TDGPT - PROJECT DOCUMENTATION

# Project Introduction

This AI-powered Document Parser automatically extracts, summarizes, and organizes content from various document types using Large Language Models (LLMs) and traditional document processing libraries.

Methods Overview

Method 1: Custom Parsing (Without Unstructured)

Method 2: Using Unstructured Partitioning

Method 1: Custom Parsing (Without Unstructured)

Project Objectives

* Extract text, tables, and images from PDFs, Word docs, and PowerPoint files.
* Perform chunk-wise summarization using LLMs.
* Generate structured JSON outputs.
* Store extracted images in local folders.
* Process native and scanned documents with OCR support.

Technologies Used

* Python: Core programming language
* pdfplumber: Text and table extraction from PDFs
* PyMuPDF (fitz): Image extraction from PDFs
* pytesseract: OCR for scanned PDFs
* python-docx: Word document parsing
* python-pptx: PowerPoint content extraction
* Pillow (PIL): Image manipulation
* Tesseract OCR: Optical character recognition
* Groq API / OpenAI / Llama3: LLMs for summarization

# LLM Comparison

GPT-3.5/4 - OpenAI: High accuracy and broad support.

Llama3-70B - Groq: Fast inference and cost-effective.

# Supported File Formats and Tools

PDF: pdfplumber, fitz, pytesseract

DOCX: python-docx

PPTX: python-pptx

Images: Pillow

# Chunking Logic

Type: Word-based chunking

Chunk Size: CHUNK\_SIZE = 300 words

Why: Optimized for LLM context window & summarization granularity

Performance (with LLM):

|  |  |  |
| --- | --- | --- |
| File Size | Pages | Time Taken |
| Small | 30–50 | ~32–50sec |
| Medium | |  | | --- | |  |  |  | | --- | | 300–600 | | |  | | --- | |  |  |  | | --- | | ~70–160 sec | |
| Large | 600+ | ~180–240 sec |

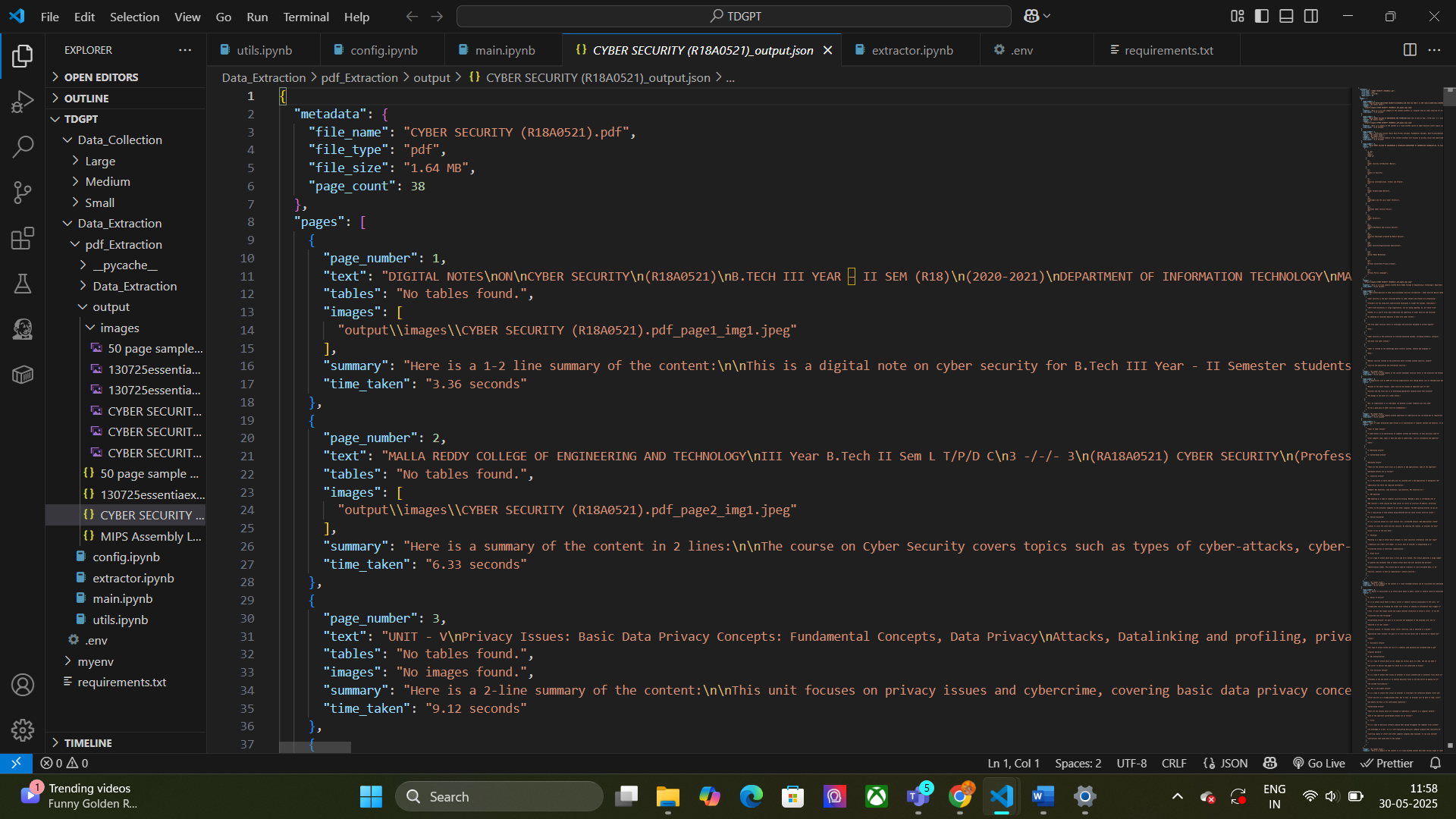
# Sample Output JSON Structure

{  
 "metadata": {  
 "file\_name": "example.pdf",  
 "file\_type": "pdf",  
 "file\_size": "1.5 MB",  
 "page\_count": 10  
 },  
 "pages": [  
 {  
 "page\_number": 1,  
 "text": "This is the extracted text...",  
 "tables": ["Table content as text..."],  
 "image\_paths": ["output/images/example\_page1\_img1.png"],  
 "summary": "This page discusses the overview of..."  
 }  
 ],  
 "overall\_summary": "Document provides a comprehensive overview of...",  
"total\_time\_taken": "25.6 seconds"

}  
  
Features Recap

* Handles PDF, DOCX, PPTX
* Extracts text, tables, and images
* OCR-enabled for scanned files
* Chunk-wise LLM summarization
* Structured JSON output
* Optimized for large-scale document processing

Sample Output





#### Pros:

* **Accurate** table extraction via pdfplumber
* **Reliable image** saving using fitz
* Full control of chunking and layout
* Works consistently across large documents
* Easier debugging and optimization

#### Cons:

* Slightly **longer initial setup**
* Requires manual format detection

Benefits

 Multi-Format Support: Handles PDFs, Word (DOCX), and PowerPoint (PPTX) files.

 Smart Summarization: Uses LLMs (like Groq LLaMA 3) to summarize content in chunks.

 Image & Table Extraction: Automatically detects and saves images and tables.

 OCR Enabled: Extracts text from scanned PDFs using Tesseract OCR.

 Structured Output: Delivers clean JSON output for integration and analysis.

 Offline Ready: Can work locally (except for LLMs), ideal for secure environments.

## Limitations

* **LLM Dependency:** Summarization needs an API key and internet access.
* **OCR Accuracy:** Depends on scan quality; poor images may reduce accuracy.
* **Layout Loss:** Does not preserve original formatting or styles.
* **Processing Time:** Large files may take time to process.
* **Complex Tables:** Might struggle with nested or irregular tables.
* **Language Support:** Primarily optimized for English documents.

Method 2: Using Unstructured Partitioning

Technologies Used

• Python – Core programming language  
• os, json, pandas – File handling and structured output  
• python-dotenv – Load environment variables  
• tqdm – Show progress during processing  
• unstructured – Extract text, tables, images from documents  
• pytesseract – OCR via Tesseract  
• Tesseract OCR – Recognize text in scanned PDFs  
• Groq API (LLaMA 3 70B) – Summarize extracted content using LLM

LLM Comparison Summary

| **Model** | **Speed (on Groq)** | **Token Limit** | **Accuracy** | **Cost Efficiency** | **Used in Project** |
| --- | --- | --- | --- | --- | --- |
| GPT-4 Turbo | Moderate | 128k | High | Costly | No |
| Claude 3 Opus | High | 200k | High | Moderate | No |
| LLaMA 3 70B | Very High | 8k | High | High | Yes |

Supported File Formats & Partition Modules

#### PDF

* **Module Used:** unstructured.partition.pdf
* **Extra Option:** ocr=True for scanned PDFs
* **OCR Engine:** pytesseract
* **Use:** Extracts narrative text, tables, and embedded images from PDFs (including scanned).

#### DOCX

* **Module Used:** unstructured.partition.docx
* **Use:** Extracts paragraphs, titles, and lists from Word files.

#### XLSX

* **Module Used:** unstructured.partition.xlsx
* **Use:** Extracts table content, headers, and sheet metadata.

#### PPTX

* **Module Used:** unstructured.partition.pptx
* **Use:** Extracts titles, bullet points, speaker notes.

#### HTML

* **Module Used:** unstructured.partition.html
* **Use:** Parses and extracts main text and structured sections.

Performance

|  |  |  |
| --- | --- | --- |
| File Size | Pages | Time Taken |
| Small | 30–50 | ~15–22sec |
| Medium | |  | | --- | |  |  |  | | --- | | 300–600 | | |  | | --- | |  |  |  | | --- | | ~50–120 sec | |
| Large | 600+ | ~180–240 sec |

Output JSON Structure

{  
"file": "sample.pdf",  
"summary": "This document covers Big Data Analytics, architectural frameworks, real-time applications...",  
"text\_elements": [  
{  
"type": "NarrativeText",  
"text": "Big data refers to datasets that are too large or complex...",  
"metadata": {...}  
}  
],  
"tables": [  
{  
"type": "table",  
"text": "Framework | Description\nHadoop | Distributed storage...",  
"metadata": {...}  
}  
],  
"images": [  
{  
"type": "image",  
"text": "",  
"metadata": {  
"page\_number": 4,  
"coordinates": [100, 200, 300, 400] } }  
]  
}

### Features Summary

* Automatic file type detection
* AI-based summarization (LLM via Groq)
* Structured output for text, tables, and images
* Export to JSON and CSV
* OCR support for scanned documents

Sample Execution Output

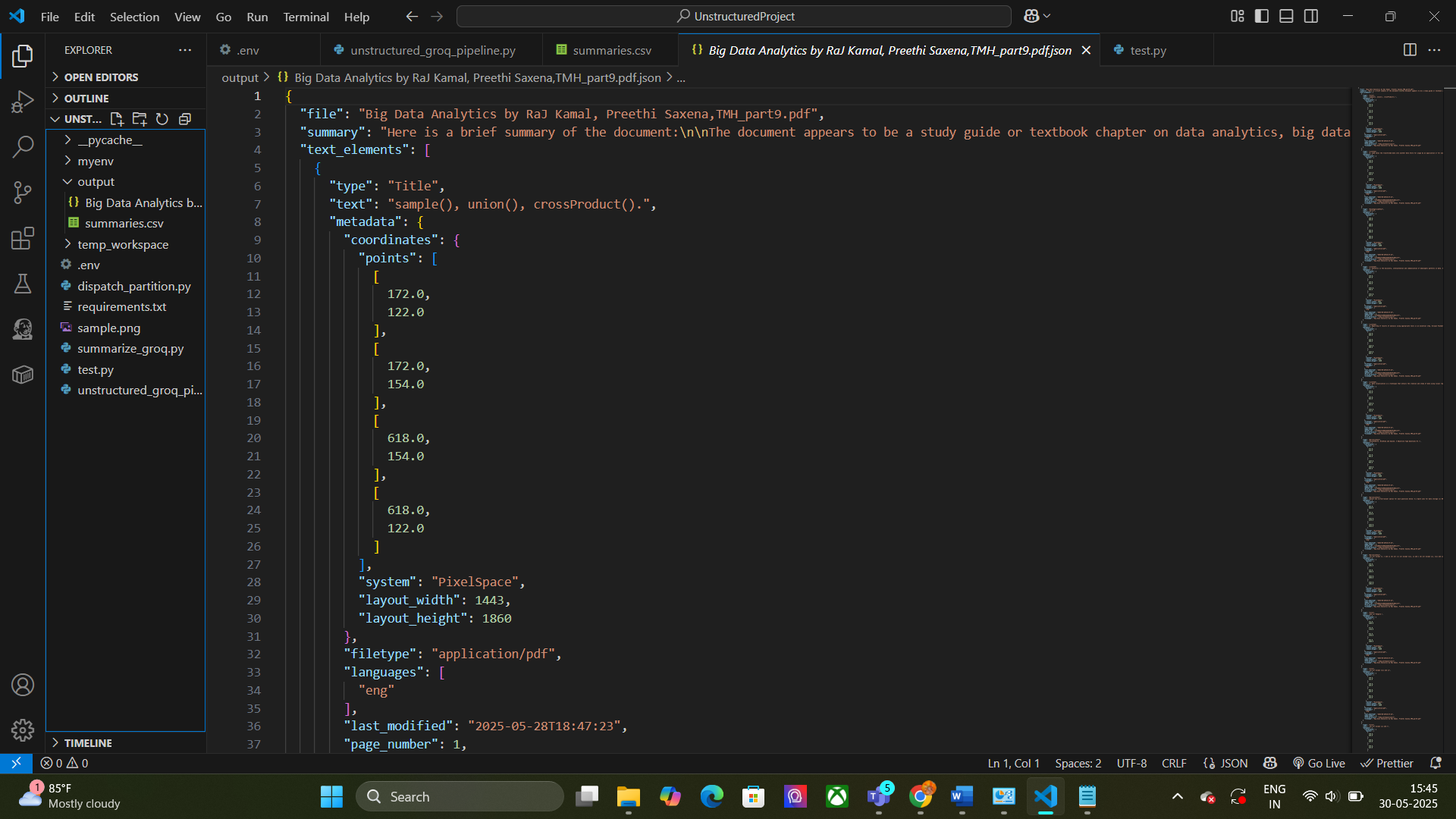
A progress bar showing :

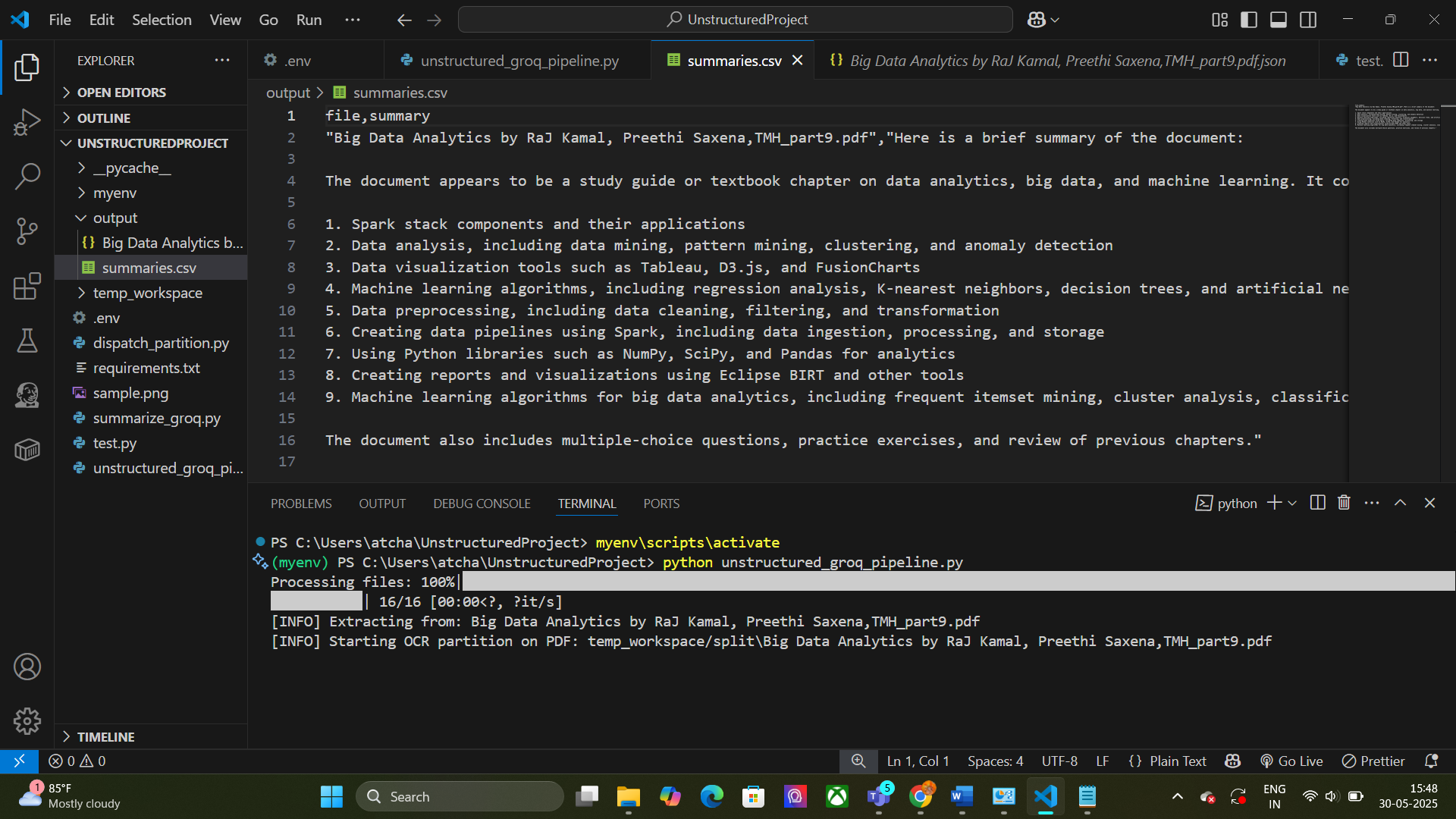
Processing files: 100%|████████████████████████████| 3/3 [00:08<00:00, 2.56s/it]

✓ Successfully processed: BigData.pdf

✓ Successfully processed: Report.docx

✓ Successfully processed: Slides.pptx

Output saved in /output/



Pros

 Handles multiple file types and extracts text, tables, and images

 Uses OCR for scanned PDFs

 Provides clear summaries with Groq LLaMA 3 70B

 Outputs structured JSON and CSV files

 Includes progress tracking for batch processing

Cons

 OCR depends on scan quality it’s not properly extract images and tables format

 Large files can take longer to process

 Limited support for complex tables and media

 Requires proper Tesseract setup

 Summarization quality varies with model and input

 Only supports specific file formats out of the box