# Companies House AI Data Pipeline: Integration Recommendations for Cursor

Date: 2025-06-12

Prepared for: Cursor AI Engineering Team

Prepared by: Strategic Counsel System

## 1. Integration Overview

This document outlines how Companies House data can be extracted, parsed, and integrated into an AI legal investigation pipeline, with specific emphasis on reliability, speed, OCR recovery, and local LLM support. The goal is to inform Cursor engineers of the design constraints, performance bottlenecks in AWS, and how a localised pipeline provides major advantages for legal-grade RAG (retrieval-augmented generation).

## 2. Pipeline Architecture

The ideal Companies House data pipeline operates as follows:  
1. Accepts UK company numbers as input.  
2. Fetches profile data, officers, PSCs, and filings via Companies House API.  
3. Downloads filing documents (PDF, iXBRL, HTML).  
4. Detects file type and processes:  
 - PDF with text layer: direct extract.  
 - PDF image only: OCR via Tesseract or CH's OCR API.  
 - iXBRL: parsed with Arelle.  
5. Chunks text for LLM embedding, tagged with metadata (company, date, filing type).  
6. Stores raw text, OCR fallback, and embeddings in a structured format for retrieval.

## 3. Known Bottlenecks and Mitigation Strategies

• OCR Failures: Low-res scans or handwritten forms can lead to unreadable output. Mitigate with 400 DPI render and Tesseract `--psm 4` fallback.  
• AWS I/O Latency: S3 read/write latency drastically slows parsing. Use EFS or local SSD.  
• Serverless Limits: Lambda is inefficient for OCR. Use EC2 compute-optimised or move OCR local.  
• Embedding Latency: OpenAI/Bedrock APIs are slow. Replace with `bge-small` or Mistral on local CPU.  
• Silent Failures: Documents with no text/OCR errors can pass silently. Implement QA + flag for manual review.  
• Rate Limits: Stay under 600 req/5 min quota via token bucket scheduling.

## 4. Strategic Recommendations for Cursor

• Local-first pipeline: Prioritise local disk, local LLM, and Tesseract for speed and privacy.  
• Use async I/O and batch logic to maximise throughput (e.g. `httpx.AsyncClient` + `asyncio.gather`).  
• Embed vector chunks locally with metadata for deterministic RAG.  
• Log SHA256 hashes of PDFs and extracted content for audit integrity.  
• Red-flag detector module: highlight DS01s, audit notes, and resignations.  
• Investigative prompt templates: e.g., “List changes in directors with citations.”  
• Provide fallback logic for unreadable or blank filings.  
• Maintain dual text source: extracted and OCR, tagged by confidence.

## 5. Optional Hybrid Cloud-Local Architecture

| Tier | Role |  
|------------------|-------------------------------------------|  
| Local SSD | Fast I/O for filings |  
| Local OCR | Deterministic and low-latency |  
| Local Embedder | bge-small or Mistral model |  
| Cloud Backup | Daily Glacier archive (not runtime) |  
| Strategic Counsel| Prompt + vector search wrapper |

## 6. Next Steps Checklist

✔ Implement local async Companies House fetcher  
✔ Integrate Tesseract OCR with DPI detection  
✔ Build vector embedding with chunk tagging  
✔ Store PDFs and extracted content with hashes  
✔ Surface unusable docs in manual review queue  
✔ Build initial RAG interface + prompt templates  
✔ Test on 1000 companies and benchmark timing