IJISEM
INTERNATIONAL JOURNAL OF INNOVATIONS IN SCIENCE ENGINEERING AND MANAGEMENT

International Journal of Innovations In Science Engineering And Management

Invoice Data Extraction and Validation Pipeline Using OCR and Intelligent Rule-Based Verification

Aditya Sonawane[1], Sarvesh Mogal[1], Mayur Sahare[1], Vaishnavi Gughe[1], S. N. Khalane[1]

[1]*Department of Computer Engineering, MVP Karmaveer Adv. Baburao Ganpatrao Thakare College of Engineering, Nashik, India*

OPEN ACCESS

Volume: 5

Issue: 4

Month: November

Year: 2026

ISSN: 2583-7117

Published: 18.11.2026

Citation:

Aditya Sonawane, Sarvesh Mogal, Mayur Sahare, Vaishnavi Gughe, S.N. Khalane "Invoice Data Extraction and Validation Pipeline Using OCR and Intelligent Rule-Based Verification" International Journal of Innovations In Science Engineering And Management, vol.5, no.4, 2026.

DOI:

10.69968/ijisem.2026v5i4331-342

This work is licensed under a Creative Commons Attribution-Share Alike 4.0 International License

Abstract

Abstract This paper provides a comprehensive overview of the process for information retrieval from invoices. Invoices serve as proof of purchase and contain important information, including the date, description, quantity, and the price of goods or services, as well as the terms of payment. Invoice processing remains a document-intensive task in accounts payable (AP) because invoices arrive as heterogeneous PDFs, scans, photographs, and digitally generated files. Manual transcription and verification introduce delays and data-entry errors, while template-dependent systems struggle when vendor layouts change. This paper presents an invoice data extraction and validation pipeline that combines optical character recognition (OCR), pretrained document understanding, structured field extraction, deterministic validation, duplicate detection, persistent storage, and human review. The pipeline is event-driven and serverless: an invoice uploaded to a Cloud Storage bucket triggers a Cloud Function that calls the Google Cloud Document AI Invoice Parser, applies a configurable validation engine, and writes the result to separate clean and flagged BigQuery tables, with a Looker Studio dashboard and a Cloud Logging audit trail for monitoring. The contribution is not a new OCR model but an application-level architecture that treats extraction and validation as separate, connected stages. The validation layer checks required fields, arithmetic consistency, tax values, dates, vendor and value ranges, duplicate invoice identifiers, and confidence-based exceptions. A reproducible evaluation protocol is defined using field-level exact match, precision, recall, F1-score, validation pass rate, human-review rate, duplicate-detection accuracy, and processing time, with project targets of at least 95% field accuracy and 80% automated validation pass rate. Measured results are to be reported once the evaluation on a ground-truth dataset is complete.

Keywords: *Invoice Processing; OCR; Document Understanding; Information Extraction; Duplicate Detection; Accounts Payable.*

INTRODUCTION

Organizations receive invoices in many representations, including digitally generated PDFs, scanned documents, photographs, and files produced by different vendors. Although the semantic content of an invoice is relatively stable, its visual presentation is not. A typical invoice contains a supplier name, invoice number, dates, purchase-order information, line items, quantities, unit prices, tax values, a subtotal, and a total amount, yet these fields can appear in different locations and use different labels, fonts, table structures, and number formats. Converting an invoice into a structured record is therefore an information-extraction problem rather than simple text copying. In many accounts-payable (AP) teams this conversion is still performed manually, which is slow and allows data-entry errors, duplicate payments, and vendor disputes to reach the accounting record.

Prior research has investigated OCR, layout-aware document understanding, deep learning, and machine-learning approaches for invoice extraction. Layout-aware models such as LayoutLM combine textual and spatial information and show that document layout is important for information extraction [1]. Research on invoice processing has also shown that model performance varies with the supplier and layout distribution, which makes evaluation on unseen layouts important [2]. Recent work continues to explore deep-learning OCR pipelines and post-processing strategies for invoices [3]. These studies establish the importance of extraction accuracy, but a practical financial workflow requires another layer: extracted values must be checked before they are accepted for downstream accounting operations.

The proposed pipeline addresses this operational gap. When an invoice arrives in a cloud storage bucket, the system sends it to a pretrained invoice parser, converts the returned entities into a normalized internal schema, executes deterministic validation rules, writes the result to a clean or a flagged table in a cloud data warehouse, and routes exceptions for human review. Google Cloud Document AI provides an Invoice Parser that performs OCR and entity extraction and returns structured entities with confidence information [4], [5]. The system therefore relies on pretrained document intelligence rather than training a new OCR model from scratch.

The main contributions of this work are: (1) a modular end-to-end architecture that separates document extraction from business validation; (2) a validation layer covering required fields, arithmetic relationships, tax consistency, dates, vendor and value ranges, duplicate identifiers, and invalid values; (3) a confidence-aware exception mechanism that supports human review instead of blindly accepting low-confidence predictions; and (4) a reproducible evaluation framework for measuring extraction and validation performance. The approach is intended for student-scale prototyping and for future extension toward enterprise invoice workflows.

The remainder of this paper is organized as follows. Section II reviews related work and the research gap. Section III states the problem, objectives, and research questions. Section IV describes the system architecture, Section V the validation model, and Section VI the implementation. Section VII defines the experimental methodology, Section VIII discusses the results framework, Section IX summarizes advantages, limitations, and applications, Section X outlines future work, and Section XI concludes the paper.

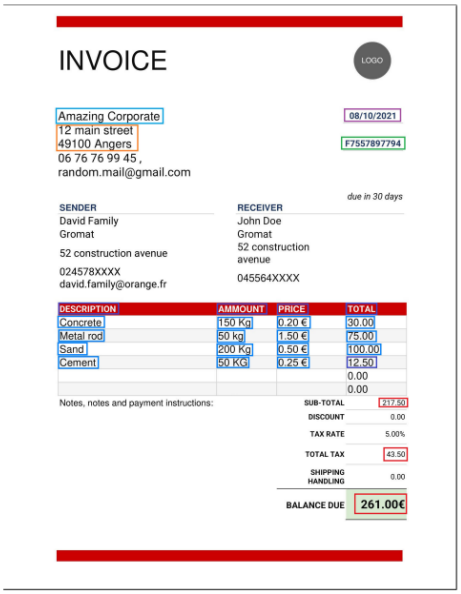


Figure 1 An Invoice Sample

The proposed system processes uploaded invoice documents in PDF or image formats. Initially, the system uses OCR technology to recognize and convert the text present in the invoice into machine-readable data. After text recognition, the system identifies specific invoice fields based on their labels, positions, and surrounding textual information. The main fields considered in this project include Invoice Number, Invoice Date, Quantity, Individual Item Calculations, and Total Amount. The extraction process focuses on identifying highlighted regions in the invoice document. The Invoice Number is identified using keywords such as "Invoice No." or "Invoice Number", while the Invoice Date is detected using date patterns and related labels. Similarly, the Quantity field is extracted from the item description table. The system identifies the corresponding item price and performs calculations to determine the individual item amount by multiplying Quantity by Unit Price. Finally, the extracted values are used to calculate and verify the subtotal and total invoice amount. To improve data accuracy, the proposed pipeline incorporates validation mechanisms that check missing fields, incorrect numerical values, calculation mismatches, and duplicate invoice records. The calculated amounts are compared with the corresponding totals mentioned in the original invoice to identify possible discrepancies. After successful extraction and validation, the structured invoice information is stored in an Excel file for further analysis, tracking, and financial record management. The primary objective of this project is to reduce manual invoice processing, improve data accuracy, minimize calculation errors, and provide an efficient automated solution for invoice information management.

Invoice Modeling

Defining a suitable representation of an invoice is an important step for clearly understanding its specifications. An invoice can include inputting data such as the invoice number, date, and amounts, as well as assigning it to a specific customer or project. In [22], a semantic network was used to describe the invoice domain by different levels of abstraction. Before going on through invoice processing techniques, we propose here a model that better focuses on relevant extraction tasks that are expected to be handled by an invoice processing application. We chose to initially limit the scope of invoice extraction. Figure 1 illustrates a basic sample of an invoice, emphasizing key information sought by automated document processing tools. The extraction of specific fields, such as the invoice date (highlighted in the purple box), supplier address (in the orange box), and organizational providers (within the cyan box), is crucial. This survey places particular emphasis on table extraction, as indicated by data enclosed in blue and red boxes. Additionally, it is worth noting that the invoice contains other pertinent information that may be valuable for Named Entity Recognition (NER) processes, including the

identification of both the sender and receiver. Figure 2 provides a comprehensive view of the typical content of a invoice by means of an UML class diagram. Different types of information must be highlighted such as addresses, tables, dates, and actors (organizations or individuals identified on the invoice). This selected information seems coherent with the analysis of multiple invoice models and the usual requirements of the companies. One may identify 6 groups of data: • Actors: individuals or companies involved in the invoice, such as a customer or a supplier. • Independent fields: fields whose value is not linked to one of the other following fields and that often represent essential data for the invoice. • Information on the document: information specific to the management of the document, such as its name or identifier in the file system, the dates of creation and processing of the document – all the data that are not extracted from the document but that come from its processing. • Addresses: addresses contained in the document, with if possible precision on their types, billing address, delivery, or sender for example • Tables: data tables are essential in invoices. They often include several lines of invoiced items, prices, quantities. . . • Date: the set of dates, specific to the invoice processes such as the date of the edition of the invoice, the date of payment or of delivery. Among these data, tables are considered complex to extract in this model because they often contain a large amount of structured data that needs to be parsed and understood. Companies need to perform verification operations on the table data, such as verifying VAT amounts and rates, or ensuring that the sum of the table lines matches the invoice amount. Efficient methods for extracting and analyzing table

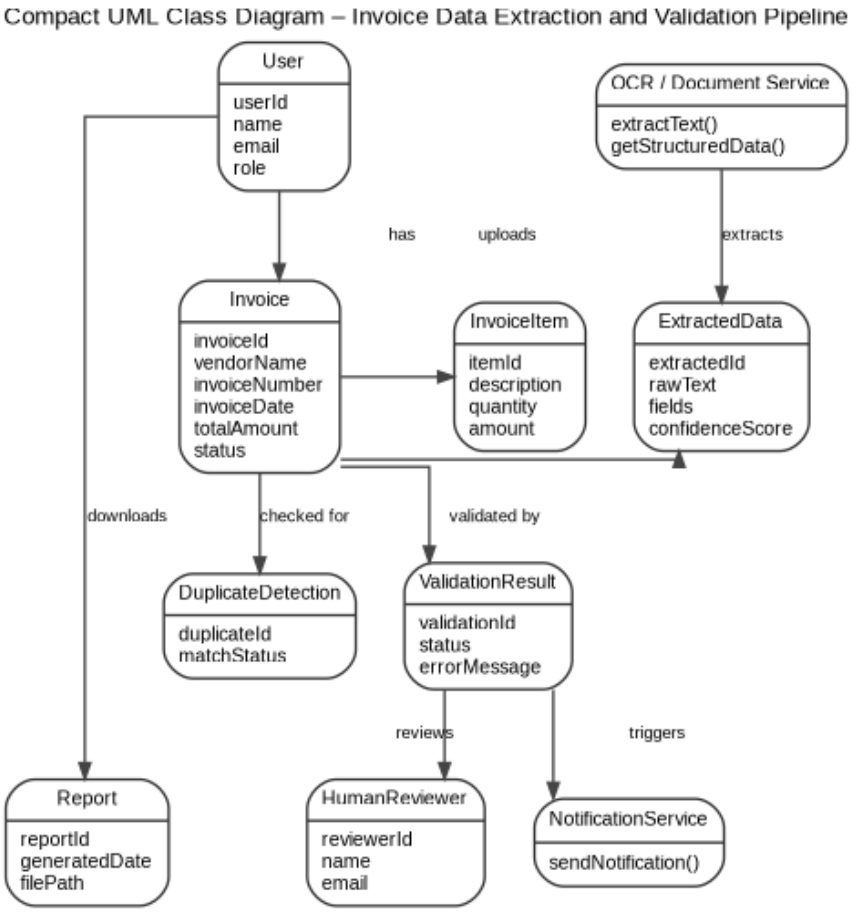


Figure 2 An Uml Model for Invoice

Data Are Crucial Due to the Time-consuming and Error-prone Nature of the Process

RELATED WORK

Rule-based and Machine-learning Approaches

Invoice information extraction has developed from template- and rule-based systems toward OCR, machine learning, and multimodal document-understanding approaches. Traditional systems often rely on fixed locations, regular expressions, dictionaries, or vendor-specific templates. Such methods can be effective for stable layouts but require maintenance when a supplier changes its invoice design. The survey by Saout *et al.* [6] reviews OCR, named-entity recognition, and table-extraction techniques for invoices. It identifies document quality, invoice content, layout diversity, the availability of

annotated data, and table quality as the main factors when choosing a method, and it points to hybrid methods that combine rule-based and neural approaches.

Ha and Horák presented an invoice information-extraction approach that combines OCR output with text analysis and layout features. Their OCRMiner system illustrates the value of coordinating textual and spatial evidence when extracting invoice metadata [7].

Layout-aware and Deep-learning Approaches

LayoutLM introduced a document representation that jointly models text and layout, establishing an important direction for document image understanding [1]. Later studies evaluated layout-aware machine-learning methods for invoice processing and observed that supplier-layout imbalance can affect generalization. In particular, research on long-tail suppliers found a larger accuracy gap between in-sample and out-of-sample layouts for some baselines than for a pretrained transformer approach [2].

Recent invoice-processing studies continue to investigate deep-learning OCR, visual features, and post-processing. A 2026 study described a pipeline that combines image preprocessing, OCR, and post-processing for invoice images [3]. Other work has explored business-document extraction using multimodal methods and fine-tuning strategies [8], additional pre-training tasks and post-processing for LayoutLM on business documents [9], and zero-shot invoice extraction that combines the Donut foundation model with spatial prompt tuning on the SROIE v2 dataset [10]. Zero-shot approaches reduce the need for template-specific training, but unseen formats and multilingual documents remain challenging. Approaches that develop a dedicated model are valuable, but they may increase the training-data, deployment, and maintenance requirements for a student or small-organization system.

Commercial Document-understanding Services

Commercial document-understanding services provide another route. The Google Cloud Document AI Invoice Parser is a pretrained processor designed for invoice OCR and entity extraction. Its documented outputs include header and line-item entities, confidence values, and normalized representations for several monetary and date fields [4], [5]. The present work builds on this capability but adds an application-specific validation layer. The distinction matters: extraction answers “what value is present?”, whereas validation asks “does the extracted value satisfy the rules of the transaction?” Table I summarizes representative approaches.

Table 1: Comparison of Representative Approaches

Approach	Main idea	Strength	Limitation
Fixed templates	Predefined positions/rules	Simple for stable vendors	Sensitive to layout changes
OCR + rules	OCR followed by regex/dictionaries	Easy to implement	Weak semantic/layout understanding
Layout-aware ML	Text + spatial features	Handles document structure	Requires training/evaluation data
Deep-learning OCR	Learned detection/recognition	Strong on visual variation	Model/deployment complexity
Pretrained Document AI	OCR + entity extraction	Rapid integration, broad schema	External service/dependency
Proposed pipeline	Pretrained extraction + validation + review	Separates extraction from business controls	Requires careful rule design and testing

Research Gap and Proposed Contribution

The literature indicates that invoice-extraction accuracy is strongly influenced by layout variability, OCR quality, and the representation of document structure. However, an extraction benchmark alone does not represent the complete AP decision process. A system may correctly identify a total amount but still require review if the total does not equal the subtotal plus tax, if the invoice number already exists for the supplier, or if a mandatory purchase-order reference is missing. The gap addressed by this work is therefore the integration of pretrained document extraction with an explicit, auditable validation stage.

The proposed architecture is deliberately hybrid. Machine intelligence is used where visual and semantic interpretation is difficult, while deterministic rules are used where correctness can be expressed mathematically or logically. This division avoids using a learned model for checks that can be performed exactly, and it makes the system explainable: an exception can be reported as “total mismatch,” “duplicate invoice number,” or “missing required field” instead of only a low-confidence prediction.

The contribution is consequently an engineering and evaluation framework rather than a new OCR algorithm. The system demonstrates how a pretrained invoice parser can be embedded in a controlled data pipeline, how validation rules can be formalized, and how automatic acceptance can be separated from human review.

Problem Statement and Research Objectives

Problem Statement

Manual invoice processing requires an operator to read a document, locate the relevant fields, enter them into a system, and independently verify totals and tax values. The task becomes difficult when invoices differ in layout, contain low-quality scans, or use inconsistent numeric representations. A system that only performs OCR may recover text without knowing which text corresponds to the invoice number, supplier, tax, or total. Conversely, an extraction system that produces structured values without validation can propagate OCR or parsing errors into accounting records. The research problem is therefore formulated as follows: how can heterogeneous invoices be transformed into structured records while applying transparent validation rules and routing uncertain or inconsistent cases to human review?

Research Objectives

- Develop an end-to-end, event-driven pipeline for invoice upload, extraction, normalization, validation, storage, and review.
- Extract header and line-item information from heterogeneous invoice documents using a pretrained document processor.
- Validate arithmetic relationships among subtotal, tax, and total values rather than relying solely on OCR confidence.
- Detect duplicate invoices using stable identifiers such as supplier and invoice number, with additional fields used to reduce false matches.
- Introduce confidence-based exception handling so that low-confidence fields are reviewed instead of being silently accepted.
- Store validated and flagged invoices in separate structured tables and provide a monitoring dashboard and audit trail.
- Define measurable evaluation criteria that distinguish extraction quality from validation quality and processing performance.

Research Questions

RQ1: How effectively can a pretrained document processor convert heterogeneous invoice documents into the target structured schema?

RQ2: Which deterministic validation rules can identify common extraction and business-data inconsistencies?

RQ3: How should confidence scores and rule failures be combined to decide whether an invoice can be accepted automatically or requires review?

RQ4: What evaluation measures provide a reproducible assessment of invoice extraction and validation quality?

Proposed System Architecture

The pipeline is an event-driven, serverless, cloud-based intelligent document processing system. It is organized into seven functional stages: document intake, extraction, normalization, validation, routing and storage, human review, and monitoring and audit (Fig. 1 and Table II). A serverless function coordinates the flow, so that volume spikes can be absorbed without a permanently running server.

Document Intake

AP staff upload a PDF, JPEG, or PNG invoice to a Cloud Storage bucket, directly or through a web interface. The upload event triggers a Cloud Function, which verifies the file type and size limit, assigns a unique processing identifier, and records the link to the original document so that the source can always be inspected. The original document is retained only according to the configured retention policy. This stage also rejects unsupported or malformed input before any external processing.

Ocr and Document Understanding

The Cloud Function submits the document to the pretrained Document AI Invoice Parser. Google documents the Invoice Parser as a pre-made model for invoices that can also be uptrained with additional labeled data [4]. The parser can return fields such as invoice number, supplier name, invoice date, due date, net amount, tax amount, total amount, currency, and line-item description, quantity, unit price, and amount [4]. Each returned entity includes a type, the extracted text, a confidence value, and optional normalized values [5].

Data Extraction

Once the OCR has been applied, we are generally left with a set of PDF documents that are expected to be searchable and exploitable. Let us first begin with a general consideration of possible data extraction at this stage. At first glance, we may consider the visual aspect of the document and the relative positions of the information that it contains. The work of Taylor et al. [132] presents an overview of the problem of document extraction from scanned documents. This article highlights the problems of alignment of the text. It also

highlights that only part of the information is relevant to extract.

Data Validation

Data Validation is an important step in the Invoice Data Extraction and Validation Pipeline. It ensures that the information extracted from invoices using Optical Character Recognition (OCR) is accurate, complete, and consistent. After extracting invoice details, the system validates important fields such as invoice number, invoice date, vendor name, quantity, unit price, and total amount.

The validation process identifies missing values, incorrect formats, duplicate invoices, and calculation errors. If any error is detected, the system marks the invoice for manual review. This process improves data accuracy, reduces manual effort, and ensures reliable invoice processing.

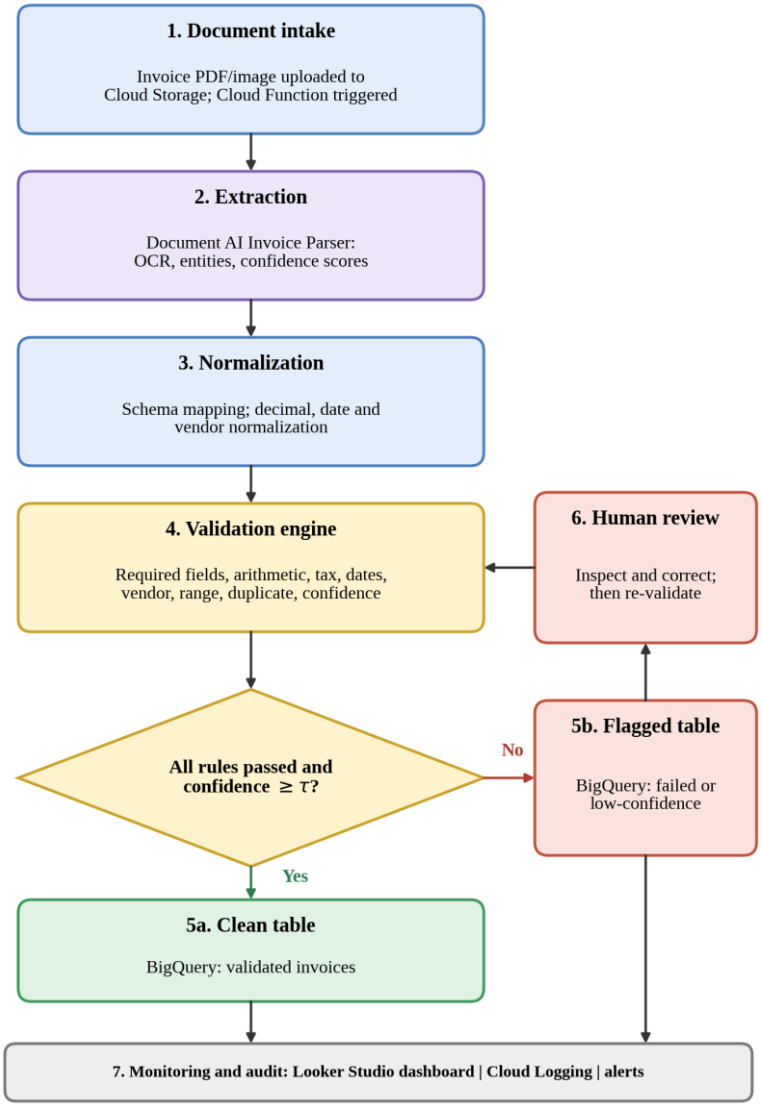


Figure 2. End-to-end Processing Flow of the Proposed Pipeline.

Table 2: Pipeline Stages

Stage	Input	Processing	Output
1. Intake	PDF/image invoice	File-type and size checks; storage in Cloud Storage; Cloud Function trigger	Accepted document, processing ID
2. Extraction	Accepted document	Document AI Invoice Parser: OCR and invoice entity extraction	Entities with confidence (JSON)
3. Normalization	Entities	Field mapping; numeric, date and vendor normalization	Canonical invoice object
4. Validation	Canonical object	Rule engine and confidence checks	Pass/warn/fail results

5. Routing and storage	Invoice and results	Write to BigQuery clean or flagged table	Searchable record
6. Review	Flagged invoice	Human verification, correction and re-validation	Resolved status
7. Monitoring and audit	Records and logs	Looker Studio dashboard, Cloud Logging, alerts	Status reports, audit trail

Entity Normalization

Provider-specific responses are converted into an internal schema. This reduces coupling between the downstream stages and the external service and allows the validation logic to operate on stable application fields. For example, monetary values are represented using decimal arithmetic rather than binary floating-point operations, dates are normalized to a common representation, and supplier names are normalized for duplicate comparison while the original extracted text is preserved for display.

Validation Engine

The validation engine evaluates each normalized invoice against independent rules. A rule produces a pass, warning, or fail result together with a human-readable reason. Independent rule results are retained so that a reviewer can understand why an invoice was accepted or flagged. The rule set is configurable, so new business logic can be added without changing the extraction or storage stages.

Routing, Storage, and Audit Record

Results are saved to BigQuery in two separate tables: a clean table for invoices that passed all checks and a flagged table for invoices that failed a rule or contain low-confidence fields. A practical schema contains invoice metadata, extracted header fields, line items, validation results, and processing status. For every processed invoice, the link to the original document and the field-level confidence scores are retained. Storing the raw provider response separately from the normalized fields supports debugging and later reprocessing.

Human-in-the-loop Review

Automatic processing should not be interpreted as automatic approval of every document. When extraction confidence is below a configured threshold or a validation rule fails, the invoice is routed to the flagged table. The reviewer can inspect the source document and the extracted values, correct fields, and record the resolution. A corrected invoice is re-validated before it is stored as a final result. This design follows the principle that uncertain financial records should be visible rather than silently propagated.

Monitoring and Audit

A Looker Studio dashboard tracks processing status, invoice statistics, and the human-review rate. Cloud Logging maintains an audit trail of processing identifiers, timestamps, validation outcomes, and reviewer actions, and alerts can notify the AP team when invoices need attention.

Validation Model

The central design principle is to avoid treating OCR confidence as a substitute for business validation. Confidence indicates the processor's certainty about an extracted entity; it does not prove that the value is correct in the context of the transaction. The proposed system therefore combines confidence-aware checks with deterministic validation.

Required-field Validation

Mandatory fields may include supplier name, invoice number, invoice date, subtotal or net amount, tax amount, and total amount, depending on organizational policy. A missing required field produces a validation failure or a review flag. Optional fields such as the purchase-order number or due date can be treated according to the workflow.

Arithmetic Validation

For a simple invoice, the expected total is the sum of the net amount and the applicable tax, subject to discounts, freight, rounding, and other invoice-specific components. A generalized consistency relation is

$$T_{\text{expected}} = S - D + F + Tax \quad (1)$$

where S is the subtotal, D is the discount, F is freight or other permitted charges, and Tax is the applicable tax amount. The system compares the computed amount with the extracted total using a configurable tolerance ε:

$$|T_{\text{extracted}} - T_{\text{expected}}| \leq \varepsilon \quad (2)$$

The tolerance is important because invoices may use line-level rounding. For example, if the line items sum to 1,000 but the stated total is 1,100 and no tax or other charges are listed, the check fails and the invoice is routed to the review queue.

Line-item Validation

For each line item i, the expected line amount is computed from the quantity and unit price, adjusted for any documented discount:

$$L_i = Q_i \times P_i - D_i \quad (3)$$

where L_i, Q_i, P_i, and D_i denote the line amount, quantity, unit price, and line-level discount, respectively. The system compares the calculated line amount with the extracted line amount. A mismatch is flagged for review rather than automatically overwriting the source value.

Tax Validation

Tax validation checks whether the extracted tax amount is mathematically compatible with the taxable base and the stated rate, when a rate is available. In jurisdictions where multiple tax components are used, the rule engine can compare the sum of the components with the reported total tax; for example, on Indian GST invoices the sum of CGST and SGST, or the IGST amount, should equal the reported total tax. The implementation should not assume a single tax format because invoices may contain multiple rates or jurisdiction-specific components.

Date Validation

Date rules verify that the invoice date is syntactically valid and falls within an allowed business range. Optional rules can compare the invoice date and due date to ensure that the due date is not earlier than the invoice date. Future dates may be allowed for specific business cases and are therefore configurable rather than universally rejected.

Duplicate Invoice Detection

Duplicate detection begins with deterministic identifiers. A strong first-pass key is a normalized combination of the supplier identifier and the invoice number:

$$K_{\text{dup}} = N(\text{Supplier}) \parallel N(\text{InvoiceNo}) \quad (4)$$

where $N(\cdot)$ denotes normalization and $\parallel$ denotes concatenation. Where supplier identity is uncertain, additional evidence such as the invoice date and total amount can be used. A second submission of the same invoice is not written to the clean table; it is routed to the flagged table with the reason "duplicate invoice number." The goal is to reduce duplicate-payment risk without incorrectly merging distinct documents.

Vendor and Value-range Validation

Vendor rules check that a supplier name is present and, where a vendor master list is available, that it matches a known vendor; an unknown vendor produces a warning rather than a silent acceptance. Value-range rules flag unusual values such as negative or zero totals, quantities or unit prices outside configured limits, and amounts far above the historical range for the vendor. All limits are configurable.

Confidence-based Review

Let c_f be the confidence returned for field f and τ_f the configured minimum confidence. A field is flagged when $c_f < \tau_f$. The document-level review decision is

$$\text{Review} = \text{RuleFailure} \vee \exists f\,(c_f < \tau_f) \quad (5)$$

An invoice with Review = false is written to the clean table; otherwise it is written to the flagged table together with the reasons. The threshold should be calibrated empirically on representative invoices, and a single global threshold may be inappropriate because different fields have different error patterns

Implementation

The prototype follows an event-driven serverless architecture on Google Cloud. A Cloud Storage bucket receives invoices; a Cloud Function triggered by each upload calls Document AI, normalizes the response, executes the validation rules, and writes the result to BigQuery. Development was organized in four phases: Document AI setup and accuracy benchmarking on sample invoices, the trigger function, the validation engine, and the BigQuery tables with the monitoring dashboard and alerts.

Technology Stack

Table III lists the technologies used at each layer of the pipeline.

Table 3: Technology Stack

Layer	Technology	Role
Ingestion	Google Cloud Storage	Invoice landing bucket; upload event trigger
Orchestration	Cloud Functions (Python)	Serverless coordination of extraction, validation and storage
Document AI	Google Cloud Document AI Invoice Parser	OCR and invoice entity extraction with confidence scores
Validation	Configurable rules engine (Python)	Required-field, arithmetic, tax, date, vendor, range and duplicate rules
Data store	BigQuery	Separate clean and flagged tables; validation results
Monitoring	Looker Studio	Processing status, invoice statistics, review rate
Audit	Cloud Logging	Audit trail of processing and reviewer actions

Processing Algorithm

Algorithm 1: Invoice Extraction and Validation

1. Receive the upload event for a new invoice in the Cloud Storage bucket.
2. Validate the file type and size; generate a processing identifier and record the link to the original document.
3. Send the document to the Document AI Invoice Parser.
4. Read the extracted entities, confidence values, and normalized values.
5. Map provider entities to the canonical invoice schema.
6. Normalize dates, currency, supplier identifiers, and numeric values.
7. Run the required-field, arithmetic, line-item, tax, date, vendor/range, and duplicate rules.
8. Combine rule outcomes with the confidence-based review condition (5).
9. If Review is false, write the invoice to the clean table; otherwise write it to the flagged table with the failure reasons.
10. Log processing events and results to Cloud Logging and update the dashboard data.
11. For a flagged invoice, let a reviewer correct the fields, re-validate, and store the final result.

Data Model

The information domain consists of the vendor (identifier, name), invoice (number, date, subtotal, total), invoice item (description, quantity, unit price), validation result (status, errors, confidence score), and human review (corrections, re-validation status). Invoice-level fields are stored in the clean and flagged tables, with separate tables for line items, rule outcomes, and processing events. Because BigQuery does not enforce foreign keys, the pipeline maintains a shared invoice identifier that links line items and validation results to the invoice record, keeping the data traceable.

Security, Privacy, and Reliability

Invoice data can contain business-sensitive information, including supplier details, addresses, tax identifiers, banking information, and transaction values. The application should therefore minimize stored data, restrict access to authorized users through role-based access control, use encrypted transport, and protect secrets and service credentials outside source code and version control.

Input handling is also a security concern. The upload path should reject unsupported file types, enforce size limits, and generate safe object names. Backend validation must not trust values supplied by the client, and database queries should use parameterized statements.

Reliability requires explicit handling of external-service failures. If the document processor is unavailable or returns an error, the system should preserve a failed processing state and a diagnostic message rather than creating an apparently valid invoice. Because event-driven triggers can deliver the same event more than once, idempotency based on the processing identifier should be used so that retries do not create duplicate records.

Experimental Methodology

A research evaluation should separate extraction accuracy from validation accuracy. The recommended dataset contains invoices from multiple suppliers and layouts, including digitally generated PDFs, scanned documents, different image qualities, and different table structures. Ground truth should be created by manually verifying each target field against the source invoice. Personally identifiable or confidential financial information should be removed or handled under an approved data-management procedure.

Dataset Design

For an initial student-scale experiment, a minimum of 100 invoices is recommended, distributed across several suppliers and document qualities. A stronger study can use 200–500 invoices and hold out several supplier layouts from the development set. The key principle is that the test set should contain layouts and document conditions that were not used to tune the rules.

Target Fields

Table IV lists the target fields, their ground-truth types, and the suggested metrics.

Table 4: Target Fields and Metrics

Field	Ground truth type	Suggested metric
Supplier name	String	Normalized exact match / F1
Invoice number	String	Exact match
Invoice date	Date	Normalized exact match
PO number	String	Exact match
Subtotal/net amount	Currency	Numeric tolerance
Tax amount	Currency	Numeric tolerance
Total amount	Currency	Numeric tolerance
Line-item quantity	Number	Exact/numeric match
Unit price	Currency	Numeric tolerance

Line-item amount	Currency	Numeric tolerance

Evaluation Metrics

Field-level exact match is the fraction of the N evaluated fields whose normalized prediction exactly matches the ground truth:

$$\text{ExactMatch} = N_{\text{correct}} / N \quad (6)$$

where N_{correct} is the number of correctly extracted fields. For entity extraction where partial matches are possible, precision (P), recall (R), and F1-score are useful. If TP, FP, and FN denote true positives, false positives, and false negatives, then

= TP / (TP + FP) (7)

= Tp / (Tp + Fn) (8)

F1 = 2PR / (P + R) (9)

Two workflow metrics complement the extraction metrics. The automated validation pass rate is the share of invoices that pass all automated checks without human intervention, and the human-review rate is the share that requires AP staff attention:

$$\text{PassRate} = N_{\text{pass}} / N_{\text{total}} \quad (10)$$

$$\text{ReviewRate} = N_{\text{flag}} / N_{\text{total}} \quad (11)$$

where N_{pass} is the number of invoices passing all checks, N_{flag} is the number flagged, and N_{total} is the total number of invoices. Numeric fields should also be evaluated with a tolerance because formatting and rounding may differ while the monetary value remains equivalent. Processing time should be measured from upload acceptance to final response, excluding manual review time when evaluating automated throughput. The project acceptance targets are a field accuracy of at least 95% and an automated validation pass rate of at least 80%.

Validation Evaluation

Validation should be tested using both clean and intentionally corrupted invoices. For example, a test copy of a valid invoice can be modified by changing the total amount, removing a required field, duplicating an invoice number, changing the tax amount, or introducing an invalid date. The expected rule outcome is recorded as ground truth, each test case is executed, and the actual pass/fail result is recorded. This allows validation accuracy to be measured independently of extraction accuracy.

Ablation Study

To demonstrate the value of the validation layer, three configurations can be compared: extraction only; extraction plus deterministic rules; and extraction plus rules plus confidence-based review. The comparison should report extraction metrics, validation detection rate, false-review rate, and processing time. This design directly tests whether the additional controls improve the reliability of the workflow.

RESULTS AND DISCUSSION

Table V lists the metrics to be reported for the final test set together with the project targets. The targets are acceptance criteria, not results. The measured values (highlighted) are to be entered from the project's test run on the ground-truth dataset, and no figure should be reported that was not measured.

TABLE

Evaluation Targets and Measured Values

Metric	Target	Measured value	Interpretation
Invoice count	—	[N]	Number of documents in final test set
Supplier/layout count	—	[N]	Diversity of invoice formats
Overall field exact match	≥ 95%	[XX%]	General extraction correctness
Invoice number exact match	—	[XX%]	Identifier extraction
Date exact match	—	[XX%]	Date normalization/extraction
Total amount accuracy	—	[XX%]	Monetary extraction
Tax amount accuracy	—	[XX%]	Tax extraction
Line-item accuracy	—	[XX%]	Table/line extraction
Automated validation pass rate	≥ 80%	[XX%]	Invoices passing without human intervention
Human-review rate	Tracked	[XX%]	Share of invoices needing AP staff attention
Validation detection rate	—	[XX%]	Injected error detection
Duplicate detection accuracy	—	[XX%]	Duplicate identification
Average processing time	—	[X s]	Automated processing latency

Error Categories for Analysis

After the experiment, errors should be grouped rather than reported only as a single overall percentage. Recommended categories are: (1) OCR character substitutions, (2) incorrect field association, (3) table/line-item boundary errors, (4) decimal or currency formatting errors, (5) tax interpretation errors, (6) missing fields, (7) duplicate false positives, and (8) low-confidence but correct predictions. This analysis can reveal whether future improvement should focus on document quality, extraction, normalization, or business rules.

Discussion Framework

If the extraction metrics are high but the validation detection rate is low, the conclusion should be that accurate extraction alone is insufficient for transaction verification. If validation detection is high but the false-review rate is also high, thresholds or rules may be too conservative. If performance decreases sharply for unseen supplier layouts, the result would support prior findings that layout diversity is a critical evaluation factor [2].

The system is designed so that these outcomes can be interpreted without changing the architecture. Extraction, normalization, validation, and review are modular, which also makes it possible to replace the document processor in future experiments without rewriting the business-validation layer.

Advantages, Limitations, and Applications

Advantages

- Separates visual document understanding from deterministic financial checks.
- Uses a pretrained document processor, reducing the need to train an OCR model from scratch.
- Provides explicit validation reasons that are easier to audit than a single confidence score.
- Supports heterogeneous PDF and image invoices rather than one fixed template.
- Is event-driven and serverless, so processing scales with invoice volume.
- Stores clean and flagged records in separate tables with a dashboard for monitoring.
- Provides a human-review path for uncertain or inconsistent documents.
- Uses a configurable rules engine that can be extended with purchase-order matching, vendor master validation, and approval workflows.

Limitations

- Extraction quality depends on the source document and the external document processor.
- Very poor scans, handwriting, unusual tables, or overlapping text may produce extraction errors and low confidence scores.
- Business validation rules are organization-specific and cannot be assumed to be universally correct.
- Duplicate detection can produce false positives if supplier-identity normalization is weak.
- Use of a cloud document processor introduces an external-service dependency and associated privacy, cost, and availability considerations.
- The system does not approve payments, authorize wire transfers, negotiate disputed amounts with vendors, or perform fraud detection.

APPLICATIONS

Potential applications include AP intake, procurement workflows, vendor-onboarding support, expense-document processing, invoice archiving, and audit preparation. In an enterprise setting, the pipeline could be connected to an ERP or accounting system after appropriate security, compliance, and approval controls are implemented.

FUTURE WORK

Future research can extend the system in several directions. First, the vendor-layout evaluation set can be expanded to quantify generalization across unseen suppliers. Second, a custom extraction model, for example an uptrained Invoice Parser, can be compared with the pretrained processor to determine whether domain-specific training gives a meaningful improvement. Third, multilingual invoices and regional tax formats can be evaluated. Fourth, purchase-order matching can be introduced so that extracted line items are compared against procurement records. Fifth, human-review feedback can be captured as labeled data for threshold calibration or model improvement.

Another direction is explainable validation. Instead of returning only a binary status, the system can produce an evidence trace showing which extracted values were used in each calculation, so that a reviewer can see the subtotal, tax, and total values and the exact equation that identified a mismatch. This is particularly valuable in financial applications where traceability matters.

CONCLUSION

This paper presented a practical architecture for invoice data extraction and validation that combines OCR-based document understanding with deterministic business rules and human review. Invoice automation has two distinct requirements: extracting the correct information from visually diverse documents, and determining whether the extracted information is internally consistent and suitable for

downstream processing. The proposed pipeline uses a pretrained invoice parser for document interpretation and an application-level validation engine for required fields, arithmetic consistency, tax values, dates, vendor and value ranges, duplicate identifiers, and confidence-based exceptions, and it stores validated and flagged invoices in separate tables.

The main research value of the system is the separation of extraction from verification, which makes the workflow modular, auditable, and measurable. The evaluation methodology defined in this paper quantifies field-level extraction accuracy and independently tests the ability of the validation rules to detect controlled inconsistencies. Once measured on a representative ground-truth dataset, the results can support evidence-based conclusions about the system's effectiveness and limitations.

Acknowledgment

The authors thank the Department of Computer Engineering, MVP Karmaveer Adv. Baburao Ganpatrao Thakare College of Engineering, Nashik, and the project guide, Ms. S. N. Khalane, for guidance during the development of the pipeline. The authors also thank Deboistech IT Solutions LLP, Nashik, the project sponsor, for industry mentorship, technical guidance, and project reviews, and acknowledge the documentation and developer resources provided by Google Cloud for understanding Document AI processing and structured entity responses.

REFERENCES

[1] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of text and layout for document image understanding," in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2020, pp. 1192–1200.

[2] F. Krieger, P. Drews, and B. Funk, "Automated invoice processing: Machine learning-based information extraction for long tail suppliers," Intell. Syst. Appl., vol. 20, Art. no. 200285, Nov. 2023, doi: 10.1016/j.iswa.2023.200285.

[3] C. Wang, J. Zhang, and Y. Yu, "Invoiveocr: Deep learning based automatic invoice recognition process," Discover Artif. Intell., vol. 6, Art. no. 636, 2026, doi: 10.1007/s44163-026-01140-3.

[4] Google Cloud, "Uptrain a pretrained processor (Invoice Parser)," Document AI Documentation. Accessed: Oct. 29, 2026. [Online]. Available: https://cloud.google.com/document-ai/docs/uptrain-pretrained-processor

[5] Google Cloud, "Specialized processors with Document AI (Python)," Google Codelabs. Accessed: Oct. 29, 2026. [Online]. Available: https://codelabs.developers.google.com/codelabs/docai-specialized-processors

[6] T. Saout, F. Lardeux, and F. Saubion, "An overview of data extraction from invoices," IEEE Access, vol. 12, pp. 19872–19886, 2024, doi: 10.1109/ACCESS.2024.3360528.

[7] H. T. Ha and A. Horák, "Information extraction from scanned invoice images using text analysis and layout features," Signal Process., Image Commun., vol. 102, Art. no. 116601, Mar. 2022.

[8] Z. Yan, Z. Ye, J. Ge, J. Qin, J. Liu, Y. Cheng, and C. Gurrin, "DocExtractNet: A novel framework for enhanced information extraction from business documents," Inf. Process. Manage., vol. 62, no. 3, Art. no. 104046, 2025, doi: 10.1016/j.ipm.2024.104046.

[9] T. Douzon, S. Duffner, C. Garcia, and J. Espinas, "Improving information extraction on business documents with specific pre-training tasks," 2023, arXiv:2309.05429.

[10] R. R. Charabuddi, "Zero-shot invoice information extraction using foundation models with spatial prompt tuning," Int. J. Intell. Syst. Appl. Eng., 2025. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/7722