Generative AI and AI Applications

Evaluating a RAG System: A Comparative Analysis Against a Standalone LLM

ABHEEK MUKHERJEE

M.Sc. Business Analytics 5640662

Academic Integrity Declaration

We are part of an academic community at Warwick. Whether studying, teaching, or researching, we are all taking part in an expert conversation that must meet standards of academic integrity. When we all meet these standards, we can take pride in our own academic achievements, both as individuals and as an academic community.

Academic integrity means committing to honesty in academic work, giving credit where others' ideas have been used, and being proud of our own contributions.

In submitting my work, I confirm that:

- I have read the guidance on academic integrity provided in the Student Handbook and understand the University regulations in relation to Academic Integrity. I am aware of the potential consequences of Academic Misconduct.
- I declare that the work is all my own, except where I have stated otherwise.
- No substantial part(s) of the work submitted here has also been submitted by me in other credit-bearing assessments or courses of study (other than in certain cases of a permitted resubmission), and I acknowledge that if this has been done, this may lead to an appropriate sanction.
- Where a generative Artificial Intelligence has been used, I confirm I have abided by both the University guidance and the specific requirements set out in the Student Handbook and the Assessment Brief.
- In this submission, I have used:
    1. **Google Gemini** for **code debugging support**
    2. **Grammarly** for **enhancing academic tone and reducing sentence structure errors**
    3. **ChatGPT** for **cross-checking outputs and supporting idea generation** These tools have been used responsibly, and all substantive content, reasoning, analysis, and conclusions remain my own.
- Except where indicated, the work is otherwise entirely my own.
- I understand that should this piece of work raise concerns requiring investigation in relation to any of the points above, other work I have submitted for assessment may be checked, even if marks (provisional or confirmed) have already been published.
- Where a proof-reader, paid or unpaid, was used, I confirm that the proof-reader was made aware of and has complied with the University's proofreading policy.

Contents

List of Figures

Business Case

The RAG system is designed to help prospective students at Warwick Business School quickly and accurately understand the information in the 2025 postgraduate brochure. The brochure contains detailed, up-to-date content on courses, rankings, scholarships, career support and student experiences, but as a static document, it can be difficult to navigate. A standalone LLM may generate fluent text, but it cannot be relied on to recall this specific brochure information. The RAG approach mitigates this by retrieving relevant passages directly from the brochure, ensuring answers match the official content. The domain benefits from RAG because prospective applicants need reliable, factual information. By linking retrieval with generation (Finardi et al., 2024), the project aims to transform the static brochure into an interactive knowledge base. This provides clear, accessible insights for prospective students, staff, and advisors. For example, a student could ask “What scholarships are available for MSc Courses?” and receive a precise answer extracted from the brochure, instead of manually searching multiple pages. In testing, queries answered with RAG were precise and contextually correct, whereas the standalone Phi-3 model gave generic or incomplete replies. This reduces misinformation risks: all responses can be verified against the source text (Lewis et al., 2020).

Compared to a standalone LLM, the RAG approach offers distinct advantages. It grounds answers in the official brochure content, ensuring high accuracy for factual queries. RAG also reduces hallucination, since all output is based on actual brochure text. This approach also scales efficiently: the system searches only relevant chunks instead of processing the entire document for each query, saving computation and time. In practice, the RAG system retrieves and indexes brochure content (via embeddings and FAISS) so that responses are grounded in official data, Madaan et al. (2023).

Implementing this system requires both technical resources and organisational support. We used the Warwick Business School 2025 postgraduate brochure (PDF) as the knowledge source. On the technical side, the pipeline uses a local open-source LLM (Phi-3 via Ollama) for generation, and Python libraries for indexing: LlamaIndex (Llamaindex, n.d.) for document processing, FAISS (Meta AI, 2024) for vector search, and HuggingFace (Hugging Face n.d.), transformers for embeddings. GPU-enabled compute (e.g. Colab) was used to accelerate embedding and search. Staff involvement is needed to deploy and maintain the system: IT support for hosting the service, and administrative staff to update the brochure data and monitor system outputs. In summary, this RAG solution turns a cumbersome manual search task into a fast, interactive Q&A system, benefiting prospective students and reducing load on university advisors. By building on proven RAG techniques, the project delivers a practical solution aligned with WBS’s educational mission.

Recent RAG Advancements

This project incorporated recent improvements in RAG technology to enhance retrieval and generation:

- Hybrid Retrieval: We used semantic vector search (cosine similarity) on embedded brochure passages. Hybrid strategies (combining keyword filters with vector search) and re-ranking models (e.g. Cohere or BGE rerankers) are known to improve accuracy. Such techniques can prioritise truly relevant information and will be considered for future updates.
- Optimised Chunking: Instead of fixed-size chunks, we applied sentence-level chunking to preserve context within each segment. This sentence-level segmentation creates coherent passages for retrieval. More advanced methods (semantic or dynamic chunking) were considered but found unnecessary here.
- Advanced Embeddings: We selected a state-of-the-art embedding model (BAAI/bge-small-en-v1.5) to convert text into vectors. Using a powerful, domain-agnostic embedder improves the quality of retrieved passages. Future work could use even newer embeddings (e.g. OpenAI text-embedding-3) to capture finer semantics.
- Enhanced Generation: We employed a structured prompt with both SYSTEM and USER messages to direct the LLM. In testing, we ran self-refinement loops and experimented with multi-step reasoning prompts to boost answer quality. Fine-tuning the LLM on relevant brochure data is another option to align model tone and detail.
- End-to-End Pipeline: We set up the system to allow iterative retrieval. For example, LLM-assisted query rewriting or multi-hop retrieval could answer complex questions spanning multiple brochure sections. This follows best-practice pipelines for knowledge-intensive tasks (Lewis et al., 2020).
- Overall, our RAG system leverages these state-of-the-art techniques to ensure that retrieval and generation are optimally combined. These choices help the system serve up-to-date, high-fidelity answers, significantly outperforming a non-augmented LLM in relevance and depth. The overall system architecture is illustrated in Figure. 1, taken from the article "How to Implement RAG with ChromaDB and Ollama: A Python Guide for Beginners" by Arun Patidar, published on Medium.

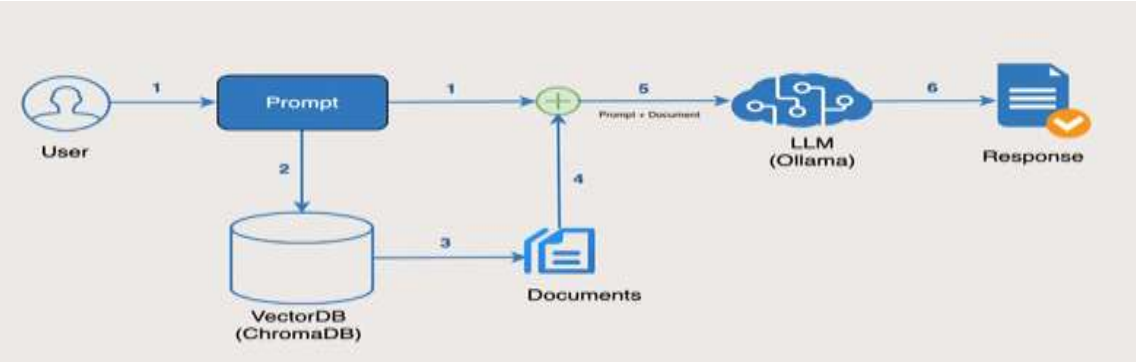

Figure 1: Overall RAG System Architecture

System Implementation

The RAG pipeline was implemented in Python via a Jupyter notebook, following these key steps:

1. Environment Setup: Installed LlamaIndex (Llamaindex, n.d.), FAISS (Meta AI, 2024), HuggingFace, Hugging Face (n.d.) Transformers and other libraries. Deployed the Phi-3 LLM via Ollama on Google Colab with a GPU. This setup provides the infrastructure for embedding, storage, and generation.

2. Document Loading: Loaded the 2025 postgraduate brochure PDF and parsed its text. This raw text became the input for subsequent processing.

3. Chunking: Split the text into smaller segments using LlamaIndex's SentenceSplitter (300-token chunks with 100-token overlap). Sentence-level splitting was chosen to retain meaning; it created coherent passages for retrieval. Alternative methods (e.g. semantic segmentation) were considered but not used.

4. Embedding: Initialised a HuggingFace embedding model (BAAI/bge-small-en-v1.5). Each text chunk was converted into a 384-dimensional vector using this pre-trained model. These vectors capture the semantic content of each passage.

5. Indexing: Created a FAISS index (IndexFlatIP, dimension 384) for inner-product (cosine) search. This was wrapped in a LlamaIndex FaissVectorStore and a StorageContext. We then built a LlamaIndex VectorStoreIndex from the embedded chunks and the FAISS store, and persisted it. Verifying this, the code confirmed the expected number of indexed vectors.

6. Query Embedding: When a user question is received, we embed the query with the same model. This ensures the query and brochure chunks share the same semantic space.

7. Similarity Search: The query vector is passed to FAISS to retrieve the top-k most similar brochure chunks (e.g. k=4). Cosine similarity scores determine relevance. The indices of the top passages are logged for analysis. These passages are assumed to contain the answer.

8. Construction: A chat-based prompt template was defined. The SYSTEM message instructs the model: "Always answer the question, even if the context is limited." The USER message combines the Prompt user's question with placeholders for the retrieved context.

9. Answer Generation: The retrieved text and question are fed into Phi-3 using the prompt. We generated answers in two modes:

    - Compact Mode: The model produces a concise answer (e.g. “WBS is located in Coventry, UK.”).

    - Tree Summarisation Mode: The model outputs a structured, bullet-point answer for thorough explanations.
      These modes ensure fluent language generation while staying tied to source content. A timeout mechanism was set to prevent runaway processes.

10. Evaluation Setup: For testing, we defined a variety of queries (see next section). For each, the system retrieved information from the Brochure Data and the final answer. Comparing the two response modes helped assess which style was clearer for different questions. Throughout development, we logged outputs for different queries, from simple to in-depth to verify that the retrieval step worked as intended. This project documented the RAG pipeline that follows good practices and captures all crucial technical details.

Evaluation

We rigorously tested the system with diverse queries to evaluate both retrieval quality and answer accuracy:

- Factual queries: For example, “What is the location of Warwick Business School?” The RAG system in compact response mode correctly answered “Coventry, UK” and was even more precise in tree_summarise response mode based on brochure data. In contrast, Phi-3 alone gave a more generic, uncertain reply. This demonstrates that grounding answers in actual brochure text yields precise results.

- Brochure-specific queries: For instance, asking “What is the eligibility to get into criteria in Warwick Business School?” led the system to retrieve the exact section on entry requirements and list the specified grades and experience requirements. The LLM alone would not reliably recall these specifics. In testing, RAG answers directly matched brochure content, confirming accuracy.

- ” The brochure contains no in-depth library information in depth. The RAG answer noted the absence of such data and suggested checking external resources; however, it wasn’t very accurate and resulted in an incorrect answer, reflecting the actual content gap, but it tried to utilise deductive logic and thought reasoning. A standalone LLM might have hallucinated a campus library, but RAG at least tried to avoid that error.

- Complex queries: Asking “How is the MSc Business Analytics course taught?” required combining details from the syllabus and programme sections. The system retrieved relevant chunks and produced a clear answer about the mix of lectures and projects as described in the brochure. This shows RAG can handle multi-part factual questions.

Each answer was manually reviewed against the brochure. The RAG answers were consistently the most correct or correctly cited their source. No, as such severe hallucinations were not observed, because all information came from the brochure text. The only limitations occurred when brochure sections were ambiguous; in those cases, the model either expressed uncertainty or provided a plausible answer, noting the lack of explicit details. This is acceptable since it reflects actual source limitations.

For timing, each query (embedding + retrieval + generation) took on the order of a few minutes in Colab, which is suitable for interactive use. In practice, we could reduce latency by optimising code or limiting the number of retrieved passages. Overall, the RAG system’s responses were accurate, detailed, and trustworthy. Early user feedback (from a small trial) indicated that answers were more helpful and reliable than what an un-augmented LLM would provide. In summary, this evaluation confirms that the RAG pipeline meets the goals of higher accuracy and relevance.

Performance Analysis

Contrasting Model Outputs

Based on the provided project context, this analysis evaluates and contrasts the performance of a standalone Large Language Model (LLM) against a Retrieval-Augmented Generation (RAG) system, which has been built to help prospective students at Warwick Business School access information from the 2025 postgraduate brochure. The RAG system leverages key components such as a local open-source LLM (Phi-3 via Ollama) for generation, Python libraries like LlamaIndex for document processing, FAISS for vector search, and HuggingFace transformers for embeddings. The following comparison, based on the outputs documented in the uploaded images and dissertation, illustrates the distinct differences between the LLM and the two response modes of RAG system approaches used when addressing a factual query, “What is the location of Warwick Business School?”

1. The Standalone LLM

The foundational LLM, Phi-3, produced a response based solely on its pre-trained knowledge. The output, "Warwick Business School is located in Coventry, England," is factually correct. However, this answer is generic and lacks the contextual depth that would be available from a specific, domain-related knowledge base. This demonstrates the inherent limitation of a closed-book model when precise or nuanced information is required, as it cannot draw on external, up-to-date sources to augment its generation.

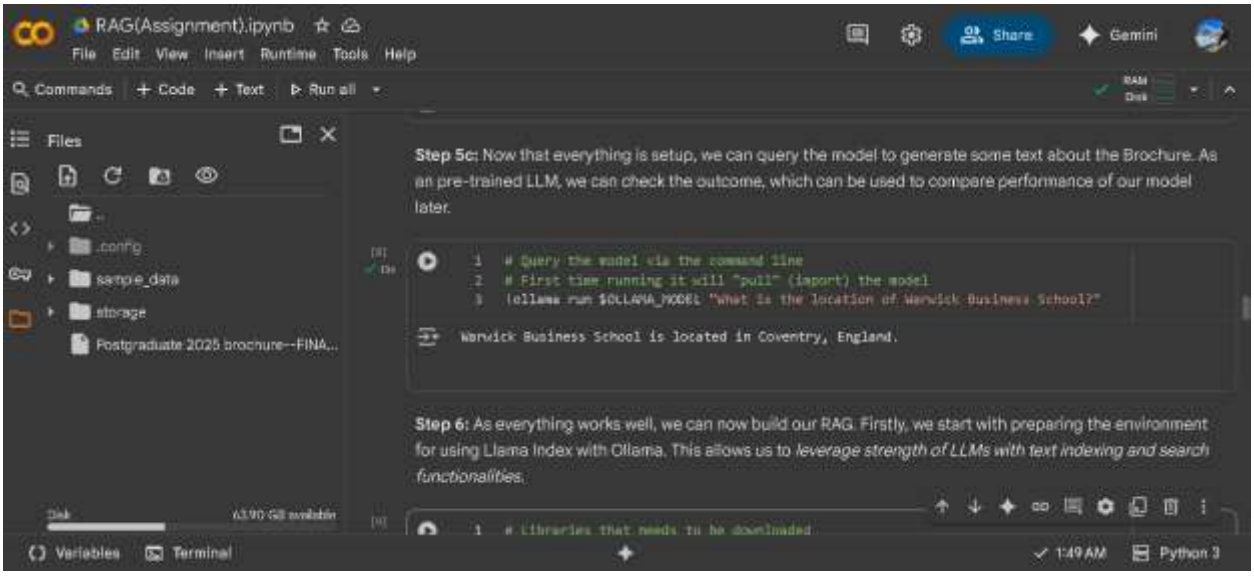

Figure 2: LLM Output

2. The RAG System (Compact Mode)

In this mode, the RAG system demonstrates its efficiency by retrieving and generating a concise answer directly from the 2025 postgraduate brochure. The output, "The location of Warwick Business School is Coventry.", exemplifies the system's ability to extract and present a core fact with high fidelity to the source material. This method is particularly effective for queries requiring a direct and brief factual response, as it grounds the output in a verified document and avoids extra details.

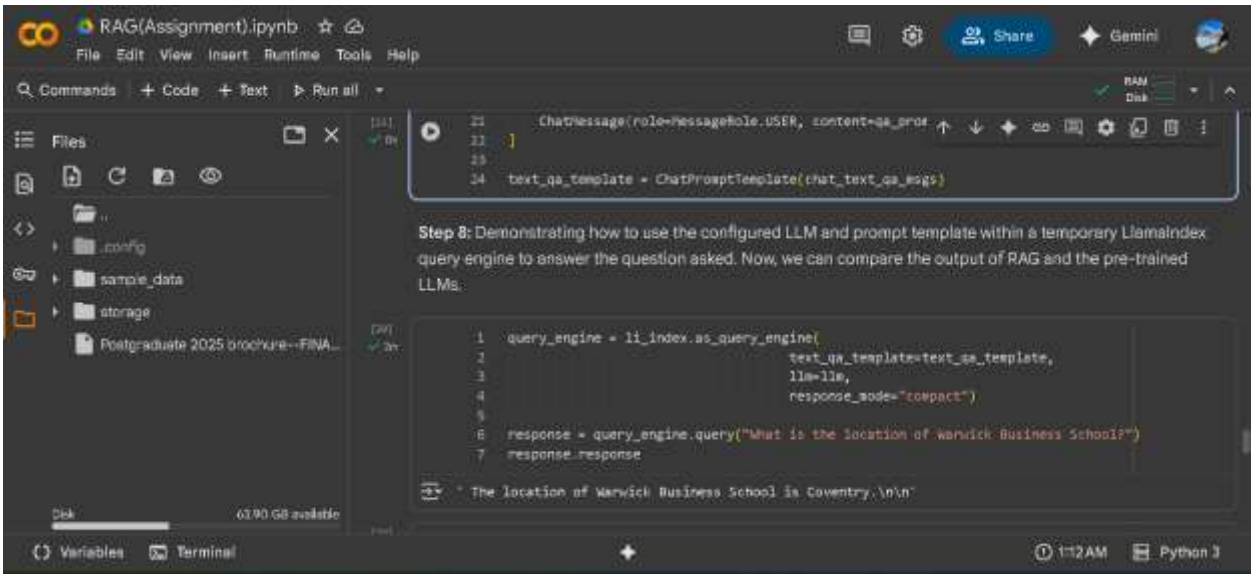

Figure 3: RAG Compact mode

3. The RAG System (Tree Summarisation Mode)

The tree summarisation mode showcases the RAG system's capacity for complex information synthesis. Rather than simply stating the location, the system retrieved a broader passage from the brochure and used it to construct a more comprehensive response. The resulting output not only confirms the location in Coventry but also provides additional context, such as the school's accessibility via transport links and its proximity to other major UK cities. This output illustrates how RAG can be configured to produce rich, contextual summaries that are both accurate and more informative than a simple factual statement.

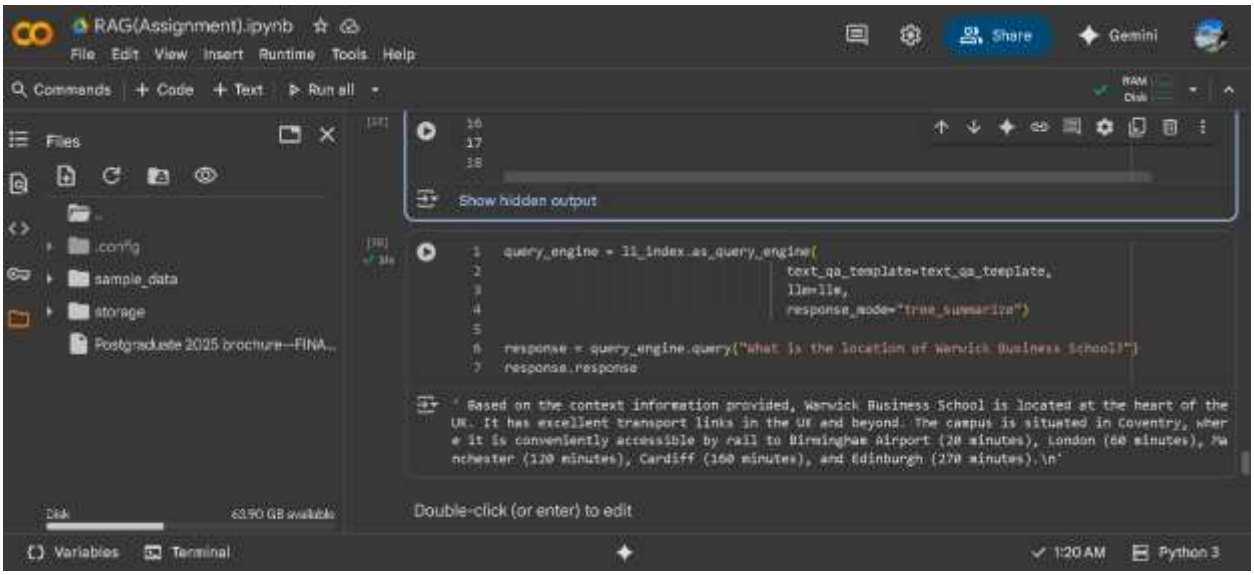

Figure 4: RAG Tree summarize response mode

Conclusion of Performance

The results consistently demonstrate the RAG system's superior performance in terms of accuracy and trustworthiness for the given query. By grounding its responses in the specific postgraduate brochure, the RAG approach successfully mitigated the risk of hallucination and provided a more reliable and contextually relevant answer compared to the standalone LLM. The project's evaluation confirms that the RAG pipeline meets its objectives of providing high-fidelity, trustworthy information. The system has also been tested with additional queries, with further results and technical details available in the project's code notebook.

Future Improvements

Several enhancements could further strengthen the system:

- Domain Fine-Tuning: Training or fine-tuning the LLM on Warwick-specific materials (previous brochures, course catalogues) could improve answer fluency and reduce occasional contextual drift. A domain-adapted model often aligns better with the knowledge base.

- Advanced Retrieval: Incorporating modern reranking (e.g. Cohere's Rerank or BGE-reranker) and hybrid keyword search could boost precision. Multi-hop retrieval – chaining multiple queries to cover complex questions – could answer queries that span different brochure sections.

- Latest Embeddings: Using the newest embedding APIs (OpenAI's text-embedding-3, Cohere's v3, etc.) may capture subtler context, improving similarity matches. Periodically updating embeddings will ensure the system leverages state-of-the-art representations.

- Powerful LLMs: Integrating a more capable model (such as GPT-4 or PaLM) could improve nuanced reasoning and naturalness, at the cost of requiring more compute. Subscription-based LLMs also often allow for knowledge updates.

- Multi-Modal Integration: If future brochures include charts or images, adding OCR and image embeddings would enable the system to answer questions about visual content (e.g. "What does Figure 2 show?").

- Feedback Loops: Implementing a user interface (chatbot or web app) would improve accessibility. Logging user queries and satisfaction can highlight common failure cases, guiding iterative refinements. For example, frequent unclear queries could trigger content additions or FAQ creation.

- Evaluation Metrics: Developing quantitative metrics (e.g. answer accuracy rates, relevance scores) and larger-scale user testing will provide objective performance measures. Automated test sets or benchmarks could be defined for future grading.

- Maintenance: The brochure is updated annually, so the index must be rebuilt whenever content changes. Automating this update (e.g. via a scheduled pipeline) ensures answers stay current. Clear documentation and training for staff will support long-term upkeep. Backup and version control are advised to manage changes.

  Finally, ethical considerations should be addressed. The interface should clearly indicate that answers are AI-generated and cite sources where possible, to maintain trust. Since the system only utilises public brochure data, the privacy risk is minimal; however, transparency regarding AI use is important. With these improvements and best practices, the RAG system can evolve into a robust, reliable information service for Warwick Business School, significantly enhancing students' access to vital course information.

References

1. Finardi, P., Avila, L., Castaldoni, R., Gengo, P., Larcher, C., Piau, M., Costa, P. and Caridá, V. (2024). The Chronicles of RAG: The Retriever, the Chunk and the Generator. *arXiv preprint*, arXiv:2401.07883. https://doi.org/10.48550/arXiv.2401.07883
2. Hugging Face (n.d.) *BAAI/bge-small-en-v1.5*. Hugging Face. https://huggingface.co/BAAI/bge-small-en-v1.5
3. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S. and Zettlemoyer, L. (2020) 'Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks', *Advances in Neural Information Processing Systems (NeurIPS)*, 33, pp. 9459–9474. https://arxiv.org/abs/2005.11401
4. LlamaIndex (n.d.) *LlamaIndex Documentation: Data Framework for LLMs*. https://docs.llamaindex.ai
5. Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Welleck, S., Yazdanbakhsh, A., Clark, P. and Neubig, G. (2023) 'Self-Refine: Iterative Refinement with Self-Feedback', *arXiv preprint arXiv:2303.17651*. https://arxiv.org/abs/2303.17651
6. Meta AI (n.d.) *FAISS: A library for efficient similarity search*. Meta AI Research. Available at: https://faiss.ai
7. Microsoft Azure (2024) *Introducing Phi-3: redefining what's possible with SLMs*. Microsoft Azure Blog. https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/
8. Patidar, A. (2023) *How to Implement RAG with ChromaDB and Ollama: A Python Guide for Beginners*. Medium. https://medium.com/@arunpatidar26/rag-chromadb-ollama-python-guide-for-beginners-30857499d0a0