Implementation Guide - ASCII Edition

Version 2025-09-01

Repo Layout (put files exactly here)

/app/main.py - FastAPI app

/app/models.py - Pydantic models

/app/parser/preflight.py - OCR and preflight

/app/parser/tables.py - ROI table discovery

/app/parser/mapping.py - Header synonyms and shapes

/app/parser/confidence.py - Confidence blend

/app/parser/pii.py - PII scrub helpers

/app/parser/snips.py - Proof snip cropping

/app/profiles/ - Learned profiles

/golden_set/ - Test PDFs

requirements.txt, Dockerfile, .env.example

Run with Docker

1) docker build -t extractor:latest .

2) docker run --rm -p 8000:8000 extractor:latest

3) curl -F "file=@/path/to/sample.pdf" http://localhost:8000/analyze

Run without Docker

1) Install: tesseract-ocr ocrmypdf ghostscript qpdf poppler-utils openjdk-17-jre-headless

2) python -m venv .venv && source .venv/bin/activate

3) pip install -r requirements.txt

4) python -m spacy download en_core_web_sm

5) uvicorn app.main:api --host 0.0.0.0 --port 8000

Implement Next

A) Candidate scoring + scenario selection under Current

B) Reconciliation math: net_sv ~= cash_value - surrender_charge - loan - interest (<= 1 percent)

C) Current-year row selection

D) PDF coordinate PII boxes + metadata strip

E) Proof snips crop and embed (base64)

F) Auto profile save and reuse

Env Vars

CONFIDENCE_THRESHOLD=0.80

PROFILE_DIR=/app/app/profiles

MAX_RUNTIME_SECONDS=60

DISABLE_TABULA=false

Deploy (AWS ECS example)

Push image to ECR; create Fargate service (0.5 vCPU, 1GB); set env vars; ALB on port 8000; HTTPS

Integrate (Zapier/Make + OpenAI)

1) Trigger: new PDF

2) POST /analyze

3) Send JSON to OpenAI for decision and narrative

4) Fill template and embed proof snips

Acceptance Tests

Accuracy >= 95 percent; zero PII leaks; runtime <= 30s incl. OCR; threshold respected; provisional banner below 0.80

ASCII-only edition to avoid font glyph issues.