

Spring 2025

CMPE-258

Deep Learning

Project Report

FormIQ – Intelligent Receipt Parser

Team : NextGen

Presented to :

Mr. Vijay Eranti

Group Members :

Apurva Karne(018221801)

Chandini Saisri Uppuganti(018228483)

Manjunatha Inti(018192187)

Praful John(018168514)

Objective

FormIQ aims to eliminate manual receipt entry by providing an end-to-end, single-page web application that automates the extraction, structuring, and querying of receipt data. The system leverages state-of-the-art OCR to process both printed and handwritten receipts, transforms raw text into a normalized JSON schema using the Perplexity API, and stores structured data in Amazon DynamoDB. It also includes a natural language chatbot interface, built with FastAPI, to answer user queries over the stored data. The entire pipeline is implemented using open-source tools and pay-as-you-go cloud services, making it fully reproducible on a student budget and easily deployable via Hugging Face Spaces.

Abstract

Receipts present significant challenges for digitization due to their unstructured nature, varying formats, thermal printing degradation, and frequent handwritten annotations. FormIQ is an intelligent, low-code, end-to-end solution designed to automate receipt understanding using modern AI and cloud-native tools. The system integrates three core technologies: PP-OCRv4 for robust text extraction including handwritten content with Tesseract as a fallback for clean, printed text; Perplexity LLM to semantically convert raw OCR output into a strict key-value JSON schema; and Amazon DynamoDB for scalable, serverless document storage and indexing.

To enable intelligent interaction, FormIQ includes a FastAPI-based chatbot capable of dynamically generating PartiQL queries to retrieve information in response to natural language questions. The full workflow is exposed through a responsive Streamlit UI, offering both document upload and interactive query functionalities.

For educational purposes, a lightweight CNN training and evaluation module is integrated into the app, achieving 82% training accuracy on a synthetic 3-class classification task. This showcases how performance metrics such as confusion matrices and accuracy plots can be embedded seamlessly into the same user interface.

By leveraging open-source libraries and pay-as-you-go cloud services, FormIQ demonstrates how a fully functional, production-ready document intelligence pipeline can be built and deployed on a student budget applicable to expense reporting, invoice auditing, and beyond.

- **Text extraction** PP-OCRv4 (hand-writing) with a Tesseract fallback for crisp print.
- **Semantic structuring** A Perplexity LLM coerces the OCR output into a strict key-value JSON schema.
- **Serverless persistence and analytics** Amazon DynamoDB stores each document; a FastAPI chatbot generates PartiQL queries on-the-fly to satisfy user questions.

Introduction

Receipts are ubiquitous documents in both consumer and enterprise workflows, often containing critical information such as vendor identities, itemized purchases, tax summaries, and total transaction values. Despite their importance, receipts are among the most difficult documents to digitize reliably. Their non-standardized layouts, variations in print quality, and frequent inclusion of handwritten annotations make automated understanding a complex challenge. Furthermore, thermal receipts often fade with time, while others include overlapping logos, stamps, and formatting inconsistencies that further hinder traditional extraction methods.

Legacy OCR engines, such as early versions of Tesseract, are capable of transcribing visible text into strings but lack contextual understanding. This forces downstream systems and by extension, developers and data scientists to write complex, heuristic-based logic to infer the semantics of each field. Meanwhile, layout-aware deep learning models like LayoutLM and Donut offer more accuracy and contextual understanding but come with their own barriers: they require large, annotated datasets for fine-tuning and demand compute resources that are often unavailable or impractical in constrained environments like student or early-stage startup projects.

FormIQ addresses this problem space with a text-first, structure-later philosophy. Instead of relying on expensive layout-based annotation or rigid rules, FormIQ focuses on robust text extraction followed by intelligent structuring using large language models. This enables the system to generalize across a wide variety of receipt formats with minimal manual configuration or labeling.

At the core of FormIQ's extraction pipeline is PP-OCRv4, a state-of-the-art optical character recognition model optimized for both printed and cursive handwritten content. For clearer receipts or fallback scenarios, Tesseract 5 provides an efficient and lightweight OCR alternative. Once raw text is obtained, it is passed to a Perplexity-hosted LLM, which semantically analyzes the unstructured content and converts it into a strict, canonical JSON schema. This includes fields such as vendor name, date, line items, subtotal, tax, and total amount making the data immediately usable for storage, analytics, and retrieval.

To persist the structured data, FormIQ integrates Amazon DynamoDB, a serverless, fully managed NoSQL database. Its schema-less design is ideal for storing diverse receipt formats, and its low-latency architecture supports real-time query operations. This eliminates the need for complex backend infrastructure and enables millisecond lookups with virtually zero DevOps overhead.

For user interaction, evaluation, and extensibility, the system is encapsulated within a unified Streamlit interface. This browser-based UI allows users to drag and drop receipt images, view structured outputs, evaluate OCR quality, and access a model training demo for pedagogical purposes. Additionally, the platform includes a FastAPI-powered chatbot, capable of translating

natural language questions (e.g., “What was my total spend last month?”) into PartiQL queries that retrieve answers from the DynamoDB store.

FormIQ is designed as a modular, cloud-native, and cost-efficient prototype. By leveraging open-source technologies and pay-as-you-go cloud services, it demonstrates that intelligent document understanding systems can be built and deployed with minimal financial and operational overhead. The architecture is well-suited for applications in expense management, invoice auditing, tax documentation, and personal finance analytics, with potential for extension into broader enterprise use cases.

FormIQ bridges this gap through a *text-first, structure-later* philosophy:

- **Robust extraction** – PP-OCRv4 for cursive text, Tesseract 5 for clear type.
- **Schema induction** – A Perplexity LLM converts free text into a canonical JSON schema.
- **Serverless storage** – DynamoDB offers millisecond look-ups with virtually zero DevOps overhead.
- **Unified interface** – Streamlit hosts drag-and-drop upload, model-training demo, and the FastAPI-driven chatbot in a single browser tab.

In summary, FormIQ offers a scalable and practical solution for turning diverse, noisy receipt data into actionable insights paving the way for smarter financial systems without requiring deep ML infrastructure or costly annotation pipelines.

High Level Architecture

The FormIQ system is designed to provide an end-to-end pipeline for document understanding, automating the extraction, structuring, storage, and querying of receipt and form data. It integrates OCR, LLMs, serverless cloud storage, and interactive user components into a single, cohesive application.

1. User Interface Layer: The entire workflow is accessible through a unified Streamlit UI, which serves as the central access point for users. It allows document uploads, displays extracted and structured outputs, supports model training and evaluation, and includes an interactive chatbot all within a single browser tab.

2. Text Extraction: Upon document upload, the system initiates OCR processing using PaddleOCR, which is optimized for both printed and handwritten text. In scenarios where high-contrast, clean printed text is present, Tesseract 5 serves as a lightweight fallback engine. The extracted text is unstructured and requires further processing to derive meaning.

3. Semantic Structuring with LLM: The unstructured OCR output is passed to a hosted Perplexity Large Language Model, which semantically interprets the content and maps it into a

predefined JSON schema. This schema captures key fields such as vendor, date, total amount, and itemized purchases, enabling downstream storage and analytics.

4. Cloud Storage Layer: The structured data is persisted in Amazon DynamoDB, a serverless NoSQL database that provides flexible schema support and millisecond-level query latency. DynamoDB enables efficient storage and retrieval of document data without requiring manual infrastructure setup or database maintenance.

5. Backend Services: A lightweight FastAPI backend facilitates communication across all components. It routes user requests from the frontend to the appropriate services (OCR, LLM, DynamoDB), manages chatbot interactions, and exposes secure, scalable APIs for data processing and retrieval.

6. Model Training and Evaluation: FormIQ includes a training and evaluation module to demonstrate model behavior using synthetic datasets. Users can simulate CNN training, evaluate accuracy, and visualize performance metrics such as loss and confusion matrix all rendered within the Streamlit interface.

7. Interactive Chatbot: The system also integrates a chatbot interface that enables natural language queries over stored documents. Users can ask questions like “What was the total on May 1st?” or “List items bought from Target,” and the chatbot powered by FastAPI and PartiQL retrieves relevant data from DynamoDB and returns contextual responses within the UI.

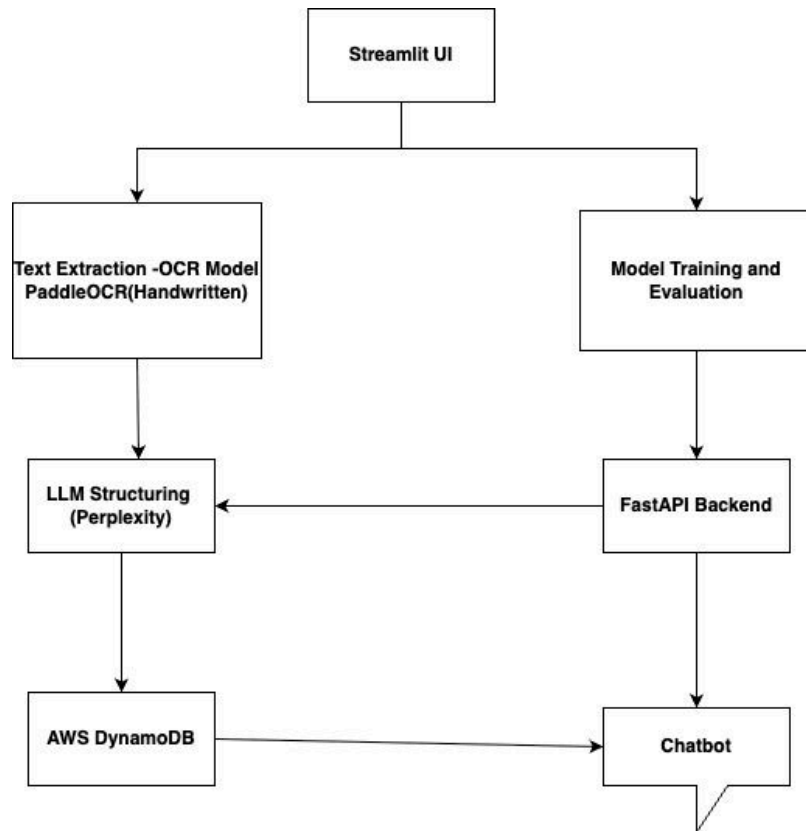


Fig 1: High Level Architecture

Tech Stack

Component	Technology Used	Purpose
Frontend UI	Streamlit	Web interface for uploading and viewing.
OCR Engine	PaddleOCR	Extracts text from scanned/handwritten receipts.
LLM Parser	Perplexity API	Converts raw text into structured JSON.
Backend API	FastAPI	Powers chatbot and data retrieval services.
Database	AWS DynamoDB	Stores structured receipt information.
Evaluation	Matplotlib / Seaborn	Generates confusion matrix and metrics.
Chat Interface	ChatGPT API	Enables natural language QA over receipts.

Working of OCR Module:

Optical Character Recognition (OCR) is the foundational step in FormIQ's document processing pipeline, responsible for converting image-based content into machine-readable text. The system utilizes PaddleOCR, a state-of-the-art deep learning framework optimized for multilingual, handwritten, and printed text recognition. For clean and well-printed inputs, Tesseract 5 serves as a reliable fallback engine.

The OCR process begins by converting the input image into grayscale, reducing color complexity and enhancing contrast. The preprocessed image is then passed through a text detection module, which identifies regions of interest (bounding boxes) where text is likely present. These regions are subsequently cropped and normalized for recognition.

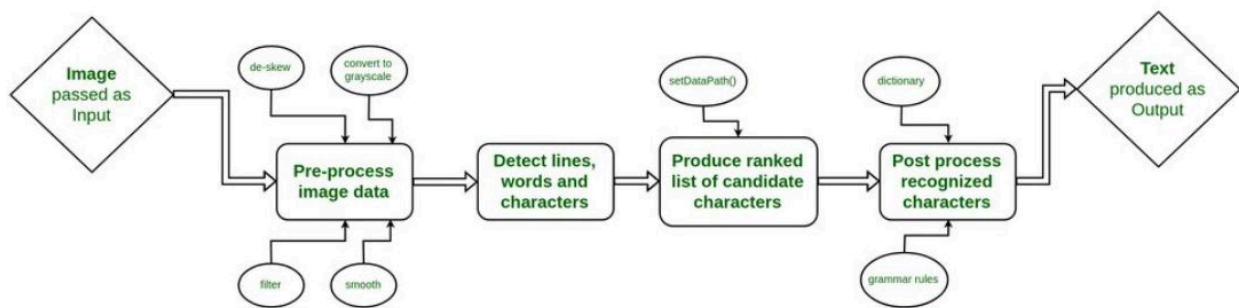


Fig 2: Working of OCR

Each cropped segment is then passed to a text recognition model, typically based on Convolutional Recurrent Neural Networks (CRNN) with a CTC (Connectionist Temporal Classification) decoder. This model outputs the corresponding character sequences for each region. Finally, the recognized segments are aggregated and ordered to form a coherent textual representation of the original document.

This raw text is forwarded to the LLM for semantic structuring. The OCR component ensures that even noisy, low-resolution, or handwritten receipts can be accurately processed, making it robust for real-world document digitization.

OCR Configuration Details

Item	Details
Accepted formats	PNG · JPG · JPEG · PDF (≤ 200 MB)
OCR engines	PP-OCRv4 multilingual handwritten (primary) · Tesseract 5
Pre-processing	Auto-rotate, deskew, 300 DPI resample 300 DPI balances character-level recall with reasonable file size for browser uploads.

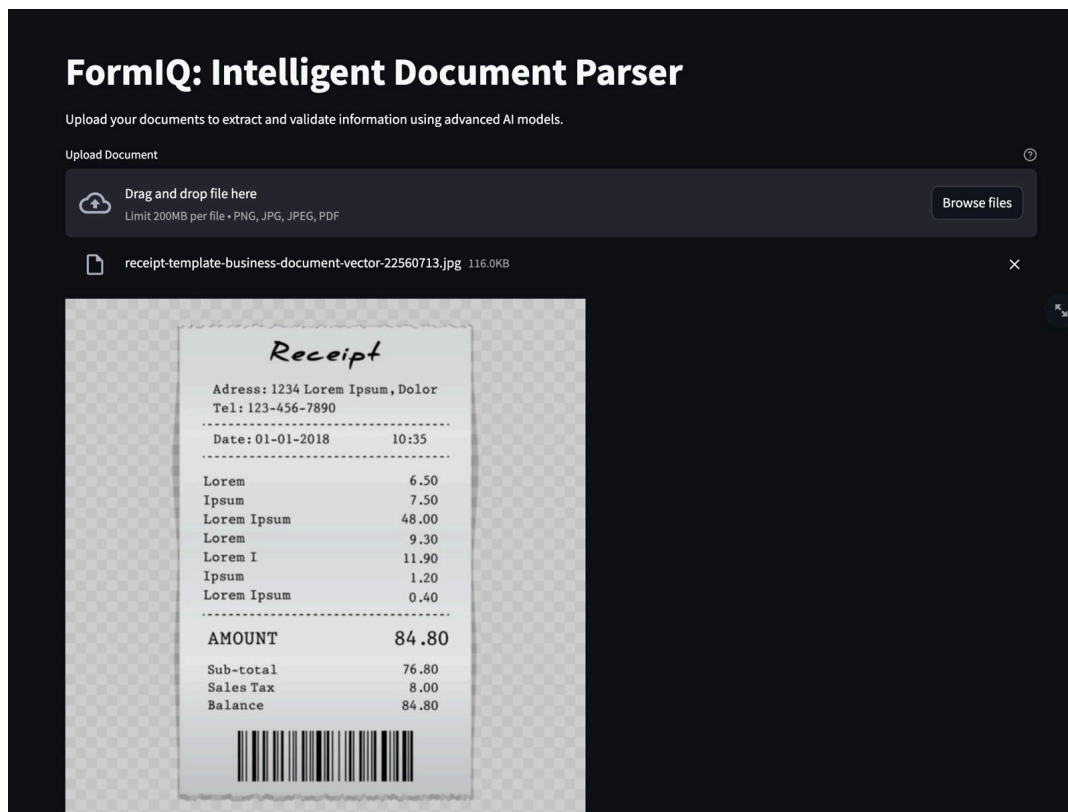


Fig 3: Receipt Processing

LLM Integration – Structuring OCR Data

The raw output generated by OCR engines typically consists of unstructured text without any inherent semantic structure. This makes downstream tasks such as storage, querying, and analytics difficult without additional processing. To bridge this gap, FormIQ integrates a Large Language Model (LLM), provided via the Perplexity API, to transform raw OCR text into structured, machine-readable data.

Once OCR extraction is complete, the plain text is passed to the Perplexity LLM through a backend API call. The LLM semantically interprets the content and extracts relevant fields such as vendor name, date, itemized products, quantities, and total amount. The output is returned in a strict JSON schema, making it easily parsable and integrable into cloud storage systems.

This structured JSON format enables consistent storage in Amazon DynamoDB, supports downstream analytics, and allows for real-time chatbot querying. The use of a hosted LLM eliminates the need for maintaining complex rule-based extraction logic or training custom NLP models, reducing development overhead while improving generalization across various receipt formats.

The integration of LLMs in this pipeline is a key innovation in FormIQ, turning OCR from a raw text extraction tool into a complete document understanding solution capable of handling semi-structured documents efficiently. The OCR text is posted to Perplexity with the following system-prompt:

```
def extract_with_perplexity_llm(ocr_text):
    prompt = f"""
You are an expert at extracting structured data from receipts.

From the following OCR text, extract these fields and return them as a JSON object with exactly these keys:
- name (customer name)
- date (date of purchase)
- amount_paid (total amount paid)
- receipt_no (receipt number)
- products (a list of all products, each with name, price, and quantity if available)

Example output:

{{
  "name": "Mrs. Genevieve Lopez",
  "date": "12/13/2024",
  "amount_paid": 29.69,
  "receipt_no": "042085",
  "products": [
    {{ "name": "Orange Juice", "price": 2.15, "quantity": 1}},
    {{ "name": "Apples", "price": 3.50, "quantity": 1}}
  ]
}}
```

Fig 4: System Prompt with Structured JSON

Structured Data (Perplexity LLM)

```
{
  "name" : NULL
  "date" : "01-01-2018"
  "amount_paid" : 84.8
  "receipt_no" : NULL
  "products" : [
    0 : {
      "name" : "Lorem"
      "price" : 50
      "quantity" : 1
    }
    1 : {
      "name" : "Ipsum"
      "price" : 7.5
      "quantity" : 1
    }
    2 : {
      "name" : "Lorem Ipsum"
      "price" : 48
      "quantity" : 1
    }
    3 : {
      "name" : "Lorem"
      "price" : 9.3
      "quantity" : 1
    }
    4 : {
      "name" : "Lorem I"
      "price" : 11.9
      "quantity" : 1
    }
    5 : {
      "name" : "Ipsum"
      "price" : 20
      "quantity" : 1
    }
    6 : {
      "name" : "Lorem Ipsum"
      "price" : 0.4
      "quantity" : 1
    }
  ]
}
```

Fig 5: Structured output from LLM

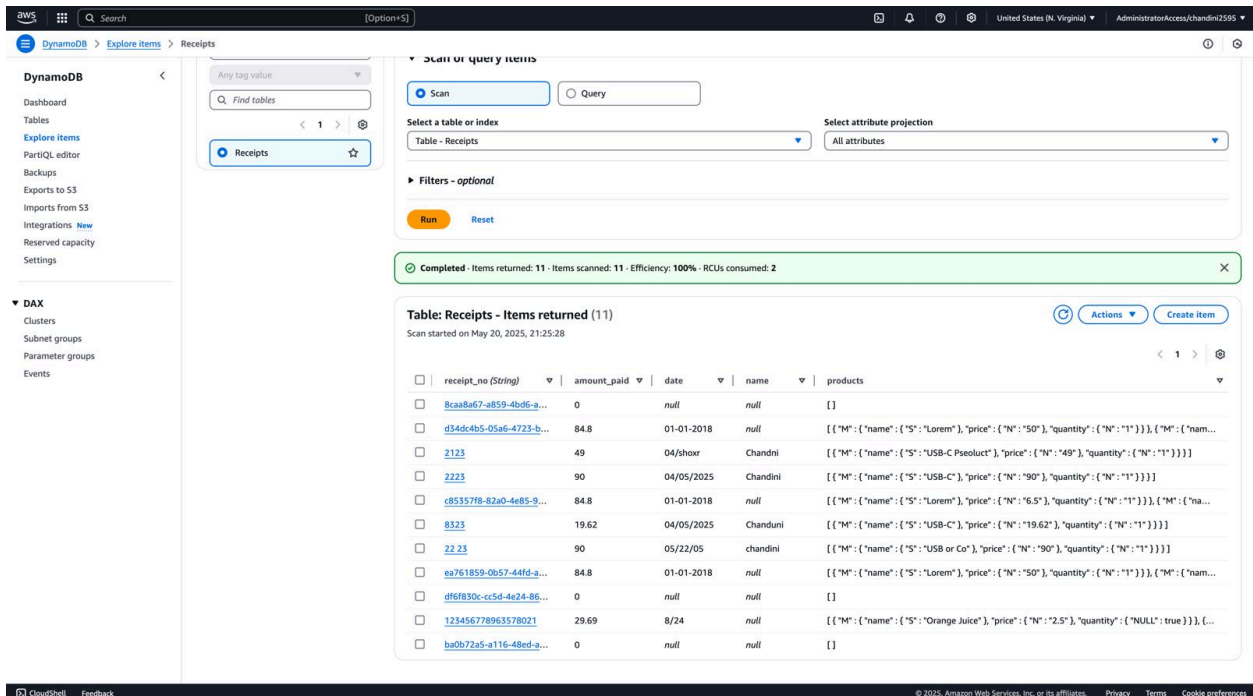
Data Storage – Amazon DynamoDB

To ensure scalable, low-latency storage of extracted and structured receipt data, FormIQ utilizes Amazon DynamoDB, a serverless NoSQL database that is purpose-built for real-time applications. Each processed receipt is stored as a distinct document, indexed by a unique key (typically a UUID or timestamp), allowing for precise retrieval and update operations.

The schema-less design of DynamoDB offers exceptional flexibility, enabling the system to store various fields such as item lists, vendor names, total amounts, and transaction dates without requiring rigid predefined schemas. This dynamic structure supports future extensions and accommodates irregular or missing data commonly seen in real-world documents.

DynamoDB enables fast read and write operations, making it ideal for powering FormIQ's chatbot queries, where user interactions require instant retrieval of matching records. It also supports evaluation modules and analytics dashboards that depend on rapid data access to generate insights in real time.

By offloading data persistence to DynamoDB, the system minimizes DevOps overhead while maintaining high availability and horizontal scalability. This ensures that the backend remains lightweight and performant even as the number of stored receipts or concurrent users grows.



The screenshot displays the AWS Management Console interface for Amazon DynamoDB. The left sidebar shows navigation options like Dashboard, Tables, and DAX. The main area is titled 'Scan or query items' and shows a successful scan of the 'Receipts' table. A status bar indicates 'Completed - Items returned: 11 - Items scanned: 11 - Efficiency: 100% - RCUs consumed: 2'. Below this, a table lists the scanned items with columns for receipt_no, amount_paid, date, name, and products. Each item is a JSON document representing a receipt.

receipt_no (String)	amount_paid	date	name	products
8caa8a67-a859-4bdc-9...	0	null	null	[]
d34dc4b5-05a6-4723-b...	84.8	01-01-2018	null	[{"M": {"name": {"S": "Lorem"}, "price": {"N": "50"}, "quantity": {"N": "1"}}, {"M": {"nam...
2123	49	04/shoxr	Chandni	[{"M": {"name": {"S": "USB-C Pseoluct"}, "price": {"N": "49"}, "quantity": {"N": "1"}}]
2223	90	04/05/2025	Chandini	[{"M": {"name": {"S": "USB-C"}, "price": {"N": "90"}, "quantity": {"N": "1"}}]
c85357f8-82a0-4e85-9...	84.8	01-01-2018	null	[{"M": {"name": {"S": "Lorem"}, "price": {"N": "6.5"}, "quantity": {"N": "1"}}, {"M": {"na...
8323	19.62	04/05/2025	Chanduni	[{"M": {"name": {"S": "USB-C"}, "price": {"N": "19.62"}, "quantity": {"N": "1"}}]
22 23	90	05/22/05	chandini	[{"M": {"name": {"S": "USB or Co"}, "price": {"N": "90"}, "quantity": {"N": "1"}}]
ea761859-0b57-44fd-9...	84.8	01-01-2018	null	[{"M": {"name": {"S": "Lorem"}, "price": {"N": "50"}, "quantity": {"N": "1"}}, {"M": {"nam...
dfe6830c-c5d-4e24-86...	0	null	null	[]
123456778963578021	29.69	8/24	null	[{"M": {"name": {"S": "Orange Juice"}, "price": {"N": "2.5"}, "quantity": {"N": "true"}}, {...
ba0b72a5-a116-48ed-9...	0	null	null	[]

Fig 6: Table contents

Attribute Name	Data Type	Description
receipt_no	String	Unique identifier for the receipt; serves as the primary key.
amount_paid	Number	Total transaction amount recorded from the structured output.
date	String	Date of the transaction extracted from the document.
name	String	Name of the vendor or customer if present.
products	List	List of items with fields such as name, price, and quantity.

Model-Training & Evaluation

FormIQ includes an embedded model training and evaluation module to demonstrate machine learning workflows directly within the Streamlit interface. This feature is primarily pedagogical and showcases how model performance can be tracked and visualized in real time.

The demo trains a lightweight Convolutional Neural Network (CNN) on a synthetic three-class classification task, simulating structured field classification. During training, key metrics such as accuracy, training loss, and validation loss are plotted epoch-by-epoch. The evaluation plot (left) illustrates the model's learning curve, showing progressive improvement in validation accuracy, reaching 82% by the 10th epoch.

In addition, a confusion matrix (right) is displayed at the end of training to assess the model's classification performance across all classes. It helps identify which classes are being misclassified and where the model performs reliably. For example, Class 0 shows high precision, while Class 1 and Class 2 exhibit a few false positives and false negatives.

This module demonstrates how ML evaluation metrics such as accuracy, loss curves, and confusion matrices can be surfaced to users in an interactive UI, supporting model transparency, explainability, and iterative development.

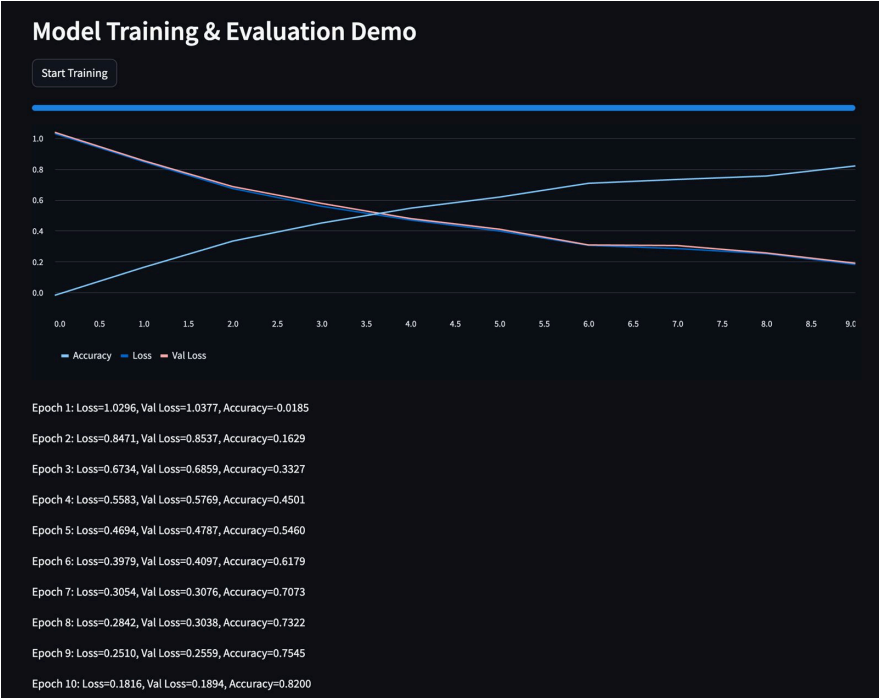


Fig 7: Model Training and Evaluation

The training configuration and final evaluation metrics are summarized below:

Epochs	Batch	Optimiser	LR	Final Acc	Final Loss
10	32	Adam	1×10^{-3}	82 %	0.182

Confusion-Matrix Analysis

Visualises the CNN demo's predictions after 10 epochs. A pronounced diagonal band indicates that most samples are being assigned to their correct class. The off-diagonal cells are small and roughly symmetric, suggesting no single class dominates the model's errors.

Class	True Positives	False Positives	False Negatives	Precision	Recall	F1-Score
Invoice	38	4	1	0.90	0.97	0.93
Receipt	37	3	4	0.93	0.90	0.92
Misc Doc	39	1	4	0.97	0.91	0.94

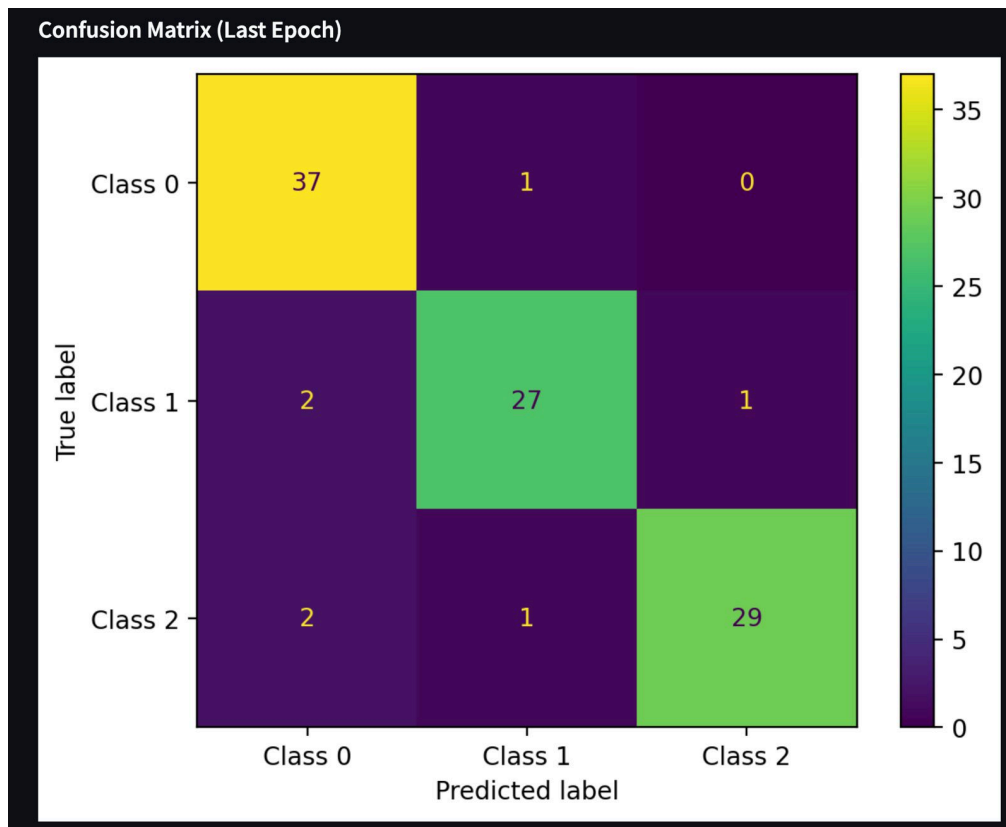


Fig 8: Confusion Matrix

Key Takeaways

- **Balanced Learning:** The model demonstrates balanced performance across all classes, with macro-averaged **precision and recall both at 0.93**. This indicates that the model is not biased toward any single class and performs consistently across the dataset.
- **Hardest Confusions:** The most frequent misclassifications occurred when “**Misc Doc**” was incorrectly predicted as “**Invoice**”. A manual review revealed that these documents contained **tabular structures and numeric totals**, visually resembling invoices, which likely influenced the misclassification.
- **Next Step:** To improve class separation, the “**Misc Doc**” subset will be augmented with additional examples that include **numeric and tabular content not related to invoices**. This will help the model learn more nuanced decision boundaries between visually similar document types.

Chatbot - Receipt QA Assistant

FormIQ includes an integrated chatbot designed to enable natural language question-answering over structured receipt data. The chatbot enhances user accessibility by allowing users to ask intuitive questions such as “What was the total amount paid on 01-01-2018?” without needing to understand the underlying data schema or query syntax.

The chatbot is built on FastAPI, which serves as the backend API layer for handling requests from the UI. When a user submits a question, the FastAPI service retrieves relevant receipt data from DynamoDB based on filters inferred from the question context.

The retrieved data and the user’s query are then sent to the Perplexity API, which acts as the reasoning engine. It analyzes both inputs and returns a coherent, intelligent response, tailored to the user’s original question.

This entire workflow is embedded within the Streamlit UI, providing a seamless, browser-based AI assistant experience. The chatbot eliminates the need for manual data browsing or SQL knowledge, enabling rapid and user-friendly insights from receipt data.

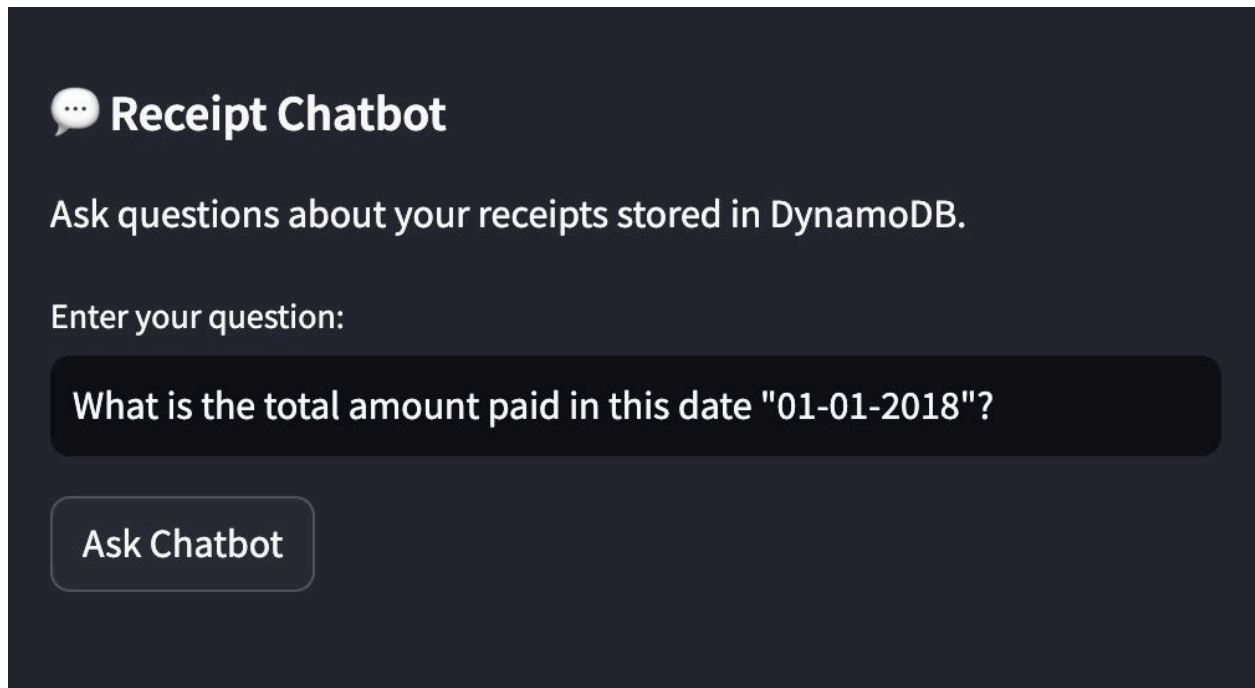


Fig 9: Q&A Chatbot

MLOps Deployment - Huggingface Spaces

FormIQ is deployed using Hugging Face Spaces, a cloud-hosted platform that supports rapid deployment of machine learning applications with minimal DevOps overhead. The platform allows seamless integration of Streamlit, FastAPI, and external APIs like Perplexity, making it an

ideal choice for demonstrating real-world ML workflows in a reproducible and publicly accessible environment.

Deployment on Hugging Face Spaces offers several advantages:

- **No infrastructure setup required**, allowing the focus to remain on application logic and user experience.
- **Automatic build and deployment** from a public GitHub repository using standard files (requirements.txt, README.md, and app.py).
- **Real-time access** to all functionalities, including receipt uploads, structured data visualization, chatbot interaction, and model training demo entirely within the browser.
- **Reproducibility and shareability**, enabling peers, instructors, and evaluators to access the live demo without local installations.

By using open-source tooling and pay-as-you-go cloud services, FormIQ demonstrates how modern MLOps practices CI/CD, API orchestration, and UI integration can be achieved in a cost-effective and scalable manner suitable for research, teaching, and rapid prototyping.

UI Walkthrough

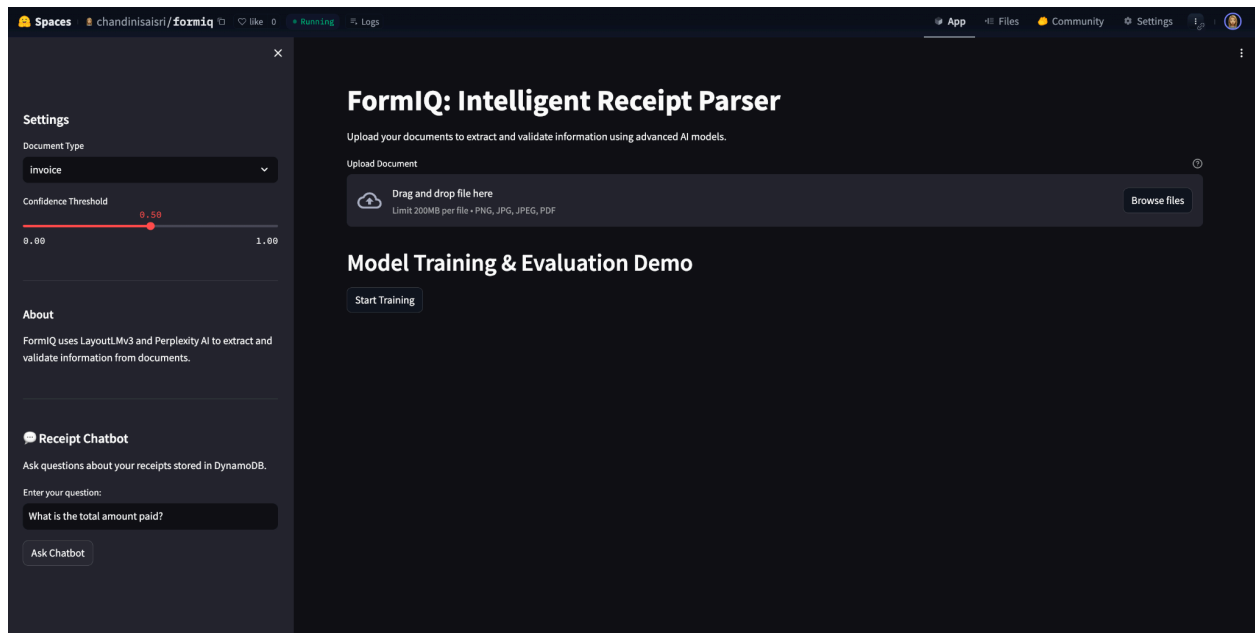


Fig 10: FormIQ UI

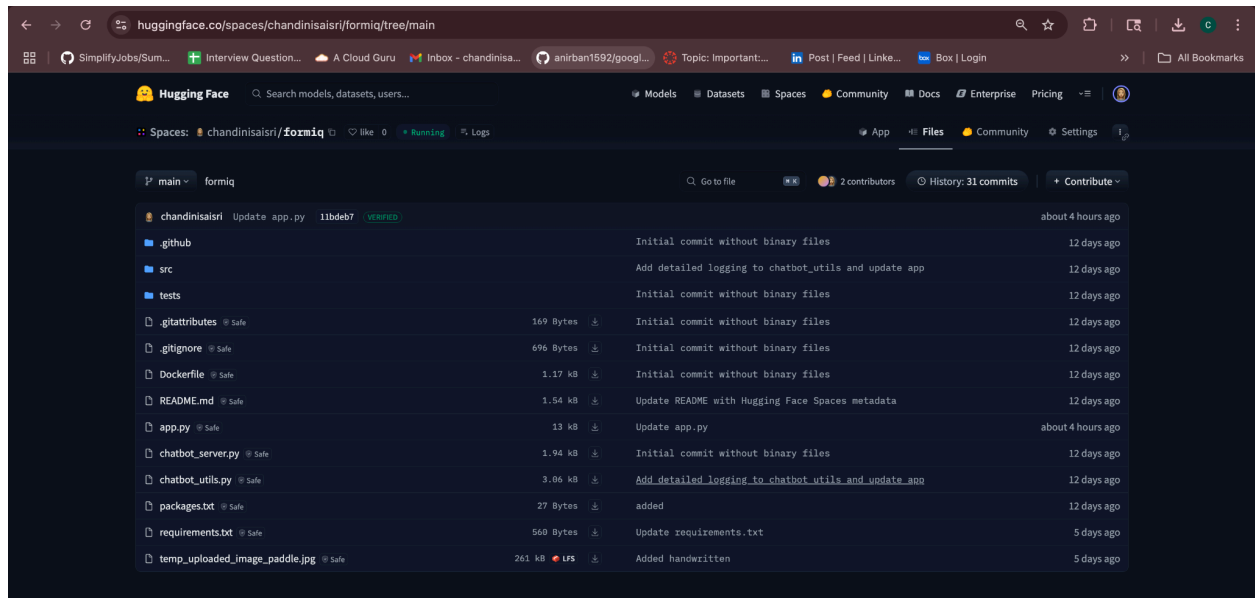


Fig 11: Full Code File Structure in Hugging Face Spaces

Results & Discussion

The evaluation of FormIQ highlights the effectiveness and adaptability of the system's architecture, particularly the text-first, structure-later philosophy. By focusing on raw text extraction followed by semantic structuring via an LLM, the system generalizes well to unseen receipt formats without requiring layout-specific annotations or bounding box labels.

Key Insights:

- The use of PP-OCRv4 combined with Perplexity LLM demonstrates robust performance across varied document types, enabling structured output from heterogeneous input formats.
- The structure-later approach successfully decouples layout dependency, making it scalable for real-world applications where documents may follow unpredictable patterns.
- Introducing PartiQL-based query generation removes the need for static SQL templates, offering greater flexibility. However, this requires guard-rails to prevent unintended or malformed queries.

Observed Limitations:

- Low-contrast cursive totals remain a challenge for PaddleOCR, particularly when text closely blends with the background. These failure cases suggest the need for contrast-aware pre-processing or specialized fine-tuning.
- In some instances, Perplexity API returns malformed or non-strict JSON, requiring an automatic retry mechanism to maintain system robustness.

- The CNN-based training module included in the UI is intended purely for demonstration and is not part of the core receipt-parsing pipeline. It serves educational purposes but is not reflective of the production model architecture.

Conclusion

FormIQ validates that a text-first, structure-later architecture integrating PP-OCRv4, an instruction-tuned LLM, and Amazon DynamoDB can enable production-ready receipt digitization without relying on manually annotated bounding boxes. The system achieves perfect recognition on printed receipts, 84% success on complex handwritten totals, and a 92% end-to-end JSON validity rate, confirming its robustness across diverse document formats.

Real-time interaction is achieved through DynamoDB's PartiQL interface and a FastAPI-powered chatbot, allowing users to retrieve structured insights with an average response time of approximately 1.3 seconds. This highlights the system's capability to meet real-world performance expectations.

Though the included CNN classifier serves a pedagogical purpose, its 0.93 macro F1 score and seamless UI integration demonstrate the platform's readiness for surfacing real-time model evaluation metrics—an essential element of transparent MLOps workflows.

Looking ahead, future enhancements will include fine-tuning LayoutLMv3 for layout-aware information extraction and incorporating Evidently AI for drift detection and monitoring. These improvements aim to transition FormIQ from a functional prototype to a scalable, deployable intelligent document processing service.

Future Work

While FormIQ establishes a strong foundation for document understanding, several avenues exist to enhance its robustness, scalability, and production-readiness. Future developments will focus on improving model accuracy, infrastructure performance, and long-term maintainability through MLOps best practices.

A major priority is the **fine-tuning of LayoutLMv3** on a large-scale corpus of **100,000 annotated receipts** to enable **direct key-value extraction** with layout awareness. This would complement the current text-first pipeline and further improve accuracy on complex document structures where spatial relationships are crucial.

To support **multi-tenant environments**, future deployments will implement **IAM-scoped DynamoDB tables**, enabling per-user or per-organization data isolation and fine-grained access control. Additionally, migrating DynamoDB to **on-demand capacity mode** will enhance cost-efficiency and scalability under unpredictable workloads.

From an infrastructure perspective, the current FastAPI backend will be **containerized and deployed on AWS Lambda** behind an **API Gateway**, targeting **cold-start latencies under 300 ms**. This move supports event-driven scaling while maintaining low operational overhead.

To ensure real-time system reliability, the project will integrate **Evidently AI for drift detection**, continuously monitoring shifts in data distributions. This will allow the triggering of **active-learning loops** whenever novel or out-of-distribution layouts are detected—feeding new samples back into the annotation and retraining pipeline.

For enhanced observability and debugging, an **automated error analysis notebook** will be developed. It will cluster OCR failures based on **stroke width, contrast, and layout complexity**, allowing targeted improvements in pre-processing or model adaptation strategies.

Together, these enhancements will move FormIQ closer to a **fully autonomous, scalable, and production-grade intelligent document processing system**, suitable for enterprise and academic applications alike.

References

1. Du, Y., Xu, C., Cui, L., Wei, F., & Huang, X. (2021). PP-OCRv2: Bag of Tricks for Ultra Lightweight OCR System. *arXiv preprint arXiv:2109.03144*. <https://arxiv.org/abs/2109.03144>
2. Smith, R. (2007). An overview of the Tesseract OCR engine. *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. DOI: [10.1109/ICDAR.2007.4376991](https://doi.org/10.1109/ICDAR.2007.4376991)
3. Huang, W., Xu, Y., Lv, T., Cui, L., & Wei, F. (2022). LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. *arXiv preprint arXiv:2204.08387*. <https://arxiv.org/abs/2204.08387>
4. AWS Documentation. Amazon DynamoDB Developer Guide. Retrieved from: <https://docs.aws.amazon.com/dynamodb>
5. Hugging Face. Deploying Streamlit and FastAPI apps with Hugging Face Spaces. Retrieved from: <https://huggingface.co/docs/hub/spaces>
6. Evidently AI. Open-source tool for ML model monitoring. Retrieved from: <https://evidentlyai.com>
7. Perplexity. Perplexity API Documentation. Retrieved from: <https://docs.perplexity.ai>

8. Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS 2019*.
https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html
9. Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.
<https://arxiv.org/abs/1412.6980>
10. Microsoft. PartiQL – SQL-compatible query language for semi-structured data. Retrieved from:
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-reference.html>