Q.Prescription

AI-Powered Medical Prescription Analysis System

Technical Report

Maria Katibi
Branly DJIME

Course: LLM and Generative AI
Date: January 2026
School : ESILV A5

1. Introduction and Problem Description

1.1 Context

Medical prescriptions are critical documents in healthcare systems, containing essential information about patient diagnoses, prescribed medications, dosages, and treatment plans. The manual digitization of these documents is time-consuming, error-prone, and expensive. This project addresses the challenge of automatically extracting structured data from medical prescription images.

1.2 Problem Statement

Medical prescriptions present unique challenges for automated processing:

1. **Handwritten Content:** Physicians often write prescriptions by hand, making them difficult for traditional OCR systems to process accurately.
2. **Mixed Content Types:** Many prescriptions contain both printed (hospital headers, forms) and handwritten (medication names, dosages) text.
3. **Signature Detection:** Verification of physician signatures is crucial for prescription validity but is particularly challenging for automated systems.
4. **Unstructured Layouts:** Unlike standardized forms, prescriptions vary significantly in layout, format, and language across different healthcare facilities.
5. **Medical Terminology:** Abbreviations, dosage units, and medical terms require domain-specific understanding for accurate extraction.

1.3 Project Objectives

The primary objectives of Q.Prescription are:

- Automatically extract structured data from prescription images (printed, handwritten, or mixed)
- Detect and analyze physician signatures for document authenticity verification
- Convert unstructured prescription content into a standardized JSON schema
- Provide a user-friendly interface for processing and reviewing extracted data
- Optimize processing costs by using LLM capabilities only when necessary

2. Project Evolution: From Invoices to Prescriptions

2.1 Initial Approach: Invoice Analysis

The project initially focused on **invoice document analysis**. The original system, named "Q.Invoice," was designed to extract data from business invoices using OCR and LLM technologies.

2.2 Pivot Decision

After presenting the initial invoice analysis prototype to the course instructor, we received critical feedback that fundamentally changed the project direction:

« *The LLM is essentially useless for invoice processing since they are printed documents. Traditional OCR can handle this task adequately without the need for expensive LLM calls*. »

This feedback highlighted that:

1. **Printed invoices** have high OCR accuracy
2. **Structured layouts** in invoices make regex-based parsing sufficient
3. **LLM costs** were not justified for documents that OCR could handle alone

We pivoted to medical prescription analysis because prescriptions present challenges that genuinely require LLM capabilities:

- Content Type : Mixed (handwritten + printed)
- Layout : variable
- Signature presence : required most of the time

2.4 Retained LLM for JSON Structuring

Despite the feedback about invoices, we discovered through testing that LLM-based JSON structuring significantly improves accuracy compared to OCR parsing, even for high confidence OCR text.

The LLM's ability to understand context, handle variations in formatting, and correctly interpret medical terminology justified its use for JSON structuring, even when OCR confidence was high.

3. System Architecture

3.1 High-Level Architecture

```
+-------------------------------------------------------------+
|                   Q.Prescription System                      |
+-------------------------------------------------------------+
|                                                              |
|  +------------+    +-----------+    +-------------+          |
|  | Streamlit  |--->|   Core    |--->| Extraction  |          |
|  | Frontend   |    |  Config   |    |  Pipeline   |          |
|  | (app.py)   |    |  Module   |    |             |          |
|  +------------+    +-----------+    +-------------+          |
|                                            |                 |
|                                            v                 |
|  +---------------------------------------------------------+ |
|  |                   Extraction Pipeline                  |  |
|  |  +-------+  +----------+  +-------+  +-----------+     |  |
|  |  |  OCR  |->|Confidence|->|  LLM  |->| Signature |     |  |
|  |  |Engine |  |  Check   |  |Extract|  | Detection |     |  |
|  |  +-------+  +----------+  +-------+  +-----------+     |  |
|  +---------------------------------------------------------+ |
|                                                              |
|  +------------+    +-----------+    +-------------+          |
|  |  Pydantic  |<---|   JSON    |--->|   Storage   |          |
|  |   Schema   |    |  Output   |    |   Module    |          |
|  | Validation |    |           |    |             |          |
|  +------------+    +-----------+    +-------------+          |
|                                                              |
+-------------------------------------------------------------+
```

3.2 Component Overview

Component	Technology	Purpose
Frontend	Streamlit 1.32	Web interface for upload, processing, and results display.
OCR Engine	PaddleOCR 2.7.3	Text extraction with confidence scoring

LLM Text Extractor	OpenAI GPT-4o-mini	Structure OCR text into JSON
Vision Extractor	OpenAI GPT-4o	Process handwritten text and detect signatures
Schema Validation	Pydantic 2.6	Ensure extracted data conforms to schema
PDF Processing	PyMuPDF 1.23	Convert PDF prescriptions to images
Image Processing	OpenCV 4.6, Pillow	Image preprocessing and enhancement

3.3 Directory Structure

```
q-prescriptions/
├── app.py                         # Main Streamlit app
├── requirements.txt               # Dependencies
├── .env.template                  # Environment template
├── README.md                      # Documentation
├── QUICKSTART.md                  # Documentation
├── Technical_report.pdf           # Documentation
│
├── core/
│   ├── __init__.py
│   └── config.py                  # Configuration
│   └── processor.py
│
├── extraction/
│   ├── __init__.py
│   ├── ocr.py                     # PaddleOCR with confidence
│   ├── ocr_parser.py              # Regex fallback
│   ├── llm_extractor.py           # LLM text structuring
│   ├── vision_extractor.py        # GPT-4o Vision
│   ├── prescription_processor.py  # Main pipeline
│   ├── pdf_converter.py           # PDF support
│   └── schema.py                  # Pydantic models
│
└── prescription_dataset/          # Your test images
    ├── handwriting/
    ├── mixed/
    └── printed/
```

4. Processing Pipeline

4.1 Pipeline Overview

The Q.Prescription system implements an **intelligent three-stage pipeline** that adapts its processing strategy based on OCR confidence levels:

```
                    +------------------+
                    |   Input Image    |
                    +--------+---------+
                             |
                             v
              +------------------------------+
              |   STAGE 1: OCR EXTRACTION    |
              |   - PaddleOCR text extraction|
              |   - Multi-factor confidence  |
              |     calculation              |
              +-------------+----------------+
                            |
                            v
              +------------------------------+
              |   CONFIDENCE THRESHOLD       |
              |         (60%)                |
              +-------------+----------------+
                            |
           +----------------+----------------+
           |                                 |
    >=60% (High)                        JSON   |        | - Image -> JSON      |
| - GPT-4o-mini        |        | - GPT-4o with vision |
| - Cheaper & faster   |        | - Handles handwriting|
+----------+-----------+        +----------+-----------+
           |                                 |
           +-----------------+---------------+
                             |
                             v
              +------------------------------+
              | STAGE 3: SIGNATURE DETECTION |
              | (Always uses LLM Vision)     |
              +-------------+----------------+
                            |
                            v
              +------------------------------+
              |   Structured JSON Output     |
              |   (ExtractedPrescription)    |
              +------------------------------+
```

4.2 Stage 1: OCR Extraction

The first stage uses **PaddleOCR** to extract text from the prescription image.

4.2.1 Image Preprocessing

```
def preprocess_image(self, image_path: str) -> np.ndarray:
    # Read image
    img = cv2.imread(image_path)

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply denoising
    denoised = cv2.fastNlMeansDenoising(gray)

    # Apply adaptive thresholding
    thresh = cv2.adaptiveThreshold(
        denoised, 255,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY, 11, 2
    )
    return thresh
```

4.2.2 Multi-Factor Confidence Calculation

A key innovation is our **multi-factor confidence scoring** that goes beyond simple OCR confidence:

```
def calculate_confidence(self, results: list, extracted_text: str) -> float:
    # Factor 1: Base OCR confidence (50% weight)
    base_confidence = average(per_line_confidences)

    # Factor 2: Detection density (25% weight)
    density_score = min(num_lines / EXPECTED_MIN_LINES, 1.0)

    # Factor 3: Text length (25% weight)
    length_score = min(text_length / EXPECTED_MIN_CHARS, 1.0)

    # Combined score
    combined_confidence = (
        base_confidence * 0.5 +
        density_score * 0.25 +
        length_score * 0.25
    )
    return combined_confidence
```

Expected minimums for prescription documents:

- `EXPECTED_MIN_LINES = 8` (prescriptions typically have at least 8 lines)
- `EXPECTED_MIN_CHARS = 150` (prescriptions typically have at least 150 characters)

After the instructor remark on confidence being not enough (can detect one word accurately and have a high confidence while missing most of the text). We decided to implement multi-factor confidence scoring which combines three elements instead of solely on OCR's raw confidence score :

1- Base OCR confidence (50%) : This is what PaddleOCR reports - how confident it is that reads each character correctly. However as explained before this can be misleading : OCR might report high confidence on a blurry image where it only detected a few clear words, or report low confidence on a dense prescription where it actually captured everything.
2- Text density score (25%) : Measures how many lines of text were detected. A typical prescription has 8+ lines(header, patient info, medications, signature area). If OCR only finds 2-3 lines, something is wrong - likely handwritten content it couldn't read. This catches cases where OCR is 'confidently wrong' about the little it found.
3- Content length score (25%) : Measures total characters extracted. A real prescription typically has 150 + characters. If OCR returns very little text despites high per-character confidence, the image likely contains content that OCR completely missed most probably hand-written.

By penalizing for missing content (low density/length), the combined score drops below the threshold correctly triggering the LLM vision to analyze the handwritten portions. This prevents the system from confidently producing incomplete JSON when handwriting is present.

4.3 Stage 2: LLM Processing

Based on the OCR confidence score, the system routes to one of two processing paths:

4.3.1 High Confidence Path (>=60%): LLM Text Structuring

When OCR confidence is high (>=60%), indicating primarily printed text:

```
if ocr_confidence >= HANDWRITING_CONFIDENCE_THRESHOLD:
    # Use LLM to structure OCR text into JSON (no vision needed)
    prescription, llm_time = self.llm_text.extract(ocr_text, document_type="prescription")
    metadata["extraction_method"] = "ocr_plus_llm_text"
```

This path uses **GPT-4o-mini** with a carefully crafted prompt that includes:

- The complete OCR text
- Expected JSON schema
- Date formatting rules (YYYY-MM-DD)
- Instructions for extracting all medical fields

4.3.2 Low Confidence Path (When OCR confidence is low)

```
if ocr_confidence < HANDWRITING_CONFIDENCE_THRESHOLD:
    # Use vision extractor with OCR text as supplementary context
    prescription, vision_time = self.vision.extract_from_image(
        image_path,
        ocr_text=ocr_text,
        ocr_confidence=ocr_confidence
    )
    metadata["extraction_method"] = "ocr_plus_llm_vision"
```

This path uses GPT-4o with vision capabilities to:

- Directly analyze the prescription image
- Read handwritten text that OCR failed to capture
- Use OCR text as supplementary context
- Identify which content is handwritten vs. printed

4.4 Stage 3: Signature Detection

Signature detection always uses LLM Vision regardless of OCR confidence, as signatures are inherently graphical elements that OCR cannot process:

```
if VISION_ALWAYS_FOR_SIGNATURES and self.vision:
    signature_info, sig_time = self.vision.analyze_signature_only(image_path)
    prescription.doctor_signature = signature_info
```

5. Technical Implementation

5.1 OCR Engine (PaddleOCR)

Configuration:

```
self.engine = PaddleOCR(
    lang=Config.OCR_LANGUAGE,
    use_gpu=Config.OCR_USE_GPU,     # False (CPU mode)
    show_log=False,
    use_angle_cls=True,             # Enable text orientation detection
    det_db_thresh=0.3,              # Detection threshold
    det_db_box_thresh=0.5,          # Box threshold
    rec_batch_num=6                 # Batch size for recognition
)
```

5.2 LLM Text Extractor

Multi-Provider Support:

```
class LLMExtractor:
    def __init__(self, provider: str = "openai"):
        if provider == "openai":
            self.client = OpenAI(api_key=Config.OPENAI_API_KEY)
            self.model = Config.DEFAULT_LLM_MODEL  # "gpt-4o-mini"
        elif provider == "anthropic":
            self.client = Anthropic(api_key=Config.ANTHROPIC_API_KEY)
            self.model = "claude-3-5-sonnet-20241022"
```

5.3 Vision Extractor

GPT-4o Vision Integration:

```
response = self.client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {
            "role": "user",
            "content": [
                {"type": "text", "text": prompt},
                {
                    "type": "image_url",
                    "image_url": {
                        "url": f"data:{media_type};base64,{base64_image}",
                        "detail": "high"  # High detail for better text reading
                    }
                }
            ]
        }
    ],
    max_tokens=4096,
    temperature=0.1  # Low temperature for consistent extraction
)
```

6. Data Schema and Models

6.1 Core Schema (Pydantic Models)

ExtractedPrescription (Main Model):

```
class ExtractedPrescription(BaseModel):
    document_type: str = "prescription"
    prescription_type: Optional[PrescriptionType]  # handwritten/printed/mixed/digital
    prescription_number: Optional[str]
    issue_date: Optional[str]

    # People involved
    patient: PatientInfo
    doctor: DoctorInfo
    hospital: HospitalInfo

    # Medical content
    diagnosis: Optional[str]
    medications: List[MedicationItem]

    # Signatures
    doctor_signature: Optional[SignatureInfo]

    # Extraction metadata
    extraction_method: Optional[str]  # 'ocr_plus_llm_text', 'ocr_plus_llm_vision',
'ocr_only_regex'
    confidence_score: Optional[float]
    ocr_confidence: Optional[float]
    llm_enhanced: bool = False
```

6.2 Supporting Models

MedicationItem:

```
class MedicationItem(BaseModel):
    name: str
    dosage: Optional[str]        # e.g., "400mg"
    quantity: Optional[str]      # e.g., "30 tablets"
    frequency: Optional[str]     # e.g., "twice daily"
    duration: Optional[str]      # e.g., "7 days"
    instructions: Optional[str]
    is_handwritten: Optional[bool]
```

SignatureInfo:

```
class SignatureInfo(BaseModel):
    is_present: bool = False
    signer_name: Optional[str]
    signer_title: Optional[str]
    location: Optional[str]      # e.g., "bottom right"
    is_legible: Optional[bool]
    confidence: Optional[float]  # 0.0 to 1.0
```

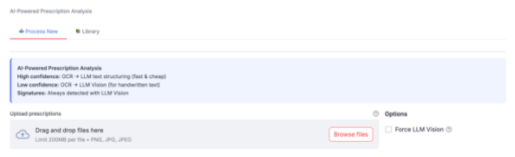
8. User Interface

8.1 Streamlit Application

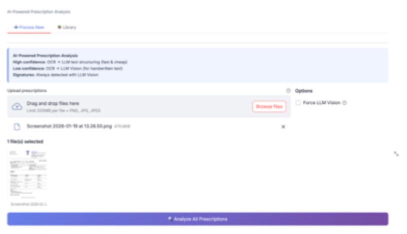
The web interface provides two main tabs:

Process New Tab:

- File upload (supports multiple files, drag-and-drop)
- Force LLM Vision option (This botton will force the LLM vision usage)
- Progress tracking during batch processing
- Preview of uploaded images



AI-Powered Prescription Analysis
High confidence: OCR → LLM text structuring (fast & cheap)
Low confidence: OCR → LLM Vision (for handwritten text)
Signatures: Always detected with LLM Vision

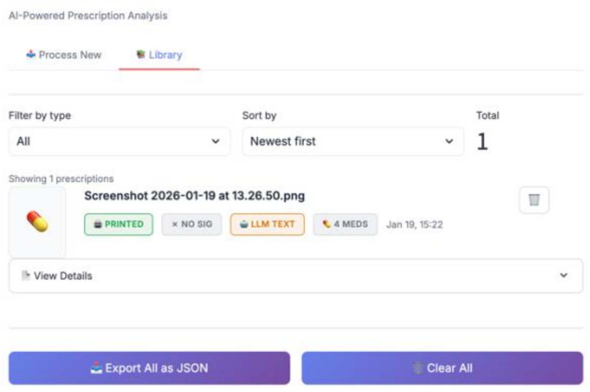


AI-Powered Prescription Analysis
High confidence: OCR → LLM text structuring (fast & cheap)
Low confidence: OCR → LLM Vision (for handwritten text)
Signatures: Always detected with LLM Vision

Screenshot 2026-01-19 at 13.26.50.png
1 file(s) selected

Library Tab:

- **List of all processed prescriptions**
- **Filter by prescription type**
- **Sort by date, name, or medication count**
- **Expandable details for each prescription**



Filter by type: All

Sort by: Newest first

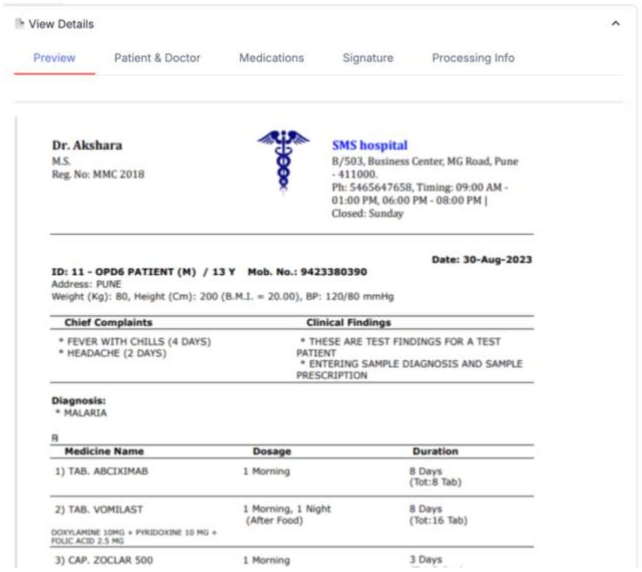
Total: 1

Showing 1 prescriptions

Screenshot 2026-01-19 at 13.26.50.png

PRINTED · × NO SIG · LLM TEXT · 4 MEDS · Jan 19, 15:22

Here is what we can see in the view details section :



Dr. Akshara
M.S.
Reg. No: MMC 2018

SMS hospital
B/503, Business Center, MG Road, Pune - 411000.
Ph: 5465647658, Timing: 09:00 AM - 01:00 PM, 06:00 PM - 08:00 PM | Closed: Sunday

Date: 30-Aug-2023

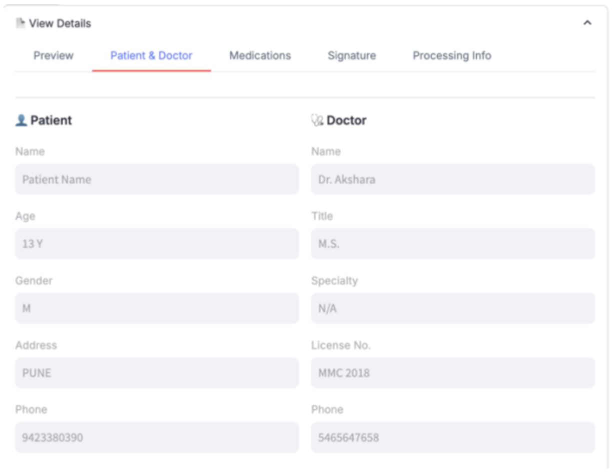
ID: 11 - OPD6 PATIENT (M) / 13 Y Mob. No.: 9423380390
Address: PUNE
Weight (Kg): 80, Height (Cm): 200 (B.M.I. = 20.00), BP: 120/80 mmHg

Chief Complaints	Clinical Findings
* FEVER WITH CHILLS (4 DAYS) * HEADACHE (2 DAYS)	* THESE ARE TEST FINDINGS FOR A TEST PATIENT * ENTERING SAMPLE DIAGNOSIS AND SAMPLE PRESCRIPTION

Diagnosis:
* MALARIA

R

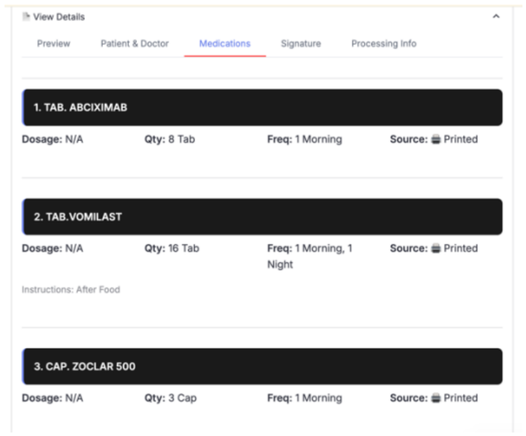
Medicine Name	Dosage	Duration
1) TAB. ABCIXIMAB	1 Morning	8 Days (Tot:8 Tab)
2) TAB. VOMILAST DOXYLAMINE 10MG + PYRIDOXINE 10 MG + FOLIC ACID 2.5 MG	1 Morning, 1 Night (After Food)	8 Days (Tot:16 Tab)
3) CAP. ZOCLAR 500	1 Morning	3 Days



View Details

Preview | Patient & Doctor | Medications | Signature | Processing Info

Patient	**Doctor**
Name: Patient Name	Name: Dr. Akshara
Age: 13 Y	Title: M.S.
Gender: M	Specialty: N/A
Address: PUNE	License No.: MMC 2018
Phone: 9423380390	Phone: 5465647658



View Details

Preview | Patient & Doctor | Medications | Signature | Processing Info

1. TAB. ABCIXIMAB

Dosage: N/A **Qty:** 8 Tab **Freq:** 1 Morning **Source:** Printed

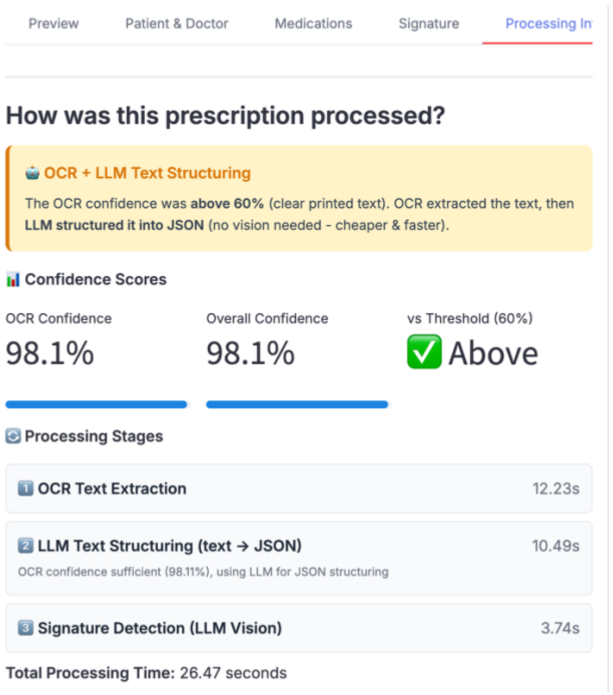
2. TAB.VOMILAST

Dosage: N/A **Qty:** 16 Tab **Freq:** 1 Morning, 1 Night **Source:** Printed

Instructions: After Food

3. CAP. ZOCLAR 500

Dosage: N/A **Qty:** 3 Cap **Freq:** 1 Morning **Source:** Printed



Preview | Patient & Doctor | Medications | Signature | Processing In

How was this prescription processed?

OCR + LLM Text Structuring

The OCR confidence was **above 60%** (clear printed text). OCR extracted the text, then **LLM structured it into JSON** (no vision needed - cheaper & faster).

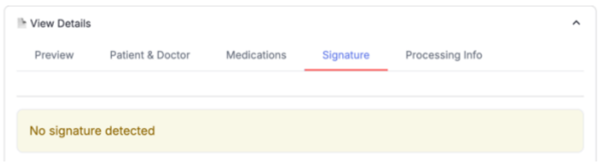
Confidence Scores

OCR Confidence	Overall Confidence	vs Threshold (60%)
98.1%	98.1%	✅ Above

Processing Stages

1️⃣ **OCR Text Extraction**	12.23s
2️⃣ **LLM Text Structuring (text → JSON)** OCR confidence sufficient (98.11%), using LLM for JSON structuring	10.49s
3️⃣ **Signature Detection (LLM Vision)**	3.74s

Total Processing Time: 26.47 seconds

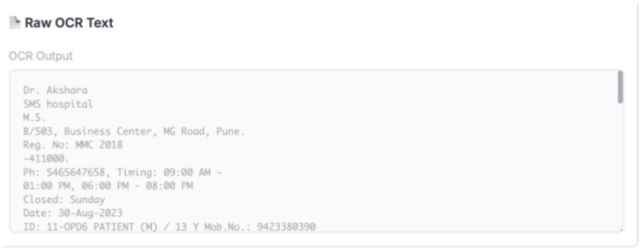


View Details

Preview | Patient & Doctor | Medications | Signature | Processing Info

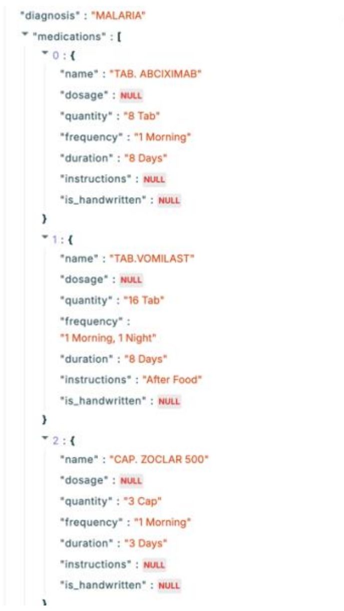
No signature detected

This is the raw OCR text extracted

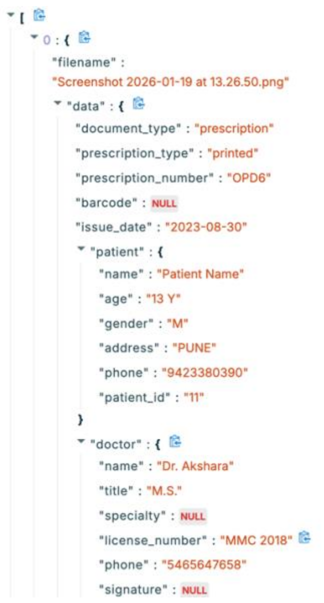


```
Dr. Akshara
SMS hospital
M.S.
B/503, Business Center, MG Road, Pune.
Reg. No: MMC 2018
-411000.
Ph: 5465647658, Timing: 09:00 AM -
01:00 PM, 06:00 PM - 08:00 PM
Closed: Sunday
Date: 30-Aug-2023
ID: 11-OPD6 PATIENT (M) / 13 Y Mob.No.: 9423380390
```

And finally the JSON schema extracted



```
"diagnosis" : "MALARIA"
"medications" : [
  0 : {
    "name" : "TAB. ABCIXIMAB"
    "dosage" : NULL
    "quantity" : "8 Tab"
    "frequency" : "1 Morning"
    "duration" : "8 Days"
    "instructions" : NULL
    "is_handwritten" : NULL
  }
  1 : {
    "name" : "TAB.VOMILAST"
    "dosage" : NULL
    "quantity" : "16 Tab"
    "frequency" :
    "1 Morning, 1 Night"
    "duration" : "8 Days"
    "instructions" : "After Food"
    "is_handwritten" : NULL
  }
  2 : {
    "name" : "CAP. ZOCLAR 500"
    "dosage" : NULL
    "quantity" : "3 Cap"
    "frequency" : "1 Morning"
    "duration" : "3 Days"
    "instructions" : NULL
    "is_handwritten" : NULL
  }
```



```
[
  0 : {
    "filename" :
    "Screenshot 2026-01-19 at 13.26.50.png"
    "data" : {
      "document_type" : "prescription"
      "prescription_type" : "printed"
      "prescription_number" : "OPD6"
      "barcode" : NULL
      "issue_date" : "2023-08-30"
      "patient" : {
        "name" : "Patient Name"
        "age" : "13 Y"
        "gender" : "M"
        "address" : "PUNE"
        "phone" : "9423380390"
        "patient_id" : "11"
      }
      "doctor" : {
        "name" : "Dr. Akshara"
        "title" : "M.S."
        "specialty" : NULL
        "license_number" : "MMC 2018"
        "phone" : "5465647658"
        "signature" : NULL
```

8.2 Processing Information Display

The UI provides detailed processing information including:

- **Extraction method used (LLM Vision, LLM Text, or Regex)**
- **OCR confidence score with progress bar**
- **Processing stages with timing information**
- **Total processing time**
- **Raw OCR text output**

9. Challenges and Solutions

9.1 Challenge: Low OCR Confidence for Handwritten Text

Problem: PaddleOCR confidence scores were unreliable for handwritten text, sometimes reporting high confidence for incorrectly recognized characters.

Solution: Implemented multi-factor confidence scoring that considers base OCR confidence, detection density, and text length.

9.2 Challenge: Variable Prescription Formats

Problem: Prescriptions from different hospitals have vastly different layouts, making field extraction inconsistent.

Solution: Used LLM's contextual understanding to identify fields regardless of position, with comprehensive schema examples in prompts.

9.3 Challenge: Balancing Cost and Accuracy

Problem: LLM Vision calls are expensive, making it impractical for all documents.

Solution: Implemented confidence-based routing, using cheaper LLM text structuring for high-confidence OCR and reserving vision calls for low-confidence cases.

9.4 Challenge: Date Format Variations

Problem: Prescriptions contained dates in various formats (DD-MM-YYYY, DD/Mon/YYYY, etc.)

Solution: Explicit date formatting instructions in LLM prompts with standardized YYYY-MM-DD output.

11. Conclusion

Q.Prescription demonstrates a practical approach to medical document processing that intelligently combines OCR and LLM technologies. Key achievements include:

1. Adaptive Processing Pipeline: The confidence-based routing optimizes cost while maintaining accuracy, using expensive LLM vision only when necessary.
2. Validated LLM Value: Our testing confirmed that LLM-based JSON structuring significantly outperforms regex-only parsing, even for high-confidence OCR text, justifying its inclusion in the pipeline.
3. Signature Detection: The dedicated vision-based signature detection addresses a critical requirement for prescription validation that traditional OCR cannot handle.
4. Project Pivot Success: The decision to shift from invoice to prescription analysis resulted in a more challenging and impactful project that better leverages LLM capabilities.