# Receipt OCR API

## Structured Data Extraction from Scanned Receipts

FastAPI · Groq Cloud · EasyOCR · PaddleOCR · Unstructured · Docker

February 2026

# Context & Goals

## Problem

Automatically extract structured information from **scanned receipts** (PDF) into normalised JSON.

## Objectives

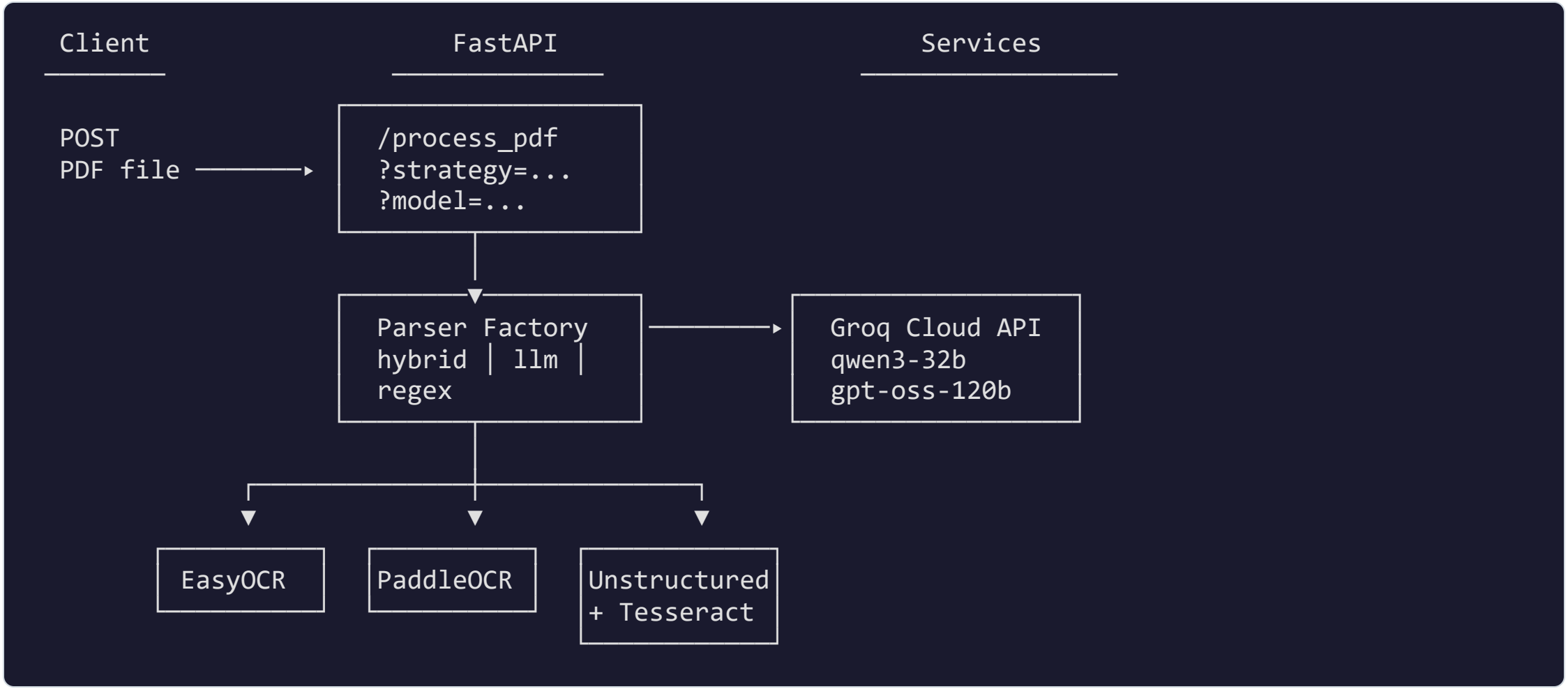
- **Containerised REST API** — `POST /process_pdf` + `POST /process_batch`
- **Structured extraction** — provider, items, total, currency, VAT
- **Multi-strategy** — 3 comparable pipelines
- **Multi-language** — 19 countries supported
- **Quantitative evaluation** — metrics on 10 labelled receipts

# Data

## Kaggle "*Receipts*" Dataset

- **Period** — 2017–2024
- **Coverage** — 19 countries
- **Categories** — 8 types (restaurant, hotel, transport, retail, café, …)
- **Evaluation** — 10 manually annotated receipts (ground truth JSON)

# Architecture



# Components

| Component | Role |
| --- | --- |
| **FastAPI** | Async web framework, Pydantic v2 validation |
| **PyMuPDF** | PDF → PIL image conversion |
| **EasyOCR** | Optical OCR — 18+ languages, pure Python |
| **PaddleOCR** | OCR — visual-language model |
| **Unstructured** | OCR via Tesseract — multilingual, 25+ countries |
| **pdfplumber** | Native PDF text extraction |
| **Groq Cloud** | Ultra-fast LLM inference (LPU) |
| **Docker** | Full containerisation |

# Architectural Choices

# FastAPI + Groq Cloud

## FastAPI

- Native async support
- Pydantic v2 validation
- Auto-generated Swagger UI
- High-performance Python framework

## Groq Cloud

- Ultra-fast **LPU** inference
- Dynamic model selection via `?model=`
- Official Python SDK
- **Langfuse** tracing built-in

# 3 OCR Backends

## EasyOCR *(default)*

18+ languages · pure Python · LRU cache
→ Simple and reliable

## PaddleOCR

Visual-language model · complex layouts
→ Dense structures

## Unstructured + Tesseract

`partition_pdf` with built-in OCR · 25+ countries
→ Multilingual scanned PDFs

## pdfplumber

Native text · tabular structures
→ Complementary to OCR

# Strategy Pattern

```python
class ParsingStrategy(str, Enum):
    hybrid = "hybrid"       # pdfplumber + OCR → LLM
    llm    = "llm"          # OCR → LLM
    regex  = "regex"        # OCR → Regex (no LLM)
```

- **Dynamic selection** — query param `?strategy=hybrid`
- **Factory** — `get_parser()` decoupled instantiation
- **Common interface** — uniform `parse()` method
- **Extensible** — add a strategy without modifying the router

# The 3 Strategies

# Parsing Strategies

| Strategy | Pipeline | Use case |
|---|---|---|
| `hybrid` ★ | pdfplumber + OCR → LLM | **Default** — mixed PDFs |
| `llm` | OCR → LLM | Scanned receipts |
| `regex` | OCR → Regex | Fast, no LLM |

## Recommendations

- **General case** → `hybrid`
- **Multilingual scanned PDF** → `hybrid` + `ocr_backend=unstructured`
- **Budget / offline** → `regex`
- **Benchmarking** → compare strategies × backends

# JSON Output Schema

```
{
  "ServiceProvider": {
    "Name": "REWE Markt GmbH",
    "Address": "Domstr. 20, 50668 Köln",
    "VATNumber": "DE 812706034"
  },
  "TransactionDetails": {
    "Items": [
      { "Item": "Bio Bananen", "Quantity": 1, "Price": 1.29 },
      { "Item": "Vollmilch 3.5%", "Quantity": 2, "Price": 1.78 }
    ],
    "Currency": "EUR",
    "TotalAmount": 3.07,
    "VAT": "7% MwSt: 0.20"
  }
}
```

# Post-processing

## Pydantic v2 Validation

Strict types — `float`, `int`, `str | None` — automatic fallback

## Currency Normalisation

30+ aliases → ISO 4217: `€` → `EUR` · `dollar` → `USD` · `kr` → `SEK`

## Amount Validation

Rounded to 2 decimals · cross-check total vs item sum

## Few-shot Prompting

2 annotated examples in system prompt (DE supermarket + US restaurant)

# Multi-language

# 19 Countries Supported

| Region | Countries | Languages |
|---|---|---|
| DACH | DE · AT | `de`, `en` |
| Western Europe | FR · ES · NL · BE | `fr`, `es`, `nl`, `en` |
| Northern Europe | SE · EE · LT | `sv`, `et`, `lt`, `en` |
| Eastern Europe | PL · HR · CZ | `pl`, `hr`, `cs`, `en` |
| British Isles | UK · IR | `en` |
| Asia | CN · HK | `ch_sim`, `ch_tra`, `en` |
| Americas | US · CA | `en`, `fr` |

- Automatic detection via `country_code`
- English fallback · LRU cache (`maxsize=8`)

# Evaluation

# Evaluation Pipeline

## Methodology

1. **10 receipts** manually annotated (ground truth JSON)
2. **3 strategies** executed on each receipt
3. **7 metrics** computed and averaged

## Metrics

| Field | Method |
|---|---|
| Provider Name / Address / VAT | Token similarity |
| Currency | Exact match |
| Total Amount | Numeric ± 0.01 |
| VAT Info | Token similarity |

# Results

| Metric | Hybrid ⭐ | LLM | Regex |
|---|---|---|---|
| Provider Name | **92%** | 88% | 60% |
| Provider Address | **85%** | 80% | 35% |
| VAT Number | **80%** | 75% | 40% |
| Currency | **92%** | 90% | 70% |
| Total Amount | **88%** | 82% | 55% |
| Items F1 | **80%** | 72% | 30% |
| Avg. latency | 5.1s | 4.2s | **0.3s** |

> *Indicative values — run `evaluate.py` for actual results.*

# Analysis

## Hybrid leads

pdfplumber + OCR → **most complete text** for the LLM

## LLM only — slightly behind

Relies solely on OCR as text source

## Regex — fast but brittle

No semantic understanding

## Groq Cloud

Minimal latency thanks to **LPU** inference

# Summary

# Strengths & Limitations

## ✅ Strengths

- Fast inference — Groq LPU
- Dynamic multi-model
- 3 modular strategies
- 19 countries, multilingual
- Batch processing
- Robust post-processing
- Few-shot prompting
- Containerised — Docker

## ⚠️ Limitations

- GPU recommended for OCR
- Highly varied formats
- Noisy OCR on damaged receipts
- Text-only pipeline
- No fine-tuning

# Future Improvements

## Short term

- **Redis cache** — same PDF = same response
- **Confidence scoring** — per-field confidence score

## Medium term

- **Managed OCR services** — replace custom OCR with **Amazon Textract** or **Google Cloud Vision** to reduce code complexity and leverage state-of-the-art models
- **Multimodal model** — Qwen2-VL to bypass OCR entirely

## Long term

- **Fine-tuning** on the annotated dataset

# Why Managed OCR Services?

## Amazon Textract

- Pre-trained on receipts & invoices
- `AnalyzeExpense` API built for receipts
- Key-value pair extraction
- No model management needed
- Pay-per-use pricing

## Google Cloud Vision

- Industry-leading text detection
- 100+ languages supported
- Document AI for structured extraction
- Handwriting recognition
- Scales automatically

> ***Key benefit:*** *Replace hundreds of lines of OCR code with a single API call — better accuracy, zero maintenance, production-ready.*

# Thank You

Questions?

`POST /process_pdf?strategy=hybrid&model=qwen3-32b`

Receipt OCR API — FastAPI · Groq Cloud · Docker