DOCUMENT EXTRACTION MVP (Assignment-1 E)

Introduction; I chose the **DOCUMENT EXTRACTION MVP** for Assignment-1 E. The reason I chose this was personal. Even for small tasks, we are now referring to LLMs; thus, all our data goes to MNCs, and we become victims of targeted ads. At a personal level, this is manageable, but for corporates, data leakage is a critical issue. The problem is that they often cannot run LLMs on their own hardware. So, I decided to undertake this project to create an energy-efficient pipeline that anyone can use with simple code, ensuring their data remains safe while running on minimal hardware.

As mentioned, I built a robust ML pipeline that can run easily on minimal hardware requirements, such as 16GB DDR4 RAM. I tested different models, including **Qwen 2 and Phi-3,** but finally found the sweet spot with **Phi-3 Mini running locally through Ollama, because it provides a good balance between performance, speed, and hardware efficiency.**. The system achieves energy efficiency through lightweight local SLM inference, intelligent chunk filtering, regex preprocessing, and targeted structured extraction that minimizes unnecessary compute usage .It is capable enough to perform the task within limited hardware constraints.

Features of the Code:

1. Energy-efficient local inference using Phi-3 via Ollama (runs on minimal hardware).
2. The full pipeline: includes PDF extraction, chunking, model loading, JSON processing, and DB storage. It is implemented in a single script.
3. Real-time pipeline monitoring using structured logging and progress tracking to provide live visibility into extraction, processing, and database operations..
4. Text cleaning using Regex before feeding it to the model, hybrid extraction (LLM + regex fallback)
5. Easily scalable to larger models by changing variables like chunk size and text limit.
6. Vector parsing, aggregation, and chunking are carefully implemented to minimize model hallucination.

(Figure 1: Extraction | Figure 2: SLM): I've also made a flow diagram to give holistic understanding of this project.

Limitations:

1. Due to token size constraints, long or highly complex document contexts may reduce extraction accuracy.
2. Once loaded, the model retains residual memory of the previous prompt, so methods to clear prompt residue need to be explored.
3. Hallucinations may occur if semantically similar tasks are processed repeatedly.
4. It is a generalized model and cannot perform highly specific domain tasks without tuning.

Future Work:

1. **Expanded Cleaning:** Expand Regex text cleaning to cover almost all document types. This could be achieved by using a small classifier model to identify the document type and apply customized Regex rules within the pipeline.
2. **Mobile Support:** Enable execution on iOS and Android via JS or Kotlin extensions.
3. **Decentralization:** Its full potential lies in Decentralized AI, which can be used everywhere to ensure privacy and ease of use.
4. **Fine-tuning:** Fine-tuning on specific tasks, like JSON extraction, could improve efficiency significantly.

Conclusion I believe that through fine-tuning, this SLM (Small Language Model) can be integrated into industries like pharma, materials science, and banking, which are highly sensitive regarding their data. It has the potential to change the world.

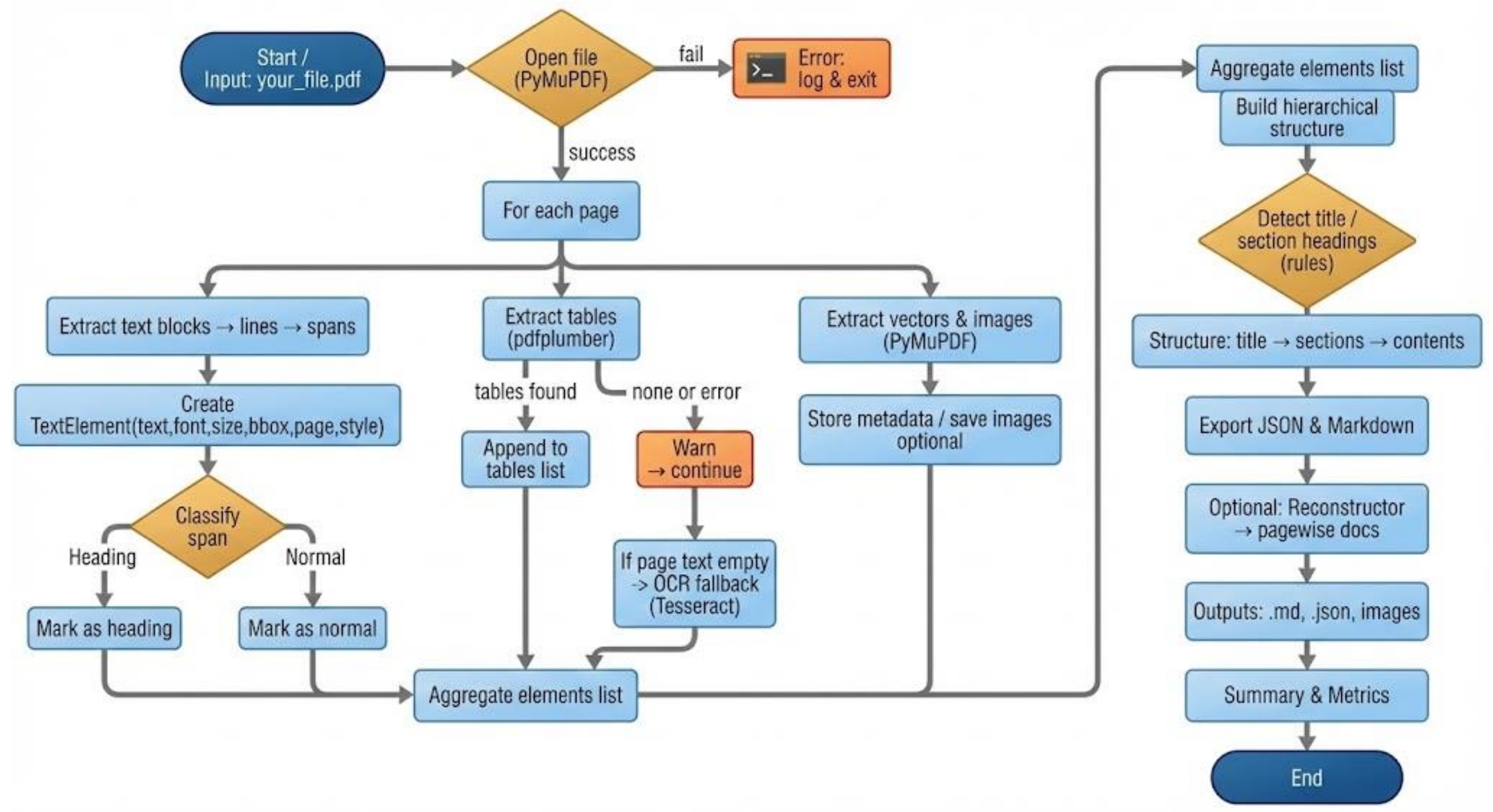
Start / Input: your_file.pdf
Open file (PyMuPDF)
fail
Error: log & exit
success
For each page
Extract text blocks → lines → spans
Create TextElement(text,font,size,bbox,page,style)
Classify span
Heading
Normal
Mark as heading
Mark as normal
Extract tables (pdfplumber)
tables found
none or error
Append to tables list
Warn → continue
If page text empty -> OCR fallback (Tesseract)
Extract vectors & images (PyMuPDF)
Store metadata / save images optional
Aggregate elements list
Aggregate elements list
Build hierarchical structure
Detect title / section headings (rules)
Structure: title → sections → contents
Export JSON & Markdown
Optional: Reconstructor → pagewise docs
Outputs: .md, .json, images
Summary & Metrics
End

FIgure.1. **Extraction**

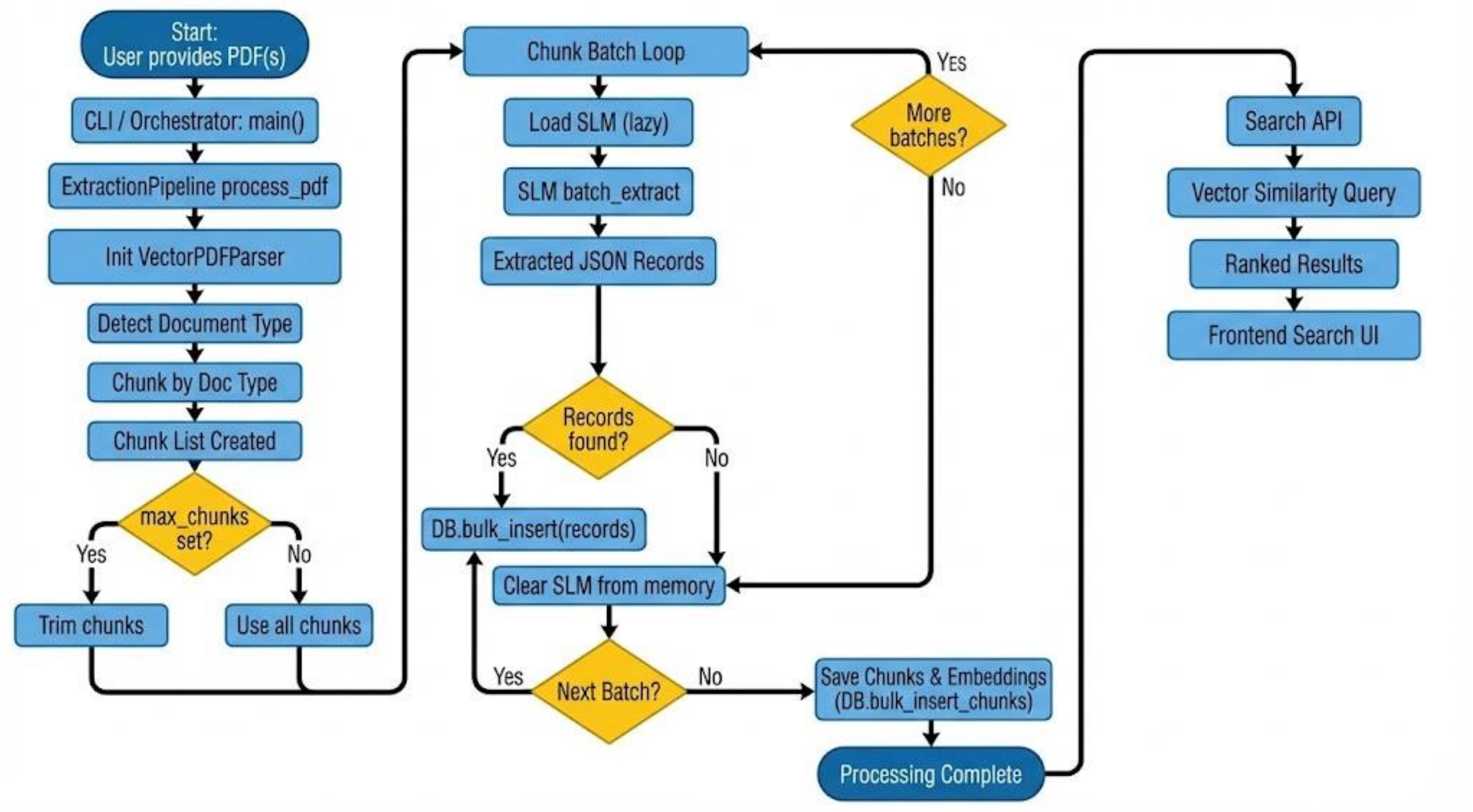

Figure.2.**SLM**

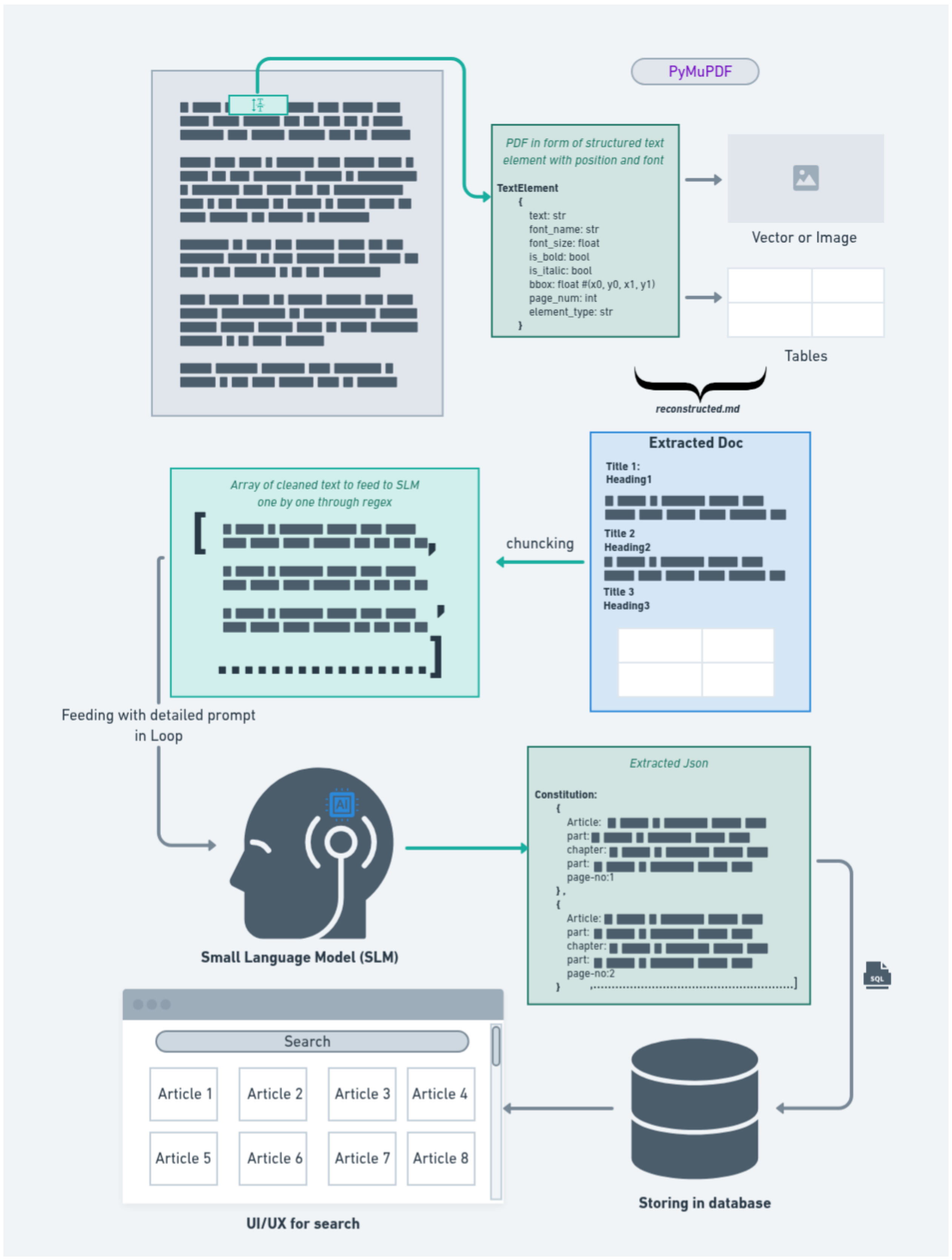

Figure.3. Semantic diagram of project