Design & Documentation

Project title : Automated Document Extraction System.

1. Tech Stack:

Programming Language: Python 3.12.

OCR Engine: Tesseract OCR v5.

Libraries: pytesseract, pdf2image, Pillow, re, json.
PDF Dependency: Poppler for Windows.

2. System Architecture :

Raw PDF/Image
↓
PDF → Image Conversion
↓
OCR (Text Extraction)
↓
Classification (Detects document type)
↓
Field Extraction
↓
Structured JSON Output

3. System Design / Workflow:

- The system follows a sequential pipeline to process documents and generate structured output:

1. **Input Document :**
The workflow begins when the user provides a document to the system.
Supported formats:

- PDF files
- Image files (JPG, PNG, JPEG)

- Input document may contain invoice or packing list information in scanned or digital format.

2. **OCR Processing :**

- Once the document is received, Optical Character Recognition (OCR) is applied.

Tools Used:

- Tesseract OCR
- pytesseract (Python wrapper)
- pdf2image + Poppler (for PDF conversion).

Description:

- If the input is a PDF, it is first converted into images using *pdf2image*.
- Each image/page is processed by Tesseract OCR.
- If the input is an image, it is directly passed to Tesseract**.**

3. **Text Extraction :**

- The OCR engine outputs machine-readable text.
- Extracted text may contain:
    1. Vendor details
    2. Invoice data
    3. Item descriptions
    4. Addresses

4. **Document Classification** :

- The extracted text is analyzed to determine document type.
- **Method Used:** Keyword-based classification**.**
- If text contains “invoice” → Classified as Invoice.
- If text contains “packing list” or “ship to” → Packing List.
- Document category is identified.

5.**Field Extraction :**

- Based on the detected document type, relevant data fields are extracted.

- For Invoice Documents: It extracts the,
    1. Vendor Name
    2. Invoice Number
    3. Invoice Date
    4. Line Items.
- For packing Documents: It extracts the,
    1. Order Number
    2. Ship To Address
    3. Line Items
- **Technique used:** Regex.

6. **JSON Output :**

- The extracted structured data is stored in JSON format.

Files Generated:

- invoice_output.json
- packing_output.json.

4.Implementation Details :

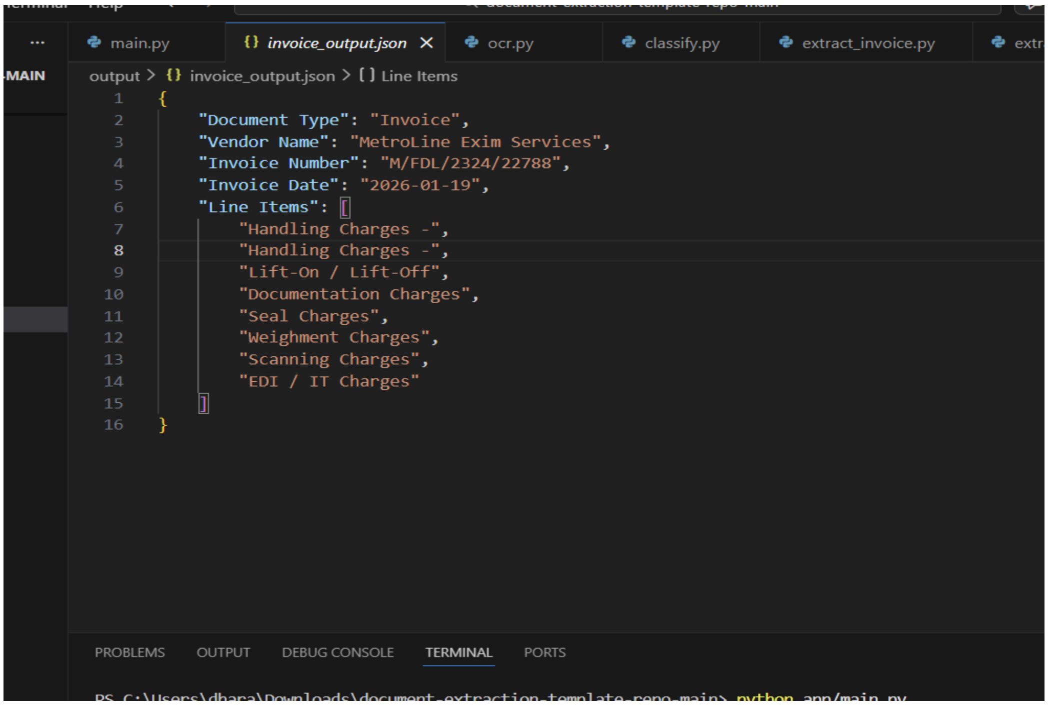
- The system was implemented using Python with a modular programming approach. Each functional component such as OCR, classification, extraction, and output handling was developed as a separate module.
- OCR functionality was implemented using Tesseract OCR via the pytesseract library. PDF documents were converted into images using pdf2image and Poppler, while image files were directly processed using Pillow.
- Document classification was performed using rule-based keyword detection. The extracted text was converted to lowercase and matched against predefined keywords such as “invoice”, “packing list”, and “ship to”.
- Key document fields were extracted using Regular Expressions (Regex). Multiple regex patterns were designed to handle variations in OCR text, formatting differences, and label inconsistencies.
- Regex patterns were used to identify invoice numbers, dates, order numbers, and line items. Flexible patterns were implemented to tolerate OCR errors and inconsistent spacing.

- The final structured data was exported as JSON files using Python's json module. Output files were stored in an automatically generated output directory.
- Core function:

```
def process_document(file_path):
    text = read_document(file_path)
    doc_type = classify_document(text)
    if doc_type == "Invoice":
        data = extract_invoice_fields(text)
        save_json(data, "invoice_output.json")
    elif doc_type == "Packing List":
        data = extract_packing_fields(text)
        save_json(data, "packing_output.json")
```

5. Results / Output :

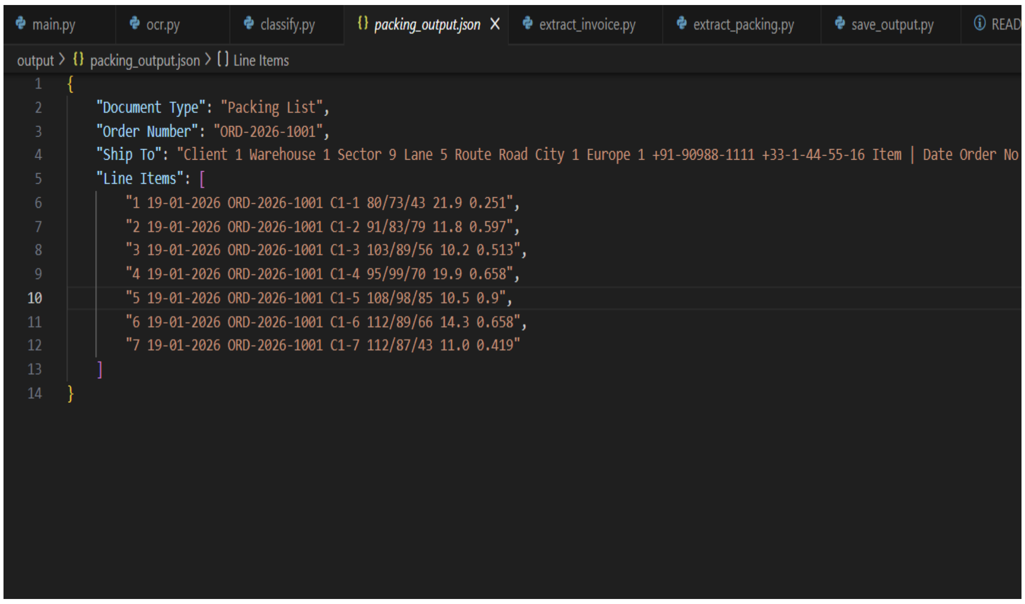
- **Output for Invoice :**



```
{
    "Document Type": "Invoice",
    "Vendor Name": "MetroLine Exim Services",
    "Invoice Number": "M/FDL/2324/22788",
    "Invoice Date": "2026-01-19",
    "Line Items": [
        "Handling Charges -",
        "Handling Charges -",
        "Lift-On / Lift-Off",
        "Documentation Charges",
        "Seal Charges",
        "Weighment Charges",
        "Scanning Charges",
        "EDI / IT Charges"
    ]
}
```

```
PS C:\Users\dhara\Downloads\document-extraction-template-repo-main> python app/main.py
```

- **Output for packing:**



```json
{
    "Document Type": "Packing List",
    "Order Number": "ORD-2026-1001",
    "Ship To": "Client 1 Warehouse 1 Sector 9 Lane 5 Route Road City 1 Europe 1 +91-90988-1111 +33-1-44-55-16 Item | Date Order No
    "Line Items": [
        "1 19-01-2026 ORD-2026-1001 C1-1 80/73/43 21.9 0.251",
        "2 19-01-2026 ORD-2026-1001 C1-2 91/83/79 11.8 0.597",
        "3 19-01-2026 ORD-2026-1001 C1-3 103/89/56 10.2 0.513",
        "4 19-01-2026 ORD-2026-1001 C1-4 95/99/70 19.9 0.658",
        "5 19-01-2026 ORD-2026-1001 C1-5 108/98/85 10.5 0.9",
        "6 19-01-2026 ORD-2026-1001 C1-6 112/89/66 14.3 0.658",
        "7 19-01-2026 ORD-2026-1001 C1-7 112/87/43 11.0 0.419"
    ]
}
```

6.Exceptions:

- Error handling was implemented using try–except blocks to manage file errors, OCR failures, PDF conversion issues, and missing data. Default values such as “Not Found” were used to maintain system stability.

7. Conclusion :

- This project successfully developed an automated document processing system capable of extracting structured information from invoices and packing lists. By integrating Tesseract OCR with rule-based classification and regex-driven field extraction, the system converts unstructured PDF and image documents into machine-readable JSON output.
- The implementation demonstrates how OCR and text processing techniques can significantly reduce manual effort, minimize errors, and improve efficiency in document handling. The modular design ensures flexibility, maintainability, and ease of future enhancements. Overall, the project highlights the practical application of OCR, pattern matching, and automation in real-world business workflows.