OCRMYPDF Integration Documentation

Enhance the file upload process to handle scanned PDFs by applying Optical Character Recognition (OCR) using ocrmypdf. If a PDF has no extractable text, OCR is performed and the resulting text is saved as a separate .txt file in the database.

The objective of this enhancement was to ensure that all PDF documents uploaded via the /user/file/upload endpoint whether digitally generated or scanned are fully machine readable and searchable. To achieve this, we integrated ocrmypdf to automatically detect and process scanned PDFs that lack extractable text. Using PyPDF2, the system first checks if a text layer exists; if not, OCR is applied to extract the textual content. This extracted text is then saved as a separate .txt file in the file_store with file_type = 'text/plain', enabling advanced downstream capabilities such as full-text search, token-based indexing, summarization, and AI powered retrieval. This upgrade ensures consistent document utility across all input types and significantly improves the platform's ability to handle scanned documents.

Server Implemented on:
(ec2-13-202-103-72.ap-south-1.compute.amazonaws.com)

(Site: https://enterprise.techpeek.in/chat, http://13.202.103.72/)

1) POST /user/file/upload:

- **Path: cd onyx/backend/onyx/server/user_documents/api.py**
- **Function: upload_user_files**

➢ **Implementation Details:**

- **Text Detection:** Utilized **PyPDF2.PdfReader** to check if the uploaded PDF contains extractable text. If no text is found, the PDF is treated as "**scanned**" and OCR is applied.
- **OCR Processing:** OCR is performed using the **ocrmypdf.ocr()** function, with **force_ocr=True** to handle scanned only PDFs. The output is a searchable PDF, but text is extracted separately using **extract_file_text().**
- **Text File Creation:** Extracted text is stored in the **file store** with **file_type** = "**text/plain**", **file_origin = FileOrigin.CHAT_UPLOAD**, Filename suffixed with **_text.txt**. This .txt file is stored alongside the original/processed PDF in the database.

- Helper Function: apply_ocr_if_needed(upload_file: UploadFile) → was introduced to encapsulate the entire OCR detection and processing logic.

- File Type Check (PDF only): if not upload_file.filename.lower().endswith(".pdf"): return upload_file → If the uploaded file isn't a PDF, it bypasses OCR entirely and returns the file as it is.

- Text Layer Detection: pdf_reader = PdfReader(upload_file.file) & has_text = any(page.extract_text() for page in pdf_reader.pages) → Tries to extract text from each page using PyPDF2. If any text is found, the PDF is assumed to be machine readable and OCR is skipped.

- Fallback on Error: except Exception: has_text = False → If PDF parsing fails (e.g., due to corruption), it assumes no text and proceeds with OCR.

- Early Exit (No OCR needed): if has_text: return upload_file.

- Temporary Files Setup for OCR: with tempfile.NamedTemporaryFile(suffix=".pdf", delete=False) as input_tmp input_tmp.write(upload_file.file.read()) input_path = input_tmp.name → Saves the uploaded PDF to a temp file (input_path). Prepares a second temp file (output_path) to store the OCR output.

- Run OCR: ocrmypdf.ocr(input_path, output_path, force_ocr=True) → Applies OCR using ocrmypdf, forcing text extraction even if the file contains images. If OCR fails, an HTTPException is raised to signal server error.

- Wrap OCR Output as UploadFile: return StarletteUploadFile(...) → Reads the OCR-processed PDF and wraps it in a StarletteUploadFile (mimicking the original format), so it can continue through the pipeline as if it were the original upload.

2) Deprecated Logic:

- OCR handling that previously existed in:

  backend/onyx/server/query_and_chat/chat_backend.py Inside the commented upload_files_for_chat() function.

- This older implementation was phased out in favor of a centralized and modern upload pipeline under /user/file/upload.

3) Dependencies Added:

- Path : onyx/backend/requirements/default.txt

- **New libraries added to support OCR:**

- ocrmypdf==14.0.0
- pikepdf==7.2.0
- PyPDF2==3.0.1
- pytesseract==0.3.10

4) Known Issue: OCR Quality Degradation on Low-Quality Scanned PDFs:

- During testing, I encountered an issue where the OCR output contained excessive diacritics or malformed characters, especially in documents with poor scan quality or faint text. This was often accompanied by internal warnings such as:

  ```
  [tesseract] lots of diacritics – possibly poor OCR
  ```

- This behavior is typically caused by →
- Low-resolution scanned images embedded in PDFs
- Excessive noise or compression artifacts
- Skewed or faded text areas

- **Proposed Solution:** The goal is to ensure cleaner input for Tesseract OCR, resulting in more accurate text extraction. By uploading clean, high-resolution documents, users can significantly improve OCR accuracy and downstream functionality like search, summarization, and question-answering.

5) OCR Validation Process:

- **To confirm that the OCR integration works as expected, I performed live containerized validation and PostgreSQL-level audits:**

- Docker Validation: Used Docker command → `sudo docker exec -it onyx-stack-relational_db-1 psql -U postgres`

- Entered the running PostgreSQL container and queried the file_store table:

  ```
  SELECT * FROM file_store WHERE file_type = 'text/plain';
  ```

- Validation Criteria: A consistent naming pattern like *_text.txt (e.g., scanned_pdf_new_1_text.txt, Digitally Scanned Pdf Pt.1_text.txt).

- All entries file_type = 'text/plain', confirming they are plain text files generated from OCR.

- The origin, CHAT_UPLOAD, was seen matching the logic in our new implementation.

6) Conclusion:

- Based on suffix patterns and metadata, these .txt files are OCR-processed extractions from scanned PDFs uploaded via the new /user/file/upload route.

- This confirms that OCR logic is being executed correctly and data is being stored as designed.

7) Example Record:

- file_name: 67045805-f472-413a-8ff9-0d9178dcc65d
- display_name: Digitally Scanned Pdf Pt.2_text.txt
- file_origin: CHAT_UPLOAD
- file_type: text/plain

- This .txt file corresponds to an OCR-processed version of Digitally Scanned Pdf Pt.2.pdf.

- Ibrahim Sultan