Phase - 2 Development

- Implemented Text-Extraction, Splitting & Embeddings in the past week
- *For text-extraction:*
    - PyMuPDF
- *For Text Chunks:*
    - Recursive Character Text-Splitter
- *For Chunk Embeddings:*
    - Word-to-Vec
    - Sentence-Transformers
    - Avg. Word-to-Vec

Text Extraction:

- PyMuPDF
    - Flexible and efficient
    - This can extract text,images & tables other than using separate libraries for different formats
- The extracted text will be then sent to chunking text.

Text Chunks:

- Here the extracted text is further split to chunks for breaking down and splitting long documents down into smaller chunks that can fit our model context window.
- We mainly worked on:
    - Token
    - Recursive
- Even though token splitting was easy to implement it has limited scalability issues as it is being restricted by the limiter.
- So we moved forward with **Recursive Text Splitter.** It works in the following way
    - Splits by paragraphs first
    - If paragraph too long ,then split by sentences
    - If a sentence is still too big , then it splits by characters

Text Embeddings:

- Word-to-Vec
    - Generates word-level embeddings but lacks contextual understanding.
- Sentence-Transformers
    - Provides context-aware embeddings for sentences and paragraphs.
- Avg. Word-to-Vec
    - Averages word embeddings for larger text but loses contextual nuances.

Process:

- Text chunks from the Recursive Text-Splitter are converted into vector representations.
- These embeddings are used for tasks like retrieval and similarity search.