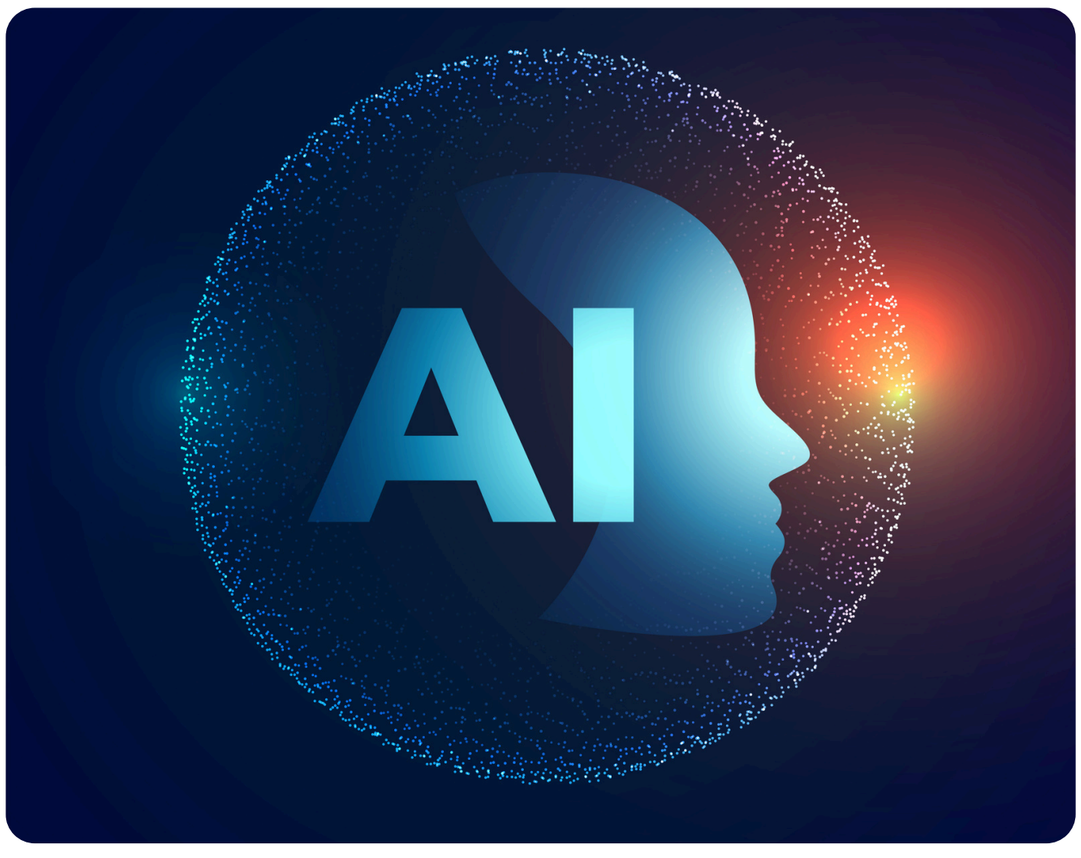
CaliberCode

CASESTUDY

CalibreCode's QA for LLM-Based Legal Document Intelligence



Client Objective:

A legal-tech SaaS firm was developing an AI assistant to extract structured data: names, dates, financials, and ownership history from scanned, handwritten, or inconsistently formatted legal documents such as house agreements and title deeds.

The system had to ensure accuracy, traceability, and compliance. The client sought a QA partner with strong LLM experience, process rigour, and domain-aware testing strategies.

CalibreCode's QA Strategy: Beyond Classical Testing

LLM-based document intelligence is non-linear. The flow involves:
User Input → Pre-processing → Prompt Construction → LLM Response → Post-processing → Display

Our QA team built a multi-layered validation framework covering:

- **Prompt QA**: Static prompts tested with rule-based updates, evolving toward dynamic adaptability.
- **Chunking & RAG Pipelines**: Validated context segmentation and retrieval strategies.
- **Human-in-the-Loop Reviews:** BA + QA sign-offs at every pipeline iteration.
- **Drift Monitoring:** Ensured model output stability across documents, formats, and prompt changes.
- **Security Compliance:** All tests used redacted real data

QA was involved **early** and **continuously**, validating both positive cases (correct extractions) and negatives (missed or incorrect data). Each **correction** triggered a full re-validation cycle, including **architectural reviews** for prompt updates, ensuring **systemic integrity.**

CalibreCode

Use Case 1: Extracting Buyer-Seller Details from Agreements

Goal

- Names of buyer and seller
- Agreement date
- Property address
- Sale value

Challenges

- Name casing ("ALL CAPS")
- Legal date phrasing
- Non-deterministic LLM retries

QA Actions

- Annotated 100 golden examples (client-provided) using Label Studio
- Side-by-side human vs. model comparisons
- Static prompts tuned for layout variations
- Scoring: Exact match, similarity, and recall
- Scripted drift tracking across scans, PDFs, and digital docs

Outcome

- **Hallucination rate dropped from 18% → <4%**
- **92% accuracy achieved**
- **Fallback logic introduced (retry+rule based)**

CalibreCode

Use Case 2: Clause & History Extraction

Goal

- Identify clause types: conditions, indemnities, governing laws
- Trace property ownership lineage
- Surface missing documents and clause cross-references

Challenges

- Narrative-style history descriptions
- Clause references (“see 4b”)
- Extraction ambiguity for complex clause

QA Actions

- Chunk-aware testing for boundary handling
- Validate accuracy (initial reviews)
- Injected adversarial inputs (missing headers, vague clauses)

Outcome

- **Retrieval memory enabled clause lineage tracing**
- **12 hallucinated clauses caught in QA**
- **Strengthened chunking logic with overlaps**

CalibreCode

Testing Methodology & Toolchain

Manual Validation

- Three-tier rubric:
  ✅ Complete | ⚠️ Partial | ❌ Incorrect
- Manual review confirmed by BA, then QA
- Document UI auto-scroll enabled precise side-by-side validation

Automation + Tracking

- Scripts generated for score AI responses, match, recall, and clause-level accuracy
- Prompt drift tracked across runs
- Regression triggered for every model/prompt change

Tool	Role
Label Studio	Clause annotation, golden set creation
Python/Pandas	Metric scripts, dashboards
Sheets/Excel	Manual review logs, audit trail

Results & Business Impact

3× model stability across formats

40% drop in missed clauses via chunk-overlap QA

3× faster legal review thanks to reliable outputs

Audit-grade QA logs aligned with regulatory needs

Traditional QA Applied to LLM Workflows

QA Type	Applied On
Unit Testing	Prompt parsing, chunking logic
Regression Testing	Prompt changes, model updates
Integration Testing	E2E: Scan → Chunk → Prompt → LLM
Functional Testing	“Extract clause X from document Y”
Exploratory Testing	Edge-case clause discovery
Smoke Testing	Stability with diverse inputs

Data Handling & Governance

Risks We Actively Mitigate

- Prompt leaks (PII to external APIs)
- Logged outputs with sensitive data
- Unsecured reuse of production documents

Our Practices

- Redacted real data only (synthetic avoided due to OCR constraints)
- No production reuse without legal clearance
- PII masking, GDPR tagging in test workflows
- API logging disabled or anonymised

BA-QA-Dev Collaboration: A Tight Feedback Loop

- Client provides golden examples with labelled expectations
- BA approval precedes QA review; any correction resets the approval cycle
- Prompt updates are reviewed by system architects to prevent cascading failures
- Initial testing is manual and deliberate to ensure a high-quality ground truth

Goal: Computer output that surpasses human accuracy in both completeness and consistency

Conclusion
QA That Makes LLMs Work in the Real World

CalibreCode delivered more than a test cycle. We designed a validation pipeline that keeps learning, correcting, and securing.

Our QA practices helped a legal-tech innovator move from prototype to production with confidence. We brought together classical QA thinking, real-world document nuances, and LLM-specific strategies to enforce output reliability, prompt safety, and data trust.

Thinking of building or scaling an LLM-backed product?
Let's talk about making your outputs accurate, trusted, and ready for prime time.

Let's Talk