Receipt & Invoice Digitizer

Milestone 1 & Milestone 2 – Combined Presentation

Automated OCR-Based Financial Document Processing System

Problem Statement

- **Time-consuming:** Manual data entry slows down financial reporting and accounting.
- **Human Error:** Manual calculation errors lead to financial discrepancies.
- **Redundancy:** Duplicate expense records are often missed in large piles of paper.
- **Poor Visibility:** Spending data is locked in physical paper, making analysis impossible.

Objective: Build an automated, accurate, and scalable digitization system.

Overall Project Objectives

Automate

Digitize receipts and invoices automatically upon upload.

Extract

Extract structured financial data (Vendor, Total, Tax) using AI.

Analyze

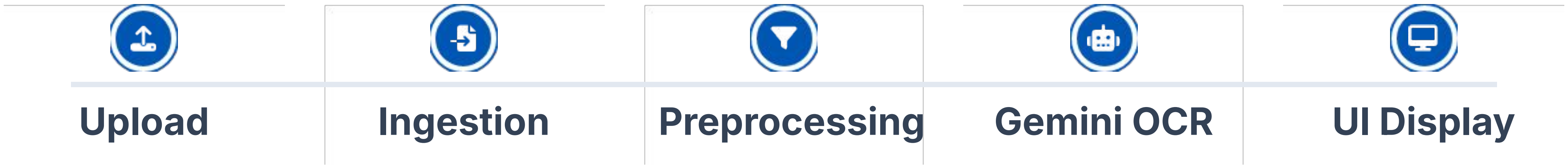
Validate numerical correctness, detect duplicates, and visualize spending.

Milestone 1

Foundation: Ingestion, Preprocessing & OCR

Objective: Establish a stable and secure foundation for document digitization with reliable OCR, preprocessing, and data normalization.

Milestone 1 Architecture

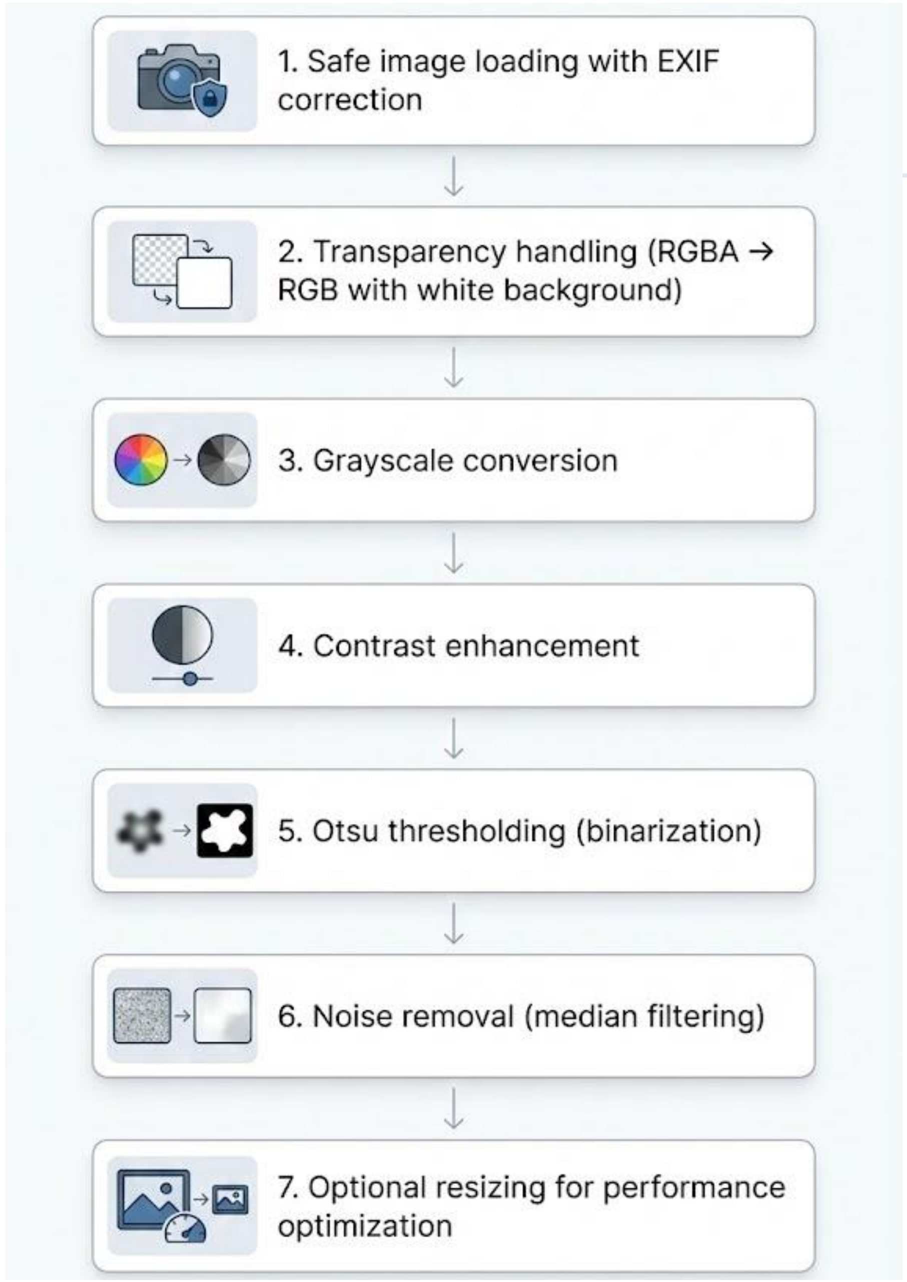


Ingestion Layer

- **File Type Detection:** Detects Image vs PDF using magic bytes.
- **Limits:** Enforces size (5MB) and page counts to prevent overload.
- **Secure Ingestion:** Handles binary streams safely in memory.
- **Change Detection:** Generates SHA-256 hash to identify file uniqueness.

Security First

The ingestion layer acts as the gatekeeper, ensuring only valid and safe documents enter the processing pipeline.



1. Safe image loading with EXIF correction
2. Transparency handling (RGBA → RGB with white background)
3. Grayscale conversion
4. Contrast enhancement
5. Otsu thresholding (binarization)
6. Noise removal (median filtering)
7. Optional resizing for performance optimization

Image Preprocessing

- **EXIF Correction:** Auto-rotates images based on camera metadata.
- **Grayscale:** Converts to grayscale to reduce processing complexity.
- **Enhancement:** Increases contrast to separate text from background.
- **Denoising:** Removes thermal paper grain and shadows.
- **Optimization:** Resizes large images for optimal API latency.

Gemini OCR Extraction

- **AI-Powered:** Uses Gemini 2.5 Flash for context-aware reading.
- **Structured Output:** Prompts the model to return strict JSON.
- **Key Fields:** Vendor, Date, Invoice Number, Tax, Total.
- **Fallback:** Preserves raw text for secondary Regex analysis.

JSON Schema

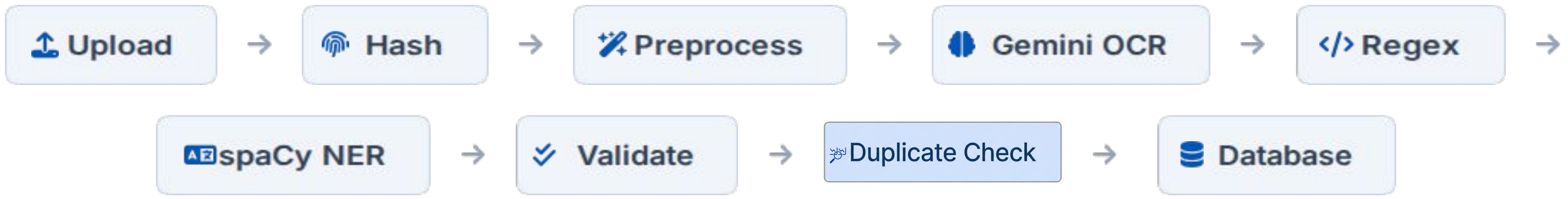
```
{
  "vendor": "Walmart",
  "date": "2023-10-12",
  "total": 45.99,
  "tax": 2.50
}
```

Milestone 2

Intelligence: Validation, Fallbacks & Analytics

Objective: Enhance the core pipeline with intelligent fallback extraction, validation, duplicate detection, and analytics.

Combined High-Level Architecture



Regex Fallback Extraction

- **Purpose:** Safety net applied when AI fields are low confidence.
- **Targets:** Dates, Invoice number,taxes,currency, Totals.
- **Method:** Deterministic pattern matching.
- **Benefit:** Catches data even if the AI "hallucinates".

Pattern Logic

```
r"Total[:\s]*\$?(\d+\.\d{2})"
```

spaCy NLP Vendor Extraction

Named Entity Recognition

Identifies **ORG** entities in the raw text block.

- **Trigger:** Activated if the primary Vendor field is missing.
- **Context Aware:** Distinguishes company names from random text.
- **Robustness:** Handles noisy OCR text better than simple regex.
- **Library:** Uses spaCy's efficiency-optimized English model.

Normalization & Currency

Dates

Standardized to ISO format **YYYY-MM-DD** for DB sorting.

Currency

All amounts normalized to numeric floats (USD default).

Text

Title-casing vendors and trimming whitespace.

Validation Layer

- **Math Check:** Verifies if *Subtotal + Tax = Total*. Tax-inclusive and tax-exclusive checks
- **Tolerance:** Allows small OCR float variations ($0.02).
- **Feedback:** Returns actionable warnings to UI.
- **Completeness:** Ensures amount correction with re-validation

Duplicate Detection Layer

Matching Logic

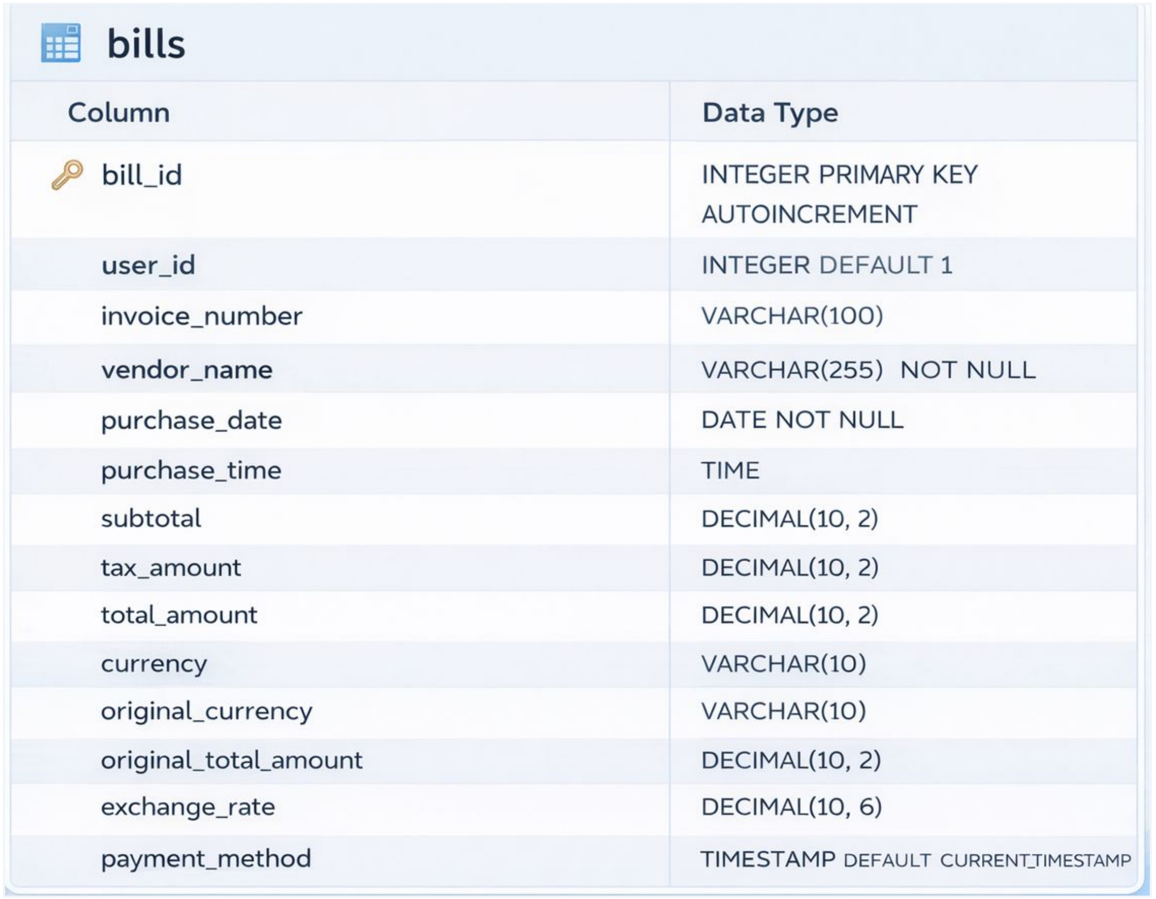
A receipt is considered a duplicate if it matches existing records on:

```
(Invoice_Num + Vendor + Date + Amount)
```

- **Hard Duplicate:** Exact match on all fields. **Action: Block Save.**
- **Soft Duplicate:** High similarity. **Action: Warn User.**
- **Hash Match:** File-level SHA-256 deduplication.

Database Design

- **Engine:** SQLite (Persistent Storage).
- **Indexing:** Optimized for date ranges and vendors.
- **Design:** Normalized schema with Bills and Line Items.



bills

Column	Data Type
bill_id	INTEGER PRIMARY KEY AUTOINCREMENT
user_id	INTEGER DEFAULT 1
invoice_number	VARCHAR(100)
vendor_name	VARCHAR(255) NOT NULL
purchase_date	DATE NOT NULL
purchase_time	TIME
subtotal	DECIMAL(10, 2)
tax_amount	DECIMAL(10, 2)
total_amount	DECIMAL(10, 2)
currency	VARCHAR(10)
original_currency	VARCHAR(10)
original_total_amount	DECIMAL(10, 2)
exchange_rate	DECIMAL(10, 6)
payment_method	TIMESTAMP DEFAULT CURRENT_TIMESTAMP

Streamlit Application Flow

1. Process

Upload, extract, and review validation results.

2. Dashboard

Visual analytics, KPIs, and spending trends.

3. History

Browsable ledger of saved bills with export.

Security & Stability

Input Limits

Strict file size and page limits to prevent DoS.

Credentials

No API keys in code; uses Environment Variables.

Defensive

Robust JSON parsing handles API errors gracefully.

Outcomes Achieved

- ✓ **Reliable Pipeline:** End-to-end processing from upload to DB.
- ✓ **High Accuracy:** AI + Regex + NLP approach.
- ✓ **Data Integrity:** Strong math validation and duplicate blocking.
- ✓ **Scalable:** Modular design ready for production.

Thank You