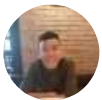


PDF Manipulation Application Part 1 — Creating an application using Streamlit and PDFRW



DesmondChoo

Follow



Jun 27, 2021 · 6 min read ★

...

Add watermark features, remove metadata or even concatenating different Doc/Docx/PDF files together in a interactive web application using python library Streamlit.

PDF MANIPULATION



Why Streamlit?

Streamlit turns data scripts into shareable web apps in minutes [1]. Most importantly, no front end experience is needed. With Streamlit, the app is able to get user input on


their requirement and inputs so as to customise a solution for them. This definitely makes it more engaging for the user as well!

Purpose of this application

This script utilizes the package Streamlit and PDFRW to add watermark, remove metadata or concatenate PDF or Document (Doc/Docx) and outputting them either into individual PDF or one single PDF (if concatenation option is chosen).

While there are many solutions out there that can solve this issue easily, there aren't many solutions on the python package PDFRW, and certainly not many combine the solution with the excellent interface offered by Streamlit.

Streamlit interface of the application



The image shows the Streamlit interface for PDF manipulation. It features a logo at the top with the text "PDF MANIPULATION" and icons for PDF and DOC files. Below the logo, there are two main sections: "Choose Options" and "Choose type of manipulation to PDF".

Choose Options

Select option:

Input files manually

Upload PDF Files

Drag and drop files here
Limit 200MB per file + PDF

Browse files

Please input the output path to house your PDFs

Choose type of manipulation to PDF

Select one or more options:

Add Watermark Remove Metadata Concatenate PDFs

Upload watermark PDF for portrait

Drag and drop file here
Limit 200MB per file + PDF

Browse files

Upload watermark PDF for landscape

Drag and drop file here
Limit 200MB per file + PDF

Browse files

Once you have selected the required options above, you can click on the button below to start processing.

Click here to start!

The interface also shows a dropdown menu for "Select option:" with the following options:

- Input files manually
- Input files manually
- Input path

Below the dropdown menu, there is a section for "Choose Options" with the following options:

- Input path

Please input the path of your folder

Please input the output path to house your PDFs

Example of the interface and possible configurations

Features of application

This web application is built based on my personal requirement: (1) Incorporating header and footer; (2) Converting doc/docx into PDF; and (3) Option of concatenating “amended” document together into one PDF.

Watermarking document (Incorporating header and footer)

Currently the watermark features is done by overlaying another PDF containing the watermark on the actual PDF. It has two version, landscape and portrait. User must drag and drop these files if watermark feature is selected. Script will automatically detect the orientation of the PDF and incorporate the right watermark onto the PDF.

Input path (doc/docx can only be ran on this option) vs input files manually

For the option “input path”, user has to input the path of the folder and the script will take into account both PDF and doc/docx files within that folder.

For option “input files manually”, user just need to drag and drop the required files into the drop down boxes. The downside of this option is that it only support PDF files but can handle multiple files at one go.

Concatenation vs non-concatenation

If concatenation option is selected, all PDFs will be save into one PDF call “Annex.pdf” in the prescribed output path. If option is not selected, individual PDF will output as “Annex 1.pdf, Annex 2.pdf, ..., Annex X.pdf”

Possible improvement of this web app going forward

While this web application does fulfill my current requirement, I do see many potential improvements that can be implemented in order to take this application to the next level.

Incorporating other file types

For starter, this web application is only able to take doc/docx file types if the user select the option “Input path” which the script will search the path for all doc/docx and pdf files for manipulation.

Secondly, this fuctionality should be able to be replicated to other file types like excel or txt. Though the usefulness of this feature might not be really necessary.

Ability to identify the size of document

Currently, while the script is able to automatically identify the orientation of the document (landscape or portrait) and apply the right watermark for it, it only works if the document is of size A4.

And of course not all the document will be in A4. In order to truly be automated, and not semi-automated, it will be really useful for the script to be able to automatically identify the size of the document (if it is A4, B5 or even letter) and apply the watermark feature of the right dimension onto them.

Enhancement to watermark features

The script can definitely do better in terms of applying watermark features onto PDF. Currently, the script overlay either a landscape or portrait watermark PDF onto the documents before exporting them as a whole. While this works, there is no customisation offered to the user.

User should be able to input the words of the watermark, the location where the watermark will be placed and also the orientation.

Hosting the app on Heroku or other host

Finally, the final step is to host the application on platform such as Heroku so that this web application can be easily shared and used.

Final Thought

This is definitely a fun project for me, and a large part were due to this script being implemented in Streamlit which forced me to think in terms of a user perspective.

Lastly, a big thanks for reading my article! :)

Full code

```
import streamlit as st
from pdfw import PdfReader, PdfWriter, PageMerge, IndirectPdfDict
import pathlib
import os
from os import path
from glob import glob
from PIL import Image
```

```

import numpy as np
import comtypes.client
from pathlib import Path

##### Streamlit #####
def load_image(img):
    im = Image.open(img)
    image = np.array(im)
    return image

st.image(load_image(os.getcwd()+"\Title.png"))
st.write("###")

st.subheader("Choose Options")

config_select_options = st.selectbox("Select option:", ["Input files manually", "Input path"], 0)

if config_select_options == "Input files manually":
    uploaded_file_pdf = st.file_uploader("Upload PDF Files", type=['pdf'], accept_multiple_files=True)
    # uploaded_file_doc = st.file_uploader("Upload doc/docx Files", type=['docx', 'doc'], accept_multiple_files=True)
else:
    input_path = st.text_input("Please input the path of your folder")
    uploaded_file = []
    if len(input_path) > 0:
        st.write(f"PDFs in this path {input_path} will be uploaded")

output_path = st.text_input("Please input the output path to house your PDFs")
if len(output_path) > 0:
    st.write(f"Amended PDFs will be housed in this path {output_path}")

st.write(" - - -")
st.subheader("Choose type of manipulation to PDF")

config_select_manipulation = st.multiselect("Select one or more options:", ["Add Watermark", "Remove Metadata", "Concatenate PDFs"], ["Add Watermark", "Remove Metadata", "Concatenate PDFs"])
if "Add Watermark" in config_select_manipulation:
    uploaded_file_wmp = st.file_uploader("Upload watermark PDF for portrait", type=['pdf'])
    uploaded_file_wml = st.file_uploader("Upload watermark PDF for landscape", type=['pdf'])

##### Actual Code #####

def main():
    ## Set up progress bar
    st.write(" - - -")
    st.subheader("Status")

```

```

progress_bar = st.progress(0)
status_text = st.empty()

status_text.text("In progress... Please Wait.")

## Checking the output directory
if not os.path.exists('output_path'):
    os.makedirs('output_path')
    status_text.text("Output path checked okay. Proceeding to next
step...")

## Define the reader and writer objects

writer = PdfWriter()
if "Add Watermark" in config_select_manipulation:
    watermark_input_P = PdfReader(uploaded_file_wmp)
    watermark_input_LS = PdfReader(uploaded_file_wml)
    watermark_P = watermark_input_P.pages[0]
    watermark_LS = watermark_input_LS.pages[0]
    status_text.text("Loaded Watermark PDF. Progressing to the next
step...")
    progress_bar.progress(0.25)

def find_ext(dr, ext):
    return glob(path.join(dr, "*.{0}".format(ext)))

wdFormatPDF = 17

if config_select_options == "Input path":

    filepath_doc = find_ext(input_path, "doc")
    filepath_docx = find_ext(input_path, "docx")
    filepath_all_doc = filepath_doc + filepath_docx

for file in filepath_all_doc:
    name = Path(file).name.split(".")[0]
    word = comtypes.client.CreateObject('Word.Application', dynamic =
True)
    word.Visible = True
    doc = word.Documents.Open(file)
    doc.SaveAs(input_path+"\\\"+ name + ".pdf", wdFormatPDF)
    doc.Close()
    word.Quit()
    filepath = find_ext(input_path, "pdf")
else:
    # for file in uploaded_file_doc:
    # #name = Path(file).name.split(".")[0]
    # word = comtypes.client.CreateObject('Word.Application', dynamic =
True)
    # word.Visible = True
    # doc = word.Documents.Open(file)
    # doc.SaveAs(output_path+"\\intermediate\\"+ file + ".pdf",
wdFormatPDF)
    # doc.Close()

```

```

# word.Quit()
# filepath_pdf = find_ext(output_path+"\intermediate","pdf")
# filepath = uploaded_file_pdf + filepath_pdf
filepath = uploaded_file_pdf

## Create a loop for all the paths
for i in range(len(filepath)):
    file = filepath[i]
    reader_input = PdfReader(file)
    status_text.text(f"Processing {file} now...")

if "Add Watermark" in config_select_manipulation:
    ## go through the pages one after the next
    for current_page in range(len(reader_input.pages)):
        merger = PageMerge(reader_input.pages[current_page])

    try:
        mediabox = reader_input.pages[current_page].values()[1]['/Kids'][0]
        ['/MediaBox']
    except TypeError:
        mediabox = reader_input.pages[0].values()[1]

    if mediabox[2] < mediabox[3]:
        merger.add(watermark_P).render()
    else:
        merger.add(watermark_LS).render()
    writer.addpages(reader_input.pages)
    status_text.text(f"Watermark done for {file}...")
    else:
        writer.addpages(reader_input.pages)

if "Remove Metadata" in config_select_manipulation:
    # Remove metadata
    writer.trailer.Info = IndirectPdfDict(
        Title='',
        Author='',
        Subject='',
        Creator='',
    )

if "Concatenate PDFs" not in config_select_manipulation:
    writer.write(output_path+"\Annex "+str(i+1)+".pdf")
    writer = PdfWriter()

status_text.text(f"{file} completed...")
progress_bar.progress(0.25+(0.75/len(filepath))*(i+1))

if "Concatenate PDFs" not in config_select_manipulation:
    status_text.text(f"All done!!!")
    st.balloons()
else:
    # write the modified content to disk
    writer.write(output_path+"\Annex.pdf")

```

```
if config_select_options == "Input files manually":
    for file in filepath_pdf:
        if os.path.exists(file):
            os.remove(file)
        else:
            for file in filepath:
                if os.path.exists(file):
                    os.remove(file)
            st.balloons()

st.write(" -- ")
st.write("Once you have selected the required options above, you can
click on the button below to start processing. ")

if st.button("Click here to start!"):
    main()
```

References

[1] <https://streamlit.io/>

Other Projects

- [Price Comparison using Streamlit and Selenium](#)
- [Tabular data from PDF: Camelot vs Tabula? Why not use both together?](#)
- [PDF Manipulation Application Part 2 — How to remove unwanted pages using PDFminer](#)
- [PDF Manipulation Application Part 3 — Deploying the application on Heroku and output result as zip file](#)

Sign up for CrunchX

By CodeX

A weekly newsletter on what's going on around the tech and programming space [Take a look.](#)

Your email



Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

[Streamlit](#)[Python](#)[Pdf](#)[Web Applications](#)[Python3](#)[About](#) [Write](#) [Help](#) [Legal](#)

Get the Medium app

