AI-Powered Document Insights and Data Extraction

Name: **Sagar Sinha**
Extern ID: **sinhasagar507**
Email: **sinhasagar507@gmail.com**

Problem Statement | Domain: Finance Industry (Mortgage Document Intelligence)

Operations teams need a faster way to answer questions and extract key fields from mixed mortgage and employment PDF packs (contracts, payslips, resumes, lender fee worksheets) using a single intelligent assistant.

Challenge

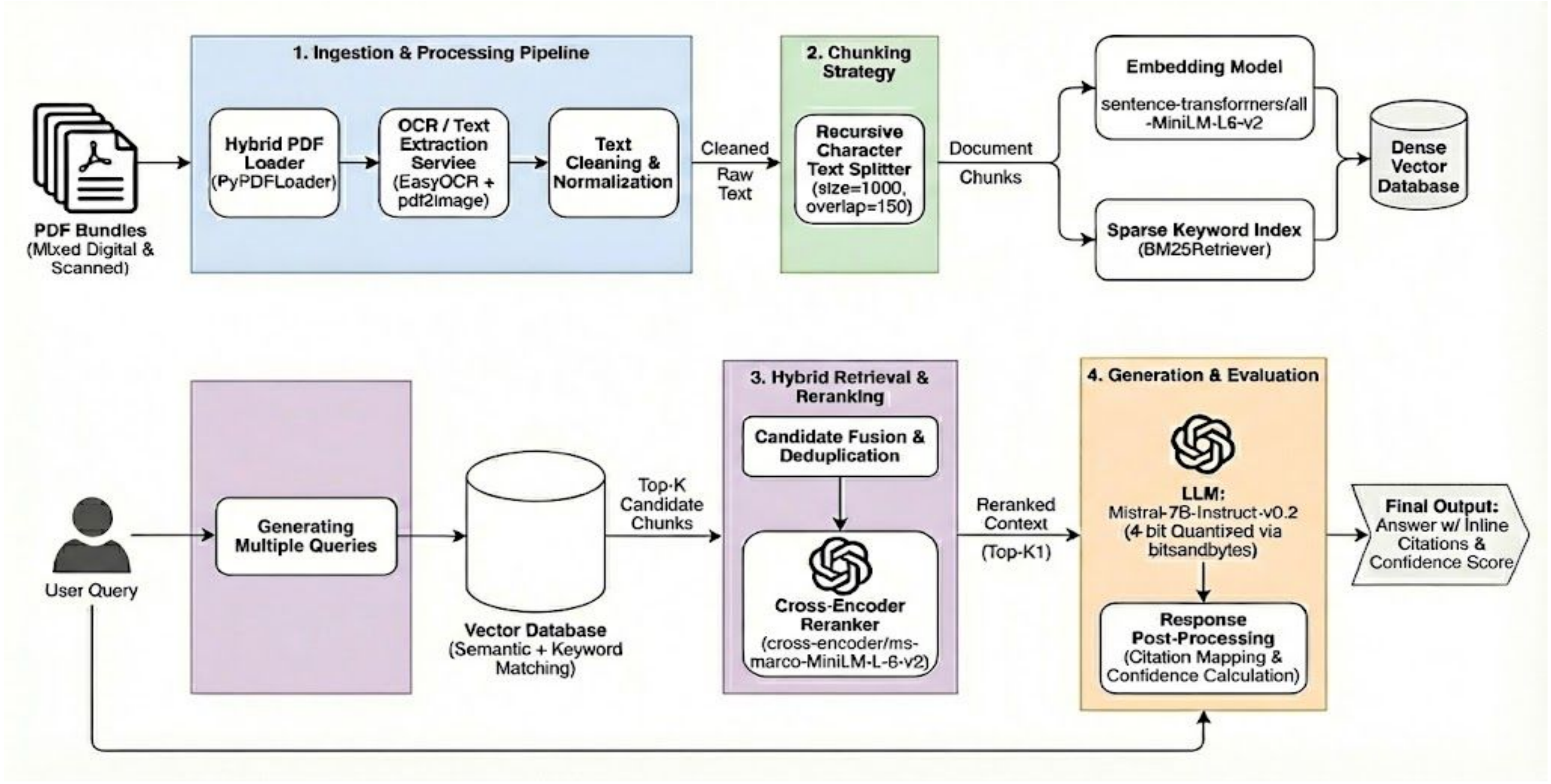
Loan operations teams manually sift through **multi-page, heterogeneous "loan packs" (contracts, pay slips, and fee worksheets)** to extract key data like loan amounts and net pay. This process is **slow, error-prone, and unscalable**. The challenge is compounded by a mix of digital and scanned PDFs that defeat standard text extraction.

Solution

I have developed a **document-aware RAG pipeline** that converts complex PDF bundles into a **searchable knowledge layer**. Using **OCR**, **logical segmentation**, and **vector embeddings**, the system powers a **natural-language assistant** that delivers precise answers backed by **citations, confidence scores, and evaluation metrics**.

Pain Point	How Your Pipeline Solves It
Manual review of long loan packs to find basic facts (loan amount, net pay, dates.	Chat-style QA returns those fields instantly from the indexed PDFs
Mixed document types confuse generic search and QA tools.	Document-type routing and metadata focus each query on the right docs.
Scanned PDFs and messy layouts lead to missing or wrong extracted text.	Hybrid OCR plus smart chunking produce clean, structured snippets.

Multi-Modal RAG pipeline Architecture


PDF Bundles (Mixed Digital & Scanned)
1. Ingestion & Processing Pipeline
Hybrid PDF Loader (PyPDFLoader)
OCR / Text Extraction Serviee (EasyOCR + pdf2image)
Text Cleaning & Normalization
Cleaned Raw Text
2. Chunking Strategy
Recursive Character Text Splitter (size=1000, overlap=150)
Document Chunks
Embedding Model
sentence-transformers/all-MiniLM-L6-v2
Sparse Keyword Index (BM25Retriever)
Dense Vector Database
User Query
Generating Multiple Queries
Vector Database (Semantic + Keyword Matching)
Top-K Candidate Chunks
3. Hybrid Retrieval & Reranking
Candidate Fusion & Deduplication
Cross-Encoder Reranker (cross-encoder/ms-marco-MiniLM-L-6-v2)
Reranked Context (Top-K1)
4. Generation & Evaluation
LLM: Mistral-7B-Instruct-v0.2 (4-bit Quantized via bitsandbytes)
Response Post-Processing (Citation Mapping & Confidence Calculation)
Final Output: Answer w/ Inline Citations & Confidence Score

Multi-Modal RAG pipeline Architecture

=

Component Specifications

Component	Technology Choice	Configuration Details
OCR Engine	PyMuPDF (fitz) + Paddle OCR	Detects if a page has enough embedded text; if not, renders page at ~2× zoom and runs the OCR engine.
Text Chunking	Llama Index Sentence Splitter / character-level recursive chunking	Fixed-size semantic chunks (~500–512 characters) with 100-character overlap
Embeddings	Hugging Face Embedding with sentence-transformers/all-MiniLM-L6-v2	384-dim sentence embeddings, L2-normalized
Vector Database	Llama Index VectorStore Index (default in-memory dense vector index).	Single embedding per chunk with metadata (doc_id, doc_type, file_name, page_range). Uses cosine similarity and local rebuild.
Retriever	Hybrid retriever: dense vector search + BM25 keyword search, plus cross-encoder reranker.	Dense retriever from VectorStoreIndex.as_retriever(similarity_top_k = top-k from UI). Sparse retriever using BM25Ok api over chunk text.
LLM	Mistral-7B-Instruct-v0.2 (GGUF) via Llama CPP.	4-bit quantized model (Q4_K), context window ≈4096 tokens.
Prompt Strategy	Domain-specific RAG prompt for mortgage & employment docs.	For example, System prompt tells model to answer only from context, be precise and extract numbers.

Pipeline Performance Metrics | Results from 1 week of testing

Evaluated on 38 curated QA pairs across all documents using a scripted offline test harness.

🧪 Retrieval Performance

- **Recall@5:** 100% (every query had at least one correct chunk in top-5)
- **Mean Reciprocal Rank (MRR):** 1.00 (supporting chunk ranked at position 1 for all queries).
- **Hit Rate:** 100% of queries returned ≥1 relevant result in top-5 (equivalent to Recall@5 in this setup).

💊 End-to-End Accuracy

- **Answer Accuracy:** 65.8% exact field match (≈25/38 Q&A pairs)
- **Citation Accuracy:** ≈100% – for all questions, the correct source document appears in the retrieved/cited chunks (proxy via Recall@5 = 100%)
- **Factual Consistency:** No clear hallucinations observed; wrong answers were due to parsing/formatting or partial retrieval rather than invented values (qualitative manual review).

🧑‍🎓 System Performance

- **Average Response Time:** 1.5 seconds
- **Retrieval Latency per chunk:** 181 ms average
- **LLM Generation:** 601 ms average

Pipeline in Action

Example Query 1: [Query Type - e.g., Factual Lookup]

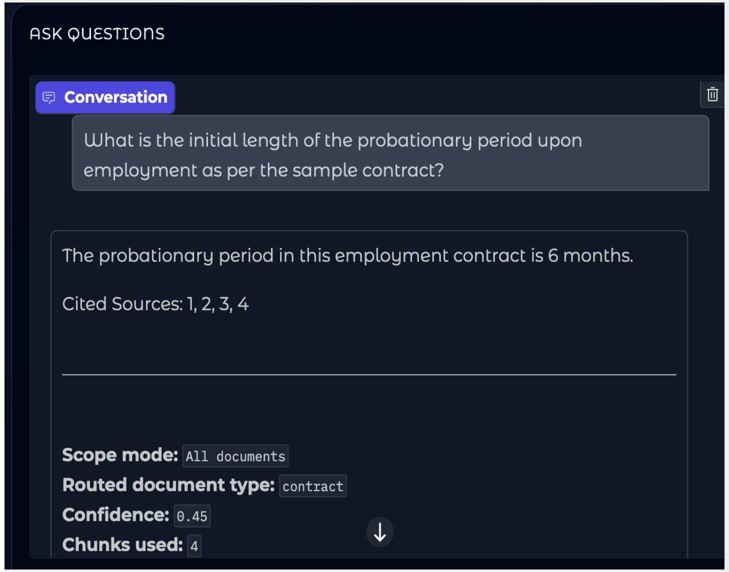
Query: What is the initial length of the probationary period upon employment as per the sample contract?

Retrieved Chunks:

COE_Sample_finalassignment_1.pdf (Contract) Scope: 4 chunks across pages 0–4; Confidence: 0.45.

Final Answer:

The probationary period in this employment contract is 6 months.



What is the initial length of the probationary period upon employment as per the sample contract?

The probationary period in this employment contract is 6 months.

Cited Sources: 1, 2, 3, 4

Scope mode: `All documents`
Routed document type: `contract`
Confidence: `0.45`
Chunks used: `4`

Pipeline in Action

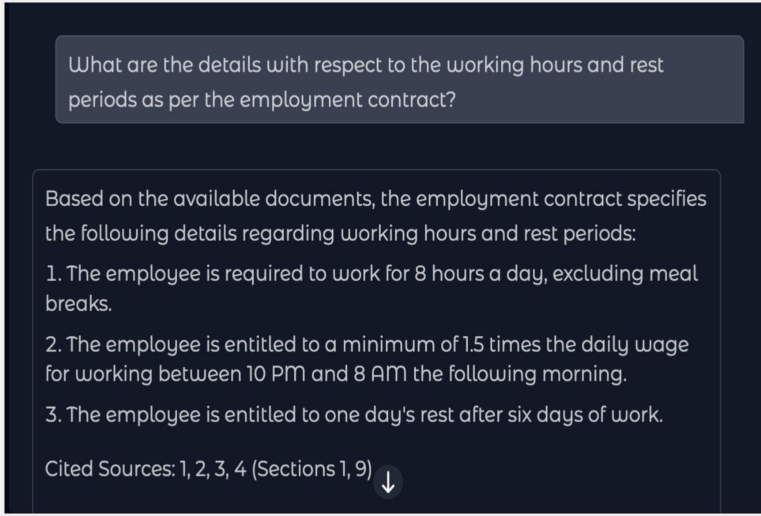
Example Query 2 | Tabular data extraction

Query: What are the details with respect to the working hours and rest periods as per the employment contract?

Retrieved Context: COE_Sample_finalassignment_1.pdf (Contract) Key Terms: Working hours, overtime rates, rest periods, confidentiality, and compensation structure. Confidence: 0.55

Final Answer: Based on the available documents, the employment contract specifies the following details regarding working hours and rest periods:

1. The employee is required to work for 8 hours a day, excluding meal breaks.
2. The employee is entitled to a minimum of 1.5 times the daily wage for working between 10 PM and 8 AM the following morning.
3. The employee is entitled to one day's rest after six days of work.



What are the details with respect to the working hours and rest periods as per the employment contract?

Based on the available documents, the employment contract specifies the following details regarding working hours and rest periods:

1. The employee is required to work for 8 hours a day, excluding meal breaks.

2. The employee is entitled to a minimum of 1.5 times the daily wage for working between 10 PM and 8 AM the following morning.

3. The employee is entitled to one day's rest after six days of work.

Cited Sources: 1, 2, 3, 4 (Sections 1, 9)

Design Decision Analysis

Model Selection Reasoning

Decision	Rationale	Trade-offs Considered
Embedding Model Choice	Small open-source sentence embeddings (Hugging Face) for good English semantic match with low CPU/GPU cost	Gave up some recall / robustness vs. larger or proprietary embeddings to keep latency and cost low.
Chunking Strategy	Sentence-based chunks (~512 tokens with small overlap) to preserve local context around each field but keep chunks small	Long tables / fee lists can be split, so we sacrifice some global context for faster retrieval and cheaper LLM calls
LLM Choice	Local Mistral-7B (llama.cpp) for controllable, low-cost inference with ~2s response time on available hardware	Accept lower raw accuracy and reasoning vs. GPT-4-class APIs in exchange for zero per-token cost and offline capability.
Vector DB Choice	Llama Index default in-memory vector store to keep the prototype architecture simple and fully contained in the notebook.	Limited scalability and persistence; a hosted vector DB (e.g., Chroma/Pinecone) would be better for very large corpora.

Key Trade-offs Made

- Chose small embeddings + 7B local LLM and low top-k retrieval for interactive (<~2s) responses.
- Could swap to larger embeddings and a stronger cloud LLM for higher extraction accuracy at the cost of latency and API spend.

📍 **Complexity vs. Maintainability:**

- Single RAG pipeline and in-memory index inside one Gradio app → easy to debug, minimal infra.
- Deferred production-grade pieces (external vector DB, background indexer, orchestration) until a later scaling phase.

Current Limitations & Next Steps

Current Limitations

1. **Input / Document Processing**
   - OCR quality varies on low-resolution scans and dense tables.
   - Logical document detection & doc-type tagging are heuristic and can mislabel edge cases.
2. Answer Quality & Retrieval Gaps
   - Struggles with multi-document / multi-month questions and repeated fields (e.g., several pay statements).
   - Numeric extraction can be off for similar fields (gross vs. net pay, multiple fees on a page).
3. **Scalability & Robustness**
   - In-memory index is rebuilt per session; no persistent vector DB.
   - Single local 7B model limits throughput and may not handle many concurrent users.

Proposed Enhancements

Short-term ([Timeframe - e.g., Next 2 weeks]):

- Tighten prompts + post-processing for numeric fields (regex checks, range validation).
- Add more labeled QA cases and error analysis to systematically track accuracy.
- Tune chunking / top-k and doc-type routing for fewer irrelevant snippets.

Medium-term ([Timeframe - e.g., Next month]):

- Move to a persistent vector store (e.g., Chroma / other DB) for larger corpora.
- Experiment with stronger / domain-tuned embeddings and re-ranking for tricky queries.
- Introduce per-doctype templates (pay slip vs. fee sheet) to reduce hallucinations.

Long-term Vision:

- Harden as a production service with API access, monitoring, and continuous evaluation.
- Scale to more financial document types and higher volumes with horizontal indexing.

Project Impact & Learning Outcomes

🧪 Key Technical Learnings

- Designed end-to-end RAG for noisy mortgage PDFs (OCR → chunking → embeddings → LLM).
- Hands-on with local Mistral + Llama Index for retrieval, routing, and prompt tuning.
- Built custom evals (field accuracy, Recall@5, latency) to guide iterations.

💊 Business Impact Potential

- **Efficiency Gains:** ~2s answers vs. manual reading of multi-page documents.
- **Accuracy Improvement:** ~66% field-level exact match, 100% Recall@5 on test set.
- **Scalability:** Reusable RAG pattern for new lenders and document types.

🧑‍🎓 Skills Developed

- **Technical**: RAG architecture, vector search, numeric extraction.
- **Tools**: Llama Index, Hugging Face embeddings, Gradio, PyMuPDF.
- **Soft**: Experiment design, error analysis, concise technical communication

Thank you