Document Uploading Task

(9-10-2024)

Downloading different types of files for document uploading

1. .PDF Extension files (2 Files)
2. .txt Extension files (2 Files)
3. .docx Extension Files (2 Files)
4. .csv Extension Files (2 Files)
5. .xlsx Extension Files (2 Files
6. .PPT Extension Files (2 Files)
7. .jpg, png Extension Files (1 File)

Code Explored:

- Install Necessary Libraries

```
!pip install PyPDF2
!pip install python-docx
!pip install openpyxl
!pip install python-pptx
```

- Import neccesary Libraries

```
import os
import pandas as pd
import PyPDF2
import docx
import csv
import pptx

import openpyxl

from google.colab import files

from PIL import Image

from openpyxl import load_workbook

from pptx import Presentation



# Dictionary to store lists of uploaded files

uploaded_files = {

    'csv': [],

    'pdf': [],

    'docx': [],

    'txt': [],

    'xlsx': [],

    'pptx': [],

    'images': []

}
```

```
# Supported file extensions
supported_extensions = {
    'csv': '.csv',
    'pdf': '.pdf',
    'docx': '.docx',
    'txt': '.txt',
    'xlsx': '.xlsx',
    'pptx': '.pptx',
    'images': ['.jpg', '.jpeg', '.png']
}

# Function to upload CSV files
def upload_csv():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.csv'):
            uploaded_files['csv'].append(filename)

# Function to upload PDF files
def upload_pdf():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.pdf'):
            uploaded_files['pdf'].append(filename)
```

```
# Function to upload DOCX files
def upload_docx():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.docx'):
            uploaded_files['docx'].append(filename)

# Function to upload TXT files
def upload_txt():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.txt'):
            uploaded_files['txt'].append(filename)

# Function to upload XLSX files
def upload_xlsx():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.xlsx'):
            uploaded_files['xlsx'].append(filename)

# Function to upload PPT files
def upload_ppt():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.pptx'):
            uploaded_files['pptx'].append(filename)


#Function to upload image files
def upload_image():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith(('.jpg', '.jpeg', '.png')):
            uploaded_files['images'].append(filename)

# Call the functions to upload files
upload_csv()
upload_pdf()
upload_docx()
upload_txt()
upload_xlsx()
upload_ppt()
upload_image()

#You can access the lists of uploaded files as follows:
print(uploaded_files)
```

10-10-2024

Updated Code for uploading files and checking all the file extensions working correctly or not.

Code:

```
# Dictionary to store lists of uploaded files

uploaded_files = {

    'csv': [],

    'pdf': [],

    'docx': [],

    'txt': [],

    'xlsx': [],

    'pptx': [],

    'images': []

}

# Supported file extensions

supported_extensions = {

    'csv': '.csv',

    'pdf': '.pdf',

    'docx': '.docx',

    'txt': '.txt',

    'xlsx': '.xlsx',

    'pptx': '.pptx',

    'images': ['.jpg', '.jpeg', '.png']

}
```

```
# Function to check if a file is supported
def is_supported(filename):
    for file_type, extensions in supported_extensions.items():
        # Single extension case
        if isinstance(extensions, str) and filename.endswith(extensions):
            return True
        # List of extensions case (for images)
        elif isinstance(extensions, list) and any(filename.endswith(ext) for ext in 
extensions):
            return True
    return False

# Function to upload CSV files
def upload_csv():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.csv'):
            uploaded_files['csv']. append(filename)
        elif not is_supported(filename):
            print (f"File type not supported: {filename}")

# Function to upload PDF files
def upload_pdf():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.pdf'):
            uploaded_files['pdf']. append(filename)
        elif not is_supported(filename):
            print (f"File type not supported: {filename}")

# Function to upload DOCX files
def upload_docx():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.docx'):
            uploaded_files['docx']. append(filename)
        elif not is_supported(filename):
            print (f"File type not supported: {filename}")

# Function to upload TXT files
def upload_txt():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.txt'):
            uploaded_files['txt'].append(filename)
        elif not is_supported(filename):
            print(f"File type not supported: {filename}")

# Function to upload XLSX files
def upload_xlsx():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.xlsx'):
            uploaded_files['xlsx'].append(filename)
        elif not is_supported(filename):
            print(f"File type not supported: {filename}")
```

```
# Function to upload PPT files
def upload_ppt():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith('.pptx'):
            uploaded_files['pptx'].append(filename)
        elif not is_supported(filename):
            print(f"File type not supported: {filename}")

# Function to upload image files
def upload_image():
    uploaded = files.upload()
    for filename in uploaded.keys():
        if filename.endswith(('.jpg', '.jpeg', '.png')):
            uploaded_files['images'].append(filename)
        elif not is_supported(filename):
            print(f"File type not supported: {filename}")

# Call the functions to upload files
upload_csv()
upload_pdf()
upload_docx()
upload_txt()
upload_xlsx()
upload_ppt()
upload_image()
```

```
# You can access the lists of uploaded files as follows:

print(uploaded_files)
```

Outputs:

```
 save_water__poster.jpg(image/jpeg) - 146985 bytes, last modified: 10/9/2024 - 100%
done

 artificial_intelligence.pptx(application/vnd.openxmlformats-
officedocument.presentationml.presentation) - 1189683 bytes, last modified: 10/9/2024
- 100% done

Saving save_water__poster.jpg to save_water__poster.jpg

Saving artificial_intelligence.pptx to artificial_intelligence.pptx

 Chat_with_MultiplePDFs_Mistral_7B_Instruct1.ipynb(n/a) - 279731 bytes, last
modified: 8/19/2024 - 100% done

Saving Chat_with_MultiplePDFs_Mistral_7B_Instruct1.ipynb to
Chat_with_MultiplePDFs_Mistral_7B_Instruct1.ipynb

File type not supported: Chat_with_MultiplePDFs_Mistral_7B_Instruct1.ipynb

 file_example_XLSX_50.xlsx(application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet) - 7360 bytes, last modified: 10/9/2024 - 100% done

 file_example_XLSX_100.xlsx(application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet) - 9299 bytes, last modified: 10/9/2024 - 100% done

Saving file_example_XLSX_50.xlsx to file_example_XLSX_50.xlsx

Saving file_example_XLSX_100.xlsx to file_example_XLSX_100.xlsx

 business-operations-survey-2022-business-finance.csv(text/csv) - 680782 bytes, last
modified: 10/1/2024 - 100% done

 annual-enterprise-survey-2023-financial-year-provisional.csv(text/csv) - 8065547
bytes, last modified: 10/1/2024 - 100% done

Saving business-operations-survey-2022-business-finance.csv to business-operations-survey-
2022-business-finance.csv

Saving annual-enterprise-survey-2023-financial-year-provisional.csv to annual-enterprise-
survey-2023-financial-year-provisional.csv
```

□ **artificial_intelligence.txt**(text/plain) - 92488 bytes, last modified: 10/9/2024 - 100% done

□ **data analytics.txt**(text/plain) - 129447 bytes, last modified: 10/9/2024 - 100% done

```
Saving artificial_intelligence.txt to artificial_intelligence.txt
Saving data analytics.txt to data analytics.txt
```

□ **Four-Steps-to-Forgiveness-William-Fergus-Martin.docx**(application/vnd.openxmlformats-officedocument.wordprocessingml.document) - 277019 bytes, last modified: 10/9/2024 - 100% done

```
Saving Four-Steps-to-Forgiveness-William-Fergus-Martin.docx to Four-Steps-to-Forgiveness-William-Fergus-Martin.docx
```

User Interface Using Gradio

```
! pip install gradio
```

Interface with different files uploading (if i upload files of csv, pdf, docx, txt, xlsx, pptx, images it is working properly, if i upload other than supported file extension it is showing "File type not supported" and showing the uploaded files list)

```
import gradio as gr
import os
# Function to check if a file is supported
def is_supported(filename):
    for file_type, extensions in supported_extensions.items():
        if isinstance(extensions, str) and filename.endswith(extensions):
            return True
        elif isinstance(extensions, list) and any(filename.endswith(ext) for ext in extensions):
            return True
    return False
```

```
# Function to handle file uploads
def upload_files(files):
    for file in files:
        filename = file.name
        if is_supported(filename):
            if filename.endswith('.csv'):
                uploaded_files['csv'].append(filename)
            elif filename.endswith('.pdf'):
                uploaded_files['pdf'].append(filename)
            elif filename.endswith('.docx'):
                uploaded_files['docx'].append(filename)
            elif filename.endswith('.txt'):
                uploaded_files['txt'].append(filename)
            elif filename.endswith('.xlsx'):
                uploaded_files['xlsx'].append(filename)
            elif filename.endswith('.pptx'):
                uploaded_files['pptx'].append(filename)
            elif filename.endswith(('.jpg', '.jpeg', '.png')):
                uploaded_files['images'].append(filename)
        else:
            return f"File type not supported: {filename}"
    return uploaded_files
```

```
# Create Gradio interface
def gradio_interface():
    with gr.Blocks() as demo:
        gr.Markdown("## Uploading Documents")
        file_upload = gr.File(file_count="multiple", file_types=['file'], label="Upload Files")
        upload_button = gr.Button("Upload")
        output = gr.Textbox(label="Uploaded Files")
        upload_button.click(upload_files, inputs=file_upload, outputs=output)
    return demo
if __name__ == "__main__":
    demo = gradio_interface()
    demo.launch()
```

Interface with Delete Button(not interactive)

```
import gradio as gr
# Dictionary to store lists of uploaded files
uploaded_files = {
    'csv': [],
    'pdf': [],
    'docx': [],
    'txt': [],
    'xlsx': [],
    'pptx': [],
    'images': []
}
```

```
# Supported file extensions
supported_extensions = {
    'csv': '.csv',
    'pdf': '.pdf',
    'docx': '.docx',
    'txt': '.txt',
    'xlsx': '.xlsx',
    'pptx': '.pptx',
    'images': ['.jpg', '.jpeg', '.png']
}
# Function to check if a file is supported
def is_supported(filename):
    for file_type, extensions in supported_extensions.items():
        if isinstance(extensions, str) and filename.endswith(extensions):
            return True
        elif isinstance(extensions, list) and any(filename.endswith(ext) for ext in extensions):
            return True
    return False
# Function to handle file uploads
def upload_files(files):
    for file in files:
        filename = file.name
        if is_supported(filename):
            if filename.endswith('.csv'):
                uploaded_files['csv'].append(filename)
            elif filename.endswith('.pdf'):
                uploaded_files['pdf'].append(filename)
            elif filename.endswith('.docx'):
```

```
            uploaded_files['docx'].append(filename)
          elif filename.endswith('.txt'):
            uploaded_files['txt'].append(filename)
          elif filename.endswith('.xlsx'):
            uploaded_files['xlsx'].append(filename)
          elif filename.endswith(('.jpg', '.jpeg', '.png')):
            uploaded_files['images'].append(filename)
        else:
          return f"File type not supported: {filename}"
    return display_files()
# Function to delete a file from the uploaded files list
def delete_file(file_info):
    file_type, filename = file_info.split('|')
    uploaded_files[file_type].remove(filename)
    return display_files()

# Function to display the remaining files
def display_files():
    file_display = ""
    for file_type, files in uploaded_files.items():
        if files:
          file_display += f"**{file_type.upper()} Files:**\n"
          for filename in files:
            file_display += f"{filename} [Delete](delete:{file_type}|{filename})\n"
    return file_display if file_display else "No files uploaded."
# Create Gradio interface
def gradio_interface():
    with gr.Blocks() as demo:
        gr.Markdown("## Uploading Documents")
        file_upload = gr.File(file_count="multiple", file_types=['file'], label="Upload Files")
        upload_button = gr.Button("Upload")
        output = gr.Markdown(label="Uploaded Files")
        # Handling uploads
        upload_button.click(upload_files, inputs=file_upload, outputs=output)
        # Handling deletions
        output.change(delete_file, inputs=output, outputs=output)
    return demo
if __name__ == "__main__":
    demo = gradio_interface()
    demo.launch()
```

Output:

```
DOCXFiles: /tmp/gradio/4b7338530f2794de4477e3ad5dd3a70843144ed50f5e3a1b7252906c
d0785cee/agricultural_techniques_wor.docx.docx Delete

TXTFiles: /tmp/gradio/e94c21b7a616ca8699db8510802e6f9f3f9b29eed0745abc95072320d1
030903/artificial_intelligence.txt Delete

 XLSXFiles: /tmp/gradio/b489bfd976cebe6988cb50d12393f3f8be7efbd5cd79d421f9e5367e0
99d8a3e/file_example_XLSX_50.xlsx Delete

IMAGESFiles: /tmp/gradio/69d4213bd280256696546b888bebcd5506e1fa16d4fb703f4b3b32
3022efce66/save_water__poster.jpg Delete
```

14-10-2024

Interface with Delete Button

Code:

```
import gradio as gr

import os

# Dictionary to store lists of uploaded files

uploaded_files = {

    'csv': [],

    'pdf': [],

    'docx': [],

    'txt': [],

    'xlsx': [],

    'pptx': [],

    'images': []

}

# Supported file extensions

supported_extensions = {

    'csv': '.csv',

    'pdf': '.pdf',

    'docx': '.docx',

    'txt': '.txt',

    'xlsx': '.xlsx',

    'pptx': '.pptx',

    'images': ['.jpg', '.jpeg', '.png']

}
```

```
# Function to check if a file is supported
def is_supported(filename):
    for file_type, extensions in supported_extensions.items():
        if isinstance(extensions, str) and filename.endswith(extensions):
            return True
        elif isinstance(extensions, list) and any(filename.endswith(ext) for ext in extensions):
            return True
    return False
# Function to handle file uploads
def upload_files(files):
    for file in files:
        filename = file.name
        if is_supported(filename):
            if filename.endswith('.csv'):
                uploaded_files['csv'].append(filename)
            elif filename.endswith('.pdf'):
                uploaded_files['pdf'].append(filename)
            elif filename.endswith('.pptx'):
                uploaded_files['pptx'].append(filename)
            elif filename.endswith('.docx'):
                uploaded_files['docx'].append(filename)
            elif filename.endswith('.txt'):
                uploaded_files['txt'].append(filename)
            elif filename.endswith('.xlsx'):
                uploaded_files['xlsx'].append(filename)
            elif filename.endswith(('.jpg', '.jpeg', '.png')):
                uploaded_files['images'].append(filename)
        else:
            return f"File type not supported: {filename}"
    return display_files()


# Function to delete a file from the uploaded files list
def delete_file(file_info):
    file_type, filename = file_info.split('|')
    uploaded_files[file_type].remove(filename)
    return display_files()
# Function to display the remaining files
def display_files():
    file_display = ""
    for file_type, files in uploaded_files.items():
        if files:
            file_display += f"**{file_type.upper()} Files:**\n"
            for filename in files:
                file_display += f"{filename} [Delete](delete:{file_type}|{filename})\n"
    return file_display if file_display else "No files uploaded."
# Function to delete a file from the filesystem
def remove_file_from_system(filename):
    try:
        os.remove(filename)
        return f"File '{filename}' deletion successfully completed!"
    except FileNotFoundError:
        return f"File '{filename}' not found!"
    except Exception as e:
        return f"Error: {str(e)}"
```

```
# Create Gradio interface
def gradio_interface():
    with gr.Blocks() as demo:
        gr.Markdown("## Uploading Documents")
        file_upload = gr.File(file_count="multiple", file_types=['file'], label="Upload Files")
        upload_button = gr.Button("Upload")
        output = gr.Markdown(label="Uploaded Files")
        # Input for deleting a specific file from the filesystem
        delete_input = gr.Textbox(label="Enter filename to delete from system (or type 'quit' to
exit)")
        delete_button = gr.Button("Delete File")
        # Handling uploads
        upload_button.click(upload_files, inputs=file_upload, outputs=output)
        # Handling deletions
        output.change(delete_file, inputs=output, outputs=output)
        # Handling file system deletions
        delete_button.click(lambda filename: remove_file_from_system(filename) if filename
!= 'quit' else "Exiting...", inputs=delete_input, outputs=output)
    return demo
if __name__ == "__main__":
    demo = gradio_interface()
    demo.launch(debug=True)
```

Output:

Uploading Documents

Upload Files

save_water__poster.jpg	143.5 KB ⇣	×
data analytics.pptx	1.4 MB ⇣	×
file_example_XLSX_50.xlsx	7.2 KB ⇣	×

Upload

File 'file_example_XLSX_50.xlsx' deletion successfully completed!

Enter filename to delete from system (or type 'quit' to exit)

Delete File