RECEIPT AND INVOICE DIGITIZER

A Smart System for Receipt and Invoice Text Extraction

January 13, 2026

HARINI KAVETI

harinikaveti2003@gmail.com

# **CONTENTS**

1. Title Page
2. Contents Page
3. Problem Statement
4. Modules
5. Architecture Diagram
6. Process Flow
7. Database
8. Validation
9. Output
10. Conclusion
11. References

# MILESTONE – 1 PROBLEM STATEMENT

In many organizations and for individuals, receipts and invoices are still handled in paper form. These paper documents are easily lost, damaged, or incorrectly entered into systems when done manually. Manual data entry also consumes time and may lead to errors in financial records.

The goal of this project is to build an automated system that can scan receipts and invoices, extract text using OCR (Optical Character Recognition), and convert them into digital records.

**Milestone 1 focuses on:**

* Uploading receipt/invoice files
* Preprocessing images for better OCR accuracy
* Extracting raw text from receipts and invoices

# **MODULE DESCRIPTION AND OBJECTIVES**

## 2.1 Document Ingestion Module

**Objective:** To allow users to upload receipt or invoice files into the system.

**Description:** This module enables users to upload documents in image or PDF format. The system supports common formats such as JPG, PNG, and multi-page PDF files. Uploaded PDFs are converted into images so they can be processed further.

**Key Functions:**

* File upload using web interface
* Support for image and PDF formats
* Conversion of PDF pages into images

## 2.2 Image Preprocessing Module

**Objective:** To enhance image quality and improve OCR accuracy.

**Description:** Before performing OCR, uploaded images are pre processed to remove noise and improve text clarity. This step is important because raw images may contain shadows, blur, or poor contrast.

**Key Functions:**

* Conversion of images to grayscale
* Contrast enhancement using CLAHE
* Noise removal using Gaussian blur

## 2.3 OCR and Raw Text Extraction Module

**Objective:** To extract readable text from processed images.

**Description:** This module uses Tesseract OCR to read text from cleaned receipt and invoice images. The extracted text is returned as raw text, which is later used for field extraction in further milestones.

**Key Functions:**

* OCR processing using Tesseract
* Extraction of raw text from images
* Handling of multi-page documents

# TECHNOLOGIES USED

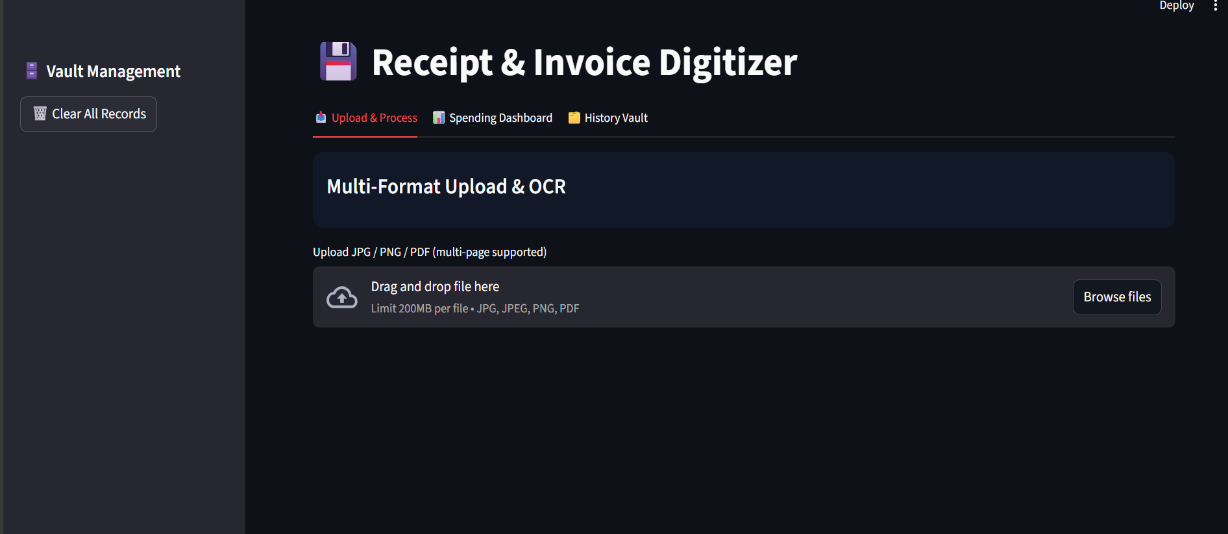
| **Technology** | **Description** |
| --- | --- |
| Python | Main programming language used for implementing the project |
| Stream lit | Web-based user interface for uploading receipts and displaying results |
| OpenCV | Image preprocessing such as grayscale conversion and noise reduction |
| PIL (Pillow) | Image handling and format conversion |
| Tesseract OCR | Text extraction from receipt and invoice images |
| PDF2Image | Conversion of PDF files into images for OCR processing |
| SQLite | Local database storage (initial setup for future milestones) |

# SYSTEM DESIGN

The user interface is developed using Stream lit. The application includes:

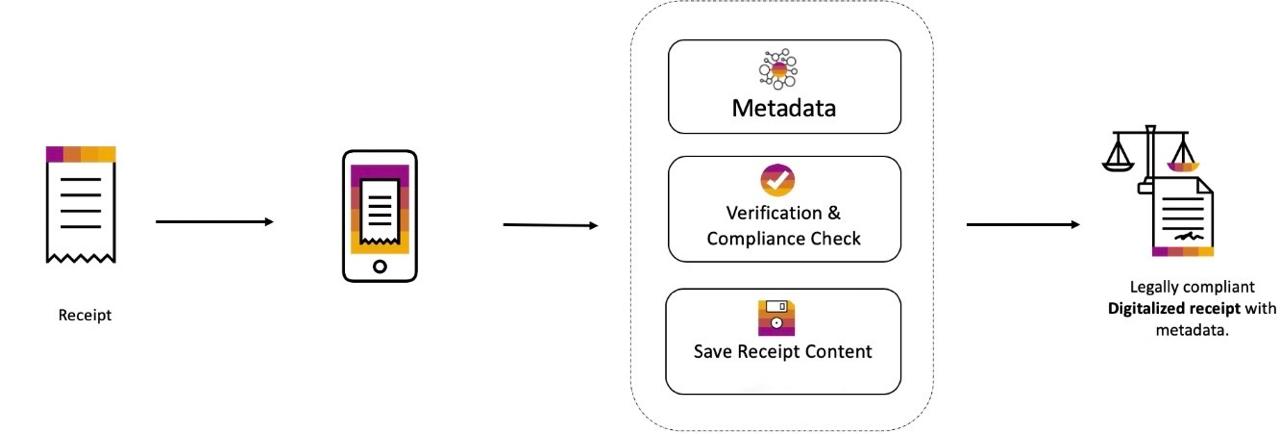
* Sidebar for vault management
* File upload section for receipts and invoices
* Display of original and pre-processed images
* Button to process and save extracted data

The design is simple and user-friendly, suitable for non-technical users.

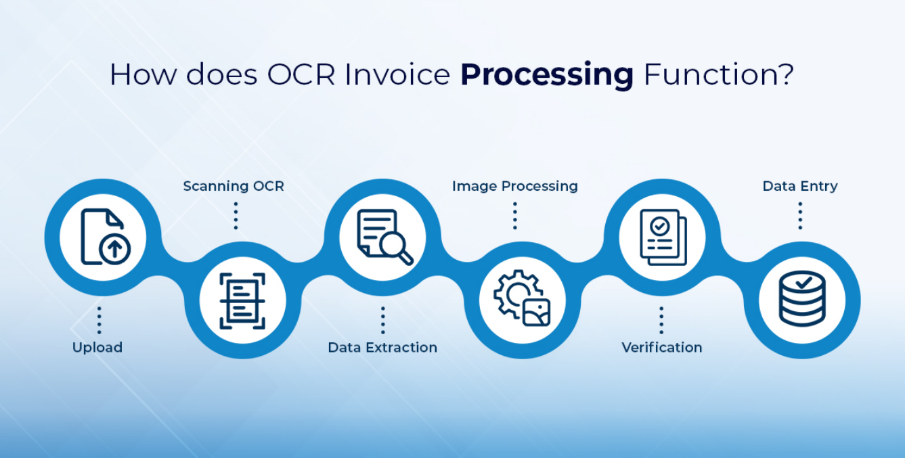




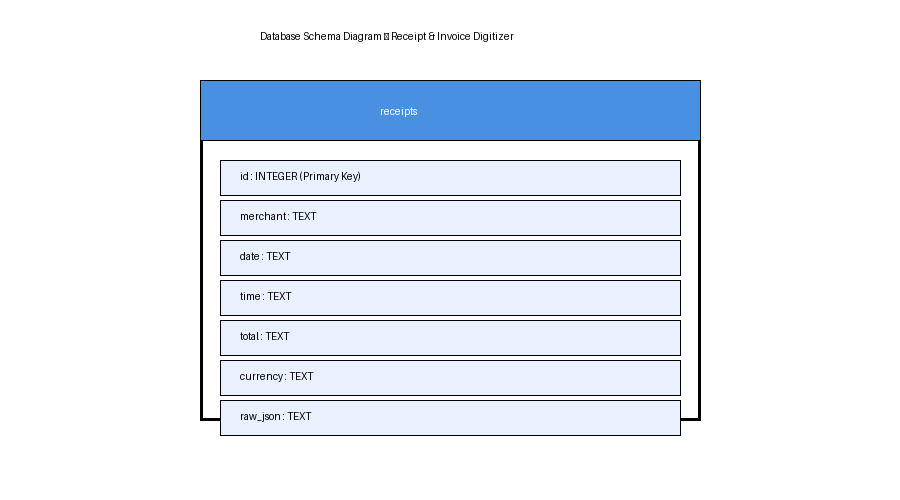
# ARCHITECTURAL DESIGN



# 6. PROCESS FLOW



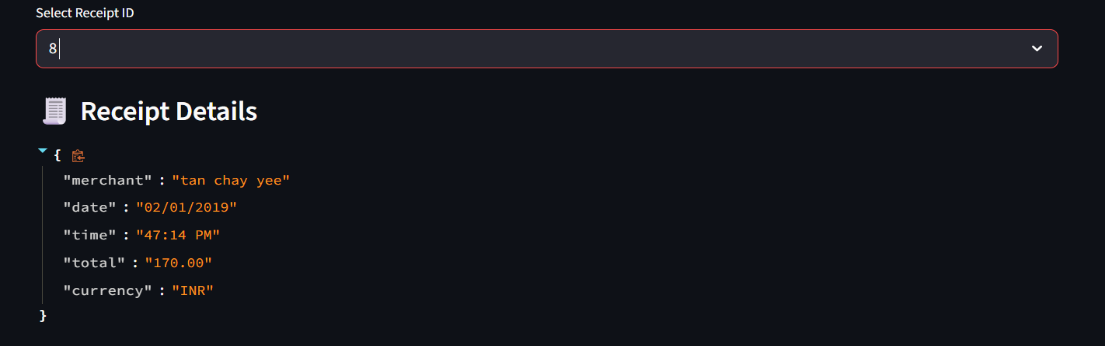
# 7. DATABASE SCHEMA



# 8. VALIDATION LOGIC

# 8.1 Total Amount Detection

* Regular expressions are used to detect keywords like "total" or "amount"
* Numeric values following these keywords are extracted



## 8.2 Duplicate Detection

* The system checks merchant name, date, and total amount
* If a record already exists, the receipt is marked as duplicate



# 9. OUTPUT

## Outputs include:

* Cleaned receipt image
* Extracted raw text
* JSON data:
* merchant
* date
* total
* currency







# 10. CONCLUSION

Milestone 1 focused on image preprocessing and text extraction from receipt and invoice images. Using OpenCV and PIL, the uploaded images were enhanced to improve OCR accuracy. Tesseract OCR was then used to extract raw text from both image and PDF files.

This milestone successfully converted receipt images into readable text, providing the base input required for further data processing. The completion of Milestone 1 laid a strong foundation for field extraction, validation, and database storage in the next milestone.

# 1. MILESTONE – 2 PROBLEM STATEMENT

Milestone 2 focuses on converting the raw text extracted in Milestone 1 into meaningful, structured, and validated data. While Milestone 1 extracts raw OCR text from receipts and invoices, Milestone 2 processes this text to identify key financial fields, validate values, detect duplicates, and store the structured data in a database.

This milestone ensures that the extracted data is accurate, reliable, and ready for analysis.

**Milestone 2 focuses on:**

* *To extract key fields from raw OCR text using regex and rule-based NLP*
* *To validate financial values such as subtotal, tax, and total*
* *To intelligently handle missing or noisy OCR data*
* *To detect and prevent duplicate receipt entries*
* *To store structured receipt data in a persistent database*

# 2. MODULES DESCRIPTION AND OBJECTIVES

## 2.1 Field Extraction Module

**Objective:** To extract meaningful receipt and invoice fields from raw OCR text.

**Description:** This module analyses the raw OCR text line by line and uses regular expressions and rule-based logic to extract important fields such as merchant name, date, subtotal, tax, total amount, and line items.

**Key Functions:**

* Merchant name detection from header text
* Date and time extraction using multiple date formats
* Subtotal, tax, and total extraction using regex
* Currency detection (INR, USD, MYR)
* Line item extraction with item name, quantity, and price

## 2.2 Validation Module

**Objective:** To ensure the accuracy and consistency of extracted financial data.

**Description:** The validation module checks whether the extracted values are logically correct. It handles missing fields gracefully and ensures that financial calculations are accurate.

**Key Functions:**

* Validation of Subtotal + Tax ≈ Total
* Automatic correction of missing values:
  + Missing tax is assumed as 0.0
  + Missing subtotal is replaced with total
  + Missing total is calculated from subtotal
* Detection of invalid or inconsistent values

## 2.3 Duplicate Detection Module

**Objective:** To prevent storing the same receipt multiple times.

**Description:** This module checks the database for existing records with the same merchant name, date, and total amount. If a match is found, the receipt is identified as a duplicate and is not saved again.

**Key Functions:**

* Duplicate check using merchant, date, and total
* Prevention of redundant database entries

## 13.4 Database Storage Module

**Objective:** To store structured and validated receipt data persistently.

**Description:** After extraction and validation, the receipt data is stored in an SQLite database. The database stores both individual fields and the full JSON representation of the receipt for traceability.

**Key Functions:**

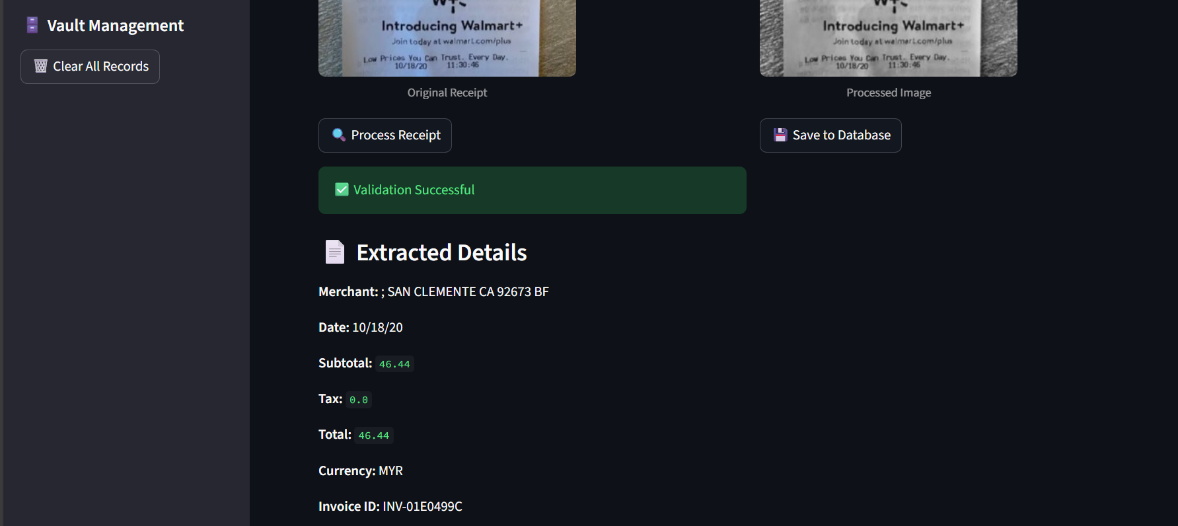
* Insert validated records into database
* Store raw JSON data for reference
* Enable future search, filtering, and analytics

# 3. TECHNOLOGIES USED

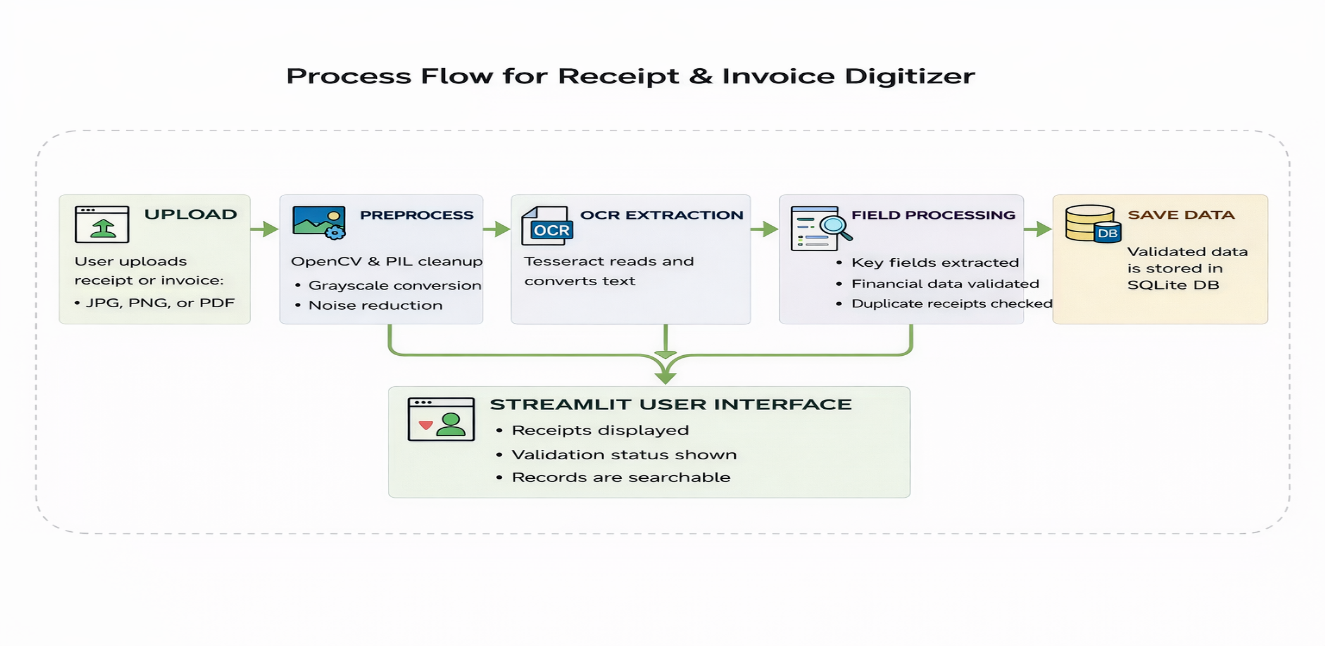
| **Technology** | **Description** |
| --- | --- |
| Python | Core programming language used for extraction, validation, and processing |
| Regular Expressions (Regex) | Used for pattern-based extraction of dates, totals, taxes, and prices |
| Natural Language Processing (NLP) | Rule-based NLP techniques used to identify merchant names, item sections, and ignore noisy text |
| SQLite | Used for persistent storage of structured receipt data |
| Pandas | Used for data manipulation, grouping, and analysis |
| Stream lit | Used to display extracted data, validation status, dashboards, and records |
| JSON | Used to store structured receipt data in serialized format |

# 4. SYSTEM DESIGN

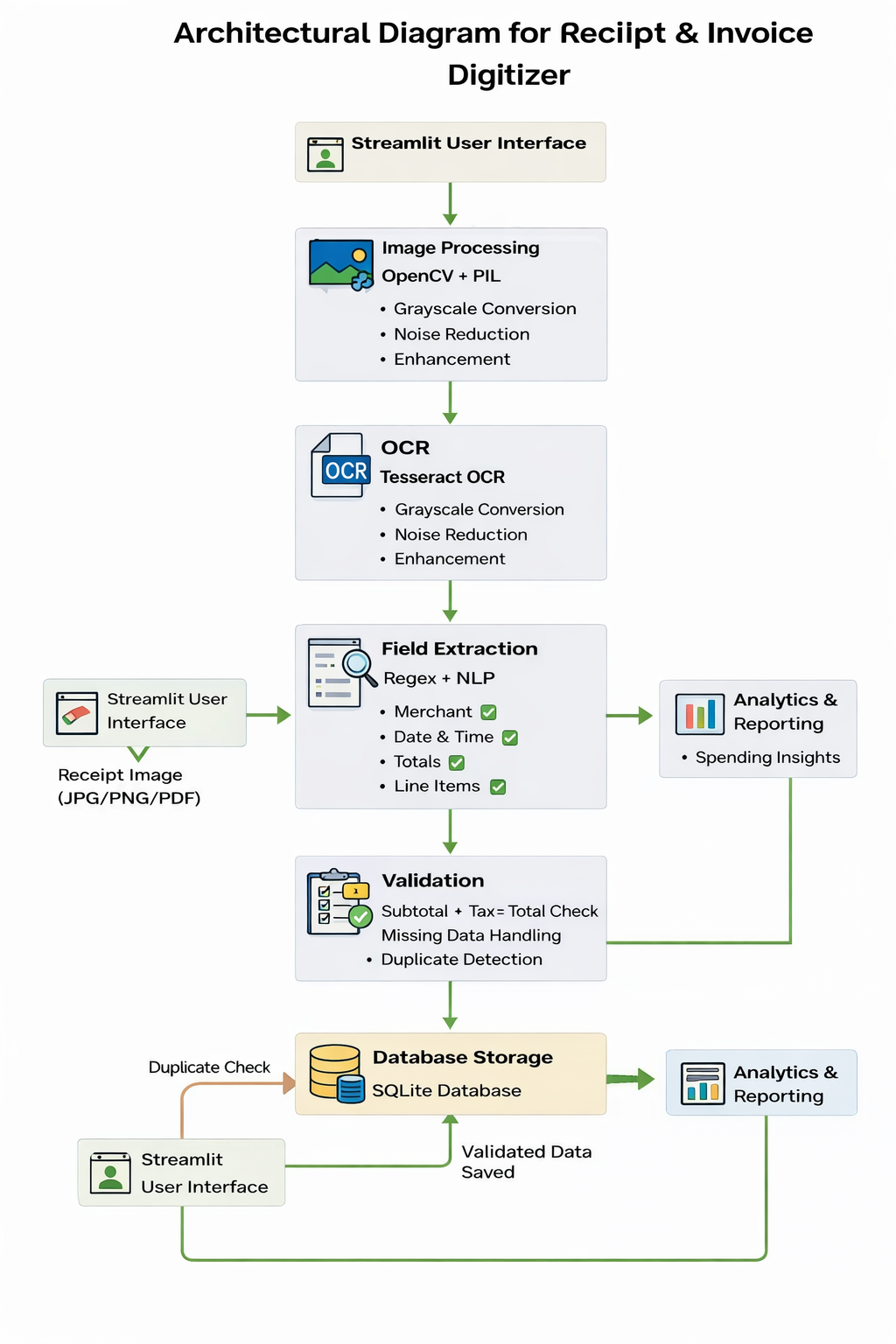
* User uploads receipt image or PDF
* Image is pre processed using OpenCV
* OCR extracts raw text from image
* Text is parsed to extract key fields
* Validation logic verifies extracted data
* Data is stored in SQLite database



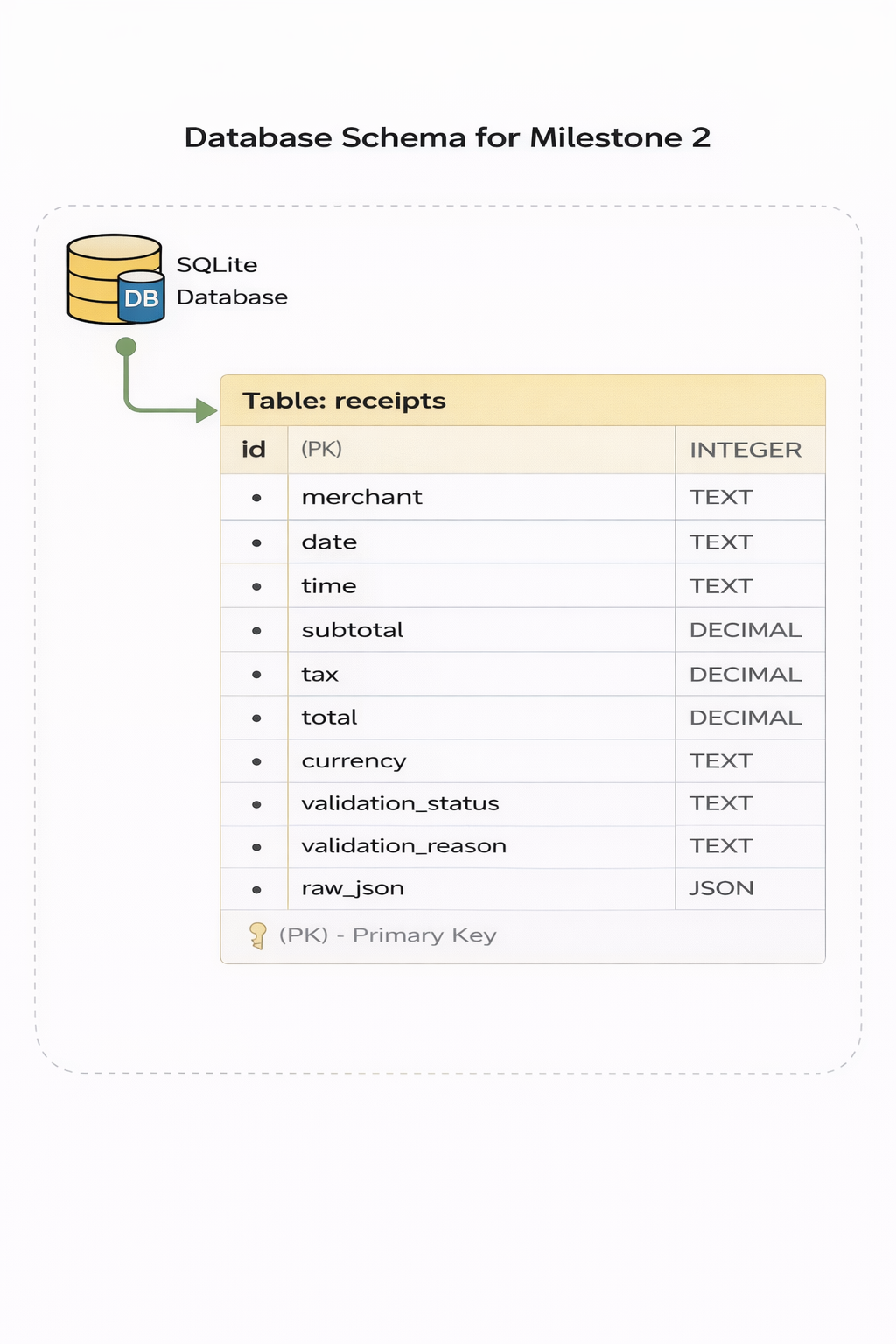
# 5. PROCESS FLOW



# 6. ARCHITECTURAL DESIGN

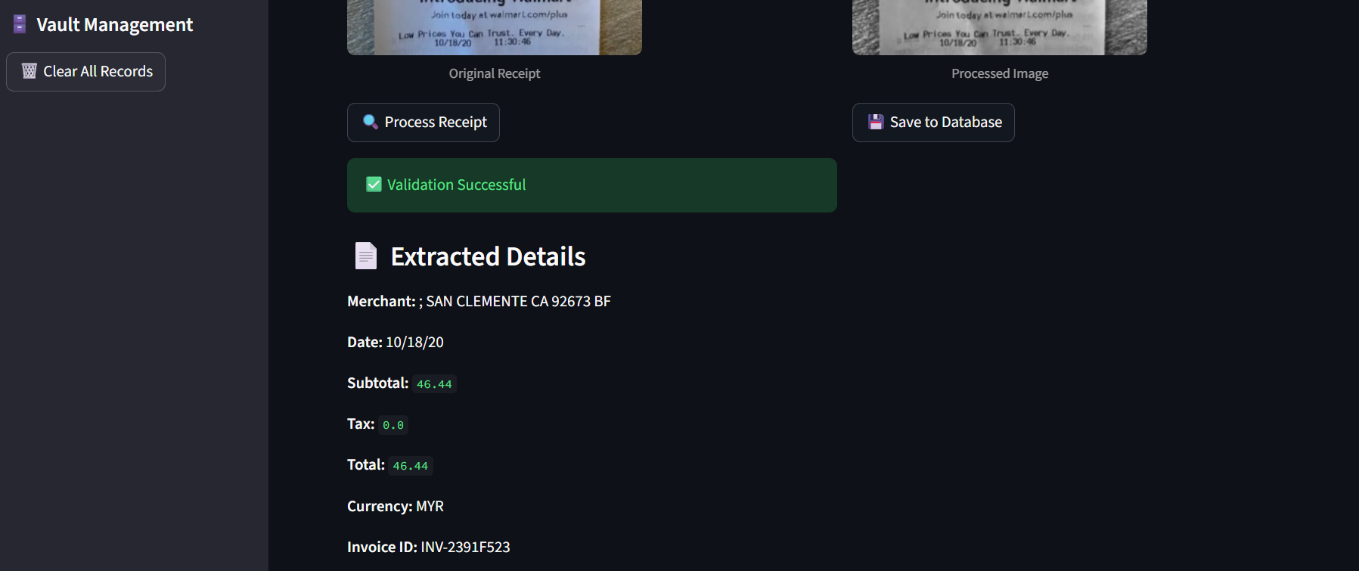


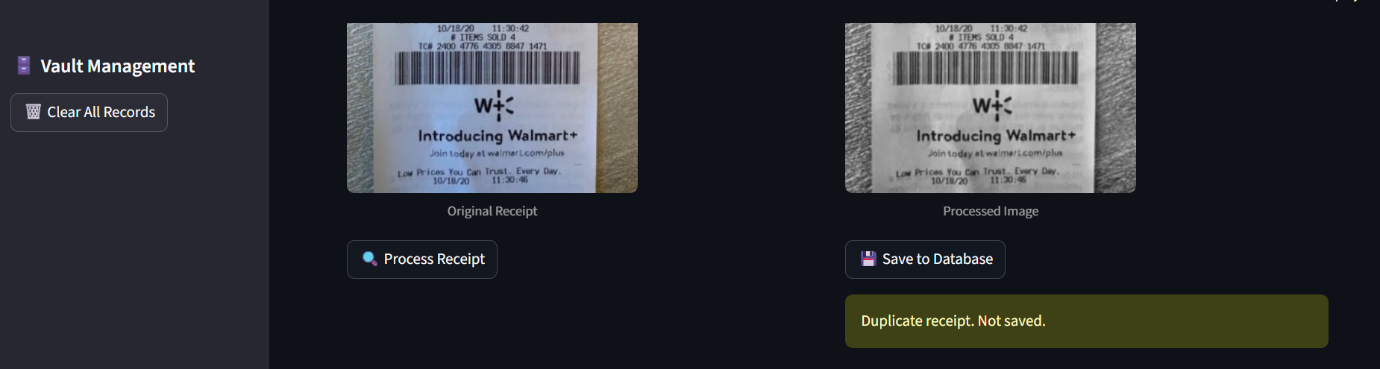
# 7. DATABASE SCHEMA



# 8. VALIDATION LOGIC

* **Total Consistency Validation**  
  The extracted data is validated by verifying that the total amount matches the logical sum of the subtotal and tax values. This ensures numerical accuracy in the extracted financial information.
* **Duplicate Receipt Detection**  
  Before storing the data, the system checks existing database records to identify duplicate receipts based on key fields. This prevents repeated storage of the same receipt.
* **Validation Status Assignment**  
  Based on the above checks, the receipt is marked as validation successful or failed and processed accordingly.

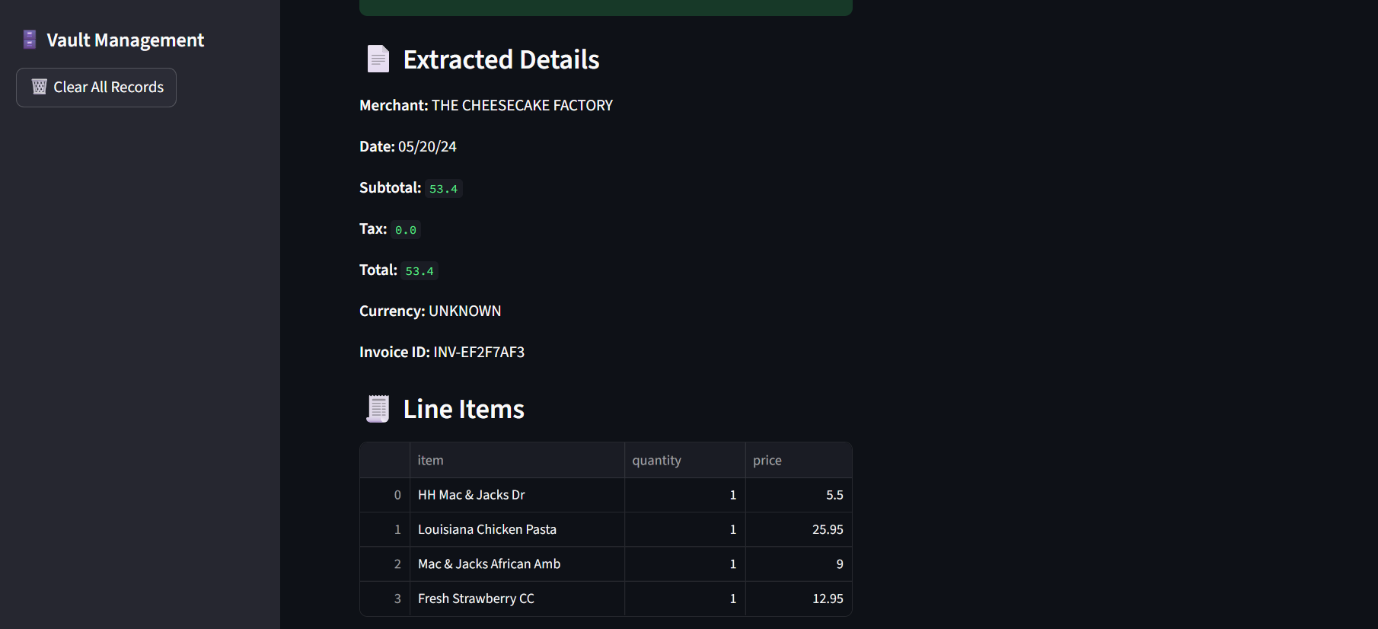


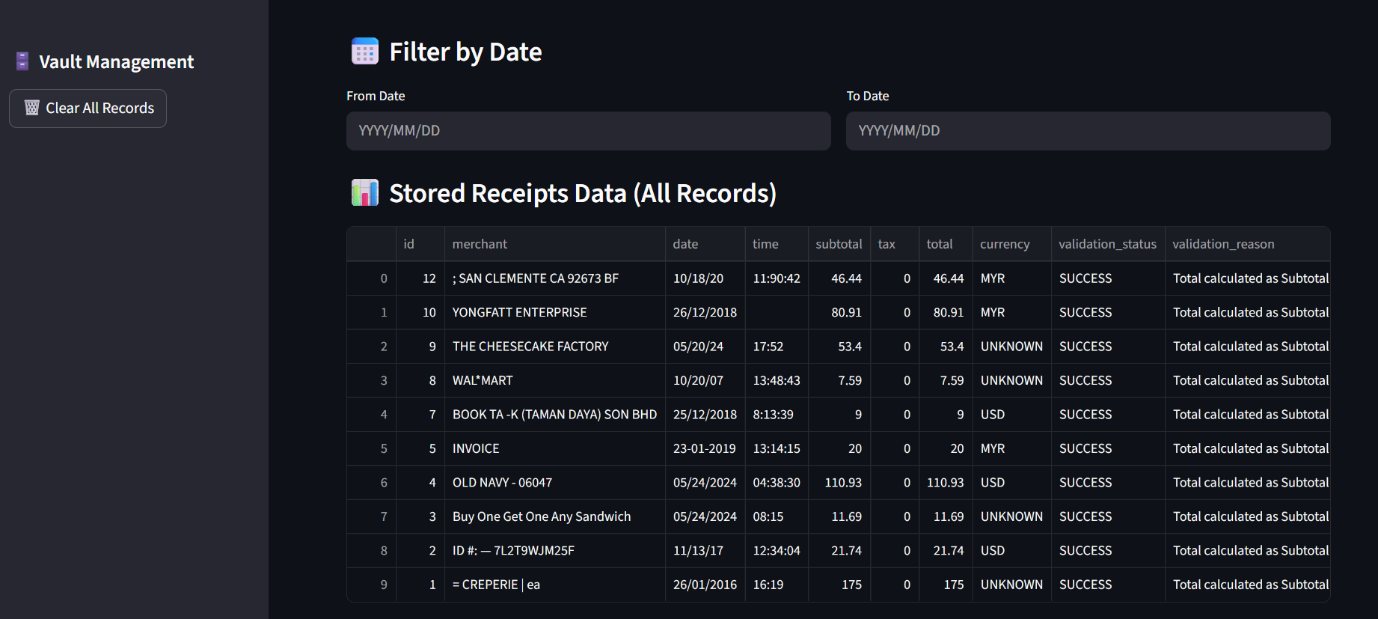


# 9. OUTPUT

The output of Milestone 2 includes:

* Extracted structured receipt data
* Validation status (Success / Failed)
* Line items with quantity and price
* Stored database records
* User-friendly display in web interface





# 10. CONCLUSION

Milestone 2 of the *Receipt and Invoice Digitizer* project successfully implemented field extraction, validation, and structured data storage. Key receipt details such as merchant name, date, subtotal, tax, total amount, and line items were accurately extracted using a combination of OCR, regex, and NLP techniques.

The validation logic ensured data correctness by verifying total consistency and preventing duplicate records before storage. All validated data was stored in a structured SQLite database, enabling efficient retrieval and analysis.

With Milestone 2 completed, the system now provides reliable and validated digital receipt data, forming a strong foundation for advanced features such as analytics, insights, and reporting in future milestones.

# 11. REFERENCES

* Smith, R., An Overview of the Tesseract OCR Engine, Google Inc.  
  Used for optical character recognition in extracting text from receipt images (Milestone 1).
* OpenCV Documentation, Open Source Computer Vision Library.  
  https://docs.opencv.org  
  Used for image preprocessing techniques such as grayscale conversion and noise reduction (Milestone 1).
* Pillow (PIL) Documentation.  
  <https://pillow.readthedocs.io>  
  Used for image handling and format conversion (Milestone 1).
* PDF2Image Documentation.  
  <https://pypi.org/project/pdf2image/>  
  Used for converting PDF receipts into images for OCR processing (Milestone 1).
* Python Software Foundation, Python Programming Language.  
  <https://www.python.org>  
  Used as the core programming language for all milestones.
* SQLite Documentation.  
  <https://www.sqlite.org/docs.html>  
  Used for structured and persistent storage of extracted receipt data (Milestone 2).
* Regular Expressions Documentation, Python re Module.  
  <https://docs.python.org/3/library/re.html>  
  Used for pattern matching to extract dates, totals, taxes, and prices from OCR text (Milestone 2).
* Natural Language Processing (NLP) – Rule-Based Text Processing.  
  Used for identifying merchant names, line-item sections, and filtering irrelevant OCR text (Milestone 2).
* Stream lit Documentation.  
  <https://docs.streamlit.io>  
  Used for building the interactive web-based user interface across all milestones.
* Pandas Documentation.  
  <https://pandas.pydata.org/docs/>  
  Used for data manipulation, grouping, and spending analysis (Milestone 2).