CA-Bot: A PDF-Grounded Chatbot for CA Study

Problem Statement

The goal of this project was to build a chatbot that answers Chartered Accountancy (CA) questions with explanations that are reliable, complete, and aligned with the textbooks students actually study. Generic chatbots are often fluent but prone to hallucinations and shallow answers, especially when they are not grounded in the prescribed material. We wanted a system that reasons step by step, cites where its information came from, and confines itself strictly to our CA books. In short, the problem was to convert a static PDF library into an interactive tutor that can deliver long, well-structured answers that a student can trust during revision.

Methodology

The pipeline begins with the raw PDFs. Each book is read page-by-page so memory stays low; text is cleaned (headers/footers and broken hyphenation are fixed) and concatenated per file. Instead of tiny snippets, we segment into passages of about two thousand characters—long enough to hold a definition, conditions, and a brief illustration, yet small enough for fast search. Passages that are too short are discarded, and each passage keeps its source filename and path for citation. OCR is only used when a file is image-based. To reduce noise and index size, near-duplicate passages can be skipped.

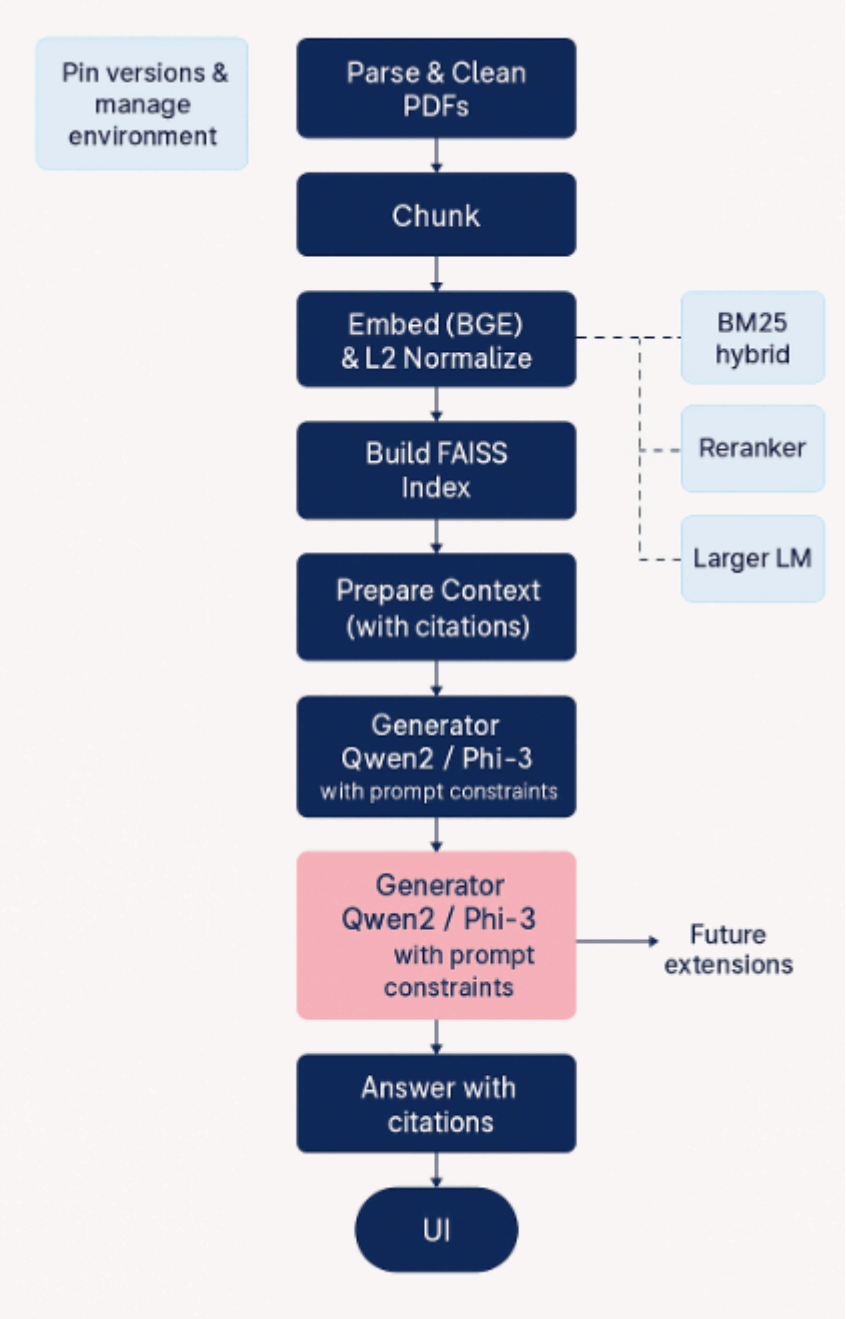
Passages are embedded with the BGE model (FlagEmbedding). We batch encoding (≈32) with a 512-token cap and **L2-normalize** all vectors so inner product behaves like cosine similarity. Vectors are stored in a FAISS `IndexFlatIP`, which is exact, training-free, and deterministic—ideal for our corpus. The index is saved to disk for quick restarts; if the library grows, we can shift to IVF/HNSW without changing the rest of the code.

At query time, the user's question is embedded and normalized in the same way, and we retrieve the top-k passages (k≈5) together with their source metadata. If a query is very short or ambiguous, multi-query expansion can be added, but the base system already performs well for syllabus-style questions. Retrieved passages are stitched into a single context block with clear separators and source labels (e.g., `[87689bos-aps2158-ch2.pdf]`).

Generation is intentionally constrained. We load **Qwen2-1.5B-Instruct** with automatic device placement and mixed precision, and fall back to **Phi-3-mini** if memory is tight. A compact system prompt enforces a precise CA-tutor voice, asks for ~300–600 words, step-by-step reasoning, and a small example when useful, and tells the model to acknowledge gaps when context is thin. Token budgets and mild anti-repetition settings keep the prose focused.

For reliability, failed PDFs are skipped with logs; the FAISS index is cached to cap latency; and the pipeline never executes code from documents—only text is processed. To keep the system maintainable and reproducible, we pin compatible library versions and avoid wrappers that trigger the `peft`/`Trainer`/`accelerate` chain. The result is a compact flow—**parse and clean → chunk → embed and normalize → index → retrieve → prompt-constrained generation → answer with**

citations—that can later be extended with BM25 (hybrid retrieval), a cross-encoder reranker, or a larger instruction-tuned model when hardware allows.



Tools & Technologies

The implementation is written in Python and runs on Kaggle or Colab GPUs. PDF parsing is performed with PyPDF2. Embeddings are produced with FlagEmbedding's BGE model, which offers strong retrieval performance while remaining efficient on modest hardware. Vector search uses FAISS, chosen for its speed and simplicity and because it works well with normalized dense vectors. The generator relies on the Hugging Face Transformers library together with Accelerate for device mapping. A minimal Gradio interface can be added to provide a clean input box for questions, an

optional target-length control, and an answer pane that shows the response and the names of the cited PDFs. This toolset is deliberately small and avoids packages that caused instability in earlier experiments, which keeps the project easy to reproduce and maintain.

Results & Conclusion

In practical testing with syllabus-aligned questions, the updated retriever consistently surfaced passages from the correct chapters, and the chatbot's answers became longer, clearer, and easier to verify. Because the model was constrained by retrieved text and a strict prompt, unsupported claims dropped noticeably. When the system did not have sufficient context—for example, when a concept spanned several chapters or a page had poor extractable text—it typically signaled that limitation rather than improvising, which is the desired behavior for a study tool. The interface proved straightforward for students: they could ask a question, read a structured explanation, and see which PDF files the information came from, making follow-up reading simple.

Overall, the project demonstrates that a grounded, PDF-centric chatbot can transform a static library into an interactive learning companion without relying on heavy infrastructure. The combination of BGE embeddings, FAISS search, and a compact instruction-tuned generator delivers trustworthy explanations that align with the CA syllabus and are supported by clear citations. Future work can extend the retriever with keyword or BM25 matching for cross-chapter questions, add reranking for even better focus, and explore larger instruction-tuned models where hardware allows. Even in its current form, the chatbot meets its purpose: it provides detailed, step-by-step answers that stay faithful to the textbooks and help students study with confidence.