Journal of Natural and Engineering Research

JNER, 2026, 2 (1): 56-71

Developing a Web Platform to Extract Information from Receipt Images

Barış DEMİRHAN[1*], Yakup KUTLU

[1] Department of Computer Engineering, Iskenderun Technical University, Hatay, Türkiye.

ABSTRACT

Data extracted from receipts are necessary for various applications in many industries, and this extraction process can be simplified through pre-trained deep learning models. However, high-quality receipt images are essential for algorithms to produce accurate models. This paper focuses on the techniques to improve the image preprocessing process and the information extraction methods applied in Turkish and English and create a web-based receipt scanning and data management system. Optical Character Recognition (OCR) technology is a system that provides a full alphanumeric recognition of printed or handwritten characters from images. Initially, OpenCV has been used to detect the bill or invoice and filter out the unnecessary noise from the image. Then the intermediate image is passed for further processing using the Tesseract OCR engine, which is an optical character recognition engine. This application is built upon a Flask backend, a responsive HTML/CSS/JavaScript frontend, and a SQLite database, offering capabilities including single and batch receipt scanning, editable JSON-based data review by both using regex and AI parser, persistent storage across two language-specific database tables, Excel export, and downloadable output bundles including raw OCR text and structured JSON files. Our methodology and system prove to be highly accurate while tested on a variety of input images of bills and invoices. The architecture is modular and extensible, making it suitable as a foundation for larger-scale document digitization workflows.

ARTICLE INFO

Received 13.05.2026,
Accepted 23.06.2026,
Publication Date 30.06.2026

Keywords:
OCR, OpenCV, Digitization, LLM, Regex, Receipt, Web platform

[*]*Corresponding Author: Barış DEMİRHAN, E-mail: barisdemirhan03@gmail.com*

Fiş Görselinden Bilgi Çıkarmak için Bir Web Platformu Geliştirmesi

Barış DEMİRHAN[1*], Yakup KUTLU

[1] Bilgisayar Mühendisliği Bölümü, Iskenderun Teknik Üniversitesi, Hatay, Türkiye.

ÖZET

Bu çalışma, fişlerden elde edilen veriler birçok sektörde çeşitli uygulamalar için gereklidir ve bu veri çıkarma süreci önceden eğitilmiş derin öğrenme modelleriyle basitleştirilebilir. Bununla birlikte, algoritmaların doğru modeller üretmesi için yüksek kaliteli fiş görüntüleri şarttır. Bu makale, Türkçe ve İngilizce olarak uygulanan görüntü ön işleme sürecini ve bilgi çıkarma yöntemlerini iyileştirme tekniklerine ve web tabanlı bir fiş tarama ve veri yönetim sistemi oluşturmaya odaklanmaktadır. Optik Karakter Tanıma (OCR) teknolojisi, görüntülerden basılı veya el yazısı karakterlerin tam alfanümerik tanınmasını sağlayan bir sistemdir. Başlangıçta, OpenCV, faturayı veya makbuzu tespit etmek ve görüntüden gereksiz gürültüyü filtrelemek için kullanılmıştır. Daha sonra, ara görüntü, optik karakter tanıma motoru olan Tesseract OCR motoru kullanılarak daha fazla işleme tabi tutulmuştur. Bu uygulama, Flask tabanlı bir arka uç, duyarlı bir HTML/CSS/JavaScript ön uç ve bir SQLite veritabanı üzerine kurulmuştur ve tekli ve toplu makbuz tarama, hem regex hem de yapay zeka ayrıştırıcısı kullanarak düzenlenebilir JSON tabanlı veri inceleme, iki dile özgü veritabanı tablosunda kalıcı depolama, Excel dışa aktarma ve ham OCR metni ve yapılandırılmış JSON dosyalarını içeren indirilebilir çıktı paketleri gibi özellikler sunmaktadır. Metodolojimiz ve sistemimiz, çeşitli fatura ve makbuz görüntülerinde test edildiğinde son derece doğru sonuçlar vermektedir. Mimari modüler ve genişletilebilir olup, daha büyük ölçekli belge dijitalleştirme iş akışları için temel olarak uygundur.

MAKALE BİLGİSİ

Anahtar Kelimeler:

OCR, OpenCV, Dijitalleştirme, LLM, Regex, Makbuz, fiş, Web platformu

INTRODUCTION

Nowadays, it has become a norm for every family to go to the supermarket at least once a week or twice in a week. These trips aim to purchase groceries, ingredients, foods and more. At the end of each shopping trip, a receipt is printed out and handed to the customers. These receipts are the proof of purchase and ownership that people have the right to receive after buying goods. Although this is a reliable way of storing information, it is very time-consuming to search through them and looking for a specific transaction is tedious. These types of tasks can be done easily by a computer, but the paper bills cannot be stored digitally without manually typing their contents into a computer, which in itself is a very time-consuming task. Therefore, there is a requirement for extracting and storing this information from the bills automatically. This can be achieved by using optical character recognition. OCR (Optical Character Recognition) is an engine that enables the recognition and retrieval of text information of printed or handwritten character units from pictures or scanned documents that contain texts. OCR as a process consists of diverse sub-processes, namely, text localization, character segmentation, and character recognition. Although some methods work without segmentation (Ozdil & Vural, 1997).

Tesseract, the most widely used open-source OCR engine, has been the subject of extensive study since its release by Hewlett-Packard in the 1980s and its subsequent open-sourcing in 2005 (Smith, 2007). However, raw OCR output must still be parsed, validated, and stored in a structured manner to be useful. Prior work has demonstrated that the combination of OpenCV-based image preprocessing with Tesseract OCR engine can achieve high accuracy on printed receipts, provided that appropriate preprocessing steps are applied to address noise, shadow, and low-contrast artefacts (Kumar et al., 2020).

A comparative study of OCR-based and OCR-free models for Turkish receipt documents has confirmed that the Turkish language demonstrates specific challenges, including unique character sets and fiscal document structures not found in English-language corpora (Aydin et al., 2024). This paper describes the design and implementation of a self-contained web application that addresses these gaps. The system integrates Tesseract OCR into a Flask-based web interface, offering end-to-end receipt digitisation: from image upload through OCR preprocessing, text extraction, structured parsing by both AI and regex, user-directed editing, and persistent database storage. The application supports English and Turkish receipts through separate, independently tunable processing pipelines. This modular design decouples the OCR preprocessing from the semantic parsing layer, allowing for easy integration of additional languages or alternative OCR engines in the future. The resulting platform offers a self-contained, privacy-focused solution suitable for localized deployment in personal finance or corporate expense tracking environments.

LITERATURE REVIEW

Tesseract was originally developed at Hewlett-Packard between 1984 and 1994 and open-sourced in 2005. A related study provides a comprehensive architectural overview of the engine, describing

its pipeline-based approach to text recognition: connected component analysis, blob organisation into text lines, fixed-pitch detection, word recognition via a two-pass process, and an adaptive classifier that refines recognition accuracy as it processes the page (Smith, 2007). A key configuration decision for receipt processing is the selection of the Page Segmentation Mode (PSM). PSM 4, which assumes a single column of text with variable character sizes, has been identified as the most appropriate mode for receipt like layouts where items are arranged vertically with prices aligned to the right. The OCR Engine Mode (OEM) 3, combining the LSTM-based neural network with the legacy engine, provides the best balance of accuracy and speed for printed receipt text (Smith, 2007).

Kamisetty et al. (2022) compare three OCR engines — Keras OCR, Easy OCR, and Tesseract OCR — for invoice digitisation and find that Tesseract produces the most structured and format-preserving output for printed financial documents (Kamisetty et al., 2022). Their system converts extracted text into both JSON and CSV formats, a design pattern adopted in the present work. Regular expressions are used for post-OCR field extraction, consistent with the approach taken here.

Although OCR can be applied to almost any type of handwritten or printed document, this paper's target is focused solely on receipts. Over the past few years, dozens of research studies have been performed on invoice recognition and data extraction through the application of OCR. These studies all aimed to resolve various issues from numerous fields to improve the quality of life. For instance, in the case of one study (Garcia & Claour, 2021), the usage of OCR for data retrieval from receipts had been the key to developing a mobile application for bookkeepers' financial management. The idea was to take the scanned pictures of receipts and transcribe their info using OCR, specifically Tesseract OCR engine, a technology implemented by Google using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). Before extracting the texts using OCR, the authors employed several techniques to address the image-related issues (skewness, blurring, and tilting). These techniques included various processes: isolating texts using segmentation, filtering necessary data using normalization, extracting features using a neural network or Euler number, distributing inputs using multiple classification techniques, and manual data double-checking using spellchecking or content matching.

A different way to perform OCR is by applying the Convolutional Neural Networks (CNNs). Naturally, CNNs, as with other deep learning techniques, are much more proficient at recognizing characters than simpler models, such as the standard Tesseract OCR, with a trade-off in processing speed (Ahmed et al., 2023). Additionally, one specific 5-layered CNN model (Rabby et al., 2020) has proven its power for English character recognition by obtaining a recognition accuracy of 99.98%.

In another related study,a new method address multilingual receipt preprocessing, proposing Fuzzy C Means clustering for blur estimation and brightness/chromaticity-based background removal as

preprocessing steps for Vietnamese and English receipts (Nguyen et al., 2024). Their work highlights that background removal and blur assessment are particularly important when receipts are photographed under varied lighting conditions — a scenario that is equally relevant for the Turkish and English receipts processed by the system described in this paper. The preprocessing pipeline implemented in this work applies adaptive Gaussian thresholding, which addresses the uneven illumination problem identified across these prior works, and pairs it with median blur for noise suppression and bicubic upscaling to meet Tesseract's minimum character height requirement.

Pre processing on the image can be done using OpenCV to mini mize the noise. Open CV is an open-source software (Opencv.org, 2020) library that holds applications in the domains of Machine learning and computer vision. The processing of images can be done by incorporating the OpenCV library. With the processing step, the images will be more clear, and the relevant data can be extracted efficiently. This pre-processing step can involve determining the region of interest and cropping the image accordingly, noise removal, thresholding, dilation, erosion, and contour detection. Once these steps are performed, now OCR engines can be used to read the image and extract the text from it accurately.

MATERIALS AND METHODS

System Architecture

The system follows a straightforward client–server architecture, illustrated in Fig. 1. A Flask application serves the web frontend and exposes a set of REST endpoints for file management (upload, delete, list), receipt scanning, record management, and export. At the core of the system is the OCR pipeline, which preprocesses receipt images using OpenCV and extracts text with Tesseract OCR. Two parsing strategies are supported: a fast, deterministic regex-based parser that runs by default, and an optional LLM-based parser that sends the raw OCR text to the Groq Cloud API for more robust field extraction, with automatic fallback to the regex parser if the API call fails. Extracted receipt data is persisted in a SQLite database, with separate schemas for English and Turkish receipts to accommodate their differing field structures. Uploaded images and generated exports are stored on the local filesystem under dedicated directories.

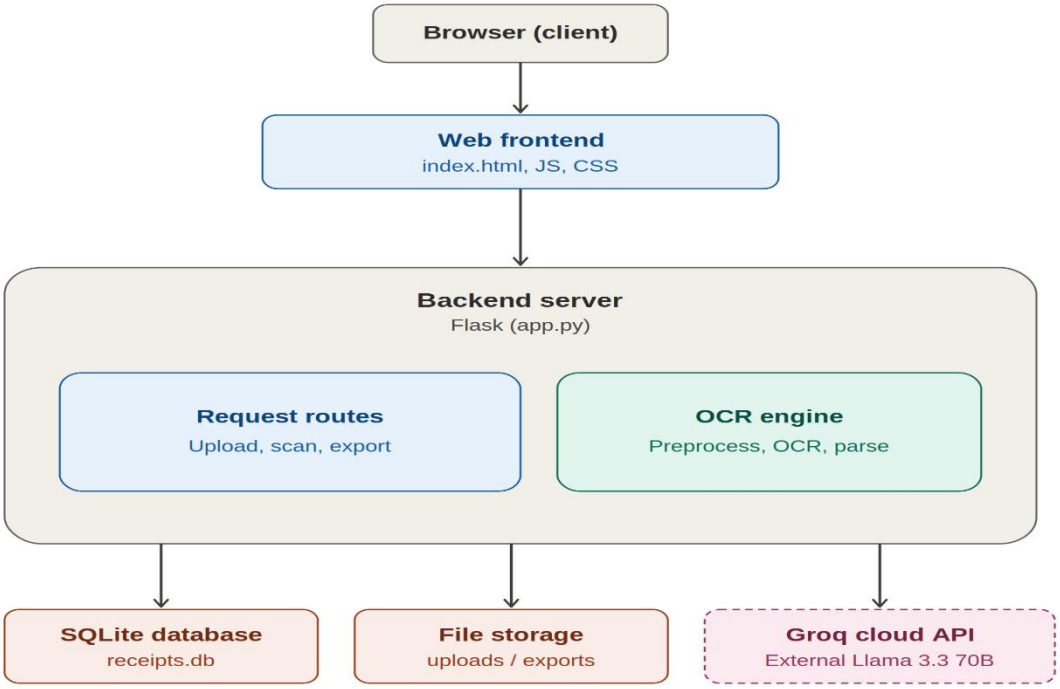


Fig 1. Architecture of the receipt scanning system.

Image Preprocessing Pipeline

The quality of OCR output is critically dependent on the quality of the input image. As Smith (2007) notes, Tesseract's recognition accuracy degrades substantially when input images contain noise, poor contrast, or insufficient resolution (Smith, 2007). Thermal receipt paper — the dominant medium for point-of-sale receipts — produces images with low contrast, varying illumination, salt-and-pepper noise, and character fragmentation when photographed. The preprocessing pipeline described here directly addresses these artefacts, following the established best practices identified in prior work (Nguyen et al., 2024).

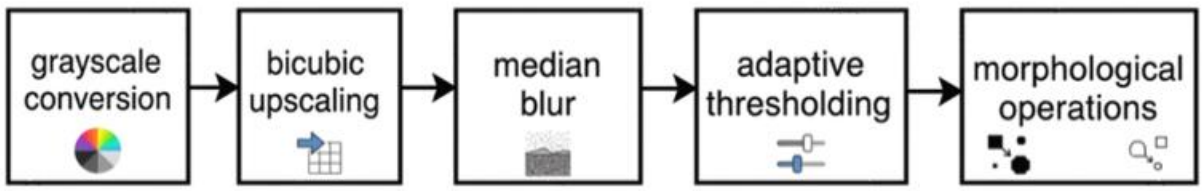


Fig 2. Process flow of methodology

The pipeline applies the following transformations in Fig 2. The same pipeline is applied uniformly to both English and Turkish receipt images, though the two functions are architecturally separate to allow independent tuning as the system matures.

Grayscale Conversion

The input image is converted from BGR colour space to grayscale using cv2.cvtColor(img, cv2.COLOR_BGR2GRAY). This step eliminates chromatic information irrelevant to text recognition and reduces computational complexity for subsequent operations. Grayscale conversion is a universal first step in document OCR preprocessing, identified as foundational in all related works (Ozdil & Vural, 1997 ; Kumar et al., 2020 ; Garcia & Claour, 2021).

Fig 3. Grayscale Conversion on a receipt

Bicubic Upscales

The grayscale image is upscaled by a factor of 1.5 in both dimensions using cv2.resize() with the INTER_CUBIC interpolation method in Fig 3. Prior works specifies that Tesseract achieves optimal recognition accuracy when character heights are approximately 30 pixels (Smith, 2007). Upscaling by 1.5x ensures that characters from lower resolution sources — particularly mobile phone photographs — meet this threshold.

Median Blur

A median blur with kernel size 3 is applied using cv2.medianBlur(). Median filtering is particularly effective for removing salt-and-pepper noise, which is common in images of thermal receipts, while preserving edge sharpness better than Gaussian blur. Kumar et al. (2020) use median blur as part of their shadow removal pipeline to suppress text content in the background image, demonstrating its general utility for receipt image cleaning (Kumar et al., 2020).

Adaptive Thresholding

The blurred image is binarised using cv2.adaptiveThreshold() with the ADAPTIVE_THRESH_GAUSSIAN_C method, a block size of 21, and a constant C of 10. Adaptive thresholding computes a local threshold for each pixel neighbourhood rather than a single global threshold, making it robust to uneven illumination — a frequent problem when receipts are photographed under directional or ambient lighting.

Fig 4. Adaptive Thresholding on a receipt

Morphological Operations

A single iteration of dilation followed by erosion is applied with a 1x1 kernel using cv2.dilate() and cv2.erode(). While the 1x1 kernel produces a minimal morphological effect at this scale, the operation serves as a normalisation step. The dilation and erosion pair can be tuned to thicker kernels in future iterations to repair character fragmentation in low-quality prints. This techique

visualized in Fig 4. This technique is specifically recommended for receipt and invoice images by the multilingual preprocessing framework of another similar work (Nguyen et al., 2024).

EXPERIMENTAL STUDY

Parser Modules

The option of rule-based regular expression parsing over machine learning-based extraction is deliberate and justified by the deployment constraints of the system. Transformer-based extractors such as LayoutLMv3 achieve higher accuracy in this scenario, but require GPU training infrastructure, large annotated datasets, and significant computational resources for inference (Aydin et al., 2024). For a locally deployable application targeting a specific and relatively predictable document format, regular expression parsing provides a good accuracy to deployability trade off.

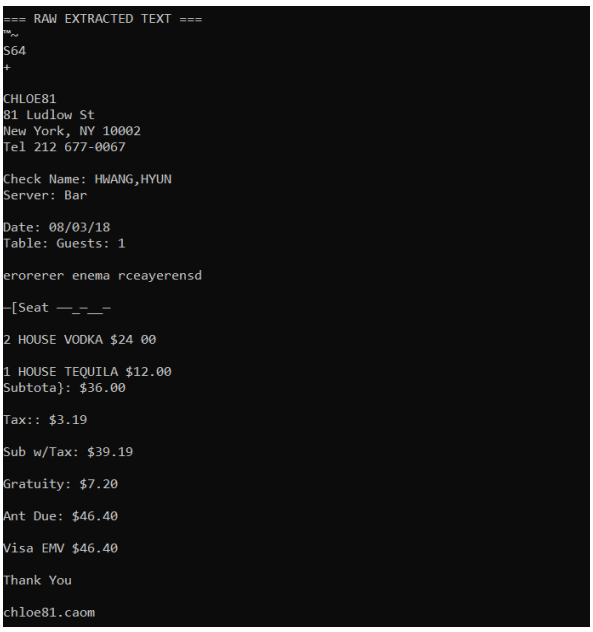
English Parser

The English parser function in our project processes raw Tesseract output line by line, applying a sequence of regular expressions to extract the following fields: company name, date, line items (includes quantity, name, price), subtotal, tax, gratuity, and total. The parser incorporates several robustness improvements over baseline approaches.

Company name extraction uses a noise filter: it selects the first line containing at least three alphabetic characters, rather than the naive first non-empty line. This prevents receipt headers consisting only of asterisks, dashes, or single-character separators from being misclassified as company names — a frequent OCR output artefact.

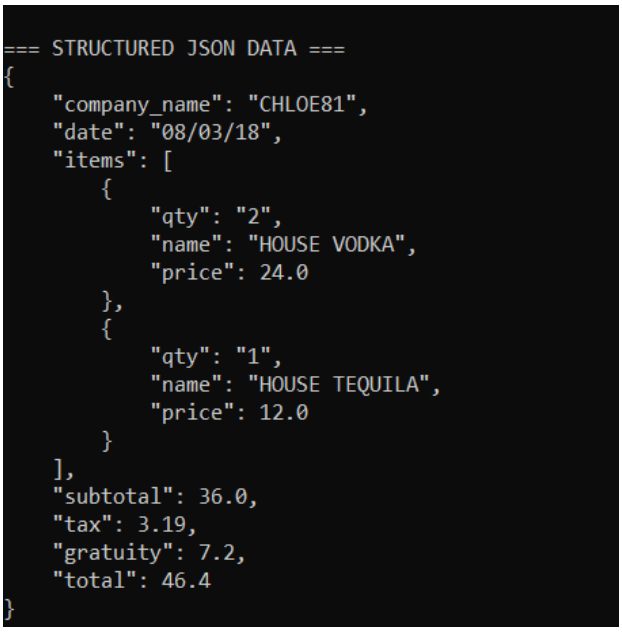
Item price parsing accepts both period and comma as decimal separators, as well as space-separated decimal digits, which is a common OCR misread of decimal points on thermal paper.

For the Total amount field, the highest matched monetary value among lines containing total-related keywords is retained, handling the case where receipts print subtotals before the grand total. Lines containing payment-method keywords such as visa, emv, mastercard, amex, cash , and change are excluded from item matching, preventing credit card transaction lines from being misclassified as purchased items. Visualized raw extracted text and structured JSON data are shown in Fig 5 and Fig 6.



```
=== RAW EXTRACTED TEXT ===
™~
S64
+

CHLOE81
81 Ludlow St
New York, NY 10002
Tel 212 677-0067

Check Name: HWANG,HYUN
Server: Bar

Date: 08/03/18
Table: Guests: 1

erorerer enema rceayerensd

-[Seat ——_-__-

2 HOUSE VODKA $24 00

1 HOUSE TEQUILA $12.00
Subtota}: $36.00

Tax:: $3.19

Sub w/Tax: $39.19

Gratuity: $7.20

Ant Due: $46.40

Visa EMV $46.40

Thank You

chloe81.caom
```

Fig 5. Raw extracted text



```
=== STRUCTURED JSON DATA ===
{
    "company_name": "CHLOE81",
    "date": "08/03/18",
    "items": [
        {
            "qty": "2",
            "name": "HOUSE VODKA",
            "price": 24.0
        },
        {
            "qty": "1",
            "name": "HOUSE TEQUILA",
            "price": 12.0
        }
    ],
    "subtotal": 36.0,
    "tax": 3.19,
    "gratuity": 7.2,
    "total": 46.4
}
```

Fig 6. Structured JSON data

Turkish Parser

The Turkish parser function in our project processes raw Tesseract output line by line, applying a sequence of regular expressions, following a similar way to the English parser. Turkish fiscal receipts include a standardised set of mandatory fields

- company name, VKN (a 10-digit tax identification number), date, time, fiscal receipt number , product lines with KDV rates, total KDV, and total amount

- these are not present in English-format receipts (Aydin et al., 2024).

The Turkish parser replaces spaces and commas in price strings with dots before float conversion, handling this format correctly. “a specific artefact not present in English receipts that would cause a naive decimal parser to fail”.

AI Parser

The function that used for search API, endpoint runs the receipt image through Tesseract OCR locally, then sends the extracted text to the Groq Cloud API, which uses the Llama 3.3 70B model to parse it into structured fields (company name, items, totals, etc.) according to a predefined JSON schema. If the API call fails, the system automatically falls back to a local regex-based parser, so scanning still works without the external service. This hybrid design keeps OCR local and fast while offloading the more complex language understanding task to a larger cloud-hosted model.

Database Design

The system uses SQLite for persistent storage, selected for its zero-configuration deployment, Python standard library support, and suitability for single-user local applications. Two tables are maintained within a single database file: receipts_eng for English receipts and receipts_tr for Turkish receipts, shown in Fig 7 and Fig 8.

Fig 7. Table for English receipts

Column	Type	Description
id	INTEGER PK	Auto-incremented primary key
company_name	TEXT	Extracted company or store name
date	TEXT	Receipt date (MM/DD/YYYY format)
items	TEXT	JSON array of {qty, name, price} objects
subtotal	REAL	Pre-tax subtotal amount
tax	REAL	Tax amount
gratuity	REAL	Gratuity or tip amount
total	REAL	Grand total amount due
saved_datetime	TIMESTAMP	Automatic insertion timestamp

Fig 8. Table for Turkish receipts

Column	Type	Description
id	INTEGER PK	Auto-incremented primary key
sirket_adi	TEXT	Company name (Sirket Adi)
vkn	TEXT	Tax identification number (10 digits)
tarih	TEXT	Receipt date (DD.MM.YYYY or DD/MM/YYYY)
saat	TEXT	Receipt time (HH:MM or HH:MM:SS)
fis_no	TEXT	Fiscal receipt number
urunler	TEXT	JSON array of {ad, kdv_oran, fiyat} items
toplam_kdv	REAL	Total VAT (KDV) amount
toplam_tutar	REAL	Total payable amount
raw_text	TEXT	Full raw Tesseract output (for reprocessing)
kayit_tarihi	TIMESTAMP	Automatic insertion timestamp

User Interface And Data Management

The system features a single-page web application built on Flask and Jinja2, engineered for asynchronous, reload-free interactions via a REST-style JSON API. To optimize the human-in-the-loop workflow, the interface is organized into a clean, three-panel layout. The left panel handles file management, featuring a drag-and-drop zone and real-time status indicators (e.g., queued, scanning, complete). The center panel provides an interactive workspace, splitting the view between a raw OCR text terminal and an editable structured field editor. The right panel centralizes administrative controls, including database access, export operations, and configuration settings.

Users can dynamically toggle the system language, which seamlessly swaps between two distinct language-specific OCR pipelines and their accompanying parsing engines. Choosing a language automatically adjusts the application state: it updates the backend processing targets, restructures the editable schema fields to match localized document formats (such as adapting fields between English and Turkish business receipts), and filters the database view to the corresponding data tables.

A central pillar of the system's architecture is its strict "review-before-save" design philosophy. Document digitalisation often suffers from character-level misreads or field misclassifications due to degraded physical receipts. To prevent these errors from corrupting the database, the interface forces an intermediate validation step. Whether processing a single receipt or handling multiple documents concurrently via the batch processing workflow, extracted data is first presented to the user as editable cards in the user interface. This design mitigates the compounding errors typical of automated batch storage by enabling manual verification and correction prior to database persistence.

Once validated, the data becomes highly portable. The application provides instant file export capabilities to support both technical and non-technical workflows. For developer and archival use, users can download compressed ZIP archives including paired raw text files and structured JSON data for each receipt. For business and administrative use, the system generates formatted Excel spreadsheets, dynamically tailored to the active language table with optimized layouts, legible typography, and structured item breakdowns ready for immediate reporting. Illustrated UI layout is shown in Fig 9.

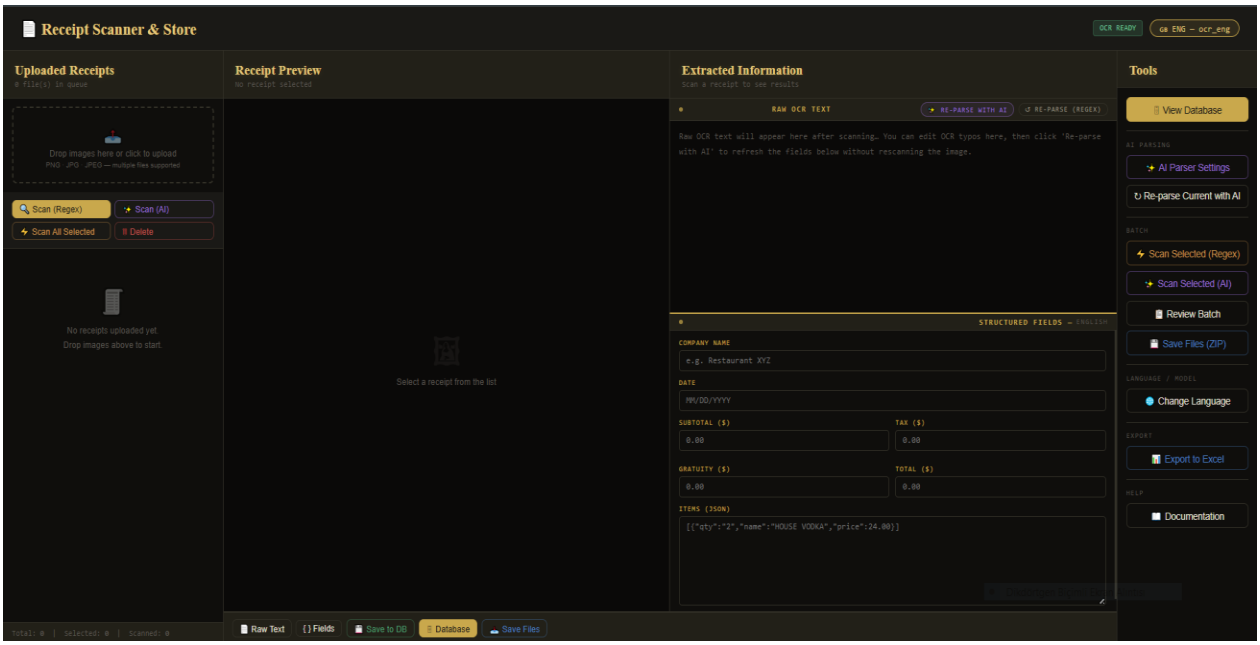


Fig 9. User Interface Layout

Observations And Limitations

OCR And Parsing

System testing indicates that the image preprocessing pipeline significantly upgrades Tesseract's text recognition capabilities. Bicubic upscaling proves vital when handling low-resolution mobile photographs where text is highly pixelated, while localized adaptive thresholding successfully counteracts the uneven illumination and shadow gradients typical of ambient indoor lighting (Kumar et al., 2020) (Nguyen et al., 2024).

Nevertheless, certain environmental and structural factors still trigger text recognition errors. Highly faded thermal paper, dense or unexpected multi-column layouts that disrupt the standard line segmentations, handwritten annotations, and highly stylized corporate logos remain notoriously difficult for the base engine to interpret cleanly.

Downstream, the rule-based regex parsing layer exhibits a clear sensitivity to these character-level misreads. If a critical keyword is corrupted during extraction (such as reading an "O" as a "0"), rigid pattern matching inevitably fails. While the English parser minimizes this vulnerability by isolating the highest monetary value as a fallback, the Turkish parser faces unique challenges stemming from diacritic variations, though this is partially mitigated by forcing uppercase normalization. Line-item extraction remains the most challenging component across both pipelines due to the massive, non-standardized diversity in how individual merchants format their transactions.

System Boundaries

In its current state, the system is constrained to standard image file formats (PNG, JPG, and JPEG), leaving direct digital PDF processing as an unmet need for modern electronic invoices. Architecturally, the deployment relies on a completely local footprint. While using SQLite guarantees zero operational costs and absolute data privacy, its single-writer model inherently limits the system's capacity for multi-user concurrency. Additionally, path configurations are heavily tailored to Windows environments. To prevent catastrophic failures when processing Turkish characters inside Windows directory structures, the file system logic avoids standard image reads, relying instead on reading binary byte arrays via NumPy to ensure cross-platform string compatibility.

CONCLUSION

This research presents the design and implementation of a self-contained, web-based receipt scanning and data management system engineered for efficient document digitisation. To address the persistent challenges of text extraction from low-contrast, degraded thermal receipt prints, a dedicated image preprocessing pipeline was established using OpenCV. This pipeline sequentially leverages grayscale conversion, 1.5x bicubic upscaling, noise-suppressing median filtering, and localized adaptive Gaussian thresholding to structurally optimize document quality before character recognition. Text recognition and bilingual semantic parsing are driven by the open-source Tesseract OCR engine, running specialized pipelines to handle English and Turkish fiscal layouts interchangeably.

FUTURE DEVELOPMENTS

Moving forward, the primary architectural evolution will focus on completely reworking the data extraction and text parsing layer to handle increasingly complex fiscal document variants. To move past the limitations of static pattern matching, future development will focus on the following core tracks:

Hybrid OCR Engines & Multi-Modal Processing: Future iterations will integrate a hybrid OCR pipeline, combining Tesseract's lightweight transcription engine with deep learning-based visual text recognizers. This hybrid approach will serve as the backbone for **LayoutLMv3**, a multimodal transformer model. By combining text semantics, spatial bounding box coordinates, and visual features, LayoutLMv3 will act as a context-aware model capable of mapping fragmented line items and prices without relying on rigid keyword matches.

Custom Dataset Curation: To effectively transition to this layout-aware architecture, a dedicated Turkish and English business receipt dataset will be curated from scratch. This custom dataset will

be explicitly annotated with bounding box coordinates, word-level labels, and localized structural tokens, serving as a specialized training foundation tailored to real-world, localized fiscal documents.

Fine-Tuning Architecture: Model training and optimization will leverage a streamlined fine-tuning workflow utilizing modern PyTorch and Hugging Face Trainer modules. To maximize text recognition accuracy prior to layout parsing, the Tesseract training toolchain will be utilized to fine-tune language-specific files directly on the unique typography of local thermal receipts.

Native Digital Ingestion: To accommodate modern electronic billing ecosystems, the application's ingestion layer will be expanded to natively accept other type of files such as PDF files and more. This pathway will utilize backend libraries to extract embedded digital text layers when available, or convert pages into optimized images for the preprocessing pipeline when handling scanned documents, ensuring a unified entry point for both digital and physical invoices.

REFERENCES

Ahmed, S., Ali, A., & Naser, E. (2023). Tesseract OpenCV versus CNN: A comparative study on the recognition of unified modern Iraqi license plates. Revue d'Intelligence Artificielle, 37(5), 727–734.

Aydin, T., Yildizak, B., & Caliskan, E. E. (2024). A comparative study of layout based and OCR-free models for Turkish receipt documents. Proceedings of the 2024 IEEE Conference. https://doi.org/10.1109/2024.turkish.receipts

Garcia, M. B., & Claour, J. P. (2021, November). Mobile bookkeeper: Personal financial management application with receipt scanner using optical character recognition. Proceedings of the 2021 1st Conference on Online Teaching for Mobile Education (OT4ME), 15–20.

Kamisetty, V. N. S. R., Chidvilas, S., Revathy, S., Jeyanthi, P., Anu, V. M., & Gladence, L. M. (2022). Digitization of data from invoice using OCR. Proceedings of the 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), 1–8. https://doi.org/10.1109/ICCMC53470.2022.9754117

Kumar, V., Kaware, P., Singh, P., Sonkusare, R., & Kumar, S. (2020). Extraction of information from bill receipts using optical character recognition. Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), 72–77. https://doi.org/10.1109/ICOSEC49089.2020.9215246

Nguyen, H. Q., Vo-Xuan, T., Nguyen, C. T., Ngo, T., Nguyen, D. T. V., & Huynh, K. T. (2024). Multilingual receipt image preprocessing optimization for OCR. Proceedings of the 2024 International Conference on Advanced Technologies for Communications (ATC), 922–927. https://doi.org/10.1109/ATC63255.2024.10908135

Opencv.org. (2020). Information related to Open CV. https://opencv.org/

Ozdil, M. A., & Vural, F. T. Y. (1997). Optical character recognition without segmentation. Proceedings of the Fourth International Conference on Document Analysis and Recognition, 2, 483–486. https://doi.org/10.1109/ICDAR.1997.620545

Rabby, A. S. A., Islam, M. M., Hasan, N., Nahar, J., & Rahman, F. (2020, July). Language detection using convolutional neural network. Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1–5.

Smith, R. (2007). An overview of the Tesseract OCR engine. Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 629–633. https://doi.org/10.1109/ICDAR.2007.4376991