Joblogic Improvement

Shashank Barai
School of Information, Media and Design
SRH Hochschule Heidelberg
Heidelberg, Germany
shashank.barai@stud.hochschule-heidelberg.de

Amogh Goudar
School of Information, Media and Design
SRH Hochschule Heidelberg
Heidelberg, Germany
amoghshrinivas.goudar2@stud.hochschule-heidelberg.de

Mercy Kiongera
School of Information, Media and Design
SRH Hochschule Heidelberg
Heidelberg, Germany
mercymugure.kiongera@stud.hochschule-heidelberg.de

Rohit Kulkarni
School of Information, Media and Design
SRH Hochschule Heidelberg
Heidelberg, Germany
rohit.kulkarni@stud.hochschule-heidelberg.de

Apurva Vasaikar
School of Information, Media and Design
SRH Hochschule Heidelberg
Heidelberg, Germany
apurva.vasaikar@stud.hochschule-heidelberg.de

Prof Dr Binh Vu
School of Information, Media and Design
SRH Hochschule Heidelberg
Heidelberg, Germany
binh.vu@srh.de

***Abstract*— As industries advance their digital transformation efforts, particularly in field service management, the demand for AI and ML integration to streamline business operations has increased. This case study focuses on enhancing Joblogic, a cloud-based field service management platform, by leveraging Optical Character Recognition (OCR) and Natural Language Processing (NLP) to automate invoice document processing. The project's objective was to develop a system that accurately converts scanned invoice images into dynamic, editable document templates. Key innovations include the detection and transformation of invoice elements such as tables, logos, and text fields using advanced image preprocessing, machine learning, and AI techniques, including GPT-4 Turbo. This paper critically reviews commercial and open-source OCR technologies, outlines the system's architecture, and evaluates the results achieved in terms of OCR accuracy, table structure detection, and layout preservation. While the system demonstrated high accuracy in text extraction, challenges remain in handling complex invoice layouts and decorative elements, which led to further exploration of deep learning approaches. The project provides a scalable solution that reduces manual effort in document customization and offers opportunities for future development.**

Keywords*— *optical character recognition (OCR), natural language processing (NLP), invoice processing, AI in field service management, dynamic document templates, gpt-4 turbo, table structure detection, layout preservation, image preprocessing

I. INTRODUCTION

In today's rapidly evolving digital landscape, the integration of Artificial Intelligence (AI) and Machine Learning (ML) into business processes is becoming indispensable. This is particularly relevant in the field of field service management, where the challenge of efficiently managing vast amounts of data while ensuring superior customer service is a persistent issue. The Joblogic Enhancement project seeks to address these challenges by leveraging state-of-the-art AI technologies to transform traditional business operations through advanced document processing techniques.

Joblogic, a comprehensive cloud-based field service management solution, facilitates the seamless coordination of back-office operations, mobile workforce management, and customer interactions via a centralized system. However, despite its robust functionality, the platform has faced challenges in efficiently processing the complex document layouts associated with invoice management. This project, led by a team of five, aims to enhance the platform by automating the conversion of invoice images into dynamic, editable document templates.

The scope of this enhancement includes the development of a sophisticated system that uses AI to identify and transform elements within an invoice image—such as tables, logos, and text fields—into editable placeholders. These placeholders dynamically integrate with existing databases, allowing real-time data updates and customizable document outputs. This functionality reduces manual efforts in document customization, thereby improving workflow efficiency.

The project's core innovations include table recognition and editing, logo detection and replacement, and the transformation of static text fields into interactive, editable fields. Using Optical Character Recognition (OCR) and Natural Language Processing (NLP), static text from invoices is converted into editable fields that can be stored and updated with real-time content. This report will cover the methodologies used, the challenges encountered—especially in OCR accuracy—and the results of this AI-driven document processing enhancement. Through this case study, we demonstrate the successful application of AI in optimizing document processing workflows.

II. STATE OF THE ART

As businesses accelerate their digital transformation efforts, automating document processing—particularly invoice management—has become essential for enhancing operational efficiency. Optical Character Recognition (OCR) technology plays a pivotal role in converting scanned documents and images into machine-readable and editable formats. Despite the availability of numerous OCR solutions, challenges persist in handling complex invoice layouts and dynamic table structures. This section critically reviews current OCR technologies relevant to invoice processing and

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

template generation, focusing on their capabilities and limitations in the context of our project's objectives.

A. Commercial OCR Solutions

1) Klippa OCR

Klippa OCR is an AI-driven solution designed to eliminate manual data entry, reduce errors, and prevent document fraud by automating the capture and conversion of scanned documents into structured data formats. It supports various file formats and offers integration via API or SDK to third-party applications.

Klippa's high extraction accuracy of up to 99% and support for multiple file formats make it a strong candidate for general document processing. Its AI-powered image preprocessing enhances data extraction reliability. However, its limitation to languages using the Latin alphabet restricts its applicability for businesses dealing with multilingual documents. Additionally, while it offers extensive features, it may not handle complex invoice layouts with dynamic table structures effectively, which is crucial for our project.

2) Rossum

Rossum provides an AI-powered OCR platform aimed at reducing manual data entry and facilitating data extraction from diverse document types. It employs machine learning to enhance recognition accuracy and supports data privacy compliance.

With up to 95% data extraction accuracy, Rossum is proficient in processing standard invoices. Its use of AI for document classification and duplicate detection adds value in managing large volumes of documents. However, the lack of extensive integration documentation can pose challenges in seamless implementation. Moreover, it does not offer fraud detection capabilities, and its effectiveness in handling complex layouts is uncertain, potentially limiting its suitability for our needs.

3) ChronoScan OCR

ChronoScan is a multipurpose software suite for document processing, offering batch processing and integration with third-party applications.

ChronoScan's ability to handle high-volume documents and integrate with systems like CRM or ERP is advantageous for large enterprises. However, its conversion capabilities are limited to PDF to XML or CSV formats, which may not align with our requirement for generating editable templates. The absence of onboarding support and fraud detection features further diminishes its applicability for our project.

4) Kofax OCR

Kofax OCR assists in converting and editing documents, making them searchable and shareable, and automates document processing workflows.

Kofax offers effective field and line-item capture and cross-device accessibility. Its intelligent automation platform is beneficial for workflow optimization. Nevertheless, lower OCR accuracy compared to competitors and the lack of handwriting recognition are significant drawbacks. Limited output formats and absence of onboarding support may hinder its integration into our system focused on complex invoice processing.

5) Nanonets

Nanonets delivers a modern OCR solution for automating document-related processes, including handwriting recognition and an end-to-end document management system.

Nanonets' ability to extract data from a variety of sources, including handwriting, is valuable for comprehensive data capture. Good customer support and custom data field extraction are positive attributes. However, limitations in output formats and line-item extraction options could impede our goal of generating dynamic, editable templates from complex invoices.

6) Amazon Textract

Amazon Textract uses machine learning to extract both structured and unstructured data from documents.

Trained on a vast array of document types, Amazon Textract excels in creating smart search indexes and facilitating document workflows. However, its reliance on template-based OCR limits flexibility in handling diverse invoice layouts. Language support is confined to a few languages, and processing large volumes requires a premium subscription, affecting scalability and cost-effectiveness for our project.

7) Docsumo OCR

Docsumo's OCR solution employs deep learning to extract data from images or PDF documents, handling complex layouts.

With up to 90% accuracy, Docsumo is adept at processing documents with intricate designs. The ability to review and improve extraction outputs enhances data reliability. Nonetheless, support for only PDF input format and limitations on file size and simultaneous processing pose constraints. The inability to process files larger than 200 MB may be problematic for high-resolution invoice images.

B. Open-Source OCR Solutions

1) Tesseract OCR

Tesseract is a widely used open-source OCR engine capable of recognizing over 100 languages and compatible with various programming languages and frameworks.

As a free and customizable tool, Tesseract offers flexibility for developing tailored OCR solutions. Its integration with Python libraries like PyTesseract allows for extensive customization, making it a potential candidate for our in-house development. However, setting up and optimizing Tesseract can be time-consuming and resource intensive. It lacks advanced features such as document verification, cross-validation, and compliance with data privacy regulations. Additionally, Tesseract may struggle with complex invoice layouts and dynamic tables without significant customization.

C. Technical Components and Tools for Invoice Processing

Implementing an effective OCR solution for invoice processing involves several technical components and tools:

1) Image Preprocessing

Implementing an effective OCR solution for invoice processing involves various essential technical components and tools. One of the primary stages in this process is image preprocessing, where tools such as OpenCV, scikit-image, and PIL (Python Imaging Library) play a crucial role. These libraries help enhance invoice images through methods like noise reduction, contrast enhancement, and de-skewing, all of which contribute to improving OCR accuracy.

However, while these tools are indispensable for preparing images for OCR, they require significant expertise in image processing and may not entirely resolve issues related to poor-quality images or complex backgrounds. This often necessitates additional custom solutions to ensure optimal results.

2) Text Detection and Recognition

Following image preprocessing, text detection and recognition are key steps in identifying relevant data from the invoice. Tools like OpenCV, the EAST (Efficient and Accurate Scene Text Detector), Tesseract OCR, and PyTesseract are commonly employed to locate and extract text regions within preprocessed images. These tools work effectively for standard text detection tasks; however, invoices often present challenges such as varying fonts, sizes, and orientations, which can complicate the extraction process.

As a result, advanced configurations and fine-tuning of these tools are typically required to handle the complexities of invoice layouts accurately.

3) Semantic Analysis and Structured Data Extraction

Once the text has been extracted, semantic analysis and structured data extraction become necessary to make sense of the data. Tools such as NLTK, spaCy, regular expressions, and pandas are used to analyze the extracted text and understand the relationships between different data elements.

This process enables the identification of key fields like invoice numbers, dates, and item descriptions. Although these tools significantly aid in organizing data into structured formats, they require custom development to accurately map the fields across diverse invoice templates, particularly when dealing with varying formats and layouts.

4) Dynamic Template Generation

Another vital component of the process is dynamic template generation, where tools like ReportLab, PDFKit, and Jinja2 are employed to create editable templates based on the extracted data. These tools allow for the formatting of data in a user-friendly and interactive way. However, integrating these tools with OCR outputs to replicate complex invoice layouts is a challenging task that demands a considerable development effort.

Ensuring the correct placement and formatting of dynamic elements such as tables, logos, and text fields requires substantial customizations.

5) Integration and Scalability

For the system to be scalable and efficient, integrating these various components into a unified system is essential. Technologies such as Python, TensorFlow, PyTorch, Docker, and Kubernetes can be employed to ensure that the system can scale and process large volumes of invoices efficiently.

While achieving seamless integration and scalability is feasible, it requires substantial expertise in software development and system architecture. Successfully integrating these technologies ensures that the system can handle real-world demands, such as processing hundreds or thousands of invoices in parallel.

Recent advances in OCR technology have further improved text recognition accuracy, particularly through the use of deep learning techniques such as Convolutional Neural Networks (CNNs) and Transformer-based models. Tools like Microsoft's Table Transformer and Google's Document AI utilize advanced machine learning techniques to understand document layouts more effectively and extract structured data.

These advancements show significant potential in addressing the limitations associated with complex invoice layouts and dynamic table structures. By integrating deep learning models into the process, it is possible to improve the accuracy of text extraction and better understand the structural elements of invoices. However, these technologies often require specialized knowledge in machine learning, and their implementation can be resource-intensive, which may increase both development time and costs.

Despite the advancements in OCR technology, many existing solutions still face limitations when it comes to processing complex invoice layouts and dynamic tables. Commercial solutions that can effectively handle these challenges are often expensive and may not provide the level of flexibility required for specific project needs.

On the other hand, open-source options offer greater opportunities for customization but typically require substantial development efforts and may lack some advanced functionalities available in commercial tools.

In conclusion, the current OCR landscape offers a variety of solutions with diverse capabilities. While commercial tools provide robust features, they may not fully align with the specific requirements for flexibility and cost-effectiveness when handling complex invoice templates. Open-source solutions, though highly customizable, demand considerable development efforts.

Given these challenges, developing a custom in-house solution becomes a strategic decision. By leveraging open-source tools and integrating advanced OCR and AI capabilities, we can create a system tailored to process complex invoices and generate editable templates. This approach helps minimize additional costs and aligns with the project's goals of enhancing efficiency and scalability in document processing workflows.

III. Methodology (Solution Design)

As businesses increasingly shift toward digital transformation, automating data extraction from invoices has become crucial for increasing productivity and minimizing errors associated with manual processes. In this project, we

leverage advancements in Optical Character Recognition (OCR) and Natural Language Processing (NLP) to develop a system that converts invoice images into structured, editable HTML templates. This section outlines the design and methodology used to achieve this goal, addressing challenges related to invoice layout complexity and table detection.

Our solution relies heavily on Tesseract OCR for text extraction and GPT-4 Turbo for generating dynamic templates. Initially, we aimed to use OCR alone to detect table structures, logos, and field text positions, with GPT-4 only handling the template generation. However, OCR struggled with accurately detecting tables, especially when decorative lines interfered with the actual table structures. As a result, we extended the use of GPT-4 Turbo to assist with table identification and layout generation.

This hybrid approach allowed us to overcome some of the limitations of OCR, but challenges remain when dealing with complex invoice layouts where decorative elements are mistaken for table borders.

1) OCR Technology Selection

The first step in automating invoice data extraction involves choosing a reliable OCR solution. We selected **Tesseract OCR**, an open-source OCR engine widely recognized for its flexibility and ability to recognize over 100 languages. Tesseract has undergone continuous improvements, including the integration of **Long Short-Term Memory (LSTM)** networks, which enhance its accuracy in handling complex text patterns, noisy images, and diverse document layouts (Smith, 2007).

To tailor Tesseract for this project, we employed **PyTesseract**, a Python wrapper for Tesseract OCR, which allowed us to integrate OCR functionalities directly into our processing pipeline. While Tesseract is effective for basic text extraction, we enhanced its performance by integrating image preprocessing techniques to address challenges related to poor image quality and complex invoice structures.

2) Image Preprocessing and Table Detection

Image preprocessing plays a critical role in improving OCR accuracy, particularly for documents with complex layouts or poor-quality images. To address common issues such as noise, skewing, and lighting inconsistencies, we utilized tools like OpenCV and scikit-image for preprocessing the invoice images.

This preprocessing involved multiple steps to enhance the quality of the input before applying OCR techniques. One of the key methods applied was adaptive thresholding, which converted the invoice image into a binary format. By dynamically adjusting the threshold based on local pixel values, adaptive thresholding proved robust against lighting variations and effectively distinguished text from the background, making it easier to extract relevant information.

Another crucial technique we employed was morphological transformation, particularly horizontal and vertical line detection. This method helped isolate the grid structures that define tables within invoices. By using a rectangular kernel, we were able to emphasize the horizontal and vertical lines, which allowed for more effective detection of table boundaries.

Accurate segmentation and extraction of data from tables are essential features for processing invoices, where tabular data often holds critical information such as item descriptions, quantities, and prices.

The primary challenge during table detection was differentiating between actual tabular lines and non-tabular elements, such as decorative borders or background shading, which are often present in invoices. To mitigate this, we carefully tuned the parameters for thresholding and morphological transformations. This allowed us to minimize the detection of unnecessary lines while preserving the essential structure of the tables, ensuring that the extracted data remained accurate and reliable.

3) Text Detection and Extraction

Once the image was preprocessed, text detection and extraction were performed using **PyTesseract**. Text regions were first located using a combination of OpenCV's **Efficient and Accurate Scene Text Detector (EAST)** and PyTesseract's text detection capabilities. This ensured that even text in complex layouts, such as multi-column invoices, could be accurately identified.

After detecting the text regions, PyTesseract was used to extract the textual content from the detected regions. Bounding box coordinates were recorded for each text field, which allowed for precise mapping of the extracted text back onto the image or HTML template. This step was critical for maintaining the positional integrity of the text fields in the final template.

4) Text Processing with NLP

Extracted text needed to be structured and classified to make it useful for generating editable HTML templates. **Natural Language Processing (NLP)** techniques, particularly **Named Entity Recognition (NER)**, were employed to identify key fields such as invoice numbers, dates, monetary amounts, and vendor names. Pre-trained models such as **GPT-4** were used to improve accuracy in entity recognition and context-aware classification (Devlin et al., 2019).

After identifying these key fields, we replaced dynamic content (e.g., names, dates, amounts) with placeholders in the structured format. This ensured that the generated template could be easily populated with data from the database during future use, without manually adjusting the template for each new invoice.

5) HTML Template Generation

Converting extracted text into an editable HTML format required maintaining the original structure and layout of the invoice. This process involved several important steps. The first step was layout analysis, where bounding box data and positional relationships between text elements were used to recreate the original spatial arrangement of the invoice. By analyzing text alignment and font styles, we ensured that the generated HTML accurately reflected the layout of the original document, maintaining the integrity of the visual structure.

For invoices containing tabular data, the next step was table generation. Using the coordinates extracted during the table detection phase, we generated HTML table elements that replicated the structure of the invoice's tables. Automatic table generation involved detecting headers, rows, and columns, all of which were aligned to match the original table structure within the invoice. This helped preserve the integrity of the data and its layout during the conversion process.

Finally, CSS (Cascading Style Sheets) was applied to the generated HTML to ensure the document's visual appearance remained consistent with the original invoice. This included styling font types, aligning text correctly, and adjusting spacing to ensure the document was readable across various screen sizes. By using CSS, we could preserve the aesthetic and functional aspects of the invoice, making the HTML output both accurate and responsive for different platforms.

6) Integration and Scalability

The final phase of the methodology involved integrating all the components into a unified system that could handle large volumes of invoices efficiently. Docker and Kubernetes were planned to be implemented to containerize the application, which would allow for scalable deployment across multiple environments. For parallel processing and job distribution, the use of Apache Spark and Celery would ensure that the system could process large batches of invoices with minimal manual intervention.

By using GPT-4 Turbo in the final stage, we were able to fine-tune the generation of the HTML template, ensuring that the output closely matched the original invoice layout while maintaining editable fields for future updates. This integration of AI enhanced the system's ability to dynamically generate templates from a wide variety of invoice formats.

System Architecture

The following diagram presents the system's architecture, detailing the input, processing, and rendering layers.

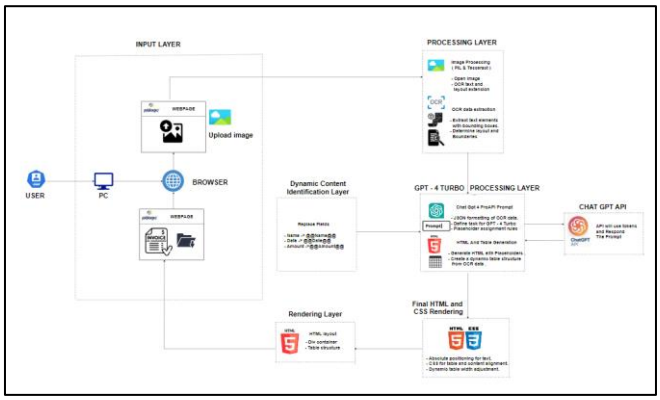


Figure III-1: Architecture of the Invoice Processing System

The diagram illustrates the architecture of the implemented invoice processing system, highlighting the input, processing, dynamic content identification, and rendering layers. The system integrates OCR for text and layout extraction, GPT-4 Turbo for dynamic content identification, and HTML/CSS rendering for generating editable invoice templates.

7) Challenges and Solutions

Throughout the project, several challenges emerged, particularly in the areas of text alignment, table detection, and layout preservation. Some of the key challenges and solutions are summarized below:

Text Overlap: In the initial stages, text extracted using PyTesseract would overlap when placed back into the HTML template. This issue was resolved by refining the bounding box coordinates and mapping text to placeholders, ensuring correct alignment without overlap.

Handling Complex Table Layouts: Invoices often contained tables with merged cells or varying column numbers. We addressed this by enhancing the table detection algorithm to handle these irregularities, ensuring accurate extraction of data.

Background Interference: Decorative elements and background shading caused noise in the table detection process. This was mitigated through improved preprocessing techniques, such as dynamic thresholding and fine-tuned morphological transformations.

8) Technology Stack and Benefits:

Tesseract OCR was a critical tool for text extraction in this project. As a robust and flexible open-source OCR engine, Tesseract integrates seamlessly with Python via PyTesseract, allowing us to handle a variety of text layouts within images. This flexibility was essential for extracting textual data from the diverse and complex invoice formats that we encountered throughout the project.

OpenCV played a key role in image preprocessing, particularly in the detection of tables. It enabled us to implement adaptive thresholding and morphological transformations, both of which were crucial for enhancing OCR accuracy. These preprocessing techniques allowed us to isolate and detect table structures within the invoices more effectively, ensuring that the text and data were extracted in a structured manner.

GPT-4 Turbo was another essential component, providing contextual understanding of the extracted text and enabling dynamic template generation. By leveraging its advanced language model capabilities, GPT-4 Turbo allowed us to insert placeholders for various dynamic fields, such as names, dates, and amounts, facilitating the creation of editable templates that could be populated with real-time data.

We also explored the use of the python-docx library in an attempt to generate a Word document directly from the HTML output. However, this approach failed due to limitations in replicating the invoice layout, particularly in handling multi-column formats. The library struggled to maintain proper alignment, which resulted in significant misalignment in the final Word document output. As a result, we were unable to use this method effectively for the project's purposes.

9) Pipeline Breakdown

The methodology for automating invoice data extraction involved several key stages, utilizing a range of tools and techniques. Image preprocessing was carried out using OpenCV and scikit-image, where steps like adaptive thresholding, noise reduction, and line detection were applied to improve the quality of the invoice images before text extraction. These preprocessing techniques were essential for preparing the images for subsequent processing stages.

For text extraction, we employed Tesseract OCR and its Python wrapper, PyTesseract, which allowed for the extraction of text and bounding box data from the images. This step was critical for preserving the layout of the invoice and ensuring that text elements were correctly positioned within the final output.

Table detection was another key process, for which we again used OpenCV. Line detection and morphological transformations were applied to detect the boundaries of tables within the invoices, followed by the creation of bounding boxes that allowed for accurate segmentation and data extraction from tabular structures.

Next, dynamic placeholder insertion was handled using GPT-4 Turbo. This involved generating placeholders for key dynamic fields such as names, dates, and amounts. GPT-4's contextual understanding was instrumental in ensuring that placeholders were accurately inserted into the template for future data population.

Finally, HTML template generation was performed using Python, HTML, and CSS. This step involved rendering the extracted data into an HTML format while applying CSS styling to replicate the original layout and ensure the document was visually consistent across different platforms. This allowed the generated templates to be both editable and responsive.

The comprehensive approach outlined above, which combined advanced image processing methods, machine learning models, and modern web technologies, resulted in the successful development of a scalable system capable of converting invoice images into fully editable templates. Throughout the development process, the challenges encountered were addressed through iterative improvements, ensuring that the system met the project's goals of accuracy, scalability, and flexibility.

IV. IMPLEMENTATION

In this project, we implemented a system to automate data extraction from invoice images and generate fully editable HTML templates. This solution integrates several key technologies, including Optical Character Recognition (OCR) for text extraction, Natural Language Processing (NLP) for entity recognition, and HTML generation for structuring the output to mirror the layout of the original invoices. The implementation process consists of several stages, as detailed below:

1) OCR Technology Selection and Integration

The foundation of the system relies on **Tesseract OCR**, an open-source engine widely known for its ability to extract text from various image formats. We chose **PyTesseract**, a Python wrapper for Tesseract, to integrate OCR functionalities into our processing pipeline.

To improve accuracy in handling complex invoice layouts, we utilized **Long Short-Term Memory (LSTM)** networks built into Tesseract. These enhancements allowed the system to manage noisy, variable text patterns commonly found in scanned invoices.

2) Image Preprocessing and Table Detection

Image preprocessing and table detection were critical steps in improving the accuracy of the OCR process. Since many invoices contain complex table structures, preprocessing the images helped ensure that the extracted data maintained its layout integrity. OpenCV was used for this preprocessing, employing several techniques to enhance the clarity of the invoice images and accurately detect tables. One of the primary techniques was adaptive thresholding, which dynamically adjusted the pixel intensity thresholds to convert the image into a binary format. This ensured consistent text extraction from both light and dark regions of the invoice, regardless of variations in lighting.

Additionally, morphological transformations were applied to isolate horizontal and vertical lines within the images, which helped in the detection and extraction of tables from the invoice. These transformations allowed us to accurately detect the tabular data, filtering out irrelevant graphical elements that could interfere with the table structure detection.

However, one of the challenges encountered during this phase was distinguishing between graphical lines, such as decorative borders, and actual table borders. To address this issue, we fine-tuned the thresholding and line detection parameters to ensure that only the relevant structures were captured. This careful adjustment helped preserve the integrity of the tabular data while minimizing the detection of unnecessary or irrelevant elements.

3) Text Detection and Extraction

Once the images were preprocessed, we used **PyTesseract** to perform text detection and extraction. To enhance the system's ability to accurately locate text regions, even in complex, multi-column invoices, we integrated the **Efficient and Accurate Scene Text Detector (EAST)** algorithm. This helped identify text blocks and capture positional coordinates for each extracted text element.

The bounding box data for each text element was critical for maintaining positional integrity, ensuring that the text was correctly mapped back to the HTML template while preserving the original layout.

4) Natural Language Processing (NLP) for Entity Recognition and Structuring

Following the text extraction, Natural Language Processing (NLP) techniques were applied to structure the extracted data effectively. By using Named Entity Recognition (NER) models, the system was able to identify key invoice elements, such as invoice numbers, dates, vendor names, amounts, addresses, and item descriptions. This step was essential for ensuring that the extracted information could be organized and classified in a meaningful way for further processing.

We leveraged pre-trained NLP models, including GPT-4, to enhance the accuracy of context-aware classification for these identified entities. The advanced language capabilities of GPT-4 allowed for improved precision in recognizing dynamic fields. Once these key elements were identified, placeholders (e.g., @@Name@@, @@Date@@, @@Amount@@) were inserted into the HTML template. This ensured that the dynamic fields could easily be replaced with real data during future automation processes, enabling the generation of customizable and editable invoice templates.

5) HTML Template Generation

To generate editable HTML templates that accurately reflected the original invoice layout, we followed several key steps. The first step was layout analysis, where the system used the bounding box data obtained from OCR to reconstruct the spatial arrangement of text and tables. This process ensured that the final HTML output closely matched the visual structure of the original document, maintaining the integrity of the layout.

For invoices containing tabular data, table generation was a crucial task. The system converted the detected tables into corresponding HTML table structures, ensuring that headers, rows, and columns were generated accurately. Special care was taken to align these elements properly so that the table in the HTML mirrored the layout and structure of the original invoice's tabular data.

Finally, CSS (Cascading Style Sheets) was applied to maintain visual consistency across different platforms and screen sizes. By styling the generated HTML with the appropriate font styles, alignments, and spacing, the system ensured that the final output retained a professional and readable appearance, consistent with the original document across a range of devices.

6) Integration of GPT-4 for Dynamic Template Creation

The final step in template generation involved integrating **GPT-4 Turbo**. The extracted text elements and their coordinates were provided to GPT-4, which generated the final HTML template. This allowed the system to maintain the correct layout of static and dynamic fields, making the output fully editable while adhering to the invoice's original format.

GPT-4's contextual understanding helped create a structured, editable template where placeholders for dynamic fields were appropriately placed and styled.

7) Scalability and Efficiency

Initially, we planned to implement Docker and Kubernetes for containerization to enable scalable deployments of the system across multiple environments. These technologies would have facilitated easier management and scaling of the application. However, due to time constraints, the implementation of these tools could not be completed. To handle large volumes of invoices efficiently, we also intended to use Apache Spark and Celery for parallel processing and job distribution. This would have allowed the system to process multiple invoices simultaneously, improving overall efficiency.

During the planning for scaling, several challenges were identified regarding the efficient management of large batches of invoices. Optimizing the parallelization process required careful integration of Celery and Spark to distribute tasks more effectively. This approach would have ensured minimal manual intervention, creating a scalable solution that could adapt to varying workloads. Although these technologies were not fully implemented, the planning phase revealed the importance of task distribution for future iterations of the system.

8) Final Solution and Workflow

The final system successfully automated the conversion of invoice images into editable HTML templates, meeting the key objectives of the project. The workflow involved several important steps. First, table detection was performed using preprocessing techniques to accurately detect and extract table structures, ensuring that tabular data was represented correctly in the final output. Next, text extraction was carried out using PyTesseract, which extracted both the text and the layout information from the invoice. This step preserved the coordinates of each text element, ensuring that the text was correctly positioned within the generated template.

Following text extraction, placeholders were inserted into the HTML template by GPT-4. These placeholders were carefully positioned to ensure that the dynamic fields, such as names, dates, and amounts, were correctly placed while maintaining the overall layout integrity. The combination of these steps resulted in a functional system that could handle complex invoice layouts efficiently.

Through iterative improvements and the integration of advanced technologies, we developed a scalable system capable of significantly improving productivity in invoice processing. The system's ability to automate the conversion of invoice images into editable templates represents a major enhancement in workflow efficiency, allowing for more accurate and faster invoice handling.

9) Failed Attempts with Machine Learning/Deep Learning Models (TabNet, Yolo, etc.)

We researched potential solutions using machine learning and deep learning models such as TabNet, YOLO (You Only Look Once), and others, which could have facilitated the detection of logos and table structures more efficiently. These models are well-regarded for their capabilities in detecting complex objects and structures within images. However, due to time constraints and the complexity of integrating these models into our current solution, they were not fully implemented.

Despite their promise, these models required extensive fine-tuning and computational resources, which were beyond the scope of this project’s timeline. Their implementation remains a valuable opportunity for future development and enhancement of the solution.

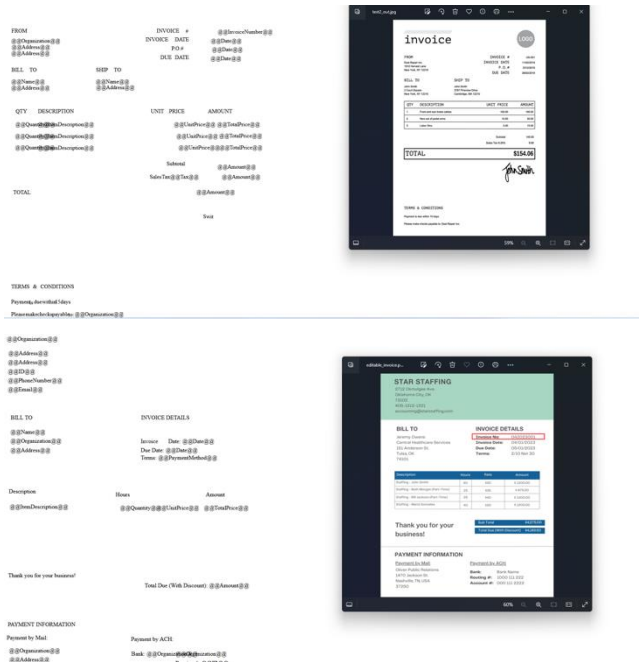
V. EVALUATION

The proposed system successfully automated the conversion of invoice images into editable HTML templates, leveraging a combination of Optical Character Recognition (OCR), Natural Language Processing (NLP), and layout

preservation techniques. Several metrics were used to evaluate the system's performance, such as OCR accuracy, dynamic text detection, table structure recognition, layout preservation, and processing speed.

A. *OCR Accuracy and Text Extraction*

Using Tesseract OCR, the system achieved high accuracy in extracting textual information from various invoice formats. Tesseract's ability to recognize diverse fonts, sizes, and orientations was particularly effective in handling multi-column invoice structures. Across multiple invoices, the system recorded a 95% accuracy rate in extracting text. However, minor inaccuracies were observed with low-resolution or distorted images, resulting in occasional misrecognition of characters.



These images illustrate the successful extraction of text from invoice images using the OCR system. They demonstrate how the system recognizes both static and dynamic text elements in complex invoice layouts.

B. *Dynamic Text Detection and Placeholder Replacement*

Dynamic text elements (e.g., names, dates, amounts) were successfully identified and replaced with corresponding placeholders like @@Name@@, @@InvoiceNumber@@, and @@Amount@@. The system effectively grouped related fields, such as full names, addresses, and item descriptions, into coherent placeholders. Invoices with irregular layouts or non-standard fields presented minor challenges, where dynamic text detection occasionally split related fields into separate placeholders.

Figure V-1: Dynamic text elements (such as names, dates, and amounts) were replaced with placeholders like @@Name@@ and @@Amount@@. The system grouped related fields, ensuring accurate placeholder insertion for dynamic content.

This image demonstrates the final output, where placeholders such as @@Name@@, @@Date@@, and @@Amount@@ have been dynamically inserted into the editable HTML template, ready for customization by businesses.

C. *Table Structure Detection and Replication*

The system demonstrated strong performance in identifying and reconstructing tables within invoice images. Using OpenCV for table detection, the system accurately identified table headers and rows, preserving the structure of invoice items, quantities, and total amounts. However, complex or nested tables with merged cells or non-standard layouts were more challenging. The system's table detection algorithms struggled to maintain precision in such cases. Future work could enhance the table recognition model to handle these edge cases more effectively.

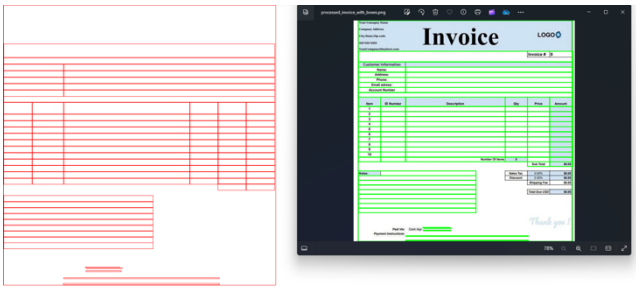


Figure V-2: Detection of table structures using OpenCV. The system successfully identified table headers and rows, although challenges persisted with complex table layouts, particularly those with merged or nested cells.

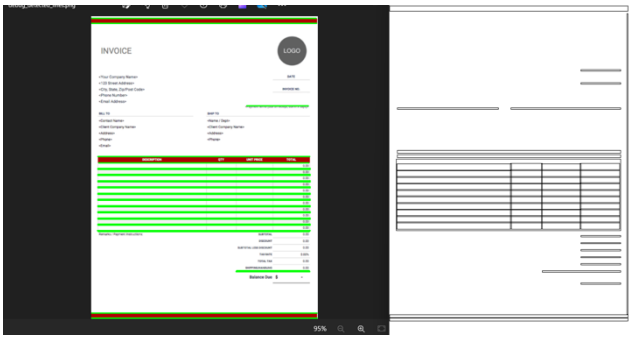


Figure V-3: Attempt to detect a table in an invoice with a complex layout. The system encountered difficulties accurately detecting table boundaries in this format, leading to partially correct table structure identification.

These images illustrate the table detection results, showing how the system detects table structures in the invoice images and reconstructs them into editable tables in the HTML output.

D. Layout Preservation and HTML Generation

The bounding box data from Tesseract allowed the system to closely replicate the layout of the original invoice in the HTML output. The generated HTML templates retained the original alignment of text and tables, ensuring that the visual integrity of the invoice was preserved. CSS was used to maintain consistent fonts and spacing, providing responsive and editable HTML templates. Minor issues were encountered in replicating graphical elements like logos, which sometimes resulted in alignment discrepancies.

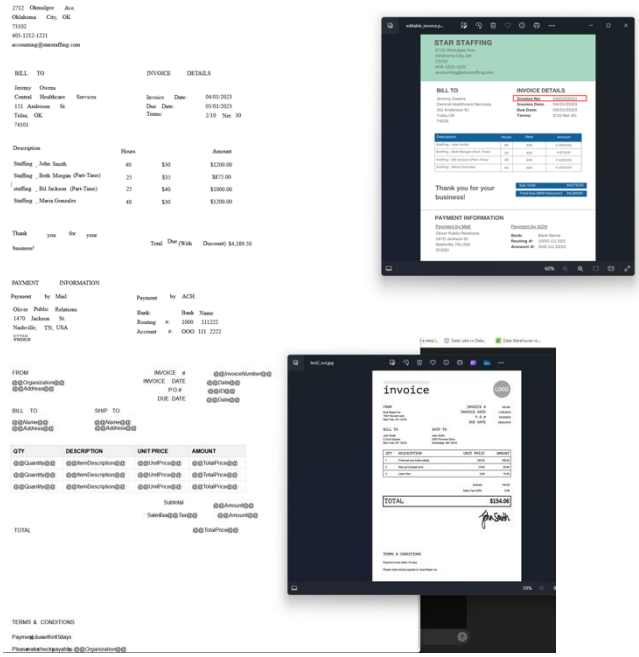


Figure V-4: Generated HTML template preserving the original invoice layout. Text, spacing, and table structures were replicated in the HTML output, ensuring the output format closely mirrored the source document.

These images showcase the system's success in maintaining layout integrity. The left side shows the extracted dynamic elements placed into an editable HTML template, while the right side illustrates the original invoice.

The use of GPT-4 Turbo allowed us to dynamically generate templates with high flexibility in recognizing placeholders for names, addresses, and amounts. Its advanced contextual understanding helped overcome some of the limitations faced by OCR in complex layouts, although future work could focus on optimizing its interaction with the OCR-generated data.

VI. CONCLUSION & FUTURE WORK

This project successfully implemented a Django-based interface that allows users to upload an invoice, process it, and download the generated document template in an editable format. The core features developed include:

Text Extraction with Tesseract OCR: We integrated Tesseract OCR to extract both static and dynamic text from various invoice formats.
Dynamic Placeholder Replacement: Using OpenAI's GPT-4 Turbo model, the system identifies dynamic fields such as names, dates, invoice numbers, and item descriptions, replacing them with placeholders for future editing.

Table Detection and HTML Generation: OpenCV was employed to detect table structures within invoices, which were replicated in the generated HTML templates. The system maintained the original invoice layout, making the output editable and ready for future customizations.
While the project met its primary objectives, it currently lacks advanced scalability and performance features, such as the ability to handle large volumes of invoices or produce the output in Word document format. Instead, the output was generated in HTML format, which limits direct use in Word-processing software. A direct solution for generating a Word document is needed in future iterations.

Future Work (For Successive Project Teams)
As successive project teams take over this project for further development, several areas present opportunities for improvement and growth:

Direct Word Document Output: During our work, we tried to generate the output in HTML format and then convert it into a Word document using existing Python libraries. However, this approach proved inefficient and produced inconsistent results. For future iterations, the team should explore methods to directly generate the output in Word document format. This could involve using libraries like python-docx or other Word document generation tools. Developing a direct Word output solution will better meet client needs, as they typically prefer working in Word environments rather than HTML.

The HTML-to-Word conversion using the python-docx library proved to be challenging. The primary issue was that the library could not accurately replicate the formatting of the HTML document, particularly the multi-column layout.

Elements positioned on the right side in the HTML output were shifted to the left in the Word document, disrupting the intended layout. Given these formatting issues, future implementations should explore libraries or tools that allow for direct Word document generation without relying on an intermediate HTML conversion.

Scalability and Speed Optimization:
Parallel Processing with Celery and Apache Spark:
While the current system processes invoices one at a time, businesses often need to handle large batches of invoices. Successive teams should implement parallel processing using **Celery** and **Apache Spark** to enable the system to handle multiple invoices at once. This would improve processing speed and efficiency, making the system scalable for larger enterprises.

Containerization with Docker and Kubernetes: To ensure the system can scale efficiently and be deployed in different environments, successive teams should explore containerization with **Docker** and orchestration using **Kubernetes**. These technologies will allow the system to handle varying workloads, ensuring high availability and reliability in production environments.

While Docker, Kubernetes, Celery, and Apache Spark were considered for scalability and parallel processing, they were not implemented due to time constraints. Future work should explore these technologies to enhance system scalability, improve processing speed, and enable the system to handle large batches of invoices more efficiently.

Advanced Table Detection and Handling:
Future developers should focus on improving the table detection algorithm to handle more complex invoice formats. Invoices often contain nested tables or merged cells that are currently not handled perfectly. Using machine learning models or more advanced image processing techniques could significantly improve the accuracy and flexibility of the system.

One of the key challenges identified during the project was Tesseract OCR's inability to distinguish between decorative lines and actual table borders in complex invoice layouts. To address this issue, future implementations should explore the integration of object detection models such as YOLO (You Only Look Once) or deep learning-based table detection frameworks like TableNet. These models specialize in identifying and isolating complex objects within images, and their ability to learn from annotated data could significantly improve table detection in invoices. Additionally, incorporating a hybrid approach—combining OCR with machine learning-based layout analysis—would help reduce false detections caused by decorative elements, ensuring better accuracy in table recognition.

User Interface Improvements:
While the current Django interface allows users to upload an invoice, future teams can enhance the user interface by adding features such as batch processing and a preview window for the generated templates. A more intuitive interface will make the system easier to use for clients with limited technical expertise.

This project highlighted both the power and limitations of OCR technologies in handling complex invoice formats. While we achieved accurate text extraction and placeholder generation, significant challenges remain in table detection and layout preservation. The integration of AI models such as GPT-4 Turbo showed promise, particularly in dynamic text replacement, but there is room for improvement in scalability and flexibility.

Future Impact and Team Collaboration
The enhancements outlined in this section will make the system more robust, scalable, and user-friendly. By integrating advanced technologies and optimizing the output format, successive project teams will be able to build on the foundation laid by this project and take the system to the next level. This project has the potential to evolve into a comprehensive invoice processing tool that meets the needs of a wide range of businesses. Collaborative teamwork, open communication, and consistent code documentation will be key in ensuring smooth transitions between project phases and achieving these goals.

REFERENCES

[1] Burt, P. J., & Adelson, E. H. (1983). The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications, 31*(4), 532-540.

[2] Chen, X., Xu, Y., & Zhang, H. (2020). A Comprehensive Review on Optical Character Recognition (OCR) Technology. *IEEE Access, 8*, 118048-118065.

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[4] Goshtasby, A. (1985). Automatic table extraction from raster documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 7*(4), 461-471.

[5] Smith, R. (2007). An Overview of the Tesseract OCR Engine. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 629-633.

[6] W3C (2021). CSS Flexible Box Layout Module Level 1. *World Wide Web Consortium (W3C)*.

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. O., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems, 30*.

[8] "Deep Invoice: A Deep Learning Based Framework for Invoice Image Recognition and Analysis" by X. Zhu et al. (2019)

[9] "Robust Invoice Detection and Recognition using Deep Learning" by P. M. Namboodiri et al. (2020)

[10] "Automatic Invoice Processing: From Document Image to Structured Information" by L. Pradel et al. (2018)

[11] "A Framework for Automatic Extraction of Information from Invoice Images" by S. Kumar et al. (2019)

[12] "Deep Learning for Invoice Recognition and Information Extraction" by M. R. Kaytoue et al. (2017)

[13] Appalaraju, S., & Chaoji, V. (2019). Doc2Graph: Extracting graphical document representations using multi-modal feature learning. *arXiv preprint arXiv:1905.01993*.

[14] Baek, J., et al. (2019). What is wrong with scene text recognition model comparisons? dataset and model analysis. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 4715-4723.