

Addendum

Interactive Model:

<https://www.figma.com/proto/r0eBzra7rWsMFfggfcmBOO/Untitled?node-id=19%3A237&scaling=contain&page-id=0%3A1&starting-point-node-id=19%3A237>

This is the interactive model for the design and flow of the online web interface.

Metadata Structure:

The metadata structure will be based on the initial files received from the technical communication department as shown in the figure below.



Figure 1: Metadata file format.

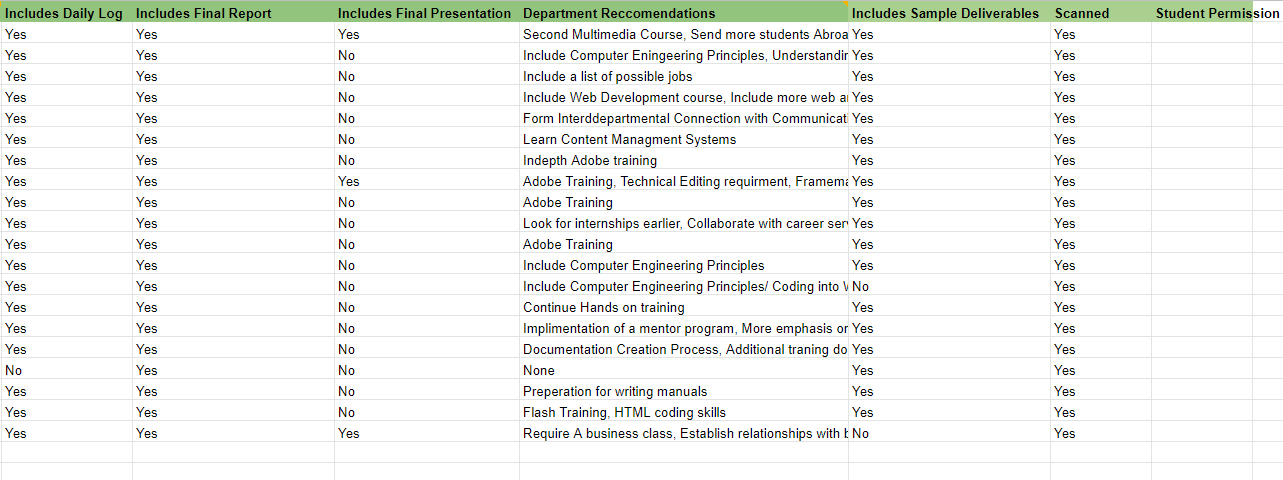


Figure 2: Metadata file format.

The data will be categorized into 12 different groups pertaining to title, author, job type, year completed, employer, supervisor, including digital media, deliverables, includes daily log, includes final report, includes final presentation, department recommendations, and includes sample deliverables. These are just examples of what the metadata will look like for some of the projects submitted to the project library and most likely will be the initial group, due to them already being established upon scan. The metadata for the other types of projects could be the certain deliverables the project withholds, engineering department, client, project manager, and etc. There are many possibilities for what the metadata can hold and most of the metadata is still to be determined, but this gives a fair example of what the metadata will ultimately look like.

Uploading Page:

One form of retrieving data from a pdf upon upload would be to use the pdfminer tool, which is a FastAPI pdf tool. For full support to use within the interface, the contingencies that need to be installed are tesseract and ghost script. It will parse through the pdf and collect .json data that can then be used to be uploaded to the metadata. As in our web interface, we will use the pdf miner to collect data from the pdf to place into the temporary metadata fields upon upload, which can then be modified by the user if they notice something wrong. This feature allows for the user to make sure all information being submitted to the metadata database is correct and the project will be able to be found correctly without worrying.

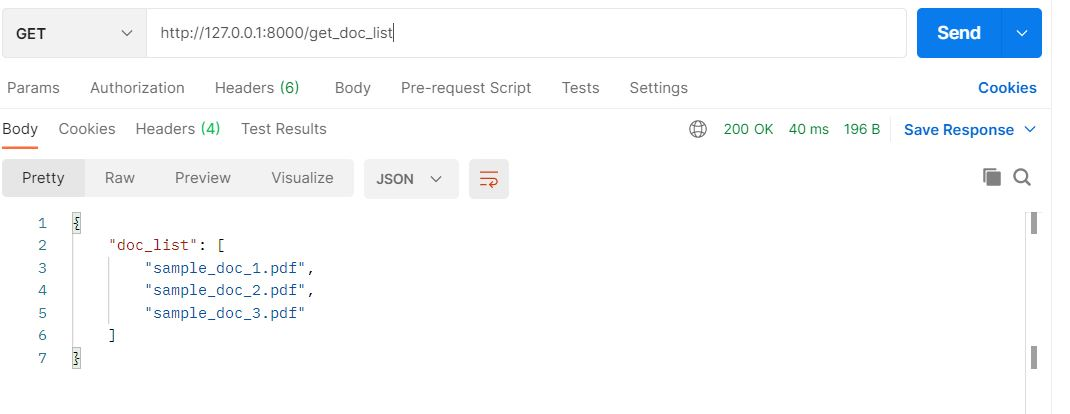


Figure 3: List of PDFs from pdfminer.

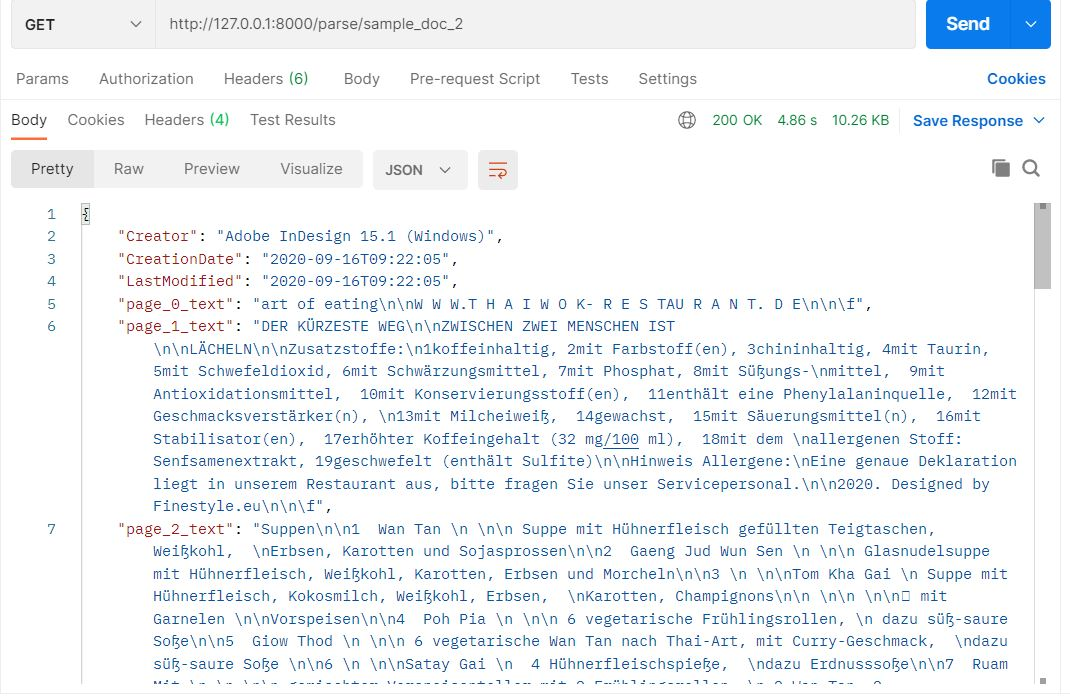


Figure 4: Sample of the information pulled from pdfminer.

Since our interface uses FastAPI, this tool will be easily implemented and allow for the least amount of changes to the necessary components used.

Plan For Improving Search Function:

We plan to optimize our Search Function for our users. When a user searches for a word or set of words, the user should be able to see a list of all the projects containing the keywords that have been typed. For example, if a user searches “trees”, the search function should display all the data containing the keyword “trees”. In addition, when the user selects a project that interests them, there will be 2-3 additional projects that will be labeled as “Project Suggestions”. We will have to consider the user’s time as well. The search function should be able to run smoothly and display results quickly. A slow search function could cause the user to feel upset and steer away from our web interface. A “No Results” page or another page similar can drive away the user from our web interface as well. To prevent this from happening features such as auto-correction and auto-completion feature will be added.We can implement this by python packages such as google and beautifulsoup4.

How will accessibility be addressed:

In our web interface, the user will have two accessibility features. While viewing a project with the PDF Viewer function, the user will be able to navigate through pages using the direction keys on the keyboard. This will make it easier for any user to flip through pages rather than using a laptop’s mouse to navigate. Another feature will include an audio transcript. This will allow the user to listen to a project without having to read through the document with a click of a button. The feature will read the document and generate an audio (text-to-speech transcription).