# Design Document - Invoice Validation System

## Overview

This document describes a proof-of-concept invoice validation system that

uses real PDF invoices from the internet, runs OCR to extract fields, compares

them against ground truth and PO/database records, highlights discrepancies,

and outputs validation status with confidence.

## Approach

1. Data ingestion + OCR

- `data/sources.csv` drives downloads into `data/raw\_pdfs/`.

- Render PDFs to PNG at 300 DPI with optional scan simulation.

- Run Tesseract OCR (eng+vie) to get word boxes, line text, and confidence.

- Save raw OCR to `outputs/ocr\_raw.json`.

2. Ground truth + database

- Parse the PDF text layer to build `data/ground\_truth.json` (header fields

+ basic line items where possible).

- Clean noisy name/address candidates and skip obviously invalid labels.

- Build `data/database.json` (PO + vendor\_master + customer\_info), and inject

negative cases for ~30% invoices to show business-rule discrepancies.

3. Normalization

- Text: lowercase, strip punctuation, collapse whitespace.

- Company suffix normalization (inc, llc, ltd, co).

- Address abbreviation normalization (street -> st, drive -> dr).

- OCR confusable mapping (O/0, I/1/l, S/5, B/8).

4. Matching and validation

- ID fields (PO): exact match after OCR-confusion normalization.

- Names/addresses: fuzzy similarity (token set + edit ratio).

- Dates: tolerance window in days.

- Amounts: absolute and relative tolerance.

- Database references:

- PO vendor used as primary expected vendor name.

- vendor\_master address used for vendor address validation when available.

- customer\_info billing\_address used for customer address validation.

- tax\_rate from PO used to cross-check tax\_amount.

- Line items: parse text to (description, qty, unit\_price, total).

- Item not in PO list -> critical.

- Quantity above max -> critical.

- Line total mismatch -> warning.

5. Discrepancy scoring and status

- Discrepancy severity: critical / warning / informational.

- Overall confidence combines OCR confidence and rule penalties.

- Status logic:

- critical -> rejected.

- warnings or low confidence -> needs\_review.

- otherwise approved.

6. Option A (ML)

- Logistic regression predicts probability a field is wrong.

- Features: OCR confidence, text length, digit ratio, fuzzy score,

amount/date diffs, field identity.

- Model output calibrates confidence per field and overall score.

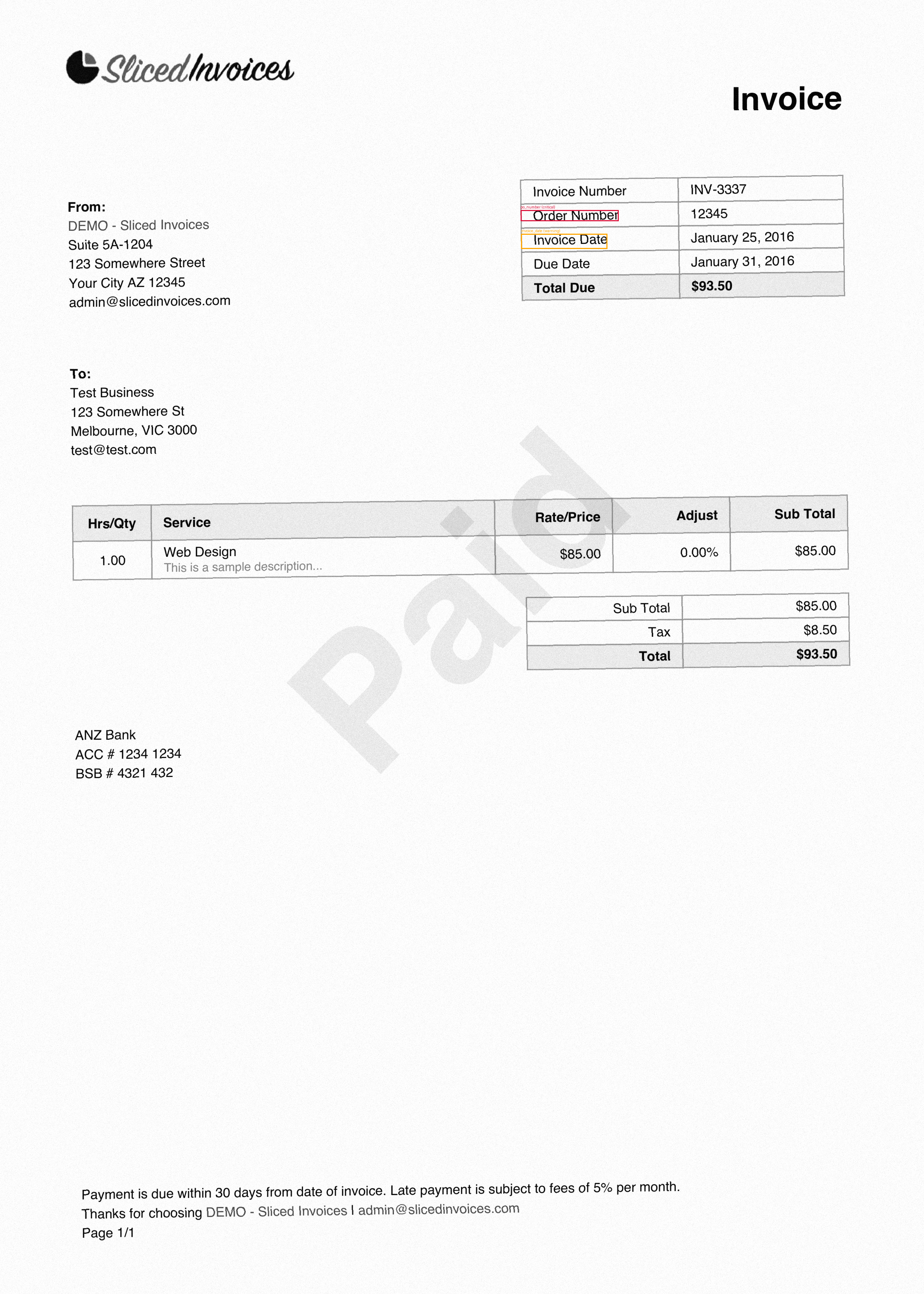
7. Output generation

- `scripts/generate\_sample\_outputs.py` writes JSON results per invoice.

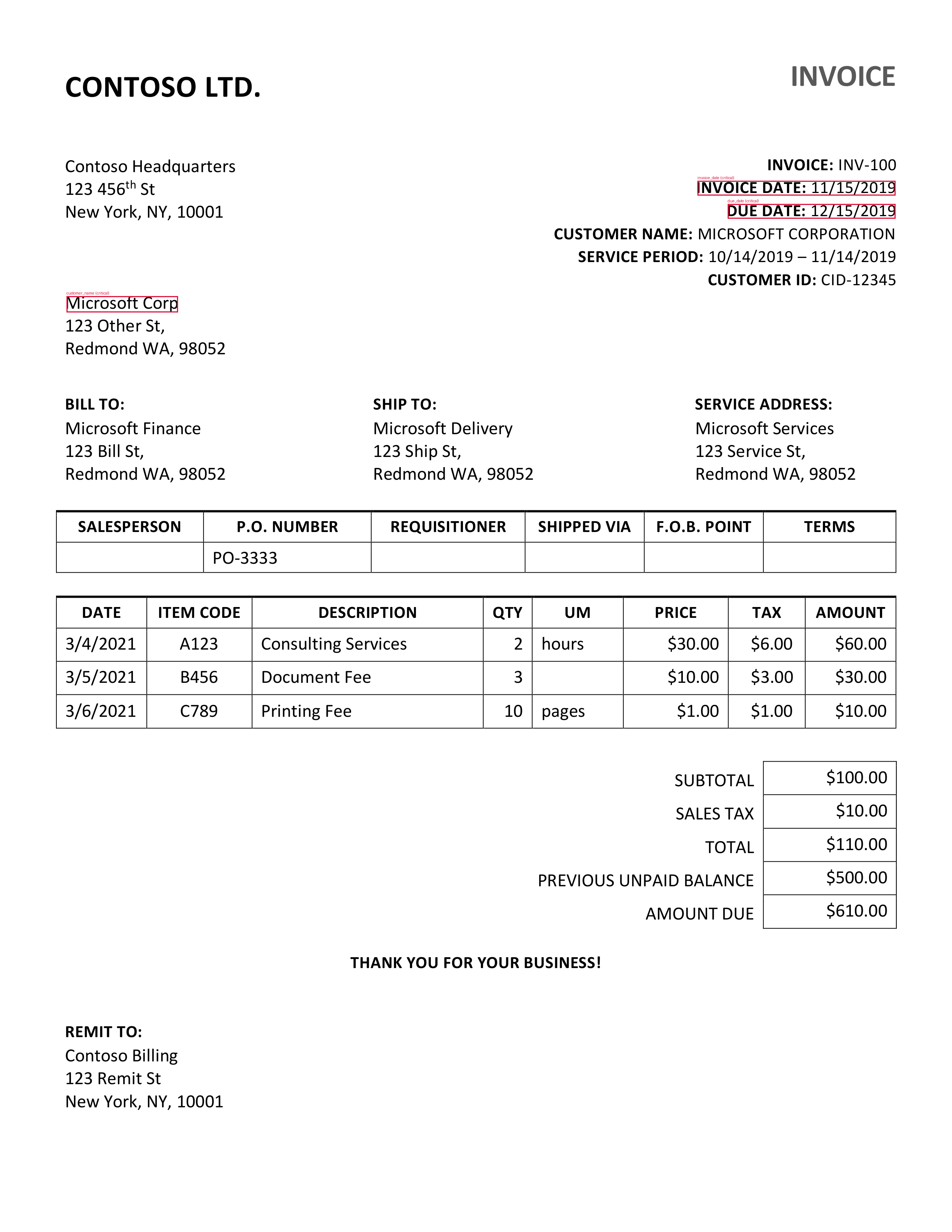
- `src/output/visualize.py` draws highlighted PNGs from discrepancy boxes.

## Sample Outputs (Images)

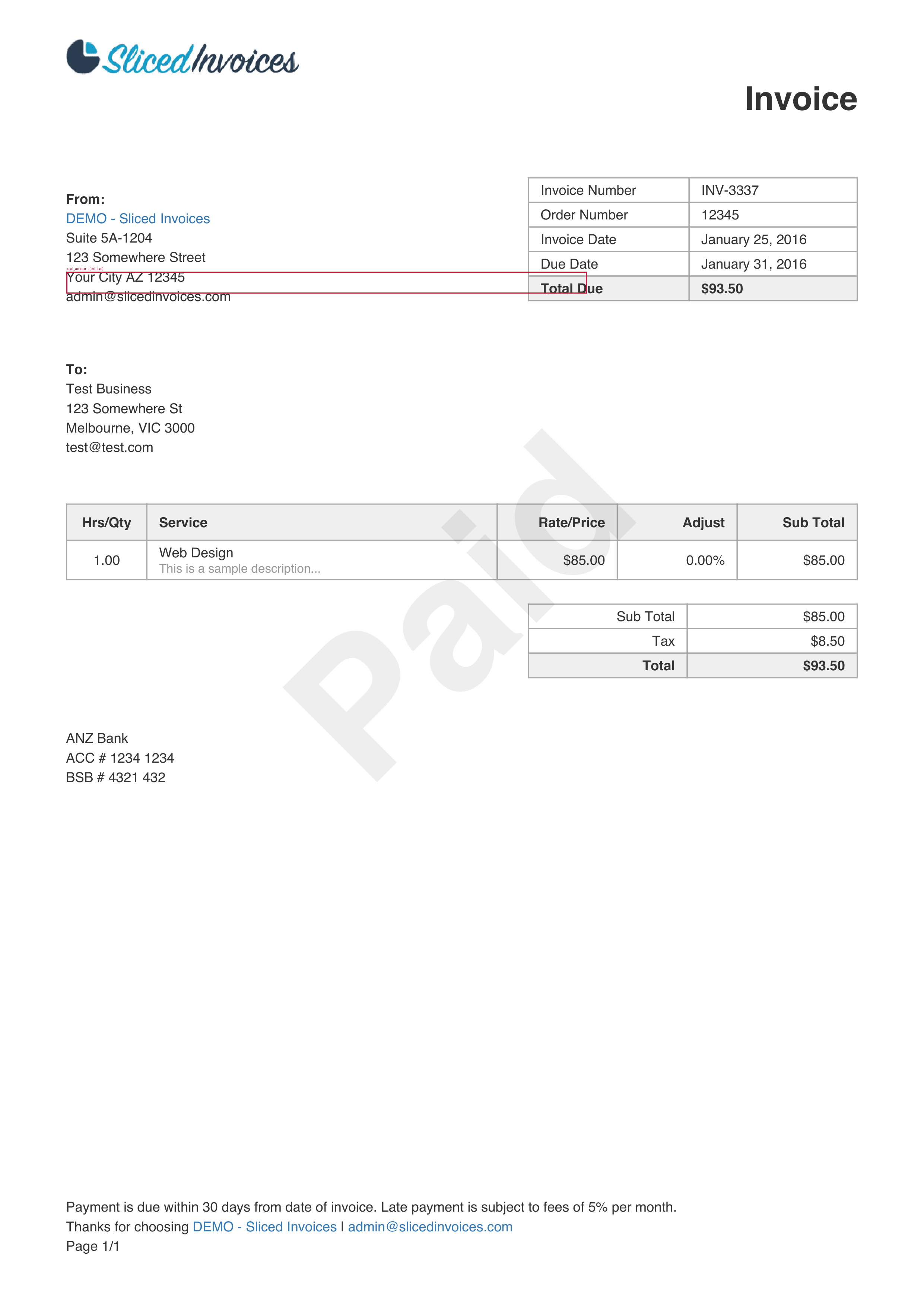
These are example highlighted outputs showing discrepancy boxes.



INV-G-001 header highlights



INV-G-003 header + line item highlights



INV-G-010 header highlights

## Ground Truth vs PO Comparison (Example)

Ground truth comes from `data/ground\_truth.json` and PO data comes from

`data/database.json`. The validator compares OCR results against ground truth

and then cross-checks against PO constraints (vendor, tax\_rate, approved\_amount).

Ground truth entry (invoice\_id = INV-G-010):

{  
 "invoice\_id": "INV-G-010",  
 "source\_pdf": "data/raw\_pdfs/INV-G-010.pdf",  
 "expected\_data": {  
 "vendor\_name": null,  
 "vendor\_address": null,  
 "customer\_name": "Test Business",  
 "customer\_address": "123 Somewhere St, Melbourne, VIC 3000",  
 "po\_number": "12345",  
 "invoice\_date": "2016-01-25",  
 "due\_date": "2016-01-31",  
 "subtotal": 85.0,  
 "tax\_amount": 8.5,  
 "total\_amount": 93.5,  
 "line\_items": []  
 }  
}

PO record used for comparison (po\_number = 12345):

{  
 "po\_number": "12345",  
 "vendor": null,  
 "approved\_amount": 93.5,  
 "valid\_items": [],  
 "max\_quantity": {},  
 "tax\_rate": 0.1  
}

## Pipeline Diagram

flowchart LR  
 SRC[data/sources.csv] --> PDF[raw\_pdfs/\*.pdf]  
 PDF --> R[Render to PNG]  
 R --> OCR[OCR (Tesseract)]  
 OCR --> RAW[ocr\_raw.json]  
 RAW --> PARSE[Field extraction]  
 PARSE --> OCRRES[ocr\_results.json]  
 GT[ground\_truth.json] --> V[Validation Rules]  
 DB[database.json] --> V  
 OCRRES --> N[Normalization]  
 N --> V  
 V --> D[Discrepancies]  
 V --> L[Line Item Parsing/Anomaly]  
 L --> D  
 D --> C[Confidence Scoring]  
 C --> S[Status Decision]  
 D --> VIZ[Visualization Coords]  
 V --> ML[Option A: Logistic Regression]  
 ML --> C  
 S --> OUT[Output JSON]  
 VIZ --> OUT  
 VIZ --> IMG[Highlighted Images]

## Rule Rationale & Thresholds

* PO/ID fields: strict matching is required because they drive payment linkage

and vendor approval; OCR-confusion normalization (O/0, I/1/l, S/5, B/8)

reduces false rejections from common OCR errors.

* Names/addresses: fuzzy matching handles abbreviations and truncation; thresholds

are softer to avoid penalizing minor formatting differences.

* Dates: tolerance of 1-3 days is treated as warning based on typical OCR/entry

drift; larger gaps are critical.

* Amounts: combined absolute and relative tolerance captures rounding/tax noise

across both small and large invoices.

* Tax: tax\_amount is cross-checked against PO tax\_rate when available to validate

accounting consistency.

* Line items: items not in PO list or quantity above max are critical because

they violate approved purchasing rules; line total drift is warning to allow

minor rounding.

* Status policy: critical issues are rejected; warnings or low confidence stay

in needs\_review for human inspection.

## Assumptions and Limitations

* OCR quality varies by template; some PDFs are closer to forms than real scans.
* Ground truth uses heuristic extraction + cleanup and may miss uncommon layouts.
* Limited sample size: ML model is illustrative only.
* Address parsing is token-based, not full geocoding.

## Scalability Considerations

* Batch or streaming validation supported (stateless rules).
* Thresholds and mappings are centralized in config for tuning.
* Add vendor-specific templates to improve field extraction.
* Introduce async processing for large volumes.

## Deliverables Mapping

* Notebook: data overview and metrics (`data\_overview.ipynb`).
* Core engine: `src/\*` modules.
* Sample outputs: `sample\_outputs/\*.json` + `sample\_outputs/images/\*.png`.
* Tests: pytest with 15 cases.