**Stepwise breakdown of project:**

**1. Loading the PDF**

* **Libraries/Tools:** PyMuPDF (fitz), PyPDF2, pdfminer.six
* Load the PDF file for processing.
* **Steps:**
  + Open the PDF file.
  + Ensure the PDF is not password-protected (if it is, you need to decrypt it first).

**2. Extracting Text from PDF**

* **Libraries/Tools:** PyMuPDF, pdfminer.six, pdfplumber
* Extract raw text content from each page.
* **Steps:**
  + Iterate over each page and extract text.
  + Clean the text (remove unwanted newlines, spaces, headers/footers if needed).
  + Normalize the text (remove special characters, convert to lower case if necessary).
  + Optionally, identify and extract metadata (like title, author, publisher).

**3. Extracting Images**

* **Libraries/Tools:** PyMuPDF, pdf2image, Pillow
* Extract images (both embedded and inline) from the PDF.
* **Steps:**
  + Identify image objects in the PDF (using libraries like PyMuPDF).
  + Convert or extract images in the desired format (e.g., PNG, JPG).
  + Store images locally or in cloud storage (if they are large).

**4. Extracting Diagrams and Tables**

* **Libraries/Tools:** pdfplumber, camelot, tabula-py
* Extract structured data such as tables and diagrams.
* **Steps:**
  + For tables, use camelot or tabula-py to extract structured data (in DataFrame format).
  + For diagrams (like flowcharts, graphs), use libraries like PyMuPDF or pdf2image to extract images and then apply OCR (if needed) or manually classify the content.

**5. Breaking Down the Extracted Text into Chunks**

* **Libraries/Tools:** nltk, spaCy, textsplitter
* Break down the text into smaller chunks to handle large documents.
* **Steps:**
  + Split the text into logical chunks such as paragraphs, sections, and sub-sections.
  + Break large chunks into smaller sizes to optimize storage and processing (e.g., limit chunk size to 500-1000 tokens).
  + Optionally, classify the chunks based on headers, chapters, and topics.

**6. Creating Embeddings for Each Chunk**

* **Libraries/Tools:** sentence-transformers, HuggingFace, OpenAI API
* Create embeddings for each chunk of text.
* **Steps:**
  + Use a pre-trained language model (like Sentence-BERT or OpenAI's GPT models) to generate embeddings for each text chunk.
  + Store the embeddings as vector representations (to later facilitate similarity search and comparison).

**7. Storing the Data in MongoDB (or another DB)**

* **Libraries/Tools:** PyMongo, MongoDB Atlas, SQLite, PostgreSQL
* Create a schema to store the extracted data.
* **Steps:**
  + Store the extracted text, tables, images, and embeddings in MongoDB collections.
  + **MongoDB Schema Example:**

{

"document\_id": "unique\_document\_identifier",

"page\_number": 1,

"text\_chunk": "text\_data\_here",

"image": "image\_path\_or\_blob",

"embedding": [vector\_of\_embeddings]

}

* + Consider storing different types of data (text, tables, images) in separate collections or as nested documents.
  + Store the document metadata (author, title, etc.) in a separate collection for easy retrieval.

**8. Indexing and Search**

* **Libraries/Tools:** FAISS, Pinecone, elasticsearch
* **Steps:**
  + Index the embeddings using a vector search engine like FAISS or Pinecone.
  + Use an inverted index for text-based search (e.g., Elasticsearch) to quickly retrieve text chunks based on search queries.

**9. (Optional) Building a User Interface**

* **Libraries/Tools:** Flask, Django, Streamlit, Gradio
* Create a simple web interface to query and retrieve chunks of text, images, and other data.
* **Steps:**
  + Provide a search bar where users can enter keywords.
  + Display results by retrieving text chunks, images, and tables from the database.
  + Highlight the parts of text or images relevant to the search.

**10. Optional: Post-processing (OCR for Scanned PDFs)**

* **Libraries/Tools:** Tesseract OCR, easyocr
* If the PDF contains scanned images (rather than searchable text), use OCR (Optical Character Recognition) to extract the text.
* **Steps:**
  + For pages with scanned images, use OCR to extract the text.
  + You may need to clean up the OCR results manually to handle misrecognitions.

**High-Level Workflow:**

1. **Load PDF** -> 2. **Extract Text, Images, Diagrams, Tables** -> 3. **Chunk Text** -> 4. **Create Embeddings** -> 5. **Store in DB** -> 6. **Index for Search** -> 7. **Provide UI for Search (optional)**

**Other Considerations:**

* **Error Handling:** Ensure to account for potential parsing issues (e.g., malformed PDFs, missing images).
* **Performance:** If working with very large PDFs, you may need to implement batching or multi-threading to handle the data processing.
* **Storage:** Depending on the size, decide whether to store embeddings directly in the database or store them in a separate vector database for fast retrieval.