BillBuddy: Receipt and Invoice analyzer

1. Abstract:

Businesses and individuals handle numerous paper receipts and invoices, which are prone to loss, errors, and manual entry delays. This project builds a system that automatically scans, extracts, and digitizes information from receipts and invoices using OCR (Optical Character Recognition) and NLP-based field extraction. The digitized data is stored in a structured format, making it easy to search, analyze, and integrate with accounting or ERP systems.

2. Introduction

2.1. What does a Receipt and Market Analyzer do?

A Receipt and Market Analyzer is a dual-function business intelligence tool that bridges the gap between internal sales data and external industry trends. It serves as a comprehensive system for businesses to track their immediate financial performance while simultaneously evaluating the broader competitive landscape to inform strategic planning.

Key components of this system include:

- Receipt Analysis: This function automates the processing of transactional data, such as Oracle's Receipt Analysis reports, to track revenue trends, payment behaviors, and customer loyalty metrics.
- Market Analysis: This component provides a bird's-eye view of the industry, assessing market size, growth potential, and competitor movements. Tools like Statista are often integrated to provide global market statistics and consumer data trends.
- Actionable Insights: By combining these data points, the analyzer identifies "unmet needs" and "market gaps," allowing businesses to pivot their product lines or marketing strategies based on real-time evidence rather than guesswork.
- Risk Mitigation: It acts as a decision-making tool that helps entrepreneurs validate new ideas, set realistic revenue benchmarks, and identify emerging threats.

2.2. Economic Relevance:

In 2026, these analyzers increasingly utilize AI-powered tools and big data to enable real-time monitoring of customer preferences and market shifts, ensuring businesses remain adaptive in volatile environments.
The invoice OCR API market stands at ~$1.5B in 2024, projected to reach $5.8B by 2033 (CAGR 16.8%), with AI receipt automation growing from $3.5B to $86.4B by 2034 (CAGR 37.8%). Businesses gain 60% efficiency via 95%+ OCR accuracy, reducing manual entry costs. Pricing ranges $300-450/1000 docs, favoring volume users; TAM for OCR hits $50B by 2034.

2.3. Limitations of Existing Solutions

Existing OCR-based receipt and invoice analyzers face significant hurdles in accuracy, adaptability, and usability, often requiring substantial human intervention that undermines full automation.

- **Accuracy Challenges:** Existing systems misread 1 in 10 characters, especially on low-quality scans, faded prints, handwritten notes, or crumpled paper.
- **Handling Variability:** layouts vary wildly by vendor, country, and format, causing keyword search failures and field misidentification.
- **Manual Intervention and Costs:** Even top tools like standalone OCR demand regular manual corrections for 10-15% of documents, inflating operational costs and slowing workflows.

3. Ease of my solution

The application allows users to upload invoices in various formats (PNG, JPG, PDF), automatically extracts key fields such as vendor name, invoice ID, date, tax, and total amount, and validates the extracted data for consistency. It detects potential duplicates, stores all processed invoices in a personal history vault, and provides interactive analytics including spending trends, vendor comparisons, and category breakdowns. Users can review raw text, view preprocessed images, manage their invoice history, and even ask natural language questions about their bills via an integrated AI chatbot.

The pipeline begins by converting uploaded files—whether images or PDFs—into high-quality grayscale images using pdf2image (for PDFs) and OpenCV-based preprocessing (denoising, adaptive thresholding). Tesseract OCR then extracts raw text, which is parsed using regex-based rules to identify structured fields. All results are saved in a SQLite database scoped per user, enabling persistent storage and history management. The frontend, built with Streamlit, orchestrates file handling, visualization (Matplotlib/Seaborn), and user interactions, while a Gemini-powered chat interface provides conversational support. The entire system is modular:

core logic resides in ocr_pipeline.py, data persistence in database.py, and UI in app.py—ensuring maintainability and extensibility.

3.1. OCR Extraction Pipeline

Our system uses a robust, multi-stage OCR pipeline optimized for real-world invoice documents. First, uploaded files—whether images or PDFs—are converted into high-quality grayscale images. For PDFs, we use pdf2image at 200 DPI to balance speed and accuracy. Each image then undergoes OpenCV-based preprocessing: noise is reduced with Gaussian blur, and adaptive thresholding creates a clean black-and-white version ideal for text recognition. Tesseract OCR (with PSM mode 1 for automatic layout detection) extracts raw text, which is then parsed using regex patterns to identify structured fields like vendor, invoice ID, date, tax, and total amount. This approach handles diverse layouts while maintaining high accuracy—even on skewed or low-contrast documents.

3.2. Data Visualization and analysis

Once invoices are processed, the system transforms raw data into actionable insights through interactive visualizations. Using Pandas, Matplotlib, and Seaborn, we generate five key chart types:

i. Bar charts compare spending across vendors,
ii. Line charts reveal monthly cost trends,
iii. Pie charts show category-wise expenditure,
iv. Scatter plots (with simulated quality scores) explore price-performance relationships,
v. Heatmaps visualize vendor performance matrices.
All charts are dynamically generated from the user's personal invoice history, enabling quick financial oversight without manual analysis.

3.3. Database Management

Data persistence is handled via a lightweight yet secure SQLite database, ensuring each user's invoices remain private and accessible across sessions. Every processed invoice—along with its raw text, extracted fields, validation status, and original file bytes—is stored in a structured schema. The system supports full CRUD operations: new invoices are saved automatically after upload, the history vault displays all entries in a searchable table, and users can delete one or multiple records with a single click. Crucially, the database design includes resilience features like short-lived connections and error handling to prevent "database locked" issues in concurrent environments like Streamlit.

3.4. UI/UX design

The frontend is built entirely in Streamlit, offering a clean, intuitive, and responsive interface. The app is organized into logical tabs: Upload, History Vault, Analytics Dashboard, and AI Chat Support. Users can drag-and-drop mixed file types (PDF + images), instantly see original vs. preprocessed previews, and review extracted text. The history vault provides tabular browsing with bulk-delete functionality, while the analytics tab delivers publication-ready visualizations. A sidebar login ensures data isolation, and session state management maintains context during interactions. The entire UI prioritizes clarity, feedback (success/error messages), and zero-config usability—making advanced document intelligence accessible to non-technical users.

3.5. Field Extraction.

The system intelligently parses unstructured invoice documents using a hybrid approach that combines regular expressions for predictable patterns (like invoice IDs and monetary amounts) with spaCy-powered Named Entity Recognition (NER) to understand contextual elements such as vendor names and dates. This dual strategy ensures robustness across diverse layouts: regex captures structured data reliably, while NER interprets semantic meaning even when formatting varies. Additional heuristics—such as scanning text regions before keywords like "Invoice" or "Total"—further improve accuracy, especially when OCR output is noisy or incomplete.

3.6. Duplicate Detection

Duplicates are reliably identified using three complementary techniques that work together to ensure robustness across real-world scenarios. First, perceptual image hashing detects visually identical or near-identical uploads—such as the same invoice re-uploaded as a screenshot or scan—by comparing structural image features rather than raw pixels. Second, normalized text fingerprinting creates a compact, noise-resistant signature from the extracted text, enabling matches between different file formats (e.g., a PDF and a photo of the same invoice), even if OCR introduces minor errors. Third, field-based matching cross-references key structured fields—vendor name, invoice ID, and total amount—with a ±2% tolerance for numerical variance to accommodate rounding or tax differences. By combining visual, textual, and semantic signals, this layered approach maintains data cleanliness without depending solely on potentially fragile OCR output, making the system resilient to real-world document variability.

3.7. Checking Validation

All processed invoices are stored in a secure, user-isolated SQLite database, enabling persistent, private access to financial records. Each entry preserves the original file bytes, extracted fields, validation status, user-added notes, and metadata like upload time—creating a rich, queryable archive. The schema supports efficient history browsing, bulk management, and future analytics, while maintaining strict separation between users. This structured yet lightweight storage layer forms the foundation of the personal “History Vault,” turning transient documents into long-term business assets.

3.8. Structured SQLite Database.

Every extracted invoice is automatically validated for data integrity to ensure reliability and prevent clutter. The system first checks whether the total amount is plausible—confirming it is a positive number within a realistic financial range (e.g., not zero, negative, or unrealistically large). Then, a multi-layered duplicate detection mechanism scans existing records to avoid redundant entries. This includes comparing perceptual image hashes (to catch re-uploaded scans), normalized text fingerprints (to identify the same invoice in different formats like PDF vs. photo), and structured field matches (vendor name, invoice ID, and total amount within a 2% tolerance). Together, these checks maintain a clean, trustworthy history vault—free from duplicates and obvious errors—even when OCR quality varies.

Core Features

1. Multi-format Invoice Upload

Users can seamlessly upload invoices in any common format — including scanned images (PNG, JPG) and multi-page PDFs — all in a single batch. The system intelligently handles each file type: PDFs are converted to high-resolution images using pdf2image, while photos are optimized through preprocessing. This flexibility ensures compatibility with real-world document sources, from email attachments to mobile snapshots.

2. Fully Automated OCR Pipeline

From raw pixels to structured data — the entire extraction process is zero-touch. Using OpenCV for noise reduction and adaptive binarization, followed by Tesseract OCR with layout-aware segmentation, the pipeline accurately captures text even from skewed, low-contrast, or cluttered invoices. Key fields like vendor, invoice ID, date, tax, and total amount are automatically parsed using robust regex patterns — no manual labeling or templates required.

3. Personal History Vault

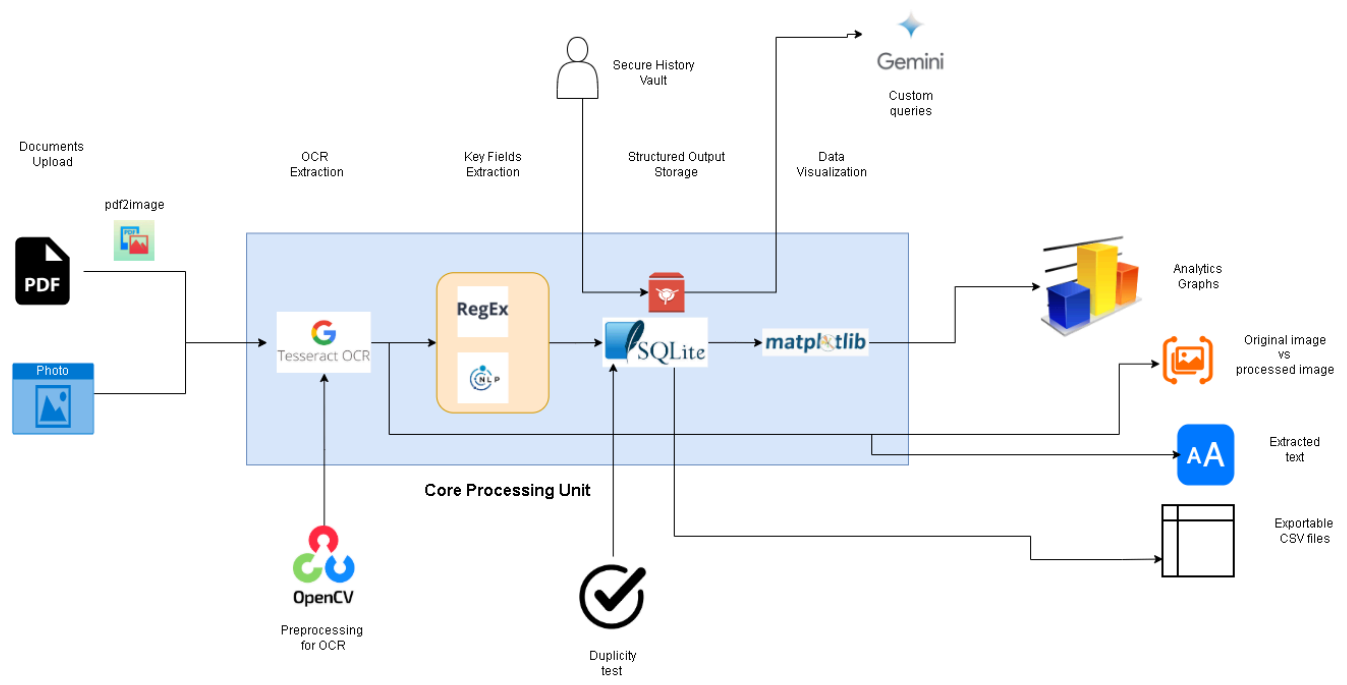
Every processed invoice is securely stored in a user-isolated database, creating a private, searchable archive of financial records. Users can review past uploads, compare extracted fields, delete unwanted entries in bulk, and revisit raw text or preprocessed previews at any time. This persistent vault turns transient documents into long-term business assets — all without leaving the app.

4. Modular Architecture

Built for clarity and maintainability, the system follows a clean separation of concerns:

- app.py handles UI and user interaction,
- ocr_pipeline.py encapsulates all computer vision and text extraction logic,
- database.py manages storage and retrieval,
- analytics.py powers visual insights.
  This modularity makes the codebase easy to debug, extend, and adapt — whether adding new field extractors, swapping OCR engines, or integrating cloud storage.

4. Architecture diagram



This architecture illustrates how raw invoice documents — whether scanned photos or multi-page PDFs — are transformed into structured, actionable business intelligence through a seamless, automated workflow.

1. Input Layer: Documents Upload

Users can seamlessly upload invoices in various common formats, including both photo images and multi-page PDFs. The system intelligently handles these file types: photo images are accepted directly, preserving flexibility for quick mobile captures or screenshots. For PDF documents, the system uses the pdf2image tool to convert each page into a high-resolution image, which ensures optimal compatibility with the subsequent Optical Character Recognition (OCR) engines.

- Users upload PDFs or photo images of invoices.
- For PDFs, the system uses pdf2image to convert each page into high-resolution images — ensuring compatibility with OCR engines.
- Photos are accepted directly, preserving flexibility for mobile captures or screenshots.

2. Preprocessing (OpenCV)

- Before OCR, images undergo preprocessing using OpenCV:
  - Noise reduction via Gaussian blur
  - Adaptive thresholding to create clean black-and-white versions

    - Optional deskewing for rotated documents
- This step dramatically improves OCR accuracy — especially on low-quality scans or photos.
- Noise Reduction: A gentle Gaussian blur smooths out grain, speckles, and compression artifacts—common in mobile photos or faxed documents—without blurring essential text edges.
- Adaptive Thresholding: Converts grayscale images into high-contrast black-and-white versions that preserve text legibility even under uneven lighting or shadowed regions. Unlike global thresholding, this method dynamically adjusts to local brightness, ensuring consistent results across complex layouts.
- Optional Deskewing: Automatically detects and corrects slight rotations (e.g., from handheld scans), aligning text horizontally to match Tesseract's layout assumptions.

3. OCR Extraction (Tesseract)

The process begins with the Preprocessing (OpenCV) stage, where images are meticulously prepared for recognition: noise is reduced with a Gaussian blur, and adaptive thresholding creates clean black-and-white versions to dramatically improve accuracy, especially on low-quality scans. Following this preparation, the preprocessed images are fed into Tesseract OCR, a powerful open-source text recognition engine. Tesseract extracts the raw text from the entire document, preserving layout structure where possible. Finally, this raw text moves to the Key Fields Extraction (RegEx + NLP) stage, where it is parsed using robust regular expressions to identify critical invoice fields (Vendor name, Invoice ID, Date, Tax amount, Total amount). This structured data is then validated, checked for duplicates, and saved to the Structured Output Storage (SQLite) before being used for Data Visualization.

4. Key Fields Extraction (RegEx + NLP)

- Raw text is parsed using regular expressions to identify critical invoice fields:
    - Vendor name
    - Invoice ID
    - Date
    - Tax amount
    - Total amount
- A placeholder for NLP (Natural Language Processing) indicates future expansion — e.g., using spaCy to extract named entities or handle unstructured text.
- Current State: Field extraction relies primarily on regex patterns, which work well for structured layouts but struggle with unstructured or noisy invoice formats.

- NLP Integration (Planned): Replace or augment regex with spaCy's pre-trained models to perform Named Entity Recognition (NER)—automatically identifying vendors, dates, currencies, and organizations from raw text.
- Context-Aware Parsing: NLP will enable understanding of semantic context (e.g., distinguishing "Total Solutions Inc." as a vendor vs. "Total Amount" as a label), reducing false positives.
- Handling Unstructured Text: For invoices without clear tables or labels, NLP can infer fields from sentence structure (e.g., "Billed to ABC Corp on Jan 15, 2026" → vendor + date).
- Custom Entity Training: Future versions could fine-tune spaCy models on invoice-specific data to recognize domain terms like "PO Number," "Due Date," or "Tax ID."
- Multilingual Support: With appropriate language models, the system could process invoices in multiple languages without changing core logic.
- Confidence Scoring: NLP pipelines can provide confidence scores per field, enabling smarter validation and user alerts for low-certainty extractions.

5. Validation & Duplication Test

- Extracted data is validated:
    - Is the total amount reasonable?
    - Does it match line item sums (when available)?
- A duplicate detection module compares new invoices against existing records using:
    - Same vendor + invoice ID
    - Similar total amount (±2% tolerance)
- Ensures data integrity and prevents redundant entries.

6. Structured Output Storage (SQLite)

- All processed data — including raw text, extracted fields, validation status, and original file bytes — is stored in a SQLite database.
- Each user has their own isolated dataset, enabling secure, persistent storage without heavy infrastructure.
- Ideal for small to medium-scale deployments — fast, lightweight, and zero-config.
- The original file bytes (whether PDF or image), enabling reprocessing or preview at any time

- The full raw OCR output, preserving unaltered text for audit or re-analysis
- Structured extracted fields (vendor, invoice ID, date, tax, total amount, etc.) in JSON format
- Validation metadata, such as confidence scores, duplicate flags, and plausibility checks
- User-provided annotations, like custom notes or tags

7. Data Visualization (Matplotlib)

Once invoices are processed through the OCR Extraction Pipeline and the Key Fields Extraction (RegEx + NLP), the system validates the data and saves it to a Structured Output Storage (SQLite) database, ensuring each user's invoice history is private and persistent.

The selected process, Data Visualization (Matplotlib), then transforms this stored, structured data into easily digestible insights. Using Matplotlib and Seaborn, the system dynamically generates interactive visualizations—specifically, bar charts (vendor spending), line charts (monthly trends), pie charts (category breakdowns), scatter plots, and heatmaps. This enables quick financial oversight from the user's personal invoice history. The resulting charts and graphs are delivered as part of the system's overall Outputs & Exports.

- Stored data is visualized using Matplotlib/Seaborn to generate:
    - Bar charts (vendor spending)
    - Line graphs (monthly trends)
    - Pie charts (category breakdowns)
    - Scatter plots & heatmaps (performance metrics).

8. Outputs & Exports

- The system delivers multiple consumable outputs:
- Analytics Graphs
- Interactive visualizations for financial oversight.
- Original vs Processed Image
- Side-by-side view to validate preprocessing quality.
- Extracted Text
- Full OCR result for review or copy-paste.
- Exportable CSV Files
- Structured data export for Excel, Power BI, or accounting software.

9. Why This Architecture Works

The four key pillars of the design are:

- Modular Design: Each part of the pipeline (like OCR, extraction, or database management) is independent, allowing for easy updates and future enhancements, such as swapping out the current RegEx field extraction for more advanced NLP models like spaCy.
- Scalable: The entire solution is built on open-source tools, ensuring it is both cost-effective and flexible for deployment, from small local setups to large-scale cloud environments.
- User-Centric: All features and outputs, including the Streamlit UI and data visualizations, are designed for the non-technical "real-world business user" to ensure maximum usability.
- Resilient: Robust preprocessing (using OpenCV) and validation steps are integrated to reliably handle messy real-world inputs, such as skewed, low-resolution, or multi-page documents, which often cause other OCR tools to fail.

5. Tools and Resources

5.1. Frontend

Streamlit – Rapid development of interactive, responsive web UIs with built-in widgets, session state, and real-time updates

5.2. Core Libraries and Frameworks

- OpenCV (cv2) – Image preprocessing (denoising, thresholding, color conversion)
- Tesseract OCR + pytesseract – High-accuracy optical character recognition
- pdf2image – Converts PDF pages into images using Poppler backend
- Pillow (PIL) – Image loading, format support, and in-memory manipulation
- Matplotlib, Seaborn – Publication-quality data visualizations (bar charts, line trends, heatmaps, etc.)
- Regex (re) – Rule-based extraction of structured fields (dates, IDs, amounts)
- spaCy – Lightweight NLP library for Named Entity Recognition (NER), enabling smarter extraction of vendors and dates from unstructured text.
- imagehash – Generates perceptual hashes of images to detect duplicate or near-duplicate visual uploads (e.g., same invoice re-scanned).

5.3. Database and Data Management

- SQLite – Lightweight, file-based relational database for persistent, user-isolated storage
- Python Standard Library (json, io, datetime) – For data serialization, file handling, and time formatting

5.4. External APIs

Google Gemini API – Powers the AI chat interface for natural language queries about invoices (e.g., “Show me all bills from ABC Corp”)

6. Future Works

6.1. Offline functionality

We aim to enable fully offline operation by packaging the entire pipeline — including Tesseract, OpenCV, and lightweight language models — into a standalone desktop or mobile application. This will allow users in low-connectivity environments (e.g., field operations, remote offices) to process invoices without internet dependency, while still benefiting from structured extraction, validation, and local analytics.

6.2. Scalability

To support organizational deployment, we plan to evolve the architecture from SQLite to a scalable backend (e.g., PostgreSQL or cloud databases) with role-based access control, API endpoints, and asynchronous processing queues. This will allow thousands of users to upload, analyze, and collaborate on invoice data securely — transforming the tool from a personal assistant into an enterprise-grade financial intelligence platform.

7. References

1. Towards Natural Language-Based Document Image Retrieval: New Dataset and Benchmark
   https://arxiv.org/abs/2512.20174
2. ReceiptSense: Beyond Traditional OCR -- A Dataset for Receipt Understanding
   https://arxiv.org/abs/2406.04493
3. E2E Process Automation Leveraging Generative AI and IDP-Based Automation Agent: A Case Study on Corporate Expense Processing
   https://arxiv.org/abs/2505.20733
4. MMDocBench: Benchmarking Large Vision-Language Models for Fine-Grained Visual Document Understanding
   https://arxiv.org/abs/2410.21311
5. Large Language Models for Simultaneous Named Entity Extraction and Spelling Correction
   https://arxiv.org/abs/2403.00528
6. TransDocAnalyser: A Framework for Offline Semi-structured Handwritten Document Analysis in the Legal Domain
   https://arxiv.org/abs/2306.02142
7. Visual Information Extraction in the Wild: Practical Dataset and End-to-end Solution
   https://arxiv.org/abs/2305.07498
8. Extending TrOCR for Text Localization-Free OCR of Full-Page Scanned Receipt Images
   https://arxiv.org/abs/2212.05525
9. Robustness Evaluation of Transformer-based Form Field Extractors via Form Attacks
   https://arxiv.org/abs/2110.04413
10. LayoutReader: Pre-training of Text and Layout for Reading Order Detection
    https://arxiv.org/abs/2108.11591
11. End-to-End Information Extraction by Character-Level Embedding and Multi-Stage Attentional U-Net
    https://arxiv.org/abs/2106.00952
12. ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction
    https://arxiv.org/abs/2103.10213
13. TLGAN: document Text Localization using Generative Adversarial Nets
    https://arxiv.org/abs/2010.11547
14. Abstractive Information Extraction from Scanned Invoices (AIESI) using End-to-end Sequential Approach
    https://arxiv.org/abs/2009.05728
15. EATEN: Entity-aware Attention for Single Shot Visual Text Extraction

https://arxiv.org/abs/1909.09380

16. Deep Learning Approach for Receipt Recognition
https://arxiv.org/abs/1905.12817
17. Business Research Insights. (2025). Invoice OCR API Market Size, Share & Trends, 2025-2033
https://www.businessresearchinsights.com/market-reports/invoice-ocr-api-market-115569
18. Veryfi. (2025). Invoice OCR in 3–5 Seconds: 2025 Benchmark.
https://www.veryfi.com/ai-insights/invoice-ocr-competitors-veryfi/
19. Tipalti. (2025). Five Reasons Why OCR Isn't Enough.
https://tipalti.com/blog/five-reasons-why-ocr-isnt-enough/
20. DocuClipper. (2025). 9 Biggest OCR Limitations And How To Overcome Them.
https://www.docuclipper.com/blog/ocr-limitations/