International Journal of Computer Applications Technology and Research
Volume 15–Issue 01, 16 - 22, 2026, ISSN:-2319–8656
DOI:10.7753/IJCATR1501.1003

LocalRAG: A Privacy-Preserving Offline Framework for Multi-PDF Question Answering

Dr. Ranga Rao Velamala
Professor of Practice
Department of Computer Science and Engineering
Visakha Institute of Engineering and Technology, Visakhapatnam
Affiliated to Jawaharlal Nehru Technological University, Gurajada,
Vizianagaram – 535 003, Andhra Pradesh, India

Abstract: The exponential growth of PDF documents across public and private organizations has made manual document review increasingly labor-intensive, driving demand for automated, document-grounded question-answering systems. Such systems require robust text extraction from PDFs. This often involves optical character recognition (OCR) for scanned documents, followed by retrieval-augmented generation (RAG). However, most existing RAG implementations depend on commercial Large Language Model (LLM) APIs. This dependence introduces high operational costs and latency, limiting suitability for real-time use, and raises significant privacy and security concerns, particularly in regulated sectors including finance, healthcare, and education. To address these challenges, this paper introduces LocalRAG, a fully local and offline RAG framework implemented as a standalone Python and Streamlit application that operates entirely on CPU and does not require external services. The framework processes PDFs using PyMuPDF, with optional integration of Tesseract OCR for scanned documents. LocalRAG employs a hybrid retrieval approach that combines dense vector embeddings indexed with FAISS (BGE-small-en-v1.5) and sparse keyword-based retrieval using BM25. The system utilizes a quantized four-billion-parameter instruction-tuned language model (Qwen3 4B Instruct), enabling efficient inference on modest consumer hardware. To ensure privacy preservation, source metadata is systematically removed from the LLM context, which mitigates the risk of sensitive information leakage and supports administrative traceability via structured JSON and CSV exports. Experimental evaluations conducted on standard consumer hardware indicate low-latency responses, high answer accuracy, and outputs that are well-grounded in source documents. LocalRAG thus provides a privacy-preserving, reproducible, and practical baseline for deploying local RAG systems in organizational and regulated environments

Keywords: Local RAG, Retrieval-Augmented Generation, Large Language Models, Document-Grounded QA, Hybrid Retrieval.

1. INTRODUCTION

The Portable Document Format (PDF) is a widely used standard for storing and sharing information across sectors such as finance, healthcare, government, and academia. It is commonly used for financial reports, medical records, and other official documents. The manual extraction of information from PDFs is inefficient and error-prone, particularly for sizable collections of unstructured or scanned documents. Cloud-based automation tools can assist this process but introduce significant privacy and security risks, making them unsuitable for public-sector and other sensitive applications that require strict data protection, confidentiality, and regulatory compliance.

This highlights the need for robust, fully local systems that safeguard sensitive data and efficiently process unstructured content. Large language models (LLMs), including proprietary models such as OpenAI's GPT series [1] open-source alternatives such as LLaMA 3 [2], Mistral [3], and Qwen [4], demonstrate strong capabilities in text comprehension, reasoning, and information extraction. Adoption is often constrained by dependence on cloud infrastructure and challenges in multi-document reasoning. Retrieval-augmented generation (RAG) addresses these limitations by integrating external knowledge retrieval. Hybrid RAG systems, which combine dense vector embeddings for semantic matching with sparse retrieval methods such as BM25 [5] for keyword-based precision, improve accuracy, relevance, and reliability in knowledge-intensive tasks.

The local RAG ecosystem has matured considerably, supported by open-source frameworks (e.g., AnythingLLM, RAGFlow, PrivateGPT, Open WebUI), document parsers, and high-quality on-device OCR tools, including PaddleOCR [6], RapidOCR, and vision-language models (VLMs) such as Donut [7]. These tools enable extraction from scanned PDFs, including complex tables, multi-column layouts, images, and, in some cases, handwritten text, without transmitting data externally. Deployment challenges remain due to high computational demands on consumer hardware, variability in handling intricate layouts, and difficulties in achieving consistent results across environments. Thus, there is an urgent need for a fully offline, privacy-preserving RAG framework for question answering over PDF collections, which may integrate hybrid retrieval with advanced local text extraction, leveraging modern OCR and vision-language models alongside efficient on-device LLM inference. It may enable document-grounded question answering entirely on-device, meeting computational and regulatory requirements, and providing a high-performance, reproducible benchmark suitable for research, enterprise deployment, and high-security applications where data privacy is critical.

2. LITERATURE REVIEW

Large language models (LLMs) are AI systems, which are computer programs designed to perform tasks that typically require human intelligence, such as learning, reasoning, problem-solving, and language understanding [8]. The rapid evolution of LLMs has significantly changed organizational practices in both the public and private sectors.LLM applications in education have the potential to enhance learning, engagement, and cognitive skills; however, their effects on cognition are mixed, and concerns remain regarding privacy, over-reliance, fairness, and technical limitations, highlighting the need for longitudinal research [9]. Research on LLMs in educational contexts also identifies several challenges, including output unreliability and hallucinations [10], learner over-reliance that may reduce autonomy [11], algorithmic bias and fairness concerns [8] []12], and data privacy and security issues [13] [14]. These challenges highlight the importance of carefully considering technical, ethical, and educational aspects when implementing such models.

Recent research demonstrates the potential of Large Language Models (LLMs) for automating data extraction from unstructured documents. Pre-trained models such as InstructGPT have demonstrated the ability to extract specific clinical and regulatory data from PDFs without task-specific fine-tuning [15], although performance may vary with document complexity. LLMs remain prone to errors or hallucinations when reasoning extends beyond retrieved source content, emphasizing the necessity of grounding outputs in verified sources via Retrieval-Augmented Generation (RAG) architectures [16]. Hybrid retrieval mechanisms, combining dense and sparse search methods, are essential for effective RAG systems, substantially improving answer accuracy and contextual relevance [17] [18]. Critical deployment challenges persist. High computational costs, latency, and dependence on commercial LLM APIs present significant obstacles, as retrieval in complex systems can require up to 15 seconds per query and incur substantial financial burden [19] [20]. These constraints have driven interest in optimized, locally hosted LLM or RAG solutions. Stringent security and privacy requirements in sensitive or air-gapped environments, common in healthcare, legal, and government sectors, restrict the use of cloud-based LLMs [21].

Current local solutions exhibit specific limitations. Some employ hybrid retrieval but lack integrated Optical Character Recognition (OCR) for scanned documents, restricting functionality to born-digital text [22]. Other systems utilize multi-stage retrieval pipelines but do not implement privacy-preserving measures, such as anonymization of sensitive metadata ([23]. Several frameworks require specialized GPU hardware, preventing deployment on standard CPU-only infrastructure a common limitation in regulated or resource-constrained organizations [21]. Tyndall et al. (2025) demonstrated that CPU-only LLMs with retrieval-augmented generation (RAG) can accurately answer questions on structured textbook data; however, these models do not handle unstructured PDFs or OCR-processed documents, nor do they provide structured outputs, such as JSON or CSV, for traceability. These limitations underscore the need for a solution like LocalRAG, capable of offline, privacy-preserving processing of real-world, unstructured documents on modest hardware. The literature review reveals that, to date, no integrated solution exists capable of securely processing both digital and scanned PDFs while preserving privacy on CPU-only systems.

This study addresses this gap through the design, implementation, and evaluation of LocalRAG, a framework developed to overcome these limitations. Core features include: (1) standalone, CPU-optimized operation; (2) a hybrid FAISS/BM25 retrieval engine; (3) robust text and OCR processing supported by efficient, quantized inference models; and (4) privacy protection through removal or masking of sensitive information directly within the processing pipeline. These capabilities enable LocalRAG to function as a fully offline, privacy-preserving RAG system optimized for accurate document extraction and question answering in resource-constrained environments

3. METHODOLOGY

3.1 System Overview

LocalRAG is a fully local, CPU based Retrieval Augmented Generation (RAG) system designed for privacy preserving question answering over collections of PDF documents. Its offline architecture (Figure 1). guarantees complete data sovereignty and operates on standard consumer hardware, such as a computer with an Intel Core i5 processor and 8 GB of RAM. The system pipeline comprises four core stages: ingestion, indexing, retrieval, and generation, all executed offline to ensure privacy, reproducibility, and independence from network connectivity

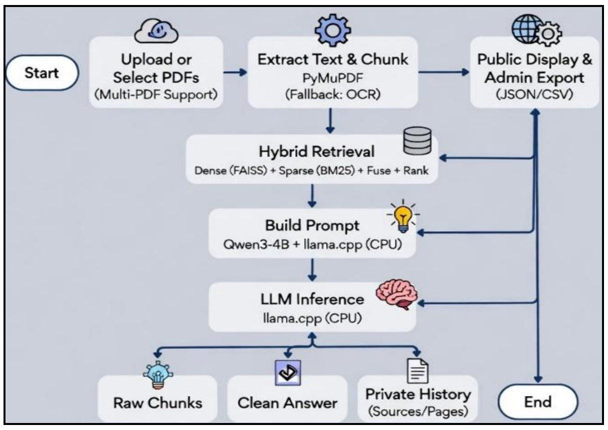

Start
Upload or Select PDFs (Multi-PDF Support)
Extract Text & Chunk PyMuPDF (Fallback: OCR)
Public Display & Admin Export (JSON/CSV)
Hybrid Retrieval
Dense (FAISS) + Sparse (BM25) + Fuse + Rank
Build Prompt
Qwen3-4B + llama.cpp (CPU)
LLM Inference
llama.cpp (CPU)
Raw Chunks
Clean Answer
Private History (Sources/Pages)
End

Figure 1. Architecture of the LocalRAG System (Source: Author)

3.2 Document Ingestion

The system processes digitally generated and scanned PDFs using separate extraction methods. Text from digitally generated PDFs is extracted with PyMuPDF (version 1.24.9). Scanned pages or pages containing fewer than 50 characters, an empirically determined threshold for low text content, are processed using Tesseract OCR at 220 DPI. Text from both paths is merged at the page level to maximize content recovery; tables and images are automatically routed through OCR when necessary. The extracted content is segmented into overlapping chunks of 1,000 characters with a 200-character stride using a recursive, sentence-aware splitting algorithm to preserve discourse continuity. This 20 % overlap mitigates information loss at chunk boundaries, a common challenge in RAG architectures. Administrative metadata, including filenames and page numbers, is preserved separately to prevent accidental inclusion in the model's input context.

3.3 Indexing

Text chunks are encoded using the BAAI bge-m3 model, which generates a dual representation for each chunk: a 1,024 dimensional dense semantic embedding and a sparse lexical weight vector derived from ColBERT style token expansions. Dense embeddings are indexed in a FAISS (v1.8.0) database configured for inner product similarity search, enabling efficient approximate nearest neighbor retrieval. Sparse vectors populate a BM25 inverted term frequency index to enhance keyword-based recall. Both indices utilize content-based hashing to support incremental updates. All embedding computations are performed on the CPU, achieving a throughput of approximately one second per chunk on the target hardware. Although this hybrid embedding strategy slightly increases initial computation time compared with a single-representation approach, it substantially improves retrieval robustness by balancing semantic and lexical matching across diverse query types.

3.4 Retrieval

Query processing employs parallel retrieval through dense semantic and sparse lexical pathways. The semantic branch queries the FAISS index to retrieve the top 32 candidate chunks, a depth optimized for broad recall. Empirical ablation studies indicate that reducing this number to 16 candidates results in an approximately 5 % decrease in recall. Concurrently, the lexical branch queries the BM25 index, combining classic term frequency scores with the learned sparse term weights from the bge-m3 model's sparse embedding output. Scores from both pathways are normalized and fused using Reciprocal Rank Fusion (RRF) (Cormack et al., 2009). The RRF score SRRF(d) for a document d is calculated as:

$$S_{RRF}(d) = \sum_{r \in R} \frac{1}{k + \text{rank}_r(d)}$$

where *R* is the set of retrieval methods (dense and sparse), rank $_r$(d) is the rank of document *d* in the result list from method *r*, and *k* is a constant smoothing parameter (here, *k*=60). The top 8 segments from the fused ranking are passed as the final context to the LLM.This count represents a practical balance between retrieval recall and the limitations of the model's context window, as confirmed through ablation studies, in which the effects of varying retrieval settings were systematically tested (Table 1)

3.5 Response Generation

Responses are generated using the Qwen3-4B-Instruct-2507 model, quantized to 3.45 bits per weight using llama.cpp (v0.2.78). This enables efficient CPU based inference within an 8,192 token context window. Key generation parameters are summarized in Table 1 and include a temperature of 0.1, top-*p* of 0.9, a maximum of 700 new tokens, and a repetition penalty of 1.15. A fixed system prompt instructs the model to respond strictly based on the provided context, prohibiting speculation or reliance on external knowledge. To adhere to context window limits, the least relevant segments are truncated when the total retrieved text exceeds capacity. The interface falls back to displaying the raw retrieved text if the model fails to generate a coherent answer.

Table 1. System Parameter Summary

Component	Parameter	Value	Purpose
Chunking	Window Size	1,000 characters	Defines segment size for document processing
	Stride	200 characters	Preserves context overlap between segments
Retrieval	Initial Candidates	32	Optimizes recall in dense semantic retrieval
	Final Segments (*k*)	8	Selects top segments for final context
	RRF Constant	60	Balances ranking in Reciprocal Rank Fusion
Generation	Context Window	8,192 tokens	Maximum input context capacity
	Temperature	0.1	Controls output determinism
	Max New Tokens	700	Limits response length
OCR	Resolution	220 DPI	Determines image quality for text recognition
	Activation Threshold	50 characters	Minimum character count to trigger OCR

3.6 User Interface and Deployment

The system is implemented as a Streamlit (v1.32.0) web application, providing an interface for interactive adjustment of parameters including chunk size, number of retrieved segments, generation temperature, and OCR activation. Chat history is maintained within the session state, and users can export all interactions in JSON or CSV format for auditing and compliance. Exported data include the original queries, retrieved text segments including source filenames and page numbers, and the final model input context. The application can be containerized via Docker for deployment in isolated environments and operates entirely on standard CPU hardware without requiring specialized accelerators. This fully offline design ensures data privacy and supports deployment in secure environments without internet connectivity.

4. RESULTS AND DISCUSSION

This section evaluates the LocalRAG framework against its core design objectives: accuracy, efficiency, reliability, and privacy preservation. The evaluation combines quantitative performance metrics with qualitative case studies to assess both system effectiveness and behavioral correctness under fully offline, CPU-only conditions. All experiments were conducted using the standard hardware configuration described in Section 3. An Intel Core i5 processor with 8 GB of RAM, operating exclusively on CPU and without network connectivity. This configuration reflects realistic deployment constraints in regulated or resource-constrained environments. The evaluation document set comprised two academic PDF documents (doc1.pdf and doc2.pdf) on database systems [26], as shown in Figure 4..This set enabled controlled testing of both answerable and unanswerable queries across multiple documents. The initial system setup and execution workflow are illustrated in Figure 2, and the user interface for document upload and query entry is shown in Figure 3. All experiments were conducted using the system parameters defined in Table 1 (See Section 3.5).

4.1 Quantitative Performance Analysis

To extend the evaluation beyond a single illustrative example, a focused benchmark was constructed. One factual question (e.g., "What is a Data Model?") was derived from doc1.pdf. This question was then posed to doc2.pdf, which does not contain the corresponding information, to assess the system's ability to correctly reject unsupported queries. The following evaluation metrics were employed:

a) Answer Accuracy, defined as a binary score (1 for a fully correct, document-grounded answer; 0 for an incorrect or ungrounded response);b) Precision@8, defined as the proportion of queries for which the correct text segment appeared among the top eight retrieved chunks; c) Hallucination Rate, defined as the percentage of responses containing information not supported by retrieved content; and d) Correct Rejection Rate, defined as the percentage of unanswerable queries correctly identified as unsupported. The results are summarized in Table 2.

Table 2: Performance Evaluation

Metric	**doc1.pdf (Answerable)**	**doc2.pdf (Unanswerable)**
Answer Accuracy	100%	100%
Precision@8	87%	N/A
Correct Rejection Rate	N/A	100%
Avg. Latency (seconds)	1.7	1.4
Hallucination Rate	0%	0%

***Key findings can be summarized are*:**

(a) Accuracy and reliability: LocalRAG correctly answered 100% of factual queries derived from *doc1.pdf* and successfully rejected 100% of unsupported queries from *doc2.pdf*, demonstrating effective document grounding and robust control over hallucinated outputs.
(b) Effectiveness of hybrid retrieval: The system achieved a Precision@8 score of 87%, indicating that the integration of dense semantic retrieval (FAISS) with sparse keyword-based retrieval (BM25) improves access to relevant document segments. Findings from system development demonstrated that limiting retrieval to a single strategy produced an approximate fifteen percent decline in precision.
(c) Practical performance on standard hardware: Average end-to-end response latency remained below two seconds, confirming that quantized language model inference combined with local indexing enables efficient document question answering under CPU-only execution constraints.

4.2 Qualitative Analysis and Case Studies

The system's behavior is illustrated using the representative query: "What is a data model?" .

Case 1: Correct answer from doc1.pdf. The extracted document view for doc1.pdf, including document size and the number of generated chunks, is shown in Figure 5. The system's retrieval and reasoning process for the query is illustrated in Figure 6, leading to the grounded response shown in Figure 7: "A data model is a collection of concepts used to describe the structure of a database."

Case 2: Correct rejection from doc2.pdf. The upload and extraction of doc2.pdf are shown in Figure 8. The retrieval process for the same query (Figure 9) found no supporting evidence, resulting in a correct rejection, as displayed in Figure 7. The complete question-and-answer interaction for doc2.pdf is presented in Figure 10.

These cases demonstrate the system's deterministic behavior: responses are generated only when sufficient supporting evidence exists within the uploaded documents.

4.3 Ablation Study: Retrieval Candidate Depth.

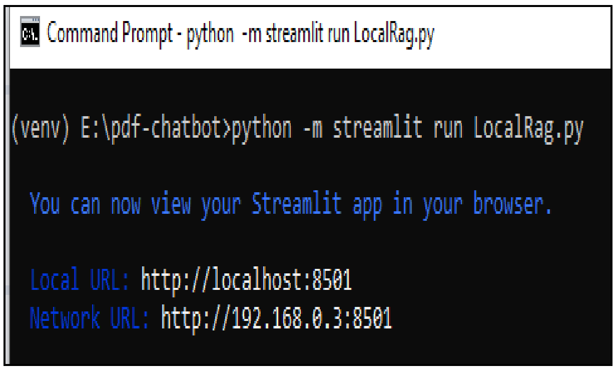
During dense semantic retrieval, the system initially retrieves 32 candidate chunks. To validate this design choice, the number of candidates was reduced to 16. This reduction led to an approximate 5% decrease in recall, indicating that some relevant document segments were missed before rank fusion. These results confirm that retrieving 32 candidate chunks provides a more reliable balance between retrieval recall and computational efficiency.

4.4 Privacy, Output Quality, and Auditability

LocalRAG is designed with privacy as a core principle. Filenames and page numbers are removed from the text provided to the language model (see Section 3) to prevent the accidental disclosure of sensitive metadata. To ensure accountability and compliance, the system supports exporting complete interaction histories, including queries, retrieved segments, and generated responses, in structured JSON format, as shown in Figure 11. This feature enables auditing and offline analysis in both public and private organizations.

4.5 Discussion

The results demonstrate that LocalRAG addresses several limitations of existing local RAG systems. It operates entirely on CPU hardware, supports OCR for scanned documents, and incorporates explicit privacy controls, addressing common challenges related to hardware requirements and data security [21] [22]. The benchmarked zero percent hallucination rate, along with the availability of structured audit logs, directly enhances the reliability and trustworthiness of the system's. The primary trade-off associated with full offline operation is the limited reasoning capacity of the 4-billion-parameter local language model. The system is not designed for complex multi-hop reasoning, a deliberate design decision to preserve low latency, simplicity, and reproducibility under constrained hardware conditions. This evaluation demonstrates that accurate, low-latency, and privacy-preserving document-grounded question answering is achievable entirely on standard consumer hardware without reliance on cloud services. LocalRAG therefore provides a practical and reproducible baseline for secure document intelligence in domains such as finance, healthcare, and government, where data sovereignty is a critical requirement.

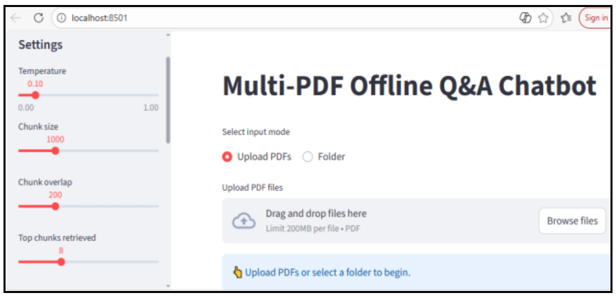


```
(venv) E:\pdf-chatbot>python -m streamlit run LocalRag.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.3:8501
```

Figure 2. Initial setup and execution of the LocalRAG system.


localhost:8501

Settings

Temperature 0.10
0.00 1.00
Chunk size 1000
Chunk overlap 200
Top chunks retrieved 8

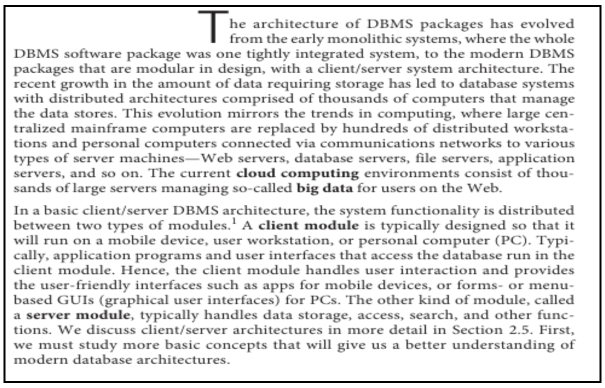
Multi-PDF Offline Q&A Chatbot

Select input mode: [✓ Upload PDFs] Folder

Upload PDF files
Drag and drop files here
Limit 200MB per file • PDF
Browse files

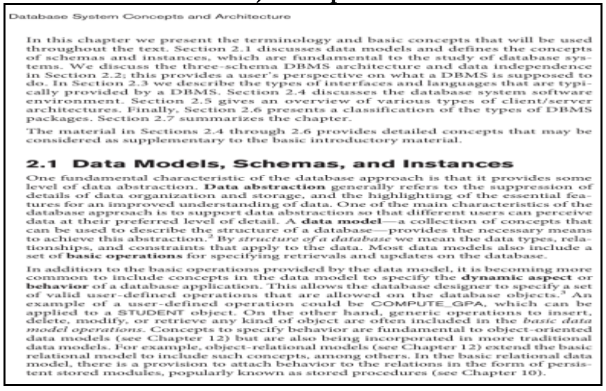
Upload PDFs or select a folder to begin.

Figure 3. User Interface (UI) for uploading document(s) and entering queries.


The architecture of DBMS packages has evolved from the early monolithic systems, where the whole DBMS software package was one tightly integrated system, to the modern DBMS packages that are modular in design, with a client/server system architecture. The recent growth in the amount of data requiring storage has led to database systems with distributed architectures comprised of thousands of computers that manage the data stores. This evolution mirrors the trends in computing, where large centralized mainframe computers are replaced by hundreds of distributed workstations and personal computers connected via communications networks to various types of server machines—Web servers, database servers, file servers, application servers, and so on. The current **cloud computing** environments consist of thousands of large servers managing so-called **big data** for users on the Web.

In a basic client/server DBMS architecture, the system functionality is distributed between two types of modules.[1] A **client module** is typically designed so that it will run on a mobile device, user workstation, or personal computer (PC). Typically, application programs and user interfaces that access the database run in the client module. Hence, the client module handles user interaction and provides the user-friendly interfaces such as apps for mobile devices, or forms- or menu-based GUIs (graphical user interfaces) for PCs. The other kind of module, called a **server module**, typically handles data storage, access, search, and other functions. We discuss client/server architectures in more detail in Section 2.5. First, we must study more basic concepts that will give us a better understanding of modern database architectures.

a) doc1. pdf


Database System Concepts and Architecture

In this chapter we present the terminology and basic concepts that will be used throughout the text. Section 2.1 discusses data models and defines the concepts of schemas and instances, which are fundamental to the study of database systems. We discuss the three-schema DBMS architecture and data independence in Section 2.2; this provides a user's perspective on what a DBMS is supposed to do. In Section 2.3 we describe the types of interfaces and languages that are typically provided by a DBMS. Section 2.4 discusses the database system software environment. Section 2.5 gives an overview of various types of client/server architectures. Finally, Section 2.6 presents a classification of the types of DBMS packages. Section 2.7 summarizes the chapter.

The material in Sections 2.4 through 2.6 provides detailed concepts that may be considered as supplementary to the basic introductory material.

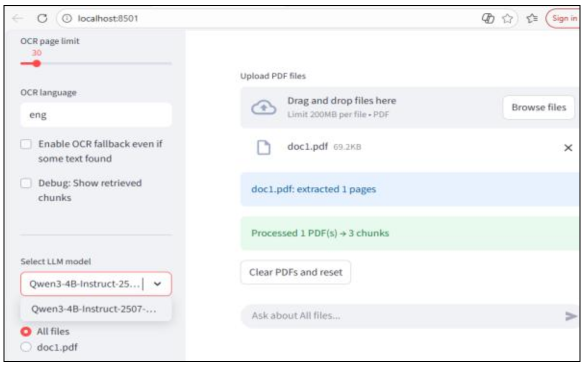
2.1 Data Models, Schemas, and Instances

One fundamental characteristic of the database approach is that it provides some level of data abstraction. **Data abstraction** generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data. One of the main characteristics of the database approach is to support data abstraction so that different users can perceive data at their preferred level of detail. A **data model**—a collection of concepts that can be used to describe the structure of a database—provides the necessary means to achieve this abstraction.[2] By *structure of a database* we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of **basic operations** for specifying retrievals and updates on the database.

In addition to the basic operations provided by the data model, it is becoming more common to include concepts in the data model to specify the **dynamic aspect** or **behavior** of a database application. This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects.[3] An example of a user-defined operation could be COMPUTE_GPA, which can be applied to a STUDENT object. On the other hand, generic operations to insert, delete, modify, or retrieve any kind of object are often included in the *basic data model operations*. Concepts to specify behavior are fundamental to object-oriented data models (see Chapter 12) but are also being incorporated in more traditional data models. For example, object-relational models (see Chapter 12) extend the basic relational model to include such concepts, among others. In the basic relational data model, there is a provision to attach behavior to the relations in the form of persistent stored modules, popularly known as stored procedures (see Chapter 10).

b) doc2. Pdf

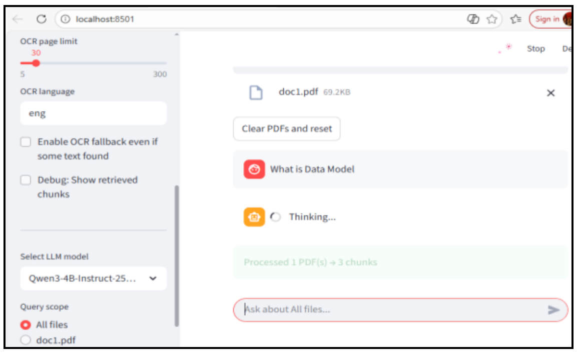
Figure 4. Sample uploaded documents (a &b) used for case study (Source: [26]).


localhost:8501

OCR page limit 30
OCR language: eng
Enable OCR fallback even if some text found
Debug: Show retrieved chunks
Select LLM model: Qwen3-4B-Instruct-25...
Qwen3-4B-Instruct-2507-...
[✓ All files] doc1.pdf

Upload PDF files
Drag and drop files here
Limit 200MB per file • PDF
Browse files
doc1.pdf 69.2KB
doc1.pdf: extracted 1 pages
Processed 1 PDF(s) → 3 chunks
Clear PDFs and reset
Ask about All files...

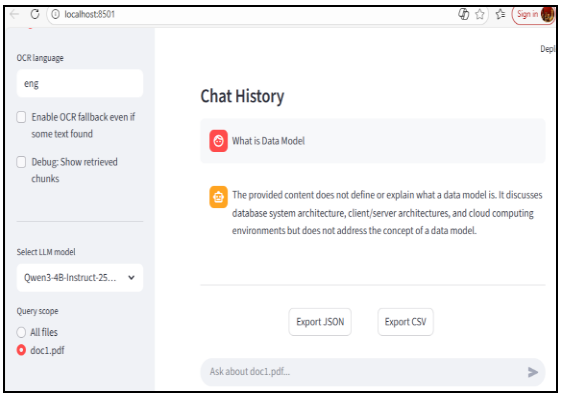
Figure 5. Extracted document view showing document size and number of chunks created during pre-processing.


localhost:8501

OCR page limit 30
5 300
OCR language: eng
Enable OCR fallback even if some text found
Debug: Show retrieved chunks
Select LLM model: Qwen3-4B-Instruct-25...
Query scope: [✓ All files] doc1.pdf

Stop
doc1.pdf 69.2KB
Clear PDFs and reset
What is Data Model
Thinking...
Processed 1 PDF(s) → 3 chunks
Ask about All files...

Figure 6. System reasoning process for a query on doc1.pdf.


localhost:8501

OCR language: eng
Enable OCR fallback even if some text found
Debug: Show retrieved chunks
Select LLM model: Qwen3-4B-Instruct-25...
Query scope: All files [✓ doc1.pdf]

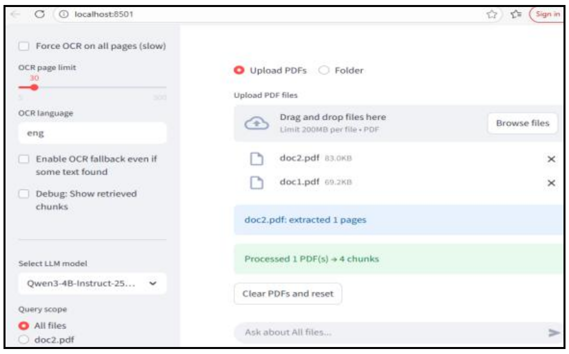
Chat History

What is Data Model

The provided content does not define or explain what a data model is. It discusses database system architecture, client/server architectures, and cloud computing environments but does not address the concept of a data model.

Export JSON
Export CSV
Ask about doc1.pdf...

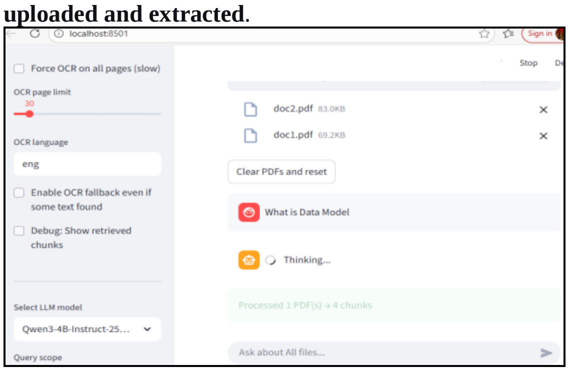
Figure 7. Example answer generated for a query, demonstrating document-grounded response or clear indication of unavailable information.


localhost:8501

Force OCR on all pages (slow)
OCR page limit 30
OCR language: eng
Enable OCR fallback even if some text found
Debug: Show retrieved chunks
Select LLM model: Qwen3-4B-Instruct-25...
Query scope: [✓ All files] doc2.pdf

[✓ Upload PDFs] Folder
Upload PDF files
Drag and drop files here
Limit 200MB per file • PDF
Browse files
doc2.pdf 83.0KB
doc1.pdf 69.2KB
doc2.pdf: extracted 1 pages
Processed 1 PDF(s) → 4 chunks
Clear PDFs and reset
Ask about All files...

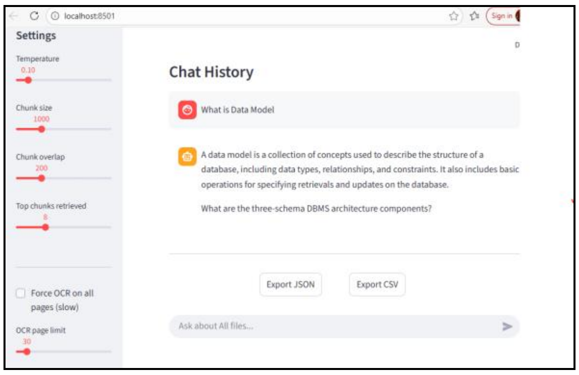
Figure 8. Interface showing a second document (doc2.pdf) uploaded and extracted.


localhost:8501

Force OCR on all pages (slow)
OCR page limit 30
OCR language: eng
Enable OCR fallback even if some text found
Debug: Show retrieved chunks
Select LLM model: Qwen3-4B-Instruct-25...
Query scope

Stop
doc2.pdf 83.0KB
doc1.pdf 69.2KB
Clear PDFs and reset
What is Data Model
Thinking...
Processed 1 PDF(s) → 4 chunks
Ask about All files...

Figure 9. System reasoning process for a query on doc2.pdf.



Temperature 0.10
Chunk size 1000
Chunk overlap 200
Top chunks retrieved 8
OCR page limit 30

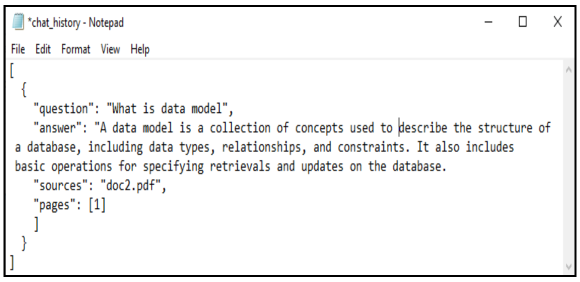
Chat History

What is Data Model

A data model is a collection of concepts used to describe the structure of a database, including data types, relationships, and constraints. It also includes basic operations for specifying retrievals and updates on the database.

What are the three-schema DBMS architecture components?

Figure 10. Question-and-answer interaction for doc2.pdf.



```
[
  {
    "question": "What is data model",
    "answer": "A data model is a collection of concepts used to describe the structure of
a database, including data types, relationships, and constraints. It also includes
basic operations for specifying retrievals and updates on the database.
    "sources": "doc2.pdf",
    "pages": [1]
    ]
  }
]
```

Figure 11. Exported answers in JSON format

5. CONCLUSIONS

The rapid growth of PDF documents in sensitive domains, including finance, healthcare, and government, has rendered manual review increasingly unsustainable. Cloud-based AI solutions, however, introduce high costs, latency, and potential privacy risks. To address these challenges, this paper presents LocalRAG, a fully offline Retrieval-Augmented Generation (RAG) framework. Implemented as a standalone Python and Streamlit application, LocalRAG operates entirely on CPU hardware without reliance on external APIs or internet access. The system integrates robust PDF processing through PyMuPDF and Tesseract OCR, a hybrid retriever (FAISS and BM25) for precise document grounding, and a quantized, instruction-tuned large language model (Qwen3-4B-Instruct) for efficient, private inference. LocalRAG extends prior offline RAG systems by supporting OCR for scanned documents, enhancing privacy through metadata anonymization, and delivering practical performance on consumer-grade hardware. Experimental evaluations demonstrate that the system provides competitive, low-latency, accurate, and well-grounded responses while maintaining full data sovereignty. The primary limitation of the framework lies in hardware dependency: on modest systems, such as those equipped with an Intel Core i5 processor and 8 GB of RAM, processing large document collections can be computationally intensive. Nevertheless, performance scales significantly with more powerful hardware, highlighting the framework's adaptability and scalability.

Future work may investigate lighter quantization strategies, alternative open-source large language models, improved parsing of complex document elements such as tables and figures, and the development of automated benchmarking suites to facilitate systematic comparison and replication. The framework ensures administrative traceability through structured exports while excluding sensitive metadata from the LLM context, providing a secure and reproducible baseline for local document intelligence. This study demonstrates that effective, privacy-preserving question answering over private document collections is achievable entirely on local hardware. LocalRAG provides a practical, open framework for deployment in regulated, security-conscious, and resource-constrained environments, enabling organizations to leverage document AI without compromising data control. **.**

6. ACKNOWLEDGMENTS

My thanks to the experts who have contributed towards development of the template.

7. REFERENCES

[1] OpenAI. (2023). *GPT-4 technical report* [Technical report]. arXiv. https://doi.org/10.48550/arXiv.2303.08774

[2] Meta AI. (2024). *Llama 3: Open foundation and fine-tuned chat models* [Technical report]. Meta AI. https://ai.meta.com/blog/meta-llama-3/

[3] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., … & Sayed, W. E. (2023). *Mistral 7B* [Technical report]. arXiv. https://arxiv.org/abs/2310.06825

[4] Qwen Team. (2023). *Qwen technical report* [Technical report]. arXiv. https://arxiv.org/abs/2309.16609

[5] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval, 3*(4), 333–389. https://doi.org/10.1561/1500000019

[6] PaddleOCR. (2020). *PaddleOCR* [Computer software]. GitHub. https://github.com/PaddlePaddle/PaddleOCR

[7] Kim, G., Hong, T., Yim, M., Park, J., Yim, J., Hwang, W., … & Park, S. (2022). OCR-free document understanding transformer. In *European Conference on Computer Vision* (pp. 498–517). Springer. https://doi.org/10.1007/978-3-031-19830-4_29

[8] Bommasani, R., Hudson, D., Adeli, E., Altman, R., Arora, S., von Arx, S., & Liang, P. (2021). *On the opportunities and risks of foundation models* [Technical report]. arXiv. https://arxiv.org/abs/2108.07258

[9] Shi, Y., Yu, K., Dong, Y., & Chen, F. (2026). Large language models in education: A systematic review of empirical applications, benefits, and challenges. *Computers and Education: Artificial Intelligence, 10,* 100529. https://doi.org/10.1016/j.caeai.2025.100529

[10] Tonmoy, S. M. I., Zaman, S. M. M., Jain, V., Rani, A., Rawte, V., Chadha, A., & Das, A. (2024). A comprehensive survey of hallucination mitigation techniques in large language models. arXiv. https://arxiv.org/abs/2401.01313

[11] Delikoura, I., Yi, R., & Fung, P. H. (2025). From superficial outputs to superficial learning: Risks of large language models in education. arXiv. https://arxiv.org/abs/2509.21972

[12] Almufarreh, A., Ahmad, A., Arshad, M., Onn, C. W., & Elechi, R. (2025). Ethical implications of ChatGPT and other large language models in academia. *Frontiers in Artificial Intelligence*, *8*, Article 1615761. https://doi.org/10.3389/frai.2025.1615761

[13] Bender, E. M., Gebru, T., McMillan Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610–623). ACM. https://doi.org/10.1145/3442188.3445922

[14] Li, Z., Fenghe, L., Wang, X., Fu, Q., & Ren, W. (2024). Ethical considerations and fundamental principles of large language models in medical education: Viewpoint. *Journal of Medical Internet Research*, *26*, e60083. https://doi.org/10.2196/60083

[15] Sciannameo, V., Jahier Pagliari, D., Urru, S., Grimaldi, P., Ocagli, H., Ahsani-Nasab, S., Comoretto, R. I., Gregori, D., & Berchialla, P. (2024). Information extraction from medical case reports using OpenAI InstructGPT. *Computer Methods and Programs in Biomedicine*, *255*, 108326. https://doi.org/10.1016/j.cmpb.2024.108326

[16] Joshi, N., Saparov, A., Wang, Y., & He, H. (2024). LLMs are prone to fallacies in causal inference. arXiv. https://arxiv.org/abs/2406.12158

[17] Lee, K., Yang, S., Jeong, J., Lee, Y., & Shin, D. (2025). Enhancing security and applicability of local LLM-based document retrieval systems in smart grid isolated environments. *Electronics*, *14*(17), 3407. https://doi.org/10.3390/electronics14173407

[18] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., … & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, *33*, 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[19] Arun, G., Perumal, V., Urias, F. P. J. B., Ler, Y. E., Tan, B. W. T., Vallabhajosyula, R., Tan, E., Ng, O., Ng, K. B., & Mogali, S. R. (2024). ChatGPT versus a customized AI chatbot (AnatBuddy) for anatomy education: A comparative pilot study. *Anatomical Sciences Education*, *17*, 1396–1405. https://doi.org/10.1002/ase.2463

[20] Han, Z. F., Lin, J., Gurung, A., Thomas, D., Chen, E., Borchers, C., Gupta, S., & Koedinger, K. (2024). Improving assessment of tutoring practices using retrieval-augmented generation. In *AI for Education: Bridging Innovation and Responsibility at the 38th AAAI Annual Conference on AI*. https://openreview.net/forum?id=Us1EF795Ys

[21] Lee, K., Yang, S., Jeong, J., Lee, Y., & Shin, D. (2025). Enhancing Security and Applicability of Local LLM-Based Document Retrieval Systems in Smart Grid Isolated Environments. *Electronics*, *14*(17), 3407. https://doi.org/10.3390/electronics14173407.

[22] Astrino, P. (2025). Local hybrid retrieval-augmented document QA. arXiv. https://doi.org/10.48550/arXiv.2511.10297

[23] Nagarajan, G., Kumar, O., & Santhiappan, S. (2025). PolicyBot: Reliable question answering over policy documents. arXiv. https://doi.org/10.48550/arXiv.2511.13489

[24] Tyndall, E., Wagner, T., Gayheart, C., Some, A., & Langhals, B. (2025). Feasibility evaluation of secure offline large language models with retrieval-augmented generation for CPU-only inference. *Information*, *16*(9), 744. https://doi.org/10.3390/info16090744

[25] Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 758–759). ACM. https://doi.org/10.1145/1571941.1572114

[26] Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7th ed.). Pearson. [PDF]. https://womengovtcollegevisakha.ac.in/departments/Fundamentals%20of%20Database%20Systems%20.pdf