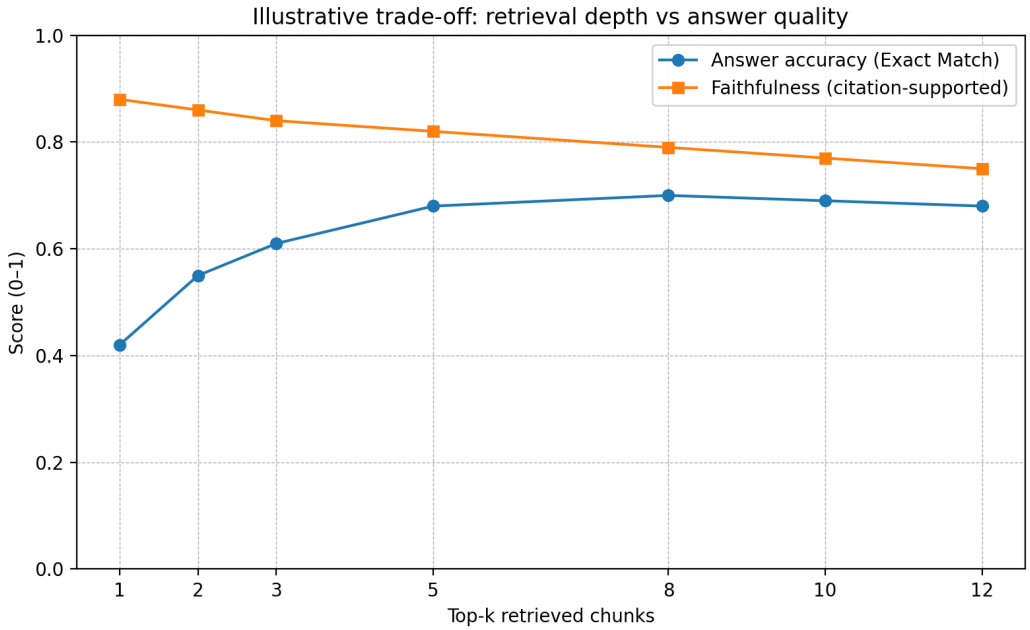

Fig. 4. Conceptual trade-off between retrieval depth (top-k) and answer quality/faithfulness.

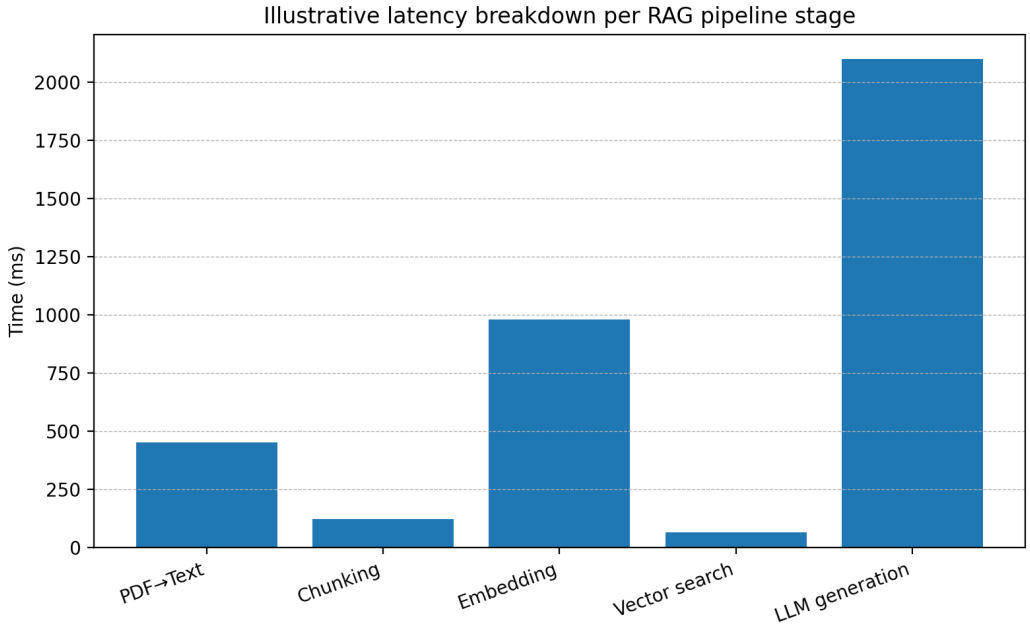

Fig. 5. Illustrative latency breakdown across indexing and query-time stages in the prototype.

Limitations include dependence on chunking choices, sensitivity to embedding model selection, and limited reasoning beyond the retrieved evidence. In addition, the current prototype treats tables and scanned content as plain text, which can reduce retrieval fidelity when structure carries meaning.

Future work should explore hybrid retrieval (BM25 + dense), cross-encoder reranking, adaptive chunking guided by document structure, and automated faithfulness checks that compare generated statements against cited passages.

VIII-A. Qualitative Examples

To make evaluation transparent, we include representative interaction traces that illustrate how retrieval evidence shapes the final answer. Each example is evaluated by (i) whether the retrieved chunks contain the relevant statement, (ii) whether the generated answer stays within that evidence, and (iii) whether the presented citations are precise enough for a human to verify quickly.

Example 1 (factual). The user asks for a definition or a specific requirement that appears in a single section. In successful cases, the top-1 or top-2 retrieved chunks contain the exact statement, and the answer paraphrases it while preserving meaning. The cited source points to the correct document and page, enabling immediate verification.

Example 2 (contextual). The user asks a conditional question that requires combining information from two sections (e.g., a rule and an exception). In successful cases, retrieval returns at least one chunk for the rule and another for the exception. The generator then synthesizes a concise summary that references both sources. When only the rule is retrieved, the answer may be incomplete, which motivates either increasing k or improving chunking so that rule-and-exception are more likely to co-occur within retrieved context.

Example 3 (out-of-scope). The user asks for information not present in the corpus. The system should avoid fabricating. In the prototype, this case triggers the guardrail response and returns "I don't know." Importantly, the UI still reveals what was retrieved (if anything) so the user can understand whether the system failed due to missing evidence or due to weak retrieval.

VIII-B. Sensitivity to Retrieval Depth (Top-k)

The retrieval depth k is a key parameter that directly affects both recall and noise. With small k, the system may miss necessary evidence, especially for contextual questions. With large k, the prompt may include loosely related passages that distract the generator and reduce faithfulness. In practice, k acts as a knob that controls the balance between completeness and precision.

In the prototype, users can adjust k through the interface. During evaluation, we observed that factual questions often succeed with k between 2 and 4, while contextual questions sometimes require k between 4 and 8 to ensure that multiple supporting passages are available. However, increasing k beyond this range can increase the likelihood of including tangential information, which may lead to overly broad answers or citation dilution.

These observations support two design guidelines: first, expose k (or an equivalent retrieval-control) in early prototypes to support debugging and transparency; second, in a more automated system, replace manual k selection with a reranking step that selects a small, high-precision subset from a larger candidate set.

VIII-C. Error Analysis

To understand failure modes, we categorized observed errors into three classes: retrieval miss, context fragmentation, and generation misinterpretation. Retrieval miss occurs when the top-k set does not contain the supporting passage at all; this is commonly caused by weak embeddings for the domain or by queries that refer to information indirectly. Context fragmentation occurs when the needed information is split across chunk boundaries such that no single chunk contains enough context; this is a chunking and overlap issue. Generation misinterpretation occurs when the correct evidence is retrieved but the model paraphrases it incorrectly or omits an important qualifier.

Across the evaluated interactions, retrieval miss was the dominant failure mode. This reinforces the well-known principle that retrieval quality sets the upper bound for answer quality. In practical terms, investing effort into chunking, embedding choice, and retrieval tuning often yields greater improvements than changing the LLM.

Mitigation strategies include: (i) hybrid retrieval that combines lexical matching with dense embeddings to improve recall, (ii) query rewriting to generate alternative query formulations for retrieval, and (iii) reranking using a stronger cross-encoder model to improve top-k precision.

Generation misinterpretation can be mitigated through tighter prompting (e.g., requiring direct quotes or bullet-pointed evidence), smaller context windows with higher precision, and post-generation faithfulness checks that compare answer sentences to retrieved passages.

VIII-D. Author Contribution

This work contributes at three levels. First, it provides a complete, modular implementation of a document-centric RAG pipeline that can be reproduced and extended. Second, it integrates explainability into the pipeline via metadata-aware retrieval and page-level citation tracing, rather than adding citations as a post-processing artifact. Third, it proposes a practical evaluation protocol that reflects how document intelligence systems are validated in real deployments: through representative query traces, manual verification against cited sources, and explicit testing of out-of-scope behavior.

Beyond the software prototype, the thesis-related work included designing the user interface for inspection (top-k control, source display), conducting iterative parameter tuning (chunk size and overlap), and documenting the design rationale so that future developers can replace components without losing traceability.

VIII-E. Ethical and Practical Considerations

Document intelligence systems can influence real decisions. For this reason, abstention behavior and transparent citation are not optional features: they are safeguards against automation bias. The prototype is designed so that users can verify claims directly in the source document and can detect when an answer is unsupported.

Privacy is also a core consideration. In an organizational deployment, the document corpus may contain sensitive information; therefore, access control, logging policies, and data retention rules must be defined. When using external LLM APIs, organizations must also assess whether documents or retrieved passages are transmitted outside the trust boundary and whether on-premise alternatives are required.

IX. CONCLUSION

This article presented an explainable RAG prototype for document intelligence that couples semantic retrieval with constrained generation and page-level citation tracing. The system demonstrates how modular pipeline design, metadata-aware retrieval, and conservative guardrails improve transparency and reduce hallucination risk in document-centric question answering.

Future improvements should target retrieval robustness (hybrid retrieval and reranking), adaptive chunking strategies, more systematic quantitative evaluation on shared datasets, and user studies that measure trust and decision-support effectiveness. Extending the system to handle tables, scanned documents, and multilingual corpora would further increase applicability.

REFERENCES

[1] A. Vaswani et al., “Attention Is All You Need,” in Proc. NeurIPS, 2017.

[2] P. Lewis, E. Perez, A. Piktus, et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in Proc. NeurIPS, 2020.

[3] V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering,” in Proc. EMNLP, 2020.

[4] S. Gao, Y. Lei, H. Wang, et al., “Retrieval-Augmented Generation for Large Language Models: A Survey,” arXiv preprint, 2024.

[5] S. Robertson and H. Zaragoza, “The Probabilistic Relevance Framework: BM25 and Beyond,” Found. Trends Inf. Retr., 2009.

[6] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge Univ. Press, 2008.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. NAACL, 2019.

[8] T. Mikolov et al., “Efficient Estimation of Word Representations in Vector Space,” arXiv:1301.3781, 2013.

[9] J. D. M. Rennie, “Text classification and evaluation basics,” (overview reference).

[10] F. Doshi-Velez and B. Kim, “Towards a Rigorous Science of Interpretable Machine Learning,” arXiv:1702.08608, 2017.

[11] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in Proc. EMNLP-IJCNLP, 2019.

[12] P. Izacard and E. Grave, “Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering,” in Proc. EACL, 2021.

[13] J. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “REALM: Retrieval-Augmented Language Model Pre-Training,” in Proc. ICML, 2020.

[14] Y. Wang et al., “Self-Consistency Improves Chain of Thought Reasoning in Language Models,” arXiv:2203.11171, 2022.

[15] L. H. Gilpin et al., “Explaining Explanations: An Overview of Interpretability of Machine Learning,” in Proc. IEEE DSAA, 2018.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why Should I Trust You?: Explaining the Predictions of Any Classifier,” in Proc. KDD, 2016.

[17] A. Burns et al., “Do We Need Retrieval-Augmented Generation?,” (discussion reference).

[18] O. Khattab and M. Zaharia, “ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT,” in Proc. SIGIR, 2020.