📄 PDF Text & Table Extraction Automation Tool

◆ Project Overview

This project is a Python-based automation tool designed to extract structured and unstructured data from text-based PDF documents. The system processes invoices, reports, and business documents to extract:

- Full document text
- Structured tables
- Key invoice fields (Invoice Number, Date, Total, PO Number)
- Keyword search results
- Batch processing summaries

The tool is designed for scalability, automation workflows, and document processing pipelines.

🎯 Project Objectives

- Automate manual PDF data extraction
- Enable structured data export (Excel/CSV)
- Improve processing speed for bulk documents
- Provide reusable automation for invoice and report parsing
- Create a production-ready, modular extraction system

⚙ Technical Implementation

◆ Technologies Used

- Python 3
- pdfplumber (PDF parsing)
- pandas (data processing)
- openpyxl (Excel export)
- Regex (pattern-based extraction)
- Logging module (production-level monitoring)

🏗 System Architecture

The system follows an object-oriented design with:

- Modular extraction methods
- Error handling for corrupted PDFs
- Caching mechanism for performance
- Structured output management
- Summary reporting

Processing Flow:

1. Load PDF(s)
2. Extract text page-by-page
3. Extract tables and clean data
4. Apply regex-based field detection
5. Export:
    - TXT (full text)
    - XLSX (tables)
    - CSV (summary report)

📊 Features

✓ Batch PDF processing
✓ Automatic folder management
✓ Table detection & cleaning
✓ Invoice field extraction
✓ Keyword search functionality
✓ Excel export with formatted sheets
✓ Timestamped output files
✓ Summary reporting for multiple PDFs

📁 Output Example

For each processed PDF, the system generates:

- `document_text_TIMESTAMP.txt`
- `document_tables_TIMESTAMP.xlsx`
- `summary_TIMESTAMP.csv`

All outputs are organized inside an `output_data/` directory.

🚀 Real-World Use Cases

- Invoice processing automation
- Financial data extraction
- Bulk report parsing
- Data engineering preprocessing
- Accounting automation
- Procurement document analysis
- Research document processing

📈 Performance & Scalability

- Optimized text caching to prevent re-processing
- Efficient regex searching
- Structured DataFrame cleaning
- Modular design for easy feature expansion
- Ready for OCR extension (scanned PDFs)

🔮 Future Enhancements

- OCR integration (pytesseract)
- REST API deployment (FastAPI)
- Docker containerization
- Cloud deployment (AWS/GCP/Azure)
- Automated workflow integration

📄 Demo

🔗 **Live Demo / Video Walkthrough:** https://youtu.be/6bwfJ72A_CM

🔗 **GitHub Repository:**

https://github.com/DrIssah

👤 Developer Summary

This project demonstrates strong skills in:

- Automation engineering
- Data extraction pipelines
- Python development
- Structured data processing
- Real-world business document handling
- Clean and scalable software design

It reflects production-level thinking suitable for freelance automation projects and enterprise document processing systems.