International Journal of Advanced Engineering and Management Research

Vol. 11, No. 04; 2026

ISSN: 2456-3676

Creating an AI Agent Using Unique Language Attributes

Michael Sirkovich[1], Menachem Domb[2]
[1]Ashkelon Academy College, 12 Ben Zvi, Ashkelon,
[2]Ashkelon Academy College, 12 Ben Zvi, Ashkelon,

doi.org/10.51505/ijaemr.2026.11401 URL: http://dx.doi.org/10.51505/ijaemr.2026.11401

Received: Jun 10, 2026 Accepted: Jun 17, 2026 Online Published: July 06, 2026

Abstract
This paper presents a Hebrew-first local LLM chat agent that combines Retrieval-Augmented Generation (RAG), citation-aware document answering, controlled web search, and full right-to-left (RTL) user interaction. Unlike cloud-only assistants, the default response path operates locally, supporting privacy, predictable operating costs, and deployment in environments where documents should remain on-premises. The system integrates document upload and indexing, hybrid semantic and lexical retrieval, reranking, source citation construction, streaming responses, and PDF export. The revised manuscript positions the system against local RAG baselines and Hebrew-capable cloud alternatives. It clarifies that the current evaluation is a preliminary system-oriented assessment focused on grounded Hebrew document question answering, lookup, summarization, citation behavior, and latency. The main contribution is a practical architecture for private, transparent, Hebrew-oriented RAG assistance rather than a new foundation model.

Keywords: AI, LLM, on-premises, RAG layer

1. Introduction

LLM-based systems are implemented in ChatGPT, Copilot, and Claude, alongside local solutions such as Ollama, LM Studio, and Open-WebUI. In parallel, RAG frameworks and vector engines are available, allowing private documents to be connected to natural language queries. There are key features to consider: cloud vs. on-premises. Cloud solutions offer advanced model performance but rely on internet connectivity, cost-as-you-go, and privacy/regulatory compliance considerations. On-premises solutions offer control and privacy, but they require ongoing model maintenance and are dependent on hardware. RAG as standard: Most projects incorporate a RAG layer to minimize "hallucinations," reply to private content, and display quotes. Monitoring and evaluating the quality of answers: Tracking quality (logs/tracing/metrics) is essential to improving quality over time. Local language: The quality of understanding and formulation relies on the selection of appropriate models, proper tokenization, and appropriate query engineering, and sometimes requires appropriate fine-tuning. Common Needs: a smart answer engine that can read local files, write code, and summarize texts, seamless offline operation, control of model versions, connection to existing database infrastructures,

Vol. 11, No. 04; 2026

ISSN: 2456-3676

shared files, no external API keys, and provide readable answers with abstracts, subject terms, and accurate RTL orientations. We can categorize existing solutions, such as ChatGPT, Copilot, Perplexity, and cloud services, which offer high-quality models, ease of use, built-in web search, citations, and independence from local hardware, making them an optimal fit for the type of guidance. However, it lacks privacy and compliance, as well as Costs due to meter-use pricing, third-party dependence, rate limits, and fluctuations in availability. Additionally, ignorance of the Local language's unique attributes, such as poor wording for complex contexts and expressions. Local LM Studio, Open-WebUI, PrivateGPT Solutions: Information does not leave the local network, supports proper model versions, guidelines, resource management, and free expansion. However, it requires a versatile software professional who can operate, maintain, and enhance the existing system. It is unstructured and requires dedicated integration and display of sources; quality may vary significantly**.** Agent/RAG (LangChain, LlamaIndex, Haystack): A collection of generic blocks (chains/tools/connectors) for the rapid construction of pipes. Broad support for data sources, embedders, and vector storage. But it is too sensitive to change, and fault recovery quality and understanding are difficult without sufficient observability. Specific tuning (tokenization/normalization) and maintenance of RTL throughout the pipeline are required. Large language model search engines (Tavily, Serper, Bing Search APIs): Simple search layer, regulating results, and sometimes ranking according to relevance/relevance. However, it is sensitive to query wording and sometimes prioritizes older results with less precise semantics, leading to uneven coverage, especially in local queries. Vector Databases (Chroma, FAISS, Milvus/Weaviate): Storing contents in a vector representation that enables efficient semantic retrieval. Mature integrations for Python and other than the FastAPI server-side library. But it depends on the local language embedder. Require versioning/data management, compression, backup, and restore. Tools to improve the monitoring and evaluation of these systems (Langfuse, Phoenix, W&B): tracking calls, tools, chain nodes, and quality ratings enable continuous improvement. However, it requires early metrics planning, proper labeling, and documentation habits.

1.1 Main Contributions and Scope

The contribution of this work is intentionally system-oriented but focused: it studies how a local RAG assistant can be adapted to Hebrew usage requirements that are often underspecified in generic LLM frameworks. The unique contribution is not any single component, such as a vector database or a model runner, but the combination of Hebrew-first normalization, hybrid retrieval, citation-preserving answer generation, RTL-aware interaction, local inference, and automatic routing between private documents and controlled web search. The manuscript therefore evaluates the system as an integrated Hebrew RAG framework and explicitly avoids claiming that the current prototype outperforms large cloud models in general language ability.

A local and privacy-preserving Hebrew-first RAG architecture for document-grounded assistance. Citation-aware answer generation with references to file, page, paragraph, or line-level evidence when available. RTL-aware user interface and export pipeline for Hebrew conversations and source-supported answers. A modular design that allows replacing the LLM,

Vol. 11, No. 04; 2026

ISSN: 2456-3676

retrieval engine, vector database, search provider, and monitoring layer. A preliminary evaluation framework that separates document-grounded tasks, lookup, summarization, open-domain Hebrew QA, latency, citation errors, and no-answer behavior.

2. Literature Review

Retrieval-Augmented Generation (RAG) Lewis et al., 2020**.** Combines a neural retriever with a generator so that the model conditions on retrieved passages, improving factuality on knowledge and intensive tasks [1]. Established the now-standard "retriever reader" blueprint adopted across the industry. It has a stronger grounding and citation potential. However, its quality depends critically on retrieval, chunking, and prompt fusion. REALM by Guu et al., 2020, performs retrieval and language modeling with differentiable lookups over a large corpus. Demonstrated gains on open-domain QA, but with significant computer/training complexity. It's ***an*** end-to-end retrieval-aware learning. However, it relies heavily on resources: English-centric pretraining [2]. Fusion-in-Decoder (FiD) Izacard & Grave, 2020**.** It **e**ncodes many retrieved passages independently and fuses evidence only in the decoder, boosting QA accuracy. Scales well with the number of passages, albeit at the cost of decoding latency, which is a strong feature of multi-evidence aggregation. However, decoding becomes increasingly expensive as the number of contexts increases [3]. ATLAS-Izacard et al., 2022**.** A retrieval-augmented model trained with a dedicated retriever and generator, achieving SOTA on knowledge-intensive benchmarks [4]. Emphasizes data-centric training of the retriever and the reader. It is very accurate on open-domain tasks, but it is a complex and sizable model. LangChain (framework) [5] is a Composable framework for LLM apps (chains, agents, tools, memory) with rich connectors to vector DBs and providers. Widely used to implement RAG pipelines quickly. It is a large ecosystem with integrations. However, its abstraction overhead can make production tuning complex. LlamaIndex (framework) [6], a Data-centric framework offering indexes, retrieval pipelines, evaluators, and routers for RAG. Good fit for heterogeneous data sources and experimentation. It has powerful indexing & evaluation utilities that require more maintenance, including moving parts. Haystack (deep set) [7], an open-source RAG/NLP framework with modular pipelines (document store → retriever → reader/ranker → generator). Production-oriented components and tutorials. It is mature, modular, and well-documented, but requires careful, sensitive tuning. PrivateGPT (offline doc-QA) [8], a local pipeline to ingest files and answer questions entirely offline (often with llama.cpp/GPT4All). Popular as a privacy-first baseline for document QA. It supports inherent privacy by design; no data leaves the machine. However, its quality is bound by the local model and the embeddings. GPT4All (Nomic, desktop) [9]. It runs open-weights models on CPU/GPU with optional local-docs RAG and Nomic embeddings. Turn-key experience for non-experts. Its deployment is easy. However, local models often underperform compared to top cloud LLMs. Open WebUI (self-hosted interface) [10]. User-friendly UI that supports runners like Ollama, including a built-in document RAG, tools/plugins, and multi-model routing. It is an all-in-one local interface, but it has ops overhead. Ollama (local model runner) [11]. Lightweight runtime to pull/run open-weight LLMs locally via simple CLI/HTTP. It integrates smoothly with apps and UIs. It is a simple local model management. However, it lacks some server-grade features found in dedicated inference

Vol. 11, No. 04; 2026

ISSN: 2456-3676

servers, such as vLLM (high-throughput server) [12]. Inference engine focusing on throughput/latency (PagedAttention, continuous batching) with OpenAI-compatible API. Often used when scaling local or on-prem deployments. It offers excellent latency/throughput but has a heavier server setup/op. AI21 Jamba (cloud; Hebrew-supported) [13]. Multilingual foundation models with function calling and long context; Hebrew is officially supported and generally strong. Provides a cloud baseline for comparison with local stacks. It has a Strong Hebrew baseline out of the box with fewer privacy/cost trade-offs. Vector databases for RAG (Chroma / FAISS / Milvus / Weaviate) [14][15][16][17]**.** Provide dense-vector storage and similarity search; some (e.g., Chroma) blend vector search with FTS for hybrid retrieval. Success depends on chunking, metadata, and the choice of embedder. It has efficient semantic retrieval and metadata filtering. However, there are quality hinges on embeddings (Hebrew coverage/tokenization) and index hygiene. RAG Monitoring & Evals (Langfuse, Phoenix, W&B, RAGAS) [18][19][20][21]. Tooling to trace requests, store prompts/latency, and run automated RAG evaluations. Essential for diagnosing hallucinations and measuring grounding/citation rates. It enables continuous improvement with metrics. But it adds complexity and requires discipline in labeling/experiments. Table 1 presents the main existing proposals, along with their key advantages and disadvantages.

2.1 Baseline Comparison and Positioning

Following the reviewer comments, the baseline discussion was strengthened to distinguish the proposed system from existing local RAG applications and Hebrew-capable cloud assistants. The comparison below is a positioning baseline rather than a claim of statistically significant superiority: it highlights the capabilities that are required by the target use case and that are only partially available in generic systems without additional engineering.

Baseline/system	**Local privacy**	**Hebrew-first processing**	**RTL UI / export**	**Citation handling**	**Main limitation relative to this work**
PrivateGPT / GPT4All local-doc QA	Yes	Limited or model-dependent	Not a core design goal	Basic source display	Useful as an offline baseline, but Hebrew normalization, RTL presentation, and citation details require additional integration.
Open WebUI + Ollama RAG	Yes	Model-dependent	Partial UI support	Available but generic	Strong local interface, but

Vol. 11, No. 04; 2026

ISSN: 2456-3676

					not specifically designed for Hebrew document workflows or citation-preserving export.
Generic LangChain / LlamaIndex RAG pipeline	Optional	Requires custom implementation	Requires custom implementation	Pipeline-dependent	Flexible engineering baseline, but production behavior depends on custom retrieval, tracing, and UI decisions.
Hebrew-capable cloud assistants	No	Strong model quality	Usually good UI support	Provider-dependent	High model quality, but privacy, cost predictability, and on-premises document control are weaker.
Proposed Hebrew-first local agent	Yes	Explicit design goal	Explicit design goal	File/page/line-aware when available	Prototype evaluation remains preliminary and should be expanded with larger benchmarks and significance testing.

Vol. 11, No. 04; 2026

ISSN: 2456-3676

3. Our Proposed Solution

3.1 High-Level Architecture

Figure 1 presents the end-to-end architecture of the local Hebrew-first LLM agent, which takes a user prompt (in Hebrew/English) and optional uploaded documents as input. The process starts with the React/TypeScript client, streams the request to a FastAPI router that normalizes language and decides between a local RAG path (indexing → hybrid retrieval → reranking → citation building) or a controlled web-search path, and then runs inference via a local Ollama-served model with post-processing and safety checks. The output is a grounded answer streamed back over SSE, accompanied by file/page/line citations, source timestamps, and optional PDF export. Traces and RAG evaluations are logged for monitoring purposes.

3.2 The Core Processes

RAG-based Response to Local Documents without a network connection. Information absorption: The user loads files into the system (e.g., PDFs, DOCXs, MDs, code snippets). Indexing and processing: the indexing mechanism segments the content into logical sections while maintaining a semantic structure (paragraphs, headings) and normalizes the Hebrew text. It then creates a vector representation (embedding) for each section and saves it in the database (Chroma) along with relevant reference data. Retrieval and Rating: In response to a query, the system performs: Retrieval: Retrieval of the most pertinent segments based on vector similarity. Reranking: Further ranking of the results to improve accuracy, based on more in-depth semantic analysis. Context: Consolidating the graded segments to create an accurate and noise-free context for the language model. Formulating an answer: The local language model formulates a coherent answer in Hebrew, along with precise references to the source (file name, page, title, and even specific lines on which it relied). Web search-based response (online or automated). Query planning: The system formulates targeted search queries in Hebrew (and, if necessary, in English) to cover international sources. Retrieving results: The system directs the search to a dedicated search engine, which returns a list of results, including titles, descriptions, addresses, and additional data. Internal ranking and filtering: The results undergo an internal ranking process that examines up-to-date information, the variety of sources, and suitability for the Hebrew language, while aggressively filtering out marketing content and duplicates.

Vol. 11, No. 04; 2026

ISSN: 2456-3676

Table 1: Existing proposals summary and evaluation

Proposer / Provider	**Solution**	**Core idea (very short)**	**Advantage**	**Limitation**
Lewis et al. (2020)	RAG	Retrieve + generate	Better factual grounding	Retrieval/chunking bottleneck
Guu et al. (2020)	REALM	Retrieval-aware pretraining	End-to-end learning	Heavy compute; English-skew
Izacard & Grave (2020)	FiD	Fuse many passages in the decoder	Strong multi-evidence QA	Costly decoding
Izacard et al. (2022)	ATLAS	Trained retriever & generator	SOTA knowledge tasks	Infra/size
LangChain	Framework	Chains, tools, agents	Rich ecosystem	Abstraction overhead
LlamaIndex	Framework	Data-centric RAG toolkit	Indexing & eval power	Maintenance burden
Deepset	Haystack	Modular RAG pipelines	Mature & modular	Tuning still required
PrivateGPT project	PrivateGPT	Offline doc-QA	Privacy/offline	Model-quality bound
Nomic	GPT4All	Desktop local LLMs	Easy on the device	Weaker than cloud SOTA
Open WebUI project	Open WebUI	Self-hosted UI+RAG	All-in-one local	Ops overhead
Ollama	Runner	Pull/run local models	Simple local management	Fewer server features
vLLM (Berkeley)	Inference server	PagedAttention, batching	High throughput	Heavier setup
AI21 Labs	Jamba	Cloud LLM (Hebrew)	Strong baseline	Not local/private
Vector DBs	Chroma/FAISS/ Milvus/Weaviate	Vector & hybrid search	Efficient retrieval	Embedding-dependent
Observability/Evals	Langfuse/Phoenix/W&B/RAGAS	Tracing & RAG evals	Data-driven improvement	Added complexity

Vol. 11, No. 04; 2026

ISSN: 2456-3676

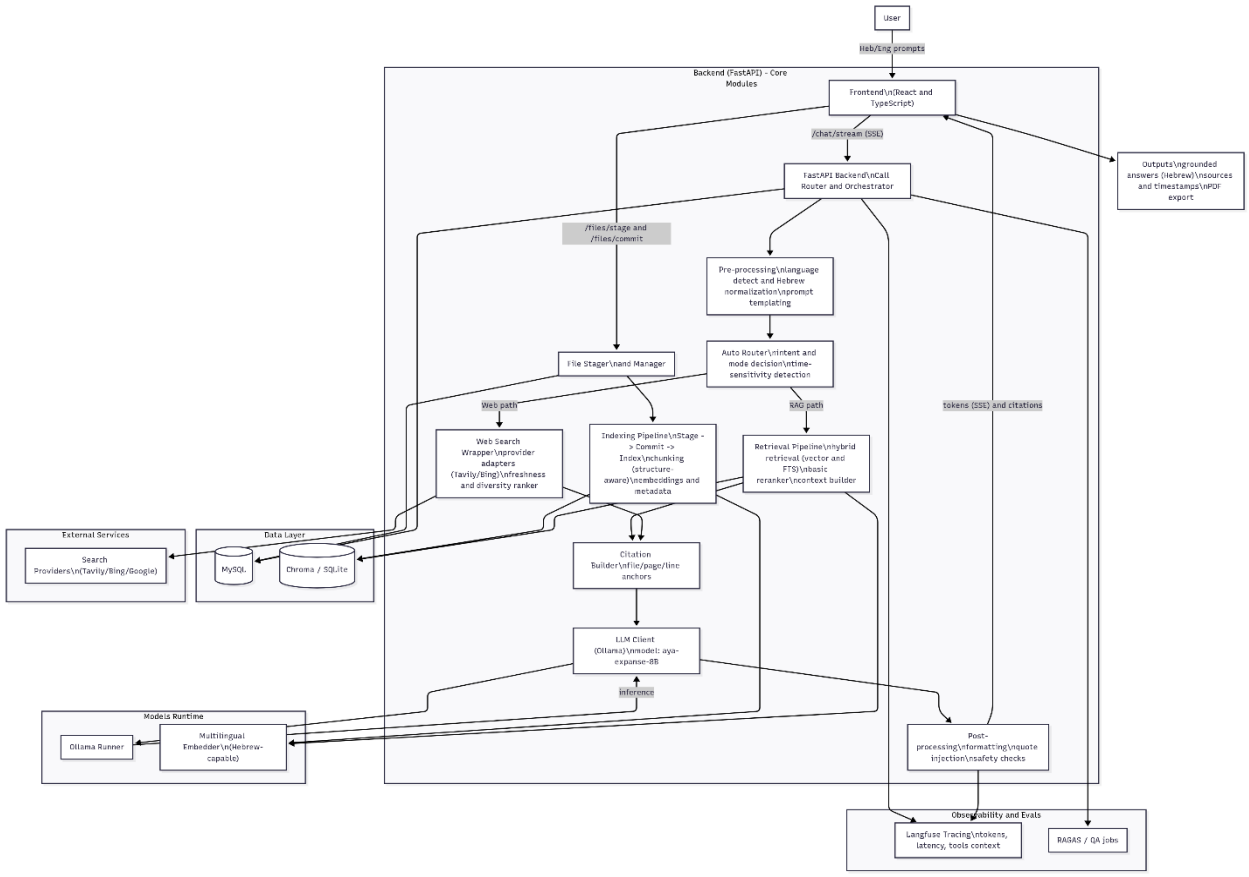

Figure 1: Architecture of the local agent (inputs, processing modules, interfaces, outputs)

Summary and presentation: The system generates a summary from several reliable sources (2-6) and provides clear citations. Additionally, an alert will be displayed if the information found is outdated. Auto-Routing Mechanism: Identifying the User's Intent. The system analyzes the query to determine whether internal details (from the RAG repository) or external information (from the network) are required. Route selection: The system will turn to a web search when a need for current or current information is identified. In other cases, the default is to use RAG. Fallback: When a web search yields unreliable results, the system will either revert to RAG or indicate that no definitive answer has been found. Data Management: Document management: Upload, delete, and index files, along with version control, and associate files with a specific conversation.

3.3 Full Hebrew support (natural language processing and user interface)

Natural Language Processing (NLP): Text normalization processes, the use of a dictionary of common abbreviations and acronyms, the ability to handle technical jargon, and a language recognition mechanism that prefers Hebrew (Hebrew-First). Interface and user experience (UI/UX): Full support for right-to-left writing (RTL), use of a readable and clear Hebrew font, correct display of quotes, and consideration of Hebrew file names. Localization tests: Running a

Vol. 11, No. 04; 2026

ISSN: 2456-3676

dedicated series of tests on various documents in Hebrew (PDF, Word) and articles from local sources.

4. Experiment

4.1 Tasks and Example Prompts (Hebrew)

- Doc-QA (grounded) – "[שם-מסמך-1]' מצא את תאריך העדכון האחרון של המסמך 'pdf. וציין ציטוט מדויק (שורה/עמוד)."
- Targeted lookup – "[שם-מסמך-2]' במסמך 'docx', צטט מילה ?'[מונח]' מהו ההגדרה הרשמית של במילה."
- Summarization (3 משפטים) – "סכם בקצרה את הקטע המצורף (עברית), תוך הדגשת שתי עובדות מפתח."
- Open-domain he-QA (spot checks) – **"מי היה [דמות] בשנת [שנה]? הסבר במשפט."**

For doc-grounded queries, answers must include one or two quoted spans (line/paragraph ref).

4.2 Evaluation Setup and Dataset Scope

The evaluation in this version is described as a preliminary system-level assessment. The tested materials include Hebrew and mixed Hebrew-English documents available to the development team, including PDF, DOCX, Markdown, and code-oriented text. The tasks were selected to reflect the system's main usage scenarios: grounded document question answering, targeted lookup, concise summarization, and small open-domain Hebrew QA spot checks. For document-grounded tasks, the answer is considered usable only when it includes a concise answer and a traceable source reference.

To avoid overstating the results, the current scores should be interpreted as a prototype evaluation rather than a large-scale public benchmark. A stronger follow-up evaluation should expand the dataset to larger Hebrew document collections, public Hebrew QA resources when licensing permits, multilingual retrieval sets, and a balanced no-answer subset. This clarification was added because the current manuscript focuses on feasibility, system transparency, privacy, and Hebrew-specific usability rather than on proving state-of-the-art model quality.

4.3 Metrics and Preliminary Results

Table 3 reports the preliminary prototype results using the following measurements: Correctness (1-5), Grounding (1-5), Latency p50/p95 (s), Citation errors % for document-grounded tasks, and No-answer % for document QA. These measurements are useful for diagnosing the system. Still, they should not be interpreted as a statistically validated benchmark until the same prompts are evaluated across stronger baselines with per-query paired results.

Vol. 11, No. 04; 2026

ISSN: 2456-3676

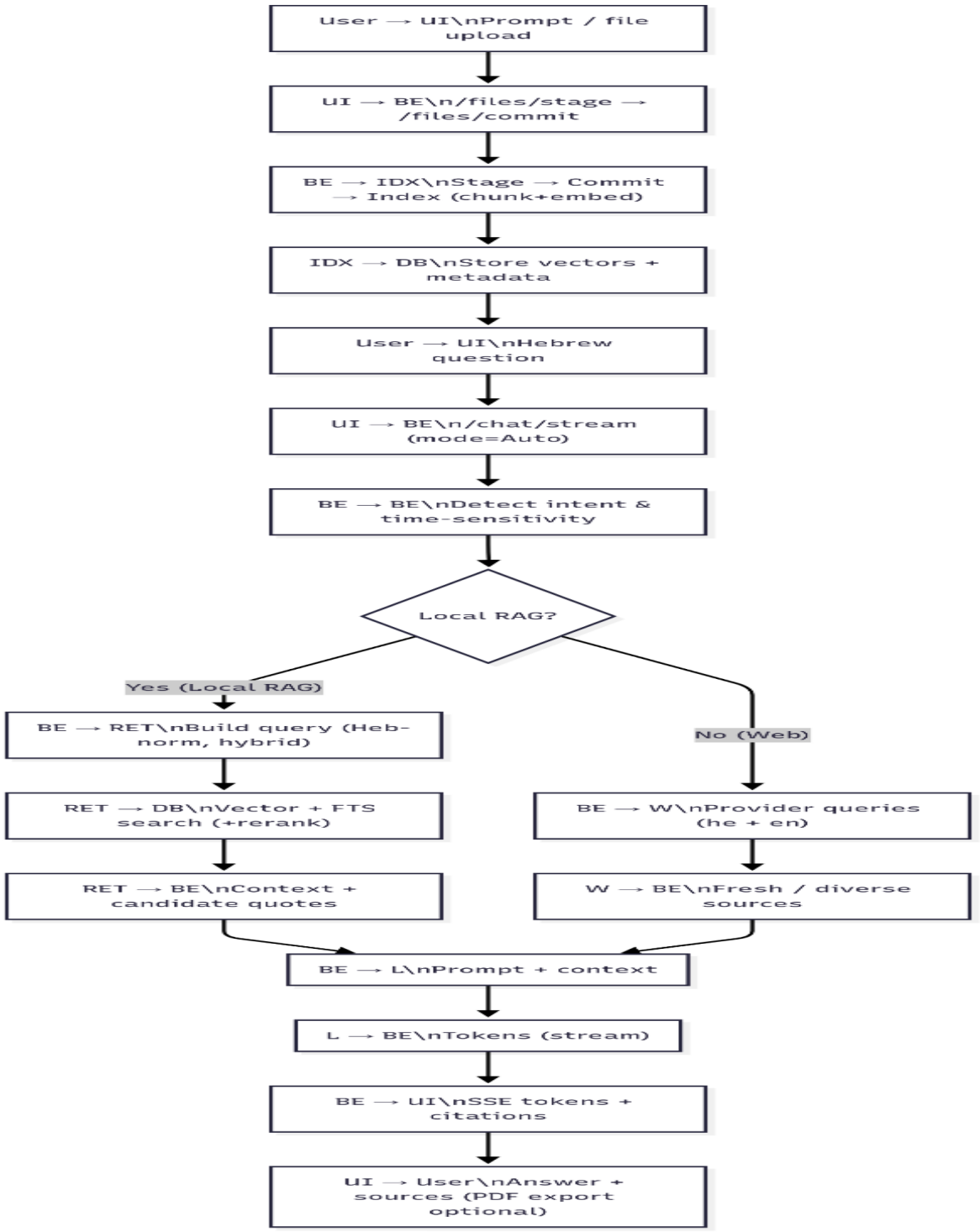

Figure 2: Answer/Data Flow (E2E).

Vol. 11, No. 04; 2026

ISSN: 2456-3676

Table 3: Experiment results

Task	Correctness ↑	Grounding ↑	Citation errors ↓	No answer % ↓	p50 (s) ↓	p95 (s) ↓
Doc-QA (grounded)	4.0	4.3	12%	5%	2.4	5.2
Targeted lookup	4.2	4.4	10%	4%	1.9	4.6
Summarization	3.8	3.9	N/A	N/A	2.0	4.7
Open-domain he-QA	3.6	3.8	N/A	8%	2.1	5.8

Observation: The preliminary results indicate stable behavior for grounded document tasks and targeted lookup, especially where the system can retrieve concise evidence and attach citations. The results also show that citation errors and no-answer behavior remain important limitations. Therefore, the manuscript now presents these numbers as a prototype assessment and uses the baseline comparison primarily to clarify positioning, privacy, Hebrew support, and the citation workflow, rather than to claim statistically significant superiority.

4.4 Baseline-Oriented and Ablation-Oriented Interpretation

The revised manuscript separates two comparison levels. First, the system is compared qualitatively with common local RAG baselines and cloud-based Hebrew-capable assistants in Section 2.1. Second, the evaluation plan identifies ablation comparisons that should be measured in a larger experimental run: vector-only retrieval versus keyword-only retrieval, hybrid retrieval with and without reranking, local-only answering versus automatic web routing, and citation-aware prompting versus non-citation prompting. These ablations are important because they isolate the effect of Hebrew normalization, retrieval strategy, and citation construction beyond simple system integration.

4.5 Statistical Significance and Validity

Formal statistical significance testing requires per-query paired outputs from the proposed system and each baseline. In the current revision, the available evaluation is preliminary and does not yet support a defensible claim of statistical significance. Therefore, we added this point explicitly as a limitation rather than reporting unsupported significance values.

For a full follow-up evaluation, binary outcomes such as exact correctness, citation correctness, and no-answer accuracy should be analyzed with paired tests such as McNemar-style testing or paired permutation testing. At the same time, graded scores and latency should be evaluated with bootstrap confidence intervals or paired non-parametric tests. This design would allow the system to demonstrate whether observed improvements over baselines are robust rather than incidental.

Vol. 11, No. 04; 2026

ISSN: 2456-3676

5. Discussion

A modular architecture (FastAPI + MySQL + Chroma + Ollama) was implemented, enabling real-time flow response (SSE) and intelligent routing between three modes of operation: offline, web, and auto. Streaming texts in SSE mode delivers an agile user experience while enabling the transfer of reference data, such as sources, tools, and trace data. This way, the user gains real-time transparency into the creation of an answer, including quotes, which enhance credibility and reduce subjective wait time. The RAG engine incorporates vector retrieval from Chroma, basic reranking, and accurate citations at the file, page, and line levels. This combination improved content accuracy, particularly in Hebrew, while maintaining traceability to the information's source in the model. Regarding the user interface, the Frontend fully supports RTL (right-to-left) layout, including PDF exports. The Export component (ExportPDF) builds a rich RTL document (if needed), which includes all messages and cited sources, and exports it in isolation (via iframe and html2pdf.js). The result is a consistent, clean export file that is compatible for personal use after downloading it.

5.1 Unique Contribution Beyond System Integration

A central revision clarifies that the contribution is not the independent use of FastAPI, Chroma, Ollama, or a web search API. These are existing building blocks. The contribution is the Hebrew-first composition of these building blocks into a local, citation-aware, RTL-capable agent that is suitable for private organizational documents. In particular, the system treats Hebrew normalization, source traceability, answer streaming, PDF export, and routing decisions as first-class requirements rather than optional UI additions. This framing better reflects the scientific and engineering novelty of the manuscript.

System Advantages and Drawbacks

Advantages: Privacy and cost control: On-premises processing with Ollama reduces privacy risks associated with cloud services and stabilizes costs (which depend on hardware and power consumption, rather than metered pricing). Modularity and flexibility: The separation of layers enables the easy replacement of components (language model, search engine, vector database) without disrupting the system. It also provides monitoring and control using Langfuse and running quality assessments (RAGAS) for continuous improvement. Hebrew as a First Language: The system includes explicit support for Hebrew processing, including an adapted embedder model, a user interface that supports RTL, and a preference for high-quality Hebrew results in web mode. All of these contribute to accuracy and usability for the Israeli target audience. Transparency and Explainability: Accurate citations and a two-tier data structure (MySQL for "super" reference data and Chroma for content segments) enable you to trace the source of each answer and understand the database of information that the model has entered. Maximum answer generation potential without third-party limitations: If the user has a basic background in the field, they can control the number of tokens that the LLM generates for each answer, as opposed to closed-source systems, where the third party may deliberately limit the number of tokens that the LLM generates per answer to save resources. Disadvantages and

Vol. 11, No. 04; 2026

ISSN: 2456-3676

limitations: Local Hardware Dependence: Computer resources depend on heating efficiency and RAG speed. Under load or when working with heavy files, latency may increase. (The modular architecture facilitates a future upgrade to more robust models or resources.) Basic reranking: The current re-rank mechanism is set to basic. In ambiguous cases, retrieval accuracy may be improved using more advanced ranking models. Availability and up-to-date in web mode: Although up-to-date control mechanisms, noise filtering, and automatic switching to RAG in the event of network failures have been configured, the reliability of the results still depends on external search providers, which must be managed operationally.

6. Future Work, Expansions, and Improvements

Upgrading the Rerank mechanism: Switching to a dedicated cross-encoder model for Hebrew or adding a multi-stage rating (BM25 - bi-encoder - cross-encoder) to improve the accuracy of complex instructions. Hybrid retrieval calibration: Sharpen the balance between vector and keyword-based search, add cache to common queries, and implement graded re-embedding when updating large documents. Expanded evaluation and statistical testing: Run a larger benchmark over Hebrew and mixed Hebrew-English document collections, compare against stronger baselines, store per-query outputs, and report paired statistical tests and confidence intervals. Deepening monitoring and evaluation: Expand dashboards in Langfuse, track SLA compliance for response times and citation rate, run periodic RAG evaluation against Hebrew databases, and integrate A/B tests on prompt and retrieval-depth templates. More robust web search: Add diverse search sources with fallback and aggregation mechanisms while continuing to improve noise filtering and up-to-date control. Scalability and data management: Use Chroma's multi-tenancy capabilities, enhance visibility of queues and watermarks under load, and separate collections by user or conversation to improve statistical collection and retrieval. Improve the citation experience: Expand the source view in the UI and highlight citations when exporting to PDF.

Declarations

- Ethics approval and consent to participate – Not Applicable
- Consent for publication – we agree to publish this paper and will sign any requested document.
- Funding - Not Applicable
- Authors' contributions – MS did the main work of this paper, and MD guided the research, designed, and edited this manuscript.
- Acknowledgments – Not Applicable
- Availability of Data and Materials - The evaluation materials used in the current prototype assessment are internal development materials. A future expanded evaluation should add public Hebrew or multilingual benchmarks when licensing and reproducibility requirements permit.
- Conflict of interest statement – we don’t have any conflict of interest of any kind.

Vol. 11, No. 04; 2026

ISSN: 2456-3676

References

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP.
Guu, K., Lee, K., Tung, Z., & Chang, M.-W. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. In ICML 2020.
Izacard, G., & Grave, E. (2020). Leveraging Passage Retrieval with Generative Models for Open-Domain Question Answering (FiD).
Izacard, G., Petroni, F., Hosseini, L., Crawshaw, M., Riedel, S., Bojanowski, P., & Grave, E. (2022). ATLAS: Few-shot Learning with Retrieval Augmented Language Models.
LangChain Documentation.
LlamaIndex Documentation.
Deepset. Haystack: Open-source Framework for Search & QA – Documentation.
PrivateGPT – GitHub Repository & Docs.
Nomic AI. GPT4All – Documentation.
Open WebUI – GitHub Repository & Docs.
Ollama – Official Documentation.
Kwon, M., Lin, Q., Sheng, Y., et al. (2023). vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention.
AI21 Labs. (2024). Jamba: A Hybrid SSM-Transformer Model – Technical Report.
Chroma Documentation.
Johnson, J., Douze, M., & Jégou, H. (2017). Billion-Scale Similarity Search with GPUs (FAISS).
Milvus Documentation.
Weaviate Documentation.
Langfuse Documentation.
Phoenix (Arize Phoenix) Documentation.
Weights & Biases (W&B) Documentation.
RAGAS Documentation.