1. Title

Receipt and Invoice Digitizer

---

2. Introduction

In the modern financial landscape, the management of physical receipts and invoices is often a bottleneck for both businesses and individuals. Traditional methods of handling these documents involve manual data entry, which is time-consuming, repetitive, and highly susceptible to human error. Furthermore, physical paper trails are prone to loss and deterioration, making historical tracking and audit compliance difficult.

The **Receipt and Invoice Digitizer** is a robust software solution designed to bridge the gap between physical documentation and digital accounting systems. This project aims to automate the end-to-end process of document management by transforming unstructured image data into structured, queryable financial records.

By leveraging advanced **Optical Character Recognition (OCR)** technologies (such as Tesseract or Google Vision) combined with **Natural Language Processing (NLP)** and Regex-based parsing, the system intelligently extracts critical data points—including vendor names, transaction dates, invoice IDs, and monetary values.

Beyond simple extraction, the system incorporates a layer of **logic and validation** to ensure mathematical consistency (e.g., verifying that subtotal plus tax equals the total). The processed data is securely stored in a relational **SQL database**, enabling users to generate insightful analytics, track spending trends via an interactive **Web Dashboard**, and seamlessly export data for further financial reporting.

---

3. Problem Statement

i) The Core Issue

Businesses and individuals currently rely heavily on manual workflows to manage high volumes of paper-based receipts and invoices. This dependence on physical documentation and manual data entry results in a slow, error-prone, and inefficient financial tracking process that is difficult to scale and integrate with modern digital accounting systems.

ii) Operational Challenges

The current manual approach presents several distinct challenges that this project aims to resolve:

Inefficient Data Entry: Manually transcribing data (Vendor Name, Date, Invoice IDs, Line Items) from paper to digital spreadsheets is time-consuming and labor-intensive.

High Risk of Human Error: Manual entry is susceptible to typos, incorrect decimal placements, and calculation errors, leading to inaccurate financial reporting.

Data Organization & Retrieval: Physical receipts are easily lost, faded, or damaged over time. Searching for a specific historical transaction within a pile of paper or unstructured image folders is virtually impossible.

Lack of Validation: Without an automated system, it is difficult to instantly verify if an invoice total matches the subtotal plus tax, or to detect if a receipt has been submitted twice (duplicate entry), leading to potential financial leakage.

Unstructured Data: Raw images of receipts are "unstructured data." Without digitization, this data cannot be easily analyzed for spending trends, vendor statistics, or monthly analytics

.

4. Objectives

The primary objective of **Milestone 1** is to design and implement a **robust, secure, and extensible document digitization foundation** capable of reliably converting physical receipts and invoices into structured digital data. This milestone focuses on establishing the **core technical pipeline** that will support all subsequent enhancements such as analytics, reporting, and long-term data persistence.

Milestone 1 is intentionally scoped to emphasize **correctness, reliability, and architectural soundness**, ensuring that downstream milestones can be built without refactoring core components. The system is designed to operate consistently across multiple document types while maintaining high OCR accuracy, predictable behavior, and a user-friendly interaction model.

Key Objectives

1. Multi-Format Document Ingestion

Enable secure and reliable ingestion of common receipt and invoice formats, including **JPG, PNG, and PDF files**. The system must correctly identify file types, validate file size and integrity, and handle both single-page and multi-page documents without manual intervention.

1. Standardized Image Conversion Pipeline

Convert all supported document formats into a **uniform, OCR-ready image representation**. PDFs must be safely rendered into individual page images, and all image outputs must be standardized (RGB format, consistent resolution) to ensure predictable downstream processing.

2. Automated Image Preprocessing for OCR Optimization

Implement an automated preprocessing layer that enhances OCR performance by applying operations such as **grayscale conversion, noise reduction, contrast enhancement, and binarization**. This preprocessing must run transparently in the background and be resilient to variations in document quality.

3. Single-Call OCR and Structured Data Extraction

Integrate **Google Gemini AI** to perform OCR and semantic understanding in a **single API call per document/page**. The system must extract not only raw text but also **structured bill data**, including vendor details, dates, line items, tax values,

totals, currency, and payment method.

4. Strict Schema Enforcement and Data Normalization

Enforce a predefined JSON schema on all extracted data to ensure consistency and database compatibility. Missing or ambiguous values must be handled using safe defaults, and numeric fields must be normalized to valid data types to prevent downstream failures.

5. Session State Management and Workflow Continuity

Maintain application state across Streamlit reruns using structured session state management. This ensures that uploaded files, processed images, extracted results, and user actions persist seamlessly during navigation, reducing redundant computation and improving user experience.

6. User-Centric and Intuitive Interface Design

Provide a **clean, minimal, and user-friendly interface** that clearly guides users through upload, processing, and result inspection. The UI must support image previews, extracted data visualization, and logical navigation without overwhelming the user.

7. Controlled Error Handling and Fault Tolerance

Implement comprehensive error handling across ingestion, preprocessing, OCR, and extraction stages. Failures must be **graceful, informative, and non-destructive**, ensuring that partial errors do not crash the application or corrupt session state.

8. Foundation for Persistent Storage and Analytics

Although advanced analytics and reporting are reserved for later milestones, Milestone 1 establishes the **data structures, schemas, and interfaces** required for seamless integration with persistent storage systems and analytical dashboards in future phases.

Outcome of Milestone 1

By the completion of Milestone 1, the system delivers a **production-ready core digitization pipeline** capable of transforming unstructured receipt and invoice documents into reliable, structured digital records.

---

5. Technology Stack

1. Programming Language

Python 3.x: The primary language for the entire backend, OCR processing, and frontend dashboard. It was chosen for its rich ecosystem of libraries like OpenCV and Pandas that simplify complex tasks.

2. Web Framework & User Interface

Streamlit: Used to build the interactive web dashboard. Streamlit allows for the rapid creation of data apps, enabling users to upload files, view extracted data, and visualize analytics without complex HTML/CSS coding.

(Alternative) **Flask:** A lightweight web framework used if a custom REST API backend is required to serve the OCR model to other applications.

3. Optical Character Recognition (OCR) Engine

Tesseract OCR: An open-source OCR engine developed by Google. It is the primary tool used to recognize and extraction text characters from the processed images.

Google Cloud Vision API (Optional): A cloud-based alternative used for higher accuracy on complex, low-quality, or handwritten receipts where Tesseract might struggle.

4. Image Processing & Preprocessing

OpenCV (cv2): The computer vision library used to "clean" images before they are passed to the OCR engine. Key functions include:

Grayscaling: Reducing color complexity.

Thresholding/Binarization: Converting images to strictly black and white to separate text from the background.

Denoising: Removing grain and visual noise from scanned papers.

Pillow (PIL): Used for basic image manipulation, such as opening file formats, resizing, and cropping.

pdf2image: A utility library to convert multi-page PDF invoices into image formats (JPG/PNG) so they can be processed by the OCR engine.

5. Natural Language Processing (NLP) & Data Extraction

Regular Expressions (`re`): The core logic for "Field Extraction." Regex patterns are defined to identify and capture specific formats like Dates (`DD/MM/YYYY`), Currency (`$`, `₹`), and Email addresses.

SpaCy / NLTK: Advanced NLP libraries used to detect entities (Named Entity Recognition) like Vendor Names (Organizations) or Locations if simple pattern matching is insufficient.

6. Database & Storage

SQL (SQLite / PostgreSQL):

SQLite: Used during development for a lightweight, serverless database to store extracted invoice records.

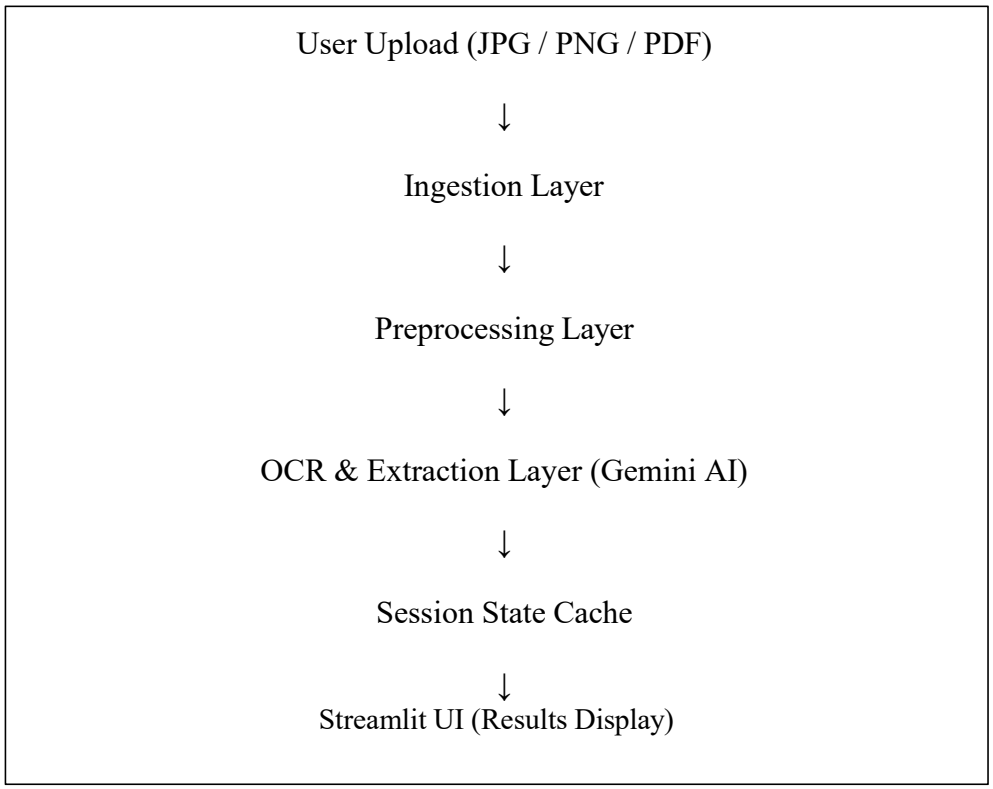
PostgreSQL: (Recommended for production) A robust relational database to handle large volumes of invoice data and complex queries.

SQLAlchemy: An Object Relational Mapper (ORM) used to interact with the database using Python code instead of writing raw SQL queries.

7. Data Manipulation & Analysis

Pandas: Used to organize the extracted text into structured tables (DataFrames), perform calculations (e.g., verifying totals), and export the final data to CSV or Excel formats.

6. System Architecture



7. Modules included

1. Document Ingestion

This is the entry point of the system, responsible for accepting files and preparing them for processing.

File Upload: Users can upload receipts and invoices in various formats, such as PDFs or standard image files (JPG, PNG).

Preprocessing: Before text can be read, the image is cleaned up. This involves resizing, denoising (removing graininess), and binarization (converting to black and white) to ensure the OCR engine gets the clearest possible input.

2. OCR & Text Extraction

Once the image is clean, this module focuses on turning the visual data into machine-readable text.

Core Technology: It utilizes **Tesseract OCR** or the **Google Vision API** to scan the document.

Layout Handling: It is designed to handle complex formatting, such as skewed scans (crooked images), multi-column layouts, or tabular data often found on invoices.

Output: The result is "raw text"—a continuous stream of characters extracted from the image.

3. Field Extraction & Validation

This is the "brain" of the project. It takes the raw text from the previous step and makes sense of it.

Parsing: Using **NLP (Natural Language Processing)** and **Regex (Regular Expressions)**, the system identifies specific data points:

Dates, Vendor Names, Invoice IDs

Tax amounts, Line items, and Total amounts

Validation Logic: It performs math checks (e.g., ensuring Subtotal + Tax = Total) to verify accuracy.

Duplicate Detection: It checks invoice numbers or generates hashes to ensure the same receipt isn't processed twice.

4. Database Storage

After the data is cleaned and validated, it needs a permanent home.

Structure: The extracted data is stored in a structured **SQL database**.

Searchability: This allows users to query the data later, such as searching for all receipts from a specific "Vendor" or finding expenses within a certain date range or amount.

5. Dashboard & Reporting

This is the user interface (UI) module where the user interacts with the system.

Web Dashboard: Built using **Streamlit** or **Flask**, this interface allows users to upload files and manually review the extracted data.

Exporting: Users can download their data in **CSV** or **Excel** formats for use in other software.

Analytics: The dashboard provides visual summaries, such as monthly spending trends or vendor-specific statistics.

---

8. **Timeline for Milestone 1** (2Weeks)

Milestone 1: Document Ingestion & OCR

Goal: Build the foundation of the application where users can upload files, and the system can process images to output raw text.

Week 1: Infrastructure & File Ingestion

Focus: Setting up the development environment and handling user inputs.

Day 1-2 Project Setup

• Initialize Git repository.

• Set up Python environment (Virtualenv/Conda).

• Install core dependencies (Streamlit/Flask, OpenCV, Pytesseract).

Day 3-4 Module 1: File Upload UI

• Create a basic web interface using Streamlit or Flask.

• Implement the "File Uploader" widget.

• **Validation:** Ensure only valid formats (PDF, JPG, PNG) can be uploaded.

Day 5 File Handling Backend

• Write backend logic to save uploaded files to a temporary directory.

• **Feature:** Convert PDF uploads to images (using pdf2image) so they are ready for OCR processing.

Week 2: Preprocessing & OCR Integration

Focus: Image manipulation and connecting the Optical Character Recognition engine.

Day 1-2 Module 1: Image Preprocessing

• Implement OpenCV pipeline.

• **Grayscale:** Convert images to black and white.

• **Denoise:** Remove "salt and pepper" noise.

• **Binarization:** Apply thresholding to make text pop against the background.

Day 3-4 Module 2: OCR Integration

• Connect **Tesseract OCR** (or Google Vision API).

• Create a function extract_text_from_image(image) that takes the preprocessed image and returns a string.

• Handle basic skewed layouts.

Day 5 Testing & Validation

• **Unit Testing:** Run the pipeline on 5-10 sample receipts.

• Check the quality of the "Raw Text" output.

• Tweak preprocessing parameters (contrast/brightness) if OCR misses text.

---

9. Conclusion

Milestone 1 successfully delivers a production-grade foundation for receipt and invoice digitization. The system demonstrates strong architectural separation, reliable preprocessing, accurate OCR extraction, and a user-friendly interface.

This milestone lays a solid technical base for:

- Persistent storage optimization
- Advanced analytics
- Multi-user workflows
- Enterprise integrations