



International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Design and Implementation of a Citation-Grounded AI-Powered Document Learning Assistant Using Hybrid Retrieval

Arun Karthik V, Balaaditya B, Niranjan J

Department of Software Systems and AIML, Sri Krishna Arts and Science College, Coimbatore - 641008, India

ABSTRACT

Students increasingly study from heterogeneous digital material such as PDF lecture notes, Word handouts and spreadsheets, yet the tools available to them either rely on exact keyword search or on general-purpose chatbots that are not grounded in the student's own documents and can produce confident but incorrect answers. This paper presents DeepDocs, a browser-based study assistant that turns uploaded PDF, Word (.docx), Excel (.xlsx) and CSV files into an interactive, verifiable knowledge base. Documents are parsed by format-specific parsers behind a shared interface that preserves headings, page numbers and sheet names, split into overlapping chunks and embedded locally with the BAAI/bge-small-en-v1.5 sentence-embedding model. Questions are answered through a hybrid retrieval pipeline that combines BM25 keyword search with dense vector search and merges the two rankings using Reciprocal Rank Fusion; the fused passages are passed to the Gemini 2.5 Flash language model under a prompt that restricts it to the supplied evidence, requires numbered source citations and requires an explicit refusal when the answer is absent. On the same retrieval foundation, the system generates three-level summaries, multiple-choice quizzes and flashcards, concept mind maps, multi-document comparisons and a scaffolded Doubt Mode explanation, while caching generated material to minimise model calls. Unit, integration, functional, system, security and acceptance testing show that the system ingests all four formats, returns cited answers, declines unanswerable questions and runs on a standard 16 GB laptop without a GPU.

Keywords: Retrieval-Augmented Generation; Hybrid Retrieval; BM25; Sentence Embeddings; Reciprocal Rank Fusion; Large Language Models

1. Introduction

Students preparing for examinations or working through coursework face several recurring difficulties when studying directly from raw documents. Study material arrives in inconsistent formats, and no single everyday tool understands all of them uniformly. Locating a specific fact inside a long document usually means manual skimming or a plain keyword search, which fails whenever the wording of the question differs from the wording of the document. Writing summaries and revision notes by hand is slow, and self-testing requires students to write their own quiz questions, an effortful step that is frequently skipped, leaving gaps in understanding undetected until the examination itself.

General-purpose conversational assistants appear to solve these problems, but they are not grounded in the specific document a student is studying. Large language models are known to produce fluent yet unsupported statements, commonly called hallucinations (Ji et al., 2023), and an answer that cannot be traced back to a page in the course material is of limited value to a learner who must ultimately reproduce that material. Furthermore, when a student does not understand an explanation, most tools simply repeat the same answer rather than re-explaining it in a simpler, more scaffolded manner.

Retrieval-augmented generation (RAG) addresses the grounding problem by retrieving relevant passages from a document collection and conditioning the language model on them (Lewis et al., 2020). The availability of compact open-source embedding models, lightweight embedded vector databases and inexpensive hosted language models now makes it possible to build such a system without dedicated infrastructure. This paper describes DeepDocs, a RAG-based study assistant designed to run entirely on a personal laptop, using free and open-source components wherever possible and a free-tier cloud language model only for the final generation step.

The main contributions of this work are as follows:

- A multi-format ingestion pipeline in which PDF, Word, Excel and CSV parsers implement a shared interface and emit text, tables and structural metadata (headings, page numbers, sheet names) in one common internal representation.
- A hybrid retrieval engine that fuses BM25 keyword ranking with dense semantic ranking using Reciprocal Rank Fusion, so that both exact terminology and paraphrased questions are handled.
- A grounded generation layer that cites the filename, section heading and page of every supporting passage and explicitly declines questions that the uploaded documents cannot answer.

- A set of study tools built on the same evidence base: three-level summarisation, quiz and flashcard generation, concept mind maps, multi-document comparison and a scaffolded Doubt Mode.
- A cost-conscious design that embeds locally on the CPU and caches generated study material per document, keeping the system usable within free-tier API limits.

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the architecture and methodology of DeepDocs. Section 4 summarises the implementation and data design. Section 5 reports the testing process and results, Section 6 discusses the findings and limitations, and Section 7 concludes the paper and outlines future work.

2. Related work

2.1 Lexical and dense retrieval

Classical information retrieval ranks documents by term statistics. The BM25 function, derived from the probabilistic relevance framework, remains a strong baseline because it rewards rare query terms and saturates the contribution of repeated terms (Robertson & Zaragoza, 2009; Manning, Raghavan, & Schütze, 2008). Its weakness is vocabulary mismatch: a passage that expresses the same idea in different words receives no credit. Dense retrieval addresses this by encoding queries and passages into a shared vector space. Transformer encoders (Vaswani et al., 2017) fine-tuned as siamese networks produce sentence embeddings whose cosine similarity reflects semantic relatedness (Reimers & Gurevych, 2019), and dense passage retrieval has been shown to outperform BM25 on open-domain question answering (Karpukhin et al., 2020). Compact general-purpose embedding families such as BGE make dense retrieval practical on commodity hardware (Xiao, Liu, Zhang, & Muennighoff, 2023).

2.2 Rank fusion and retrieval-augmented generation

Lexical and dense retrievers make different errors, so combining them is attractive. Reciprocal Rank Fusion (RRF) merges ranked lists using only rank positions, requires no score normalisation or training, and has been shown to outperform individual rankers and more complex fusion methods (Cormack, Clarke, & Buettcher, 2009). Retrieval-augmented generation couples such a retriever with a generative model so that answers are conditioned on retrieved evidence rather than on parametric memory alone, improving factuality on knowledge-intensive tasks (Lewis et al., 2020). Grounding is also one of the principal mitigations for hallucination identified in the literature (Ji et al., 2023).

2.3 Limitations of existing study practice

In current practice, students read documents in separate PDF, Word or spreadsheet viewers, search with exact-match tools such as Ctrl+F, write summaries and quiz questions manually, and draw concept relationships on paper or in a separate diagramming tool. When they use a general-purpose chatbot, the assistant has no persistent knowledge of the document and offers no citations. Table 1 contrasts this practice with the capabilities targeted by DeepDocs.

Table 1 - Conventional study practice compared with DeepDocs.

Capability	Conventional practice	DeepDocs
Format handling	Separate viewer per format	One pipeline for PDF, DOCX, XLSX and CSV
Search	Exact keyword match only	Hybrid BM25 + semantic search with RRF
Answer reliability	Ungrounded chatbot output, no source	Answers restricted to retrieved passages, with citations and refusal
Summaries and self-testing	Written manually, often skipped	Generated in three levels; 5 MCQs and 5 flashcards with source pages
Concept relationships	Drawn separately from the document	Interactive mind map linked to source pages
Unclear explanations	Same answer repeated	Doubt Mode gives a scaffolded re-explanation
Cross-document analysis	Manual side-by-side reading	Automatic comparison of two or more documents

3. Proposed system and methodology

3.1 System architecture

DeepDocs follows a layered client-server architecture, shown in Fig. 1. The presentation layer is a React single-page application built with Vite, React Router and Tailwind CSS, with pages for sign-up and login, upload, question answering, summarisation, quizzes, mind maps and comparison. It

communicates with a Python FastAPI backend through a JSON REST interface, attaching a session token to every request. The backend exposes one router per feature and builds a shared application state at start-up that holds the long-lived services: authentication, ingestion, hybrid retrieval and generation. Persistent data is split between a SQLite database for users and cached study material and a persistent ChromaDB collection for document chunks and their embeddings. The embedding model runs locally on the CPU, and the Gemini 2.5 Flash model is called through the google-genai SDK only for the final natural-language generation step.

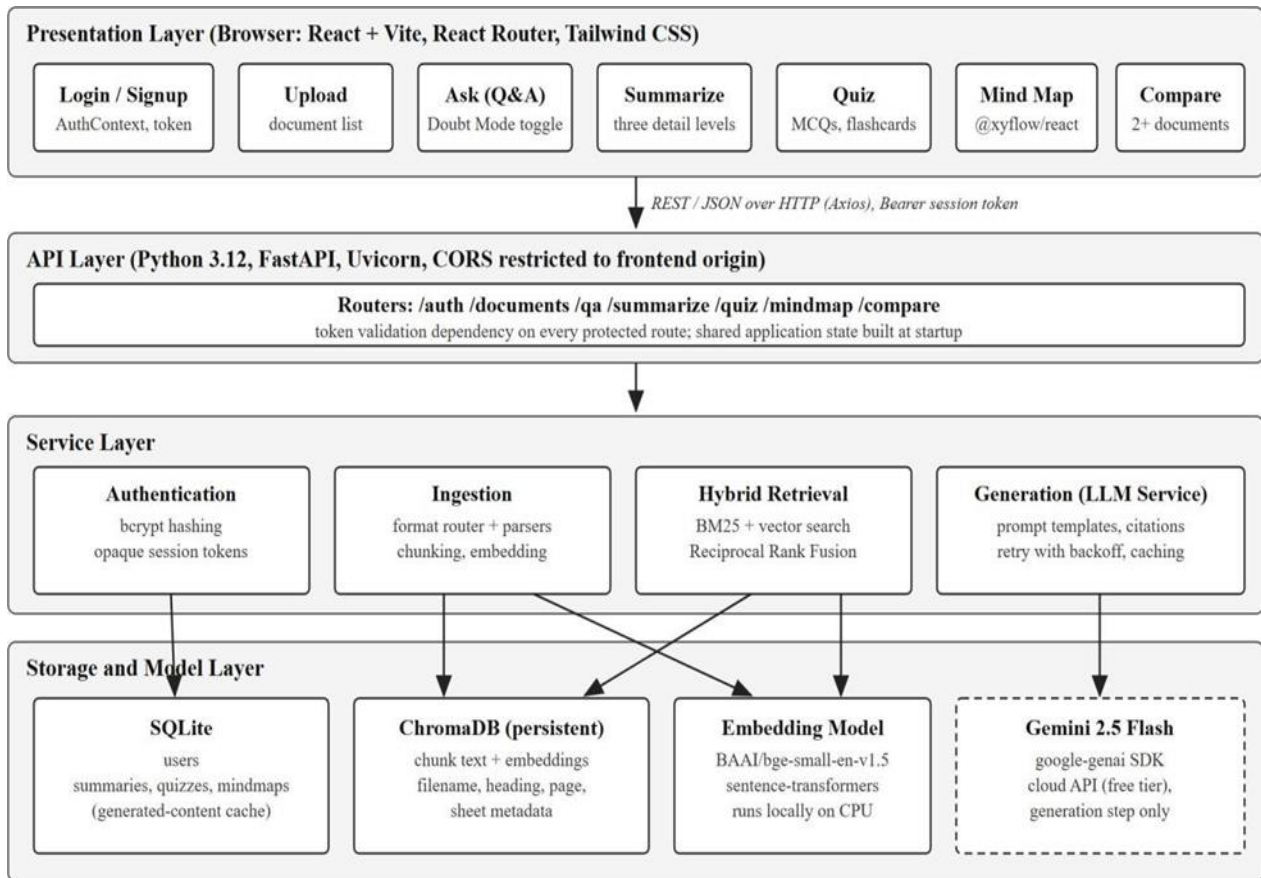


Fig. 1 - Layered architecture of DeepDocs.

3.2 Multi-format document ingestion

When a file is uploaded, a format router inspects its extension and dispatches it to a matching parser: PyMuPDF for PDF, python-docx for Word documents, and pandas with openpyxl for Excel workbooks and CSV files. Files of any other type are rejected. Every parser implements the same parsing interface and returns extracted text, tables and structural metadata in a common internal format, so that the downstream pipeline is independent of the source format and new formats can be added by implementing one additional parser. The preserved metadata (original filename, section heading, page number and, for spreadsheets, sheet name) is what later allows each answer and study artefact to be traced back to its exact location.

The extracted content is split into overlapping chunks so that sentences near a boundary remain available in context. Each chunk is encoded with the BAAI/bge-small-en-v1.5 model through the sentence-transformers library and stored in ChromaDB together with its metadata. In parallel, the chunk text is tokenised, stopwords are removed, and the tokens are added to an in-memory BM25 index. Both indices are updated immediately, so a newly uploaded document becomes searchable without restarting the server. The full ingestion and query flow is shown in Fig. 2.

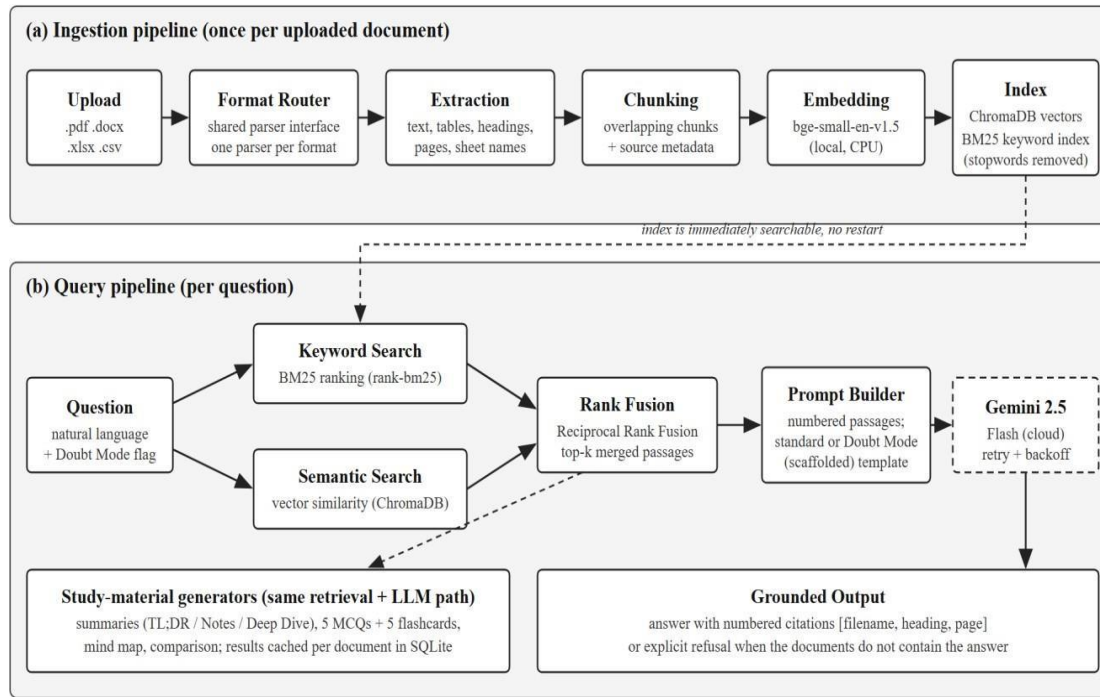


Fig. 2 - (a) Ingestion pipeline; (b) hybrid retrieval-augmented query pipeline.

3.3 Hybrid retrieval with Reciprocal Rank Fusion

A question is processed by two retrievers at the same time. The keyword retriever scores chunks with BM25 (Robertson & Zaragoza, 2009), which favours passages containing the exact technical terms of the question. The semantic retriever embeds the question with the same bge-small model and retrieves the nearest chunks from ChromaDB, capturing passages that express the same idea in different words. Because the two retrievers produce scores on incompatible scales, DeepDocs merges their result lists by rank position using Reciprocal Rank Fusion (Cormack et al., 2009):

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

where R is the set of rankings (here the BM25 list and the vector-search list), $\text{rank}_r(d)$ is the position of chunk d in ranking r , and k is a smoothing constant (60 in the original formulation) that limits the influence of any single top-ranked result. Chunks that appear high in both lists receive the largest fused score, and the top-ranked chunks form the evidence passed to the language model.

Unit testing of the keyword retriever exposed a practical issue: without stopword filtering, a query containing words such as "what" and "is" ranked an unrelated chunk highly simply because it contained those common words. Stopwords are therefore removed before indexing and before querying.

3.4 Grounded answer generation and Doubt Mode

The fused passages are numbered and assembled into a context block together with their filename, heading and page metadata. The prompt instructs Gemini 2.5 Flash to answer only from the supplied passages, to cite the numbers of the passages it used, and to state plainly that the uploaded documents do not contain the information when no passage supports an answer. The frontend renders the citations as source references (filename, section heading and page number), allowing the student to verify every claim. Calls to the model are wrapped with retry-with-backoff logic so that temporary rate-limit responses from the free-tier API do not surface as failures.

Doubt Mode is implemented as a toggle on the question-answering page rather than as a separate feature. The same retrieval pipeline is used, but the prompt is replaced by a scaffolded-explanation template that breaks the answer into simpler steps and ends with a short comprehension-check question. This gives the student a genuinely different explanation grounded in the same evidence.

3.5 Study-material generation

Several study tools are built on the stored document content:

- *Multi-level summarisation*: a TL;DR mode of five exam-oriented bullet points, a Notes mode organised heading by heading using the section metadata captured during ingestion, and a Deep Dive mode giving a longer, example-driven explanation.
- *Quiz and flashcards*: five multiple-choice questions, each with four options, a correct answer and a source page, and five front-and-back flashcards with source pages, returned as structured JSON and rendered with correct and incorrect answer highlighting.
- *Mind map*: a skeleton of nodes, one per section heading, is built locally from ingestion metadata without any model call; a single Gemini call then adds key concepts and the relationships between them. The node count is capped for readability, and the graph is rendered as an interactive, draggable and zoomable diagram with @xyflow/react, each node linking back to its source page.
- *Document comparison*: the stored content of two or more selected documents is retrieved and analysed together to produce a narrative of their shared content and their differences in content and coverage.

3.6 Caching and security

Generated summaries are cached in SQLite per document and per mode, and quizzes and mind maps are cached per document. Repeated requests, including switching back to a summary mode that has already been generated, are served from the cache without calling the language model, which keeps the interface responsive and the system within free-tier quotas. Passwords are hashed with bcrypt and never stored or returned in plain text. A successful login issues an opaque session token that must accompany every request to a protected endpoint, login failure messages do not reveal whether an email address is registered, and Cross-Origin Resource Sharing is restricted to the application's own frontend origin.

4. Implementation

The system was developed in Visual Studio Code with a Python virtual environment for the backend, Node.js and npm for the frontend, and Git for version control. Table 2 lists the technology stack. The system needs no GPU: the recommended configuration is a quad-core 2.5 GHz processor with 16 GB of RAM and an SSD, and the minimum is a dual-core 2.0 GHz processor with 8 GB of RAM. An internet connection is needed only for the Gemini API and the first download of the embedding model.

Table 2 - Technology stack of DeepDocs.

Layer	Technology	Role
Backend	Python 3.12, FastAPI, Uvicorn	REST API, request validation, application state
Document parsing	PyMuPDF, python-docx, pandas, openpyxl	Text, table and metadata extraction
Embeddings	sentence-transformers, BAAI/bge-small-en-v1.5	Local CPU sentence embeddings
Vector store	ChromaDB (persistent, embedded)	Chunk and embedding storage, similarity search
Keyword search	rank-bm25	BM25 ranking with stopword filtering
Relational store	SQLite	Users and generated-content cache
Authentication	bcrypt, opaque session tokens	Password hashing, protected endpoints
Language model	Gemini 2.5 Flash (google-genai SDK)	Answers, summaries, quizzes, mind-map relations
Frontend	React (Vite), React Router, Tailwind CSS, Axios	Single-page user interface
Visualisation	@xyflow/react	Interactive mind-map graph

The storage design is summarised in Table 3. A user may upload many documents, each document is split into many chunks, each document has at most one cached summary per mode, and each document has one cached quiz and one cached mind map.

Table 3 - Storage design.

Store / table	Principal fields	Purpose
users (SQLite)	id (PK), email (unique), password_hash, created_at	Account records with bcrypt hashes
chunks (ChromaDB)	chunk_id (PK), document_id, text, embedding, filename, Searchable chunks with source metadata heading, page_number, sheet_name	
summaries (SQLite)	document_id + mode (composite PK), content, created_at	Cached TL;DR, Notes and Deep Dive output

Store / table	Principal fields	Purpose
quizzes (SQLite)	document_id (PK), content (JSON), created_at	Cached MCQs and flashcards
mindmaps (SQLite)	document_id (PK), content (JSON), created_at	Cached graph nodes and edges

Input validation is enforced at the API boundary: email addresses must be well formed and unique, passwords must have at least eight characters, uploads must have one of the four supported extensions, questions must not be empty, the summary mode must be one of the three defined values, and the comparison action is enabled only when at least two documents are selected.

5. Testing and results

5.1 Testing methodology

DeepDocs was verified with black-box testing of every input form, white-box testing of the chunking algorithm, the BM25 tokeniser and the fusion logic, regression testing of the core upload, ask and summarise workflows after each new feature was added, and performance checks of the caching pipeline. Table 4 summarises the testing levels and their outcomes.

Table 4 - Summary of testing activities and outcomes.

Level	Scope	Outcome
Unit	Parsers, chunking, embedding, authentication, BM25, fusion, LLM retry logic	Passed; stopword distortion in BM25 found and fixed
Integration	Parse-chunk-embed-store pipeline, RRF merge, API routes and shared state	Passed; a circular import and a missing service registration corrected
Functional	All features exercised through the web interface with real documents	Passed; original-filename display bug found and fixed
System	Full session from sign-up to logout, cache reuse, simulated rate limits, Chrome and Edge	Passed; cached results returned without new model calls
Security	Missing and invalid tokens, password storage, login messages, CORS	Passed; no plain-text passwords, no account enumeration
Acceptance	Test users studying their own material with every feature	All stated objectives met

5.2 Results

Functional testing with real documents of all four formats confirmed that files are parsed, indexed and listed under their original filenames and become searchable immediately after upload. For questions whose answers were present in the uploaded material, the system returned correct answers with numbered citations identifying the source document, section heading and page. For questions whose answers were absent, the system stated that the documents did not contain the requested information instead of fabricating an answer. This refusal behaviour is the property that most distinguishes a grounded study assistant from a general-purpose chatbot.

All three summary modes produced output of the intended depth and structure, and switching between modes after the first generation returned the cached result instantly. Quiz generation consistently produced exactly five multiple-choice questions and five flashcards with source pages. Mind maps were built on the document's section headings, enriched with model-identified concepts and relationships, and stayed within the configured node limit. Doubt Mode returned simpler explanations ending in a comprehension-check question, and comparison of two and three documents identified both similarities and differences.

At system level, regenerating a summary, quiz or mind map for an already-processed document was served from the cache without a repeated model call, and simulated rate-limit responses from the Gemini API were retried with backoff until the request succeeded. Security tests confirmed that protected endpoints reject missing or invalid tokens, that passwords are never stored or returned in plain text, that login errors do not reveal whether an account exists, and that cross-origin requests are restricted to the application's frontend. Acceptance testing by a small group of users studying their own material confirmed that the system meets its objectives while running on a standard laptop.

6. Discussion and limitations

The results indicate that grounding, rather than model size, is the decisive factor for trustworthy study assistance. By restricting generation to retrieved passages and surfacing their provenance, DeepDocs lets a student verify every statement against the original page, and the explicit refusal path prevents the most harmful failure mode of an ungrounded assistant. Hybrid retrieval was important in practice: keyword ranking reliably surfaces passages containing exact technical terms, while semantic ranking recovers paraphrased content, and rank-based fusion combines them without tuning score scales. The stopword issue found during unit testing shows that classical lexical components still require careful preprocessing when they are combined with neural ones.

The design also shows that a useful RAG application need not depend on expensive infrastructure. Local CPU embeddings, an embedded vector database and a file-based relational store remove any need for separate servers, and per-document caching of generated material means each summary, quiz or mind map is paid for once. Reusing ingestion metadata, for example building the mind-map skeleton from section headings without a model call, further reduces model usage.

Several limitations remain. All authenticated users currently share one document pool rather than having per-user isolation. Session tokens are held in memory and are lost when the server restarts. Very large documents of several hundred pages have not been tested extensively and may require retrieval tuning. Scanned documents yield no extractable text because optical character recognition is not yet supported, and the evaluation reported here is functional and qualitative rather than a quantitative benchmark of retrieval accuracy.

7. Conclusion and future work

This paper presented DeepDocs, a hybrid retrieval-augmented study assistant that converts PDF, Word, Excel and CSV documents into a searchable, citation-backed knowledge base and builds summaries, quizzes, flashcards, concept maps, comparisons and scaffolded explanations on top of it. Combining format-aware parsing, local sentence embeddings, BM25 keyword search and Reciprocal Rank Fusion with an evidence-restricted prompt allows the system to answer from a student's own material, cite its sources and decline questions it cannot support. Its modular services make it straightforward to maintain and extend, and its caching strategy keeps it responsive and inexpensive on ordinary hardware.

Future work will add per-user document isolation and persistent sessions for multi-user deployment; OCR to support scanned handouts and photographed notes; parsers for additional formats such as slide decks; spaced-repetition scheduling of flashcards and adaptive quiz difficulty driven by a student's performance history; an analytics dashboard of study progress; containerised cloud deployment for access from any device; and a quantitative evaluation of retrieval and answer quality. Because the embedding and language-model services are isolated behind clear interfaces, newer models can be adopted with minimal change to the rest of the system.

Acknowledgements

The authors thank Arun Karthik V and the Department of Software Systems and AIML, Sri Krishna Arts and Science College, Coimbatore, for their guidance and support during this work, and the students who took part in acceptance testing.

8. References

- Cormack, G. V., Clarke, C. L. A., & Buettcher, 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). Boston, MA: ACM.
- FastAPI. (n.d.). *FastAPI documentation*. Retrieved from <https://fastapi.tiangolo.com/>
- Google. (n.d.). *Gemini API documentation*. Retrieved from <https://ai.google.dev/gemini-api/docs>
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), 1–38.
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769–6781).
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press.
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992).

-
- Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.
- Xiao, S., Liu, Z., Zhang, P., & Muennighoff, N. (2023). C-Pack: Packaged resources to advance general Chinese embedding. *arXiv preprint arXiv:2309.07597*.