Global Architecture Document (GAD)

Augmented Document Reader (ADOR)

CMI Architecture & Innovation

August 21, 2025 v1.0

Purpose: Define the end-to-end architecture for a financial document reader that supports classification, summarization, topic modelling, NER, and Q&A across heterogeneous inputs (chat, DOCX, PDF) with both synchronous and asynchronous processing modes.

Contact: kpdravid@gmail.com

Contents

1 Executive Summary

Augmented Document Reader (ADOR) enables business users and systems to ingest financial documents via UI and APIs, and run **classification**, **summarization**, **topic modelling**, **Named Entity Recognition (NER)**, and **Q&A**. The platform abstracts parsing differences across *chat logs*, *DOCX term sheets*, and *PDFs*, applies the appropriate extraction strategy (rule-based parser, general NER model, or LLM/RAG pipeline), and returns structured JSON and audit evidence.

The PoC implementation exposes a single FastAPI endpoint and a simple HTML uploader. It demonstrates:

- Rule-based DOCX parsing,
- General NER model for chat/TXT,
- LLM extraction for PDFs using `Gemini` (default) or `GPT` as a selectable provider.

2 Scope and Goals

2.1 In Scope

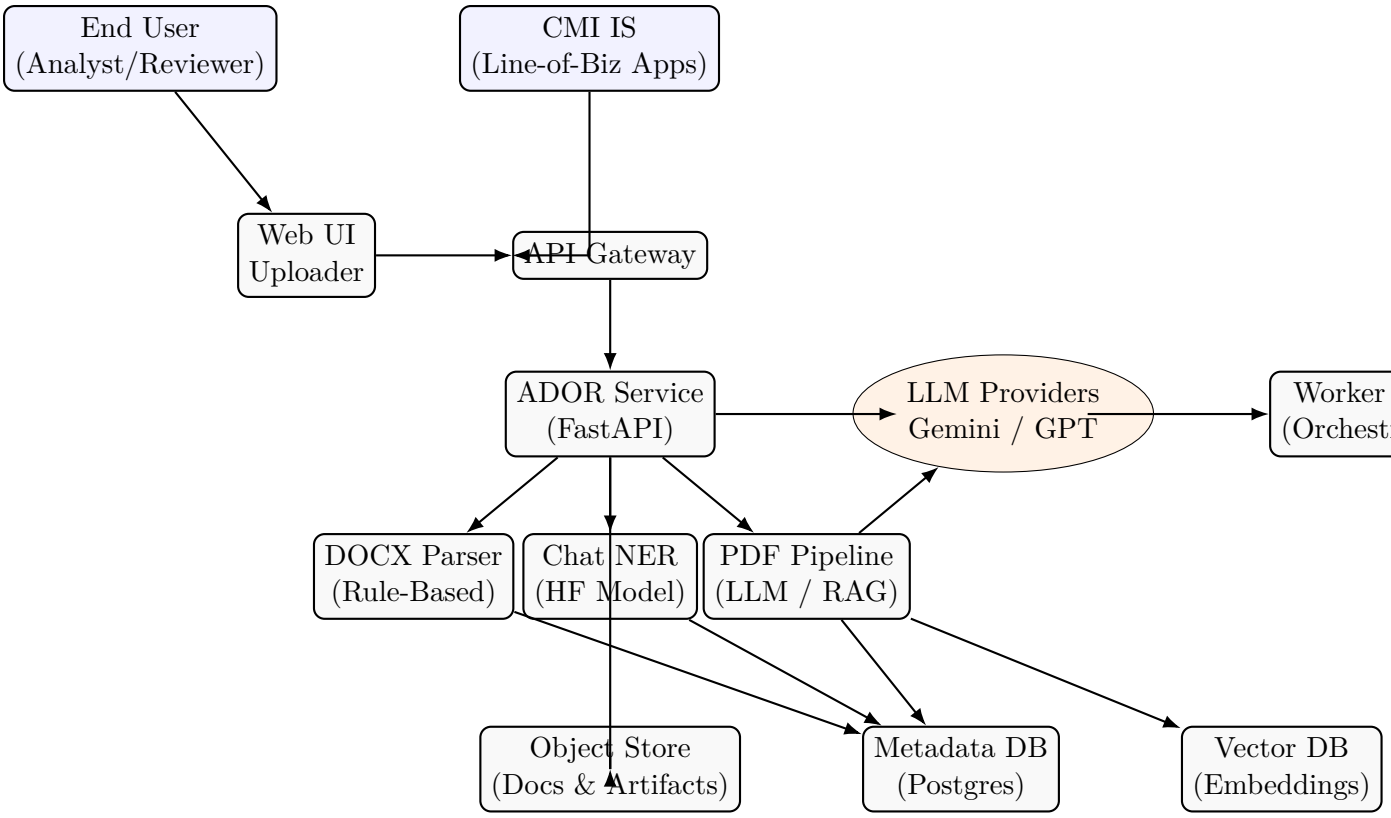
- Document ingestion via: Web UI, REST APIs, future connectors (SFTP, E-mail, Object store drop).
- Sync/async processing with job tracking and retry.
- Feature family: classification, summarization, topic modelling, NER, Q&A.
- Security: OIDC-based authN, role-based authZ, encryption in transit/at rest, audit logging.
- Observability: logs, metrics, traces; data lineage of extracted entities to source spans/pages.

2.2 Out of Scope (PoC)

- Enterprise DLP integration, full data residency controls, multi-region DR.
- Human-in-the-loop UI for redlining/approval (stubbed via evidence links).

3 System Context

3.1 Context Diagram



3.2 Actors

- **End Users** (Analyst/Reviewer) via UI; **Systems** via REST API.
- **ADOR Service** routes documents to the correct pipeline and consolidates results.
- **External LLM Providers** for PDF extraction and Q&A when allowed.

4 Functional Architecture

4.1 Ingestion Channels

1. **Web UI**: upload single/multiple files; choose feature (classification/summarization/topic/NER/Q&A).
2. **REST API**: programmatic submission; supports sync (`wait=true`) and async (`wait=false`).
3. **Future connectors**: SFTP drop, E-mail inbox, object store notifications.

4.2 Routing Logic

Type	Detector	Processing
Chat/TXT	MIME/text size hint	HF NER model (BERT family) + regex post-rules; returns entities
DOCX	MIME & extension	Rule-based parser on `python-docx`; maps headings to fields
PDF	MIME/pdf	LLM extraction: Gemini (default) or GPT; optional RAG for Q&A

4.3 Feature Capabilities

- **Classification**: model predicts document type (term sheet, trade confirm, factsheet, invoice, chat, etc.).
- **Summarization**: extractive or abstractive summary with length and style controls.
- **Topic Modelling**: keyword/topic discovery; surfaced as tags.
- **NER**: schema-constrained extraction (financial entities) + evidence spans.
- **Q&A**: grounded answers referencing specific sections/pages (RAG).

5 Processing Modes

5.1 Synchronous (low-latency)

For small text/DOCX and short PDFs. API responds immediately with results and minimal artifacts (JSON entities, summary).

5.2 Asynchronous (batch/large/confidential)

Large PDFs or multi-file uploads enqueue a job. Client receives a `job_id` and polls or subscribes to webhooks. Artifacts (normalized text, chunks, evidence) are persisted.

6 Data & Storage

- **Object Store**: original files, normalized text, chunk JSON, evidence snippets, redacted variants.
- **Metadata DB (Postgres)**: documents, jobs, features requested, status, entity JSON, lineage (page/span).
- **Vector DB**: embeddings for RAG and semantic retrieval (e.g., FAISS/pgvector).

7 Security Model

7.1 AuthN/Z

- OIDC/OAuth2 with JWT; roles: `admin`, `analyst`, `reviewer`, `service`.
- Row-level authorization: tenants/projects; per-document ACLs.

7.2 Confidentiality Policies

We define levels and allowed compute:

Level	Examples	Compute Policy
Public	Marketing PDFs	Any provider
Internal	Generic trade confs	

Cloud LLMs allowed; redact PII before send	
Confidential	Client term sheets
Prefer on-prem NER/rule-based; LLM via private gateway only	
Restricted	Regulated / NDA
No external calls; offline models only; local inference	

7.3 Privacy & Compliance

TLS 1.2+, encryption at rest, secret management (vault), PII minimization, audit logs with immutable storage. Data retention per policy; hard delete with tombstones.

8 Observability & Reliability

Structured logs (JSON), metrics (latency, tokens, queue depth), traces through request → parse → LLM. Dead-letter queue for failed jobs; idempotent job keys; exactly-once consolidation.

9 Interfaces (APIs)

9.1 Core Endpoints

Method/Path Notes
POST /api/extract Form fields: `doc_type` = pdf|docx|chat; for pdf: `provider` = gemini|openai, optional `model`. Returns JSON entit
GET /health Returns `{ok:true}`
POST /api/jobs Returns `job_id`
GET /api/jobs/{id} `pending|running|done|error`; result JSON or error

9.2 Response Shape (NER)

```
{
  "ok": true,
  "doc_type": "pdf",
  "provider": "gemini",
  "result": {
    "provider": "gemini",
    "model": "gemini-1.5-flash",
    "entities": [
      {"label": "ISIN", "text": "FR001400QV82"},
      {"label": "Notional", "text": "200 mio", "currency": "EUR"}
    ],
    "raw": "...raw LLM JSON string...",
    "meta": { "file_uri": "providers://...", "usage": {...} }
  }
```

```
}
```

10 LLM/RAG Methodology (GMD excerpt)

10.1 Prompting Strategy

- **System**: "strict JSON only", schema enumerated, no invention, copy values as-seen.
- **User**: either (a) full PDF text (Gemini Files) or (b) extracted text chunks (OpenAI).
- **Validation**: JSON parse + schema normalization; regex sanity (ISIN, dates, currency).

10.2 Chunking & Retrieval

For long PDFs: split by pages/headings with overlap; compute embeddings; retrieve top-K for field groups (dates; parties; amounts). Perform two-pass extraction: per-group then consolidate.

10.3 Evidence Capture

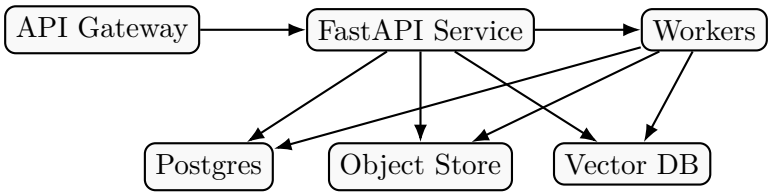
For each extracted field, persist *evidence*: page no., character span, snippet. Expose in UI for reviewer trust.

11 Deployment Architecture

Containerized services:

- API Gateway & FastAPI service
- Worker pool (async jobs)
- Postgres, Object Store (S3/Azure/MinIO), Vector DB
- Secret store (Vault/KMS)
- Observability stack (Prometheus/Grafana/ELK/OpenTelemetry)

11.1 High-Level Diagram



12 Non-Functional Requirements

- **Performance**: P95 sync requests < 2.5s for small docs; async SLA as per queue depth.
- **Scalability**: horizontal scale for API and workers; back-pressure via queue.
- **Reliability**: at-least-once job execution; idempotent result writes.

- **Security**: encryption in transit/at rest; least-privilege; audit trails.
- **Cost Controls**: LLM provider quotas, model tiering (flash vs pro), cache of embeddings/chunks.

13 PoC Implementation Snapshot

13.1 Tech Stack

- **Backend**: FastAPI, Python 3.11, Uvicorn.
- **Parsers**: `python-docx` (DOCX), HF NER (`dslim/bert-base-NER`) for chat, Gemini/OpenAI for PDFs.
- **UI**: minimal HTML uploader with doc type & provider selection.

13.2 Key Modules

- `main.py`: central API; sync returns; JSON-safe serialization.
- `pdf_utils.py`: Gemini Files API or OpenAI chunking path; strict JSON normalization.
- `docx_parser.py`: rule-based field extraction from headings/tables.
- `chat_parser.py`: NER pipeline + regex post-processing.

14 Risks & Mitigations

Risk	Mitigation
LLM drift / schema variance	Strict JSON prompts, response validation, unit tests with fixtures
Confidential data exfiltration	Policy routing: offline models for Restricted, redaction prior to LLM
PDF text quality (scans)	OCR fallback; evidence capture for reviewer
Vendor lock-in	Provider abstraction: Gemini/GPT interchangeable; optional on-prem models

15 Change Log

Date	Version	Notes
August 21, 2025	v1.0	Initial GAD for PoC