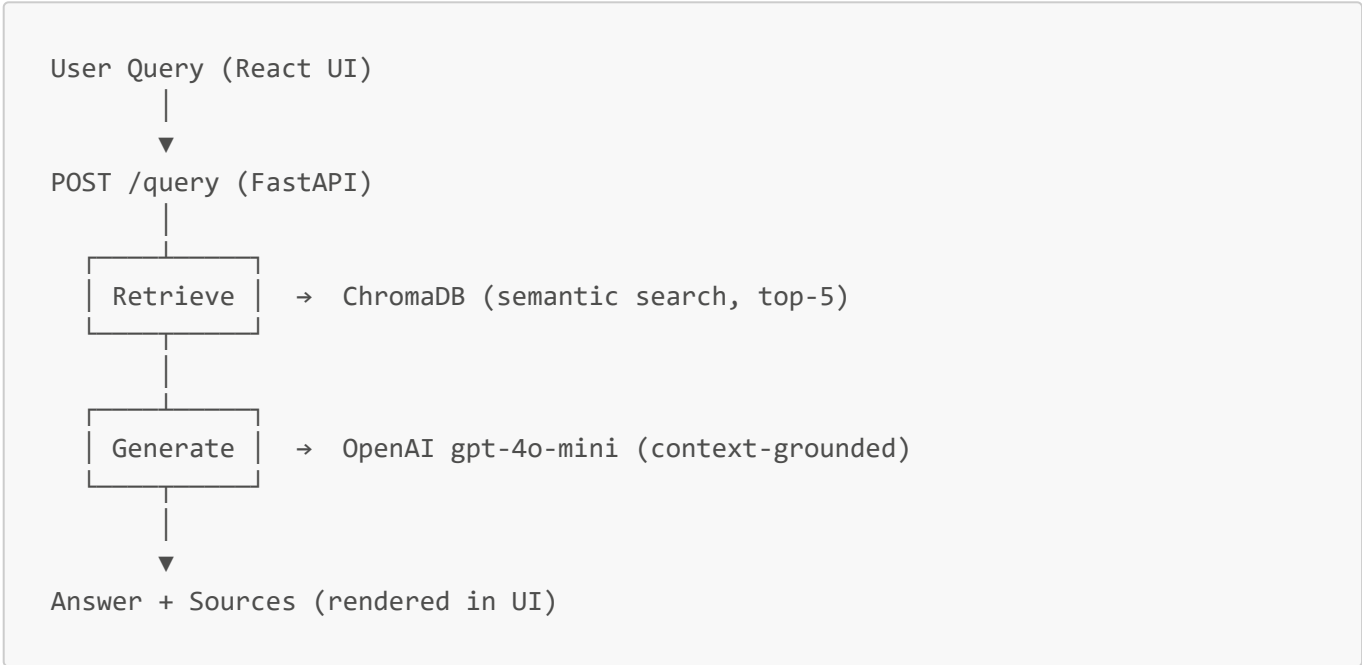
RAG Docs Assistant — Report

Motivation

The motivation for this project came from a real onboarding experience. The information was all there, but knowing which document to look at, which section was relevant, or even which keywords to search for was knowledge that took time to build up on its own. RAG is well-suited to this kind of problem, and tools like Confluence AI and Notion AI already tackle it commercially, which makes building one from scratch a good opportunity to understand what those tools are actually doing under the hood and where they break down.

Implementation Approach

System Architecture



Document Ingestion and Chunking

The knowledge base is 15 Markdown files in `backend/docs/`, covering onboarding, expense policy, API design, auth, error handling, and deployment workflows.

Rather than splitting on character count, ingestion uses a **heading-aware chunking strategy**:

1. Split on `##` (H2) headings — each section becomes its own chunk.
2. Sections over 1,500 characters are split further at paragraph boundaries to prevent oversized chunks.
3. Every chunk is prefixed with the document title and section heading so embeddings carry context even in isolation.

Embedding and Retrieval

Chunks are embedded with OpenAI's `text-embedding-3-small`, using the same model at query time to keep the embedding space consistent. Similarity search runs against ChromaDB via cosine distance (`score =`

`1 - distance`), returning the top-5 chunks by default.

Generation

Retrieved chunks are concatenated into a context block and injected into a system prompt that instructs the model to answer **only from that context**, returning an explicit "I don't know" when context is insufficient.

Model: `gpt-4o-mini` | **Temperature:** `0.1`

> *"You are an internal documentation assistant. Answer the user's question using ONLY the provided context below. Do not use any prior knowledge. If the context does not contain enough information to answer the question, respond with: 'I don't know based on the available documentation.' Be concise and direct. Cite the source document when possible."*

Frontend

The React + TypeScript UI has a query textarea, a Markdown-rendered answer panel, a collapsible sources panel, and a collapsible retrieved chunks panel for transparency into what the retriever pulled.

Evaluation

Retrieval Evaluation (Precision@5)

8 test questions were run across different documents in the knowledge base, checking whether the expected source document appears in the top-5 retrieved chunks. **Precision@5** measures what fraction of queries pull the right document within the top 5.

Script: `backend/eval.py` | **Endpoint:** `POST /retrieve` with `top_k=5`

Question	Expected Document	Retrieved	Rank
What should I focus on in my first week?	`first-week-playbook.md`	✓	1
How do I open a pull request?	`pull-request-checklist.md`	✓	1
How do I set up my development environment?	`onboarding.md`	✗	—
How do I add authentication to an endpoint?	`authentication-and-authorization.md`	✓	1
How do I handle errors in my service?	`error-handling-and-logging.md`	✓	1
How do I write tests for a backend API?	`testing-backend-apis.md`	✓	1
What does the overall system architecture look like?	`system-architecture-overview.md`	✓	1
How do I add a new API endpoint?	`adding-a-new-api-endpoint.md`	✓	1

Precision@5: 88% (7/8)

The one miss — "*How do I set up my development environment?*" targeting `onboarding.md` — is a **corpus collision**: the corpus also contains `local-development-setup.md`, which is a closer semantic match to that phrasing. The retriever picked the more relevant doc; the test label simply didn't account for answer overlap across documents.

Qualitative Response Quality

- **Groundedness:** The context-only system prompt at temperature 0.1 keeps answers close to the source text without drifting.
- **Source attribution:** The UI surfaces which documents contributed to every answer, so engineers can verify in one click.

Limitations and Future Work

- **Generation quality metrics** via RAGAS (faithfulness and answer relevancy) would measure whether answers are factually consistent with retrieved context, not just whether the right doc was retrieved.
- **Hybrid retrieval** combining BM25 with semantic search would improve recall on exact-match queries where embeddings sometimes miss.
- **A larger, more diverse test set** covering multi-document queries, ambiguous phrasings, and out-of-scope questions.

Conclusion

88% Precision@5 on retrieval, with the one miss attributable to the evaluation setup rather than a retrieval failure. The heading-aware chunking, consistent embedding space, and context-grounded generation at low temperature give engineers grounded, citable answers without needing to know which document to search or which keywords to use.